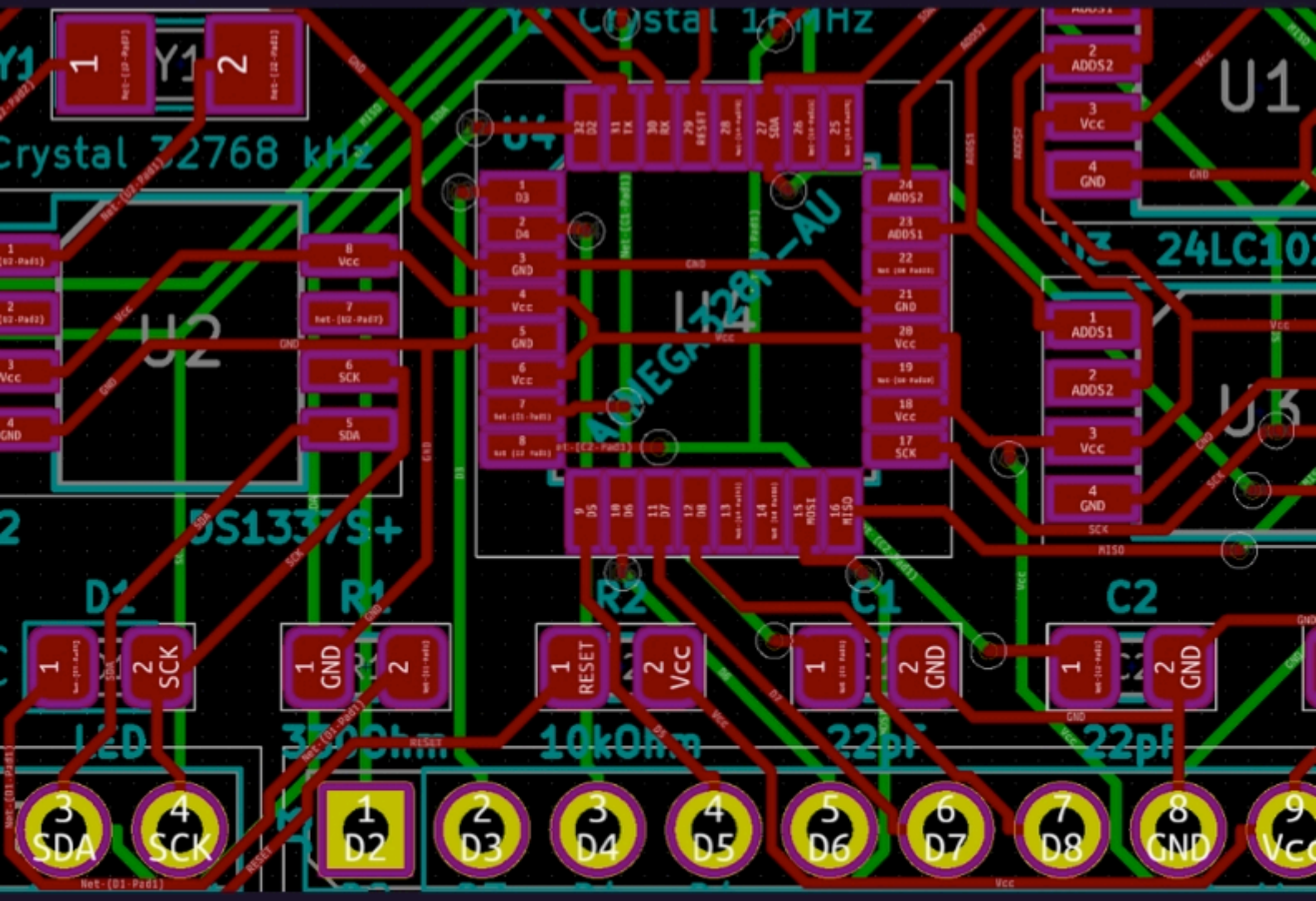
2ND EDITION

KICAD LIKE A PRO

A COMPREHENSIVE HANDS-ON GUIDE FOR LEARNING THE WORLD'S FAVOURITE OPEN SOURCE PRINTED CIRCUIT BOARD DESIGN TOOL

DR PETER DALMARIS



Tech Explorations
txplore.com

KICAD LIKE A PRO

Dr Peter Dalmaris

KiCad Like a Pro, 2nd Edition
By Dr Peter Dalmaris

Printed in Australia

First Printing, 2018

ISBN (PDF) : 978-1-64440-886-5
ISBN (epub): 978-1-64440-885-8
ISBN (mobi): 978-1-64467-528-1

Tech Explorations Publishing
PO Box 22, Berowra 2081 NSW
Australia

www.techexplorations.com

Cover designer: Michelle Dalmaris

Disclaimer

The material in this publication is of the nature of general comment only, and does not represent professional advice. It is not intended to provide specific guidance for particular circumstances and it should not be relied on as the basis for any decision to take action or not take action on any matter which it covers. Readers should obtain professional advice where appropriate, before making any such decision. To the maximum extent permitted by law, the author and publisher disclaim all responsibility and liability to any person, arising directly or indirectly from any person taking or not taking action based on the information in this publication.

Version 1.51

Did you find an error?

Please let us know.
Using any web browser, go to **txplo.re/KiCadbook**, and fill in the form.
We'll get it fixed right away.

About the author

Dr. Peter Dalmaris is an educator, an electrical engineer, electronics hobbyist, and Maker. Creator of online video courses on DIY electronics and author of several technical books. Peter has recently released his book 'Maker Education Revolution', a book about how Making is changing the way we learn and teach in the 21st century.

As a Chief Tech Explorer since 2013 at Tech Explorations, the company he founded in Sydney, Australia, Peter's mission is to explore technology and help educate the world.

Tech Explorations offers educational courses and Bootcamps for electronics hobbyists, STEM students, and STEM teachers.

A lifelong learner, Peter's core skill lies in explaining difficult concepts through video and text. With over 15 years of tertiary teaching experience, Peter has developed a simple yet comprehensive style in teaching that students from all around the world appreciate.

His passion for technology and the world of DIY open-source hardware, has been a dominant driver that has guided his personal development and his work through Tech Explorations.

About Tech Explorations

Tech Explorations creates educational products for students and hobbyists of electronics who rather utilize their time making awesome gadgets instead of searching endlessly through blog posts and Youtube videos.

We deliver high-quality instructional videos and books through our online learning platform, txplore.com.

Supporting our students through their learning journey is our priority, and we do this through our dedicated online community and course forums.

Founded in 2013 by Peter Dalmaris, Tech Explorations was created after Peter realised how difficult it was to find high-quality definitive guides for the Arduino, written or produced by creators who responded to their reader questions.

Peter was frustrated having to search for Youtube videos and blog articles that almost never seemed to be made for the purpose of conveying knowledge.

He decided to create Teach Explorations so that he could produce the educational content that he wished he could find back then.

Tech Explorations courses are designed to be comprehensive, definitive and practical. Whether it is through video, ebook, blog or email, our delivery is personal and conversational.

It is like having a friend showing you something neat... the "AHA" moments just flow!

Peter left his career in Academia after his passion for electronics and making was rekindled with the arrival of his first Arduino. Although he was an electronics hobbyist from a young age, something that led him to study electrical and electronics engineering in University, the Arduino signalled a revolution in the way that electronics is taught and learned.

Peter decided to be a part of this revolution and has never looked back.

We know that even today, with all the information of the world at your fingertips, thanks to Google, and all the components of the world one click away, thanks to eBay, the life of the electronics hobbyist is not easy.

Busy lifestyles leave little time for your hobby, and you want this time to count.

We want to help you to enjoy your hobby. We want you to enjoy learning amazing practical things that you can use to make your own awesome gadgets.

Electronics is a rewarding hobby. Science, engineering, mathematics, art, and curiosity all converge in a tiny circuit with a handful of components.

We want to help you take this journey without delays and frustrations.

Our courses have been used by over 70,000 people across the world.

From prototyping electronics with the Arduino to learning full-stack development with the Raspberry Pi or designing professional-looking printed circuit boards for their awesome gadgets, our students enjoyed taking our courses and improved their making skills dramatically.

Here's what some of them had to say:

"I'm about half way through this course and I am learning so much. Peter is an outstanding instructor. I recommend this course if you really want to learn about the versatility of the amazing Raspberry Pi" -- Scott

"The objectives of this course are uniquely defined and very useful. The instructor explains the material very clearly." -- Huan

"Logical for the beginner. Many things that I did not know so far about Arduino but easy to understand. Also the voice is easy to understand which is unlike many courses about microcontrollers that I have STARTED in the past. Thanks" -- Anthony

Please check out our courses at techexplorations.com and let us be part of your tech adventures.

From the back cover

Printed circuit boards (PCBs) are, perhaps, the most undervalued component of modern electronics. Usually made of fibreglass, PCBs are responsible for holding in place and interconnecting the various components that make virtually all electronic devices work.

The design of complex printed circuit boards was something that only skilled engineers could do. These engineers used expensive computer-aided design tools. The boards they designed were manufactured in exclusive manufacturing facilities in large numbers.

Not anymore.

During the last 20 years, we have seen high-end engineering capabilities becoming available to virtually anyone that wants them. Computer-aided design tools and manufacturing facilities for PCBs are one mouse click away.

KiCad is one of those tools. Perhaps the world's most popular (and best) computer-aided design tool for making printed circuit boards, KiCad is open source, fully featured, well-funded and supported, well documented. It is the perfect tool for electronics engineers and hobbyists alike, used to create amazing PCBs. KiCad has reached maturity and is now a fully featured and stable choice for anyone that needs to design custom PCBs.

This book will teach you to use KiCad. Whether you are a hobbyist or an electronics engineer, this book will help you become productive quickly, and start designing your own boards.

Are you a hobbyist? Is the breadboard a bottleneck in your projects? Do you want to become skilled in circuit board design? If yes, then KiCad and this book are a perfect choice. Use KiCad to design custom boards for your projects. Don't leave your projects on the breadboard, gathering dust and falling apart.

Complete your prototyping process with a beautiful PCB and give your projects a high-quality, professional look.

Are you an electronics engineer? Perhaps you already use a CAD tool for PCB design. Are you interested in learning KiCad and experience the power and freedom of open-source software? If yes, then this book will help you become productive with KiCad very quickly. You can build on your existing PCB design knowledge and learn KiCad through hands-on projects.

This book takes a practical approach to learning. It consists of four projects of incremental difficulty and recipes.

The projects will teach you basic and advanced features of KiCad. If you have absolutely no prior knowledge of PCB design, you will find that the introductory project will teach you the very basics. You can then continue with the rest of the projects. You will design a board for a breadboard power supply, a tiny Raspberry Pi HAT, and an Arduino clone with extended memory and clock integrated circuits.

The book includes a variety of recipes for frequently used activities. You can use this part as a quick reference at any time.

The book is supported by the author via a page that provides access to additional resources. Signup to receive assistance and updates.

How to read this book

I designed this book to be used both to learn how to use KiCad, and as a reference.

All examples, descriptions and procedures are tested on the production release of KiCad 5.0.

In January 2020, I updated several parts of this book to better reflect the newer version of KiCad, version 5.1, that was available at the time. In Part 2 of this book, I have added a new chapter that summarises the most important differences between KiCad 5.0 and 5.1.

If you have never used KiCad and have little or no experience in PCB design, I recommend you read it in a linear fashion. Don't skip the early chapters in parts 1, 2 and 3, because those will set the fundamental knowledge on which you will build your skill later in the book. If you skip those chapters, you will have gaps in your knowledge that will make it harder for you to progress.

If you have a good working knowledge of PCB design, but you are new to KiCad, you can go straight to Part 2, zoom through it very quickly, and then proceed to the projects in Part 4.

Once you have the basic KiCad concepts and skills confidently learned, you can use the recipes in Part 5 as a resource for specific problems you need solved. These recipes are useful on their own. Throughout the text, you will also find prompts to go to a particular recipe in order to learn a specific skill needed for the projects.

Images: Throughout this book, you will find numerous figures that contain screenshots of KiCad. To create these screenshots, I used KiCad 5.0.0 running on Linux Ubuntu (I have updated some chapters with screenshots and descriptions from KiCad 5.1.5). If you are using KiCad under Windows or Mac OS, do not worry: KiCad works the same across these platforms, and even looks almost the same.

Although I took care to produce images that are clear, there are cases where this was not possible. This is particularly true in screenshots of an entire application window, meant to be displayed in a large screen. The role of these images is to help you follow the instructions in the book as you are working on your computer. There is no substitute to experimenting and learning by

doing, so the best advice I can give is to use this book as a text book and companion. Whenever you read it, have KiCad open on your computer and follow along with the instructions.

This book has a web page with resources designed to maximise the value it delivers to you, the reader. Please read about the book web page, what it offers and how to access it in the section 'The book web page', later in this introductory segment.

Finally, you may be interested in the video course version of this book. This course spans over 17 hours of high-definition video, with detailed explanations and demonstrations of all projects featured in the book. The video lectures capture techniques and procedures that are just not possible to do so in text.

Please check in the book web page for updates on this project. Be sure to subscribe to the Tech Explorations email list so I can send you updates.

Requirements

To make the most out of this book, you will need a few things. You probably already have them:

- A computer running Windows, Mac OS or Linux.
- Access to the Internet.
- A mouse with at least two buttons and a scroll wheel. I use a Logitech MX Master 2S mouse (see https://amzn.to/2ClySq0).
- Ability to install software.
- Time to work on the book, and patience.

The book web page

As a reader of this book, you are entitled access to its online resources.

You can access these resources by visiting the book's web page at *http://txplo.re/klpr*.

The available resources are:

1. **A link to the book forum**. The forum is a place where you can ask book-related questions and have a conversation about your projects. I spend time in the forum weekly, answering questions and participating in discussions.
2. **Errata and error report form**. As I correct bugs, I post information about these corrections in this page. Please check this page if you suspect that you have found an error. If you have found an error that is not listed in the errata page, please use the error report form in the same page to let me know about it.
3. **Downloadable resources**. Download all photos, screenshots and illustrations from the book in this Google Photos album, as well as the illustrations and schematic files from the book. All of these resources in high-definition.

From time to time, I will be posting additional KiCad resources and updates on the Tech Explorations website. These updates will be available through the Blog (https://techexplorations.com/blog/) or the KiCad guide pages (txplo.re/829e9). By subscribing to the Tech Explorations email list, you'll be sure to receive my regular book updates and news. The subscription form is here: https://techexplorations.com/subscribe/.

Table of Contents

An introduction: Why KiCad?

Since KiCad first appeared in the PCB CAD world in 1992, it has gone through 5 major versions, and evolved into a serious alternative to commercial products. Once thought clunky and barely usable, it is now a solid, reliable CAD application. While it is true that KiCad is still behind its commercial competitors in specific areas,[1] I believe that the benefits we get from truly free ('free'' as 'free and open source[2]') software are worth the trade-off in polish and finish.

One of those benefits is KiCad's very active and growing community of users and contributors. KiCad has a dedicated developer team, supported by contributing organisations such as CERN, the Raspberry Pi Foundation, Arduino LLC, and Digi-Key Electronics. The community is also active in contributing funds to cover development costs. A fund-raising campaign covered the requested amount by 160%, ensuring 600 hours of development towards KiCad version 6. These alone, to a large extent, guarantee that KiCad's development will accelerate, and will continue to in the future.

Next to the core team, are the people that make the KiCad community. These people support the KiCad project in various ways: writing code, sharing libraries, helping others learn. Over the last five years, I have seen an explosion of interest in KiCad. As a consequence, the Internet is flooded with relevant resources: guides, tutorials, libraries, scripts. Manufacturers have also taken notice. Many of them now publish tutorials, explaining how to order your boards. Some have even made it possible to do so by uploading a single file from your KiCad project instead of having to generate multiple Gerber files, making you prone to make simple errors because of the multiple export options of this process.

Why do I use KiCad? I'm glad you asked. First, let's look at my background. I am an electrical engineer with a background in electronics and

---

[1] An example is KiCad's library manager, traditionally the topic of complaints and the reason of a lot of headaches. With version 5, however, library management is far better than it used to be.

Another area when KiCad is still behind some of the alternatives is in collaborative tools and workflows for teams. With source control systems like Github, however, KiCad collaborative projects are possible.

[2] Learn about Free and Open Source Software: https://en.wikipedia.org/wiki/Free_and_open-source_software

computer engineering. Above all, I am an educator and electronics hobbyist. The majority of my PCB projects eventually find themselves in my books and courses. My projects are very similar to those of other hobbyists, in terms of complexity and size. I make things for my Arduino and Raspberry Pi courses. It could be an Arduino clone, or shield, a Raspberry Pi HAT, or a stand-alone relay board, power supply or motor controller. Nothing I would brag about. As a hobbyist, KiCad proved to be the perfect tool to me. But I do plan to design bigger and better boards.

This is why I decided not to use another excellent tool, Fritzing. In KiCad, I saw a lot of benefits, without any show-stopping problems. I will list and briefly discuss my top 10 KiCad benefits here.

Benefit 1: KiCad is open source. To me, this is very important, especially as I find myself spending more time creating new and more complicated boards. Open source, by definition, means that the code base of the application is available for anyone to download and compile on their computer. It is why technologies such as Linux, Apache, and Wordpress essentially run the Internet (all of them open source). While I am not extreme in my choices between open source and closed source software, whenever a no-brainer open source option does appear, like KiCad, I take it.

Benefit 2: It is free! This is particularly important for hobbyists. CAD tools can be expensive. Without a revenue resulting from the hobby capable of supporting the licensing fees, it is hard to justify hundreds of dollars spent, especially when there are viable alternatives. Which brings me to Benefit 3…

Benefit 3: KiCad is unlimited. There are no 'standard', 'premium' and 'platinum' versions to choose from. It's just a download, and you get everything. While there are many free commercial PCB tools, there are always restrictions on things like how many layers and how big your board can be, what can you do with your board once you have it, who can manufacture your board, and much more. I'll say again: KiCad is unlimited! This is so important, that I choose to pay a yearly donation to CERN that is higher than the cost of an Autodesk Eagle license to do my part in helping to maintain this.

Benefit 4: KiCad has awesome features. Features such as interactive routing, length matching, and differential routing, are professional-grade. While you may not need to use some of them right away, you will use them

eventually. Features that are not included 'in the box' can be added through third-party add-ons, one of the benefits of open source. The autorouter is one example. The ability to automate workflows and extend capabilities through Python scripts is another.

Benefit 5: KiCad is continually improved. Especially since CERN & Society Foundation became involved in their current capacity, I have seen a very aggressive and successfully implemented roadmap. At the time of writing this, KiCad 5 is about one month old (it was released in early August, 2018). The funding for KiCad 6 is complete, and the road map living document published. When I look at this roadmap, I get very excited: an improved and modernised user interface, improvement in the schematic editor and the electrical rules checker (hopefully, with better error messages), better net highlighting, and much more, are in the works right now.

Benefit 6: KiCad's clear separation of schematic and layout is a bonus to learning and using it. Users of other PCB applications often find this confusing, but I really believe that it is an advantage. Schematic design and layout design are truly two different things. You can use them independently. I often create schematic diagrams for my courses that I have no intention in converting into PCBs. I also often create multiple versions of a board, using the same schematic. This separation of roles makes both scenarios easy.

Benefit 7: I can make my boards anywhere: I can upload my project to any online fabricator that accepts the industry-standard Gerber files; I can upload it to an increasing number of fabricators that accept the native KiCad layout file; and, of course, I can make them at home using an etching kit (I do not cover this option in this book).

Benefit 8: KiCad works anywhere. Whether you are a Mac, Windows or Linux person, you can use KiCad. I actually use it on all three platforms.

Benefit 9: KiCad is very configurable. You can assign your favourite keyboard hotkeys and mapping, and together with the mouse customisations, you can fully adapt it to you preferences.

Benefit 10: If you are interested in creating analog circuits, you will be happy to know that KiCad now has integration with Spice. You can draw the schematic in Eeschema, and then simulate it in SPICE, without leaving KiCad. When I need to simulate an analog circuit, I normally use iCircuit, an excellent

desktop app. But I do plan to start using KiCad and Spice for this kind of work.

These are the ten most important reasons for which I have chosen KiCad as my tool of choice for designing PCBs. These reasons might not be right for you, but I hope that you will consider reading this book first before you make your own decision.

In this book, I have packed almost everything I have learnt as a KiCad user over the last four years. I have organised it in a way that will make learning KiCad quick. The objective of this book is to make you productive by the time you complete the first project, in part 4.

If you come from another PCB CAD tool and are already experienced in designing PCBs, I only ask that you have an open mind. KiCad is most certainly very different to the tool that you are used to. It looks different, and it behaves differently. It will be easier to learn it if you consciously put aside your expectations, and look at KiCad like a beginner would. As per the Borg in Star Trek, 'resistance is futile', and in learning, like in so many other aspects of life, you are better off if you just go with the flow.

Let’s begin!

Part 1: A quick introduction to PCB design

1. What is a PCB?

As a child, I remember that my interest in electronics grew from admiration of what these smart engineers had come up with, to curiosity about how these things worked. This curiosity led me to use an old screwdriver that my dad had left in a drawer (probably after fixing the hinges on a door), to open anything electronic with a screw large enough for the screwdriver to fit in.

A record player, a VCR, a radio. All became my victims. I am still amazed that a charged capacitor didn't electrocute me. At least, I had the good sense to unplug the appliances from the mains. Inside those devices, I found all sorts of amazing things. Resistors, transformers, integrated circuits, displays.

All of those things were fitted on small boards, like the one in Figure 1.1. This is an example of a printed circuit board, or PCB, for short.

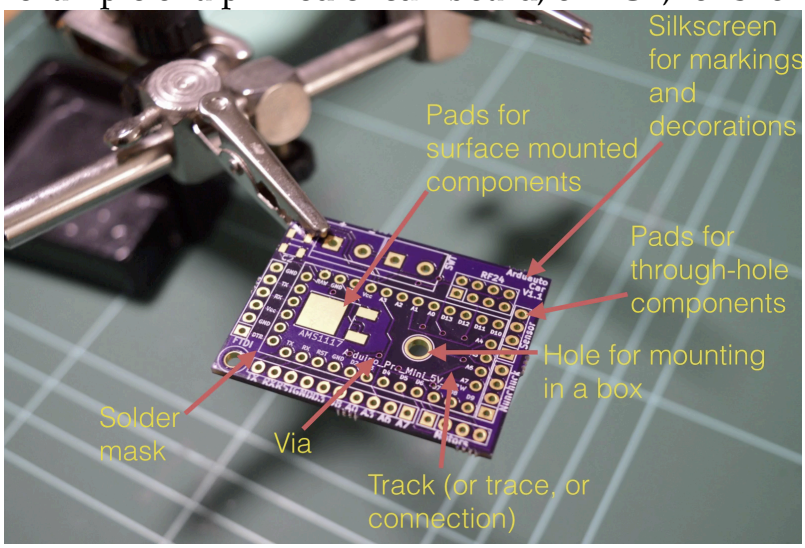


Figure 1.1: The top side of a printed circuit board.

Let's have a look at the components of a PCB, what a PCB looks like and the terminology that we use. The example PCB is one I made for one of my courses (Figure 1.1).

The top side of the PCB is the side where we place the components. We can place components on the bottom side too; however this is unusual.

In general, there are two kinds of components: through hole or surface mounted components. Through hole components, are attached on the PCB via inserting the leads or the pins through small holes, and using hot solder to hold them in place. In the example pictured in Figure 1.1, you can see several

holes into which you can insert the through-hole component pins. The holes extend from the top side to the bottom side of the PCB, and plated with a conductive material, such as tin, or in this case, gold. We use solder to attach and secure a component through its lead onto the pad that is surrounding the hole (Figure 1.2).

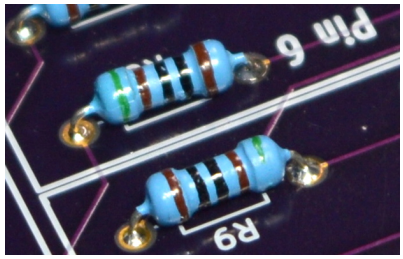


Figure 1.2: A through-hole component attached to a PCB.

If you wish to attach a surface-mounted component, then instead of holes, you attach the component onto the surface of the PCB using tin-plated pads. You will use just enough solder to create a solid connection between the flat connector of the component and the flat pad on the PCB (Figure 1.3).

Figure 1.3: A surface-mounted component attached to a PCB.

Next, is the silkscreen. We use the silkscreen for adding text and graphics. The text can provide useful information about the board and its components. The graphics can include logos, other decorations, and useful markings.

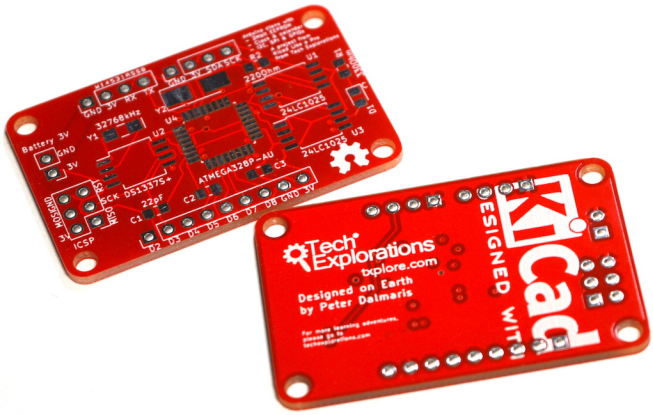


Figure 1.4: The white letters and lines is the silkscreen print on this PCB.

In Figure 1.4, you can see here that I've used white boxes to indicate the location of various components. I've used text to indicate the names of the various pins, and I've got version numbers up there. It's a good habit to have a name for the PCB and things of that sort. Silkscreen goes on the top or the bottom of the PCB.

Sometimes, you may want to secure your PCB onto a surface. To do that, you can add a mounting hole. Mounting holes are similar to the other holes in this board, except that they don't need to be tinned. You can use a screw and a nut and bolt to the other side so that the PCB is secured inside, for example, a box.

Next are the tracks. In this example (Figure 1.5), they look red because of the color of the masking chemical used by the manufacturer.

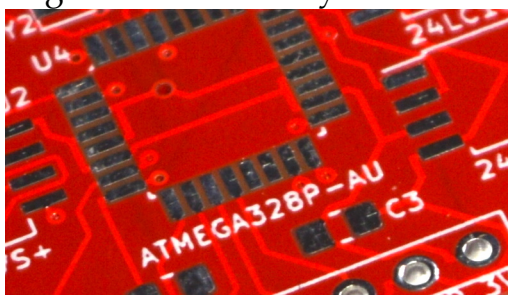


Figure 1.5: The bright red lines connecting the holes are tracks.

Tracks are made of copper, and they electrically connect pins or different parts of the board. You can control the thickness of a track in your design. Tracks can also be referred to as 'traces'.

Notice the small holes that have no pad around them? These are called 'vias.' A via looks like a hole but is not meant to be used to mount a component on it. A via is used to allow a track to continue its route to a different layer. If you're using PCBs that have two or more layers, then you can use vias to connect a track from any one of the layers to any of the other layers. Vias are very useful for routing your tracks around the PCB.

The red substance that you see on the PCB is the solder mask. It does a couple of things. It prevents the copper on the PCB from being oxidised over time. The oxidisation of the copper tracks negatively affects their conductivity. The solder mask prevents oxidisation.

Another thing that the solder mask does is to make it easier to solder by hand. Because pads can be very close to each other, soldering would be very difficult without the solder mask. The solder mask prevents hot solder from creating bridges between pads because it prevents it from sticking on the board. (Figure 1.6). The solder mask prevents bridges because solder cannot bond with it.

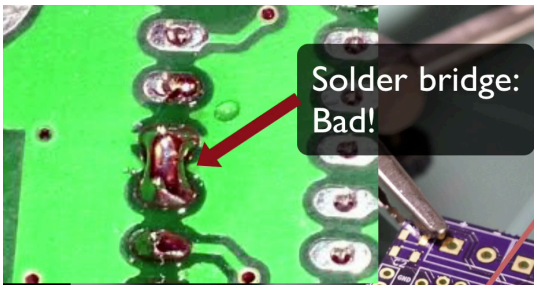


Figure 1.6: A solder bridge like this one is a defect that solder mask helps in preventing.

Often, the tip of the solder, the soldering iron is almost as big or sometimes as bigger than the width of the pads, so creating bridges in those circumstances is very easy and solder mask helps in preventing that from happening.

In Figure 1.7 you can see an example of the standard 1.6mm thick PCB.

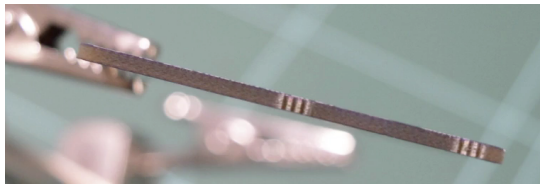


Figure 1.7: This PCB has a thickness of 1.6mm, and is made of fiberglass.

Typically, PCBs are made of fiberglass. The typical thickness of the PCB is 1.6 millimeters. In this close-up view of a PCB picture (Figure 1.8), you can see the holes for the through-hole components. The holes for the through hole components are the larger ones along the edge of the PCB. Notice that they are tined in the inside, electrically connecting the front and back.

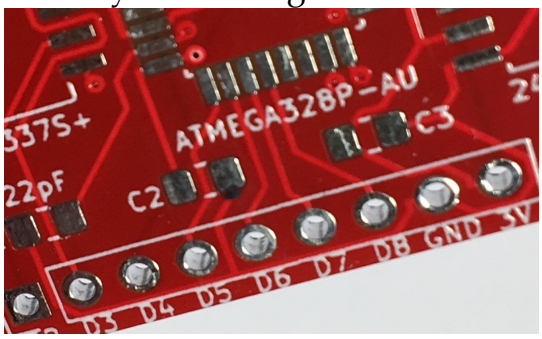


Figure 1.8: A closeup view of the top-layer.

In Figure 1.8 you can see several vias (the small holes) and tracks, the red solder mask, and the solder mask between the pads. In this close up, you can also see the detail of the silkscreens. The white ink is what you use in the silkscreen to create the text and graphics.

Figure 1.9 is interesting because it shows you a way to connect grounds and VCC pads to large areas of copper which is called the copper fill.

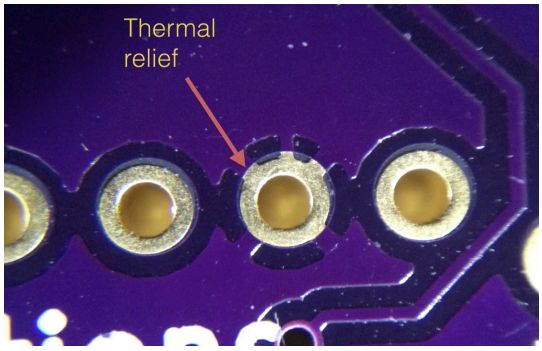


Figure 1.9: Thermal relief connects a pad to a copper region.

In Figure 1.9, the arrow points to a short segment of copper that connects the pad to a large area of copper around it. We refer to this short segment of copper as a 'thermal relief.' Thermal reliefs makes it easier to solder because the soldering heat won't be dissipated into the large copper area.

Figure 1.10 gives a different perspective that allows to appreciate the thickness of the tracks.

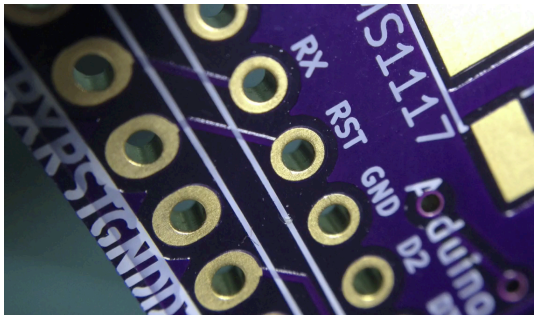


Figure 1.10: The plating of the holes covers the inside of the hole, and connects that front end with the back end.

Notice the short track that connects the two reset holes (RST)? The light that reflects off the side of the track gives you an idea of the thickness of that copper which is covered by the purple solder mask.

In this picture, you can also see a very thin layer of gold that covers the hole and the pad and how that also fills the inside of the hole. This is how you electrically have both sides of the hole connected.

Instead of gold plating, you can also use tin plating in order to reduce the manufacturing costs.

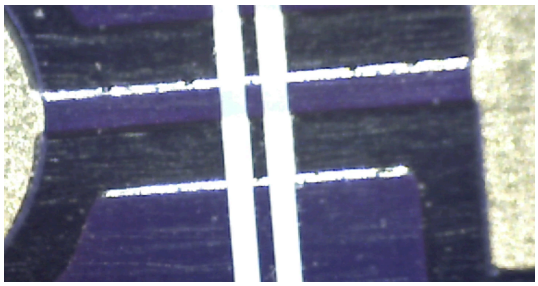


Figure 1.11: A details of this example board at 200 times magnification.

The image in Figure 1.11 is at 200 times magnification. You can see a track that connects two pads, and the light that reflects off one side of the track.

2. The PCB design process

To design a printed circuit board, you have to complete several steps, make decisions, and iterate until you are satisfied with the result. Because you are creating something that must be able to perform a specific operation (or multiple operations) well, your design work must be of high quality, safe, and manufacturable. There is no point designing a PCB that can't be manufactured.

Apart from the practical considerations of designing a PCB, there are also the aesthetic ones. You want your work to look good, not just to function well. Designing a PCB, apart from being an engineering discipline, is also a form of art.

In this book, you will learn about the technical elements of designing a PCB in KiCad, but I am sure that as you start creating your PCBs, your artistic side will emerge. Over time, your PCB will start to look uniquely yours.

PCB design is concerned with the process of creating the plans for a printed circuit board. It is different from PCB manufacturing. In PCB design, you learn about the tools, process, and guidelines useful for creating such plans. In PCB manufacturing, on the other hand, you are concerned about the process of converting the plans of a PCB into the actual PCB.

As a designer of printed circuit boards, it is useful to know a few things about PCB manufacturing, though you surely do not need to be an expert. You need to know about the capabilities of a PCB manufacturing facility so that you can ensure that your design does not exceed those capabilities and that your PCBs are manufacturable. PCB manufacturers will supply "design rules" which specify the dimensions and tolerances they can meet. For example, they may require that all traces be a minimum of 5 thousandths of an inch wide.

As a designer, you need to have an understanding of the design process, and the design tools. To want to design PCB, I assume that you already have a working knowledge of electronics. Designing a PCB, like much else in engineering, is a procedural and iterative process that contains a significant element of personal choice. As you build up your experience and skills, you will develop your unique designing style and process.

While a personalised design process contains unique elements, it still follows the generic process you will learn about in this book. I distilled this process by drawing from my own experience and learning from other

people's best practices. I also tried to simplify this process and make it suitable for people new to PCB design.

Since, in this book, you will be using KiCad, I have created a schematic that shows the PCB design process using KiCad terms and tools. You can see it in Figure 2.1.

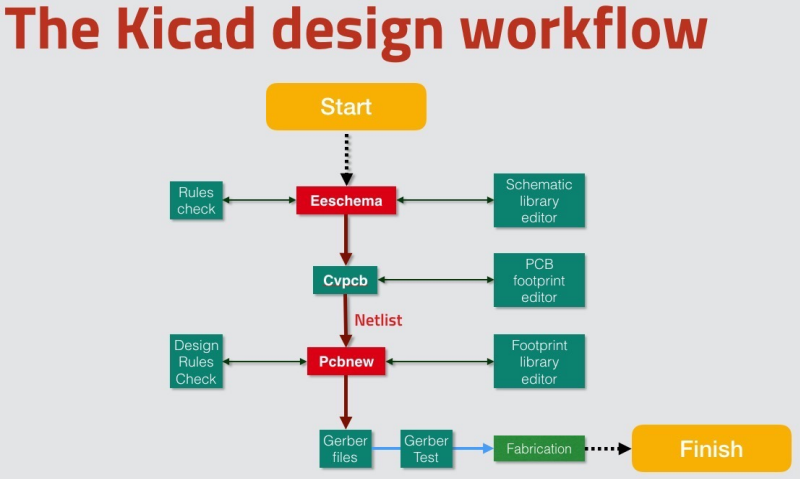


Figure 2.1: The KiCad design workflow.

From a very high-level perspective, the PCB design process only has two major steps:

1. it starts with a schematic diagram;
2. it ends with the layout.

The goal of the design is to create the layout. The layout is a file that contains information about your board, which the manufacturer can use to create the board. The layout must contain information about the size and shape of the board; its construction (such as how many layers it must have); the location of the components on the board, the location of various board elements, like pads, holes, traces and cutouts; the features of these elements (such as the sizes of holes and traces); and much more (which you will learn in detail later in this book).

When we bring this generic process to KiCad, we can map it to the elements that you see in Figure 2.1.

The process begins with Eeschema. In Eeschema you create the electrical schematic that describes the circuit that eventually will be manufactured into the PCB. You draw the schematic by selecting symbols from the library and adding them to the schematic sheet. If a component that you need doesn't exist in the library, you can create it using the schematic library editor.

Running regular electrical rules checks helps to detect defects early. Eeschema has a build-in checker utility for this purpose, as has Pcbnew, the

layout editor. These utilities help to produce PCBs that have a low risk to contain design or electrical defects.

There are two things that you need to do before we go to Pcbnew:

1. associate the components in Eeschema with footprints;
2. create the netlist file, which contains information that Pcbnew needs to set up the layout sheet. The netlist file is what connects Eeschema and Pcbnew.

A symbol is a graphical representation of a real component in the schematic; it does not have a physical counterpart. However, in the layout editor (Pcbnew), everything is real in the sense that it has a real-world counterpart. Therefore, you, as the designer, must associate the symbol with a footprint. The footprint is a real thing; it represents resistors, switches, and pads on the PCB. This allows us to match schematic components with footprint modules.[3]

Once you have created the associations between symbols and modules, you will then export the Netlist file from Eeschema. Then, you'll import the netlist into Pcbnew, and all the footprints that you associated in Eeschema will appear in a new sheet so you can start working on the layout.

You use Pcbnew to position the footprints on the sheet and wire them. Wiring can be very time-consuming, especially for large boards. It is possible to use tools that do the wiring automatically, a capability that can significantly reduce the layout time.

Once you have your PCB laid out and have its traces completed, you can go ahead and do the design rules check. This check looks for defects in the board, such as a trace that is too close to a pad or two footprints overlapping.

Note that different PCB manufacturers may have different capabilities, and/or they may offer enhanced capabilities for a higher price. If you are designing a complex board with densely packed components, you may want to shop around to find a manufacturer with suitable design rules for your needs.

When you are finished working on the layout, you can continue with the last step which involves exporting the layout information in a format that is compatible with your board manufacturer's requirement. The industry standard for this is a format called 'Gerber.' Gerber files contain several related files, with one Gerber file per layer on your PCB, and contain instructions that the fabrication house needs to manufacture your PCB.

Let's move on to the next chapter where we'll talk about fabrication.

---

[3] In KiCad, a schematic contains symbols, and a layout contains footprints.

3. Fabrication

Imagine that you have finished with laying out your board in KiCad and you're ready to make it. What are your options? One option is to make your PCBs at home. There's a guide available on the Fritzing website, at:

http://fritzing.org/learning/tutorials/pcb-production-tutorials/diy-pcb-etching/.

The process described in the Fritzing guide is called etching. It involves the use of various chemicals, in chemical baths. Some of these chemicals are toxic. You have to have special safety equipment, and keep your children and pets away. The process emits smelly and potentially dangerous fumes. Once you have your board etched, you still need to use a drill to make holes and vias, and then figure out a way to connect your top and bottom layers.

If this sounds like not your kind of thing (I'm with you!), then you can opt for a professional PCB manufacturer service. PCBWay, OSH Park and other manufacturers like ExpressPCB and Seeed Studio are very good at what they offer.

You can get a professionally made PCB around $15 for several copies, and without danger to yourself as well. I've used OSHPark (great for beginners thanks to its straightforward user interface) and PCBWay (great for more advanced projects that need a large array of manufacturing options) extensively. I'm always happy with the result. Using an online manufacturer does take a little bit of planning because once you order your PCBs it can take up to several weeks for them to be delivered. If you're in a hurry there are options to expedite the process if you are willing to pay a premium.

The typical small standard two-layer order costs around $10 for a two square inch board; you get three copies of that. This price works out to around $5 per square inch. The pricing is consistent in the industry, where the main cost factor is the size of the PCB. There is a strong incentive to make your PCBs as small as possible. Be aware of this when you design your layout.

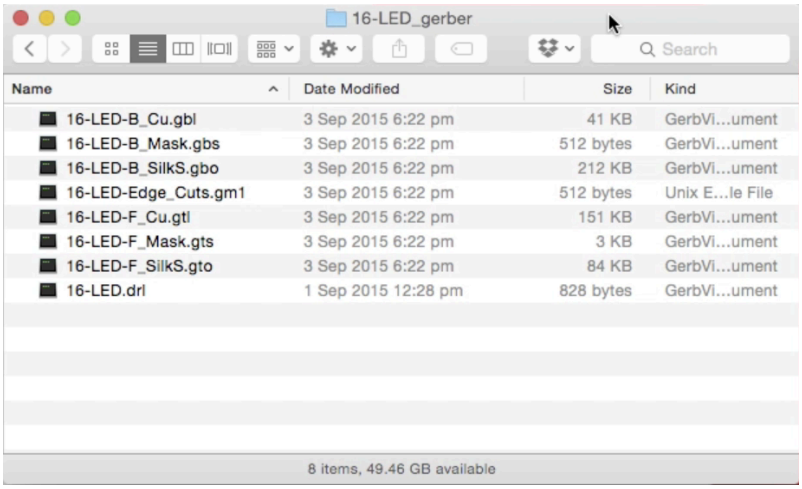


Figure 3.1: An example of the Gerber files that the manufacturer will need in order to make your PCB.

Now, let's turn our attention to the files that you need to upload for these services — and the files are Gerber files. Each layer on your PCB has its own Gerber file which is simply a text file. Figure 3.2 shows the contents of an example Gerber file.

```
1  G04 #@! TF.FileFunction,Soldermask,Bot*
2  %FSLAX46Y46*%
3  G04 Gerber Fmt 4.6, Leading zero omitted, Abs format (unit mm)*
4  G04 Created by KiCad (PCBNEW (2015-07-06 BZR 5891, Git 351914d)-product) date
   03/09/2015 18:22:08*
5  %MOMM*%
6  G01*
7  G04 APERTURE LIST*
8  %ADD10C,0.100000*%
9  %ADD11R,1.727200X2.032000*%
10 %ADD12O,1.727200X2.032000*%
11 G04 APERTURE END LIST*
12 D10*
13 D11*
14 X122555000Y-42545000D03*
15 D12*
16 X120015000Y-42545000D03*
17 X117475000Y-42545000D03*
18 X114935000Y-42545000D03*
19 X112395000Y-42545000D03*
20 M02*
21
```

Figure 3.2: Gerber files contain text

You can see that this is just a text-based file that contains instructions. An advantage of this text format is that you can use a version control system like Git and keep your projects stored in repositories like Github.com.

The Gerber files system and standard has been designed by Ucamco. They make equipment and write software for PCB manufacturers — things like a PreCAM software, PCB CAM, laser photoplotters and direct imaging systems. If you're curious about how to read these Gerber files then you can look up the specification of the Gerber format specification on Ucamco's web site. Beware, it's a huge file.

4. Installation

You can install KiCad on Windows, Mac OS and several flavours of Linux using operating system-specific installers. Its source code is also available, so you can download it and compile it yourself. You can find the version of the installer for your OS at the KiCad download page: http://KiCad-pcb.org/download.

I suggest you install the stable version of KiCad unless you feel compelled to use the cutting-edge releases which contain the latest features (and bugs). In this case, you can download the latest nightly build. You can find information about these builds in the KiCad's download page for your operating system (Windows and OS X). If you are using Linux, you can install KiCad from the command line using tools like apt-get (Debian, Ubuntu) and dnf (Fedora).

You can find detailed instructions on how to do the installation on the KiCad web site.

Please install your copy now before you continue with the next chapter. I also recommend that you install the demo projects because they provide multiple examples of design best practices. I have learned a lot about KiCad by browsing and studying these examples.

In Ubuntu, you can do this by running this command:

```
$ sudo apt install KiCad-demo
```

The demos are installed in /usr/share/KiCad/demos, from where you can copy them in your working folder (Figure 4.1).

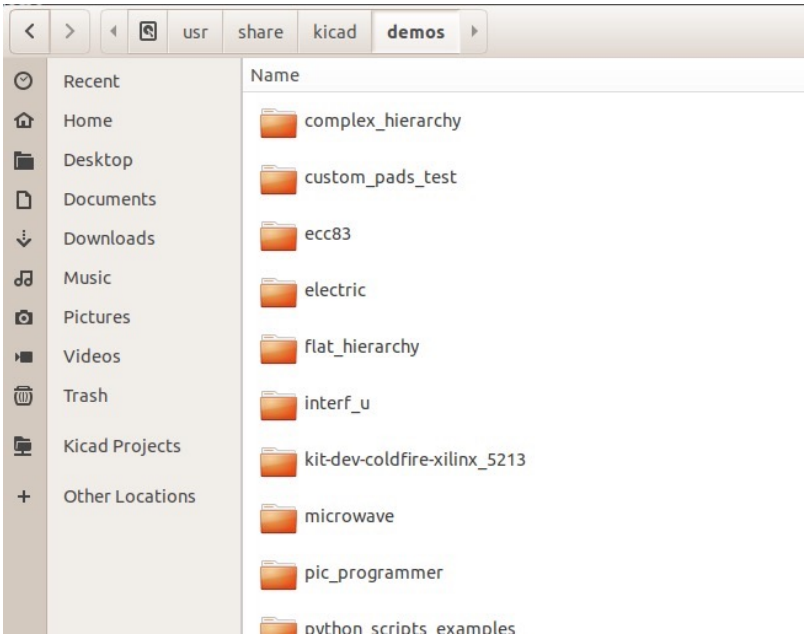


Figure 4.1: The KiCad 'demos' directory in Ubuntu.

In Mac OS, the demos are packaged with the installer. The installation package looks like the example in Figure 4.2.

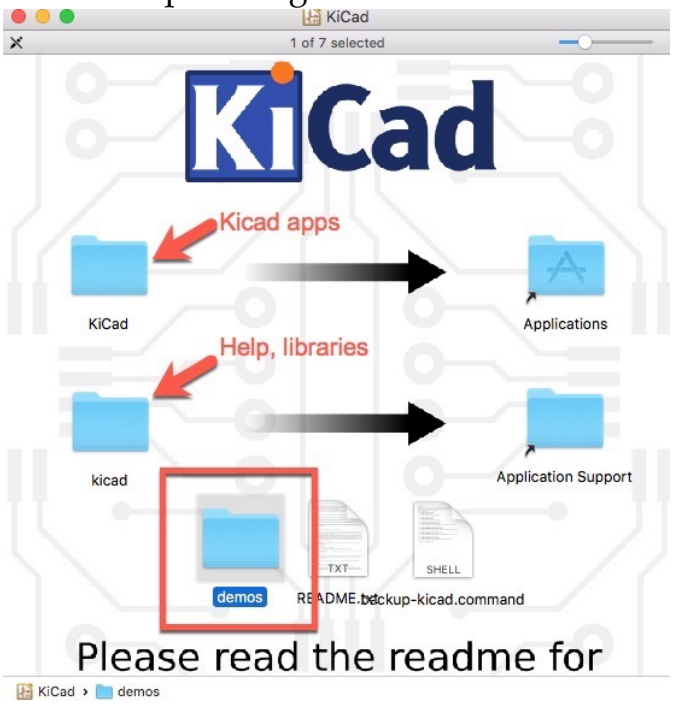


Figure 4.2: The Mac OS installer contains the demo projects in a seperate folder.

Copy the 'demos' folder in your Documents directory, and the other two folders as instructed ('KiCad' to Applications, and 'KiCad' to Application Support). You can see the contents of this folded in Figure 4.3. The original name of this folder is 'demos,' but I have renamed it to 'KiCad demos' to make it easier to find among the other contents of my Documents folder.

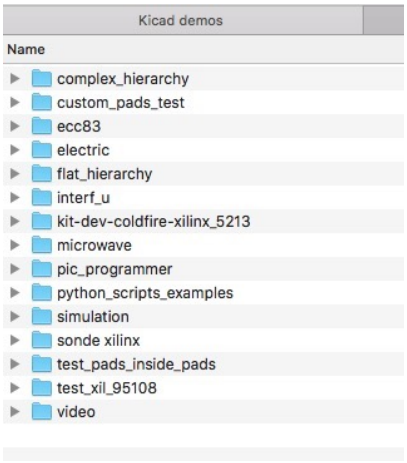


Figure 4.3: The demo folder in my ~/Documents/KiCad demos folder (renamed from 'demos').

In the Windows installer, the demos are available as an option (Figure 4.4).

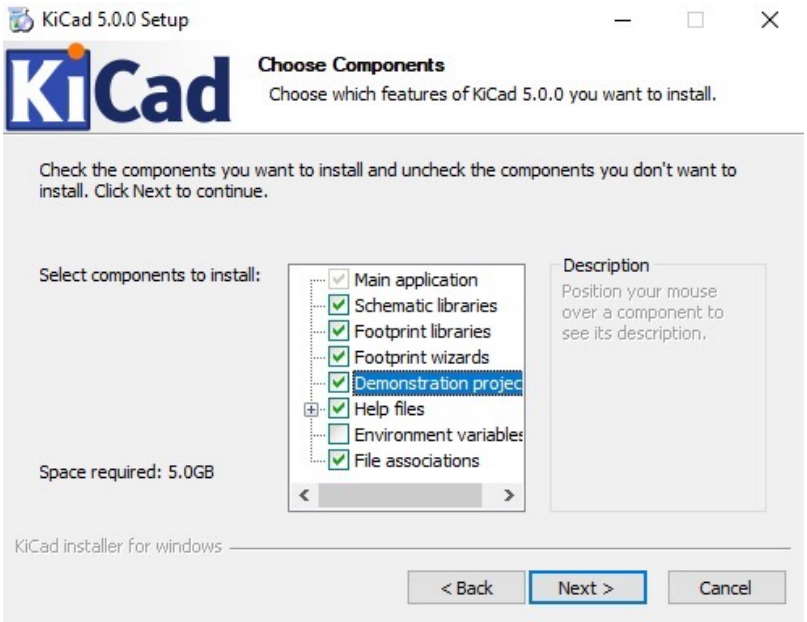


Figure 4.4: The Windows installer contains the demo projects as an optional component.

The default location of the KiCad demos in Windows is at C:\Program Files\KiCad\share\kicad\demos.

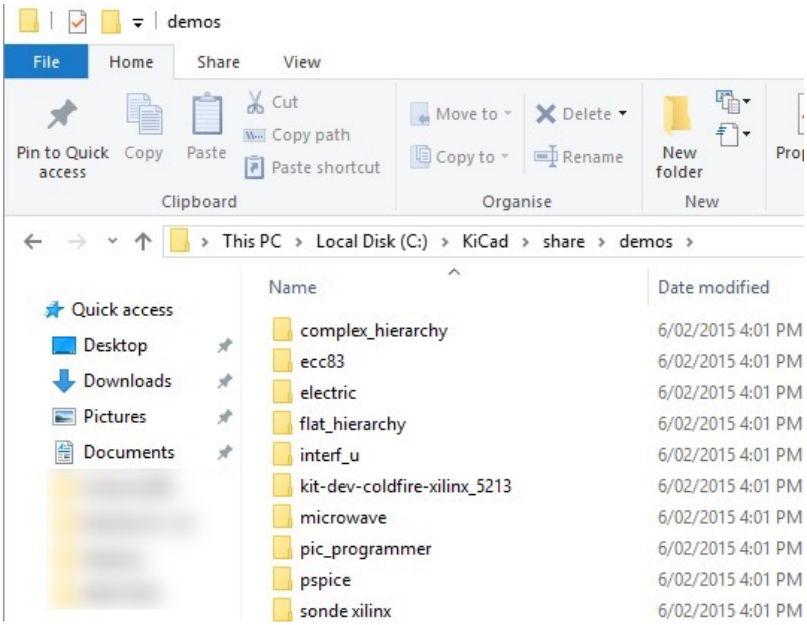


Figure 4.5: The default demo directory in Windows.

With KiCad and its demos installed, you can continue with the next chapter, where you will take a look at one of the demo projects.

5. Examples of KiCad projects

Now that you have installed your instance of KiCad let's start your familiarisation with it by looking at one of the examples that come with it. Browse to the KiCad demos folder, and access the one titled 'pic_programmer' (Figure 5.1).

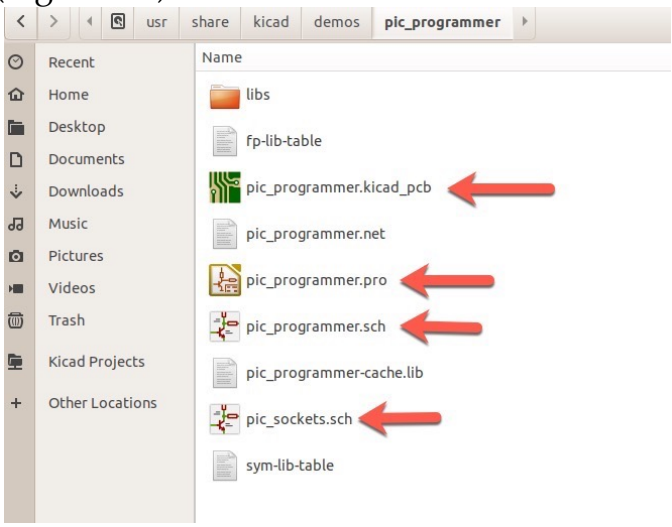


Figure 5.1: The contents of the 'pic_programmer' demo project folder.

The demo project folder contains several files that make up the project. The ones to focus on for now have the extensions 'pro,' 'kicad_pcb' and 'sch.' The file with the 'pro' extension contains project information. The 'kicad_pcb' file contains layout information. The files with the 'sch' extension contain schematic information. There are two 'sch' files because this project includes two schematics.

Double-click on the 'pro' (project) file. The main KiCad window will appear. This window is the launch pad for the other KiCad apps, like Eeschema (the schematic editor) and Pcbnew (the layout editor). You can see the main KiCad window in Figure 5.2.

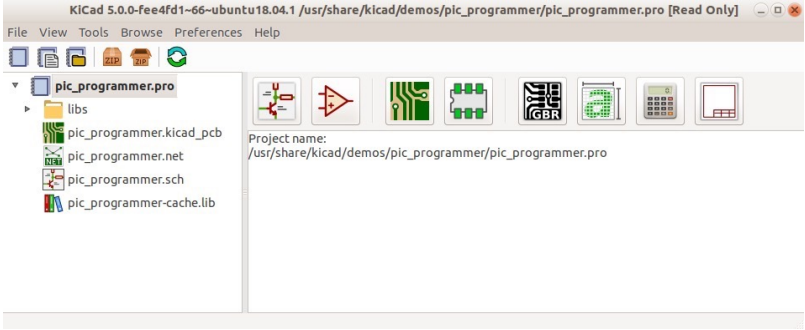


Figure 5.2: The main KiCad window.

The main KiCad window shows the project files in the left pane, the various app buttons in the top right pane, and various status messages in the bottom right pane. Let's explore the schematic of this demo project. In the top right pane, click on the first button from the left. This button will start the Eeschema app, the schematic layout editor. You should see the editor as in the example in Figure 5.3.

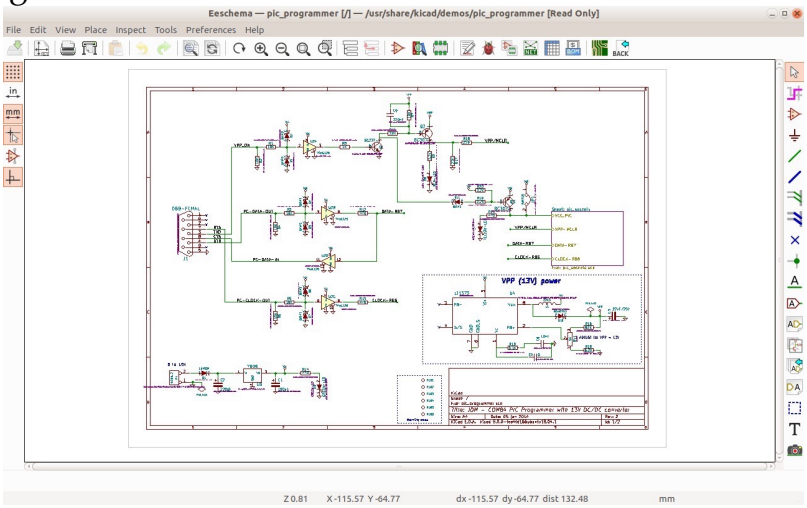


Figure 5.3: The schematic editor.

There are a few things going on here. At first, this window might seem overwhelming. Don't worry about the various buttons and menus, just concentrate on the schematic itself. Look at the various symbols, like those for the diodes, the transistors, and the operational amplifiers. There are symbols for resistors, and connectors, with green lines connecting their pins. Notice how text labels give names to the symbols, but also to the wirings between pins. Notice how even the mounting holes, at the bottom right side of the schematic, have names. Even though these mounting holes are not electrically active, they are depicted in the schematic. The values of the capacitors and resistors are noted, and any pins that are not connected to other pins are marked with an 'x'.

In the right side of the schematic, there is a rectangular symbol with the title 'Sheet: plc_sockets' (Figure 5.4).

Double click on it. What happened?

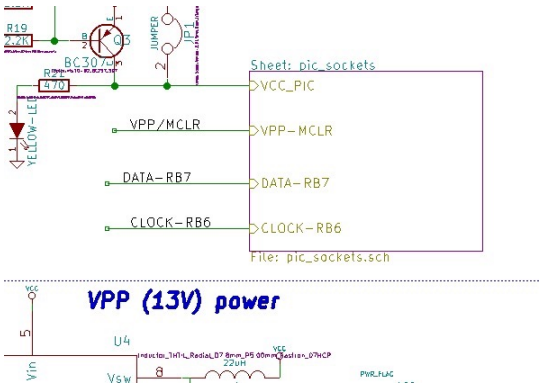


Figure 5.4: A link to another sheet.

This symbol is a link to another sheet, which contains additional symbols that are part of the same schematic. It looks like the example in Figure 5.5.

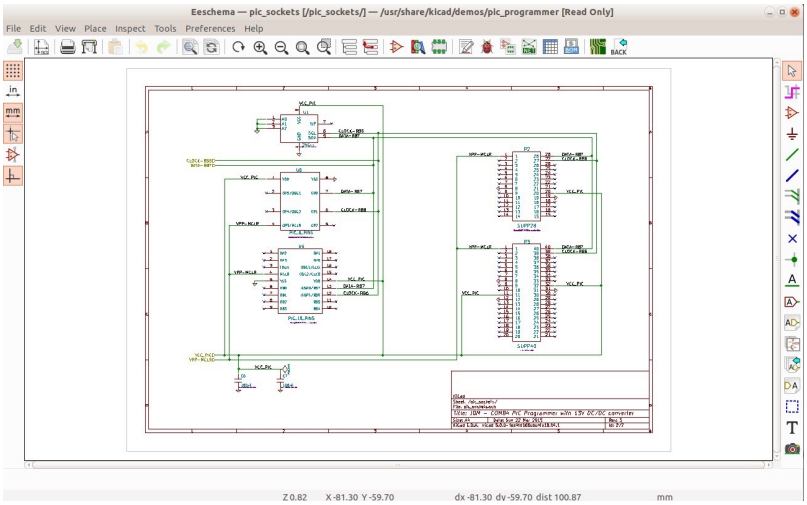


Figure 5.5: KiCad's schematics can span over multiple sheets.

KiCad's schematics can span over multiple sheets. If your schematic is too large to comfortably fit in one sheet, just add more (you will learn how to do this in this book).

I encourage you to spend a bit of time to study this schematic. You can learn a lot about how to draw good schematic diagrams by studying good schematic diagrams, just like you can learn programming by studying good open source code.

Go back to the main KiCad window. Click on the third button from the left, the one that looks like a PCB. This will launch Pcbnew, the layout editor. The window that appears will look like the example in Figure 5.6.

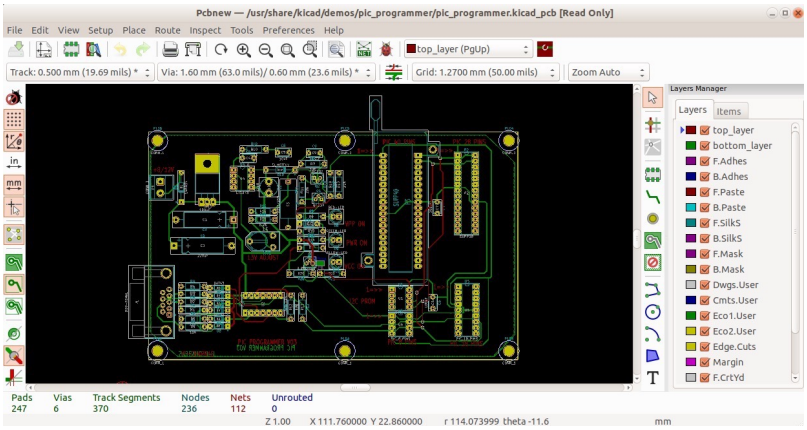


Figure 5.6: Pcbnew, the layout editor.

Again, don't worry about the various buttons and menus, just concentrate on the layout inside the sheet. Use the scroll wheel of your mouse to zoom in and out, and the Alt+right mouse button to pan (you should also be able to pan by holding down the middle mouse button). Zoom in and look at some of the layout details, such as the pads, how they are connected to traces, the names that appear on the pads and traces, and the colours of the front copper and back copper layer traces. Note: in Linux, panning is done with the middle mouse button, and the alt key is not used.

Also, compare how a footprint in the layout compares to the symbol in the schematic. You can see a side-by-side comparison in Figure 5.7.

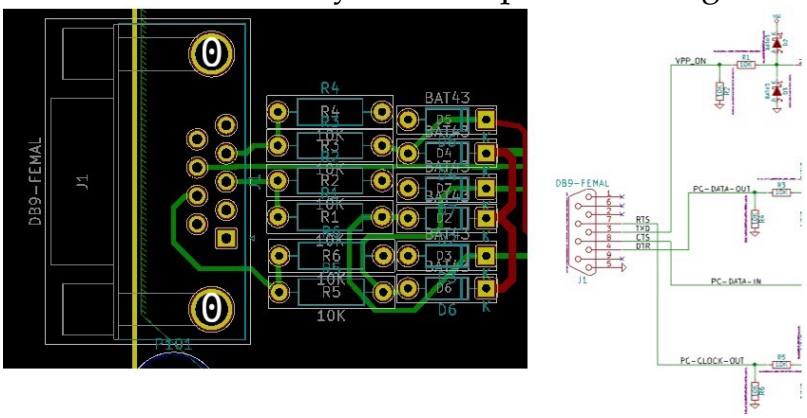


Figure 5.7: A side-by-side comparison of a footprint (left) and its schematic symbol (right).

Associated symbols and footprints have the same designator (J1, in this example), and the same number of pins. The layout shows the traces that correspond to the wires in the schematic.

Everything you see here is configurable: the width of the traces, which layer they belong to, the shape, size, and configuration of the pads. You will learn all of this in this book. In the layout, zoom in the J1 connector to see one of its details: the name of the trace that connects pad 7 of J1 to pad 1 of R5. Traces, like everything else in KiCad, have names. The names of everything that you see in Pcbnew are defined (manually or automatically) in Eeschema.

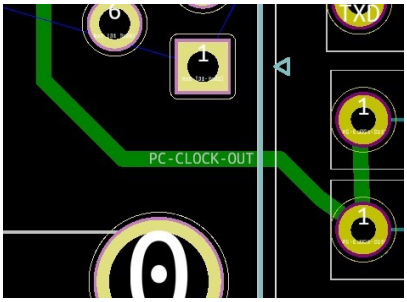


Figure 5.8: Traces have names.

Try one more thing: In Pcbnew, click on the View menu, and then choose the 3D Viewer. The 3D viewer will show you a three-dimensional rendering of the PCB, with remarkable detail. You can zoom and turn the board around to see it from any angle you want (Figure 5.9). Many of the components are populated, like the LED, resistors and some of the integrated circuits. For the rest, you can still see their pads and outlines on the board.

Figure 5.9: The 3D viewer will give you a realistic rendering of your board that you can examine in 3D.

As with the schematic editor, I encourage you to spend a bit of time to study the layout of this demo project. Later in this book, you will learn about the most important layout guidelines that will help you to design well-functioning and elegant PCBs.

Apart from the demo projects that KiCad ships with, you should also have a look at some of the very impressive showcased projects of boards designed using KiCad. For example, the CSEduino is a 2-layer PCB that contains an Atmega328P microcontroller and implements a simple Arduino clone. You will be able to easily create a board like this by the time you finish this book. Got to txplo.re/madewkicad for more examples of projects made with KiCad.

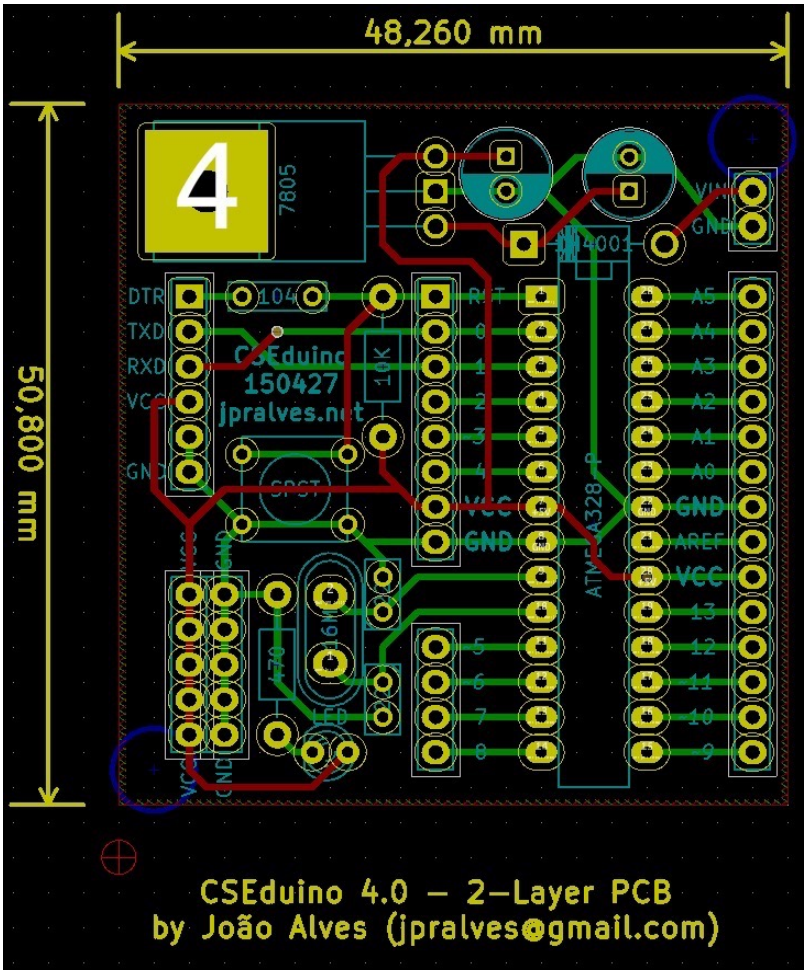


Figure 5.10: Featured board 'Made with KiCad': CSEduino.

Another featured board is Anavi Light, a HAT board for the Raspberry Pi. This is also a 2-layer board that allows you to control a 12V LED strip and get readings from sensors.

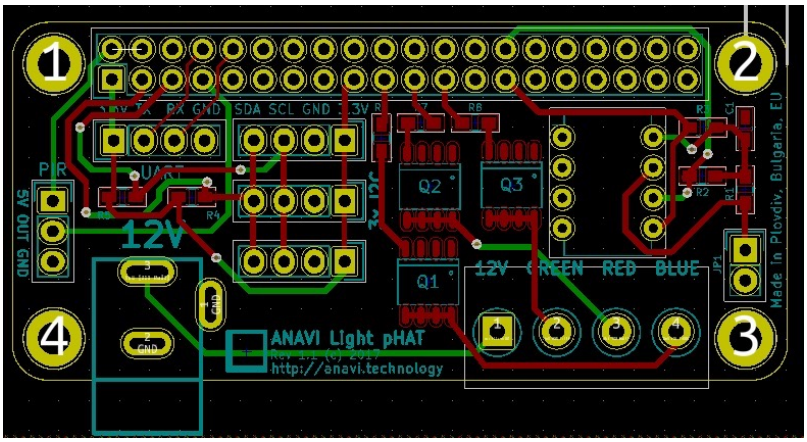


Figure 5.11: Featured board 'Made with KiCad': Anavi Light.

Finally, a truly impressive board made with KiCad is Crazyflie (Figure 5.12). Crazyflie is a dense 4-layer PCB with a rather elaborate shape. The board implements a the flight controller of a tiny drone. The shape is

specifically designed to implement the drone's body and arms. You will also learn how to create PCBs with complicated shapes in this book.

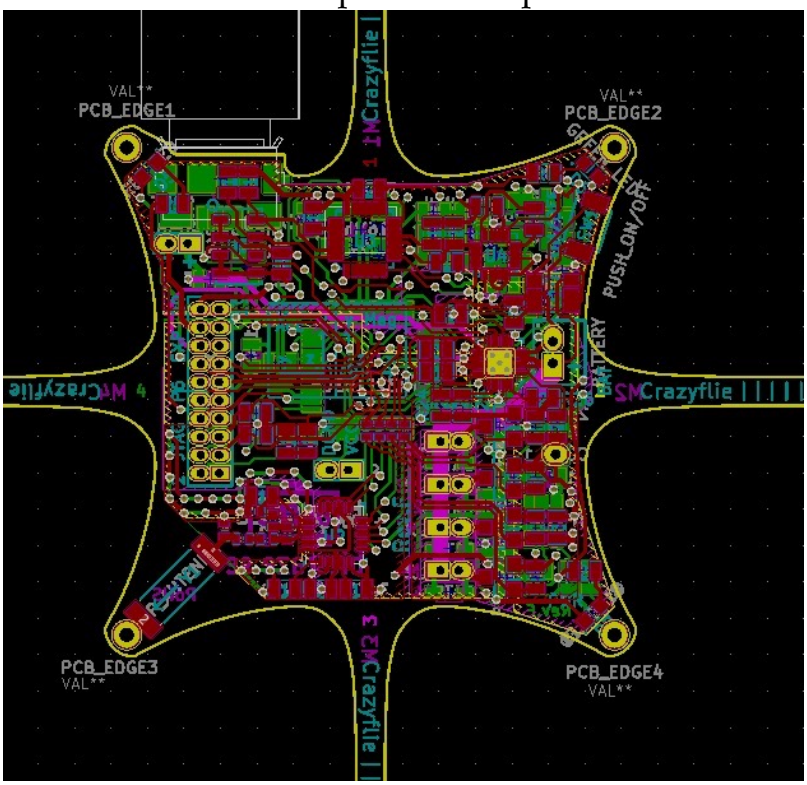


Figure 5.12: Featured board 'Made with KiCad': Crazyflie.

With this chapter complete, you should have a better understanding of the kinds of projects that people use KiCad for. These are also the kinds of boards that you will be able to design by the time you complete this book. Let's get straight into the first project so that you can start discovering this amazing tool by doing.

Part 2: A hands-on tour of KiCad with a very simple project

6. Introduction to this section

In this section, you will learn about the fundamental functions of KiCad, and create, perhaps, the simplest possible printed circuit board that contains components. I encourage you to carefully follow the instructions in this section and follow along by creating your version of this PCB on your computer. If you are already skilled in PCB design, feel free to skip this section.

Persevere, and you will come out the other end with:

1. the necessary skills for creating simple PCB in KiCad.
2. a basic but broad understanding of the KiCad process for producing a PCB.
3. a preview of some more advanced features that you will learn and practice in later projects in this book.

6.1. Main differences between KiCad 5.0 and 5.1

This "wedge" chapter contains a listing of the most important differences between KiCad 5.0 and KiCad 5.1. The bulk of this book is based on KiCad 5.0.0, which was released in August 2018. KiCad 5.1.5 was released November 2019.

KiCad 5.1 brings a lot of internal improvements and stability, but also a few significant changes in the organisation of the various menus and windows, as well as how the user interface works.

In this chapter, I list the most important changes that KiCad 5.1 brings, focusing on the user interface. There are many other changes that I have not listed here, most of them "under the hood".

If you are interested in knowing all the changes that the KiCad developers introduce with every release, please have a look at KiCad's Launchpad https://launchpad.net/kicad. For example, the release of version 5.1.5 contained 96 bug fixes, as you can see at the Milestone info page: https://launchpad.net/kicad/5.0/5.1.5.

Here are the main differences between KiCad 5.0 and KiCad 5.1:

1. Pcbnew: New Board Setup window replaces the Setup menu.

In KiCad 5.0, to create custom track widths and vias sizes, you would bring up the Design Rules Editor window via the Setup menu (Setup —> Design Rules). Under Setup, you could also bring up windows to customise settings for Differential Pairs, Text, Graphics, Layers etc.

In KiCad 5.1, the Setup menu was removed, and replaced by a single window, titled Board Setup. You can access this window through the File menu (File —> Board Setup).

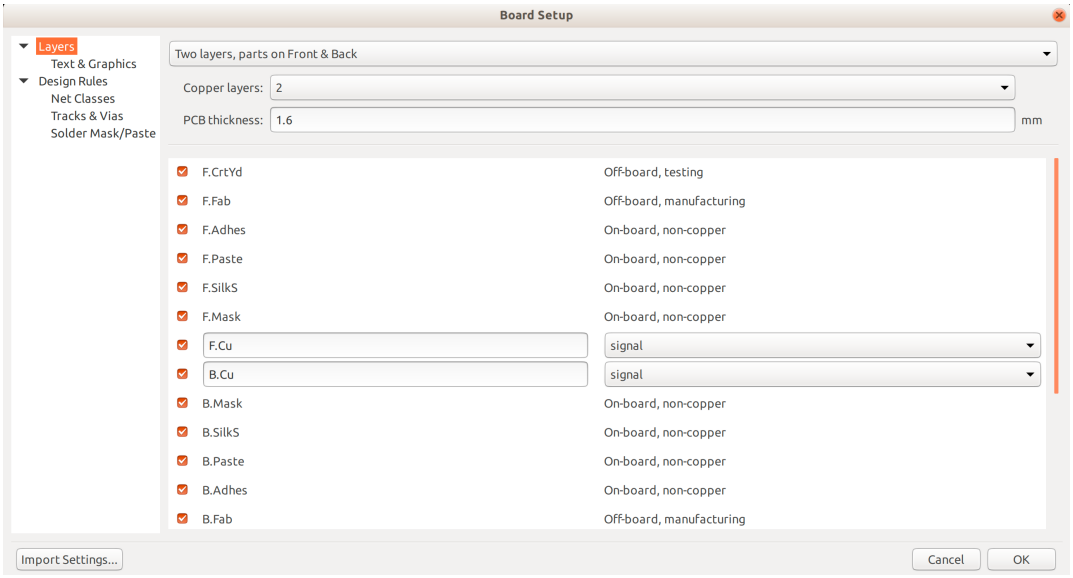


Figure 6.1.1: The Board Setup window was introduced in KiCad 5.1.

As you can see in Figure 6.1.1, the new window contains all of the options that formerly were available under the Setup menu item.

The Board Setup window also introduces a very useful capability: to import board settings from another project. To use this feature, click on the Import Settings button (bottom left of the Board Setup window), and select the ".pro" file (the project file) of the project from where you would like to import the settings. You can also select the settings that you would like to import (Layers setup, Text & Graphics, Design rules, etc.).

2. Pcbnew: Bulk-edit of text and graphics properties

A welcome enhancement of the user interface is the ability to change the attributes of text, track, graphics and via items in bulk. You can find the relevant windows under the Edit menu item.

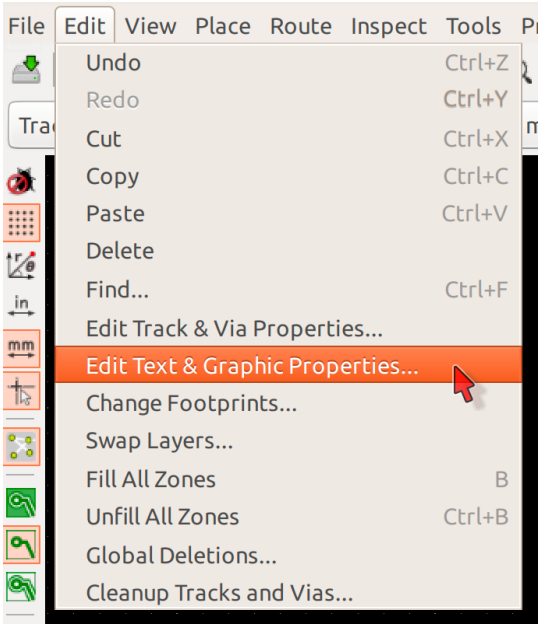


Figure 6.1.2: You can edit properties for track, via, text and graphics items in bulk.

The Edit Properties windows contain powerful filters for selecting the specific items that you want to change, based on attributes like their reference, footprint values, layer, and more.

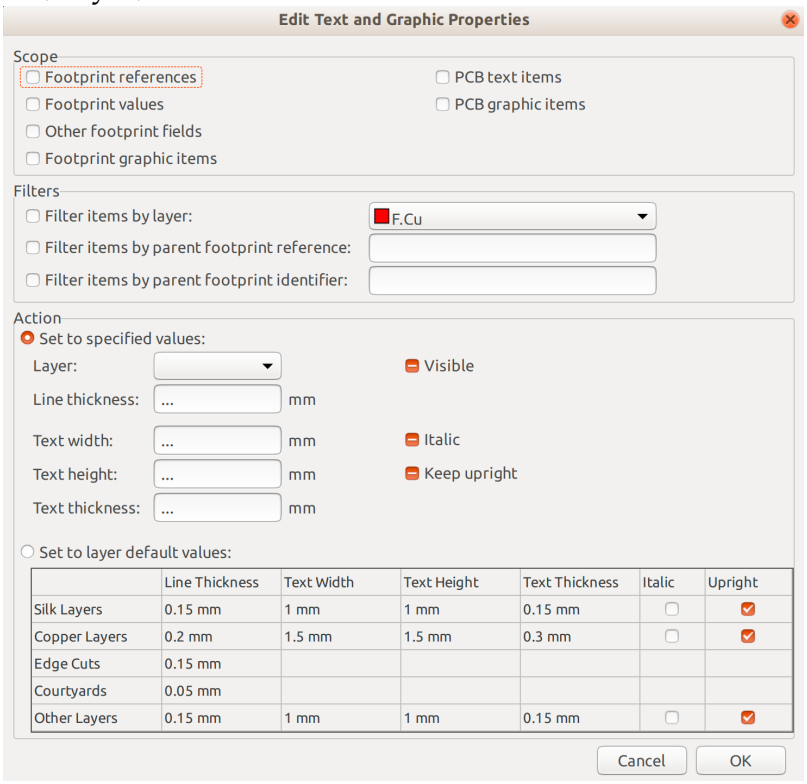


Figure 6.1.3: The Edit Properties window include powerful filters for selecting specific items.

I now use this new feature extensively to do things such as ensure that all my silkscreen text items are uniform and all my graphic lines have the same width.

3. Pcbnew and Eeschema: unified Preferences window

Another improvement is the introduction of a unified Preferences window. It is available under the Preferences menu item, and it is identical between Pcbnew and Eeschema.

In the Preferences window, you can set hotkeys that are common in both applications, and specialised for each one (so, the same hotkey can behave differently in Eeschema and in Pcbnew). There are multiple other settings that you can access here, that are common between the applications: the autosave frequency, which text editor to use, and the scaling of the toolbar icons (another new feature in KiCad 5.1).

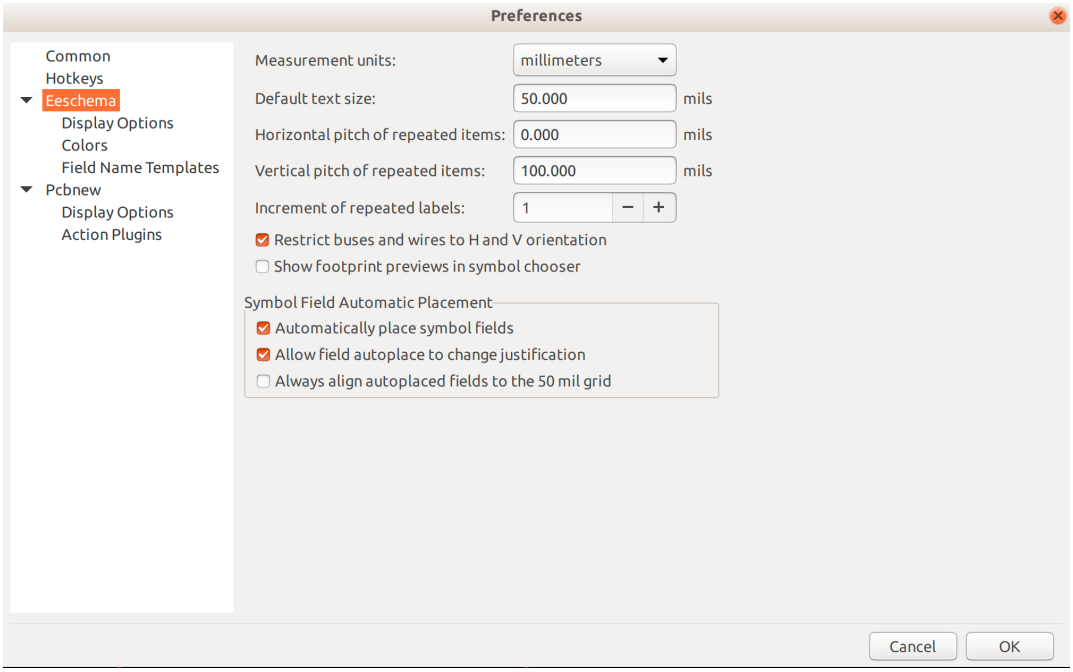


Figure 6.1.4: The new Preferences window.

The Preferences window has separate panes for the specific configurable attributes of Eeschema and Pcbnew. For each application individually, you can control the grid style and size, the cursor type, bus dimensions, colours, etc.

7. Start KiCad

Once you know what kind of PCB you want to design, go ahead and start Eeschema, KiCad's schematic editor. Start KiCad's main window and create a new project, as you can see in Figure 7.1.

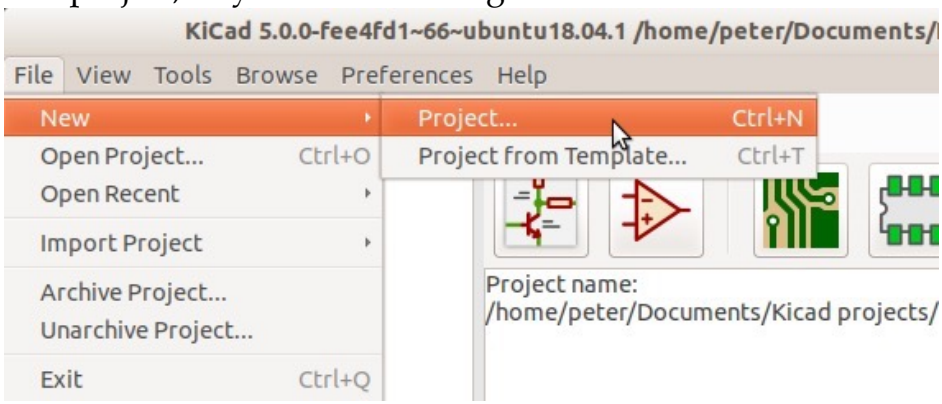


Figure 7.1 To start a new project, click on File, New Project, New Project.

Give it a sensible name, like 'LED and Resistor board,' and store the project in a new folder that is easy for you to find on your computer. In my example, I have named the project 'Breadboard_5V_power_supply', which is the first real project you will be working on later in this book. For this reason, the various file names and title bar names that you will see in this section will contain the text 'Breadboard_5V_power_supply'.

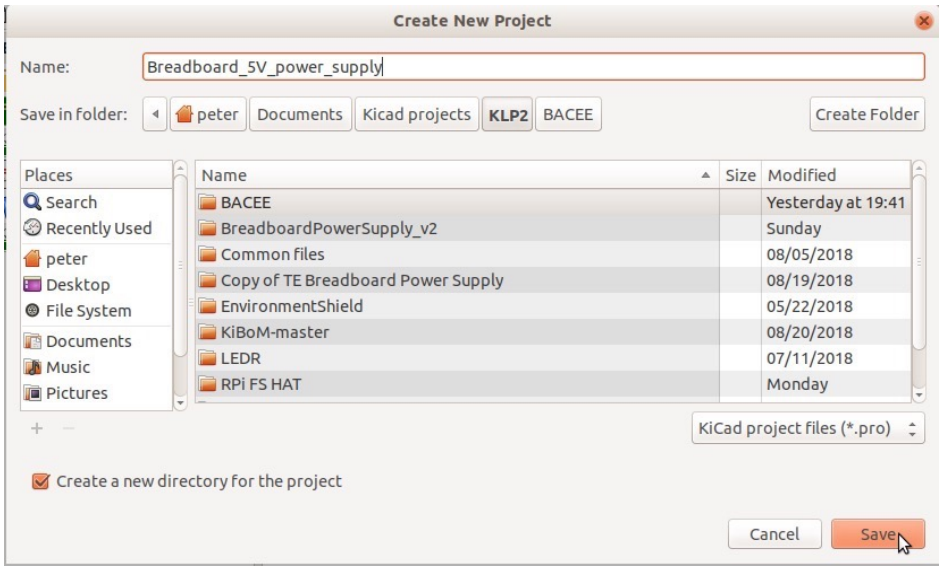


Figure 7.2: Give your project a reasonable, descriptive name.

Your KiCad main application window should look like the example in Figure 7.3:

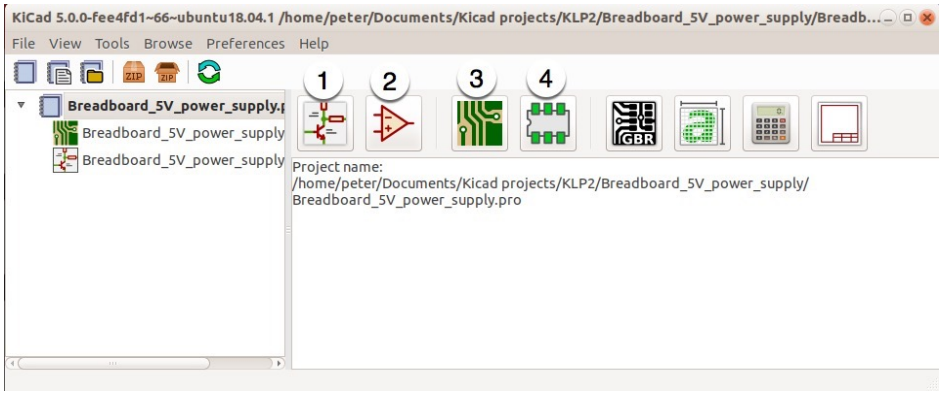


Figure 7.3: The main KiCad application window.

The main KiCad application window provides access to its various applications. It also provides a way to configure paths to the library and other files. In your new project, you should be able to see three new files:

- Breadboard_5V_power_supply.pro (depicted as a binder)
- Breadboard_5V_power_supply.kicad_pcb
- Breadboard_5V_power_supply.sch

In Figure 7.3, you can see that the '.sch' and '.kicad_pcb' files are depicted as nested inside the '.pro' file.

Have a look at the project directory using the file manager (Finder in Mac OS, and Windows Explorer in Windows). You will see all three files listed inside the project directory:

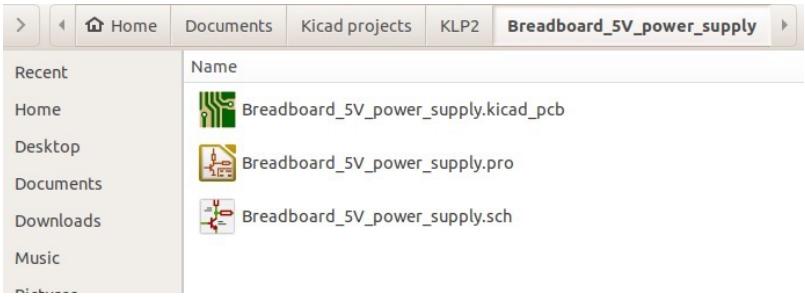


Figure 7.4: The files of a new KiCad project.

In a KiCad project, the information that binds the various files that make up the project is contained in the file with the '.pro' extension. This file is maintained by the project manager.

In a new project, the project manager will automatically create two files, one with the extension '.sch' and one with the extension '.kicad_pcb'. As you can probably guess, 'sch' stands for 'schematic'. This file contains data for your project's schematic and is maintained by Eeschema.

The file with the 'kicad_pcb' extension, is the one that contains the data of your project's PCB layout. This file is maintained by Pcbnew.

KiCad uses several other files, but you don't need to be concerned about them at the moment. If you are curious, you can find more information in the KiCad documentation. You can find the documentation at txplo.re/kcff.

You can start Eeschema in multiple ways. Let's try the most common option.

First, click on the button marked as '1' in Figure 7.3.

Second, by clicking through the Tools menu, as you can see in Figure 7.5 (below):

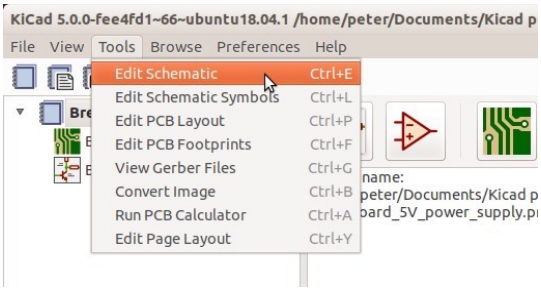


Figure 7.5: You can start Eeschema using the Tools menu.

This will bring up Eeschema, displaying an empty sheet (Figure 7.6). Let's begin working on the schematic next.

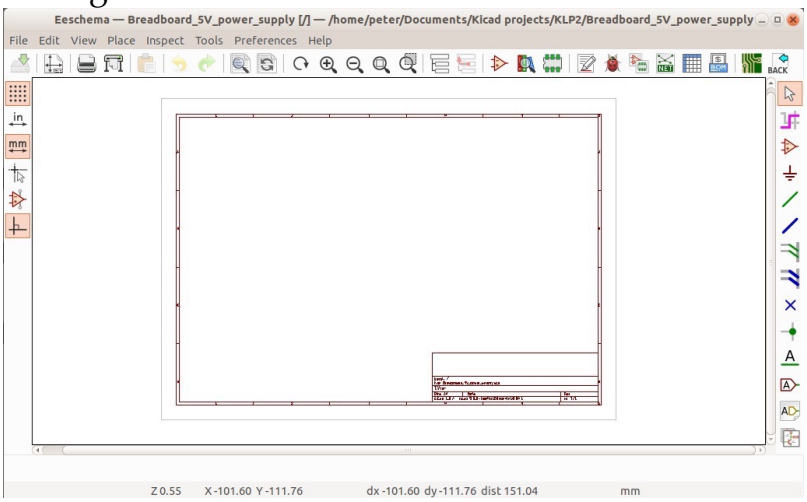


Figure 7.6: Eeschema with an empty sheet.

8. Schematic design in Eeschema

You are now in Eeschema and looking at an empty sheet, as in Figure 8.1. The annotations help us walk through Eeschema's menus items.

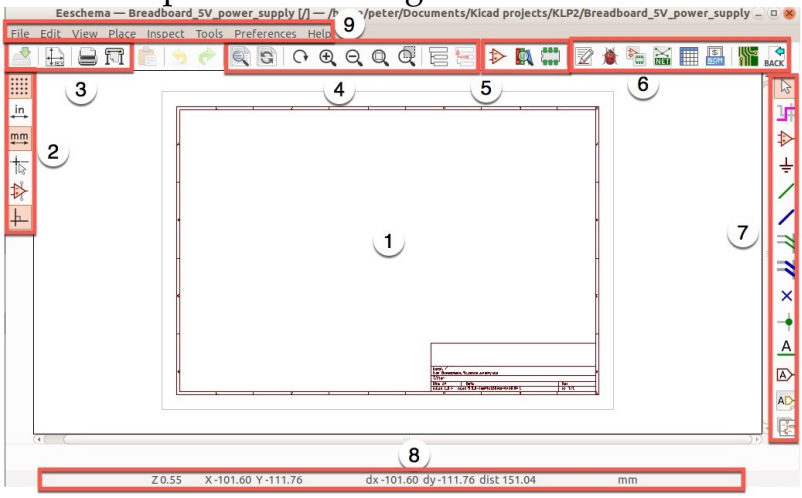


Figure 8.1: An empty Eeschema sheet.

Let's become familiar with Eeschema by taking a quick tour around its various buttons, menus, and controls. Eeschema consists of the sheet area in which you compose your schematic (1), multiple menu bars (2, 3, 4, 5, 6, 7), the status bar (8), and the menus (9).

Let's have a look at the useful mouse and keyboard controls before we look around the Eeschema user interface.

8.1. The schematic sheet

In Eeschema, you create schematics inside the sheet area. As is typical with engineering documents, at the bottom right corner of the sheet is the title block. The title block contains information about your schematic, such as its name, the name of its author (you!), and the date of the last edit.

In Figure 8.2, I have edited the title block to contain some example information. You can add as much or as little information as you wish. At the very least, I provide a title, the author name, revision, and a date.

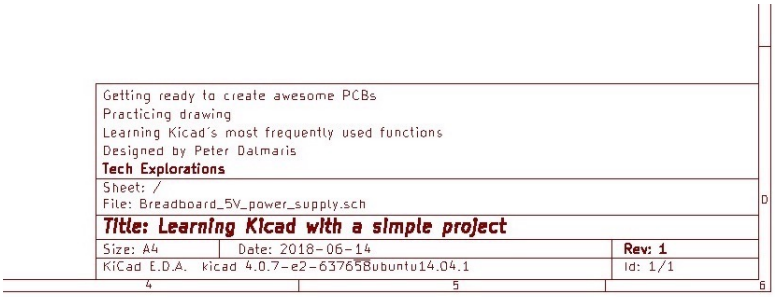


Figure 8.2: The title block filled with example text.

To edit the contents of the title block, bring up the Page Setting dialog box by clicking on File, then Page Settings (Figure 8.3).

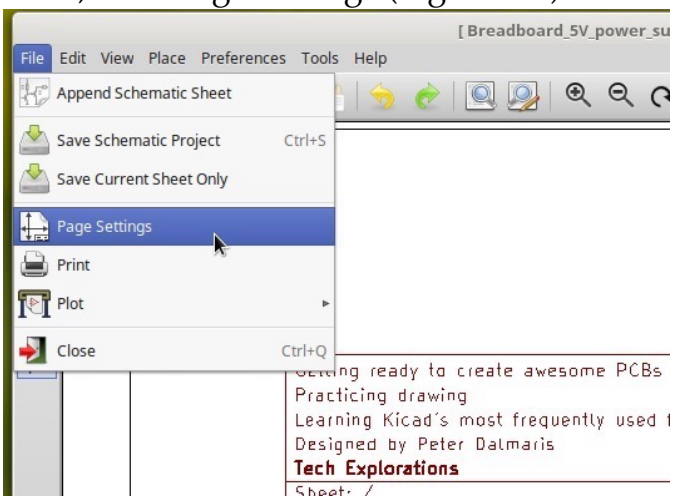


Figure 8.3: Bring up the Page Settings dialog box.

The Page Settings dialog box (Figure 8.4) contains two parts. On the left side you can control the size and orientation of the schematic page (sheet), and on the right, you can edit the information that appears in the title block.

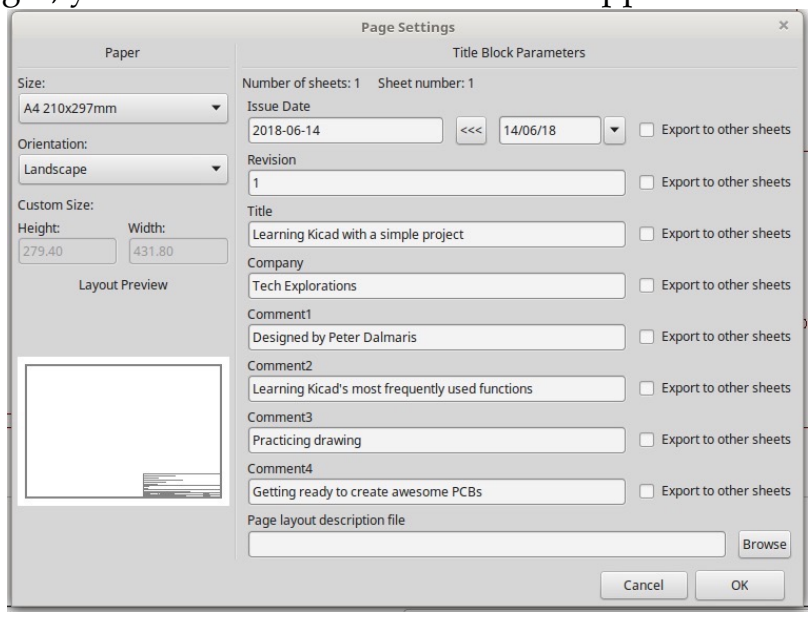


Figure 8.4: The Page Settings dialog box.

Go ahead and bring up the Page Setting dialog box. Start with setting your sheet's size and orientation. Engineering schematics tend to be oriented in landscape. Choose the appropriate paper size for your locality.

The right side is more interesting. At the very top is the Issue Date field. You can either type a date manually or click on the '<<<' button to have today's date copied into the field.

Continue with the rest of the fields. They are all self-explanatory. I always fill in these fields:

- Revision: Each time I make a change to my schematic, I increase the revision number by 1. Since we are starting with a new schematic, let's use '1' as the revision.
- Title: Something sensible that describes the project without having to inspect the schematic diagram in detail.
- Comment1: The author name.

Comments 2, 3 and 4 are additional lines that you can use to provide more information about the project.

Notice that on the right side of each field is a checkbox titled 'Export to other sheets'. KiCad allows for a single schematic to be spread out to multiple sheets, as you will learn in one of the more complicated projects in Section 5. When you select one of these checkboxes, the content of the corresponding text field will be copied and displayed in the title block of all sub-sheets.

At the bottom of the window is a field titled 'Page layout description file'. This makes it possible to use a custom layout for your schematic sheets. In my normal day to day work with Eeschema I rarely have to use a layout for the sheet other than the default, but know that this is possible.

I will return to this topic shortly to show you how to customise the sheet layout.

Go ahead and fill in your own information to populate the title block in your project's schematic sheet. Then click OK to commit the changes, and check that your text appears in the title block.

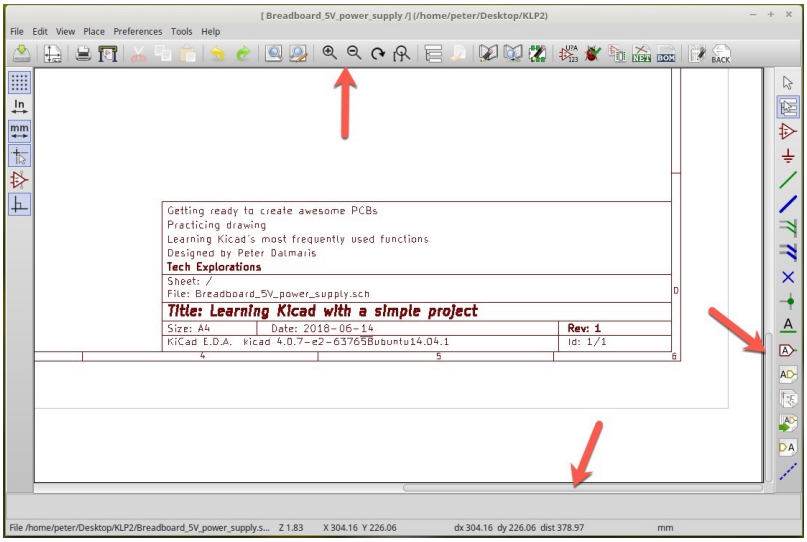


Figure 8.5: Zoom in/out and pan controls.

You can zoom in and out, and pan using the controls in the top toolbar and sheet window to reach a closer look of the title block. In the next part of this introduction, you will learn about more efficient ways to do this.

Page layout description file

Let's return to the 'Page layout description file' that we briefly looked at earlier. Remember that this is a field at the bottom of the Page Settings dialog box (Figure 8.4) that allows you to select a sheet layout file other than the default one.

If you wish to customise the way that the page looks in Eeschema, you can do so. Please jump over to the recipe titled '42. How to design a custom page layout' to learn how to do this.

8.2. Mouse buttons and hotkeys

We use the mouse and keyboard to interact with the sheet and its elements. If you have a mouse with three buttons (recommended), you can use all three to work with the symbols and the schematic on the sheet. I use a two-button mouse, with several programmable buttons (a Logitech MX Master 2S). Using my mouse configuration software, I can assign the keyboard keystrokes I use most often to the mouse programmable buttons. This can boost productivity significantly. Also important, is the scroll wheel. A scroll wheel is useful for zooming in and out of the sheet.

This table contains information about the default role of each button or button and key combinations:

Button, Keys	Role
Left mouse button (LMB)	Move a selection

Right mouse button (RMB)	Show the context menu for selection
Alt/Cmd + Right mouse button (RMB)	Pan
Scroll wheel	Zoom in/out at the position of the cursor

Table 8.1: Mouse controls in Eeschema.

Many other key and mouse button combinations are mentioned in the KiCad documentation; however, they will not always work as expected. This is particularly true if you have remapped your keyboard and mouse, or if you are using KiCad on a Mac, Linux PC, or inside a virtual machine.

In my experience, simple keyboard shortcuts work reliably. Try this now: with Eeschema in focus, type '?'. You should see the Hotkeys List (Figure 8.6, one of the most useful windows in KiCad (a similar Hotkeys List is available in Pcbnew). If the '?' Hotkey doesn't work, try Ctr-F1, or click on the Help menu item and choose 'List Hotkeys…'.

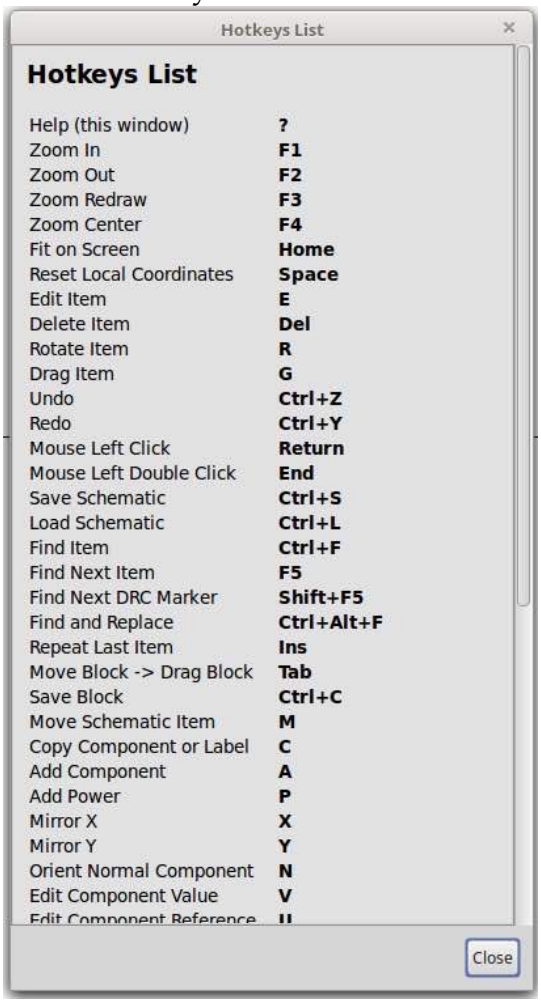


Figure 8.6: The Hotkeys List in Eeschema.

In Ubuntu Linux, which is my preferred OS for running KiCad in, you pan with the middle button. To resize the whole window, use alt-right-mouse.

There are a lot of hotkeys available, but for your quick introduction to Eeschema only a few are necessary. To use a hotkey, simply place the pointer of your mouse over a component and type the hotkey.

Hotkey	Role
A	Add a new component to the sheet
E	Edit, reveals a dialog box where you can edit all component attributes
R	Rotate (each time you press 'R', the component will rotate by 90 degrees)
G	Drag, useful for repositioning a component without disconnecting any wires
M	Move. If the component has wires connected, the wires will be disconnected
del	Delete
C	Copy
W	Start drawing a wire
T	Add graphics text, useful for providing information about the schematic
F3	Redraw the sheet, useful when you want to clear graphics artefacts

Table 8.2: A list of the most frequently used hotkeys in Eeschema.

There are more hotkeys available in Eeschema, but the ones listed in Table 8.2 are the most useful ones. Before long, you will commit them to memory and use them constantly.

You may wish to change some of those hotkeys if they interfere with your existing keyboard mappings. To do this, click on the Preferences menu item, then Hotkeys, and then Edit Hotkeys (Figure 8.7).

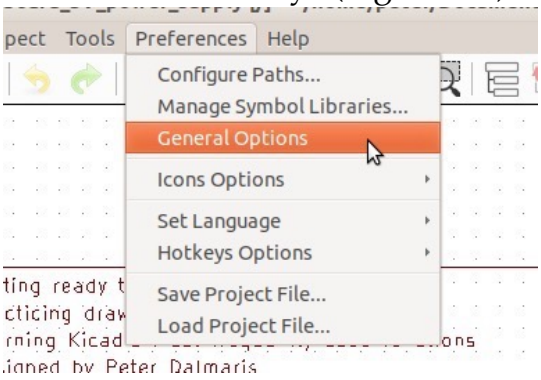


Figure 8.7: How to edit your hotkeys.

This brings up the Schematic Editor Options window. Click on the Controls tab to access the hotkeys editor (Figure 8.8):

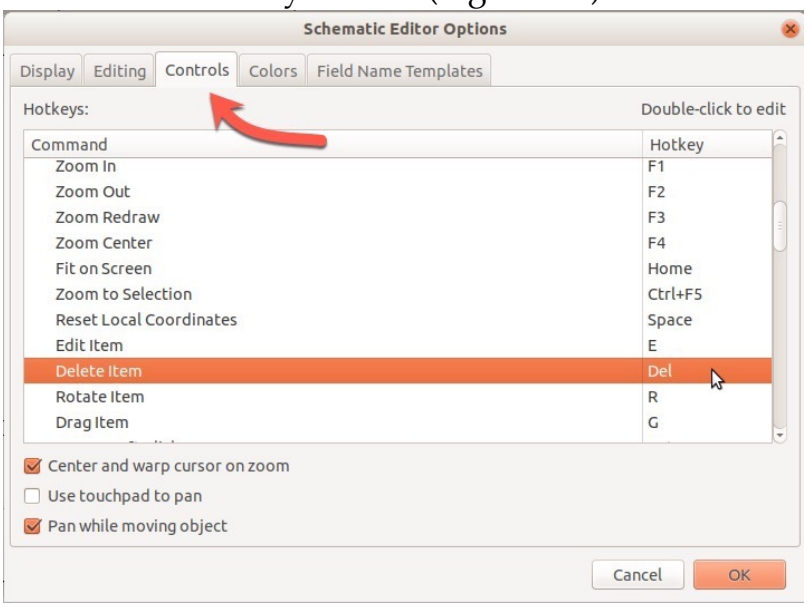


Figure 8.8: Select an existing Hotkey to highlight it, and then type in a new key or key combination.

To change a Hotkey, select it with your mouse and type in a new key or key combination.

Beware that these menus in Figures 8.7 and 8.8 may look different to yours what you see in your instance of KiCad, depending on your operating system and version of KiCad.

8.3. Eeschema buttons and menus

As you can see in Figure 8.1, Eeschema provides several menu bars, as well as a status bar and the menus. Let's look at the purpose of each one now.

Left toolbar

Menus item marked as '2' in Figure 8.1 consists of the left toolbar.

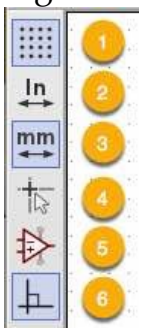


Figure 8.9: The Eeschema left toolbar.

The buttons in the left toolbar allow you to control various visual aspects of the sheet. The button marked with '1', for example, allows you to show or hide the grid. You can further control the size of the grid via the

Schematic Editor Options windows which are available through the Preferences menu item (Figure 8.10). Inspect the options in the Display and Editing tabs. In the Editing tab, you can select your preferred measurement unit (millimetres, inches etc.) among other things.

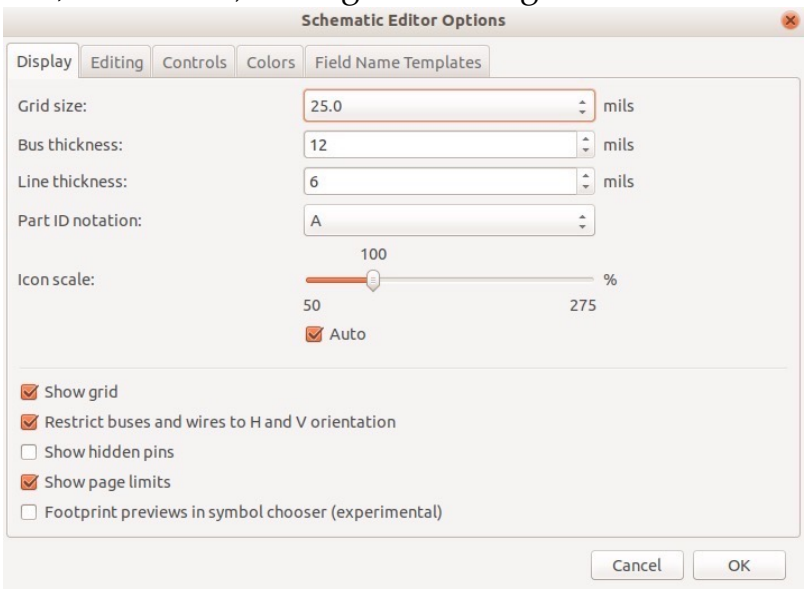


Figure 8.10: The Eeschema Preferences menu.

Depending on the grid size you have selected, you may need to zoom in before you can see the grid points. In Figure 8.11 below, you can see my 25mils grid. To be able to see it, I had to zoom in to zoom factor 3.67 using my mouse's scroll wheel. You can see the zoom factor in the leftmost edge of the status bar, at the bottom of the Eeschema window.

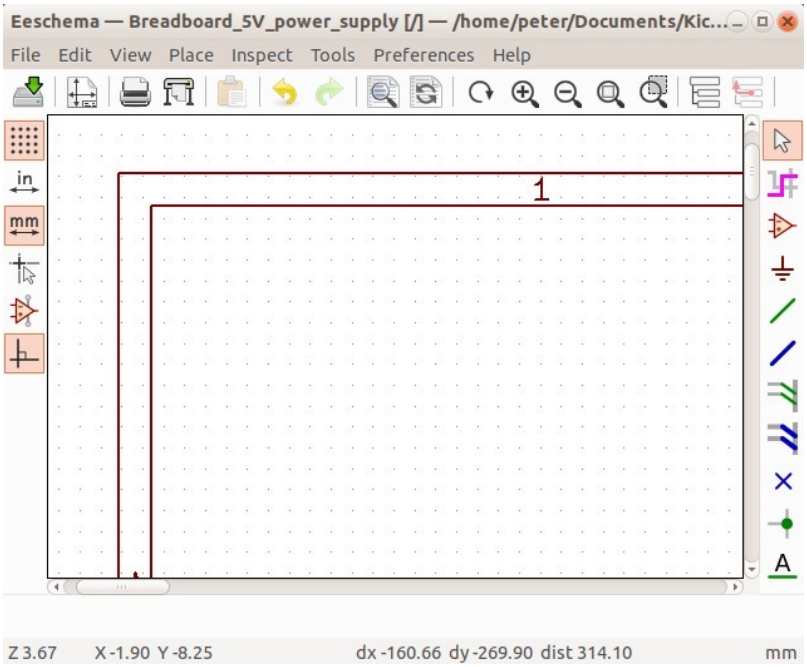


Figure 8.11: An example of my 25mils grid, at Zoom factor 3.67.

Buttons marked 2 and 3 in Figure 8.9 allow you to select your preferred length unit. You can select between inches (button 2) and millimetres (button 3).

Button 4 allows you to select the shape of the cursor. There are two options: small crosshairs, or long crosshairs (Figure 8.12). I find that long crosshairs are useful when you work on large and spread-out schematics. The XY lines extend across the full height and length of the sheet so that you can easily determine if the position of the cursor is aligned with the position of other components in the schematic.

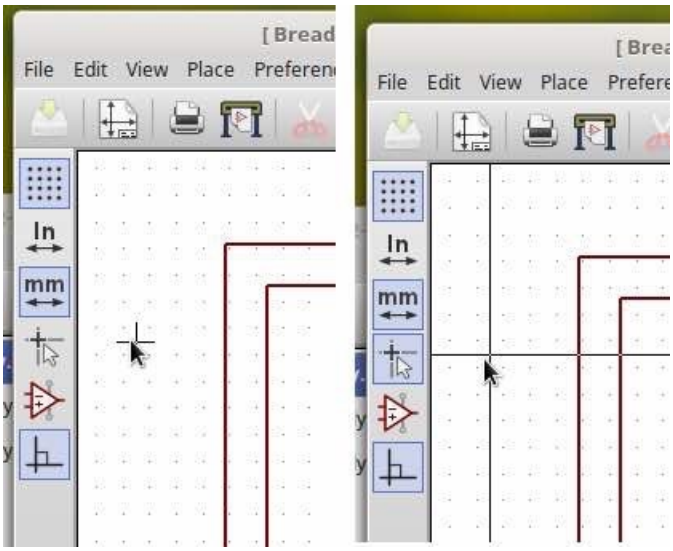


Figure 8.12: Short (left) and long (right) crosshairs.

Button 5 in Figure 8.9 allows you to show or hide normally invisible pins in integrated circuit schematic components. Such components are often designed to not show certain pins, such as power or ground, in order to reduce visual clutter. KiCad can automatically connect these hidden pins to the appropriate power sources if certain conditions apply.

Finally, Button 6 in Figure 8.9 allows you to lock the orientation of wires to 90 degrees when it is enabled. You can see an example of what this button does in Figure 8.13. The left green wire is restricted to 90 degrees orientation (button 6 was pressed), while the green wire on the right is not restricted.

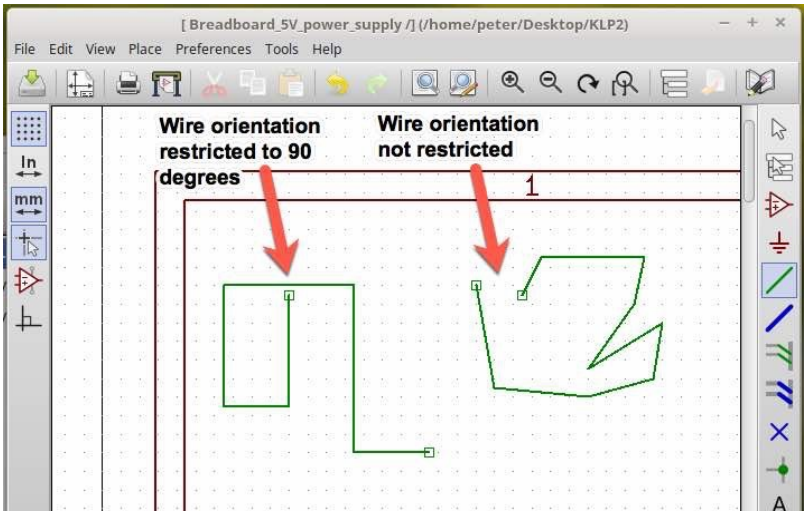


Figure 8.13: You can restrict wire orientation to 90 degrees, or leave it unrestricted.

You can experiment with the wire function too. Place your mouse pointer inside the sheet, and type 'W' to enable wiring mode. Move your cursor and click, then repeat to define the first segment of the wire. Continue for a few more segments. When you are finished, double click to end drawing the wire, or type the escape key to exit wiring mode. By using the Escape key, you actually cancel the drawing process, and the wire that you draw will disappear. Click on the orientation button to change orientation mode, and create a few more wire segments. Can you see the difference between the two modes?

I find that schematic diagrams look much better and are more readable when wire orientation is restricted to 90 degrees, so I do almost all my wiring that way.

Right toolbar

The right toolbar includes tools that allow you to add the various elements that make up a schematic, including symbols, wires, labels, and text. This toolbar also contains functions useful in manipulating sub-sheets.

You can see the right toolbar in Figure 8.14. In the following paragraphs, you will become familiar with the buttons you will need in this introductory project, such as the component and power symbol selectors, wires, and delete. We'll leave the rest (like the bus and entry points tool, global labels tool and the hierarchy navigation tool) for more complicated projects in Section 4.

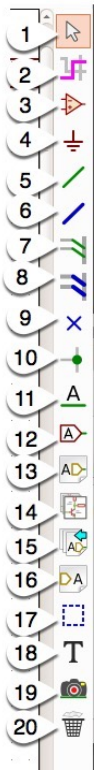


Figure 8.14: The right toolbar allows you to control the various elements that appear on the sheet, and sub-sheets.

When you click on the button labeled '1', you will get you out of whichever other command or tool you are using. For example, if you are working on creating wires using the wire tool (item 5 in the right toolbar), you can either type the ESC key or click on button 1 to exit the wire tool.

I often forget about the toolbar and instead use the hotkeys to switch from one tool to another, and the ESC key to reset the cursor. As I mentioned previously, hotkeys will become committed to your muscle memory and help you speed up your work with KiCad. Until then, let's continue with understanding the most important buttons in the right toolbar.

Component selector

Button 3 allows you to select a component from the library and drop this component onto the sheet. There are two ways to access the component chooser:

1. Click on button 3 to choose the component chooser tool, and then click on the sheet on the location where you would like to drop the component,
2. Or, place the cursor somewhere on the sheet where you would like to drop the component and then type 'A' (for 'Add').

Either way, the dialogue box in Figure 8.15 will appear.

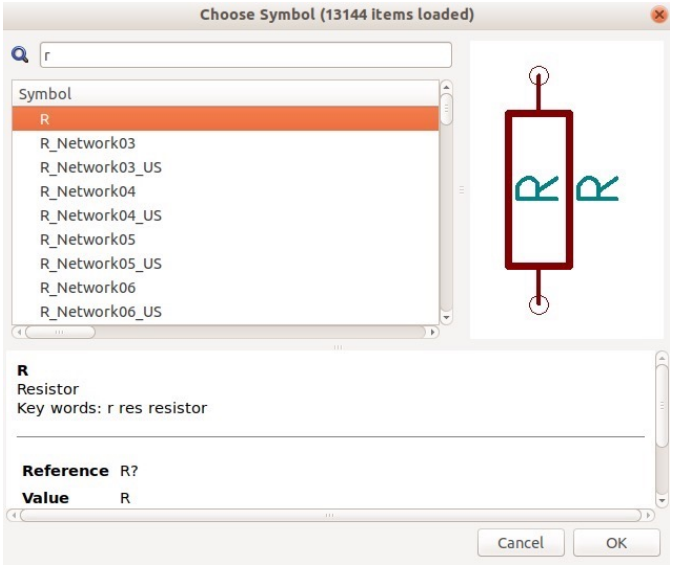


Figure 8.15: The component chooser.

To find a component, you can type its name in the filter box (top of the window), or browse through the libraries. In Figure 8.15 I typed 'r' in the filter as I was looking for a resistor. The resistor component appears first in the list of results, followed by other components that have a name that begins with 'R'. To drop the component to the sheet, click on it to select it and then click the 'OK' button.

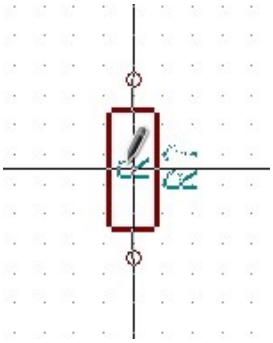


Figure 8.16: When you close the component chooser, the component is attached to the cursor.

When you close the component chooser, the component is attached to the cursor. You can move your mouse around and you will see how the component moves with it. You can also use the 'R' hotkey to rotate the component to the orientation that you need. When you have placed the component on the correct location and orientation, simply click to commit it in place. The component will then have a solid outline and label, and will no longer be attached to the cursor.

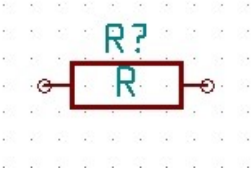


Figure 8.17: When you commit a component to the sheet, its outline becomes solid.

If you want to move it again, place the cursor on it and type ‘M’ (for ‘Move’). The component will become attached to the mouse and you will be able to move it until you commit to a new location with another click.

Power port selector

Button 4 in the right toolbar allows you to select and place a power port. A power port is a symbol that represents a power connection, such as a 5V voltage source or a ground (GND). You can find and select a power port using the normal component selector tool (button 3 in Figure 8.14). After all, a power port is just another symbol. The power port button simply provides a shortcut to this particular kind of symbol. Clicking on the power port button will bring up the same symbol chooser window that is depicted in Figure 8.15, except that now the filter takes us straight to the power-related components, saving us a bit of time.

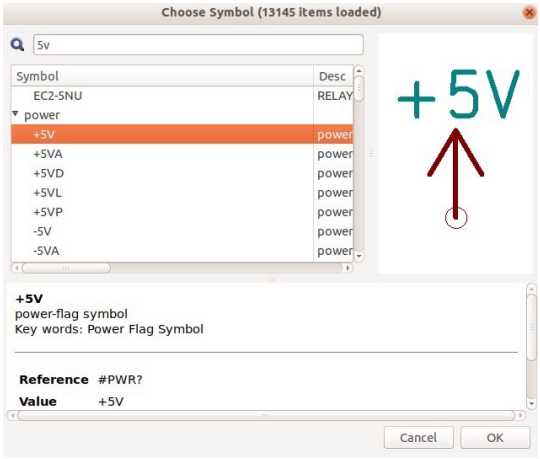


Figure 8.18: The power component chooser.

In Figure 8.18 I have also used the filter to find a 5V power source. Clicking the 'OK' button will add this component to the sheet. I now have 2 components in my sheet, as you can see in Figure 8.19:

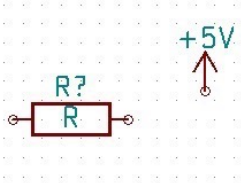


Figure 8.19: Two components are now in the schematic sheet.

Because the next tool you will learn about is the wire tool, let’s add two more components to our sheet: a GND power component, and an LED. After that, we will use the wire tool to connect them. 8.20 shows my current schematic. Try to make yours look as similar as you can do this by adding the

two new components and using the ‘R’ and ‘M’ hotkeys to rotate and move them to their correct positions.

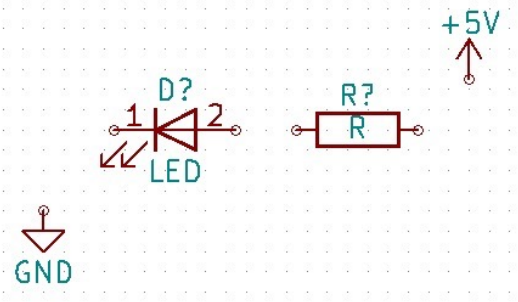


Figure 8.20: Four components are now in the schematic sheet.

Wiring and delete tools

Now that we have these four components in our sheet, let’s wire them. To do this, we’ll use the Wire tool, which is labeled as ‘5’ in Figure 8.14. To enable wiring mode, click on the wiring button or type ‘W’ (for ‘Wire’).

If you typed 'W' then your cursor is already drawing a new wire on the sheet. Move your cursor around and you will see the green wire segment following the cursor. Most likely, the beginning of the wire will not be within one of the circles that indicate a pin on a component. It is unlikely that you want to create a ‘floating’ wire (I.e. a wire that is not connected to something), so hit the ESC key to stop drawing the wire. The wire tool is still enabled (you can confirm that by looking at the state of the wire button in the right toolbar).

Let’s try again to draw the first wire. We’ll make this wire connect the GND component and the cathode of the LED component. Use the scroll wheel of your mouse to zoom in to the pin that is coming out of the GND symbol, and pan the sheet so that you can comfortably see both GND and LED components.

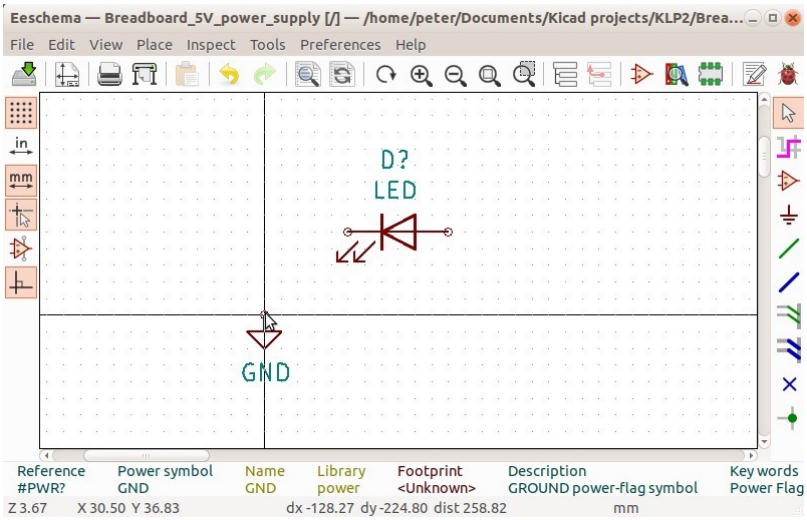


Figure 8.21: To connect a wire to a pin, place the cursor in the middle of the pin circle.

With your cursor right in the middle of the circle at the edge of the pin, and with the wire tool selected, click and release. You now have the start of the first wire segment.

Move the cursor to the centre of the circle that marks the edge of the cathode pin of the LED component. Assuming that you have turned off the wire orientation restrictions, you will now have a wire that connects the GND and LED in one straight line, like in Figure Figure 8.22:

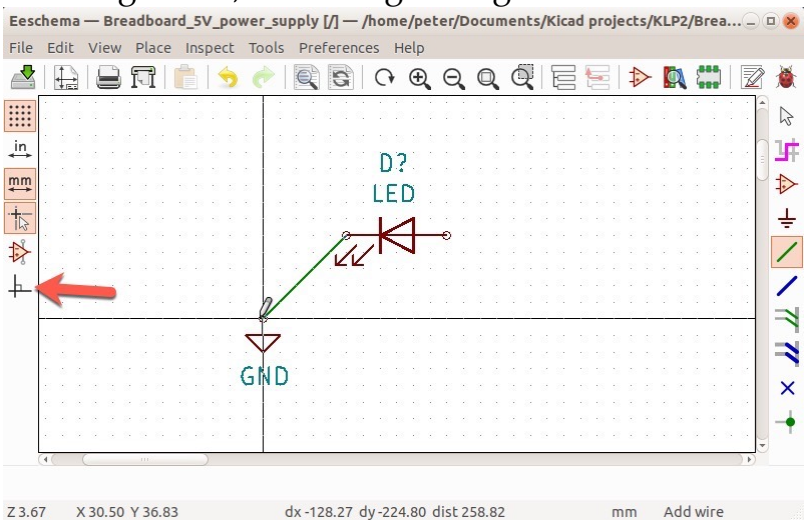


Figure 8.22: GND and LED are connected with a straight wire. Notice that orientation restrictions are off.

I prefer my wires drawn using horizontal and vertical lines, so let's try again. First, delete the existing wire by selecting the delete tool (marked as '20' in Figure 8.14). Then, click on the green wire to delete it. If your grid is large, you may not be able to get your cursor right on the line, and your clicks will have no effect. You can either change the grid size to something finer or click on the wire on a location where a grid dot and the wire touch.

Once you delete the wire, type 'W', click on the orientation button in the right toolbar to enable it, and try again to connect the GND and LED components. Just like the first time, click on the circle of the GND component, then click again on the circle of the LED's cathode pin. You should have something like this in your sheet:

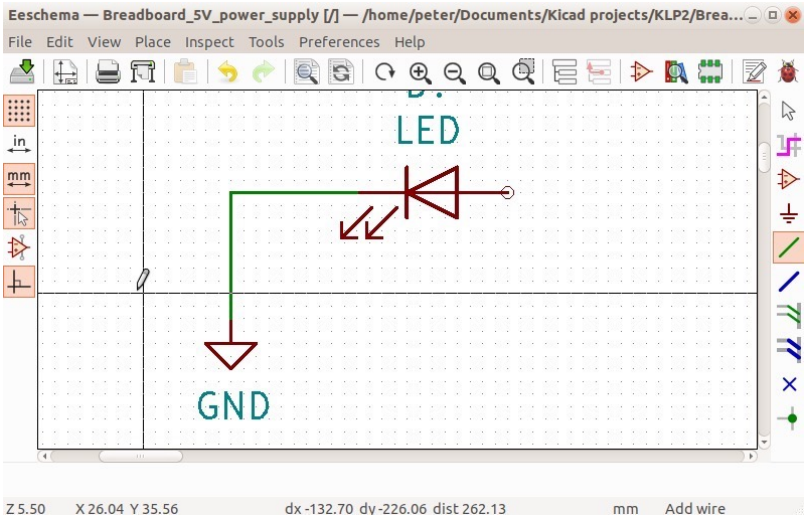


Figure 8.23: GND and LED are connected with a wire that forms a 90 degree angle. Notice that orientation restrictions are on.

The wire now has a 90-degree angle as it travels from the GND component to the LED. I think this is a better way to draw wires as it results in a more readable schematic. Let's continue with wire orientation restriction on, and connect the resistor and the 5V power source. At the end of the process, your schematic should look like this:

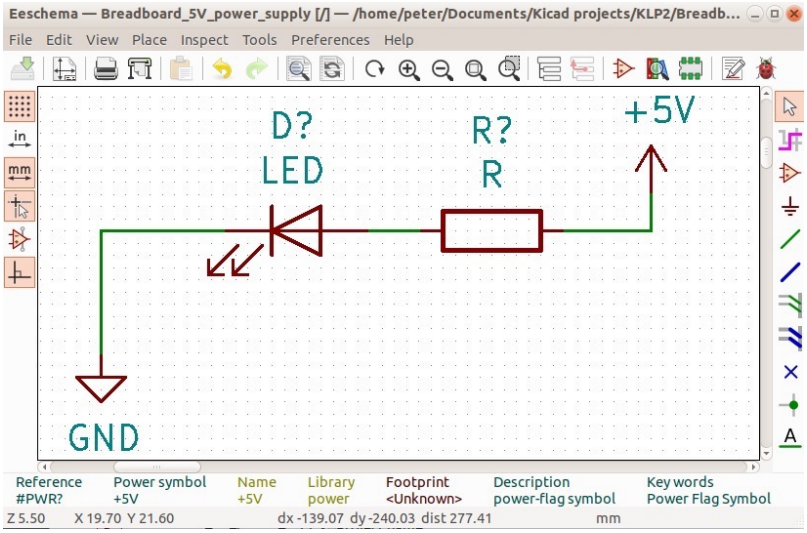


Figure 8.24: The final schematic of a simple circuit with four components.

Junction tool

The next one of the buttons that you will need for the first project is labeled '10' in Figure 8.14. This is the 'junction' button.

With the junction button, you can create connections between wires. Let's have a look at a simple example.

Go ahead and create two wires that intersect each other, like the example in Figure 8.25. To end drawing a wire in mid-air (not connecting the wire to a pin), double-click.

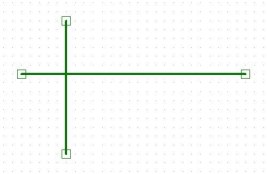


Figure 8.25: two wires intersecting but not connected (there is no junction).

These two wires may look like they are connected because the intersect, but they are not. There is no electrical connection between them. To connect them electrically, use the junction button (labelled '10' in Figure 8.14 or indicated by the arrow in Figure 8.26).

Try that now. Click on the junction button to enable it, and then click on the exact location where the two wires intersect. In Figure 8.26 you can see that a large green dot marks the exact location where the two wires are electrically connected. Also, notice that the junction button is pressed.

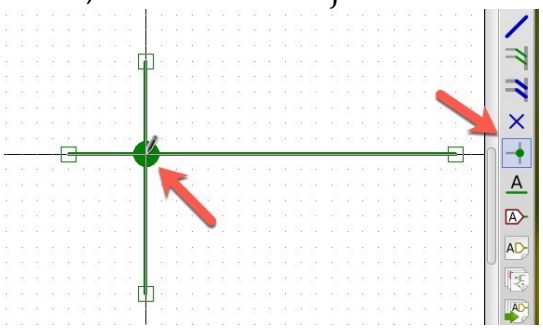


Figure 8.26: The two wires are electrically connected as indicated by the large green dot, that junction symbol.

You can also connect two wires by drawing the end of one onto any part of an existing wire.

Try this: use the 'W' hotkey to get back into wiring mode, start a new wire, and click on any part of an existing wire to end the drawing. Notice that a junction dot is added to confirm that the two wires are now electrically connected, not simply intersecting each other. Your schematic should look like the one in Figure 8.27.

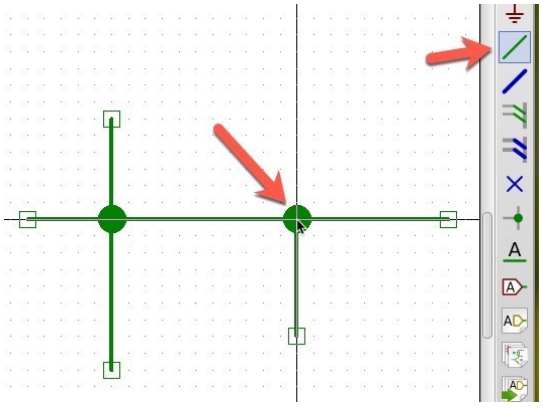


Figure 8.27: You can connect two wires by drawing the end of one into another.

You now have a completed circuit schematic, created using only 5 of the buttons in the right toolbar. That is enough to allow you to create your very first PCB, which is what you will do later in this section. You will learn about the rest of the buttons in the right toolbar later in this book, starting in Section 4.

Let's continue with the most important functions in the right toolbar.

Text tool

The last two of the buttons that you will use in this introductory project are the Text button and the Graphics Line button. These do exactly what their names suggests.

Using text and simple line graphics in your schematic have the potential to increase readability, especially in a large and complicated project.

Let's start with the Text tool. Click on the text button (labeled '18' in Figure 8.14), then click somewhere close to the LED in the schematic that you started earlier in this section. You will see the text properties window, as in Figure 8.28.

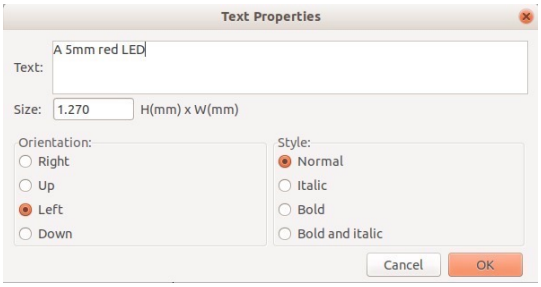


Figure 8.28: the Text Properties window allows you to insert free text in your schematic.

Type in something sensible to provide useful information that will assist the reader. I'd like my circuit to use a 5mm red LED, so that is what I typed in the Text field. I also like the text to appear in a small size font, so I entered '1.00' in the Size field. I left the rest of the widgets unchanged, but feel free to experiment with them and see what they do.

When you click 'OK', the new text will be attached to your mouse pointer so that you can position it at the exact location you want. The text appears faded as you move it around, like in Figure 8.29.

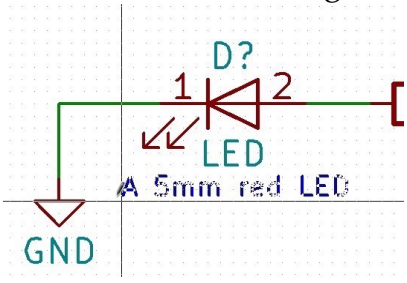


Figure 8.29: When you exit the Text Properties window, the new text is attached to the mouse pointer so you can position it.

When you have the new text in position, left-click your mouse to commit it. The text will look solid, as in Figure 8.30.

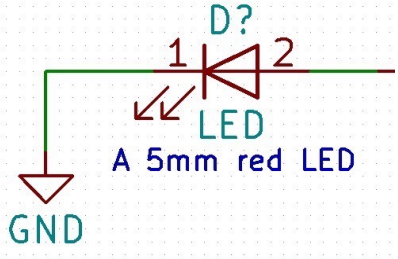


Figure 8.30: The new text is committed to the sheet.

You can use the text tool by using the 'T' hotkey (for 'Text'). Place your cursor on the position where you would like to place the text and type 'T'. The same dialog box that you saw in Figure 8.14 will appear, and from there onward the process is the same as if you had clicked on the Text button.

If you make an error in the text, you can easily fix it by placing your mouse over the text and typing 'E' (for 'Edit'). Again, the Text Properties dialog you saw in Figure 8.14 will appear, and any changes in the text or its properties will appear when you click on the 'OK' button.

The 'E' hotkey works with any other components in the sheet, not just text. Go ahead and try it on the LED and resistor components, and see what happens.

Try adding one more text item to provide additional information about the resistor. For example, add text the says 'current limiting resistor', so you end up with this result (Figure 8.31):

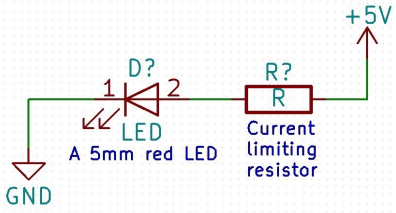


Figure 8.31: More text, this time in three lines.

Notice how the text 'Current limiting resistor' appears in three lines instead of one? You can do this simply by inserting a carriage return after each word, or after a word where you would like the line to end (Figure 8.32).

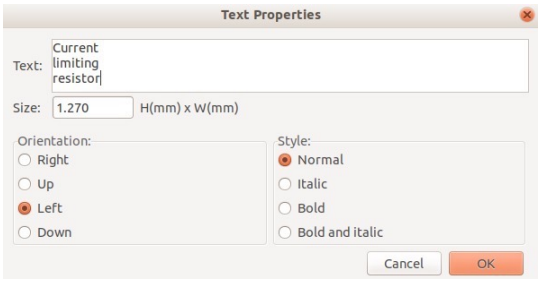


Figure 8.32: You can spread text over multiple lines.

In Figure 8.32, I spread the text over three lines, as you can see in the Text text field.

One more thing before we look at the graphics line and polygon tool: To move any component or schematic element to a new position, use the 'M' hotkey (for 'Move'). Try it now: place your mouse cursor over the text 'Current limiting resistor'. The text is now attached to the cursor. Move your cursor with the attached text to a new position, then click to commit.

I remind you that you can use the 'M' hotkey on any component, not just text. However, if the component that you are moving has wire connections, these will be broken, and you will have to rewire it manually. To move a component without breaking its wiring, you can use the 'G' hotkey (for 'Grab').

Play with the 'M' and 'G' hotkeys until you feel confident that you understand them, and then move to the next part where you will learn about the Graphics line and polygon tool.

Graphics line tool

The last of the buttons in the right toolbar that you will need for this introductory project is the Graphics Line of Polygon button. It is the fourth one from the bottom, labeled '17' in Figure 8.14.

By clicking on the button you will be able to draw straight lines. You can draw multiple straight lines and create boxes or other polygons. You can use these lines of polygons to do things such as group parts of your schematic together, put boxes around text or simply highlight text, or add some simple graphical elements to your schematic in order to improve its readability and make it more aesthetically pleasing. You can also enable this tool by using the 'i' hotkey.

Let's make a box to enclose our small circuit. Place your mouse around the bottom left edge of the circuit (or anywhere else where you would like the box edges to exist), and type 'I'. Draw the box by moving your mouse to the next corner and left-clicking to mark the end of that segment. To finish

drawing, double left-click. You should have something like the example in Figure 8.33.

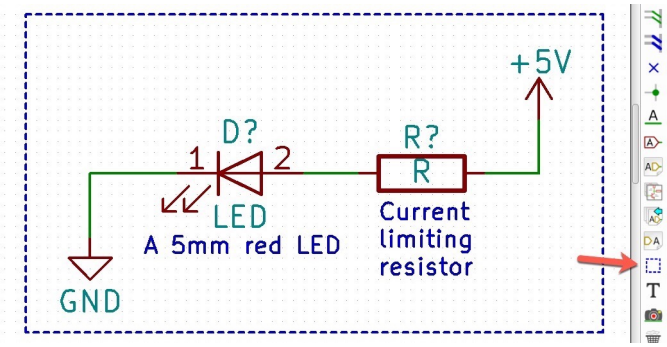


Figure 8.33: Use the Line tool to draw lines and polygons.

The line tool is very basic. You can change its color, thickness, or pattern. The only kind of line you can draw is what you see in the example of Figure 8.33.

If you make a mistake, you will need to use the Delete tool and click on each segment that you want to delete, then re-draw. The 'E' and 'G' hotkeys also work on lines.

Top toolbar

The top toolbar includes collections of buttons that perform a variety of functions. There are buttons for saving your schematic diagram to disk, printing it to paper or in other file formats, zooming in and out of the sheet, checking the correctness of the schematic, and more.

In this chapter, you will become familiar with the top toolbar and learn how to use the functions you will need for this first project. Figure 8.34 depicts this toolbar, annotated with numbers for easy reference.

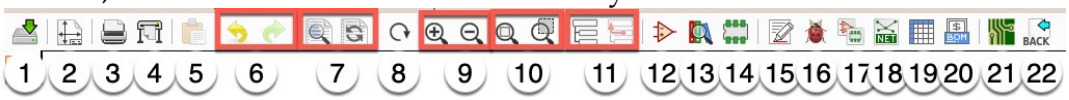


Figure 8.34: The top toolbar allows you to save and print your sheet, as well as perform many other functions.

Quick tour of the top toolbar

Let's have a quick look at the functions available through the top bar before we concentrate on those that are useful for this first project. Table 8.3 (below) describes the functionalities that you can access through the top-bar buttons.

Button	Description

1	Saves the file to disk (you can also type Ctr-S to achieve the same result, or click on File and then 'Save Schematic Project').
2	Configure the page settings. You have already learned about this in the segment titled '8.1. The schematic sheet'.
3	Print the sheet to paper.
4	Export the sheet to various file formats like PDF, PostScript, SVG, DXF and HPGL.
5	Paste a copied item
6	Undo and redo the last command.
7	Find and replace text in the sheet
8	Redraw (useful when the sheet graphics rendering leave artefacts behind)
9	Zoom in and out
10	The left button in this group will change the zoom factor so that the schematic fits in the window. The right button, when enabled, allows you define a rectangle with your mouse, and zoom in to the contents enclosed in the rectangle.
11	Navigate through hierarchical sheets
12	Access the library editor, which allows you to create or edit symbols
13	Access the library browser so that you can find a symbol and edit to your schematic
14	Access the footprint editor. With the footprint editor, you can create new footprints or modify existing footprints.
15	Annotate the schematic components so that each component has a unique identifier
16	Perform the electrical rules check (ERC). This check detects errors such as unconnected wires.
17	Associate symbols and footprints. This will trigger the Cvpcb application.

18	Generate the Netlist file. This file is then imported into Pcbnew so that you can continue the process with the board layout.
19	Symbol field editor. In this editor, you can edit the symbol fields in bulk, instead of doing so individual for each symbol in the symbol's properties window.
20	Create a Bill of Materials. This uses a plugin that can read the contents of the schematic sheet and generate a spreadsheet that contains a list of its components.
21	Start Pcbnew. Using Pcbnew, you can create the board layout.
22	Back-import component footprints (selected using CvPcb) into the 'footprint' fields.

Table 8.3: The functions that are available through the top toolbar.

Of the functions you can see in Table 8.3, the ones you will need in this project are 1, 2, 8, 13, 15, 16, 17, 18 and 20.

I'd like to concentrate on functions 13 (library browser), 15 (annotator), 16 (ERC), 17 (associator) and 18 (Netlist) since the rest are self-explanatory or you already know how to use them (1, 2, 8).

Library browser

Click on button 13 to bring up the Library Browser. It should look like the example in Figure 8.35.

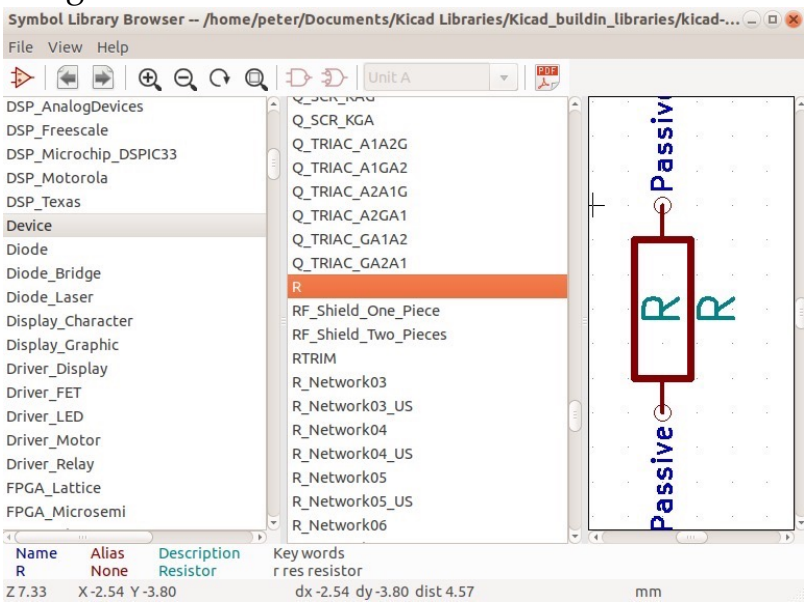


Figure 8.35: The symbol browser allows you to inspect the available symbols.

Using the library browser, you can find a symbol among the available symbol libraries. The symbol library browser is just that, a browser. It is useful for browsing through and editing the available libraries and symbols. When you find a symbol you want to use, you can insert it in your schematic; use the 'Place symbol' button from the right toolbar or the 'A' key for that.

The browser consists of three panes. The left-most pane contains the libraries. The middle pane contains the symbols that are part of the selected library. And the right pane shows the selected symbol.

You can resize the panes by placing your mouse over one of the vertical dividing lines, then dragging the line to the left or right using your left mouse button.

Try this out: in the left pane, scroll up and down to become familiar with the available libraries. Find the one named 'device', and click on it. Did you notice that the contents of the middle pane were updated as soon as you selected the device library? Now scroll up and down the middle pane and look for the symbol named 'R'. This is the standard resistor symbol. Click on it and see how the graphical representation of this symbol appears in the right pane.

You can use the scroll wheel in the right-most pane to change the size of the symbol, but please note that this won't change its size on your schematic; it only affects the symbol size in the library browser.

To quickly browse through the available symbols, you can use the filter. Try this now: Select the symbol library browser from the top toolbar, and then click on the Filter button (at the start of the arrow in Figure 8.36).

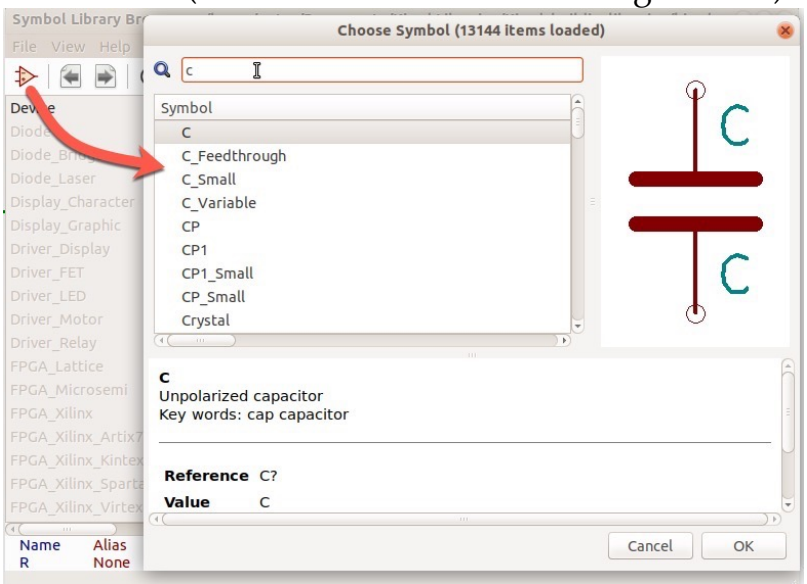


Figure 8.36: The symbols browser filter allows you to quickly find a symbol.

The filter window will come up, where you can add text in the filter text field. In the example in Figure 8.36, I typed 'C' as I was looking for a standard capacitor. Once you make your selection, double left-click on it or click Ok, to have the symbol's graphical symbol appear in the browser's right pane.

The annotator

Annotating your schematic is a required step of the PCB design process. To annotate your schematic, use the Annotator tool, which is accessible by clicking on button 15 in Figure 8.34.

Let's return to our tiny schematic, that consists of a resistor and an LED. Here it is again, in Figure 8.37:

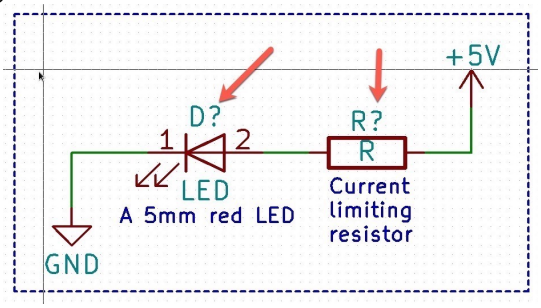


Figure 8.37: In this schematic, the diode and resistor symbols do not yet have a unique identifier.

In this schematic, notice that the diode and resistor symbols do not have a unique identifier. The arrows point to the question marks next to the 'D' and 'R' symbols, that indicate that the schematic has not been annotated yet.

To do the annotation, click on the annotator button (15 in Figure 8.34). This will bring up a new window that is titled 'Annotate Schematic', that you can see in 8.38.

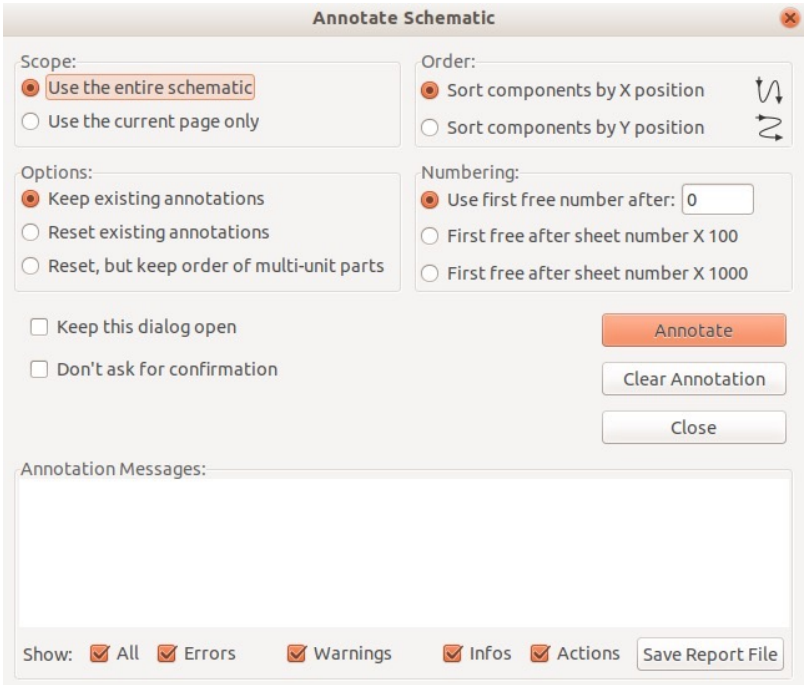


Figure 8.38: This dialog box allows you to control the annotator.

The annotator dialog box allows you to control how the annotator will complete its task. For example, if your schematic spans across multiple sheets, you can choose to annotate all of the sheets or only the current one. If you have annotated the schematic in the past and added new symbols which are not annotated, you can also choose to retain existing annotations and only annotate the new symbols. You can also control the direction of annotation, and the numbering system.

Let's keep things simple, and accept all the defaults. Click on the Annotate button, and accept the warning dialog box that may pop-up. This will complete the annotation, and update your schematic. Click on the Close button to close the dialog. The result should look like the example in Figure 8.39:

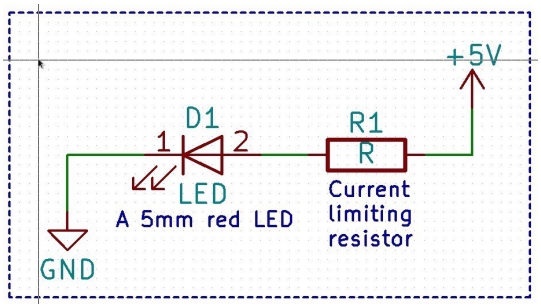


Figure 8.39: The symbols now have a unique identifier.

Let's continue with the electrical rules check (ERC).

Electrical rules check (ERC)

Our simple schematic diagram looks correct, from an electrical point of view. All wires seem to be correctly connected. The power ports also look correctly connected.

Not much can go wrong in a small schematic like the one we are currently working on. But as you start working on larger and more complicated circuits (as we will be doing very soon), the risk of errors will increase.

It is critical that we can identify and correct any errors before we continue our work with the PCB layout editor (Pcbnew), and this is the purpose of the Electrical Rules Check (ERC) function.

Let's go ahead and try this now, in our simply LED and resistor circuit.

Click on the ERC button (labeled 16 in Figure 8.34). This will bring up the ERC dialog box, which is empty (Figure 40).

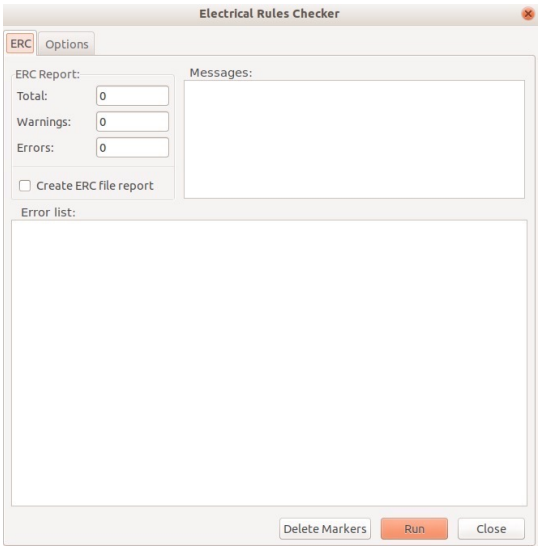


Figure 8.40: Starting the Electrical Rules Checker.

To start the check, click on the Run button. The ERC will do its work, and produce a list of errors if any. In my case, it did produce two errors, as you can see in Figure 8.41:

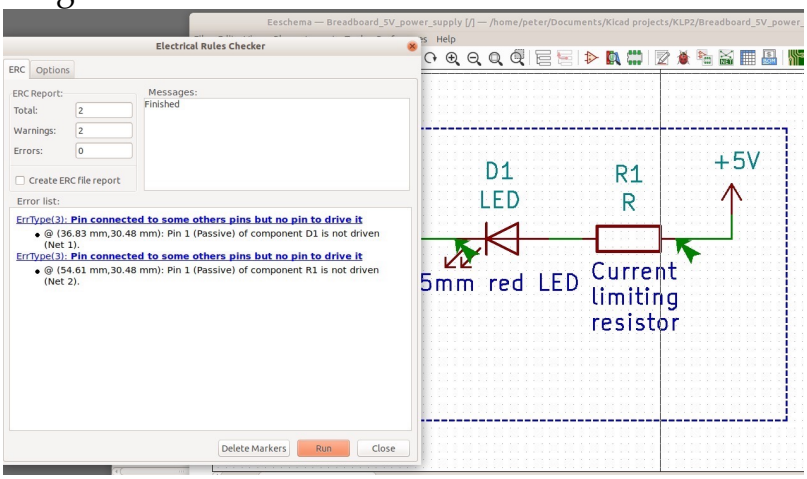


Figure 8.41: Even in such a simple circuit, errors are lurking!

The checker returned two instances of the same error:

```
ErrType(3): Pin connected to some others pins but no pin to
drive it
```

To help us find and fix the error, it also marked the location of the errors in the schematic using green arrows. This is probably one of the most frequent errors that I encounter in KiCad. The error message itself may not make much sense, but it is straight-forward. KiCad is complaining that there is no power source for the circuit. It 'knows' that every circuit needs a power source to operate, and that in the case of this circuit, it can't find one.

Even though we do have a 5V and a GND symbol, these have to be explicitly marked as power sources to satisfy the ERC rules. While not intuitive for people new to PCB design, it will make sense if you think about.

In general, a circuit contains wires that convey signals (like the voltage that comes from an analog sensor) and power (like the current that comes from a battery). KiCad needs to be able to tell the two apart, and it is our responsibility as designers to mark our schematic explicitly.

Let's go ahead and fix ERC's errors by marking the wires that are coming out of the 5V and GND symbols as power wires. Going forward, I will also be using the word 'net' to mean a wire. The word net is used extensively in KiCad, and it is good at this point to start becoming used to it.

Let's go to the schematic, place the cursor near the 5V symbol and type the 'A' key so we can add a new symbol. The symbol chooser will appear (Figure 8.42). Type 'pwr' in the filter to get to what we are looking for faster. The PWR_FLAG symbol is in the 'power' library.

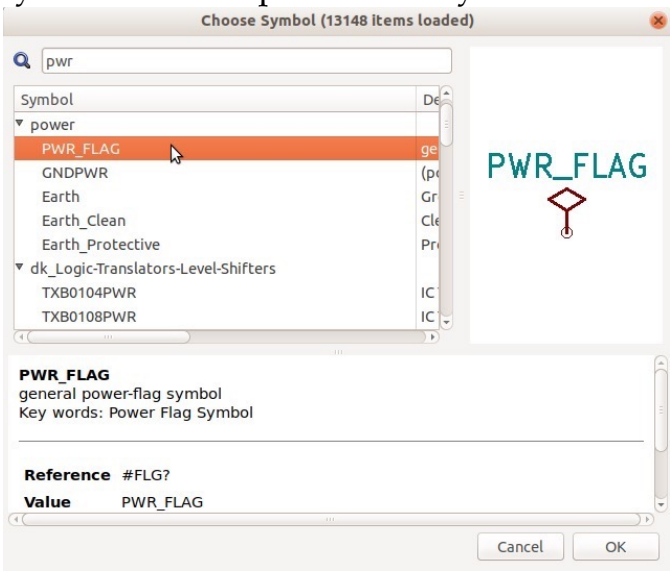


Figure 8.42: The power flag (PWR_FLAG) symbol.

Click 'OK' to add the PWR_FLAG symbol to the schematic. Position it close to the 5V symbol, and connect it to the net that connects the resistor and the 5V symbols. You should now have something like the example in Figure 8.43.

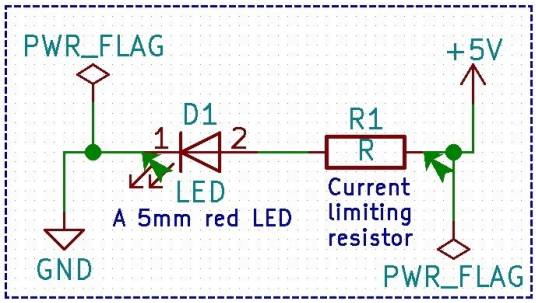


Figure 8.43: With the PWR_FLAG symbols we can mark nets explicitly as power nets.

What we have now is a circuit with the exact same symbols as before (an LED, a resistor, and the GND and 5V symbols). But now, we have

explicitly told KiCad that the nets between the LED and GND, and resistor and 5V convey power. The PWR_FLAG symbols are not 'real', in the sense that they do not have a footprint that will appear in our PCB. Think of them as tags, or instructions for providing KiCad with the information it needs to operate.

Connecting the PWR_FLAG symbols to the two nets with the error messages should fix those errors. Let's verify. Run the ERC again by clicking on the ERC button (labeled 16 in Figure 8.34), and then on 'Run'. Both errors should be cleared, and there should be no green arrows in the schematic. Your schematic should look like the example in Figure 8.44:

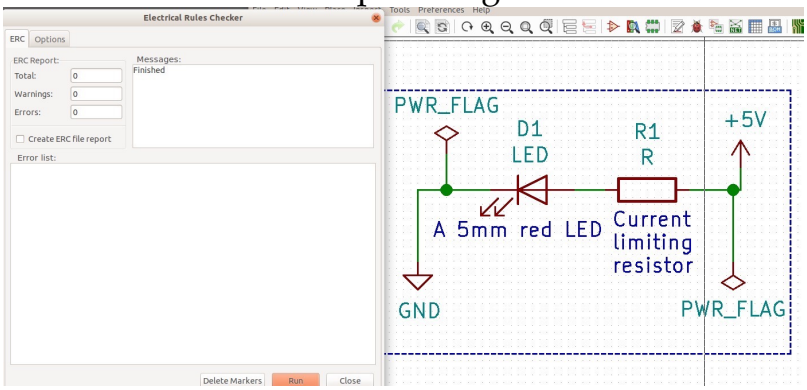


Figure 8.44: Thanks to the PWR_FLAG symbols the ERC errors are cleared.

Now that this simple schematic is electrically correct, we can continue by associating the schematic symbols (the D1 and R1 symbols that you see in Figure 8.44, with the footprints that we will work within the layout editor (Pcbnew).

Cvpcb: Associate components and footprints

KiCad's approach to PCB design is to recognise that a PCB starts its 'life' as a schematic diagram. In the schematic, abstract symbols represent the components that will eventually populate the PCB. But, the keyword here is 'abstract'. A simple resistor symbol in the schematic can be implemented as one specific physical resistor out of many possible ones.

You can choose to use a surface mount resistor, that come in a variety of sizes and packages: 0201, 0402, 0603, 0805, and so on. You may choose a through-hole resistor, which also comes in a variety of sizes and packages: 3mm, 6.5mm, 8.5mm and so on.

KiCad's designers decided that it makes sense to separate the schematic symbols from the footprints and give maximum flexibility to us, the PCB designers, to assign a footprint to a symbol. Once you get used to this idea, you will appreciate the power of this approach. You will be able to create a

single schematic, and from it derive multiple PCB layouts. For example, you can create a through-hole component version of the PCB during the prototyping stage to make sure that your PCB works, and then create a surface-mounted version of the PCB which is better for assembly using pick-and-place machines.

The tool that connects the schematic symbols with the footprints is Cvpcb. Let's use it now.

Click on the Cvpcb button (marked as '17' in Figure 8.34). In Figure 8.45 you can see Cvpcb before we make any associations.

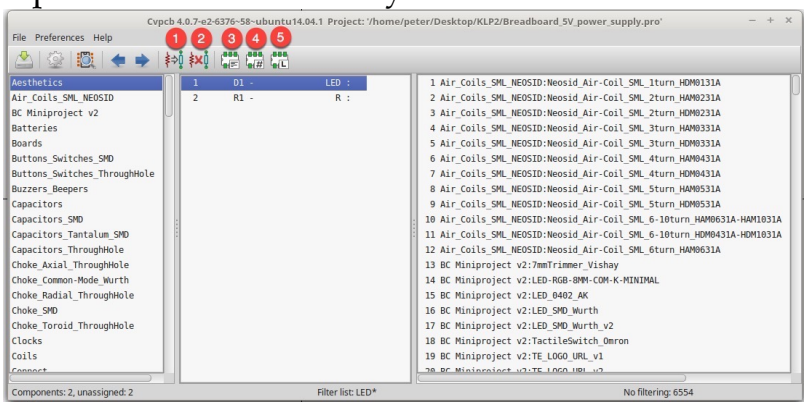


Figure 8.45: Cvpcb before we make any associations.

The tool contains three parts. In the Left pane are the footprint libraries. These libraries contain footprints. We will search in them to find appropriate footprints for the resistor and LED. In the middle pane is the list of schematic symbols and their associated footprints. At the moment, there are no associations.

Let's start by creating an association for the LED, which is already selected in Figure 8.45. By default, the first library in the left pane is selected, and a long list of items showing in the right pane. Cvpcb provides search filters in the form of the buttons 3, 4 and 5 in Figure 8.45. If none of the filters are selected, then in the right pane you will see all of the footprints that are included in all of the libraries. This is a very long list to search through even if you are only using the default footprints that come with KiCad. Click on any other library in the left pane, and you will notice that the list in the right pane changes.

Try out the Library filter, that is labeled as '5', and you can also easily recognise thanks to the 'L' in its button icon. This filter allows you to display in the right pane only those footprints that belong to the selected library in the left pane. Click on the Library filter button to enable it, and then try out a few libraries. See how the contents of the right pane change as you switch through libraries?

The button labeled as '4' is another filter that narrows down the items you see in the right pane based on the number of pins that the selected device in the middle pane has. For example, in Figure 8.45 the selected device has two pins. When you enable the 'number of pins' filter, only footprints with two pins will appear in the right pane.

Finally, the button labeled as '3' is the keyword filter. It narrows down the items you see in the right pane to only those that contain a keyword that matches that of the selected item in the middle pane. Click on it to select it. Unless you have selected the LEDs library in the left pane, the right pane should be empty, because the random library you have selected does not contain any items with the keyword 'LED' in its name.

With all three filters selected, use the scroller of the left pane to find a library named 'LEDs'. In the right pane, you will see all of the available matching footprints (Figure 8.46).

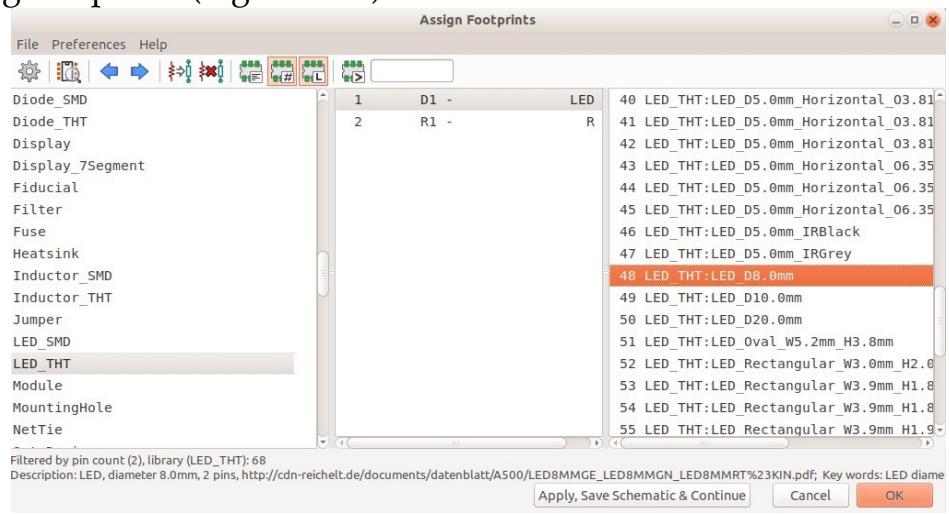


Figure 8.46: Finding the right footprint for the LED schematic symbol.

There are a lot of options. You can associate the LED symbol with a Surface-Mounted Device (SMD) or Through-hole (TH) component of various sizes and orientations. I am not too concerned about which one I will select in this example since we are only warming up, and will not actually manufacture this PCB. Select a reasonable footprint, like the one I have in Figure 8.46, which is an 8mm LED.

You can see the footprint by clicking on the Footprint button, as in the example in Figure 8.47 .

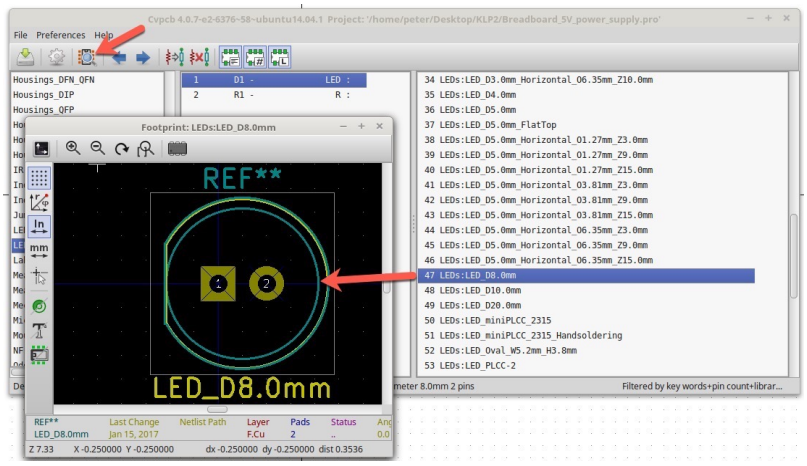


Figure 8.47: A view of the selected footprint.

To complete the association, double click on the footprint item. It will appear in the middle pane, next to the schematic symbol.

Practice this process one more time, and find a footprint for the resistor symbol. In Figure 8.48 you can see what my associations look like.

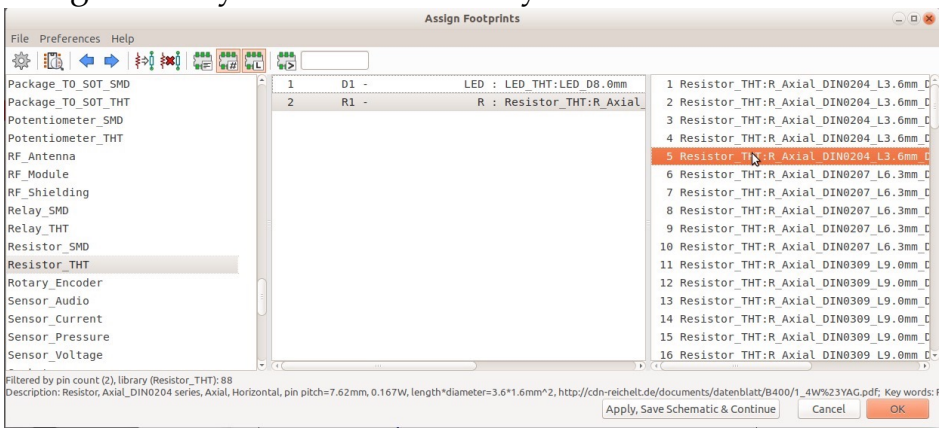


Figure 8.48: The resistor and LED symbols associated with footprints.

Our associations are now complete. Remember to save your work with Cvpcb before you exit this application.

Create the Netlist file

When your schematic is ready, meaning that you have completed drawing it, the ERC has passed, and you have associated symbols to footprints, it is time to continue the work in the layout editor, Pcbnew. Because in KiCad, the schematic editor and the layout editor are two different applications (Eeschema and Pcbnew respectively), there is an intermediate step necessary when you move from Eeschema to Pcbnew.

You must create a file called 'Netlist'.

The Netlist contains information about the symbols, their wiring, and their association with footprints. This information is used by Pcbnew to fetch the footprints from their libraries and place them in the layout design area and

to create the ratnets, which are thin lines indicating the electrical connections between the pins that we will need to implemented as as copper tracks.

Let's go ahead to create the Netlist file for our small circuit. Click on the button labeled '18' in Figure 8.34. It is easy to remember since it contains the acronym 'NET' in its button icon. The Netlist dialog box will appear (Figure 8.49).

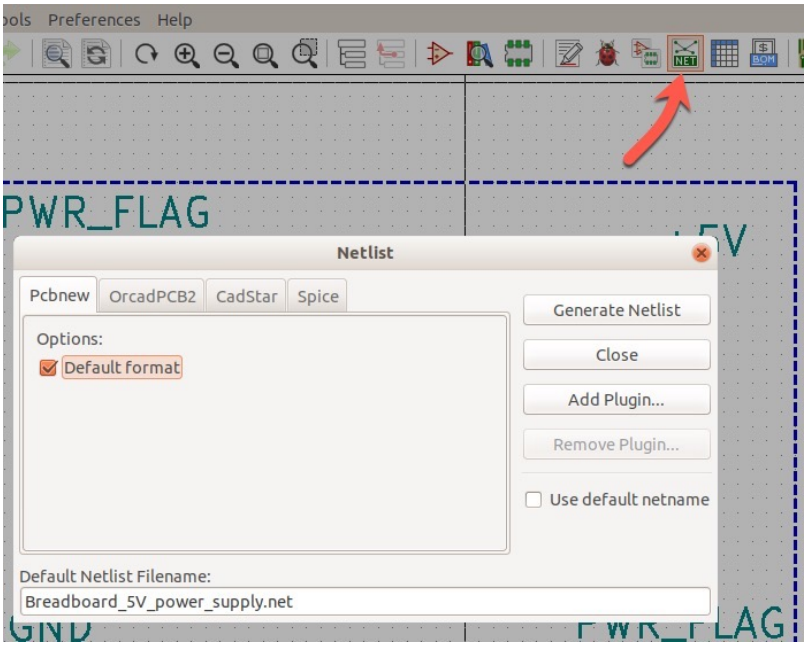


Figure 8.49: The Netlist dialog box.

The Netlist generator can create files that are readable by a variety of layout programs, but since we will be using Pcbnew, we will only work with the Pcbnew tab. Simply click on the Generate button to produce the Netlist file. By default, the Netlist file will be saved in the project directory, which will be easy to find once we get into Pcbnew shortly.

When you click on Generate, a new dialog box will ask you to confirm the save location of the file. Accept the default project directory by clicking on the Save button.

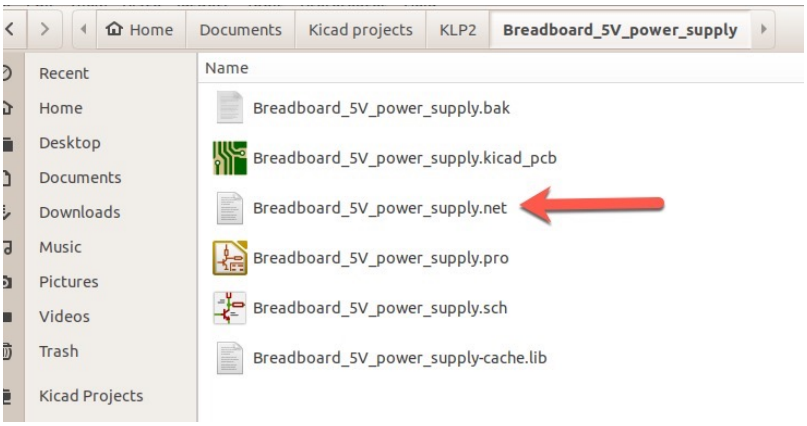


Figure 8.50: The new Netlist file is saved in the project directory.

You can confirm that the new Netlist file is created by looking inside the project directly. It is the file with the '.net' extension in its filename.

Before we continue work on this simple PCB, let have a quick look at the status bar and the menus in Eeschema.

Status bar

The status bar is the area in the bottom of the Eeschema window. You can see an example in Figure 8.51.

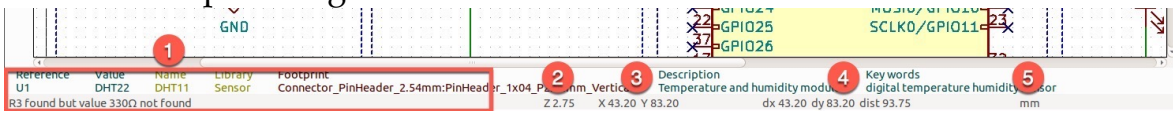


Figure 8.51: The status bar in Eeschema.

The status bar contains useful information in five segments. With reference to the labels in Figure 8.51, we have:

1: Status messages such as the total number of nets in a schematic or the next command you are about to select from a menu item.

2: The zoom factor. As you zoom into the sheet to increase the level of detail you can see, the zoom factor also increases.

3: The absolute coordinates of your mouse cursor. The coordinate system in KiCad has its origin in the top left corner of the sheet. The coordinates in segment 3 of the status bar are in relation to the origin (see Figure 8.52).

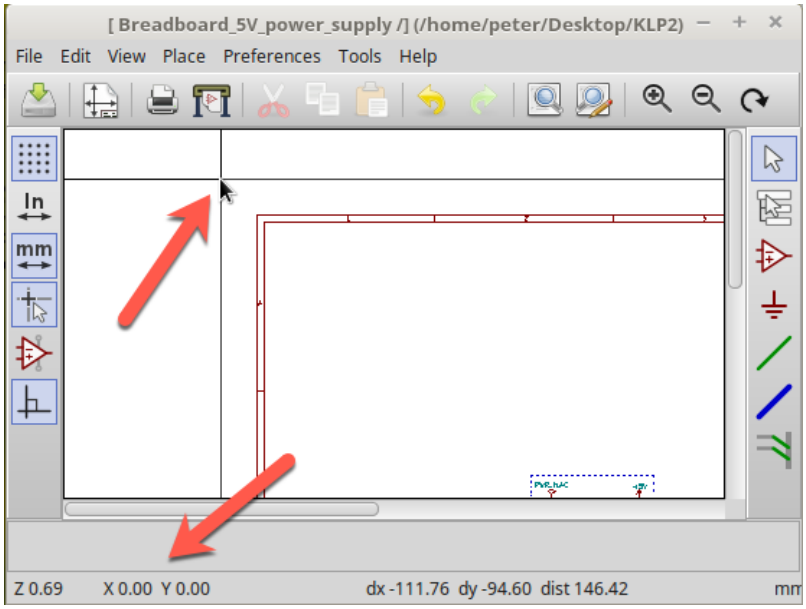


Figure 8.52: The status bar in Eeschema.

4: The relative coordinates, which allow you to measure the distance between two parts of the Sheet. To make a measurement, place the cursor on the location where you would like the measurement to start, and press the

spacebar to reset the *dx*, *dy*, and *list* counters. Then, move the cursor to the location where you would like to measure the distance to. The results of this measurement will be in the *dx* (difference in the x-axis), *dy* (difference in the y-axis) and *dist* (absolute distance between the two points) figures.

As an example, let's measure the distance between the two junction points in our resistor and LED schematic. Place your cursor on the left junction and press the spacebar to reset the counters (Figure 8.53).

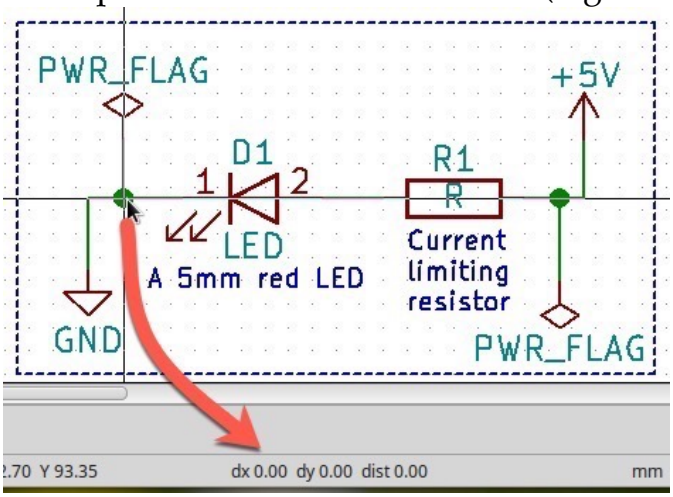


Figure 8.53: Reset the counters by pressing the space bar.

Then, simply move the cursor to the location you want to measure the distance to, let's say the junction on the right side of the schematic. In the example in Figure 8.54, we learn that the distance is 23.50 mm (*dist*), as is the difference on the x-axis (*dx*). The two junctions are on the same location on the y-axis, so the *dy* value is 0.

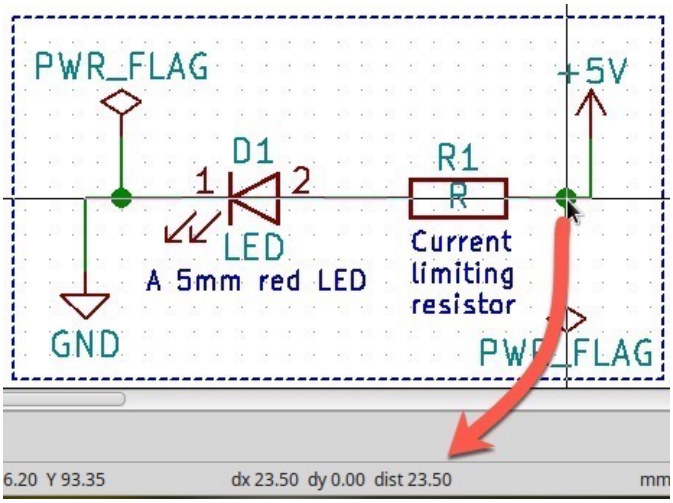


Figure 8.54: Move the mouse cursor to a new location and see the *dx*, *dy* and *dist* counters to update.

5: Show the selected unit of length. I find that the ability to measure distances is extremely important, not so much in Eeschema, but in Pcbnew. In Pcbnew these measurements will help us place a footprint that requires

precise physical dimensions. For example, we will use this measurement technique to design a PCB that will fit precisely in our project box, or that will have a screw terminal precisely placed to correspond with a provision for it on our box.

Spend a few minutes to become comfortable measuring distances in Eeschema. It is a skill you will also use in Pcbnew, the component library editor, and the footprint editor.

Menus

To complete our tour of Eeschema, let's have a quick look at the menus.

The first thing to notice is that these menus contain items that are also available as buttons in the top and right hand toolbars:

- File
- Edit
- View
- Place
- Tools

These two menus contain unique functionality:

- Preferences
- Help

The File menu contains the 'Append Schematic Sheet' item, which is not available as a button. Selecting this item allows you to insert the contents of another schematic sheet into the current sheet. This is useful in cases where you have, for example, a schematics that contains standard circuits, like power supplies and LED arrays. Instead of re-drawing those circuits, you can simply append them into your current design.

Let's have a look at the Preferences menu. We will not be making any changes there at this time since this is not necessary for the purposes of this small introductory project. But we will be working with the preferences in later projects, especially as you learn to create your own components, or import third-party component libraries (Figure 8.55.

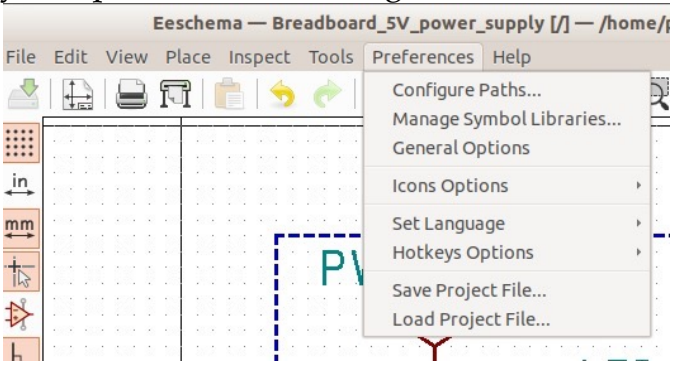


Figure 8.55: The preferences menu item.

From the Preferences menu, click on the Manage Symbol Libraries option. The Symbol Libraries window will appear (Figure 8.56).

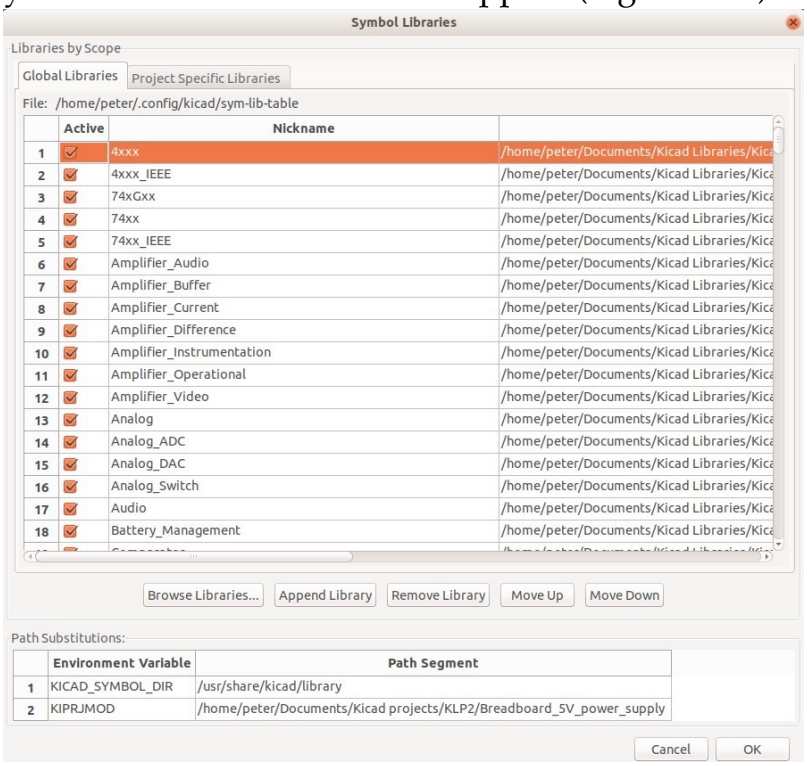


Figure 8.56: The Component Libraries window.

You can use the Symbol Libraries window to manage… symbol libraries. KiCad is endowed with a huge selection of libraries that its users create and share. You can find and install them on your computer using the functions in this window. I will show how to import and use a third-party components library in a more involved project in Section 4 and in the recipe titled '20. Adding a schematic symbol library in Eeschema'.

Also look at the General Options window. You are already familiar with the Display, Editing and Controls tabs. Have a look at the Colors tab (Figure 8.57).

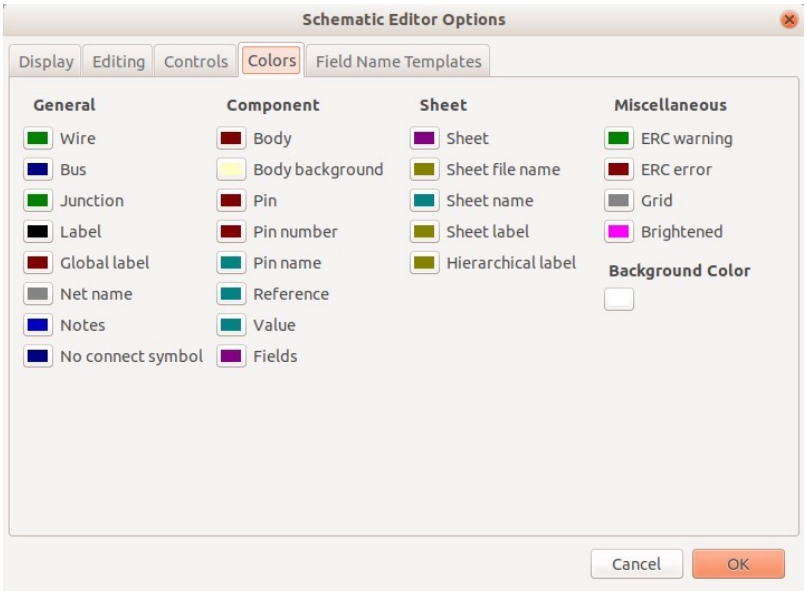


Figure 8.57: The Eeschema Colors Scheme window.

Through the Colors Scheme windows, you can customise the colours that Eeschema uses. You can change the colour of practically any widget and schematic attributes, like wires, junctions, and notes. Simply click on a colour box to bring up a colour chooser and replace the original colour.

In the 'Set Language' Preferences menu item, you can change the KiCad user interface language. KiCad has support for more than 10 languages, but beware that this feature will not work unless your operating system also supports the language you want to choose.

Continuing in the Preferences menu item, you will find the Hotkeys item. You learned about the hotkeys in a previous segment of this book. Here, I will only add that through this menu you can also export and import Hotkeys. This is useful if you use KiCad in multiple computers and you want to have the same Hotkeys configures across all of them.

At the bottom of the Preferences menu, you will find two functions that allow you to save and load your custom preferences to your project file.

9. Layout in Pcbnew

Pcbnew is KiCad's layout editor. Think of it as a drawing program that is specifically designed for drawing printed circuit boards. Pick up a PCB and, and try to identify and list its most important attributes.

Your list should be similar to this:

1. Components, like resistors and integrated circuits,
2. Wires (or traces, as we will call them), that connect the components,
3. A board made of fibreglass or similar material, with a particular shape,
4. Graphics, like text or drawings, printed on the board,
5. Holes, large or small.
6. Components, wires, and graphics may appear on one, or two sides of the board. Wires can also appear in many layers that are sandwiched between the top and bottom layer to allow for more complicated circuits.

We use the layout editor to design a PCB with all of these elements. With KiCad's Pcbnew, you can design PCBs that are very simple, like the one we are working on in this section, and extremely complicated and dense ones, like the one in Figure 9.1.

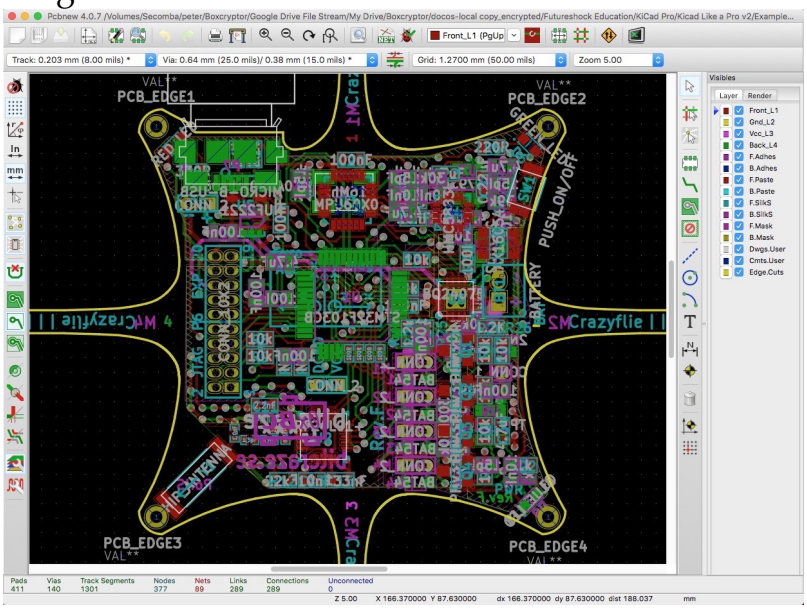


Figure 9.1: An example of a relatively dense layout in Pcbnew. It belongs to the Crazyflie project[4].

In the next few pages, you will take a hand-on tour of Pcbnew and learn about it most important features.

9.1. The user interface

Let's start Pcbnew. The objective is to become familiar with its user interface and the functionality that is absolutely necessary for our very simple first project.

To start Pcbnew, go back to the KiCad main application, and click on the Pcbnew button, as in the example in Figure 9.2.

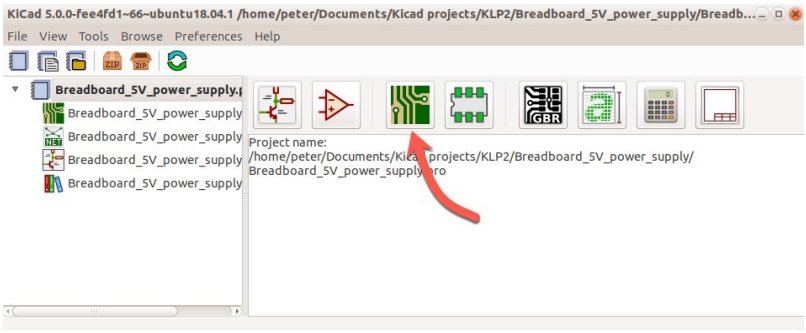


Figure 9.2: Start Pcbnew from the main KiCad application window.

You can also start Pcbnew through the menu. Click on Tools, and then 'Run Pcbnew', or through a keyboard shortcut (Ctrl-P on Windows and Linux, Cmd-P on Mac OS).

Pcbnew will start, and you will see the main window. It should look like the example in Figure 9.3.

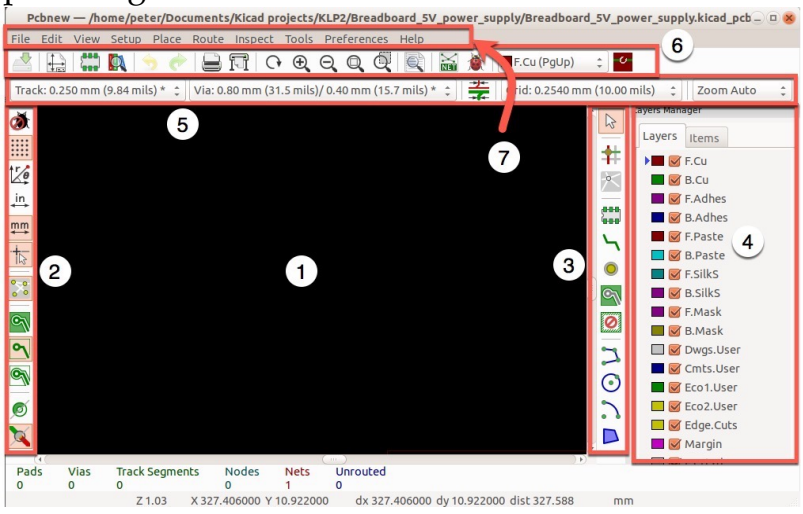


Figure 9.3: The Pcbnew main window, with its toolbars annotated.

[4] The Crazyflie is an open-source flying robotics platform. Visit the project website to learn about the amazing products and projects that are based on Crazyflie: https://www.bitcraze.io/crazyflie-2/

Compared to Eeschema, Pcbnew may seem more complex and even intimidating, but rest assured, it isn't. In this part of the book, I will help you to learn only the most important functionalities so that you can start designing your PCB very quickly. Just like we did in the Eeschema segment, we'll start with the basics of the layout sheet (item 1 in Figure 9.3), then learn about the mouse and hotkeys controls, and then the toolbars and menu bars (the rest of the items in Figure 9.3).

Let's start with the layout sheet.

9.2. The layout sheet

Similarly to Eeschema, in Pcbnew you design your PCBs inside the designated sheet area. It does look very similar to the schematic sheet. You can also edit the layout template in Pcbnew in the same way you did in Eeschema. That is, by editing the page layout description file using the Pl_editor application.

You can also load an existing custom layout exactly as you did in Eeschema. In Pcbnew, click on the File menu, then click on Page Settings. At the bottom right side of the window, you will see the 'Page layout description file' text field. Use the 'Browse' button to find and select the template you created earlier in this section.

While you have the Page Settings window open, let's fill in the form so that it looks like the example in Figure 9.4.

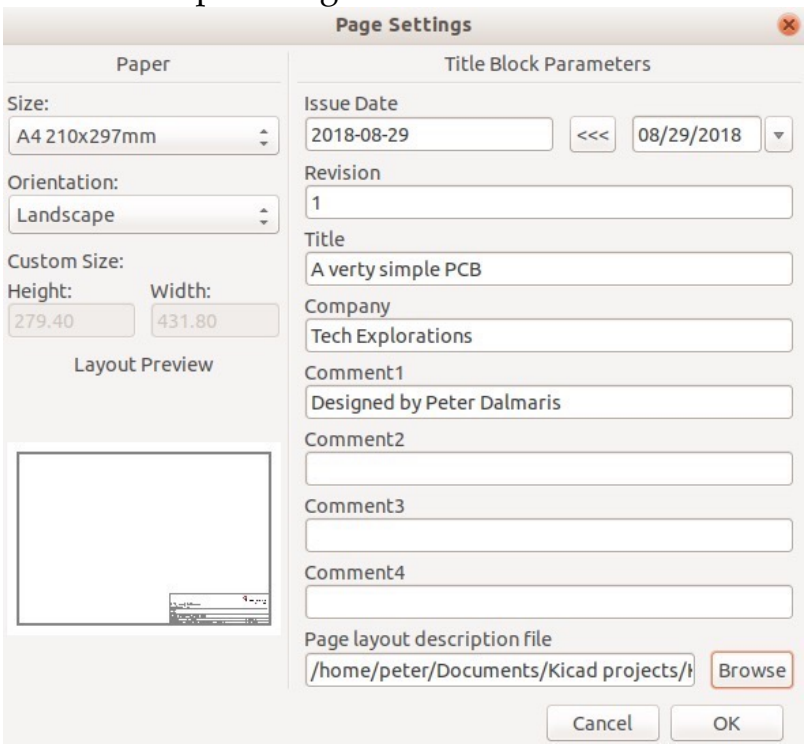


Figure 9.4: The Pcbnew Page Settings dialog box.

As you can see, the fields in the Pcbnew page settings window are identical to the ones in Eeschema. Click on the Ok button to commit the title block text, and confirm that your text appears in Pcbnew's title block (as in the example in Figure 9.5).

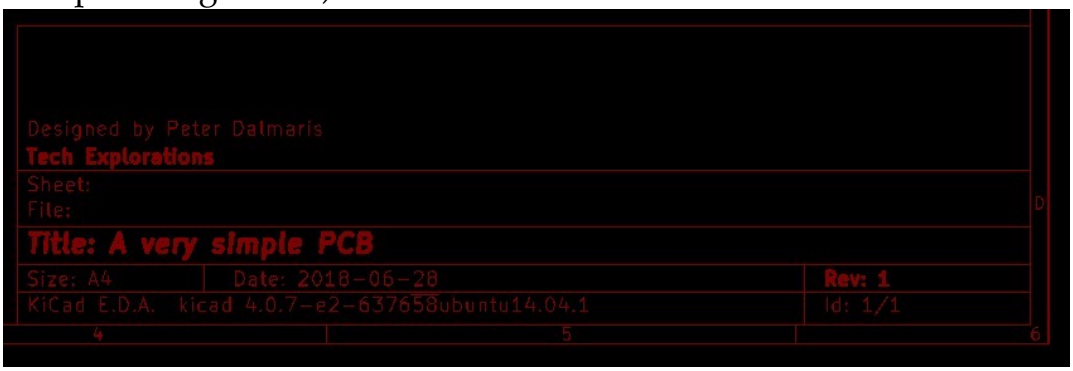


Figure 9.5: The Pcbnew title block.

The layout sheet uses a grid to help you place components on specific, evenly spaced, locations. You can control the size of the grid from the grid drop-down widget in the top toolbar.

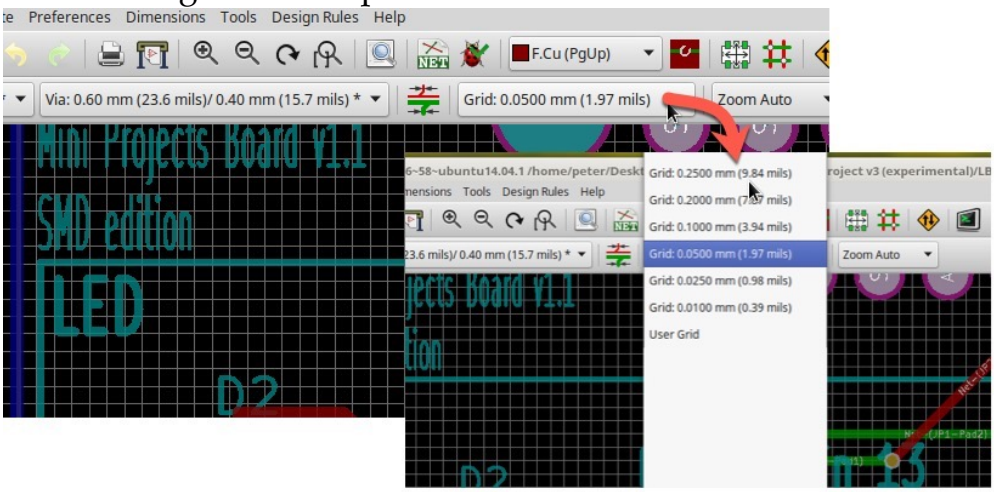


Figure 9.6: You can change the grid size from the Grid drop-down widget.

When you are working on loosely populated PCBs (like the one we are working on in this segment), you can opt for larger grids.

Another useful feature that you will use a lot, is called 'Zoom to Fit' (Figure 9.7).

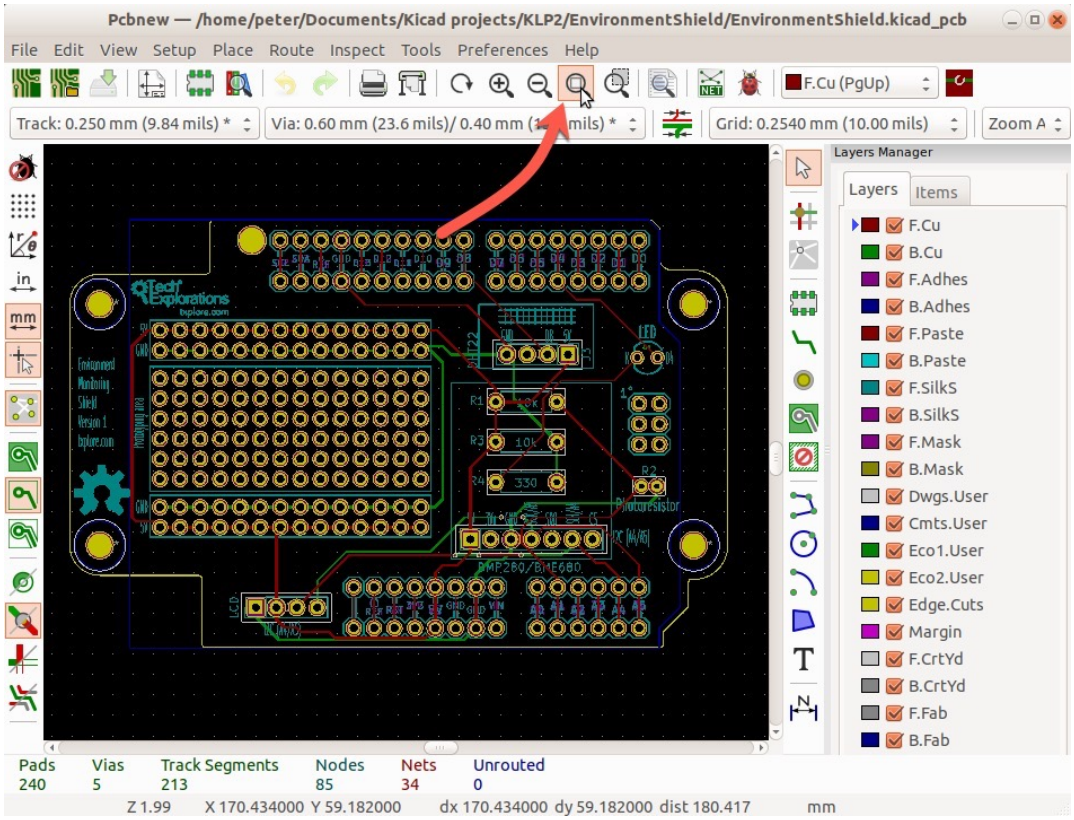


Figure 9.7: Zoom to fit changes the zoom level so that the entire PCB fits in the Pcbnew window.

When you click on the 'Zoom to fit' button, situated in the top toolbar, the zoom level of the sheet changes so that the complete PCB can fit within the Pcbnew window.

9.3. Mouse buttons and hotkeys

In Pcbnew, the mouse and hotkeys work almost exactly the same as they do in Eeschema. I use a mouse with two buttons and a scroll wheel. The scroll wheel is useful for zooming in and out of the sheet.

This table contains information about the default role of each button or button and key combinations:

Button, Keys	Role
Left mouse button (LMB), single click	Show selected footprint's details in the status bar
Left mouse button (LMB), double-click	Show footprints properties window for the selected footprint
Right mouse button (RMB)	Show the context menu for selection

Alt/Cmd + Right mouse button (RMB)	Pan (please note: this combination may be different for you, as it depends on how your mouse and keyboard are setup on your computer).
Scroll wheel	Zoom in/out at the position of the cursor

Table 9.1: Mouse controls in Pcbnew.

When you do a single left click on a footprint, the footprint's details are shown in the status bar, at the bottom of the Pcbnew window, like in the example in Figure 9.8.

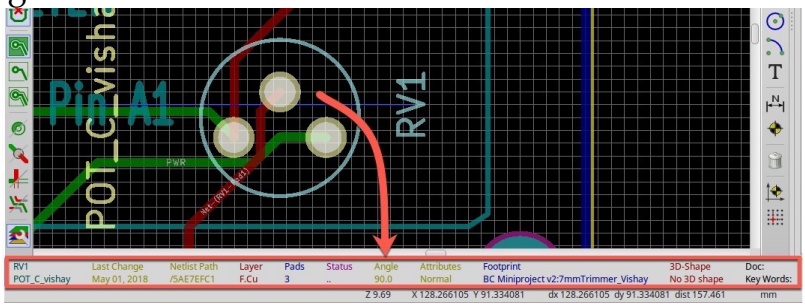


Figure 9.8: To see a footprint's details, click on it.

In the example of Figure 9.8, I have left-clicked on the footprints of a potentiometer ('RV1'). In response, Pcbnew shows me the details of this footprint in the status bar, in the area inside the red box. From this, you can see the number of pads included in this footprint (3), it's orientation (90 degrees from its default orientation), which library it belongs to ('BC Miniproject v2:7Trimmer_Vishay'), and more.

When you double left-click on a footprint, the Footprint Properties window for this footprint will come up.

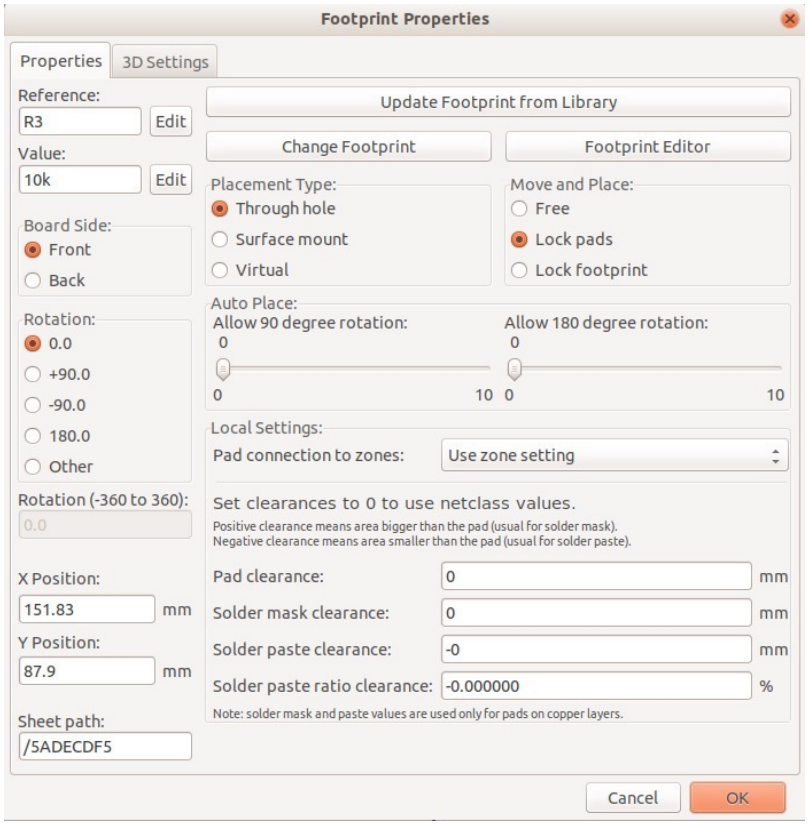


Figure 9.9: Double left-click on a footprint to open the footprint properties window.

In the footprint properties window, you can edit that footprint's properties, such as its reference designation, it's orientation, position, and much more. You can also change the association of the underlying component to a different footprint, or start the footprint editor so that you can edit the current footprint.

The right mouse button produces a contextual menu that depends on what you clicked on. For example, if you click on a footprint, you will get the context menu in Figure 9.10.

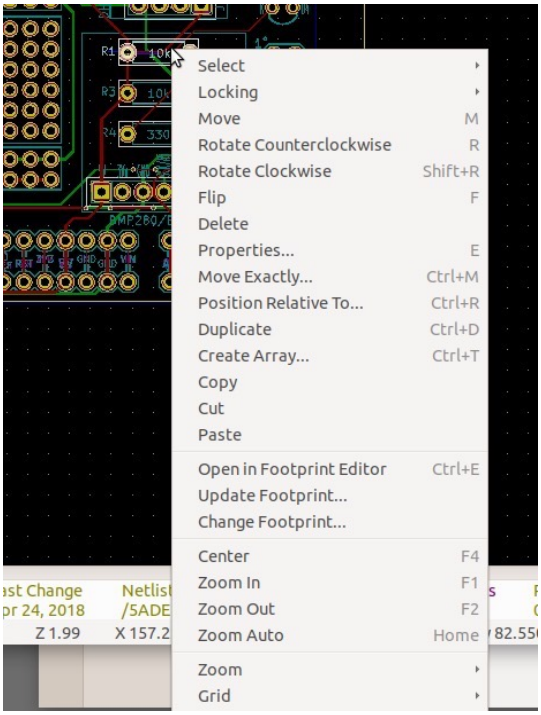


Figure 9.10: Clicking on a footprint produces this context menu.

Many of the functions in this menu are accessible through simple hotkeys, like 'E' for the Properties window, 'R' to Rotate, and 'M' to Move. These hotkeys should be familiar to you from Eeschema.

If you right-click on a track, you will get the context menu in Figure 9.11.

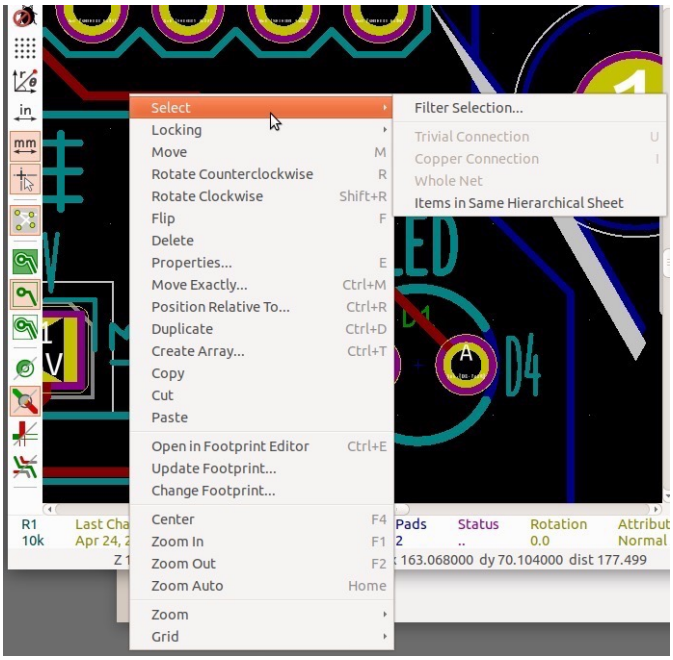


Figure 9.11: Clicking on a track produces this context menu.

Notice the there is very little difference between the two context menus. The track contextual menu contains the 'Select' options, which the footprint contextual menu does not. The footprint contextual menu, on the other hand,

contains the 'Open in Footprint Editor' option, which the track contextual menu does not. Don't worry about what these specific functions do, for now, as we will not need them until later in this book.

Finally, if you right-click in an empty area of the sheet, you will get the context menu in Figure 9.12.

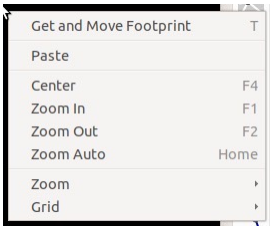


Figure 9.12: Clicking on an empty area of the sheet produces this context menu.

These are functions that allow you to reposition the sheet, zoom in and out, and change the grid size.

Hotkeys in Pcbnew are as important as they are in Eeschema. Most of the hotkeys that you are already familiar with from Eeschema will work in Pcbnew, but there are some differences, so it is worth spending a few minutes here.

With Pcbnew in focus, type '?' Or Alt-F1. You will see the Hotkeys List (Figure 9.13.

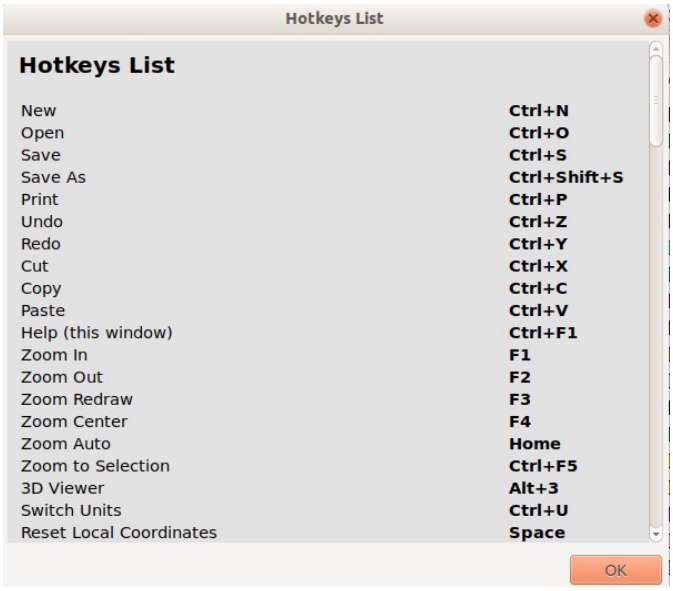


Figure 9.13: The Hotkeys List in Pcbnew.

There are a lot of hotkeys available, but for the purposes of your quick introduction to Pcbnew, only a few are necessary. To use a hotkey, simply place the pointer of your mouse over a footprint, track, text or another part of the layout sheet and type the hotkey.

Hotkey	Role

O	Add a new footprint to the sheet
E	Edit, reveals a dialog box where you can edit all footprint attributes
R	Rotate (each time you press 'R', the footprint will rotate by 90 degrees)
G	Drag, useful for repositioning a footprint without disconnecting any wires. At the time of this writing, the Drag feature is not working as expected; instead it operates as a regular Move.
M	Move. If the footprint has tracks connected, the wires will be disconnected
del	Delete track or footprint
C	Copy item
X	Start drawing a track
V	Add a via
T	Add graphics text, useful for providing information about the schematic
F3	Redraw the sheet, useful when you want to clear graphics artefacts

Table 9.1: A list of the most frequently used hotkeys in Pcbnew.

There are more hotkeys available in Pcbnew, but the ones listed in Table 9.1 are the most useful ones. Before long, you will commit them to memory and use them constantly.

If you wish to change some of those hotkeys, you can. The process is the same as the one you learned about in the Eeschema segment. Click on the Preferences menu item, then Hotkeys Options, and then Edit Hotkeys (Figure 9.14).

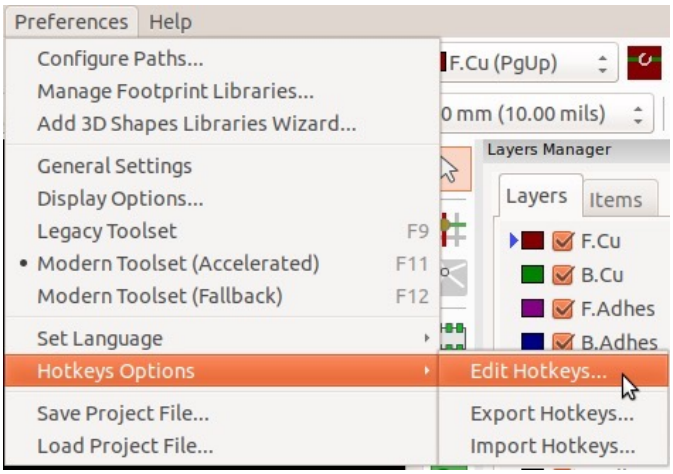


Figure 9.14: How to edit your hotkeys in Pcbnew.

This will reveal the Hotkey editor (Figure 9.15):

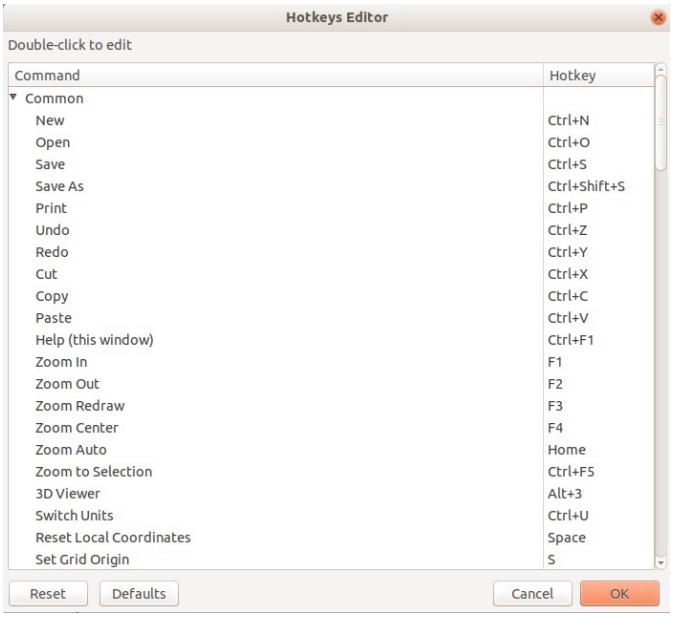


Figure 9.15: Select an existing Hotkey to highlight it, and then type in a new key or key combination.

To change a Hotkey, simply select it with your mouse and type in a new key or key combination.

Now that you know how to use your mouse and keyboard to control Pcbnew, let's move on to the next topic and learn about the toolbar functions.

9.4. Pcbnew toolbars and menus

In Pcbnew, the available functions and tools are grouped in toolbars, menus, and the status bar.

You can see the standard Pcbnew window in Figure 9.3. Following is a quick summary of the purpose of each bar. We'll look at some more details at later parts of this book.

The left toolbar, marked '2' in Figure 9.3, contains functionality for controlling the visual representation of the various elements of the PCB, as well as of the user interface. For example, you can turn on or off the display of rats nest lines. These lines show incomplete connections between pads. You can also show or hide the part of the right toolbar that allows you to switch to particular layers.

The right toolbar, marked '3' in Figure 9.3, contains the mains functions needed for editing the contents of the sheet. This is where you will find buttons for adding a new footprint, new tracks, and vias, text or graphics on your sheet.

Next to the right toolbar is the Layers manager, marked '4' in Figure 9.3. You can hide it by clicking on the Layers button in the left toolbar if you are working on a small monitor and you want to maximize the available space in

the sheet. The Layers manager allows you to select the board layer on which you want to work next. For example, if you want to work on the front layer, you will click on the 'F.cu' option to select it. If you want to work on the back layer, you will click on the 'B.cu' option.

At the top of the Pcbnew window are three groups of functions. Two toolbars, and the drop-down menus. Marked as '5' in Figure 9.3 is the Sizes toolbar. It allows you to select sizes for the tracks, vias and the sheet grid.

On top of it is the main functions toolbar, marked as '6' in Figure 9.3. This is where you will find functions for saving your work to a file, printing your layout on paper, exporting it for manufacturing, performing the design rules check (an equivalent of the Electrical Rules Check in Eeschema), and importing the Neltlist file that you created in Eeschema.

Finally, at the very top of the Pcbnew user interface, we have the drop-down menus. Many of the functions that are available through the toolbar buttons are also available through the menus and, of course, through hotkeys. But there are several functions that are only available through the menus. An example is the Preferences menu, where its contents are only available through it.

Let's have a closer look at the items that I described above, and learn the most frequently used functions by continuing the work we started in Eeschema. We'll design a very simple PCB for our resistor and LED circuit.

Left toolbar

In the left toolbar (Figure 9.16) there are several functions there are familiar from Eeschema. For example, buttons labeled 2, 4, 5, and 6 have the same functionality in Pcbnew as they have in Eeschema.

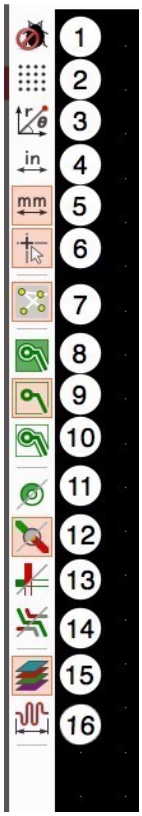


Figure 9.16: The left toolbar.

For now, let's not be concerned with buttons 1, and 7 to 16. These will make more sense when we are working on a more elaborate PCB so that we can make use of them.

Let's try out the grid button (2). Click on it to enable it. You should see a thin black box around it, indicating that the grid is enabled. White horizontal and vertical lines should be visible in the sheet if the grid size is of an appropriate size. In Figure 9.17, I have enabled the grid and set its size to 1 mm.

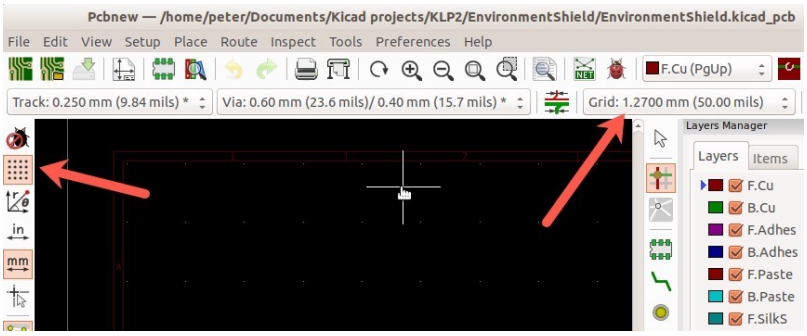


Figure 9.17: Enable the grid and adjust the grid size.

Button 3 allows you to change the relative coordinate system from Cartesian to Polar. I am more accustomed to the Cartesian system so I stay with the default. With the relative coordinate system, you can measure the

distance between two locations in the sheet. This is a very useful function in Pcbnew as we want to design very accurate boards that will match well with the components we plan to attach. In Figure 9.18 you can see the Cartesian and Polar coordinate values as they appear in the status bar.

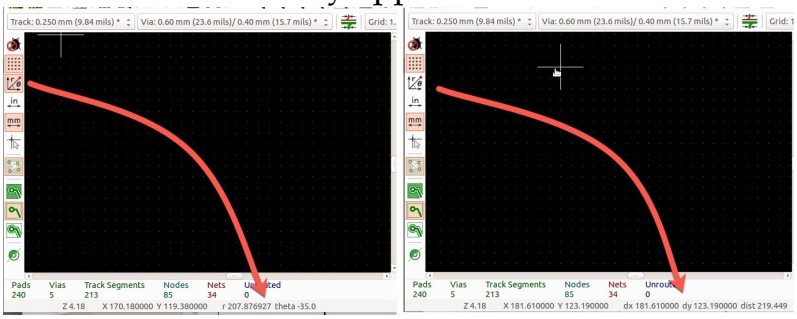


Figure 9.18: Cartesian (right) and Polar (left) coordinates.

Buttons 4 and 5 allow you to switch the length measuring units to millimetres or inches.

Button 6 allows you to change the shape of the cursor to be either a simple pointer or a pointer with crosshairs that reach the sides of the Pcbnew window, as you can see in Figure 9.19.

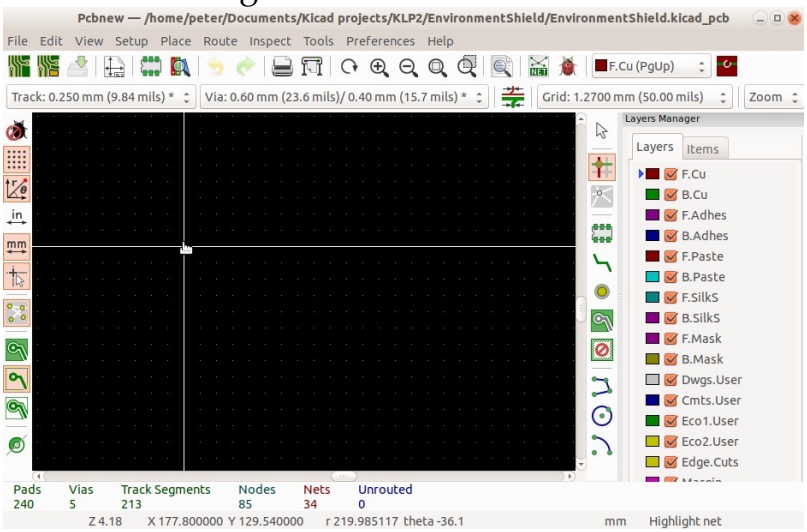


Figure 9.19: Pointer type changed to crosshairs.

In the next segment, we'll have a look at the top toolbar. You will learn how to use the Netlist button to import the schematic from Eeschema so that you can start working on the layout for your first PCB.

Top toolbar

The top toolbar contains several groups of buttons that include principal functions in Pcbnew. These functions allow you to save the layout to disc, print it to paper or to a format that is supported by PCB manufacturers, select a layer in which to continue work, import a netlist, and more.

In Figure 9.20 you can see an example of the top toolbar.

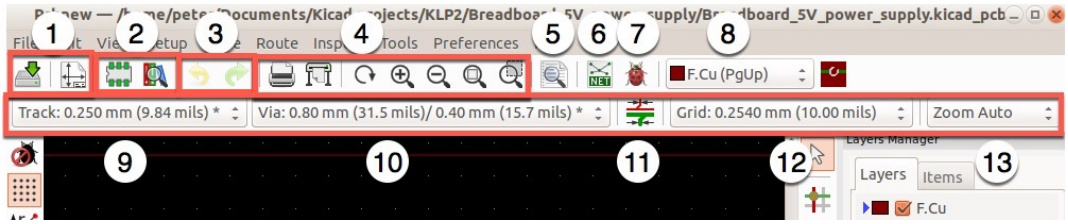


Figure 9.20: The top toolbar in Pcbnew.

As you can see, there is a lot going on here. Some of the buttons should be familiar to you; you saw them in Eeschema. With reference to the numbers in Figure 9.20, buttons in groups 1, 3 and 4 work the same in Pcbnew as they do in Eeschema.

Let's explore some of the most commonly used functions of the toolbar by continuing with the simple project we started in the Eeschema segment.

Import the Netlist

The layout work begins by importing the Netlist file. I remind you that this file contains the information that Pcbnew needs in order to import the necessary footprints and setup the ratsnests in the layout sheet. The ratsnests are thin lines that connect the footprint pins that must be electrically connected.

Let's do this now. Click on the Netlist button, marked '6' in Figure 9.20.

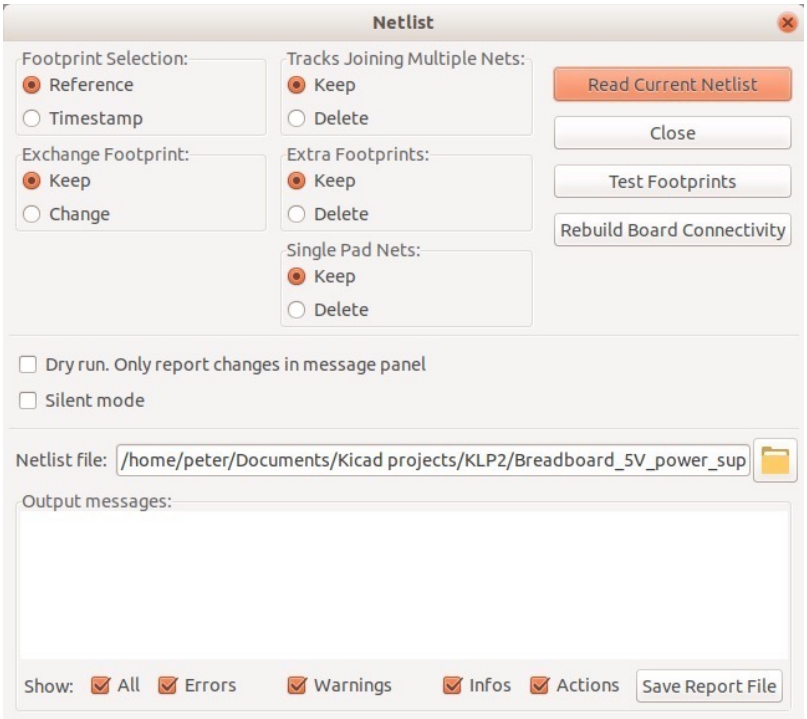


Figure 9.21: The Netlist window.

The Netlist window will appear. There are a lot of options that allow you to fine-tune the behaviour of the Netlist import function. As we are importing a Netlist to an empty sheet, the actual setting don't matter. However, imagine that you are importing a Netlist to a layout that already has footprints in it. Perhaps you had to go back to Eeschema to make updates and corrections, which resulted in a new Netlist file. How would you like Pcbnew to handle components that have a new footprint? Or a footprint that exists in the sheet but not in the Netlist you are about to import? These are things we can control by making appropriate choices in the Netlist window.

In this example, we will accept the default options and let the import proceed. Notice that in the 'Netlist File', the Netlist file we created in Eeschema is already listed. If it isn't, click on the Browse button to find it and select it.

Next, click on the 'Read Current Netlist' button. Pcbnew will read the file and show you the results in the Messages text area. The window will not disappear. Move it a little to the side, and notice that the circuit's footprints are now in the layout sheet.

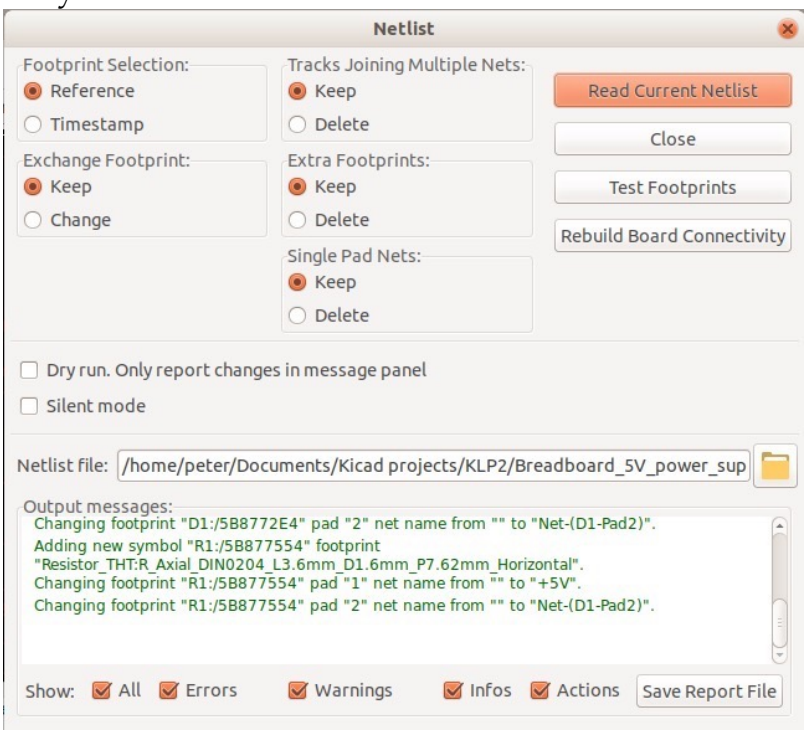


Figure 9.22: Reading the Netlist file will place the footprints to the sheet.

Click on the 'Close' button to dismiss the window. Let's concentrate on the footprints that we just added to the layout sheet.

Move footprints

The two footprints should be attached to your mouse pointer. Click anywhere inside the sheet to place them.

Place your cursor on the footprint bundle and use the scroll wheel to zoom in.

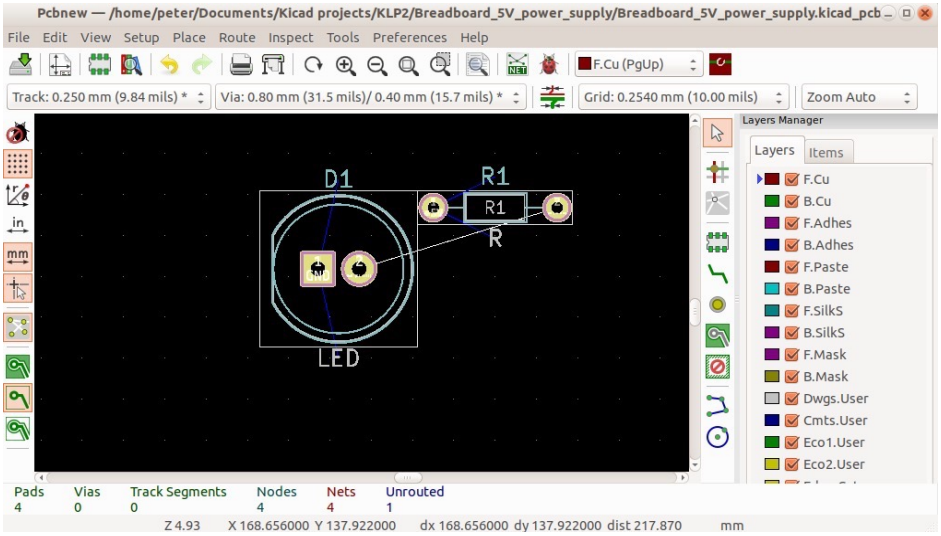


Figure 9.23: Zoom into the footprint bundle.

Let's separate the footprints manually. Use the 'M' hotkey to move one of the footprints away from the other. When footprints and other items are overlapping, the exact location where you mouse course is at may be occupied by more than one of them. When that happens, Pcbnew does not know which item you want to operate on. It will produce a context menu asking you to clarify.

It looks like in the example in Figure 9.24, where I placed the mouse pointer on a location where three items 'exist', a pad, and the two footprints. Then, I typed 'M', for 'Move'. I am working at zoom level 7.33 and grid size 0.25 mm.

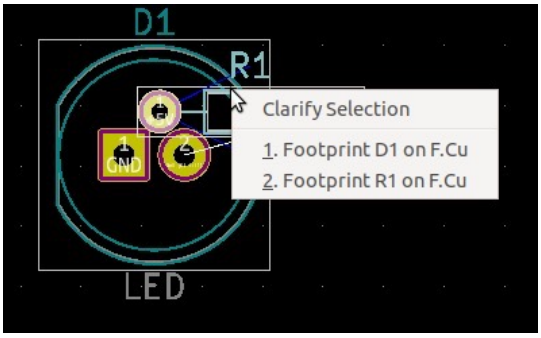


Figure 9.24: A context menu asking for clarification of item selection.

You can click and select one of the footprints. This will attach the footprint to your mouse pointer. You can then move it, and do a left-click to place the footprints on its new position.

Alternatively, because the LED footprint is larger than the resistor, it is an easier target. Place your cursor on it, taking care to be clear of the resistor, and again type 'M'. In Figure 9.25, you can see how I moved the LED footprint to the left of the resistor.

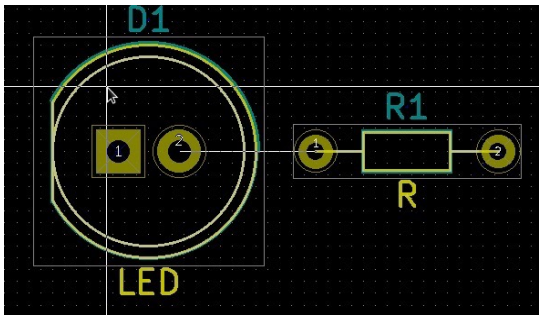


Figure 9.25: The two footprints are not overlapping.

Let's rotate the resistor by 90 degrees. Place your mouse pointer over the resistor footprint, and type 'R', for 'Rotate'. You will also need to use 'M' on the resistor to move it a bit to the right, as in the example in Figure 9.26.

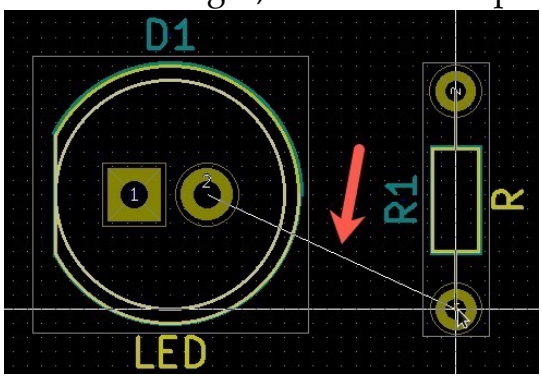


Figure 9.26: Move the resistor by 90 degrees. The arrow shows the ratsnest line.

In Figure 9.26 you can also see the single ratnest line, which indicates that we must create a track that connects pad 2 of the LED footprint (D1) with pad 1 of the resistor footprint (R1).

Design Rules Check (DRC)

At the moment, we have the two footprints in the sheet, placed at locations where I think are appropriate for this project. We will need to use some of the tools in the right toolbar, but before we get to it, I'd like to introduce a few more of the functions from the top toolbar.

One of the most important functions is the Design Rules Check (DRC) which you can access via the button labeled '7' in Figure 9.20. The DRC is similar to the ERC in Eeschema. It checks your layout for errors.

The most common errors that the DRC can find are things like:

- two tracks are too close to each other
- no tracks connecting pads that should be connected

- a track is too close to a pad
- a track is to close to the edge of the board
- a track is too close to a via
- a via is too close to a pad

There are more, of course. The details of things such as 'when are two tracks too close to each other' and ' when is a track too close to a pad', are all configurable by editing the Design Rules for the project. You can do that by calling the Design Rules Editor from the Pcbnew main menu item 'Setup' (item '7' in 9.3). The Design Rules Editor looks like the example in Figure 9.27.

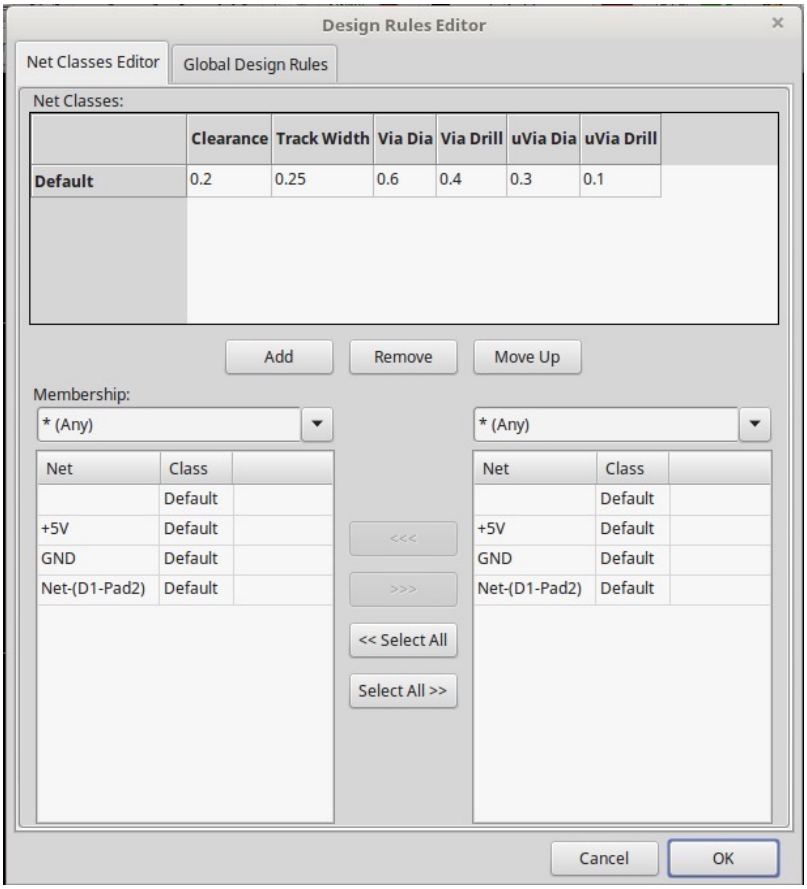


Figure 9.27: The Design Rules Editor in Pcbnew.

In Figure 9.27 you can see the default design rules. All the values are in millimeters. As per these rules, tracks must be 0.2mm away from other tracks, vias, and pads, or more. The width of a track must be 0.25mm or more, and so on. You can also see the default global design rules by clicking on the 'Global Design Rules' tab. For the purposes of this simple project, we will work with these default rules. In later projects, you will learn how to make modifications to the design rules. Let's click on the 'Cancel' button to exit this dialog box.

Next, let's do a DRC now, even though we know that our layout is not complete. I'd like you to see what a DRC error looks like, and learn to

interpret it. Click on the DRC button, and the window depicted in Figure 9.28 should appear.

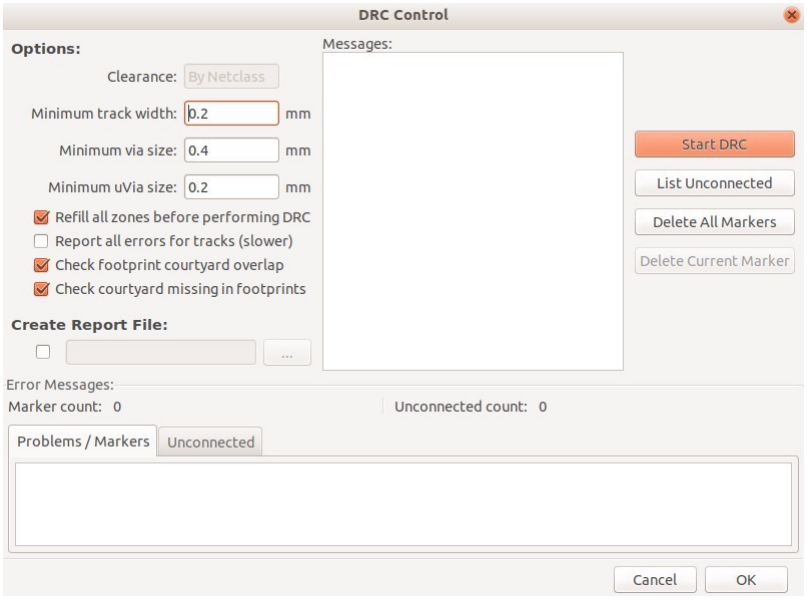


Figure 9.28: The DRC Control window.

On the left side of the DRC Control window, notice that there are several options, such as 'Min track width' and 'Min via size'. These values are copied from the Global Design Rules settings. You can confirm that by clicking on the 'Global Design Rules' tab in the Design Rules Editor (Figure 9.27). The DRC Control window allows you to change these values to you can try the DRC with custom values.

For example, let's say that the default values case the DRC to report an error because a track to closer than 0.2mm to a pad. If you know that your PCB manufacturer can accommodate clearances of 0.1mm, then you can use '0.1' as a custom value in DRC Control to see if the error still comes up. If not, then you can continue with your work, knowing that your manufacturer will be able to produce the board. But if the error comes up again, you will know that you have to move the track further away from the pad.

To start the check, click on the 'Start DRC' button. The result should be what you see in Figure 9.29.

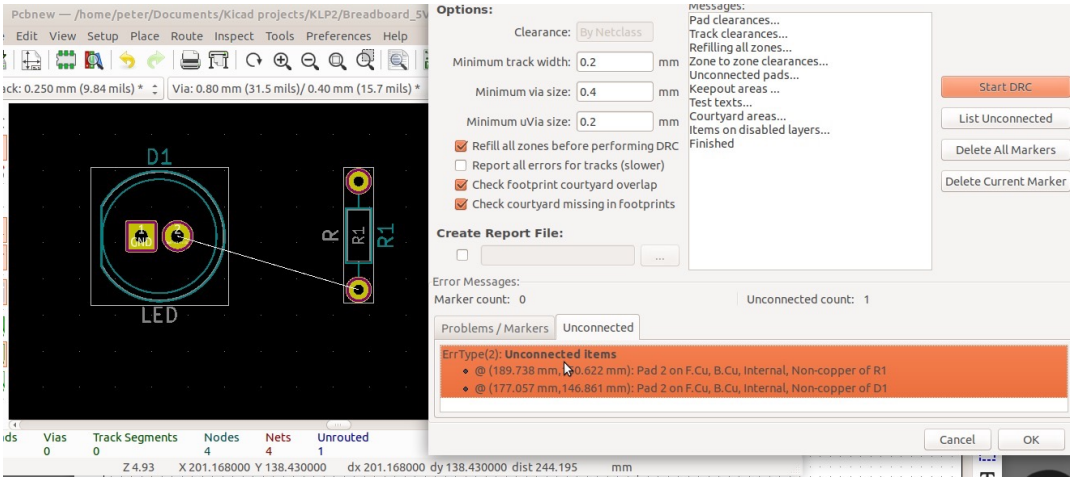


Figure 9.29: The DRC reveals that two pads are not connected.

The DRC reveals no problems in the Problems/Markers tab, but show two unconnected pads in the Unconnected tab. If you click on the error message, it will automatically zoom and pan into the location of the error. This error is easy to understand: Pad 1 of the R1 footprint on the front copper layer, which also exists in the back copper layer (since this is a through-hole component), should be connected to Pad 2 of the D1 footprint.

Let's fix this. We will need to use the Wire tool that belongs to the right toolbar. In Figure 9.30, the arrow points to the Wire tool. You can also activate the Wire tool by typing the 'X' hotkey, which is how I prefer to do it.

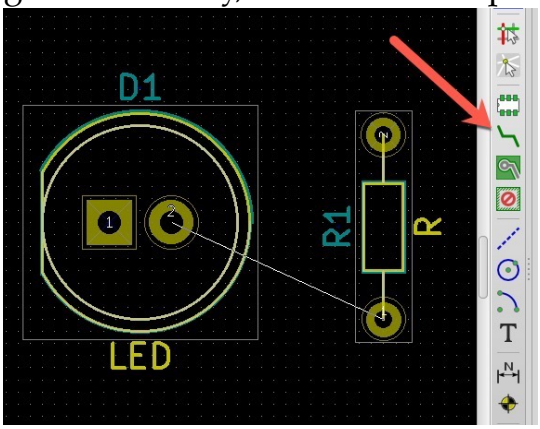


Figure 9.30: The Wire tool.

Place your mouse pointer over pad 1 of the resistor, and then type 'X'. With the wire tool selected, mouse your mouse slightly to the left, and notice that a thick red line that connects pad 1 of the resistor with the mouse cursor (Figure 9.31).

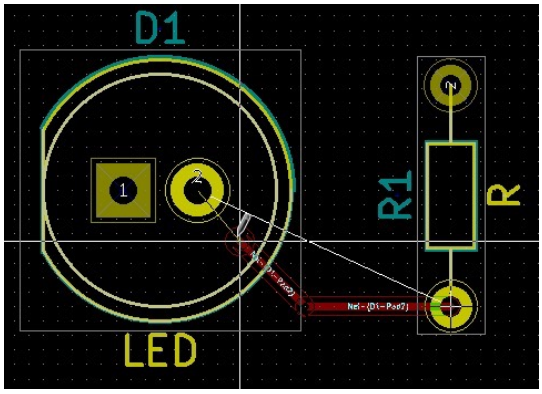


Figure 9.31: Drawing a new track.

As you are drawing the track, notice that the ratsnest line that connects pads 1 and 2 is still showing. This indicates that the electrical connection is not complete. Place your cursor in the middle of pad 2 of D1, and double click to complete the new track.

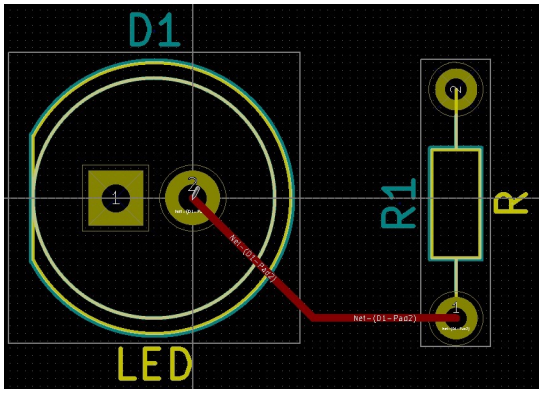


Figure 9.32: Completed drawing a new track.

You now have a completed electrical connection between the two pads, and the ratsnest line has disappeared. A solid red line with two segments represents the new track. When you manufacture this board, this red line will be implemented with copper.

The line of the track is important. By default, red refers to the top copper layer, and green refers to the bottom copper layer. You can see which colours represent the various PCB layers by inspecting the 'Visibles' panel on the right side of the Pcbnew window. Figure 9.33 shows the first few items in the list, which includes the top and bottom copper layers.

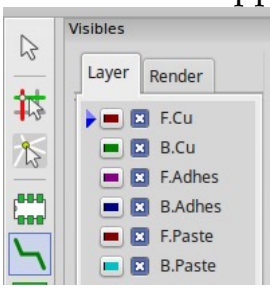


Figure 9.33: Each layer has its own colour.

With the only unconnected pair of pads connected, there should be no more DRC errors. Repeat the DRC to make sure. It should come back clean, with no errors in the Problems/Markers and Unconnected tabs.

There is still more work to be done. At the very least, we must add a connector so that we can connect this simple circuit to a battery. We also need to mark the borders of the PCB so that Pcbnew, and eventually the manufacturer, know where to cut the board. But before we get to that, let's have a look at the 3-D representation of our work so far. In the main menu, click on View, and then click on '3D Viewer' (Figure 9.34).

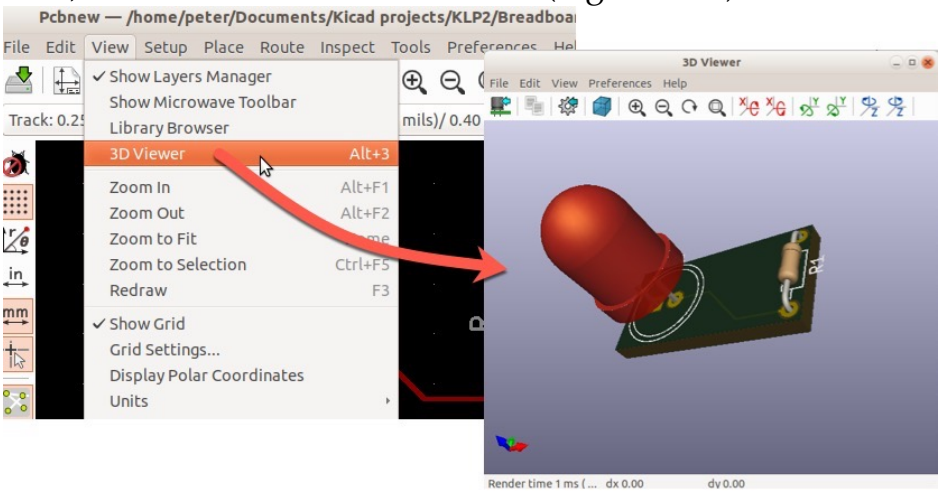


Figure 9.34: a 3-D representation of our incomplete board.

In the 3D viewer, you can manipulate the model using your mouse. Zoom in and out, pan and rotate. You can get a very accurate preview of what the PCB will look like if you went ahead to manufacture it. Many footprints, like the LED and the resistor, have 3D models that are used to render this view. If a footprint does not have a 3D model, only its pads will be shown.

Plot for Gerbers

While we are still on the top toolbar, let's have a quick look at the Gerber plotter and the layer chooser. You can access the Gerber plotter by clicking on the button with the plotter icon in the group of buttons marked '4' in Figure 9.20. In Figure 9.35 you can see the Plot window.

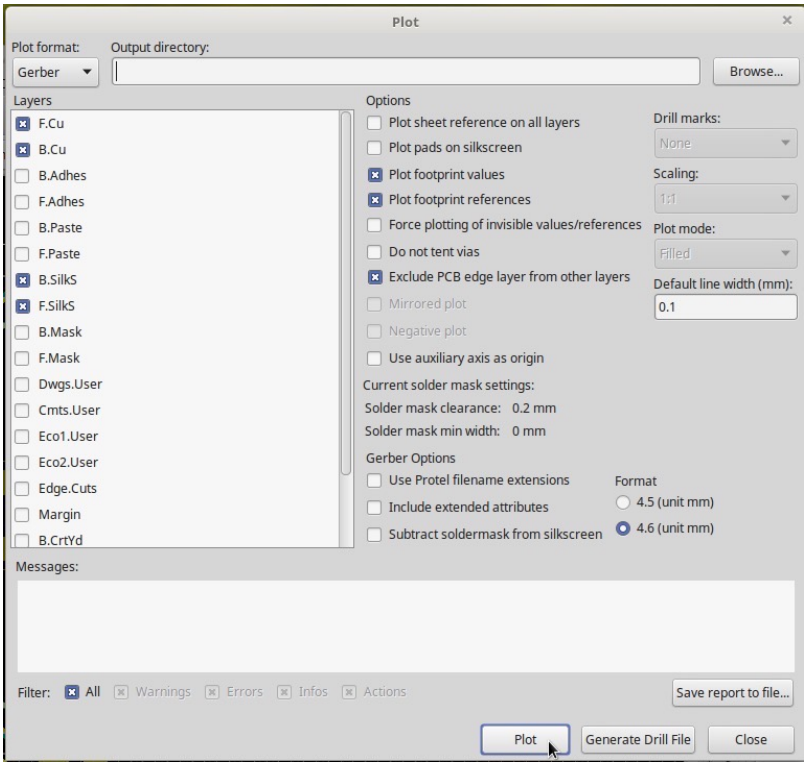


Figure 9.35: The Plot window from where you create the Gerber files.

When your layout work is complete and you are ready to send it to a manufacturer, you will need to create several files that contain the necessary board information. The Plot window is where this operation takes place. Several files are needed because a PCB is the combination of multiple layers and holes. You already know about the top and bottom copper layers, but there are more. For example, there is the layer on which the graphics are printed, both on the top (front silkscreen, or 'F.Silk') and the back (back silkscreen, or 'B.Silk'). There is also the solder mask, front, and back ('F.Mask' and 'B.Mask' respectively). The solder mask is a polymer material that is applied over the copper traces to prevent short circuits and protect against oxidisation. It also makes it easier to solder components onto the board by preventing the solder from spreading away from the pads and onto the tracks.

Each of the layers that your PCB uses is described by its own file. In the Plot window, you will select which files you'd like to create and then click on the Plot button to have them created.

Most PCBs also have holes, used to implement things like vias, pads for through-hole components, and opening for mounting screws. There is a separate file that you must create that will contains information about these holes, called the 'Drill File'. To create the Drill File, click on the 'Generate Drill File' button, which is right next to the Plot button. This will bring up the Drill Files Generation window, that you can see in Figure 9.36.

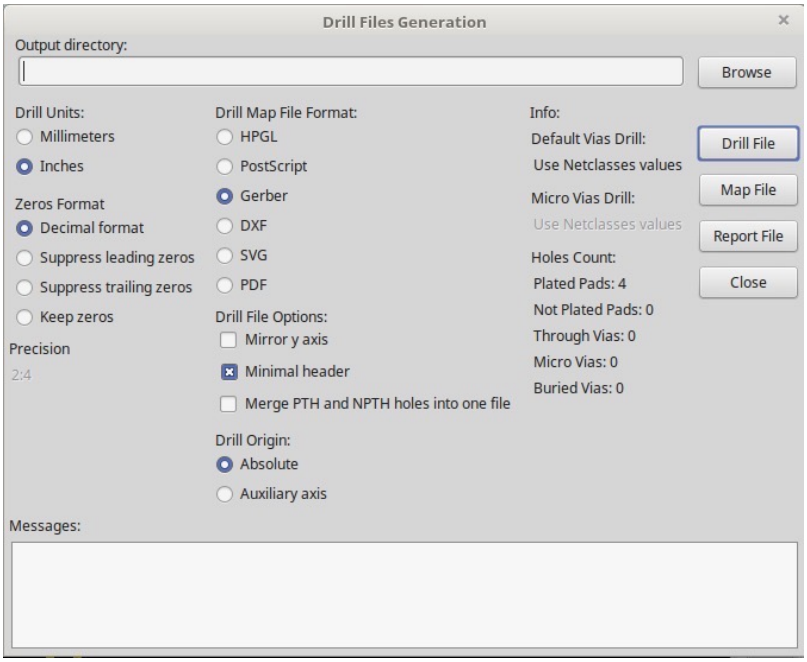


Figure 9.36: The Drill Files Generation window from where you create the Drill File.

Pcbnew can explore files in several different formats, but every manufacturer I have ever used can work with Gerber-formatted files, so we will use this type in this book. By default, Gerber is selected for the Drill Files and the Plot files.

The exact options that you should select depend on the expectations of your PCB manufacturer. For example, OSHPark has a guide to help you with the Gerber export step. OSHPark also allows you to submit the Pcbnew file with the .KiCad_pcb extension, and avoid having to create the Gerber files altogether, which is a welcome convenience. Pcbway.com also has a guide that details the Gerber file export process from KiCad.

I will show you how to do both in the next project, in which we will design and produce a PCB.

Layer chooser

The last top-menu item that I'd like to introduce is the Layer Chooser. It is labeled '8' in 9.20. You can see it extended in Figure 9.37.

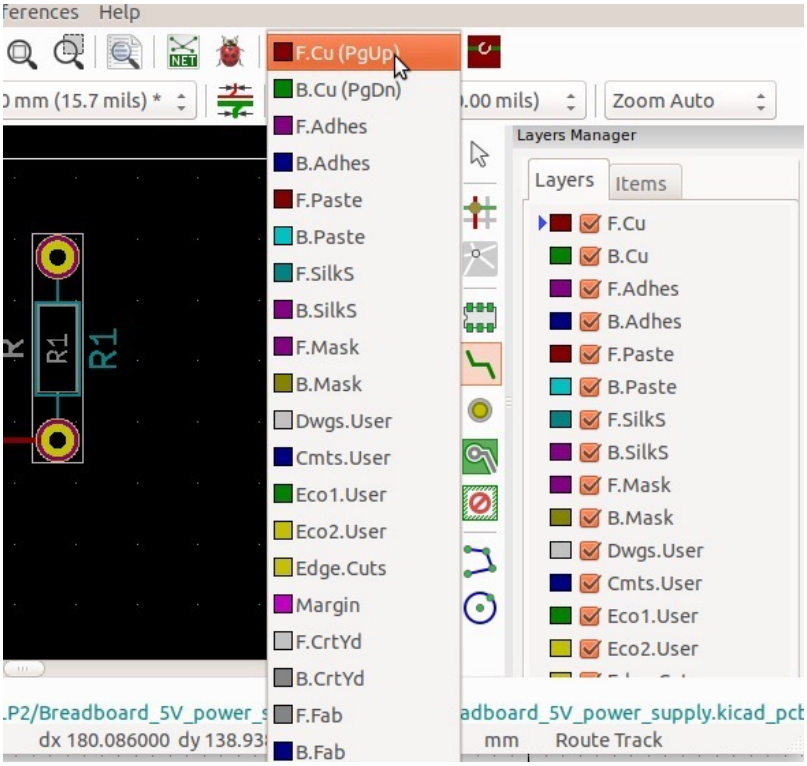


Figure 9.37: The top-menu dropdown allows you to switch between layers.

The Layer chooser in the top toolbar contains the exact same items as the Layer tab in the Visible pane next to the right toolbar. It allows you to switch between layers.

Let's do a quick experiment. Let's delete the only track so far, and redraw it, but this time in the bottom layer.

First, the delete part. Place your mouse cursor over the red track. Try to avoid any areas where the track overlaps something else, like a pad. If you do, Pcbnew will ask you to clarify your selection with the context menus you saw previously. With your mouse over the track, hit the 'delete' key on your keyboard. The track will disappear, and the ratsnest will appear in its place.

Next, click on the Layer Chooser to open it, and click on B.Cu (Back Copper) to select it. Notice the small triangle pointing to 'B.Cu' in the Layer tab of the Visibles pane (Figure 9.38).

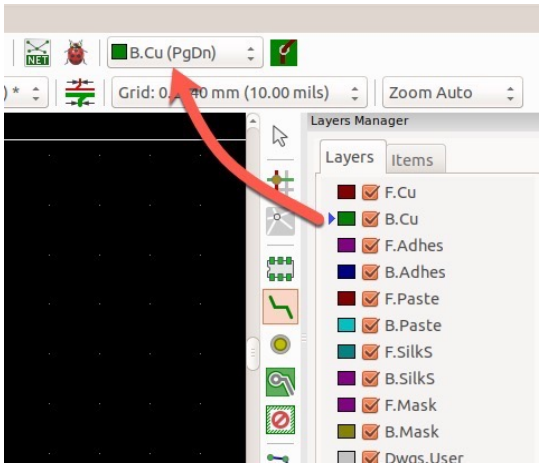


Figure 9.38: The back copper layer is selected.

Now, let's re-draw the track. Place your mouse over one of the pads indicated by the ratsnest, and type 'X' to start drawing. Complete the drawing by double left-clicking on the other pad. Your layout should look like the example in Figure 9.39. The track is now green instead of red since it is in the bottom copper layer.

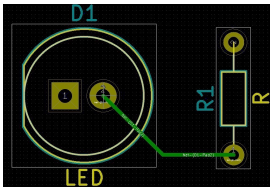


Figure 9.39: A new track in the back copper layer.

With two-layer boards, it does not matter much whether a track is placed on the top or the bottom layers. These days, single and double layer boards bring no material differences in cost or manufacturing processes.

We will discuss PCB design principles and considerations later, and learn good common practices for choosing positions for footprints and tracks, but for now let's continue to focus on Pcbnew.

Right toolbar

The right toolbar is where you will find the various drawing tools. With these tools, you can add and remove footprints, graphics, text, dimensions and, of course, tracks to the layout sheet. In Figure 9.40 you can see the right toolbar annotated with numbers to make reference easier.

Figure 9.40: The right toolbar.

Standard mode

The button, marked as '1' in Figure 9.40, allows you to the standard drawing mode. Most likely, you will enter the standard mode by hitting the ESC key. In Standard mode, you can click and select footprints and other items, right-click to reveal the context menu and measure distances between two points in the sheet.

Try this now: Hit the ESC key to enter standard mode, and click on the LED footprint. You will see that the footprints properties appear in the status bar, at the bottom of the Pcbnew window.

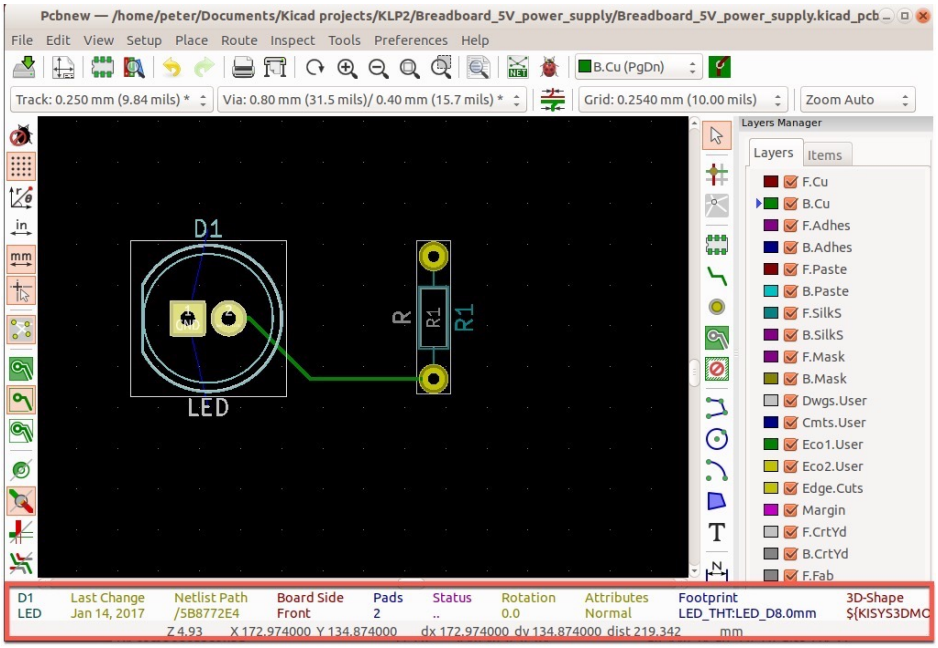


Figure 9.41: Click on an item while in the standard mode to display the item's properties.

In Figure 9.41, I have clicked on the LED footprint. In the status bar, I can see that this footprint designator is 'D1', it is on the front copper layer ('F.Cu'), it has two pads, and more.

Try clicking on the green track, that connects the LED to the resistor. The status bar will tell you that the name of the net which is implemented by the track is 'Net-(D1-Pad2)', and that it is drawn in the back copper layer ('B.Cu').

Net highlighter

The next button in the right toolbar, marked '2', is the net highlighter. While our current PCB is minimal, with a single net[5], you will quickly find yourself working on PCBs with many nets. In Figure 9.42 you can see an example PCB that contains several more nets than our current PCB does.

[5] I remind you that a 'net' describes an electrical connection in Eeschema and in Pcbnew. A track is the implementation of this connection. The track retains the name of the net.

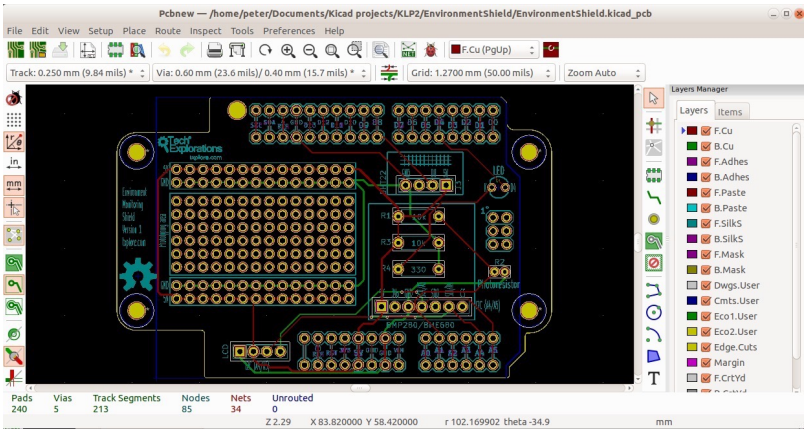


Figure 9.42: A PCB with several nets.

With only a glance at this board, it is hard to determine which tracks belong to, say, the 5V track or the GND track. But with the help of the Net Highlighter tool, this task becomes much easier. Click on the Net Highlighter button.

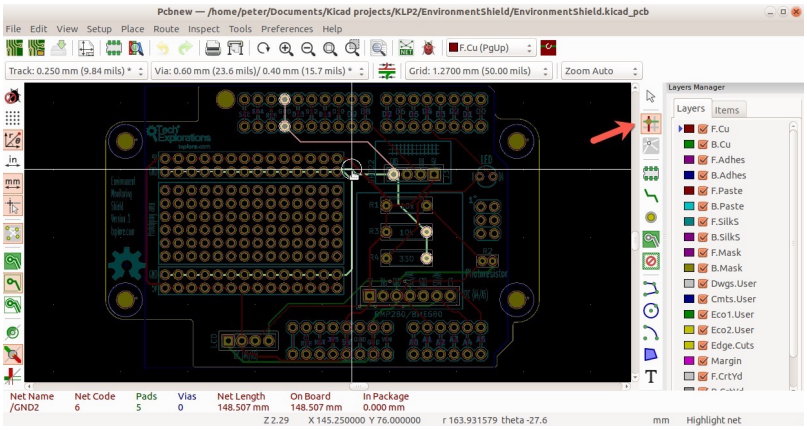


Figure 9.43: The Net Highlighter makes it easy to see tracks among other tracks.

In Figure 9.43 you can see the highlighted green track, with the net name 'GND2' running from the bottom left of the image to around the middle right side. Notice that the Net Highlighter tool is enabled.

Add footprint

When you imported the Netlist file into Pcbnew, Pcbnew automatically added the specified footprints into the sheet. If you want to add additional footprints, you can do so using the Add Footprint button, marked as '4' in Figure 9.40.

In our simple PCB, there is currently no provision for an external power source that will provide the power needed to light up the LED. It would be good to be able to add a simple connector for this purpose.

Let's go ahead and do this now.

First, you need to decide what kind of connector to add to your board. You can choose between something very simple, like two large pads on which you can solder the power source wires. You can also choose a barrel connector plug, a header connector like the ones used on the Arduino, or a screw terminal. Of course, there are many more options.

Let's assume that you have chosen a screw terminal, like the one in Figure 9.44 for the convenience of being able to securely attach the wires that come from a small AA battery pack.

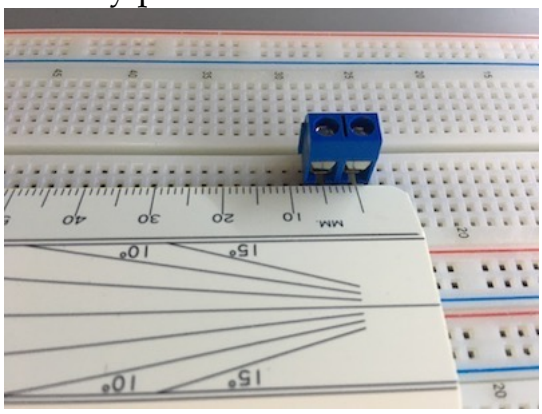


Figure 9.44: We'll add this screw terminal to our project layout.

To add a new footprint, click on the Add Footprint button, or type the 'O' hotkey. If you use the hotkey, first place your cursor on the approximate location where you would like to attach the new footprint. The Load Footprint dialog box will appear, as in Figure 9.45.

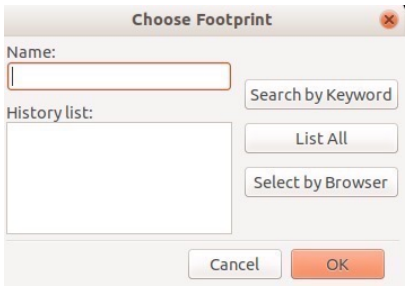


Figure 9.45: The Load Footprint dialog box.

This is not very useful. We don't know or remember the name of the screw terminal footprint, and the history list is empty. The easiest way to find a footprint when you don't know exactly what you are looking for is to use the footprint browser.

Click on the button marked 'Select by Browser' to reveal the footprint browser window (Figure 9.46).

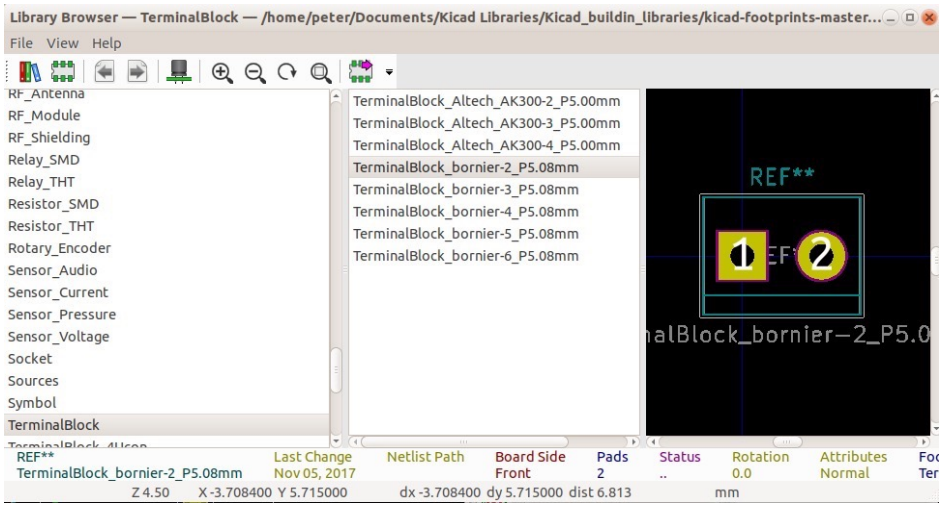


Figure 9.46: The footprint browser.

The footprint browser contains three panes. The one in the left lists the available libraries; the one in the middle lists the footprints that belong to the selected library; and the one in the right shows a visual representation of the selected footprint. Take some time to browse around the various libraries and footprints.

The footprint browser also gives you access to the 3D viewer. The 3D viewer gives you a 3D rendering of what the selected footprint looks like on a 'virtual' PCB. To use it, click on the button with the integrated circuit icon, as in the example of Figure 9.47.

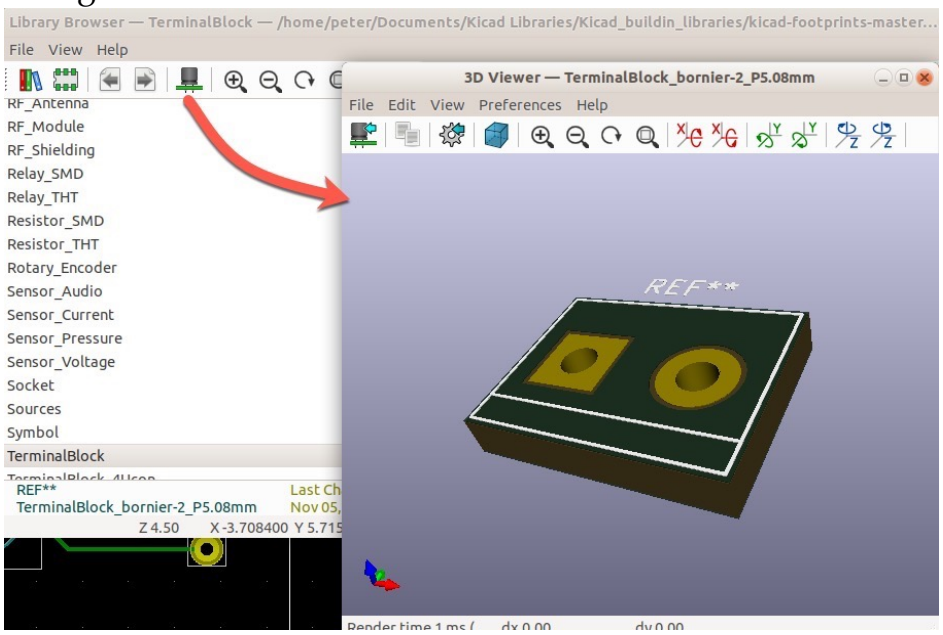


Figure 9.47: The footprint browser.

A good candidate for the 2-pole screw terminal we are looking for is inside the Terminal_Blocks library. There are several footprints in this library, and we want the one with the 2 pads, and 5.08 mm distance between the

centres of the pads. The distance between the pads is conveniently provided as part of the name of the footprint, but that is not always the case. I have cultivated the habit of measuring a footprint to confirm it will match the real-world part before I drop it to the layout sheet.

First, notice that in Figure 9.44, the rules against the pins of the screw terminal shows that the distance between them is a bit over 5.08 mm. If you want high accuracy, you can use a calliper to make these measurements (this is something I will show you how to do in a later project). Another thing you can do is to measure the pin distance against a length that you already know is correct. In this case, You can plug the screw terminal into a standard breadboard that we know has a socket pitch of 2.54 mm. The screw terminal pins span across three breadboard holes, so the total distance is 2.54 x 2 = 5.08 mm.

Let's go back in the footprint browser. Remember in Eeschema that we can use the distance counter to measure the distance between to locations of the sheet? We can use the exact same function in most apps in KiCad. Let's use it to confirm we have the right footprint. Place your mouse cursor in the middle of pad number 1, and press the space bar to zero the *dx* and *dy* counters. Then move your cursor to the middle of pad number 2. Look at the *dx* reading in the status bar. It should show 5.08 mm (Figure 9.48).

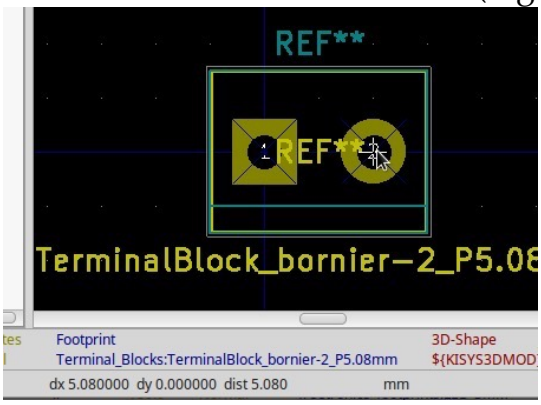


Figure 9.48: The distance between the pads is 5.08 mm.

This is the confirmation we need to be sure that this is the correct footprint for the screw terminal. Double-click on the footprint name to accept it and drop it to the layout sheet (Figure 9.49).

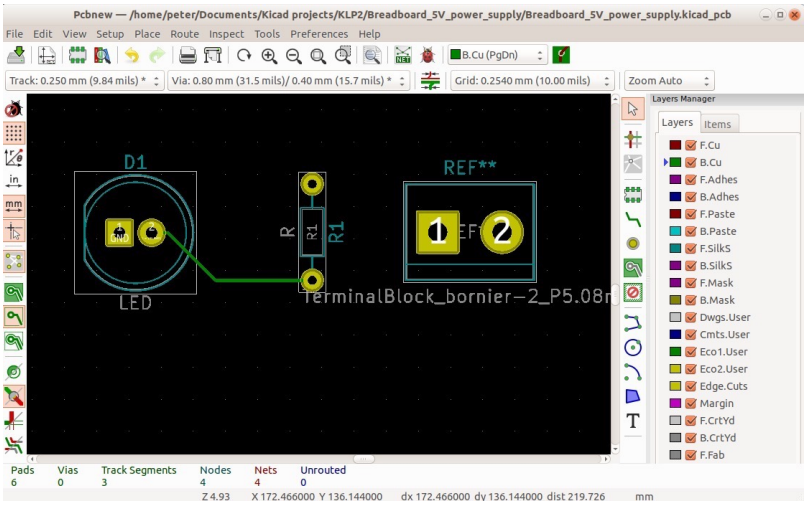


Figure 9.49: The screw terminal footprint is added to the layout.

Before we do the wiring, we should move it to a better position and orientation. Place your mouse cursor over the screw terminal footprint and use the 'M' and 'R' hotkeys to move and rotate the footprint. My version of the layout looks like the one in Figure 9.50.

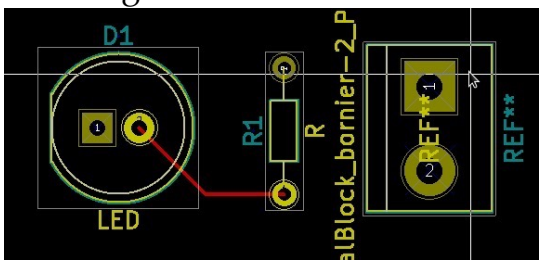


Figure 9.50: The screw terminal footprint at its final position.

With the new footprint in the layout, the next thing to do is the wiring. Let's do that next.

Wiring

Right below the 'Add footprint' button in the right toolbar is the 'Add tracks and vias' button. We use this button to create, you guessed it, tracks and vias. As always, there are hotkeys to quickly enable these functions: 'X' to add new track, and 'V' to add new vias.

Let's work on adding new tracks. We need a new track that connects the cathode of the LED to one of the pads of the screw terminal, and another to connect the free pad of the resistor to the other pad of the screw terminal

First, ensure that the front copper layer is selected. We are working on this project as a single-layer project.

Next, place your mouse cursor over pad 1 of the LED, and press 'X' to get into track mode. The tool will start drawing immediate. As you move the mouse, you will see the track being laid behind it. If you make a mistake, just

hit the ESC key to cancel out of the track tool, and try again (press 'X' to begin drawing again).

Try to draw the track that starts from the LED cathode pad to pad 1 of the screw terminal. Remember that to end the drawing of a track, double-click. In Figure 9.51, you can see the start of my attempt.

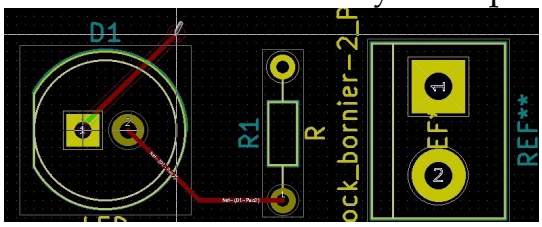


Figure 9.51: A question mark appear as I am trying to draw an ad-hock track.

When you try to end the track by double-clicking, you will notice that Pcbnew is not allowing you to do so. This is because as you are drawing the new track, Pcbnew is automatically checking for violations of the design rules. Because we are trying to create a track that does not exist as a ratsnest, to a footprint that is not in our Eeschema (and Netlist) sheet, Pcbnew assumes that we are doing something illegal.

It is true that a better way to add the screw terminal footprint to our PCB would be to first add it in the schematic diagram (Eeschema), and then import it into Pcbnew through the Netlist, as we did with the other two components. But while that is the orthodox way of doing things, KiCad allows us to work in a more ad-hoc day by turning off the design rules checking. I rarely work this way, but in some cases, this is necessary, as you will see in a more appropriate example later.

To turn off the design rules checking by clicking on the top button in the left toolbar (Figure 9.52).

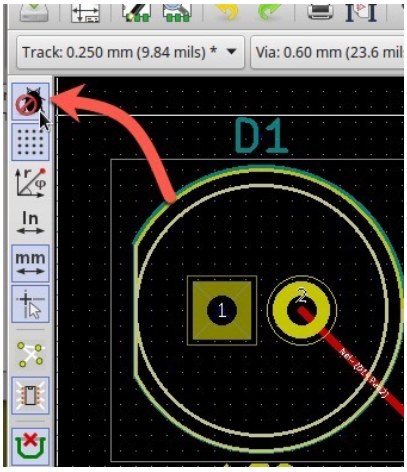


Figure 9.52: Turn off design rules checking.

With the rules checking off, you can draw the new track as you wish. To work with rules checking turned off under KiCad 5 and newer, you will also

need to switch to the Legacy Toolset. To do that, click on Preferences and then Legacy Toolset (Figure 9.53).

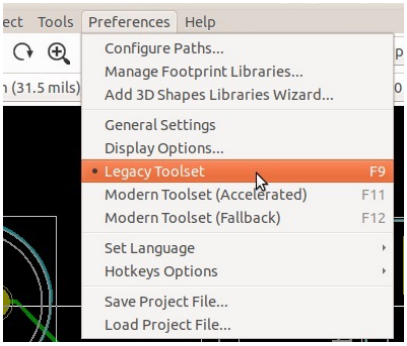


Figure 9.53: To turn off rules checking you must enable the legacy tool set in KiCad 5 or newer.

Place your mouse over Pad 1 of the LED and type 'X'. Do single clicks to create segments, and double-click to finish drawing over Pad 1 of the screw terminal footprint. Notice the '?' at the end of my incomplete track in Figure 9.54? This is a reminder that you are working with design rules checking turned off.

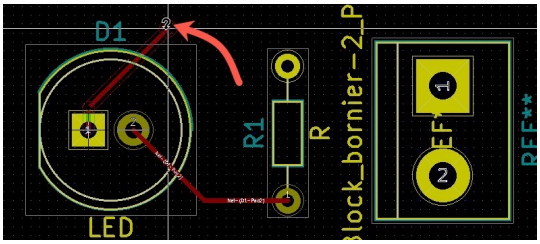


Figure 9.54: The '?' at the end of the track reminds you that you are working with design rules checking off.

Complete the wiring for the two ad-hoc tracks, so that the PCB looks like the one in Figure 9.55. The two new tracks complete the circuit. Remember to turn back on the design rules checking.

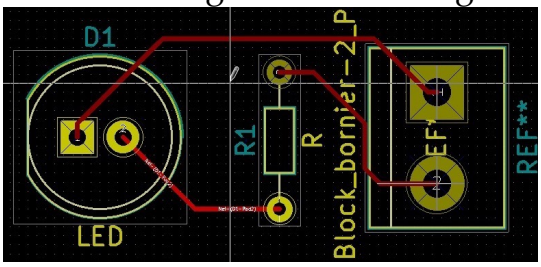


Figure 9.56: Ad-hoc tracks added.

I must repeat that adding ad-hoc wires and footprints is not best practice and should only be used for a good reason. It is best to include all components and nets in the schematic diagram, and then bring this information over to Pcbnew where you can do the layout under the supervision of the design rules checker. In this example, however, I wanted to show you the flexibility of KiCad to work without any restrictions, while

acknowledging that restrictions are in place for a very good reason: to avoid human errors.

We are close to finishing this first PCB. The few remaining on our to-do list are adding the board cut-out (that defines the borders of the board) and the graphics (things like lines and text). But before we do that, let's have a quick look at fill zones and keep-out areas.

Fill zones and keepout areas

Buttons marked '6' and '7' in the right toolbar (Figure 9.40) allow you to create copper fill zones and keep-out areas respectively. We will not be using any of the two in this example, but will we in later projects. For now, I'd like you to be familiar with these functions at a high level.

A copper fill is an area on the PCB from where the copper has been left intact, instead of being etched out. In Figure 9.57 you can see an example of copper filled areas in the back of this Arduino Uno clone.

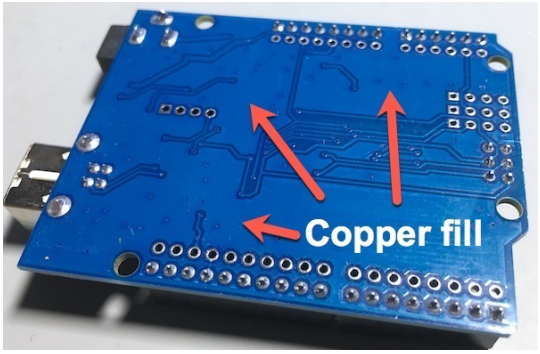


Figure 9.57: The back of this Arduino Uno contains several copper filled areas.

Copper fills (also known as 'copper pours') are commonly used to create a ground plane (as well as 5 V or 3.3 V planes) as a better way to distribute the ground level around the board (as opposed to using traces). Another reason for including copper planes is to reduce the amount of etching chemical needed to remove copper from the raw board[6] and to improve the electrical and radio interference characteristics of the final PCB.

There are different types of copper fills. Most designers tend to opt for a solid copper pour, like the one you see in Figure 9.57. It is also possible to create copper fills that include a pattern of a lattice, like the one in Figure 9.58.

[6] A raw PCB is covered with a layer of copper. Using an etching chemical process, the copper is removed so that only the needed traces are left behind.

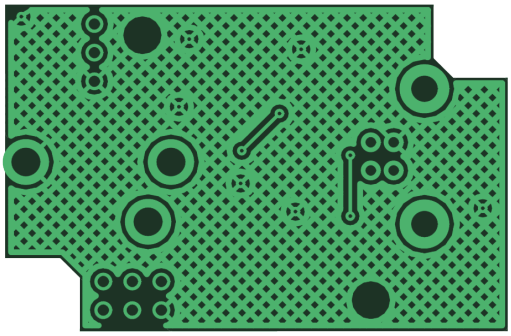


Figure 9.58: PCB board showing copper pour with a hatch fill (Image reused under Creative Commons License, https://commons.wikimedia.org/wiki/File:PCB_copper_pour_hatch_filled.png).

A keep-out area is a part of the PCB where you wish to keep empty of tracks, vias and copper pours. When you create a keep-out area, you can specify which of the above you would like to keep out of it. Your choice becomes part of the design rules checks, and the checker prevents you from adding any of those items in the keep-out area.

You may want to create a keep-out area for things such as having an area that provides structural support for other parts of a gadget, areas that can be snapped off to help in manufacturing, to name a few.

In our current simple design, we will not create a copper fill or keep-out areas, but we will in our later projects.

Edge cut

At the very least, the manufacturer needs to know the shape and dimensions of the PCB, as well as the tracks, holes, and vias on it. So far in this part of the book, we have been working on the tracks and holes for the pads, but not on the PCB's shape and dimensions. We also haven't added any graphical design elements, like text descriptions, to be printed on the PCB. Graphics are useful because they both make our PCB more aesthetically pleasing, and because they make the PCB easier to use.

Let's start with defining the shape and dimensions of our PCB first. This is something that many designers do at the start of the layout process in Pcbnew, instead of the end as we are doing here. The advantage of doing it at the start is that we can set the physical constraints of our PCB early on. Imagine that you want to design a PCB that fits nicely inside a project box. This is a major constraint, so you should set it in your design before you start placing footprints and tracks on it. If you leave it for the end of the process, you may find that the footprints and tracks are placed in a way that makes the PCB bigger than the box. This will require a lot of rework so that everything fits.

To draw the border of the PCB, first, select the Edge.Cuts layer from the layers manager toolbar or drop-down (Figure 9.59). The Edge.Cuts layer is a special layer that contains information that the manufacturer needs in order to know where to cut the board. When you export the Gerber files, the Edge.Cuts layer information is exported into its own file.

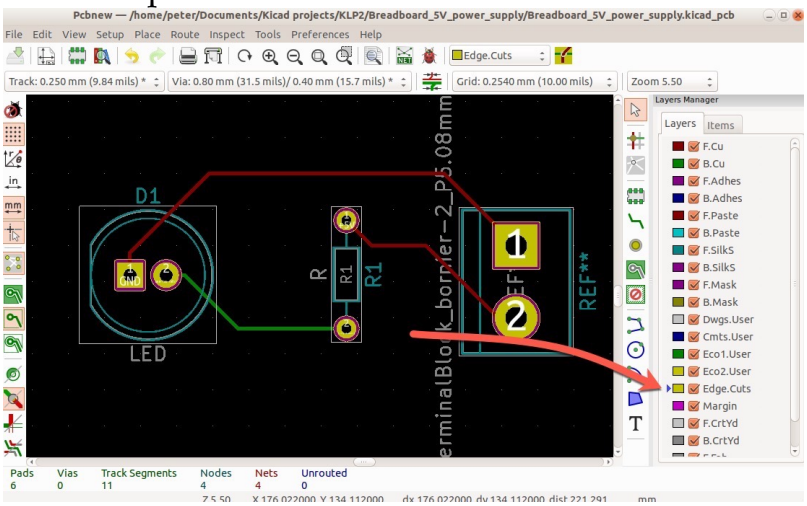


Figure 9.59: To create the PCB perimeter, select the Edge.Cuts layer first.

With the Edge.Cuts layer selected, you can use one of the graphics tools marked as '8' in Figure 9.40 to draw it. Using the line, arc, and circle tools you can create elaborate shapes for your PCB. Let's start with something simple: a rectangular PCB. Once we have this simple design ready, we can retrofit it easily by adding space for two mounting holes along the two ends of the rectangle.

Click on the line tool and start drawing the perimeter. As you are drawing the perimeter, keep in mind that the polygon you produce must be contiguous. That is, there must be no breaks anywhere along the polygon that makes up the perimeter. A common problem here is that the start of the line with the end does not meet precisely. Close the polygon so that your PCB now looks like the example in Figure 9.60.

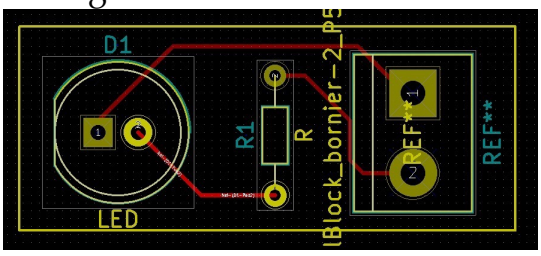


Figure 9.60: The perimeter of the PCB is a simple polygon.

Look at the 3D representation of your board by clicking on View and 3D Viewer. It should look like the example in Figure 9.61.

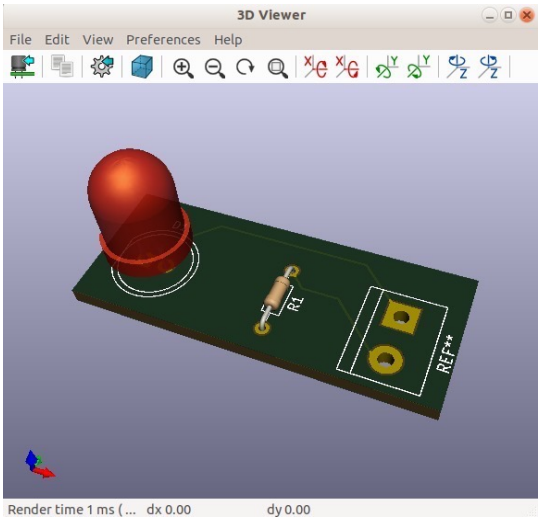


Figure 9.61: The 3D representation of the rectangular PCB.

Let's add another feature to this board, a mounting hole. With a mounting hole, you can secure your PCB to an enclosure with a screw. The easiest way to add mounting holes to your PCB is to add a single pad and modify it's drill size to accommodate the size of the screw.

First, think about the screw you'd like to use. In my case, I'd like to use a nylon screw, with a width of 2.97 mm, as measured with my calliper (Figure 9.62).

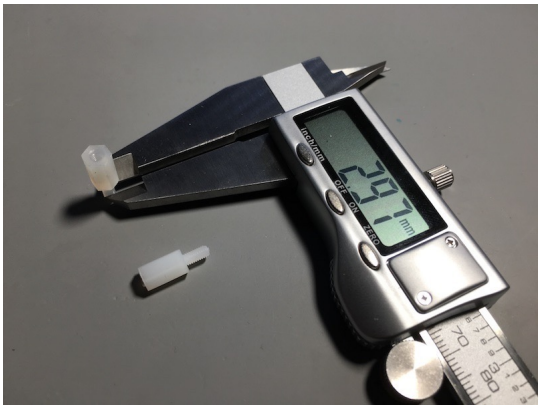


Figure 9.62: Measuring sizes with a calliper is easy and accurate.

We will need an opening slightly larger than 2.97 mm on the board. When you consider adding mounting provisions to your PCB, you should think about where the PCB will be eventually installed. It could be placed inside a project box, or stacked onto another PCB, or perhaps used on a breadboard. In the case of our example, let's plan to have it mounted on a plastic surface, using two screws. We can place the screws along the X or Y axis of the board. I'll go for the X-axis. I'll place one screw hole on the left of the LED, and another on the right of the screw terminal.

Let's add the first opening. Place your cursor on the right side of the LED, as I show in Figure 9.63, and type 'O' to add a new footprint. Do a right

click to bring up the Load Footprint dialog box, and then click on 'Select by Browser'.

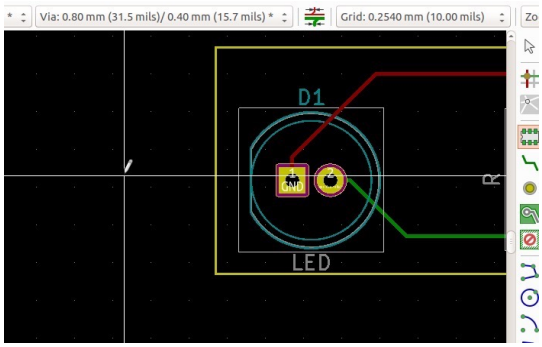


Figure 9.63: We'll place the new screw opening at the approximate location of the cursor.

In the Library Browser windows, look for the 'MountingHole' library in the left pane, and select the 2.7mm hole footprint (Figure 9.64).

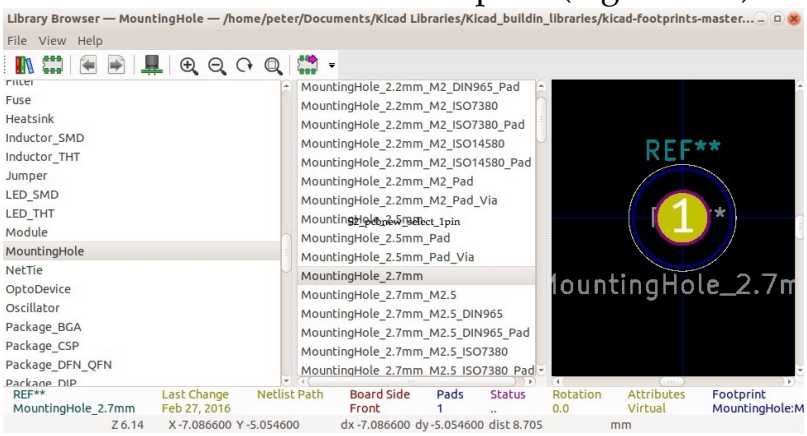


Figure 9.64: 1 pin footprints make good screw holes.

Double-click on the 2.7mm item to select it and drop it to the layout sheet. Position it close to the LED, as in the example of Figure 9.65.

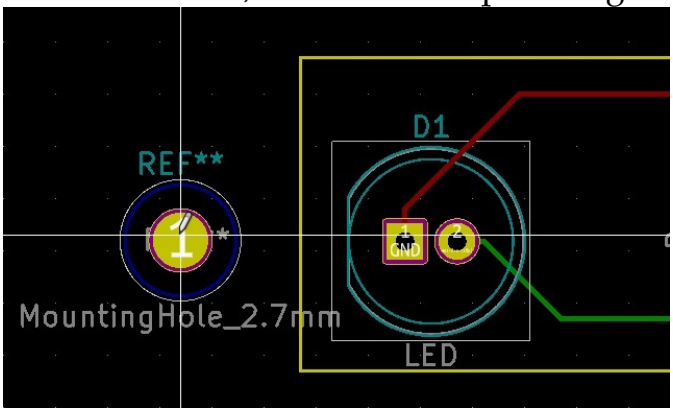


Figure 9.65: The first screw hole is placed.

We'll add a second screw hole next to the screw terminal, but first, we have to confirm that the one we just added has the correct drill size. Place your mouse pointer over the 1pin footprint, and type 'E' (for 'Edit'). Be careful where you place your mouse pointer because its position dictates the

properties window that will appear. For example, if you place your mouse over the blue circle in the outer region of the footprint, you will get a properties window that belongs to the back or front paste layer. We want the properties of the hole, so make sure to place your mouse pointer right in the middle of the footprint, as I show in Figure 9.65.

You can see the screw hole properties window in Figure 9.66.

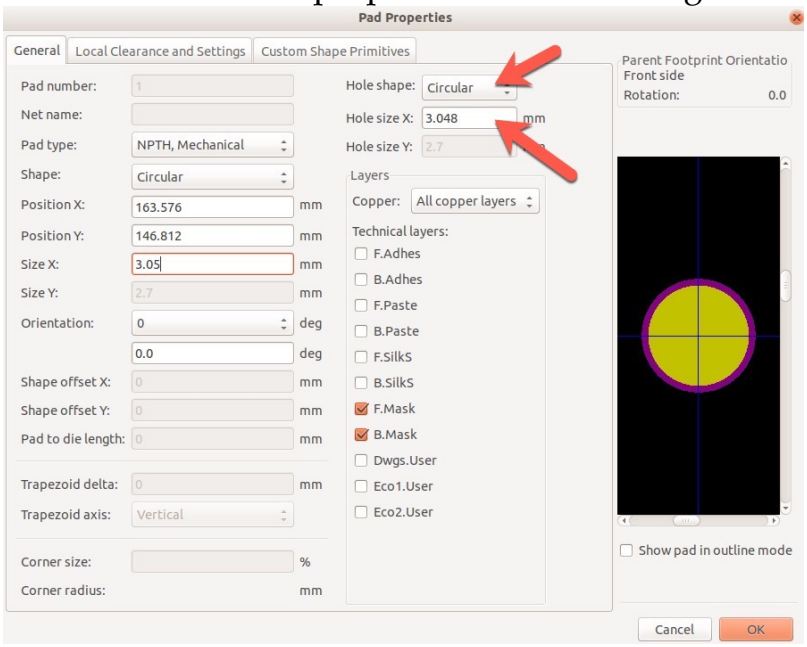


Figure 9.66: properties for the screw hole.

The arrows in Figure 9.66 show the properties that are most relevant to what we are trying to do here: the shape and size of the drill. We want a circular hole (not oval), so the default shape is good. We want a drill size that is slightly larger than the diameter of the nylon screw I am planning to use (which I measured at 2.97 mm). At 3.048 mm, the drill size is also good. With this information, we know that the screw opening is appropriate. Click Ok to close the properties window.

Let's duplicate this hole so that we can add an identical one next to the screw terminal. Place your mouse pointer over the yellow circle at the perimeter of the hole, and type Ctrl-D to duplicate the footprint (Figure 9.67).

Figure 9.67: When duplicating a footprint using Ctrl-D, take care to place the mouse pointer on the yellow line.

Using Ctrl-D can be tedious because the correct target is a very thin line. You can zoom in to increase the size of the target, or you can use the context menu. To use the context menu, place your mouse point right in the middle of the footprint and left-click. You can then follow the context menu by selecting the footprint name and then 'Duplicate Footprint'.

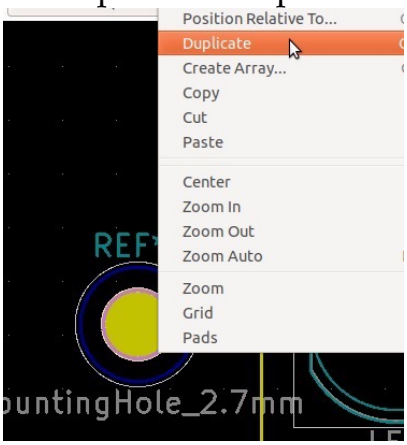


Figure 9.68: You can duplicate a footprint via its context menu.

This process will produce a copy of the original pad. Use the 'M' hotkey to move the new pad on the right side of the screw terminal, like in the example of Figure 9.69.

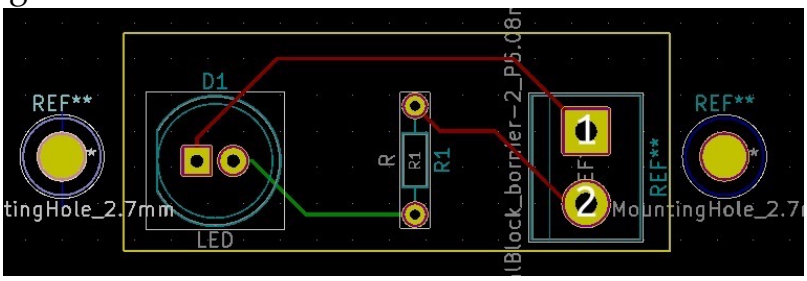


Figure 9.69: The second screw hole is on the right side of the PCB.

The two new footprints are outside the edge cut. View the 3D rendering of the PCB (View, 3D Viewer), and you will see that the new footprints 'exist' outside the PCB.

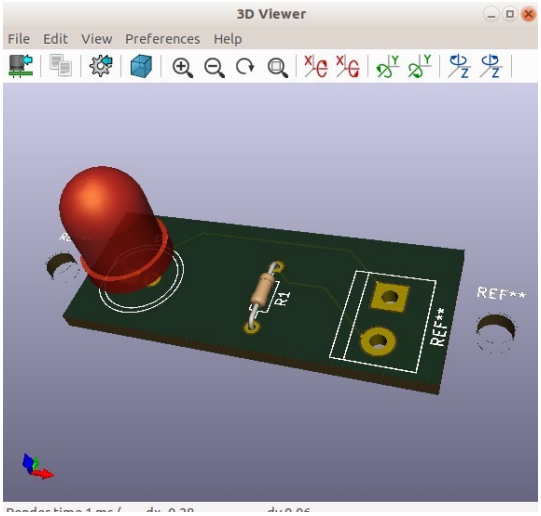


Figure 9.70: The new footprints are outside the PCB; we must redefine the edge cuts.

To fix this, you must redefine the edge cuts drawing. You can do this without deleting the existing edge cut. Let's work on the left side first. Select the Edge.Cuts layer, and the click on the line graphic button. Zoom in to a comfortable level so that you can make the yellow edge line thick enough. In my example, I work at a zoom level of 13.75. Start drawing a new line from the bottom left corner of the existing edge polygon, as in Figure 9.71.

Figure 9.71: Extend the edge cut polygon from a sensible place, like the bottom left corner of the existing edge cut.

Continue the drawing of the new segment of the edge cut until it meets the top left corner of the existing polygon. In the example of Figure 9.72, I have also deleted the vertical segment of the original edge cut polygon. It is good practice to do so in order to reduce the risk of a manufacturer becoming confused. An edge cut polygon should enclose a single area within its sides.

Figure 9.72: The left side of the PCB, redrawn with a new edge cut.

Repeat the same process on the right side of the PCB. Always take care to create polygon line without interrupts, as the polygon must be contiguous. Zoom in and pan as you do the drawing in order to work with larger 'click targets'. Figure 9.73 shows the PCB with the completed edge cut polygon.

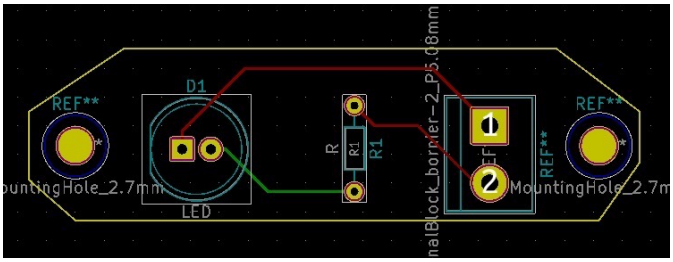


Figure 9.73: The project PCB with its new edge cut polygon.

Use the 3D viewer to see a rendered version of the PCB. In Figure 9.74 you can my PCB, complete with its screw mounting holes and the edge cut extension.

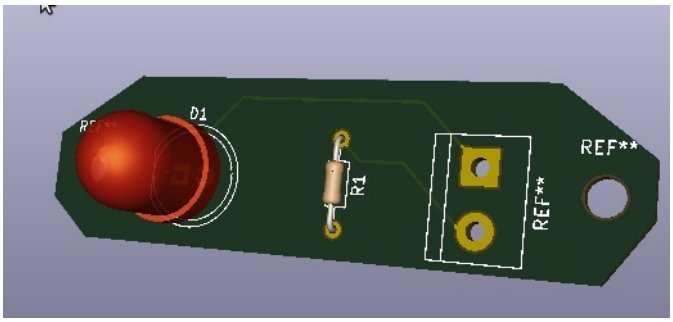


Figure 9.74: The project PCB with its new edge cut polygon, in 3D.

Graphics

With the edge cuts drawing complete, let's use the same graphics tools to add some cosmetic graphics to the board. Many footprints carry their own graphics in the silkscreen layer. In our PCB, you can see that the LED, resistor and screw terminal already have graphics. You can see them depicted in white, in Figure 9.74.

Let's add some custom graphical elements.

First, change the layer setting to front silk screen ('F.SilkS'). Then try adding some graphics using the line, circle, and arc tools.

You already know how to use the line tool. I've used it to create a box around the LED and the resistor.

To use the circle tool, select it, then click somewhere on the PCB where you would like the centre of the circle to be, and move your mouse to draw it. The further the mouse is from the centre, the larger the circle will be. I have added two circles around the screw holes.

The arc tool is good for drawing parts of a circle. In later projects, we will also use it to create round corners for our PCB, which look better to the eye and are not as sharp. You can use the arc tool similarly to the circle tool, but instead of drawing a full circle it will draw a partial circle. I have added a

few decorative arcs on the inner sides of the two screw holes. In Figure 9.75 you can see my PCB art, and in Figure 9.76 its 3D rendering.

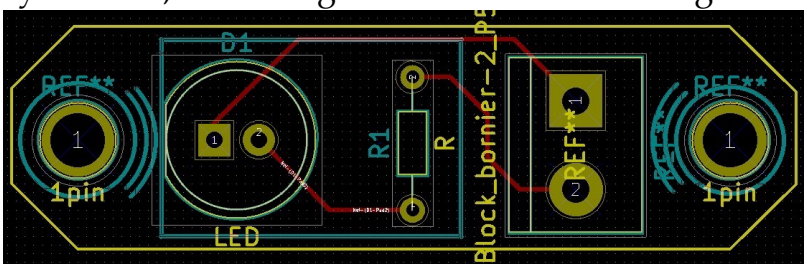


Figure 9.75: An example of simple graphics created using the line, circle, and arc tools.

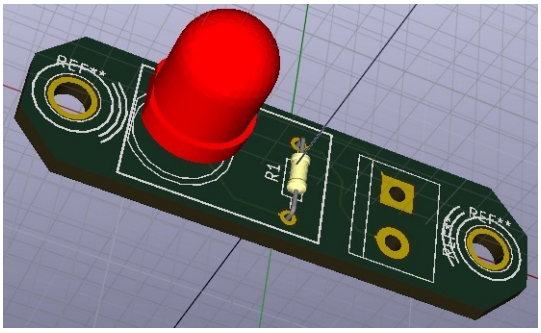


Figure 9.76: A 3D rendering of simple graphics created using the line, circle, and arc tools.

You can also add graphics to the bottom of the PCB. At the moment, there are no graphical elements there. Let's add a couple of boxes and a circle.

First, select the bottom silkscreen layer ('B.SilkS'). All elements on the PCB will overlap unless you un-select a layer. I find it easier to work with silkscreens when I only have the one I am working on enabled and the other disabled. You can do that from the layers manager toolbar. In Figure 9.77 I have disabled the front silkscreen layer by clicking on its checkbox (in the red box). This hides the silkscreen drawings in the front silkscreen so I can work on the bottom silkscreen layer.

Figure 9.77: When you work in the bottom silk layer you may disable the front silk layer in order to reduce clutter.

In Figure 9.78 you can see my PCB's silkscreen drawings in purple. I have unchecked the top silkscreen layer to make it easier to see the bottom silk screen elements.

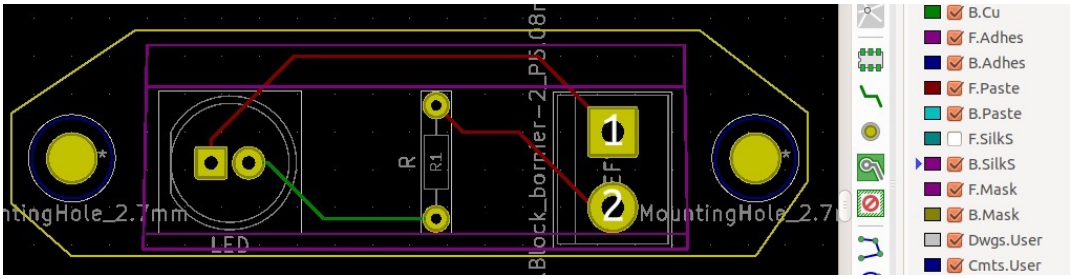


Figure 9.78: The purple line belong to the bottom silkscreen, with top silkscreen disabled.

In Figure 9.79 you can see the 3D rendering of the bottom of the PCB, which shows the bottom silkscreen.

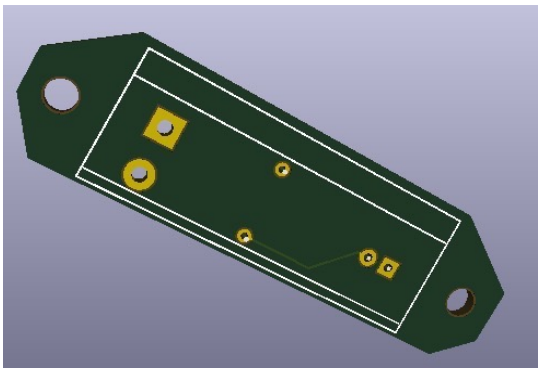


Figure 9.79: 3D rendering of the PCB showing the bottom silkscreen.

In Figure 9.80 you can see the current state of the PCB with all layers visible.

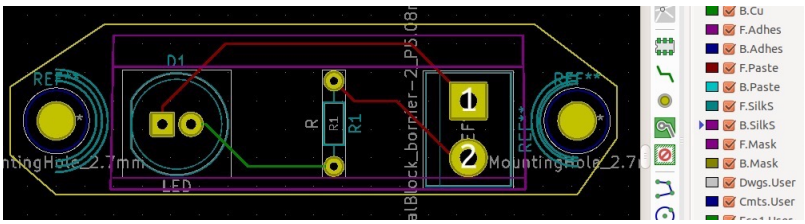


Figure 9.80: The PCB showing top and bottom silkscreen contents.

It is also to add actual graphics to the silkscreen layer of the PCB, not just simple lines and circles. You can have logos, for example, or elaborate artwork. To do so you need to follow a process that involves converting a bitmap graphics file into a footprint, before inserting it into the PCB. You will learn how to do this in the breadboard power supply project.

This PCB is almost complete. There is only one more thing we need to do, and that is to add text to the front and bottom silkscreen.

Text

To add text to the PCB, we can use the Text tool, which is one of the graphics tools marked as '8' in Figure 9.40 . Let's go right ahead and use it to add your name to the board. Select the front silkscreen layer, and then click on the 'T' button. Click on the location on the PCB where you would like the text to appear. The Text Properties window will appear. Type your name in the Text field, and double check that:

- 'F.SilkS' appears in the Layer drop-down menu,
- 'Normal' appears in the Display drop-down menu.

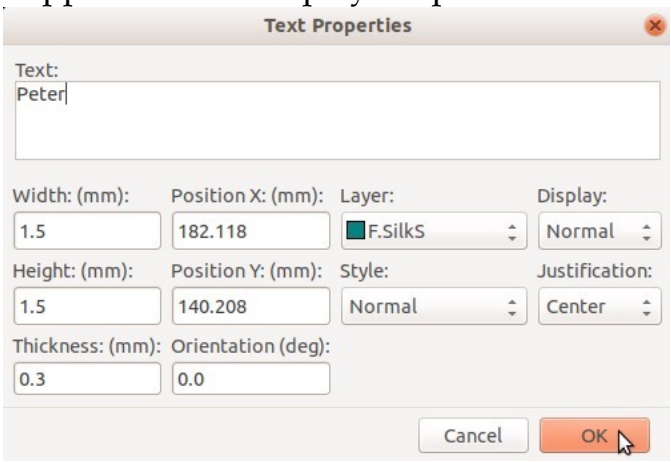


Figure 9.81: The Text Properties window.

You can control the size of the text and its style, but other than that, the text controls are limited. You have the option of placing the text in other layers, including in the top or bottom copper layer. Copper layers are meant to

be used for pads and tracks, so unless you have a good reason for this, use the silk layers for text.

If you choose to add text to the bottom layer, take care to set the Display drop-down menu to 'Mirrored'. This will inverse the text so that when it is printed by the manufacturer it is readable.

I will also add text that provides information about the size of the resistor, and also the polarity of the screw terminal pads. This information is very used for the end user of the board. In Figure 9.82 you can see the final top layer silk screen, and in Figure 9.83 is its 3D rendering.

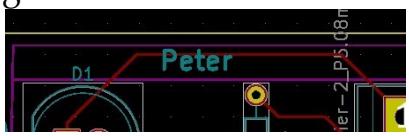


Figure 9.82: Top silk screen with additional text.

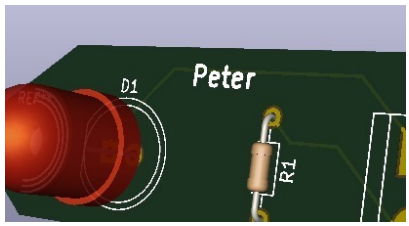


Figure 9.83: Top silk screen with additional text.

The text '330' has a character width of 1 mm. The rest of the characters are 1.5 mm wide each. I placed the '+' and '-' symbols outside the perimeter of the screw terminal so that they remain visible after the screw terminal is soldered on. For the LED, the only use of these symbols is during the assembly, to help me place the LED in the correct orientation. Once the LED is on, the two symbols will be hidden, but that is not a problem since I will not need them anymore.

I will add some text to the bottom of the board, where there is much more space than the front. I typically include a board name, a version number, a date, and my name. Go ahead to add your text, but remember to choose 'Mirrored' in the Display drop-down menu.

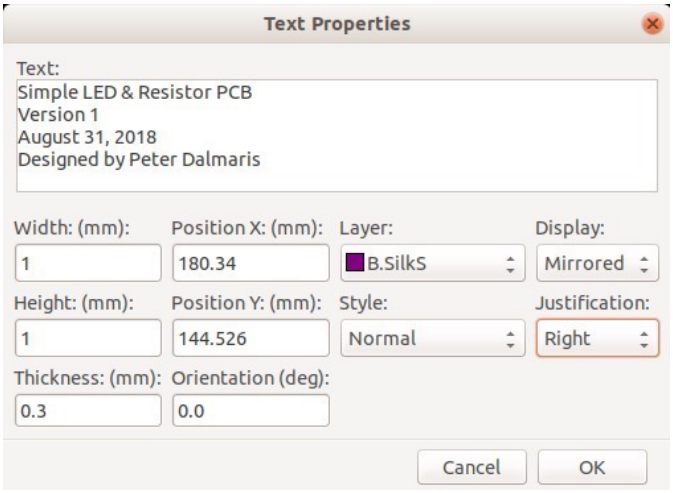


Figure 9.84: The Text Properties, set to print in the bottom silkscreen layer.

I have changed the character width to 1 mm to allow the full text to fit in the available space. I also changed the Layer to B.Silk, the Display to Mirrored, and the Justification to Right. You can use the 'M' hotkey to precisely place the block of text on the PCB, just like you do with any footprint.

Figures 9.85 and 9.86 show the final result.

Figure 9.85: The PCB, with all layers showing enabled.

Figure 9.86: 3D rendering of the back of the PCB, with the text clearly visible.

There's one thing you will need to attend to before we can declare this project complete. With reference to Figure 9.85, notice the text 'REF**' appearing over the screw hole footprints? This isn't text that we would like to include in our manufactured PCBs. Text or other graphical elements that you

do not want appearing in the final PCB can be made invisible by editing the item's properties.

Let's make the three instances of 'REF**' invisible. Place your mouse cursor exactly over the 'REF**' text of the right screw hole. Then, use the 'E' hotkey to show the text item's properties window.

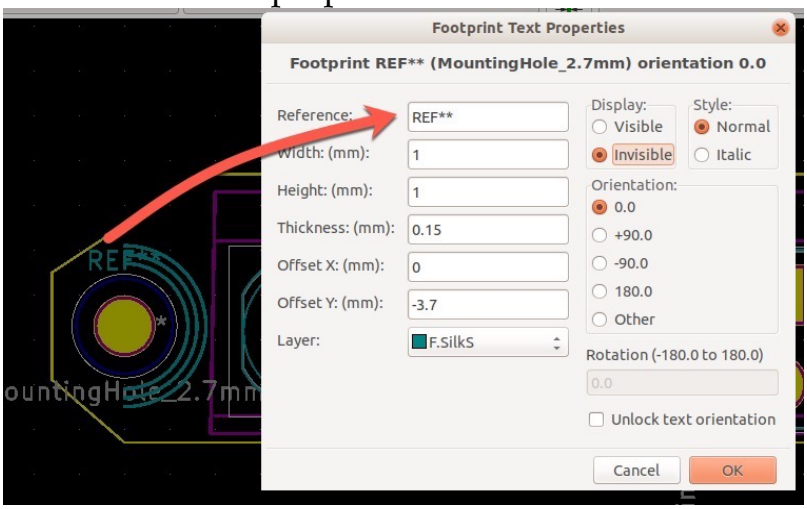


Figure 9.87: To make a text item invisible, edit it's Display property.

The Display property of the item is right above the OK button. Click on the 'Invisible' radio button, and the 'Ok'.

In Figure 9.89 and Figure 9.88 you can confirm that the 'REF**' instances are now invisible.

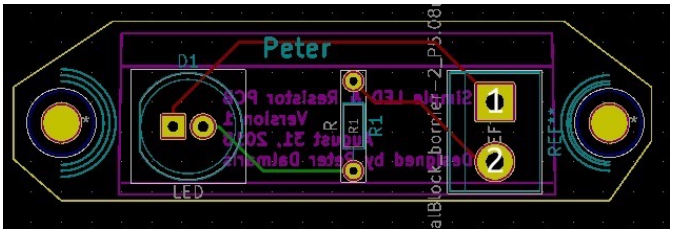


Figure 9.89: The 'REF**' instances are now invisible.

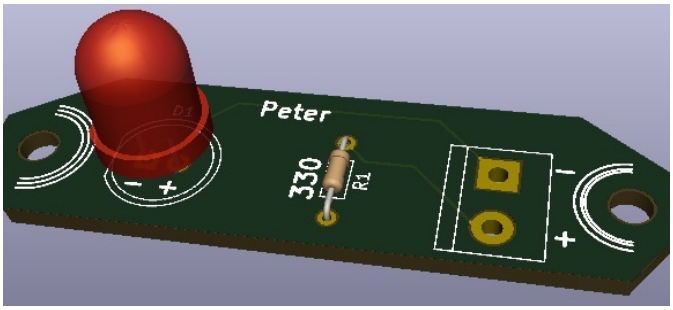


Figure 9.88: The invisible 'REF**' instances are not showing in the 3D rendering.

Well done! At this point, your PCB is complete. Make sure you save your work (KiCad does not do this automatically).

You might be wondering, why not go ahead to manufacture this board? While the board you just created is an excellent first project, I'd like you to

extend your skills by designing a more challenging board. That board, a simple breadboard power supply, will also be very useful for many of your future projects, so it is an excellent candidate for sending to manufacture.

Let's complete this section by taking a closer look at the Layer Manager, Status bar and the menus, and then we'll continue with another project.

Layers manager

You are already familiar with the Layers manager from your work on the simple LED and resistor PCB. The layer manager, apart from allowing you to select the layer on which you would like to work next, also allows you to select which aspects of the layout are rendered.

The Layer manager appears in two different forms. It's available as a simple drop-down as part of the top toolbar, and in its full-featured variation on the right side of the right toolbar. Because the full layer manager takes up a substantial real estate in Pcbnew, which you may need to use in the layout sheet, Pcbnew provides a button in the right toolbar that allows you to show or hide the layers manager toolbar.

You can see all this in the example of Figure 9.90.

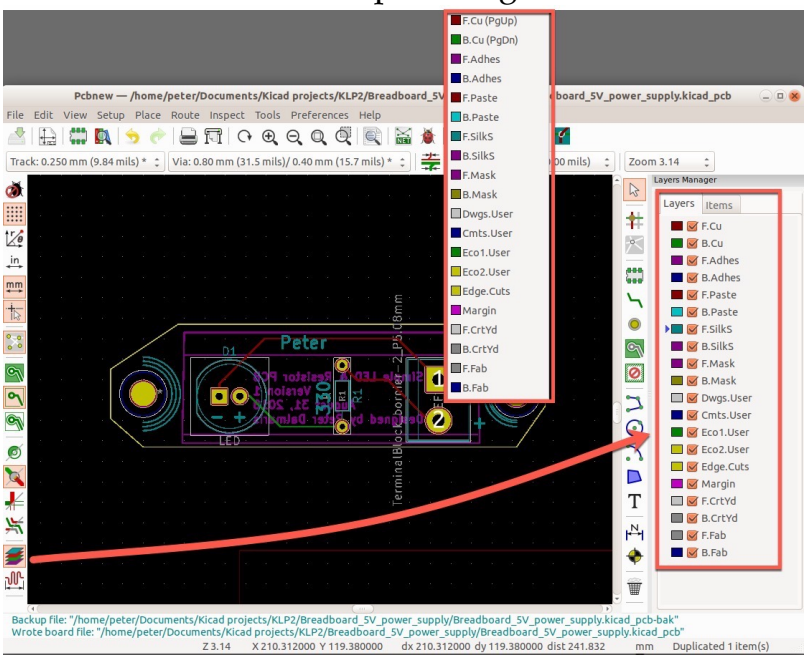


Figure 9.90: The two variation of the layer manager, and the show/hide button.

The drop-down version of the layout manager and the Layer tab of the toolbar version are almost identical. They both allow you to select the current layer. But the toolbar version also allows you to hide or show the items that belong to a layer. For example, to make the tracks drawn in the front copper

layer, just un-check the F.Cu option. The red track in our small PCB will disappear.

Try hiding the items in the front and back silkscreen layers as another example.

Hiding and showing layer items is not possible through the drop-down version.

The toolbar version also has a second tab, titled Items (Figure 9.91).

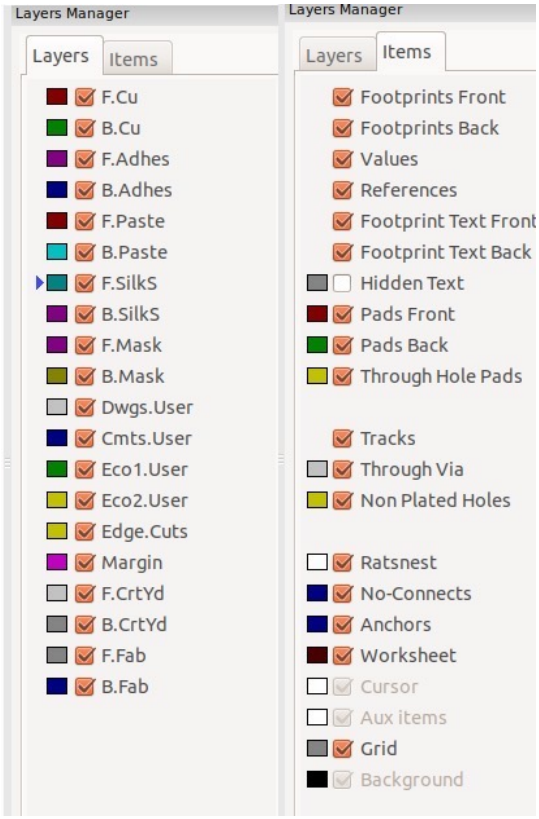


Figure 9.91: The Layer (left) and Items (right) layers of the Layers Manager toolbar.

In the Items tab, you can control the visibility of additional items of your PCB. For example, try un-checking the 'Text Front' item. This will make text to disappear, even if it belongs to multiple layers. Experiment with some of the other options in the Items tab, including the Grid option. When you check and uncheck the Grid option, notice how the Grid button (at the top of the right toolbar) updates its state.

Status bar

The status bar appears in the bottom of the Pcbnew window. It contains two segments. The top one displays the properties of the currently selected item, which the second one shows coordinates, distance, and zoom information.

Let's have a look at an example. In your simple LED and resistor PCB, click on the LED footprint. The properties of the LED footprint will appear in the status bar (see Figure 9.92).

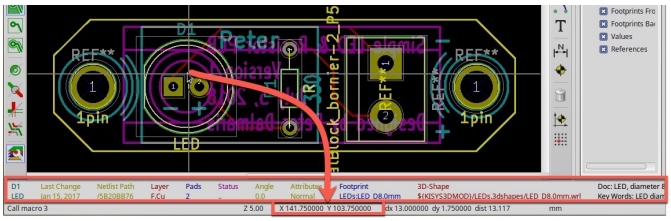


Figure 9.92: The selected LED footprint with its properties displayed in the status bar; the coordinates of the mouse pointer are also displayed in the status bar.

This way, you can quickly get information about footprints, tracks, text and graphical items.

A very useful feature of the status bar is distance counter. You have seen this feature before, in Eeschema. It works in the exact same way in Pcbnew. Let's look at an example.

What is the width and height of the PCB you have just designed, in millimeters? We can measure it. Please your mouse pointer on the left extreme of the PCB, press the spacebar to reset the counters, and move it to the right extreme. The *dx* counter will show a value of 44.50 mm. The *dy* shows a value of -0.25 mm because I move my pointer down during the measurement (Figure 9.93).

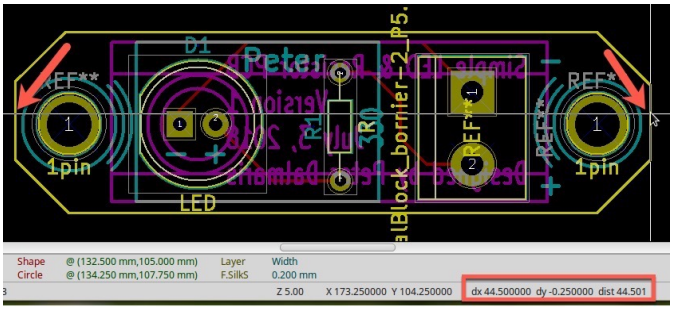


Figure 9.93: The width of this PCB is 44.50 mm.

Do the same on the Y axis to find out the height of your PCB. The height of my PCB is 12.75 mm (Figure 9.94).

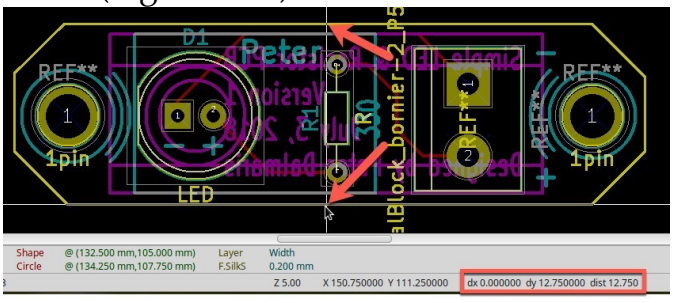


Figure 9.94: The width of this PCB is 44.50 mm.

Just like the status bar in Eeschema, the status bar in Pcbnew also provides information about absolute cursor coordinates, the zoom level, and the distance unit selected.

Menus

To complete our tour of Pcbnew, let's have a quick look at the menus.

The first thing to notice is that many of the functions that are available through the toolbars are also available through the menu system. For example, in the File menu, you will find options for printing, editing the page settings, plotting and saving the layout, and in the View menu, you will find options to zoom in/out, redraw and fit the layout to screen.

All of the menus except for Place, contain unique functionality. In this segment, we'll review the most commonly used of those functions.

File

In Figure 9.95 you can see the contents of the File menu.

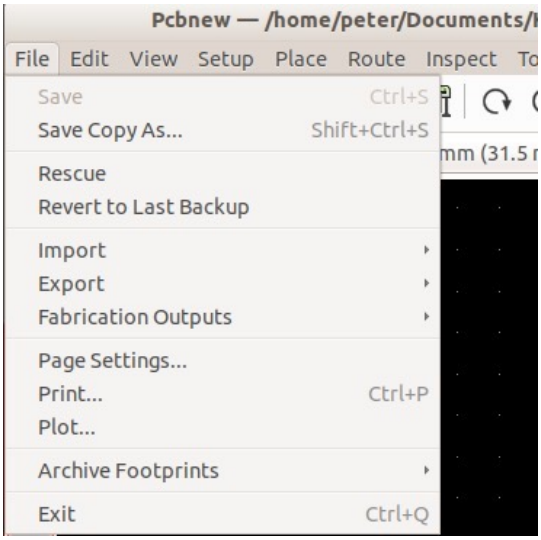


Figure 9.95: The Pcbnew Edit menu.

You should be able to recognise many of the items in it:

1. **Page Settings**, allows you to edit the contents of the title block and the page layout, as you have already learned.
2. **Print**, allows you to print the layout on paper.
3. **Export**, allows you to export the layout in various formats. For example, you can use the SVG files (one per layer) in third-party drawing applications, and Specctra DSN for using external autoroutes that support this format (such as FreeRouting, which you can learn about in a relevant recipe).
4. **Plot**, allows you to export the layout to a format appropriate for sending to a manufacturer, like Gerber. Interestingly, you can also use Plot to

export to SVG, PDF and other formats. All of those can be used to manufacture the final PCB.

5. **Close**, to close Pcbnew.

Apart from the above, several functions available through the File menu are new:

1. **Save a Copy As…**, allows you to save the layout under a new file name. This is useful when you want to branch out your layout. For example, you can have one layout designed to use surface mounted components with the file name 'LED_resistor_smd.KiCad_pcb', and another with the file name 'LED_resistor_th.KiCad_pcb'.

2. **Revert to Last Backup**, allows you to load the layout recovery file. Pcbnew automatically saves your work every 10 minutes. This value is configurable via the Preferences, General window. The backup file has the '.KiCad_pcb-bak' extension. When you click on Revert to Last, Pcbnew will ask you to confirm that you want to revert to the content of the backup file and replace your current work.

3. **Rescue**, allows you to load an automatically created backup. By default, Pcbnew will back up your work every 10 minutes. You can control the interval through the General Settings window, from the Preferences menu item. I personally save my work manually every couple of minutes, and commit to the project Git repository and prefer not to rely on KiCad's autosave function.

4. **Fabrication Outputs**, give you access to several types of file formats that can be used in a PCB fabrication process. In this book, you will learn how to export for manufacturing using the industry-standard Gerber files.

5. **Import**, allows you to import Specctra auto routing and DXF (Drawing Exchange Format) files that contain CAD models. DXF is a format developed by Autodesk. With it, for example, you can create a layout of a PCB placement in a program like Fusion 360 and import it into KiCad.

6. **Export**, allows you to export layout data to Specctra, GenCAD (a file format used in PCB manufacturing and testing), IDFv3 (a PCB file format) and others. These file formats make it possible to use KiCad PCB projects with other CAD software. Unless you have a need to inter-operate with third-party CAD software, you are unlikely to need to venture into this menu.

7. **Archive Footprints** gives the ability to package the footprints found in a layout in a directory within your project so that you can share the project with your collaborators. Your collaborators will be able to work with

your project even if they don't happen to have the needed footprint libraries installed.

Let's have a look at the functionality found in the Edit menu next.

Edit

In Figure 9.96 you can see the contents of the Edit menu.

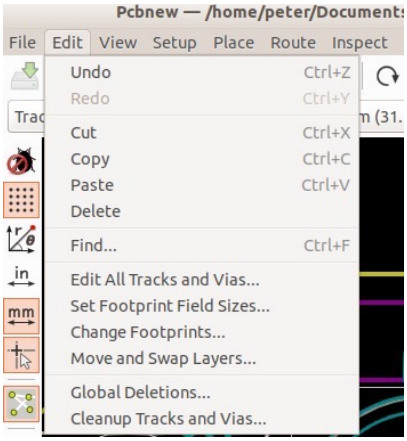


Figure 9.96: The Pcbnew Edit menu.

In the Edit menu you can see the standard Undo, Redo, Delete and Find buttons, but in addition, there are several more buttons that are worth a closer look.

In summary:

- **Global Deletions**, which allows you to delete all items of a specific kind.
- **Cleanup Tracks and Vias**, which allows you to clean up stray items, like unconnected tracks and redundant vias.
- **Move and Swap Layers**, which allows you to swap a layer with any other layer.
- **Set Footprint Field Sizes**, which allow you to set default properties for footprint text values.

The Global Deletions window provides a quick and efficient way to remove all items of a specific kind of items from the layout. This is one of those features that you will use rarely, but that will save you a lot of time when you need them. While it is a blanket approach to removing items, remember that you still have access to the Undo function, so if you make a mistake you can restore your work.

Try to be selective, when you use Global Deletions. Selecting the 'Clear board' radio button will delete everything in the layout. Selecting 'Text' will only delete the text items. You can also apply your choice to a specific layer or to all layers.

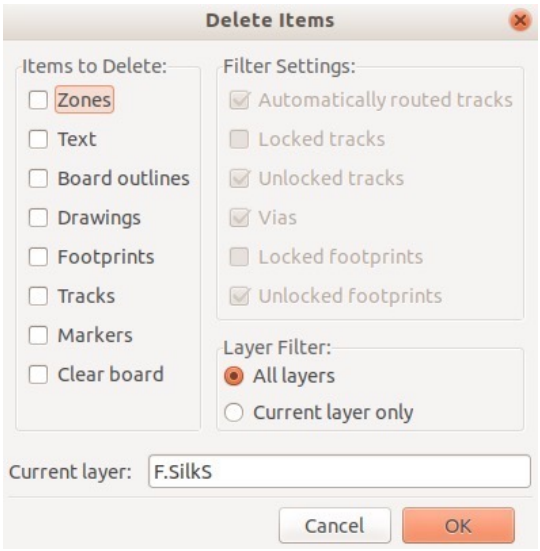


Figure 9.97: The Global Deletions window.

The Cleanup Tracks and Vias windows help us tidy up leftover tracks and vias, quickly. Try this experiment. In your LED and resistor layout, add a few stray tracks. As you are laying a track, type 'V' to add a via, and continue drawing. Double-click to end drawing. Don't connect the stray track to any pads. That is what makes it 'stray'. Now, bring up the Cleaning Options window by clicking on Edit and Cleanup Tracks and Vias. Leave the current default selections as they are, and click OK.

The stray track and vias that you just added should disappear.

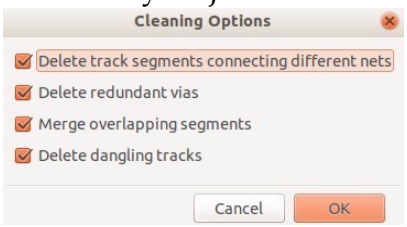


Figure 9.98: The Cleaning Options window.

The Swap Layers function (Figure 9.99) is very useful when you want to transfer all tracks from one layer to another. For example, say that you want to move all track from the front copper layer ('F.Cu') to the bottom copper layer ('B.Cu').

To do that, first bring up the Swap Layers window by clicking Edit and the Swap Layers. Then, click on the button marked '…', right next to the 'F.Cu' label, and from the small window that appears, select 'B.Cu', then dismiss the small window by clicking on the 'x' button, and click OK. Try that now in your LED and resistor layout. See how the (originally' red tracks are now green? That happened because the tracks were transferred from the front copper layer to the back copper layer.

To restore the tracks to the front copper layer, do the opposite. Back in the Swap Layers window, click on the '…' button that is next to the 'B.Cu' label, and select 'F.Cu' from the small window that appears. Then, dismiss the small window and click OK.

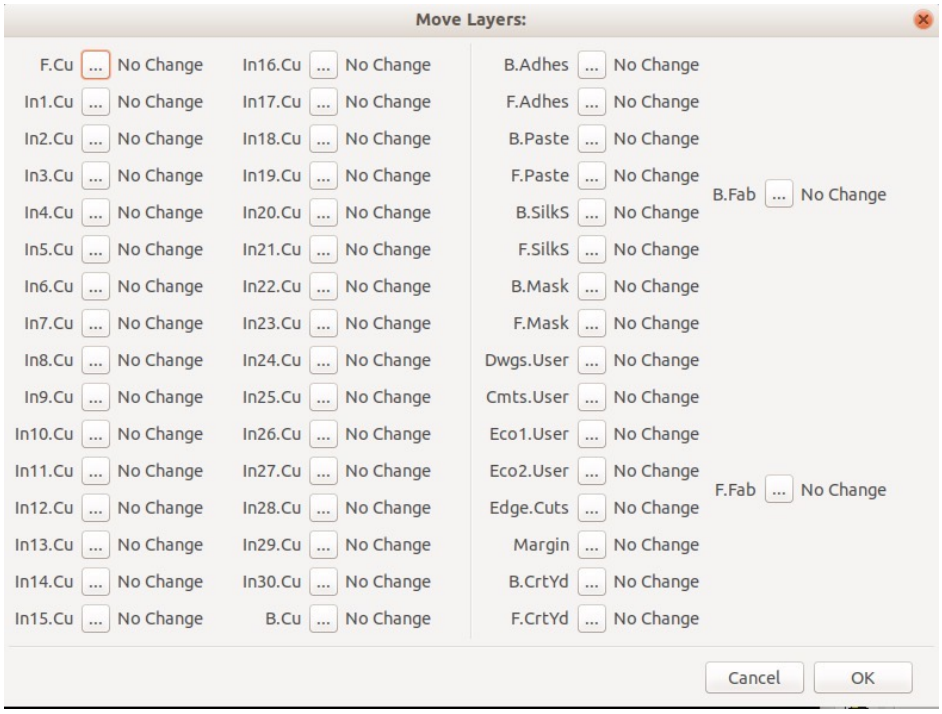


Figure 9.99: The Swap Layers window.

In Figure 9.100 you can see the Set Text Fields window. The width, height and thickness properties control how the footprint text values are rendered in the editor. Do a small experiment to see how this works: With the simple LED and resistor layout loaded in Pcbnew, bring up the Set Text Size window. Change Width and Height to 2, and Thickness to 0.35. Click OK. You should see that all text items that belong to properties will be re-drawn with the new properties, and look around twice as large. To make them go back to their original sizes, restore Width and Height to 1, and Thickness to 0.15.

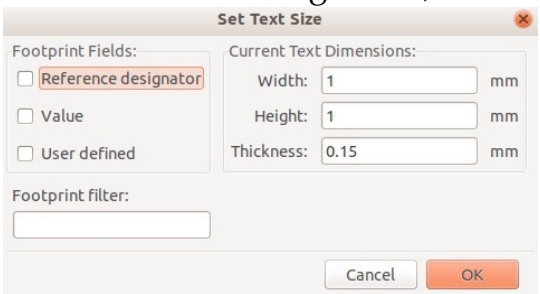


Figure 9.100: The Set Text Size window.

View

In Figure 9.101 you can see the contents of the View menu.

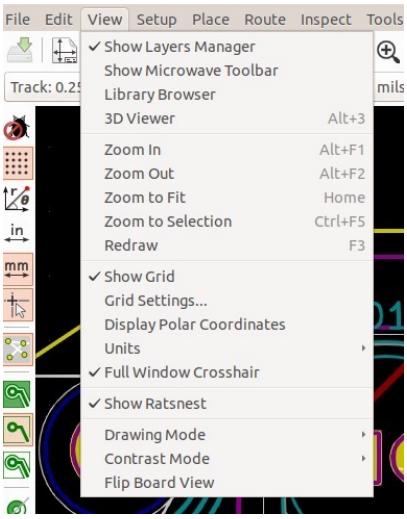


Figure 9.101: The Pcbnew View menu.

In the View menu you can see the familiar Zoom, Fit and Redraw buttons, but in addition, there are several more buttons that I'd like you to look at.

1. **3D Viewer**, which renders a 3D view of the PCB
2. **Show Ratsnest**, which shows you a list of all unconnected nets in the PCB
3. **Drawing Mode,** which allows you to choose from a variety of graphics options that help when you work on busy boards

You have already used the 3D viewer in several occasions (Figure 9.102). Apart from the default settings of the viewer, there are several more features you should be aware of. First, experiment with the top menu bar, which allows you to zoom in/out, pan and rotate the model.

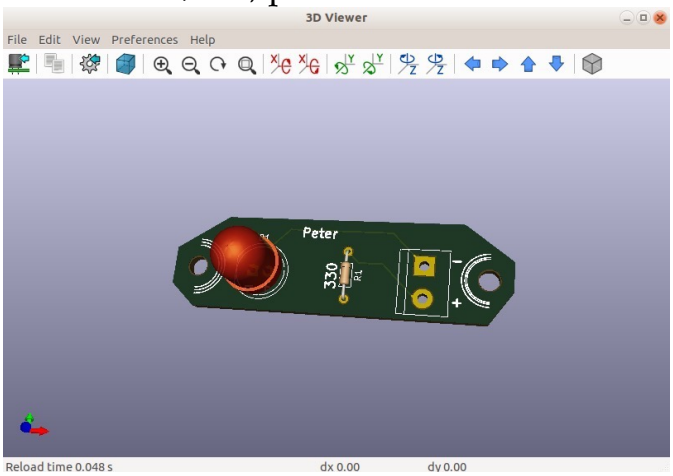


Figure 9.102: The 3D Viewer.

You can control much of the rendering behavior of the viewer through its display options window, which you can access by clicking on the cog-wheel button in the top toolbar (Figure 9.103). Try out the various options to

see the effect they have on the rendering of the model. There are more rendering controls through the Preferences menu.

Another useful function is available in the File menu. There, you can export the rendered view to PNG or JPEG image files. This is a good way to capture a high-quality render of your board that you can insert in a blog post or documentation.

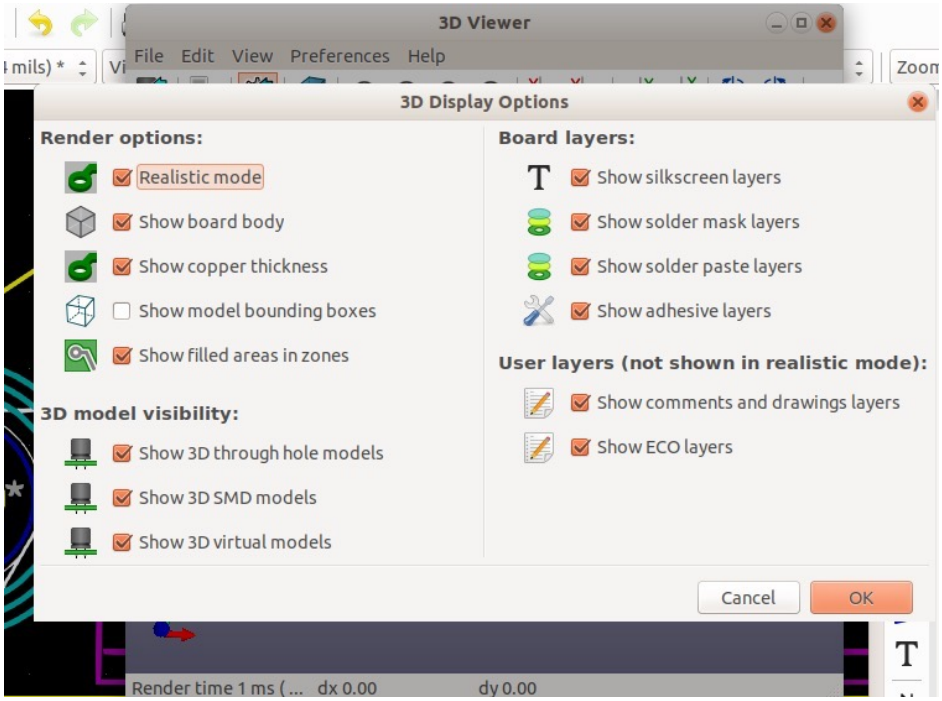


Figure 9.103: The 3D Viewer's display options window.

Setup

The Setup menu contains just two items, Design Rules and Layers Setup. You can see the Setup menu in Figure 9.104.

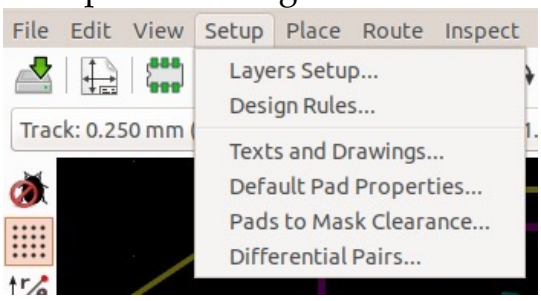


Figure 9.104: The Setup menu

The two most important items in this menu are the top two: Layers Setup and Design Rules. You will use them in all projects of this book. You can read a summary of what these items do in the next few pages.

The bottom four items allow you to configure secondary aspects of how the layout, such as how text should look like by default, and what should be the default configuration of a pad.

Let's take a closer look at those.

Design Rules Editor

In the Breadboard Power Supply and later projects, you will be spending a lot of time learnings on how to use the Design Rules Editor. Briefly, this Editor allows you to create classes of nets that have specific design attributes. Tracks that belong to those nets are then drawn with adherence to this attributes.

For example, you can create a net class called 'PWR' that dictate the track width, via diameters and other attributes that should apply specifically to a track that belongs to this class. You could create another class called 'SGN' that dictate these attributes for tracks that convey signals, instead of power. This makes sense because a typical signal track will only need to convey a few milliamps of current, while a track that feeds power to the circuit components will be required to convey much more than that.

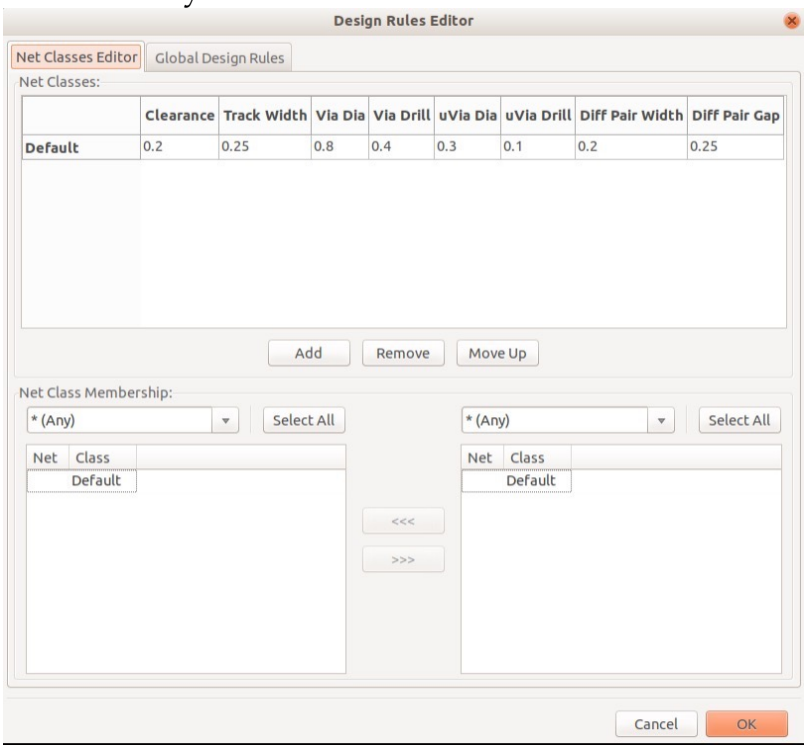


Figure 9.105:The Net Classes Editor.

In Figure 9.105, you can see Net Classes Editor with its default values. In our simple PCB, there's only one net, and it is assigned to the Default class. In later projects, you will learn how to create classes and assign nets to one of them.

In the same figure, there is also the Global Design Rules tab (Figure 9.106) .

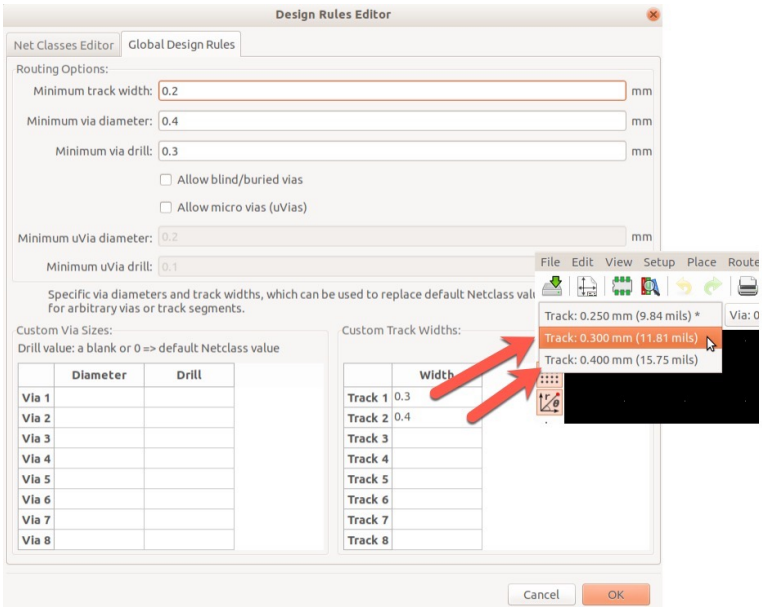


Figure 9.106: The Global Design Rules.

There, you can create custom track and via sizes, and have them available in the top toolbar drop-down. This way you can create tracks and vias using ad-hoc sizes that can override their assigned class.

Layers Setup

In Layers Setup (Figure 9.107), you can control the number of layers you would like your board to have, and, to some extent, the type of these layers.

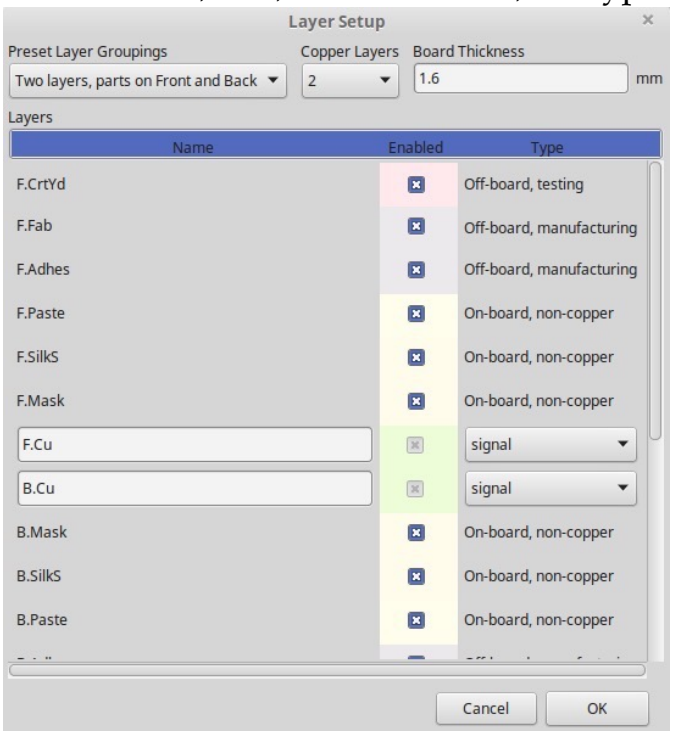


Figure 9.107:The Layers Setup.

Pcbnew can produce PCBs with up to 32 copper layers, and this is configurable in the Layers Setup window.

There are presets of up to four layer-PCBs with parts on the front, back, or both. As you experiment with selecting one of them, notice the effect on the enabled or disabled status of each layer. For example, if you choose the 'Two layers, parts on Front only' preset, you will notice that many of the copper layers are disabled. If you click on 'OK' to commit the change, these layers will disappear from the Layers Manager, leaving only the enabled ones showing.

You can change the layer configuration as you are working on your PCB. For example, you may start with a 2-layer PCB and then realise that you need additional layers to complete the routing. That is not a problem. Just go into the Layer Setup window and add a couple of layers.

Other setup options

The bottom four options of the Setup menu contain:

1. Text and Drawings
2. Default Pad Properties
3. Pads to Mask Clearance
4. Differential Pairs

The Text and Drawings settings allow you to configure defaults for things like the text character width, height and thickness, and the thickness of a graphic segment (such as a line or an arc).

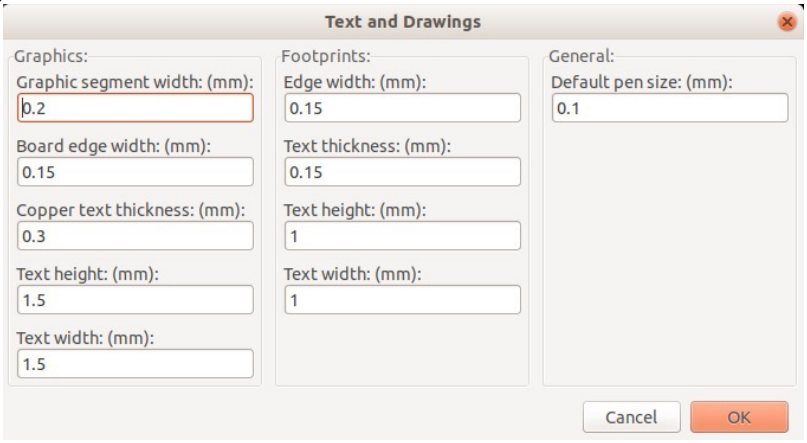


Figure 9.108: Default text and graphics settings.

I find the defaults in Figure 9.108 appropriate for most boards.

The Default Pad Properties is fairly involved, but you only need a few options tweaked in relation to setting defaults (Figure 9.109).

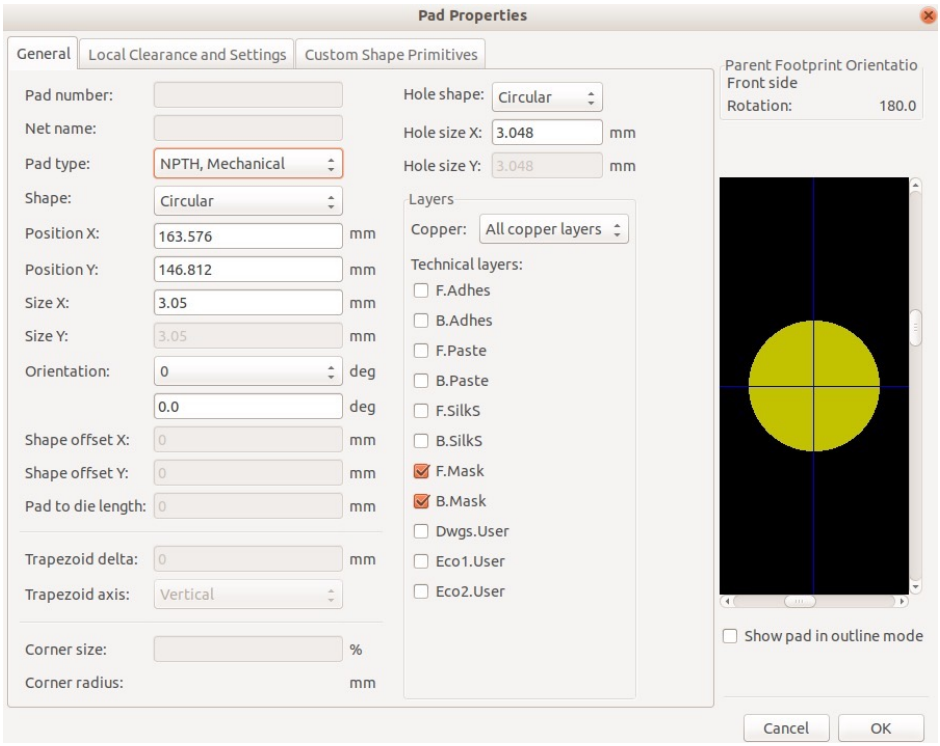


Figure 9.109: Default pad properties.

In the General tab, the pad type offers four choices:

1. Through-hole, best for using with through-hole components. This option creates a hole that is internally plated, with pads on either side so that components can be soldered.
2. SMD, suitable for SMD components. An SMD pad does not contain a hole.
3. Connector, which is similar to the SMD type, but is round and typically used as a contact for probes
4. NPTH, Mechanical, which is a non-plated through hole pad, suitable for creating mounting holes.

If you choose 'Through-hole' as your default, you will also be able to specify the hole shape, as either oval or circular, and for each type, the exact dimensions.

In the Local Clearance and Settings tab you can specify minimum clearances (Figure 9.110). I prefer to keep the values there at 0 so that the footprints in my projects inherit these values from their parent footprints.

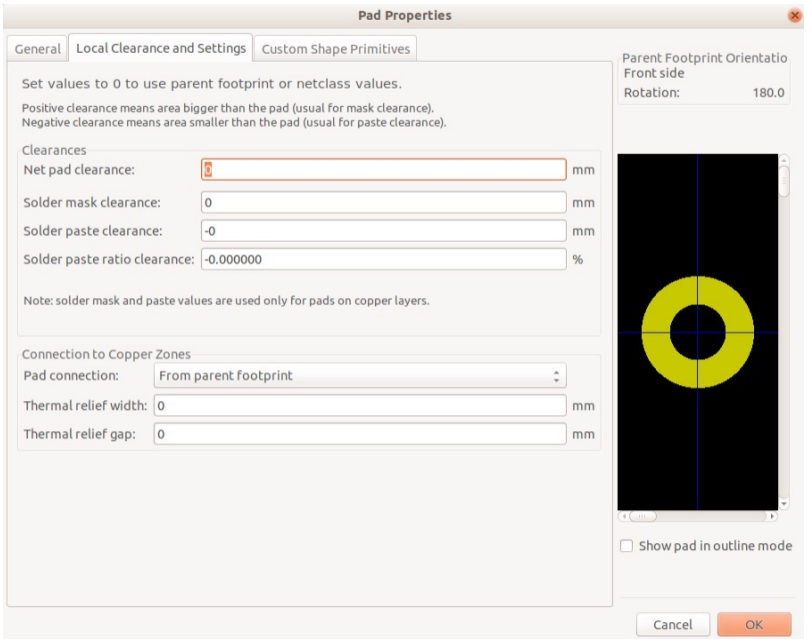


Figure 9.110: The Local Clearance and Settings pad

The Pads Mask Clearance window (Figure 9.111) allows you to control the pad mask clearance. The mask is the thin layer of non-conductive materials that covers the front and back of the board and also prevents solder from creating bridges between pads. The values you use here control how much of the pad will be covered with the masking material. If you cover too much, you will decrease the risk of solder bridges, but you will make soldering harder since there will be less of the pad exposed.

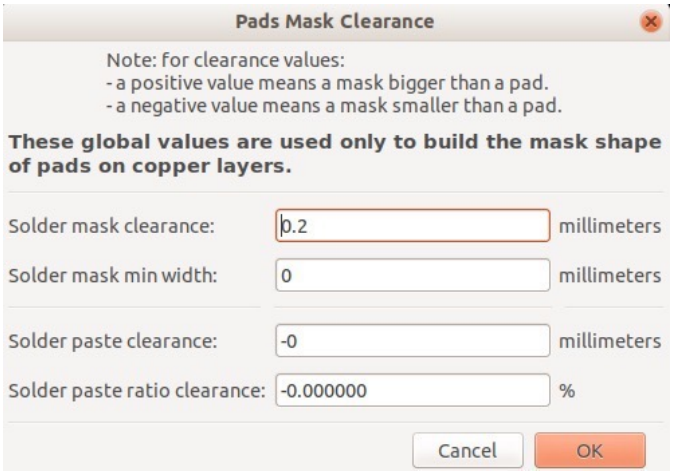


Figure 9.111: The Pads Mask Clearance window.

Finally, the Differential Pair Dimensions allow you to control how close together will be drawn traces that belong to the same pair, and how wide each trace should be.

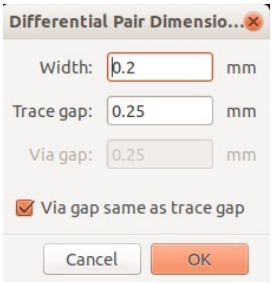


Figure 9.112:Differential pair settings.

Place

In Figure 9.113 you can see the contents of the Place menu.

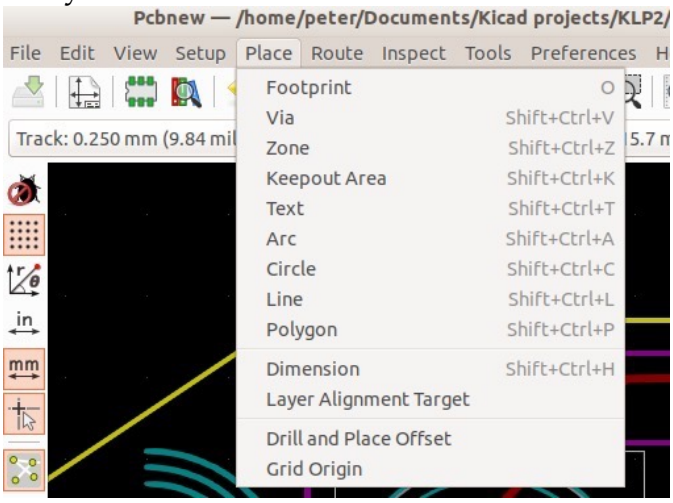


Figure 9.113: The Pcbnew Place menu.

In the Place menu, you will find the same tools as in the right toolbar. Let's not spend any more time here and move straight to the Route menu.

Route

In Figure 9.114 you can see the contents of the Route menu.

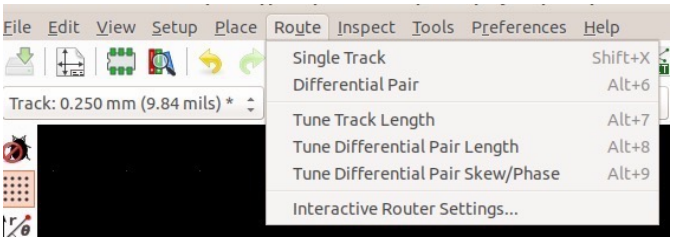


Figure 9.114: The Pcbnew Place menu.

With the Single Track option, you can create regular tracks and vias. It is the option you will use most often. The rest of the tools in the Route menu are useful in advanced layouts that involve things such as high-frequency digital busses or analog circuits where the radio properties of the tracks must be taken into account.

Let's have a quick look at each one:

1. **Single Track**, allows you to create single tracks and vias. You have already used this tool in our simple LED and resistor project.
2. **Differential Pair**, allows you to route two tracks at the same time. KiCad identifies tracks that should be routed together based on the ending of their net names. In the current version 5, two tracks can be identified as a pair if their net names end with '+' and '-' respectively, and their start pads are adjacent to each other. While you are drawing the paired tracks, you can also create vias, for both at the same time.
3. **Tune Track Length**, allows you to adjust the exact length of a single track. This is useful if you want to control precisely the signal propagation time.
4. **Tune Differential Pair Length**, which is a function similar to Tune Track Length, but for a pair of tracks ('differential pair') at the same time.
5. **Tune Differential Pair Skew/Phase**, allows you to precisely control the clock skew of differential pairs, another consideration in high-frequency digital electronics design. Clock skew is a phenomenon in which the same clock signal arrives at slightly different times to its destination. The effect of this is that different parts of the circuit can end up operating at a slight clock offset, which is not desirable. Learn more about clock skew on Wikipedia.

For example, we can use Differential Pairing for tracks that involve serial communications, such as the TX and RX tracks in a UART circuit, or the MISO and MOSI tracks in an SPI circuit. You can learn more about differential pairs in the recipe titled '44. How to use differential pairs'.

Interactive Routing

Interactive routing is one of the most important features of Pcbnew. It makes drawing tracks quick and efficient.

It helps you create tracks while avoiding items in the layout that can't be moved, or moving them (the term is 'shove') if they are movable.

Let's do an experiment with Interactive routing. Open the simple LED and resistor layout, and delete the track that connects pad 2 of the LED and pad 1 of the resistor. In Figure 9.115, I have deleted the track. To make the layout easier to read, I have disabled the front and back silkscreen layers by unchecking F.SilkS and B.SilkS in the Layer Manager.

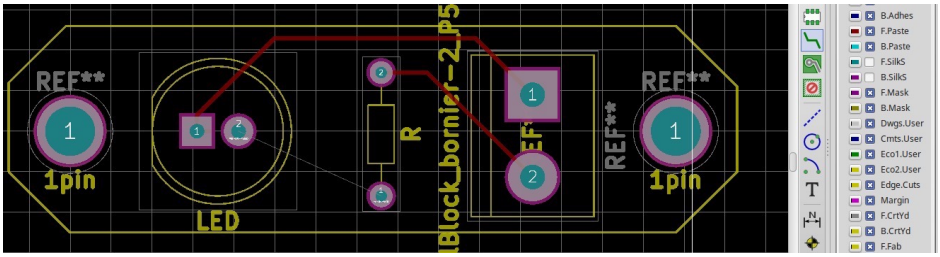


Figure 9.115: The resistor and LED are not connected.

The ratsnest line between the LED and resistor is telling us that this track must be implemented. Type the 'X' hotkey to enter the track drawing mode. Place your mouse cursor in the middle of pad 2 of the LED. Move your mouse towards the screw terminal footprint, and notice how the track that the router leaves behind does not intersect any other tracks and goes around pads. Try to finish the drawing by double-clicking on pad 1 of the resistor. You can see my, admittedly terrible, design in Figure 9.116.

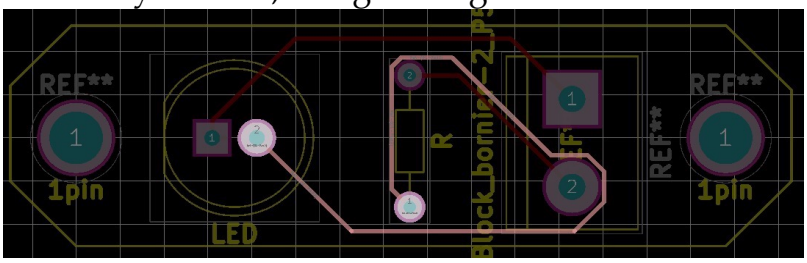


Figure 9.116: A new route using the Interactive Router, avoiding other tracks and pads.

In Figure 9.116, I have highlighted the new net using the 'Highlight Net' button in the right toolbar. Of course, this new net is a very poor design since it is much longer than necessary, with a lot of corners. In PCB design, we aim to create tracks that are as short as possible, with as few corners as possible. More about PCB design principles later.

The point of this example is that as I was moving my mouse pointer through a particular path, the Interactive Router was able to find a way to place the track without violating any of the design rules.

You have fine control over the designer through the context menu Figure 9.117.

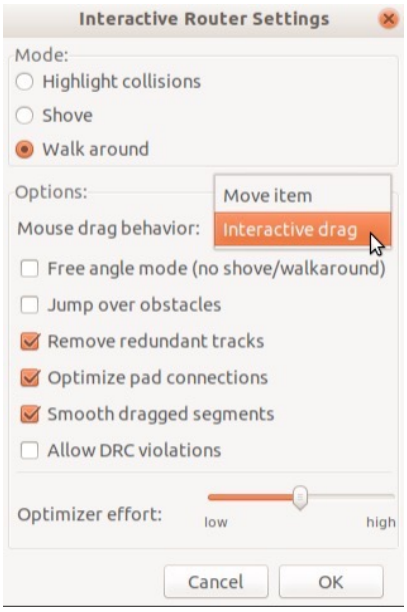


Figure 9.117: The Interactive Router Settings.

You can change these setting at any time, even during the drawing of a track. I usually set the Mode to 'Walk around', which produces a result similar to the one you see in Figure 9.116. As long as you choose a sensible path with your mouse, the interactive router will produce a sensible track. You can also try out the Shove mode. With Shove mode on, the interactive router will push over any items that are not explicitly fixed on the layout. You can have footprints on a fixed position by locking them through their properties and then allow for existing tracks and vias to be moved by the router as needed for laying new tracks. It is work spending some time to get used to this feature, as it is powerful and it will make your work with KiCad much more efficient in comparison to drawing in Default mode without Interactive Routing.

In Figure 9.117 I have selected the option 'interactive drag' for the mouse drag behaviour. This allows me to use the 'G' hotkey to drag a track to a new position without breaking it (which is the 'move' operation). Beware that you can drag a track by clicking on it to select it and then dragging to move it. The ‘G’ hotkey is a shortcut meant to save you one click. This way I can re-arrange a track that I or the Interactive Router created, without having to repair broken connections. When using the 'Move' method, the typical result is that a single segment of a track is moved, and its connection to the rest of the net is broken, which needs to be repaired.

Another powerful feature of the Interactive Router is the ability to redefine a segment of the track or the full track without having to manually delete the original. You simply type 'X' to start the track drawing process, and you draw the start and end of the new track or track segment. When you

double-click to finish the drawing process, the Interactive Router will automatically remove the old track or track segment, as it is now redundant.

Let's use this feature to repair my awful track from Figure 9.116. Type 'X' and click on pad 2 of the LED to start a new track. You can also start the drawing at another position of the existing track is you want to redefine a segment of the track instead of the complete track (Figure 9.118).

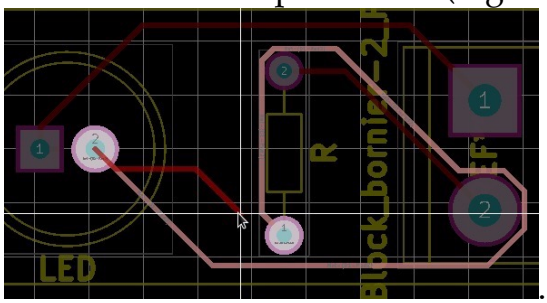


Figure 9.118: Starting a new track to replace the old one.

Then, move your mouse on pad 1 of the resistor and double-click to complete the drawing. The new track is created, and the old track is automatically deleted (Figure 9.119).

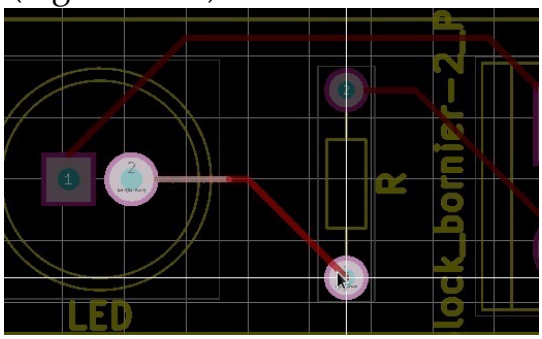


Figure 9.119: The newly created track replaces the original.

It is worth spending some time here to get used to these capabilities of the Interactive Router. We will be using it extensively in the projects that will follow.

Inspect

The Inspect menu (Figure 9.120) includes functions for looking into nets, measuring distances on the sheet, and launching the design rules checker. You can do the latter two by using buttons in the top and right toolbars.

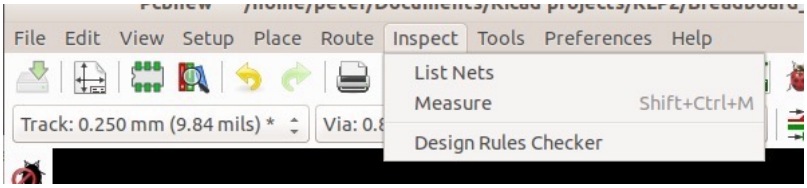


Figure 9.120: The Inspect menu.

The List Nets options give you a list of the nets that are present in the PCB. In Figure 9.121, you can see the Nets window. You can type in the name of the net you want to inspect in the filter box or click on the net row. In Pcbnew, whichever net you select will be highlighted.

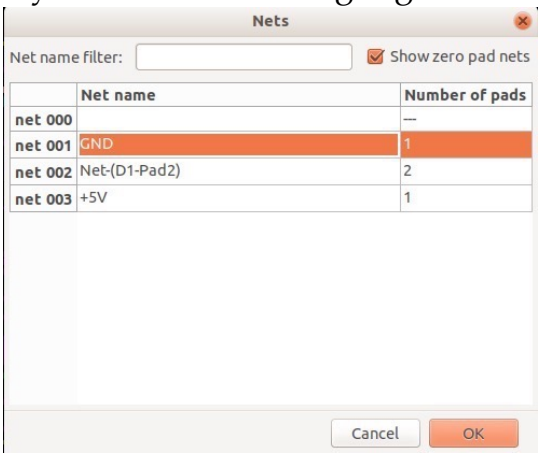


Figure 9.121: The Net Filter window.

In Figure 9.122 you can see the GND net that I selected in Figure 9.121, highlighted with a high-contrast colour. This makes it easy to see how the selected net is routed.

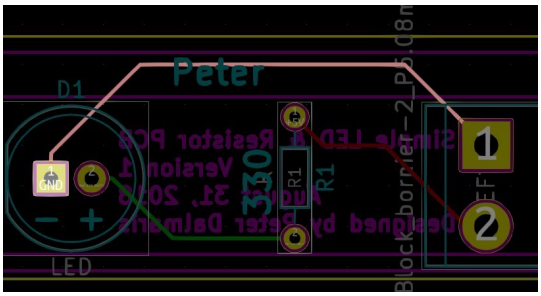


Figure 9.122: A list of nets present in this simple PCB.

With the Measure tool, you can measure distances on your board (Figure 9.123). Notice that there is a keyboard shortcut for this tool. Just click to start measuring distance and angle, and click again to end. You can use this tool by clicking on the Measure tool button from the right toolbar ( ).

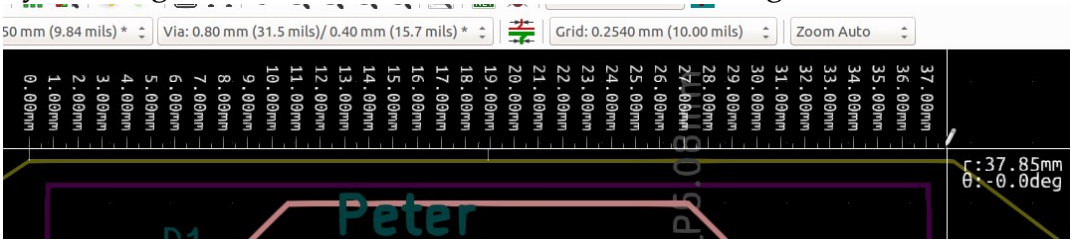


Figure 9.123: The Measure tool.

Finally, you can start the design rules check by clicking on the Design Rules Checker option in the Inspect menu, or by clicking on the bug button

from the top toolbar (). Either way, the DRC window will come up (Figure 9.124).

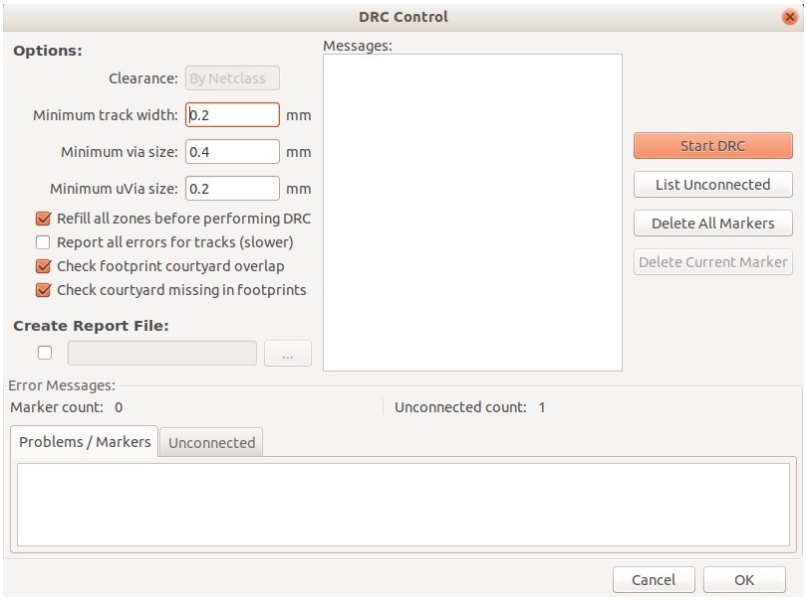


Figure 9.124: The DRC tool.

The DRC tool will check for problems in your PCB, such as unconnected nets, or traces that are too close to other traces. You will become very familiar with this essential tool as you practice working on the projects in this book.

Tools

All but one of the items in the Tools menu are available in the top toolbar. You can see the Tools menu in Figure 9.125.

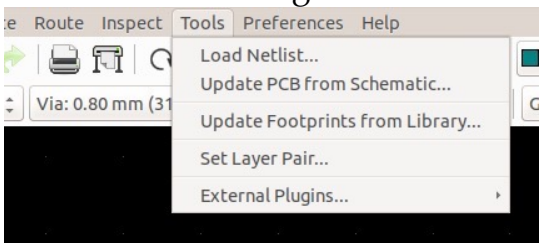


Figure 9.125: The Tools menu.

Here is a summary of the items in the Tools menu.

1. **Load Netlist**, allows you to import the Netlist file from Eeschema.
2. **Update PCB from Schematic** is an alternative to the Netlist. If you make changes to the schematic after you begin work in Pcbnew (perhaps you have added a new symbol, or fixed the wiring), you can choose the Update function to automatically annotate the symbols, create the new netlist,

and import it into Pcbnew. I am very used to working with the Netlist directly, so I don't use the Update function, however, I suggest you give it a go as it can speed up your work.

3. **Set Layer Pair**, is useful in boards with 4 or more layers. It allows you to choose which layer should be connected when you create a via.

4. **Update Footprints from Library**, will check for updated versions of the footprints that exist in your layout, and apply them. This saves you doing this work manually, as was the case with older versions of KiCad.

5. **External Plugins** allows you to extend the capabilities of KiCad through Python scripts.

I'd like to take a closer look at the Set Layer Pair option.

Let's start with the Layer Pairs option. When you work with PCBs that contain more than 2 layers, say 4, or more, you can configure which of these layers should be considered pairs when you create vias. You can configure the Layer Pairs using the Select Copper Layer Pair window. When you are working on a 2-layer board, the Select Copper Layer Pair window looks like the example in Figure 9.126. There's not much to do here, and the only option is to pair the top and bottom layers. Now, when you create a via, a track from the top layer will continue to the bottom layer, and vice versa. The 'x' marks the selected layer.

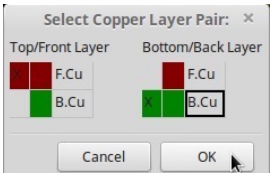


Figure 9.126: The Select Copper Layer Pair window in a two-layer board.

Change your board configuration so that it is a four-layer board. You can do that through the Design Rules menu (we will look at this in more detail in the next segment). Bring up the Layers Setup window, and change the number of layers to 4 (Figure 9.127).

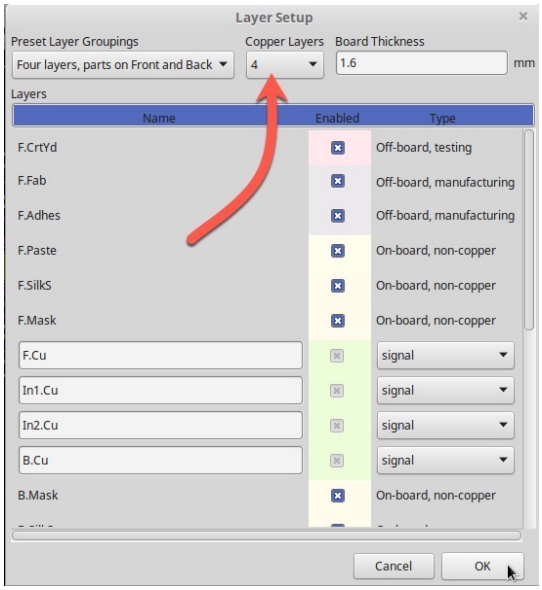


Figure 9.127: To change your board to a four-layer board, open the Layer Setup window from the Design Rules menu.

Now open the Select Copper Layer Pair window again. It will look like the example in Figure 9.128.

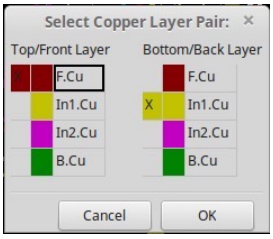


Figure 9.128: The Select Copper Layer Pair window in a four-layer board.

In this example, I have set the pairs to be the front copper and the first inner copper layers. With this configuration, when I create a via, a track that starts in the top copper layer will continue in the first inner copper layer, and vice-versa.

Please switch your board back to a two-layer configuration now before continuing.

Preferences

Through the Preferences menu, you can control various aspect of the operation of Pcbnew (Figure 9.129).

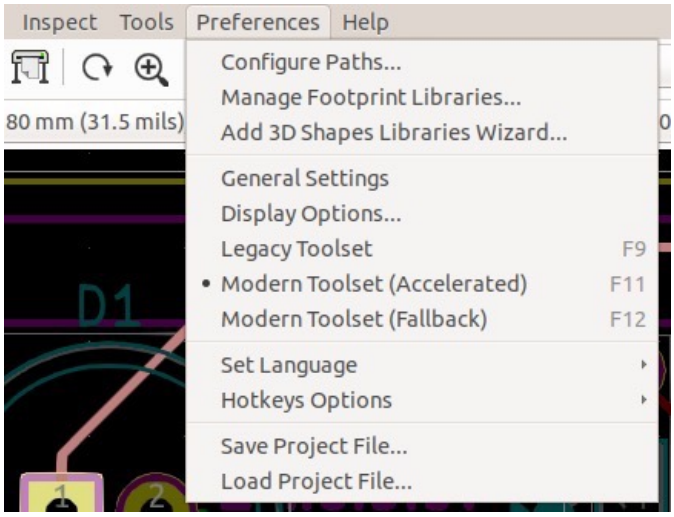


Figure 9.129: The Preferences menu.

You will be using most of the items in the Preferences menu as you work through the projects in this book, but following is a summary:

1. **Manage Footprint Libraries Wizard**, allows you to import footprint libraries. Once a new library is imported, you can use its footprints in your layout. You will learn how to use this tool in a later project in this book. There is also a recipe that explains how to do this; look for the one titled '21. Adding a footprint library in Pcbnew'.
2. **Configure Paths**, allows you to configure the locations of symbol, footprint, and 3D shape libraries. You can also access the path configuration window from the main KiCad window. Look at the Kicad documentation for details on the roles of the environment variables and the paths that they point to.[7]
3. **Add 3D Shapes Libraries Wizard**, allows you to download 3D shapes from KiCad's own repository and 3rd party repositories that are used in Pcbnew's 3D Preview. You can learn how to do this by reading the recipe titled '55. How to install 3D shapes'.
4. **General settings**, allow you to control a variety of aspect of Pcbnew's operation, such as the frequency of the autosave function, the measurement units and type of coordinate system used.
5. **Display options**, allow you to control Pcbnew operations details such as the use of accelerated graphics, the grid style and how net names are displayed.
6. **Set Language**, allows you to change the language used in the Pcbnew user interface.

[7] http://docs.kicad-pcb.org/master/en/kicad.html

7. **Hotkeys Options**, allows you to customise the various hotkeys, as you learned earlier.

8. **Save and Load project File** allows you to save the .pro file that contains information about the project (but not the layout and schematic files, since that information is stored as individual files).

9. There are recipes that cover the usage of the libraries wizard and the 3D shapes wizard. In the next few pages, you'll take a closer look at some of the other options in the Preferences menu.

General

Through the General Setting window (Figure 9.130), you can configure much of the behaviour of Pcbnew.

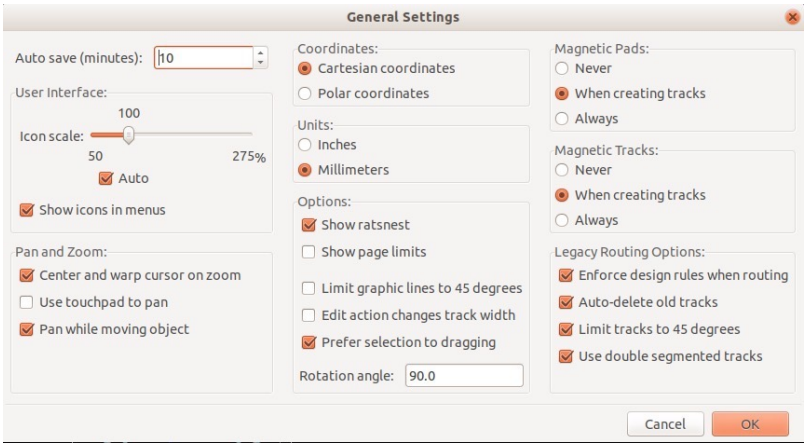


Figure 9.130: The General Setting window

Here, you can control the drawing behavior of Pcbnew. For example, you can make tracks and pads 'magnetic' so that as you draw a new track, it will 'stick' to a pad when you mouse your mouse close enough, ensuring an electrical connection.

You can control autosave, select your preferred coordinate system and unit of measurement, the shape of the course, and much more.

I find that the default settings work well and rarely make any customisations here.

Display

Similarly to the General Settings window, the Display Options window (Figure 9.131) allows you to control the graphical attributes of Pcbnew.

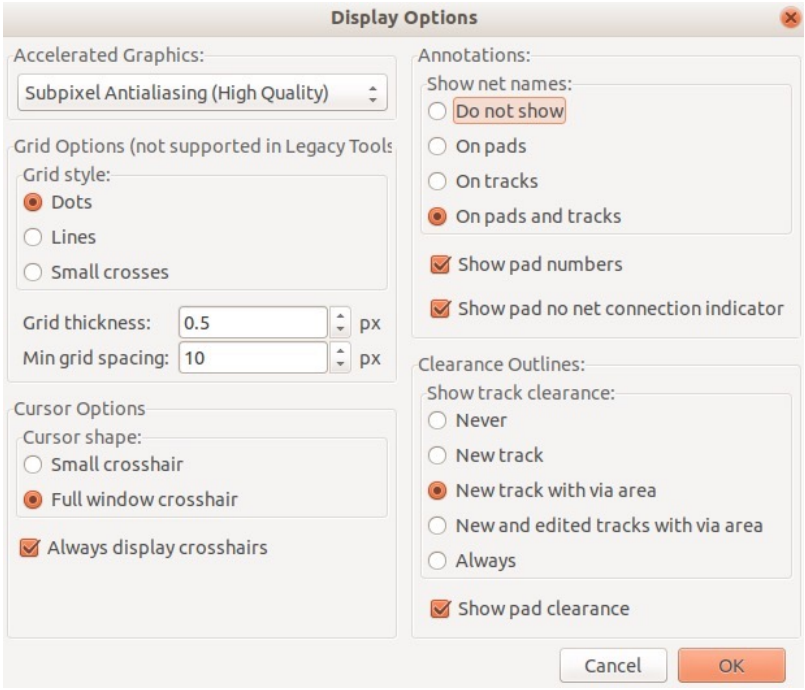


Figure 9.131: The Display Options window.

For example, you can choose if you would like to show pad numbers and footprint outlines, net names, and track clearances.

Again, the default settings are reasonable and I prefer not to make any modification to the Display options.

Language

The Language option allows you to choose one of 19 languages available for KiCad's user interface. Whichever language you choose, must also be supported by the locale of your operating system (Figure 9.132).

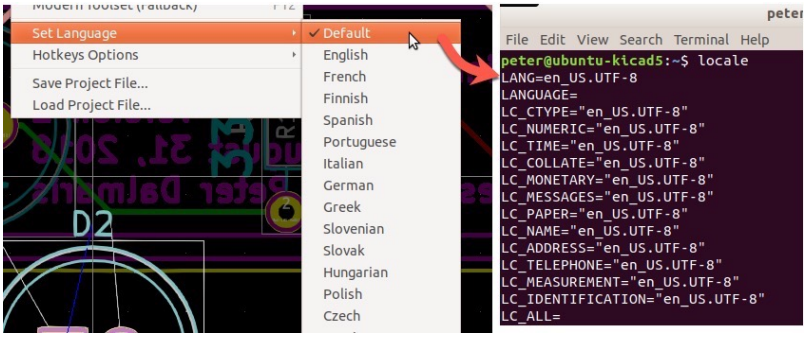


Figure 9.132: The Pcbnew language settings have to match the locale of your operating system.

Hotkeys

You have learned about the Hotkeys options in earlier segments of this book. You know that you can show the current hotkeys by selecting Preferences, Hotkeys, and List Current Keys and that you can set up your own hotkeys through Preferences, Hotkeys and Edit Hotkeys.

You can also import and export hotkeys from the same menu item, which is a good way to ensure that you have the same settings across the computers where you use KiCad.

Part 3: Design principles and basic concepts

10. About this Part

In Part 2 of this book, you learnt about some of KiCad's most commonly used features. You used Eeschema and Pcbnew to design the schematic diagram and the layout of a very simple board. You also had your first experience with some of the decisions that a PCB designer has to make as they are working on their PCB.

Before you continue your learning journey with a few more realistic PCB projects, it is appropriate to become familiar with concepts, conventions and design patterns that will help you in being more efficient in the way you work, produce better performing boards, and create boards that you feel proud to look at.

Some of the things that you will learn in this chapter are straightforward. You will learn about the symbols and units that appear in the schematic diagrams and the layout diagrams. The terminology used to describe the various components and characteristics of a printed circuit board.

You will also learn about the general processes of the schematic and layout steps of designing a PCB. These are processes that every designer will go through to one degree or another, regardless of which CAD application they are using. These processes will become natural as you become more proficient and confident. The layout chapter in this Part of the book will take you through it, and you will have the opportunity to practice it in the projects of Part 4.

In this part of the book, you will also learn about the most important practical design considerations and conventions that can help you create amazing PCBs. For example, how should you arrange the various parts of a circuit on the PCB? How should you supply power to each section? How should you arrange components such as buttons and potentiometers (which make up the user interface of a device)? How thick should the copper of a signal or power trace be? Is it appropriate to have traces that contain 90-degree angles? The answer to these questions and many more will be covered in this Part 4 of the book.

11. Schematic symbols

Electronics and PCB design has its own symbolic language. This language is written in schematic diagrams. In Figure 11.1 you can see an example of a schematic that contains several symbols of this language.[8]

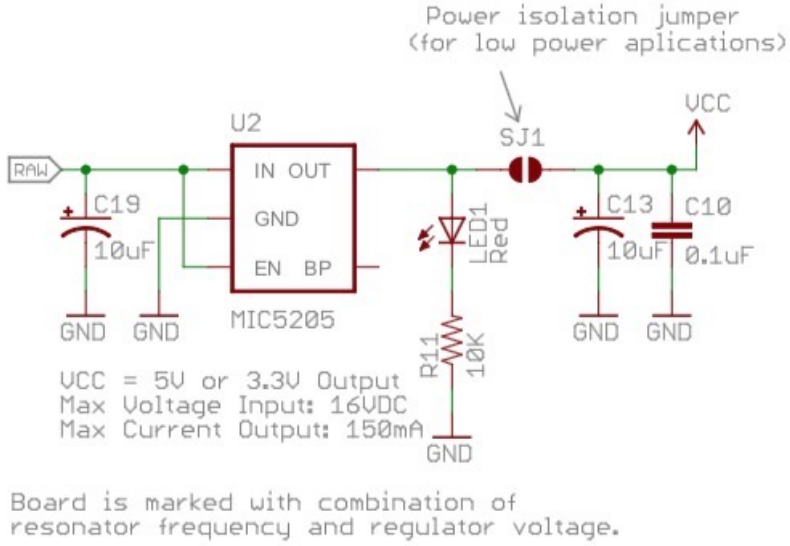


Figure 11.1: A segment of the Arduino Pro Mini schematic diagram.

In this example, you can see the schematic symbols of several components that make up the Arduino Pro Mini. The large rectangular symbol with designator 'U2' is a linear voltage regulator integrated circuit. Its model number is 'MIC5205'. The symbol contains several pins that provide inputs and outputs, and each of them is named.

You can also see three capacitors, two of them are polarised (designated C19 and C13) and one isn't (C10). You can also see a resistor, R11. The capacitors and the resistor also have their values marked in the schematic. There are also symbols for an LED, a jumper connector, and power (Vcc, RAW, and GND).

All of these symbols follow a particular standard. There are several standards available, but most notably engineers around the world tend to work with the American style ('IEEE') or the European ('IEC') style. The symbols in Figure 11.1 follow the American style.

[8] Get the full schematic here (last accessed 22/11/2018: https://cdn.sparkfun.com/datasheets/Dev/Arduino/Boards/Arduino-Pro-Mini-v14.pdf

In KiCad you can choose to use either the American or the European style symbols. Whichever one you choose, be consistent. Do not mix American-style resistor symbols with European-style capacitor symbols in the same schematic.

The KiCad standard symbols library contains symbols of both styles, though the European style symbols seem to be more plentiful. In most cases, American-style symbols will have the postfix 'US' in their name. In Figure 11.2, you can see an example of a resistor symbol, with the European style in the left and the American on the right. The name of the American style symbol ends in 'US', and you can take that into account when you are searching for a symbol in the Symbol Chooser.

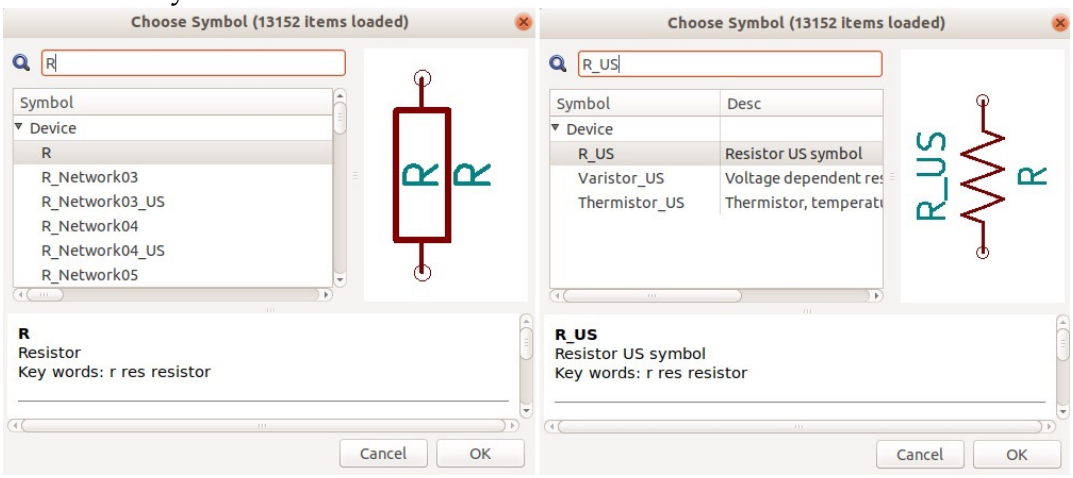


Figure 11.2: A resistor, European-style (left) and American-style (right).

12. PCB key terms

Creating printed circuit boards is an engineering discipline, and as such, it has its own 'language'. In this chapter, you will learn the most commonly used terms so that you can understand the information found in places such as PCB fabrication websites and CAD tool documentation.

Here we go.

12.1. FR4

The most common material used to make printed circuit boards is FR4 (or FR-4). This is a glass-reinforced epoxy laminate composite material, or in simpler terms, fibreglass cloth bound using an epoxy resin.

Go over to the Wikipedia article to read more about this material (https://en.wikipedia.org/wiki/FR-4).

The 'FR' part of the name stands for 'Flame Retardant', an obviously desirable quality for a board that will hold together components that can potentially ignite when they fail.

Other useful attributes of the FR4 are:

- Very light and strong
- Does not absorb water
- It is an excellent isolator
- Maintains its quality in dry and humid environments

Apart from the standard FR4 and variants (like FR4 tracking resistant and halogen-free), there are other materials that can be used in rigid or flexible printed circuit boards. Examples include High Tg (HTG) for applications that must operate in high temperates, paper-based materials with PF resins, aluminium, or various kinds of TG rigid and flexible boards.

12.2. Trace

Traces (also called 'tracks') are conductive paths. Most often they made of copper. Traces are used to transmit signals and power throughout a circuit.

In Figure 12.1 you can see the traces in the front side of this PCB as thin purple lines that provide the connections between the golden pads where the component terminals will eventually be. You will learn more about pads next.

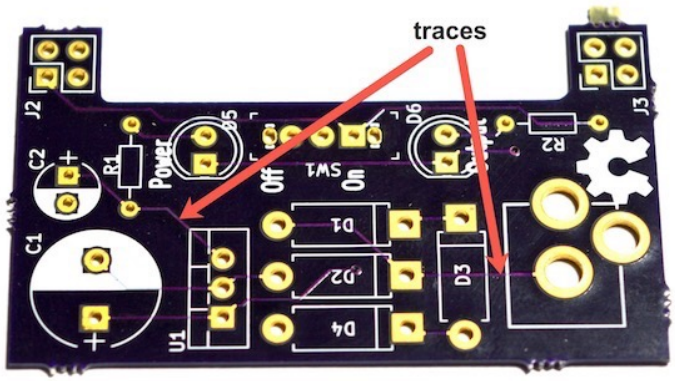


Figure 12.1: An example of traces.

As the designer of a PCB, you have total control over the characteristics of traces. You can control their width, height, and route, including the angles by which a trace changes direction. If you want a trace to be able to accommodate a large current flowing through it with little resistance and temperature rise, you can design it to be wider and thicker. This is useful when a trace is meant to feed power to the components of your board. Traces that are meant to convey low power-current signals (less than 20 mA) can be made narrower, with less copper.

Keeping the width of traces to around 0.3 mm (or even less, depending on your manufacturer's guidelines) makes it possible to draw traces closer together and reduce the final size of your PCB.

Figure 12.2: An example of wide traces.

In Figure 12.2, you can see an example of traces that are much wider than regular signal traces. These traces connect the terminals of a 240 Volt relay.

The traces in these examples are purple because of the solder mask chemical used to finish the manufacturing process. You will learn about the solder mask further down in this chapter.

12.3. Pads and holes

Pads and holes are the most prominent feature of a printed circuit board. Pads come in two varieties and in several shapes. There are TH (through-hole) pads and SMD (surface-mounted device) pads.

In Figure 12.3 you can see an example of a board that contains exclusively TH pads, and in Figure 12.4 you can see a board that contains TH and SMD pads.

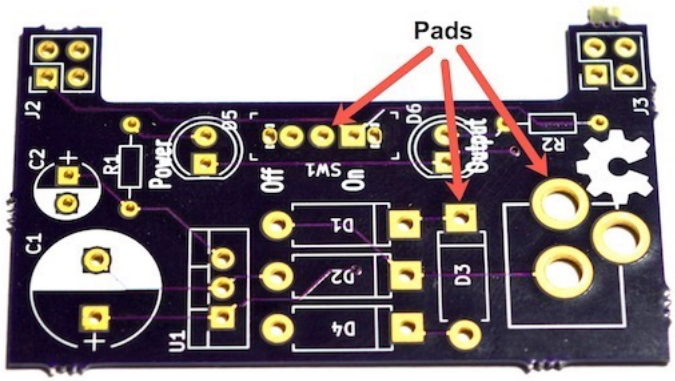


Figure 12.3: Through hole pads.

Through-hole pads, unlike SMD pads, connect the front for the PCB with the back electrically. In the examples, you can see that the gold plating of the pad fills the inside of the hole. If you turn the PCB around, you will see that a matching pad exists in the back.

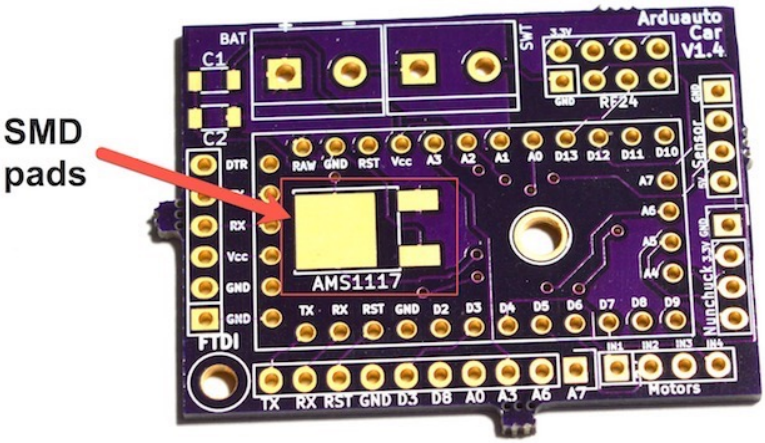


Figure 12.4: An example of SMD pads.

Boards with mostly TH pads are popular among hobbyists because through-hole components are easier to work with, at least in the beginning. SMD components are smaller; hobbyists tend to not use them until they are more comfortable with their soldering skills. I find that with a bit of practice, SMD components are as easy to work with as their TH counterparts.

In the industry, on the other hand, the vast majority of PCBs are designed to contain SMD components. This is because SMD components can be populated on the board automatically using pick and place machines and because their small size results in smaller PCBs.

Apart from the two varieties I described above, pads also come in several shapes. Most often you will see round pads, but rectangular and oval shapes are also possible. Using KiCad, you can create such pads and control their geometry to the extent that your PCB manufacturer allows.

In Figure 12.5 you can see an illustration of a cross-section of a PCB showing the configuration of pads, and two types of holes, Plated-Through Hole (PTH) and Non-Plated Through Hole (NPTH).

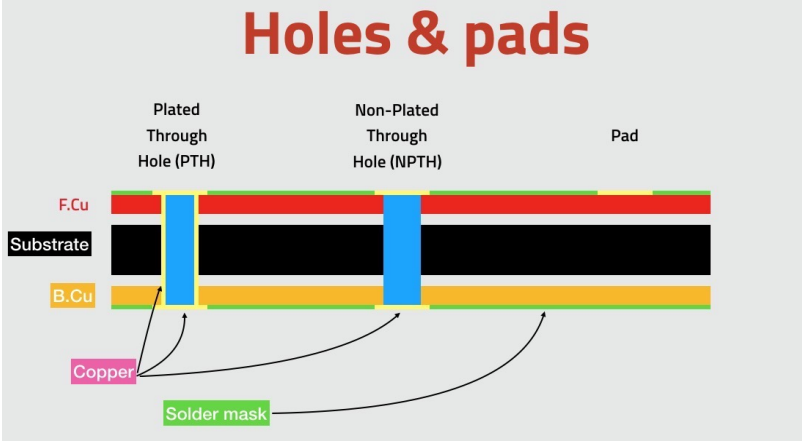


Figure 12.5: Pads and holes.

Plated-Through Holes is the more common variety and the default type of hole in more cases. A drill is used to create the hole, and then copper is used to cover the hole's sides so that its two ends (at the front and back copper layers) are electrically connected. Vias are constructed in the same way, except that they have a much smaller diameter so it is not possible to accommodate component pins.

On the other hand, in a Non-Plated Through Hole, we use the same drill to create the hole, but there is no copper used to cover the sides of the hole, so there is no electrical connection between its two ends.

Finally, pads without holes are useful for attaching surface-mounted component, as you learned earlier.

12.4. Via

When you want to move a signal that travels across a trace from one side of a PCB to another (say, from front to back), you can create a via. A via is a hole with its sides covered with copper or gold (or other conductive material), that allows a trace to continue its route across layers.

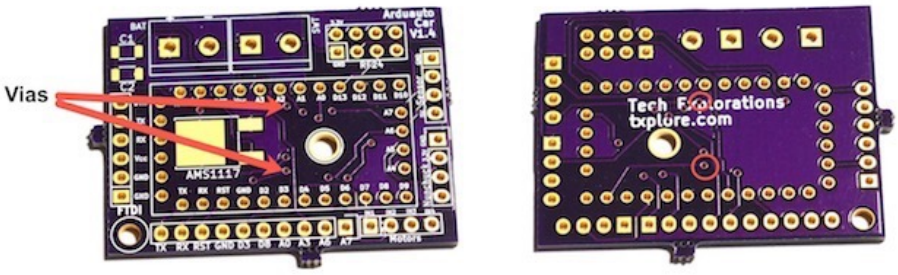


Figure 12.6: Vias allow a trace to continue between layers.

In Figure 12.6, you can see the two sides of the same PCB. On the left, the arrows point to two vias in the front of the PCB, and on the right, the circles indicate the same vias on the back of the PCB. Vias are very similar to through-hole pads, except that they don't have any exposed copper (they are covered by the solder mask), and they don't have a pad (so you can't solder a component).

In simple circuits with only a few components, it is possible to create all of the traces on one layer of the PCB. When a PCB gets busy with more components it quickly becomes impossible to do the routing on a single layer. When multiple layers are needed, vias provide the simplest method of allowing a trace to use the available board real estate.

In the Figure 12.7 you can see the types of interconnections between layers that are possible.

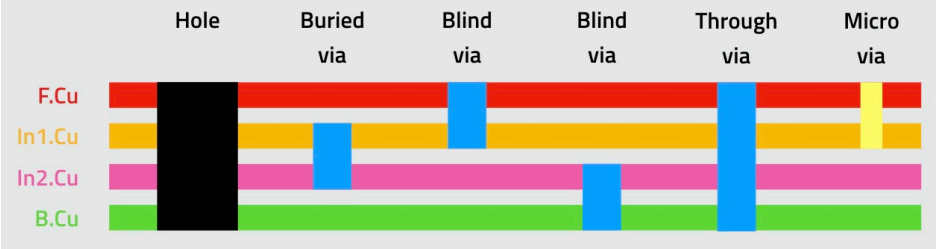


Figure 12.7: Types of interconnections between layers.

For through-hole components, you would design a hole that connects the top and bottom copper layers. This hole is implemented using a drill. It is wide enough to allow for the pin of the component to go through it.

Vias are smaller than holes in terms of their diameter. They are not wide enough for pins to go through them, but they are plated, like holes, and they allow for electrical connection between layers to take place.

A 'through via' is like a hole, but narrower. It connects the top and bottom layers. A buried via is a via that connects any two internal layers. In the four-layer example of Figure 12.7, the buried via connects the In1.Cu and In2.Cu. A 'blind via', also connects two layers, but has one end exposed on to the outside of the board, either top or bottom.

In high-density boards, another option for interconnecting layers is to use a 'micro via' (‘uvia’). A micro via is made using high-powered lasers, instead of a mechanical drill; the use of lasers makes it possible to dramatically reduce the diameter of the via.

12.5. Annular ring

The annular ring is a term that describes the area on a pad that surrounds a via. An important related metric is the width of the annular ring, which we define as the minimum distance between the edge of the pad and the edge of the via or pad hole.

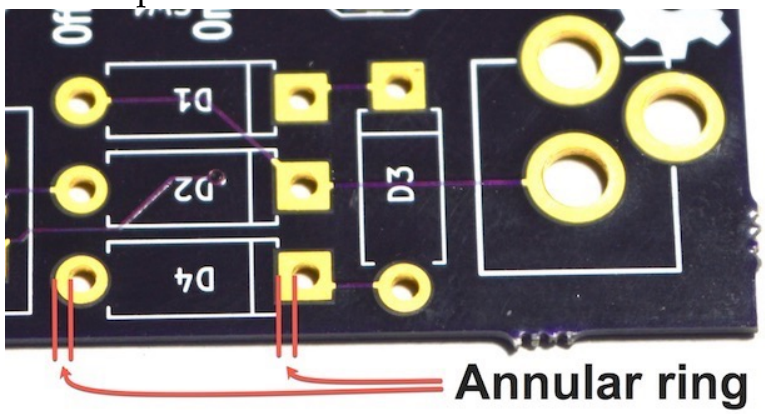


Figure 12.8: Annular rings and width.

In Figure 12.8, the width of two annular rings is indicated with the two red lines. Ideally, the drill hit (the location on the board where the drill lands and creates a hole) is right in the middle of the pad. This is what we usually want. If the drill bit is not aligned correctly, then the hole can be closer to one edge

of the pad (a 'tangency'), or it could even miss the pad completely (a 'breakout').

12.6. Soldermask

As you know, traces are made of copper. Copper slowly reacts with oxygen in the air, resulting to oxidisation. Oxidised copper produces a pale green outer layer. To prevent this from happening, PCB manufacturers cover the exposed copper with solder mask, a thin layer of polymer that insulates it from oxygen. As an additional benefit, the solder mask also prevents solder bridges from forming between pads.

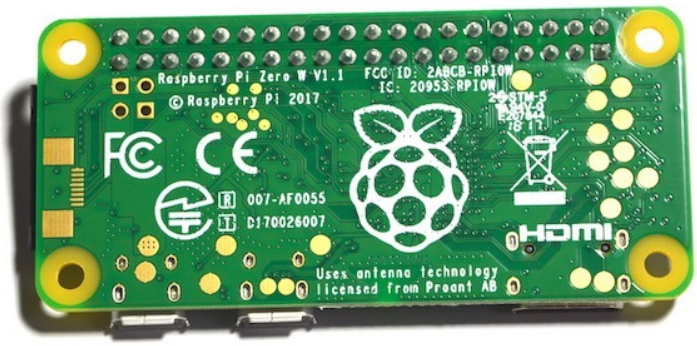


Figure 12.9: The rear of a Raspberry Pi Zero protected with a thin layer of solder mask.

In Figure 12.9, you can see the back of a Raspberry Pi Zero. In this example, the copper is protected by a thin layer of green solder mask. Only the pads and the mounting holes are not covered by the solder mask.

Solder mask polymers are available in different colours, with green being the most common and cheaper. You can create fancy looking PCBs with black, blue, red, purple and many other colours.

12.7. Silkscreen

Printed circuit boards are not complete without text and artwork that convey useful information and add a touch of elegance. In Figure 12.9 you can see an example of such text and artwork on the back of a Raspberry Pi Zero. You can see the Raspberry Pi logo, logos of various certifications, and various text items that inform us about the model etc. All this consists of the silkscreen.

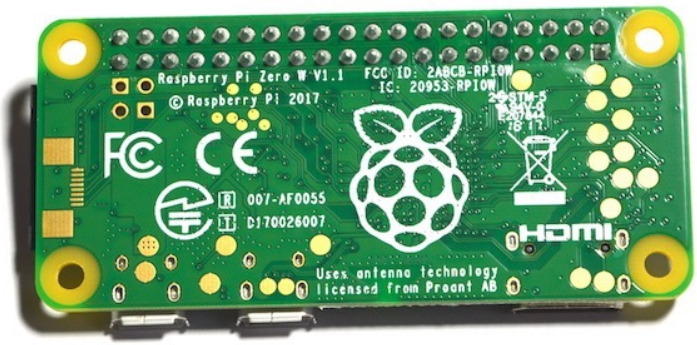


Figure 12.9: The white text and graphics on this Raspberry Pi Zero consist of the silkscreen.

The name 'silkscreen' is rather misleading. Of course, no actual silk is used to produce the white elements on the PCB. The method used to print the silkscreen in large numbers is a relative of the traditional screen printing process that you can use to print a graphic on a T-shirt. The silkscreen text and graphics are printed on the boards while they are still in their panels.

White is the most common colour for the silkscreen, but black and yellow are also available.

In the projects that you will work through later in this book, you will spend a considerable amount of time creating the informational and decorative text and graphics in the silkscreen layer of the PCB.

12.8. Drill bit and drill hit

Drill bits are used to create holes and vias, but also cutouts. Drill bits are typically made of solid coated tungsten carbide material and come in many sizes, like 0.3 mm, 0.6 mm and 1.2 mm. The look like the one in Figure 12.10. These drills are attached to computer controlled drilling machines and are guided by a file that contains information about the coordinates and the drill size for each hole on the PCB.

Figure 12.10: An example drill bit.

It is interesting to note that for very small holes, usually vias, drill bits are replaced with lasers. These vias are often called 'micro-vias'. With laser drilling, it is also possible to create vias that connect in-between layers of the PCB.

The term 'drill hit' describes the location on the PCB where the drill bit comes in contact with the PCB and creates a hole.

12.9. Surface mounted devices

If your objective is to create a PCB that is easy to manufacture in very large numbers, with a minimum size, then you should design it to contain surface mounted components instead of through-hole components.

In Figure 12.11 you can see an example of what is possible to do with SMD on a board. A full computer, on a tiny board, for a few dollars.

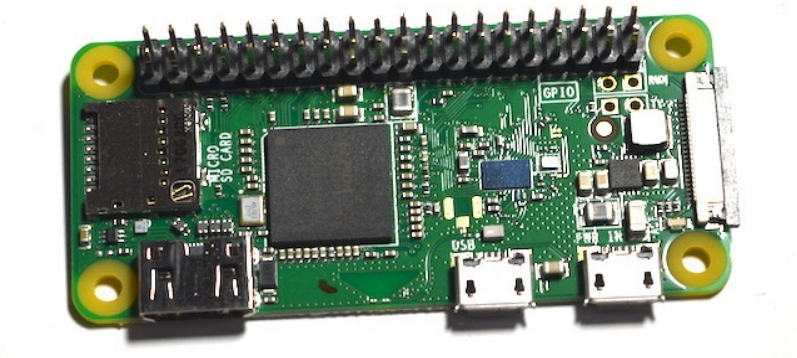


Figure 12.11: The Raspberry Pi Zero contains almost exclusively SMD components.

On this board are a highly integrated microprocessor, memory, communications, and connectors. Even the connectors are SMD. In fact, the only component that is through-hole is the header.

Creating something like this using through-hole components, if at all possible, would result in a board that was many times the size of the Raspberry Pi Zero, and would cost many times more because most of the assembly would have to be done by hand.

While hobbyists prefer to work with TH components because they are easier to solder and repair, learning to work with SMD, at least the larger ones, is certainly possible.

In this book, you will learn how to create an SMD version of a PCB, in addition to the TH version.

12.10. Gold Fingers

Appropriately called 'Gold finders' are gold-plated connectors placed on the edge of a PCB. Gold fingers are useful for interconnecting one board to another. You can see an example in Figure 12.12. This is the micro:bit educational single board computer.

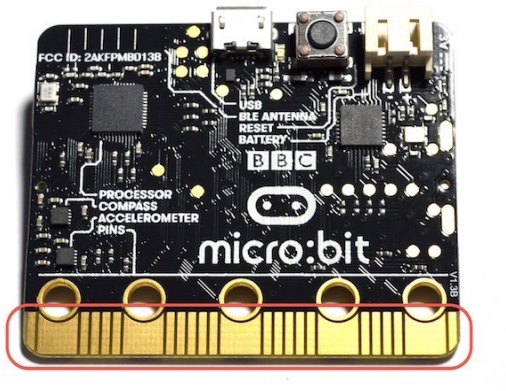


Figure 12.12: Gold fingers on a Micro:bit.

The micro:bit uses its gold fingers to connect to other devices, like motor controllers and sensors, via a slot. Gold fingers, when manufacture properly and with sufficient thickness, makes it possible to attach and detach the PCB to a slot at least 1,000 times before they start to wear out.

12.11. Panel

To manufacture PCBs economically, manufacturers use machines that can work on large panels. Each panel can be designed to contain many copies of the same PCB. It is also possible to use clever algorithms that place different PCBs on the same panel so that the capacity of the panel is fully utilised, and that the individual cost of each PCB is reduced. This is how it is possible to have a single 'hobby' PCB manufactured for a few dollars. This panelisation process is key to this reduction in costs.

In the example of Figure 12.13, a single panel contains four individual PCBs. The four PCBs are populated while they are still part of the panel using an automated pick and place machine. A pick and place machine is a robot that uses an arm to pick each component from a container and places it precisely on the pads. Once the components are on the board, the panel moves into the next step of the process in which they are 'baked' and secured in place.

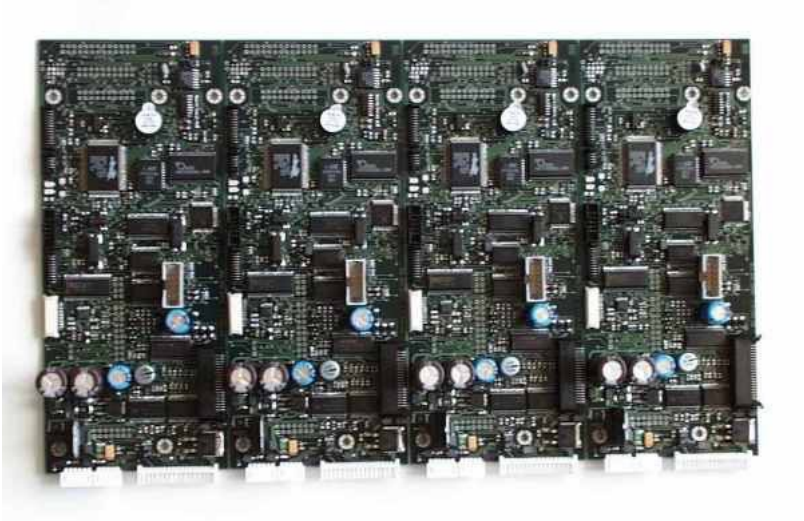


Figure 12.13: A panel with four individual PCBs.

To remove the PCBs from the panel, manufacturers utilise defined breakaway routes and points on the board to snap them off. In Figure 12.14 you can see the breakpoints along the edges of this PCB.

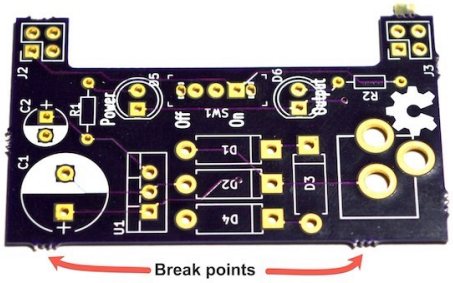


Figure 12.14: This PCB used to be part of a panel.

Using a drill, the manufacturer removed the substrate material in between the breakpoints so that with a small amount of force, individual PCBs can break free from the panel without damage.

12.12. Solder paste and paste stencil

Solder paste (or solder cream) is a soft and sticky material (at room temperature) that is applied on pads in preparation to attaching a component. Think of solder paste as normal solder. While with normal solder you will need a soldering iron to heat it, melt it, and apply it on a component pin that is already in place, with solder paste you will first use a syringe (or one of the other application methods) to cover the pad, then place the component on the pad, and provide heat in the form of an oven to heat the paste and bond it with the pad and the component's plated area.

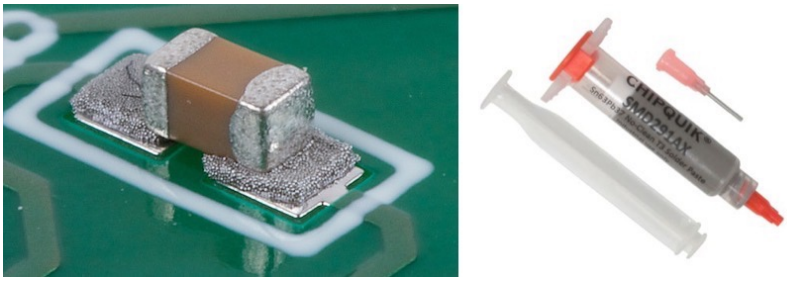


Figure 12.15: Solder paste in a syringe dispenser, and an SMD component.

In Figure 12.15 you can see an example of a solder paste in a syringe dispenser, that you can purchase from retailers like RS Components. Using the syringe equipped with a thin nozzle, you can manually deposit a small amount of solder paste on the pads. Using tweezers, you can place the component you want to attach on the solder paste. Because solder paste is sticky (before it's baked), the component will attach to it. Once you have all the components you want on the board, you can place it in an oven to bake it. After the baking process is complete, the solder paste will be solid, like normal solder. The SMD components will be mechanically secure and electrically connected to the pads.

Solder paste also comes in a tub, which is more appropriate for application to a board using a stencil (Figure 12.16). Stencils are useful in large-scale productions.

Figure 12.16: Solder paste is a tub container.

A stencil, typically made of stainless steel, is cut so that it has openings of the exact size and the exact location of the pads of the board. The technician will place the stencil over the board, and then apply the paste on the openings. When the stencil is removed, the paste remains on the pads only.

Then, either manually or using an automated pick and place machine, the components are placed on the pads and stick on them because of the

paste. The last step is to bake the board in a reflow in order to solidify the paste.

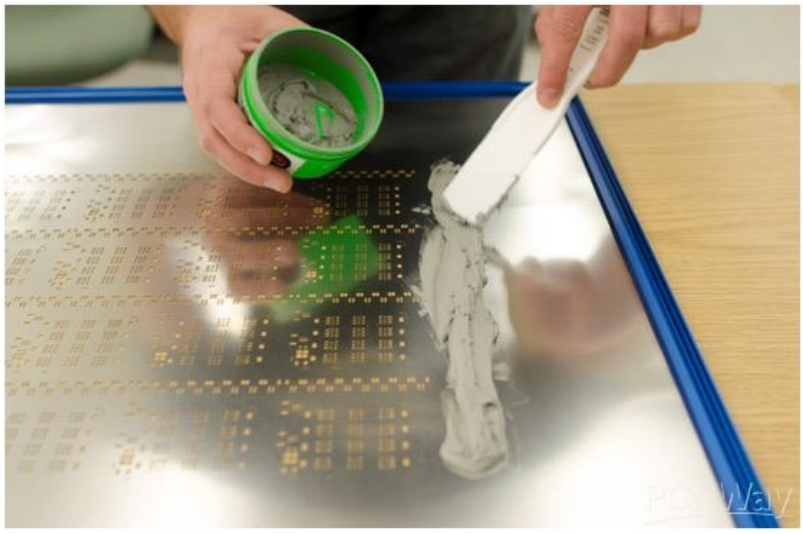


Figure 12.17: A stencil, with solder paste being applied using a squeegee (photo courtesy of Pcbnew.com).

A reflow oven is an industrial-sized machine that is used to complete the process of attaching SMD components on a PCB. You can also purchase or make a reflow oven for use at home. People have even made reflow ovens for their projects using discarded toasters. In either case, a reflow oven is designed to operate under a specific program that controls the amount of heat a board receives over time. This is important because the heat must be appropriate for the purpose of converting the solder paste into good-quality electrical connections, without causing damage to the board or the components on it.

12.13. Pick-and-place

Pick and place machines are robots that assemble the various components on the surface of a circuit board. When you contract a manufacturer to not only make your boards but also to populate them, they will be using a pick and place machine. You can see an example of a pick and place machine in Figure 12.18.

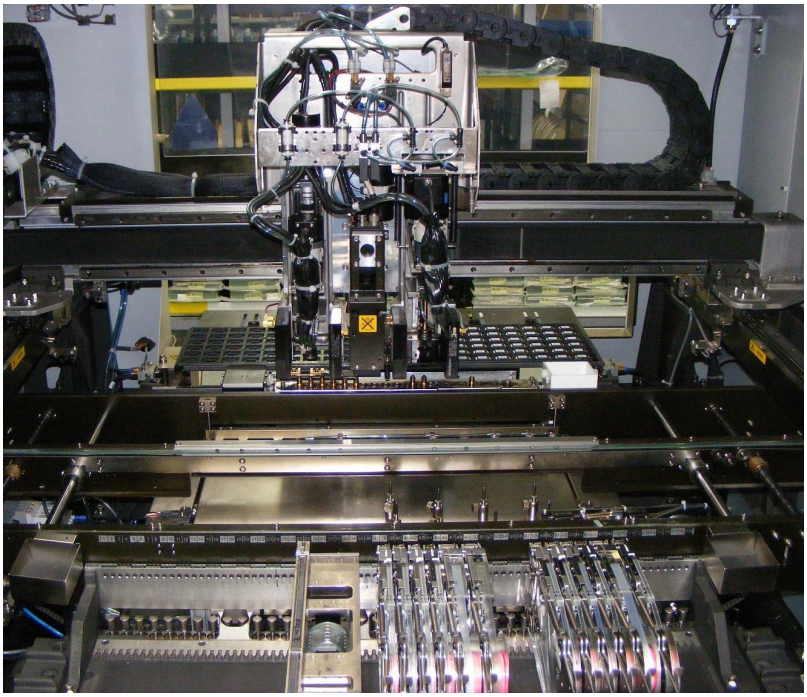


Figure 12.18: A large pick and place machine (By Peripitus [GFDL (http://www.gnu.org/copyleft/fdl.html) or CC BY-SA 3.0 (https://creativecommons.org/licenses/by-sa/3.0)], from Wikimedia Commons).

A typical pick and place machine, like the one in Figure 12.18, includes:

1. a repository of the various components that are to be placed on the board,
2. a conveyor belt that brings in the boards,
3. an inspection system composed of cameras that can optically recognise the board, components and other guidance markings on the board,
4. a robotic arm that can pick a component from the repository and place it on the board (these arms are usually fitted with suction cups so they can pick and manipulate components).

Modern high-end machines are very versatile, optimised for short runs of complicated boards, and artificial intelligence designed to assembled and test boards ensuring high levels of reliability.

13. Schematic design workflow

In Part 1 of this book, you learned about the PCB design workflow, without getting into any details. It is time to take a closer look at the two most important parts of the process: the schematic design, and the layout. In KiCad, the schematic design is done using Eeschema, and the layout is done in Pcbnew.

Let's learn about the workflow of creating a schematic in Eeschema. You can see it in Figure 13.1.

But before we get into it, I'd like to highlight something very important: I designed the workflow that you see in Figure 13.1 to help you take your first few steps in PCB design. As you gain confidence and knowledge, you will develop your own workflow. While you should follow the workflow in Figure 13.1, know that you are by no means 'locked' in it. KiCad is very versatile and can accommodate a huge variety of working styles. It can also work in tandem with other tools in your toolchain, such as other CAD applications, and autoroutes (both of which you will learn about in this book).

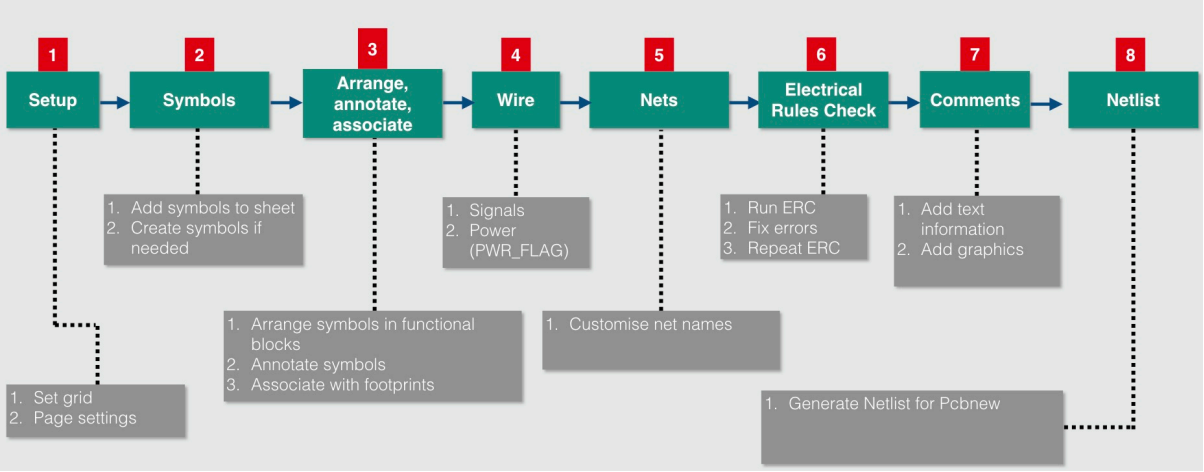


Figure 13.1: The schematic design process.

In addition, engineering and design is a massively iterative process. Linear simply doesn't work. Yes, the workflow you see in Figure 13.1 is obviously linear, but that's ok because you will abandon it in favour of an iterative model before you finish working with this book.

Let's have a look at the 8 steps of the schematic design workflow.

13.1. Step 1. Setup

When you start to work on a new schematic in Eeschema, there are two things that you will want to do: configure the grid size and set up the page. The Setup in Eeschema is simpler in that it is in Pcbnew, as you will see.

You can set the grid size from the Display tab of the Schematic Editor Options window, which you can find in the Preference menu (Figure 13.2).

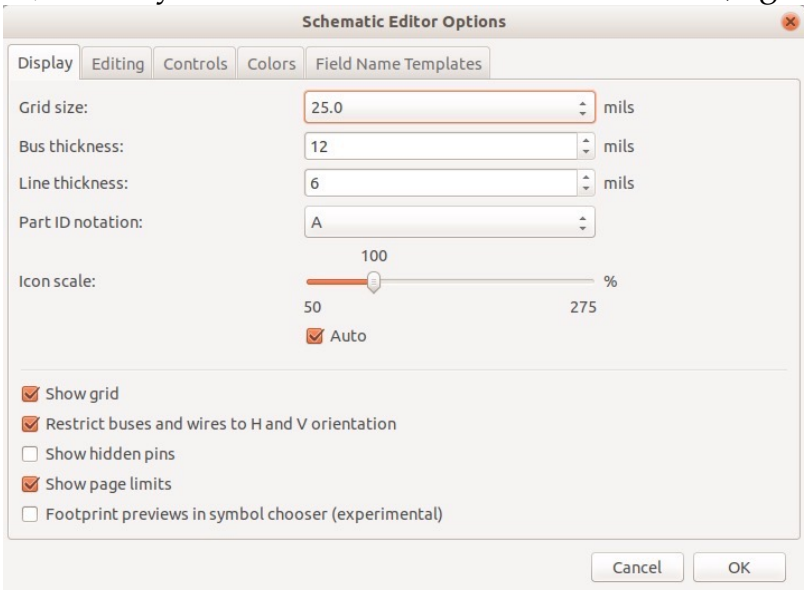


Figure 13.2: You can set the grid size from the Display tab.

The grid size determines where your mouse pointer will snap on the schematic sheet. The snap feature helps in the drawing process so that the end result looks good. The default setting is 25 mils, which I find good for most projects I have worked in.

You should definitely take a few minutes to set up your page in the Page Settings window, available from the File menu (Figure 13.3).

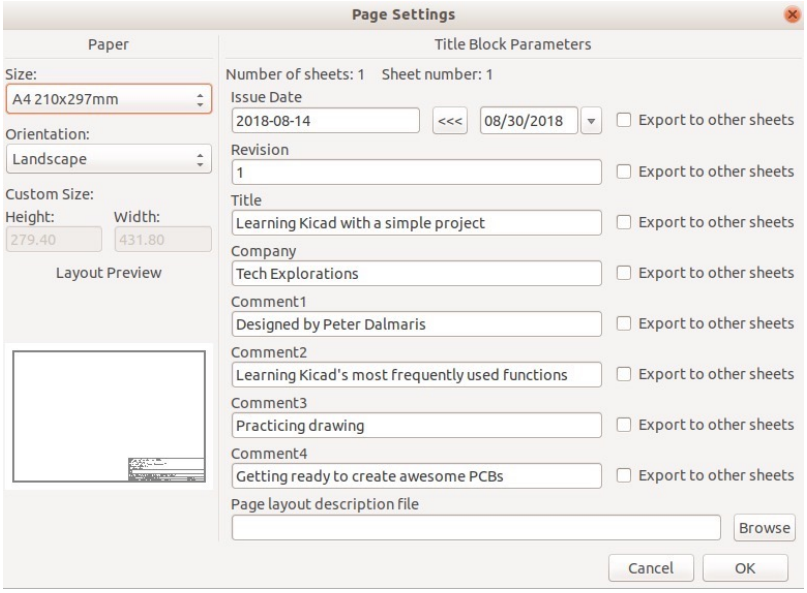


Figure 13.3: The Page Settings window.

As in my example in Figure 13.3, you should type in the details of your project, such as the date (you can copy the current date into the field by clicking on the '<<<' button), the revision number and a title. When you print out your schematic (or export it as PDF to share it with others), this information will also be printed. In Pcbnew, you will find the same Page Settings window, as you will see in the next chapter.

13.2. Step 2. Symbols

The two most important elements of schematic diagrams are the symbols and the electrical connections between the symbols. The symbols are graphical representations of real-life components. For example, a resistor looks like the symbol in Figure 13.4.

Figure 13.4: The symbol for the resistor.

Assuming you know what you are designing, in the second step of the workflow involves getting all the needed symbols from the symbol chooser and adding them to the schematic sheet (Figure 13.5).

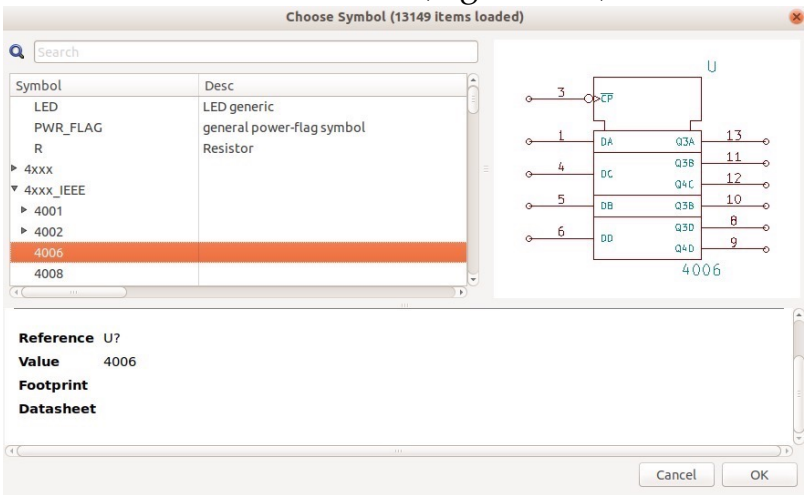


Figure 13.5: Use the symbol chooser to find symbols.

KiCad has a huge variety of symbols and symbol libraries to pick from. On top of that, you can create your own custom symbols and your own libraries. Those custom libraries can contain your custom symbols or symbols that you have collected from other sources.

In this step, simply get all the required symbols on the sheet. If any are missing, create them yourself (you will learn how to do that in this book), or find them by searching on the Internet. Once you have all of them, continue to step 3.

13.3. Step 3. Place and annotate symbols

In step 3, you will move the symbols in place, and prepare them for wiring. In Figure 13.6, I have placed the symbols in the approximate position, in relation to the other symbols, so that I can easily wire them in the next step. I want to keep the wires short, with as few angles as possible in order to produce a readable outcome.

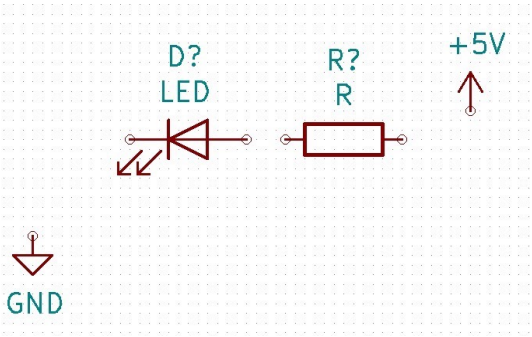


Figure 13.6: Move the symbols in place.

Before you move on to the wiring step (step 4), you should also annotate the symbols. Notice how the LED and the resistor have designators 'D?' and 'R?' in Figure 13.6? The question marks indicate that these symbols have not been annotated. With annotation completed, the schematic will look like the example in Figure 13.7.

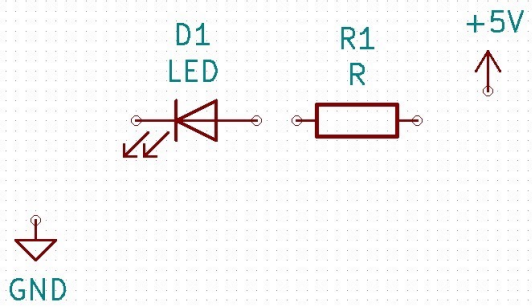


Figure 13.7: The symbols are now annotated.

KiCad can annotate all symbols with the click of a button, as you will see later.

With your schematic symbols annotated, you can move on to Step 4, wiring.

13.4. Step 4. Wire

In step 4, you will connect the symbols using wires. Each wire is attached to a symbol pin and creates a net. In the example of Figure 13.8, the green wires connect the LED and the resistor to the 5V voltage source and the GND. The 5V and GND symbols are special kinds of symbols, 'power' symbols. This schematic contains three wires, so it contains three nets.

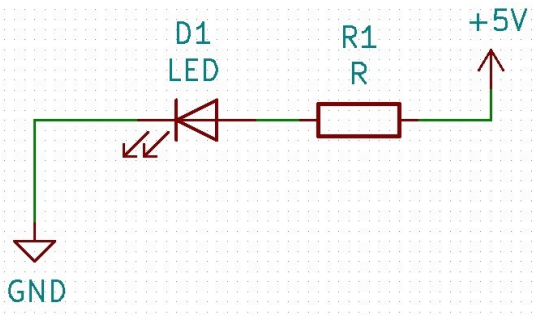


Figure 13.8: The symbols are connected using wires.

Understanding nets is important because of what you can do with them in Pcbnew. The ones in the example here don't have a custom net name, but they do have an automatically assigned name that Eeschema has generated. With or without a custom net name, this schematic is fully wired, so you can continue in the next step of the workflow, where you will actually assign custom names for these nets, or at least the most important nets.

13.5. Step 5. Nets

In step 5 you will name your schematic nets, at least the most important ones. Nets that are important are, for example, those that connect to the GND and 5V symbols. By giving them a custom name to replace the generic one assigned by Eeschema, you will make it easier to work with it in Pcbnew. Typically, power nets are implemented with wider traces to allow them to accommodate higher currents. Other examples of 'important' nets are those that implement antennas, and data or address busses. In either case, the geometry of those nets, once implemented as traces is important, and having a custom name makes their manipulation easier.

The analogy from the world of programming is this: imagine you have a variable that is used as a counter of the number of users participating in a forum discussion. This variable could have any time you like, as long as it is valid. You could call it 'number_or_forum_participants' or 'var32'. Which one would you rather use?

In Figure 13.9 you can see two labels attached to the wires that connect the circuit to the GND and 5V symbols.

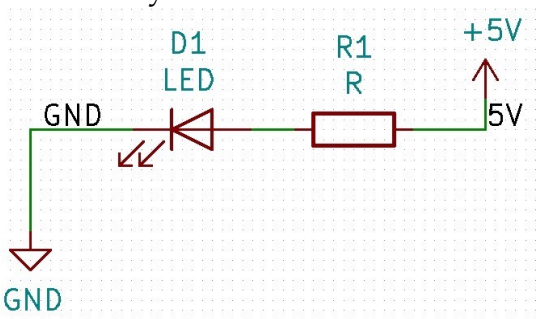


Figure 13.9: This example shows two labeled nets ('GND' and '5V').

13.6. Step 6. Electrical Rules Check

In more realistic circuits than the one with the LED and resistor symbols we have been using as an example, it is a good habit to regularly check for errors. In programming, the equivalent is to regularly run the compiler every time you complete a new segment of code. The compiler will produce messages to draw your attention to defects. You will need to fix those defects before you continue to write new code.

In Eeschema, this kind of check is done by the Electrical Rules Check utility (ERC). The ERC will check for a variety of error conditions, like unconnected pins, problems with power pins, problems with pins connecting to incompatible pins, etc.

In Figure 13.10, the ERC revealed two problems with this schematic. This is one of the most common problems that you will come across to. You learned about this in Part 2, in the section with the title 'Electrical rules check (ERC)'.

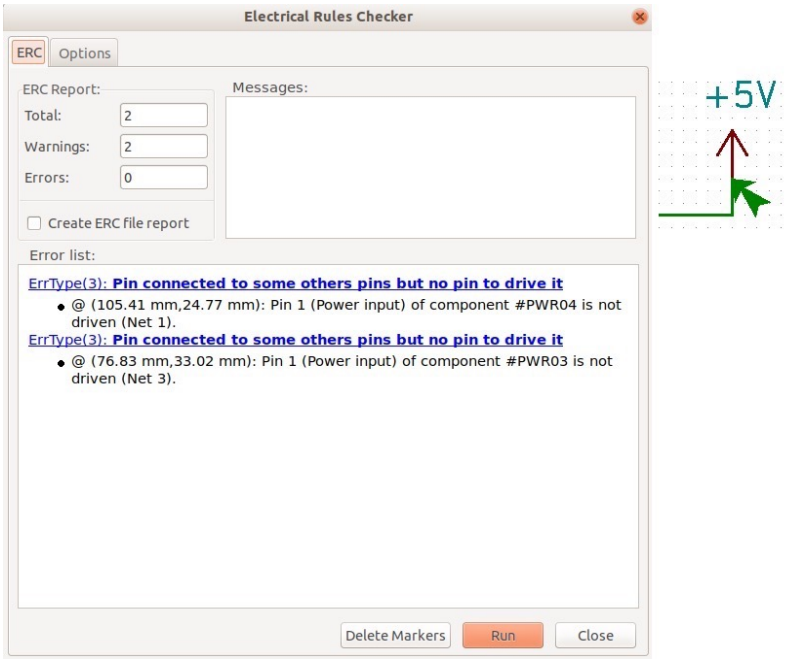


Figure 13.10: This ERC revealed two problems with this schematic.

13.7. Step 7. Comments

Let's continue with the programming analogy. Just like in programming comments are very important to increase the value of your code, similarly, in CAD design and in particular in schematic and layout design, you can use

comments for the same purpose. I try to be liberal with comments, especially when I work on schematics and layouts that I plan to share with other people.

Comments come in the form of text labels that highlight features of the schematic, and graphics, such as lines and boxes, that help group segments of the schematic into functional components.

You can see an example in Figure 13.11. I may have gone a bit 'overboard' with my comments, however, the result is that for the reader/learner it is very clear as to what each group of symbols is, and how it relates to the other symbols.

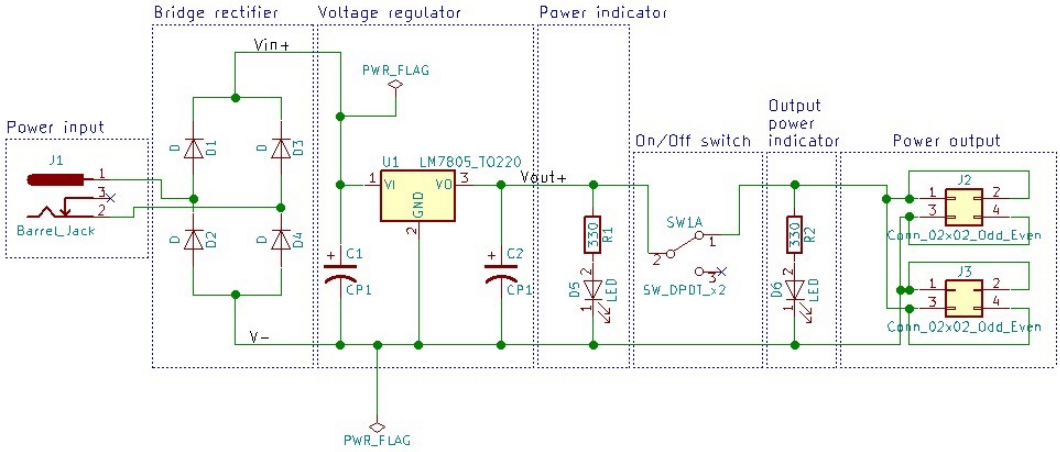


Figure 13.11: This schematic is commented to improve comprehension.

13.8. Step 8. Netlist

Your work in the schematic design of your board was complete in Step 7. The only thing left to do in Eeschema is to export the necessary schematic data to Pcbnew, so that you can continue with the layout process. In other CAD applications, a tighter integration between the schematic design and the layout components means that you can jump directly from one to the other. But in KiCad, the two components are separate. To start work on the layout, you need to export the 'netlist' file from Eeschema, and import it into Pcbnew (Figure 13.12).

I find that the separation of the schematic and layout editors in KiCad is a positive design decision. It helps in reinforcing the fact that these two operations are, indeed, separate, and need focus and planning to perform each one well.

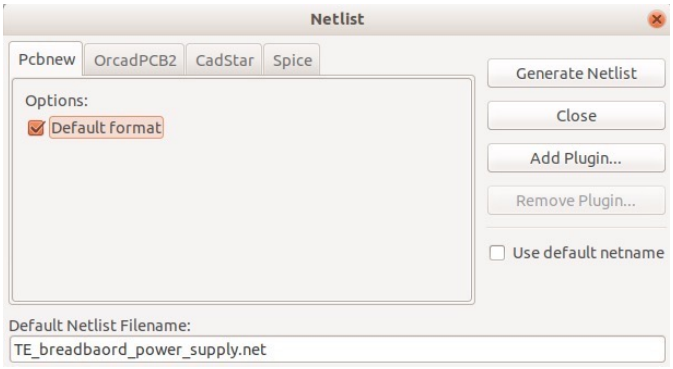


Figure 13.12: Export the netlist from Eeschema prior to starting work on the layout.

The separation between schematic and layout designs means that you must be forward-thinking and disciplined when you are working with the schematic. Imagine that you are working on the layout when you realise that you forgot to place a switch or a pull-up resistor in the schematic. You will have to go back to Eeschema, add the missing symbols, re-export the Netlist, re-import the Netlist to Pcbnew, and start the layout again. If only you had not forgotten in the first place! After a few times of this, I found myself becoming better at thinking ahead and planning during the schematic design work. In turn, this meant fewer iterations, and faster start to end PCB design duration.

Nevertheless, starting with KiCad 5, this overhead is shortened, since a shortcut was introduced by its developers. You can update the netlist from the Pcbnew Tools menu (Figure 13.13).

I still find myself clinging to the old ways of doing things, but I must say that with this shortcut, the integration between Eeschema and Pcbnew is tighter than ever before, without losing the advantages of having separate applications.

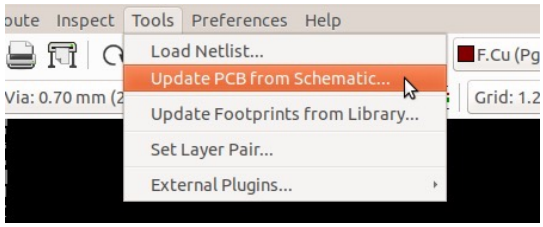


Figure 13.13: Quick import of the netlist into Pcbnew is now possible.

14. PCB layout workflow

In Part 1 of this book, in the chapter titled 'The PCB design process in KiCad', you saw a high-level view of the process of creating a PCB in KiCad. The process starts with the design of the schematic diagram that describes your circuit using Eeschema, and concludes with the layout of the physical components of the PCB using Pcbnew. KiCad contains other tools that assist us in this process, like the footprint and component editors and Cvpcb which allows us to associate components to footprints.

In this chapter, we'll focus on the PCB layout step of the process. In KiCad, this is the step that involves Pcbnew. This is because I perceive the layout step as the one that uniquely shapes the final 'real world' product. What it will look like, how well it will function, how 'manufacturable' will it be, how durable it will be; all this depends on the layout step.

The process I describe in this chapter is applicable to any PCB layout tool, although I will be using KiCad terminology where necessary to make it easier for the reader to associate the process with the functionalities inside Pcbnew.

You can see the process in Figure 14.1. For simplicity, as I mentioned in the chapter on the schematic design workflow, I have rendered it to look linear. In reality, the process is iterative. It is often necessary to move, for example, from step 3 back to step 1 in order to adjust the grid so that parts fit better in the available space. Having said that, this diagram should help you understand the steps through which every PCB design has to go through. In this chapter, I will describe each of those steps. Once you apply this process to the projects in Part 4 of this book, it will become 'natural'.

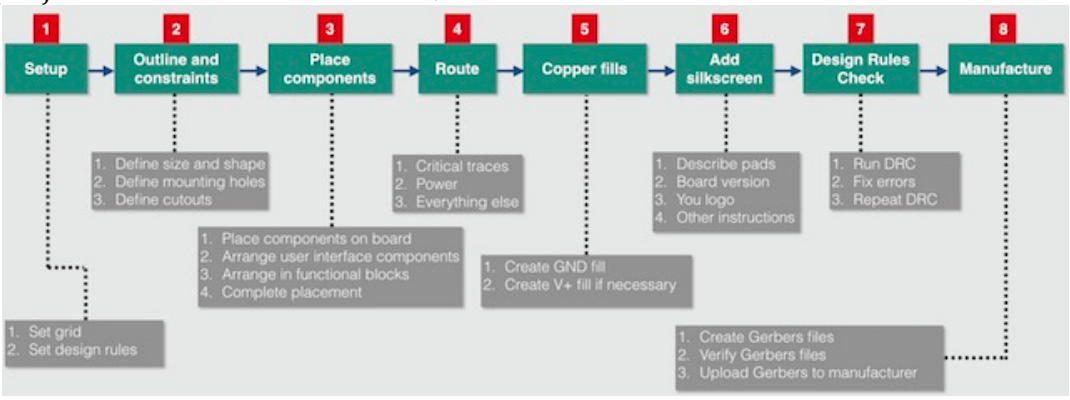


Figure 14.1: A serialised view of the PCB layout process.

14.1. Step 1. Setup

In the setup step, you configure the Pcbnew drawing grid size, the layers and set the design rules for the layout. The grid size and number of layers depends on the dimensions and complexity of your PCB. The design rules are governed mostly by the constraints set by your PCB manufacturer, but also by the requirements of your project. Let's start with the grid first.

The grid provides assistance with the placement of footprints, drawing of traces, and the drawing of the cutout (boundary) of your PCB. Pcbnew provides multiple grid sizes, but you can also define your own custom size.

When you start a new project, select the grid that is most appropriate for the design of the board's outline and the placement of the footprints. Typically, we will choose a larger grid size for the outline, and a smaller one for the footprints. We may go for a third, smaller grid size to help with the drawing of the traces as this often needs to negotiate tiny details on the PCB, like going around pads and vias.

Larger grids produce more coarse drawings.

Let's say that you want to produce a rectangular PCB that measures 50 mm by 30 mm. On it, you want to place a few resistors, LEDs and headers. A reasonable grid choice for the outline and the larger features is 1.27 mm, and half that (0.635 mm) for the components. When you start work on the traces, you can consider going one level down from 0.635 mm, to 0.508 mm or even 0.2540 mm if there are too many pads to connect.

The larger grid will allow you to draw the cutout of the board in precise 1.27 mm segments. With this grid, you will be able to draw the long side (50 mm) with 39 x 1.27 mm segments, for a total length of 49.53 mm, or with 40 x 1.27 mm segments for a total length of 50.8 mm. If you need to go for an exact 50 mm side, then you can define a custom grid side, say 0.5 mm, using the Grid Setting dialog box (Figure 14.3), and select it from the Grid drop-down menu (Figure 14.2).

Once you are finished drawing the cutout, you can switch to the smaller Grid side and continue by placing the footprints inside the board.

KiCad makes it easy to switch between two grid sizes by using the grid fast switching shortcuts. The default shortcuts are Alt-1 and Alt-2, although I have changed this to something else to avoid conflict with the Ubuntu operating system's Alt-[X] shortcut that allows fast switching between applications (where 'X' is the number of the application on the taskbar).

To define which grid size you want to work with, use the Grid drop down menu (Figure 14.2).

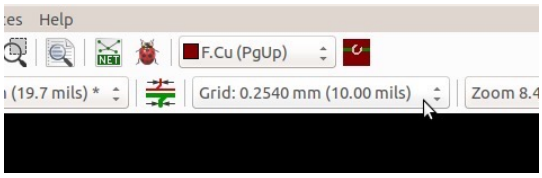


Figure 14.2: Select your working grid from the top menu.

To define the two Grid Fast Switching sizes, and to set your custom grid sizes, open the Grid Setting dialog box by clicking on View, Grid Settings (Figure 14.3).

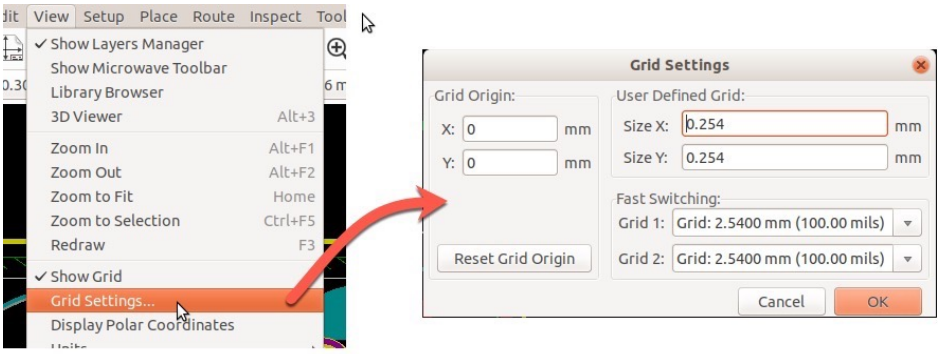


Figure 14.3: The Grid Settings dialog box.

After the grid size, you need to define and confirm the number of layers you want to use and the design rules. In regards to the number of layers, the vast majority of hobbyist projects seem to be best implemented with two layers. The best online manufacturers have optimised their processes to work with two layers, both from a cost and also quality perspective. Even if you create a single-layer PCB design, these manufacturers will use a two-layer process on it, and therefore the cost will be the same. If you are creating a more complicated PCB, you can setup Pcbnew for more than two layers. You can select the number of layers for your PCB from the Layer Setup dialog box, that you can reach from the Setup menu (Figure 14.4).

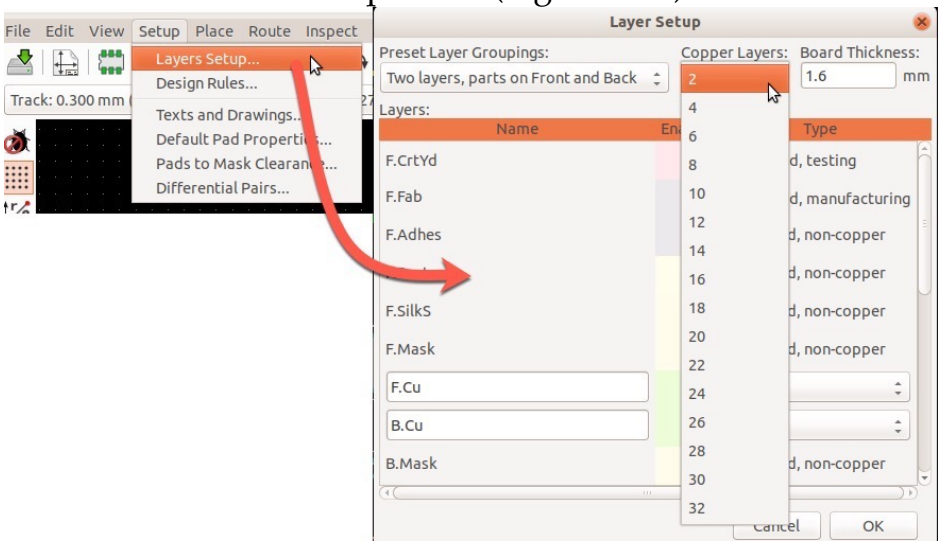


Figure 14.4: KiCad can produce PCBs with up to 32 layers.

The last item in the Setup todo list is to define and confirm the Design Rules. The design rules that you want your PCB to comply with are dictated by your project's technical requirements but also by your preferred manufacturer's capabilities and guidelines. To set up your project's design rules, open the Design Rules Editor from the Setup menu. In Figure 14.5 you can see the editor with the default values for one of the projects in this book.

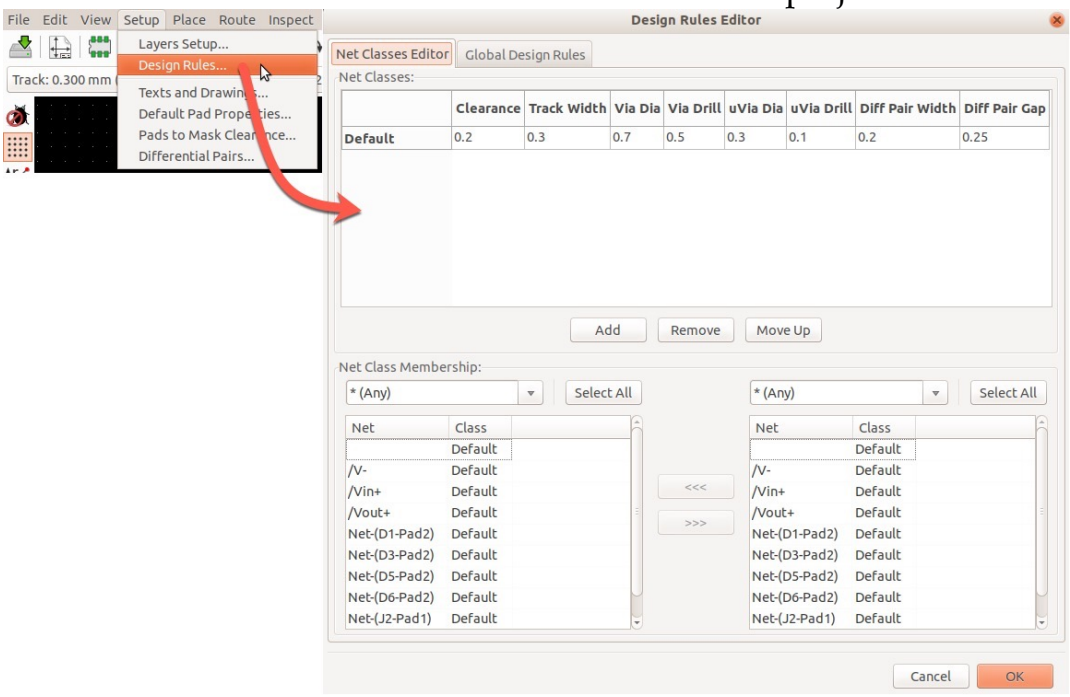


Figure 14.5: The Design Rules Editor with its default settings.

When starting a new project, it is a good habit to confirm that at the very least, the design rules are compatible with the guidelines set by your preferred manufacturer. The best online manufacturers publish these guidelines on their website. For example, OSHPark has a web page[9] with design rule information specifically for KiCad, where we learn that the minimum track (trace) width is 0.006' and the minimum via diameter is 0.027'. PCBWay also publishes this information in a page on their website,[10] where we learn that their minimum track (trace) width is 0.1 mm and minimum drill size is 0.2 mm. Beware of the units that manufacturers report these numbers as they may be in imperial units (inches), metric (mm), or in mil.

[9] https://docs.oshpark.com/design-tools/kicad/kicad-design-rules/
Accessed November 19, 2018

[10] https://www.pcbway.com/capabilities.html
Accessed November 19, 2018

You should ensure that the values in the Design Rules editor, both in the Net Classes Editor tab and in the Global Design Rules tab, are equal or larger to those that your manufacturer specifies as a minimum. If you want to be able to manufacture your boards with multiple manufacturers, then you must ensure that your design rules comply with the largest minimum of their requirements.

The default values in the Design Rules Editor are larger than the minimum values of both OSHPark and PCBWay, so I usually don't make any changes to them. I do, however, add at least one new row in the Net Class Editor tab to define larger track widths and vias for power tracks as these convert more current compared to the default (signal) tracks. You will learn how to do this in the projects. There is also a recipe in Part 5 where you can learn how to do this quickly.

14.2. Step 2. Outline and mechanical constraints

The next step in the process is the definition of the board outline. This outline defines the shape of your board. It is good practice to define the shape of your board before you add any components to it so that you can ensure that it will fit properly within the confines of a project box or other mechanical constraints.

Apart from defining the shape and the dimensions of your PCB, this is the step where you must also define fixed features such as mounting holes and cutouts. Again, these items must match your project box or other external mechanical constraints. If you don't deal with these issues now, it is likely that you will have to relocate components and traces later, a much more tedious exercise. In Figure 14.6 you can see the board outline from one of the projects in this book. I have unchecked all layers except for Edge.Cuts to make is easier to see the yellow line of the board outline.

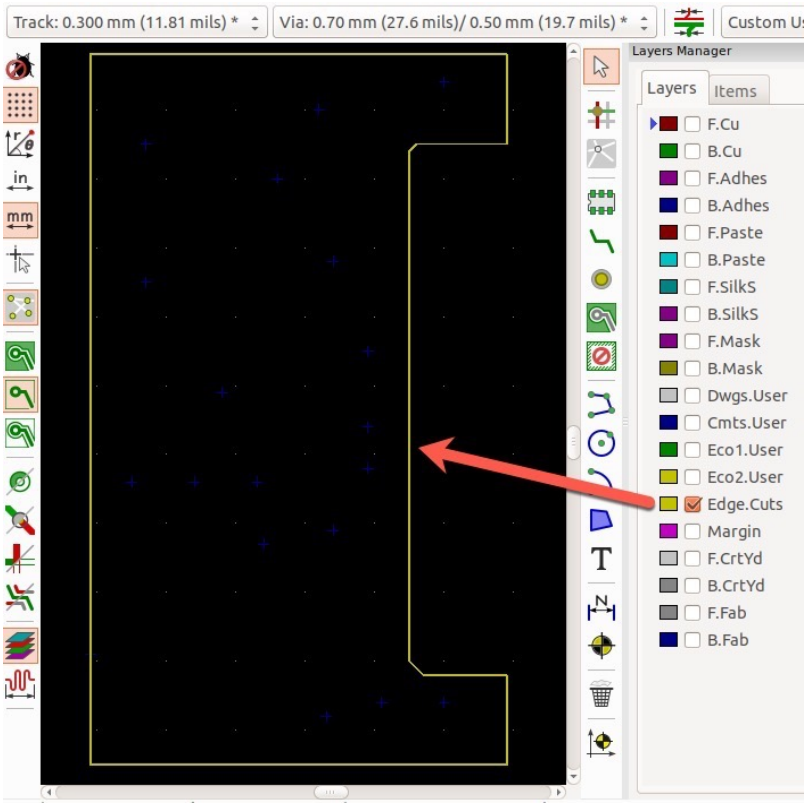


Figure 14.6: Define the board outline in the Edge.Cuts layer.

To define the outline and the cut outs of a board, select the Edge.Cuts layer from the layers manager and use the graphics tools to draw it. As mentioned in the Setup section, I prefer to use a large grid size while I am drawing the outline.

The easiest shape you can draw for your PCB is rectangular. However, using Pcbnew's Arc and Circle graphics tools you can create elaborate non-rectangular shapes that will make your PCB stand out. In the projects in this book, we'll use these tools to create rounded corners and other interesting features. If you can't wait to learn how to do this, you can jump to the 'Irregular shaped PCB's' recipe in Part 5 of this book.

To create mounting holes and cutouts, you can choose one of a couple of methods. You can create them in the Edge.Cuts layer, or using pads during the component placement step. If your preferred manufacturer supports the first method, then I suggest you go with that as it is faster and helps to keep your design clean because you are not mixing the electrical and mechanical aspects. See an example in Figure 14.7.

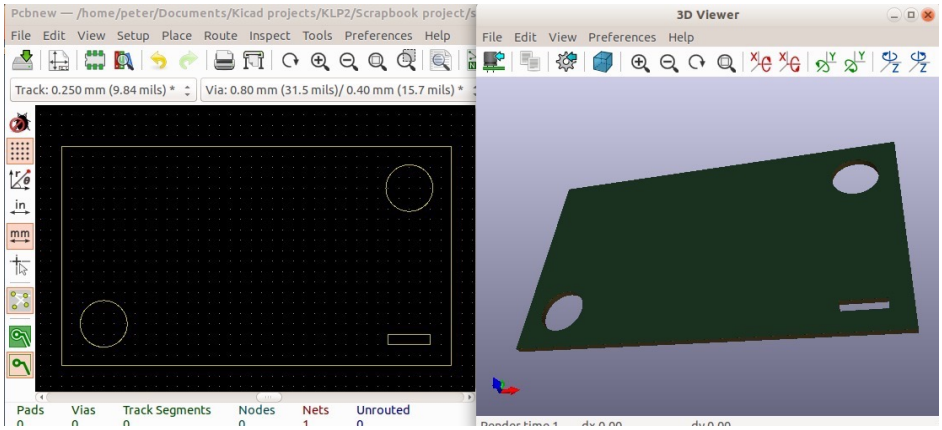


Figure 14.7: You can create openings within the outline of the PCB in Edge.Cuts if your manufacturer supports this method.

In this example, I have created a simple rectangular PCB. Within it, I used the circle graphics tool to create two round openings (screw holes), and the polygon tool to create a small rectangular opening. You can learn more about how to create openings by jumping to the Creating mounting holes and openings recipe.

14.3. Step 3. Placement of components

After we have defined the mechanical characteristics of our PCB we can proceed by placing the components (called 'footprints' in Pcbnew) on it. Like everything in engineering (and in life, in general) a good amount of thinking and planning here will pay off dividends later in the form of fewer errors and need to move things around in order to fix those errors.

When you import the project footprints to PCBnew by reading the netlist file, all of the footprints are arranged in a matrix adjacent to each other.

Continue by placing in position the footprints that make up the user interface of your PCB. These are things like connectors, indicator LEDs and buttons. You can see an example of this placement process in Figure 14.8.

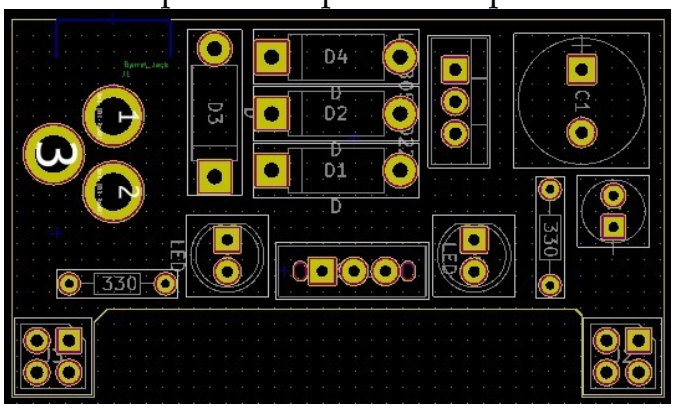


Figure 14.8: An example outcome of the component/footprint placement step.

In this example, I started the placement process by positioning the user interface components along the edges of the board. You should think carefully where to place those components. For example, at the top left of the board, you can see the barrel connector. A cable will plug into this component. Therefore there is a requirement here to ensure that there is sufficient space around the board for the cable. I could have placed the connector facing in the opposite direction, but then the cable would interfere with my work in the breadboard. In Figure 14.9, you can see how the barrel connector, in the top left corner of the PCB, makes it easy to connect the power supply without obstructing the use of the breadboard.

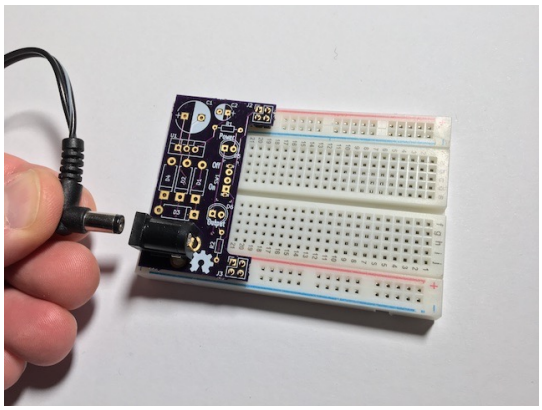


Figure 14.9: Consider the user interface requirements of your PCB carefully.

Other parts of the user interface are the mini slide switch, the two indicator LEDs, and the headers. The headers consist the mean by which the power supply PCB plugs into the breadboard, so their positions are severely restricted by the geometry of the breadboard. I actually placed these components first and locked them in place.

I have more freedom to decide the position of the slide switch. My final decision was guided by the principle of placing UI components along the edges of the PCB for easier access, and ergonomics. I felt it would be easier to reach the switch if it was away from bulky components (like the large capacitor) and closer to the breadboard. I placed the two LEDs around the switch after I had locked the switch in place. Apart from better ergonomics, the LEDs are electrically connected to the switch so it makes sense to place them close as this results in shorter traces.

Once I had finalised the positioning of the UI components, I continue with everything else. I followed these principles:

1. Components that are functionally related should stay close to each other.
2. Shorter traces are better.
3. Consider how placement will affect assembly.

4. Consider component manufacturer specifications.

The four diodes belong to the same functional block (the bridge rectifier), so I placed them as closely as possible. The same applies to the two capacitors and the voltage regulator (the regulator stage). Finally, I placed the current limiting resistors close to their LED for the same reason.

In this example, there is no particular manufacturer specification that I had to take into consideration, other than providing sufficient space for each component. In other cases, however, you may have to deal with components that need, for example, specific provisions for removing excess heat. This can be done by providing additional space for a heat sink, or provide thermal vias for the same purpose. Thermal vias are vias that are not connected to a trace. They are useful for moving heat away from a component, such as an integrated circuit, through a die-attached paddle in between the via and the component. If the manufacturer specifies this as a requirement, you should implement the heat vias or other provisions in this step.

With the components placed on the board, the next step of the process is the routing of the traces.

14.4. Step 4. Routing

With the component placement step complete, you can move on to the next step, routing the traces. This step involves the drawing of the copper connections between the pads. To implement the traces between the pads, you can follow this process:

1. Start with any critical traces. This could include signal traces that have specific shape and length requirements, like an onboard antenna.
2. Continue with power traces.
3. Finish with the rest of the traces.

Let's have a look with an example of what a critical trace might look like. In Figure 14.10 you can see the front and back view of the Microbit board. The Microbit includes a trace Bluetooth antenna. You can see its trace in the top left corner of the front of the board (left image). Because the antenna has strict geometrical specifications for it to operate properly (and legally), it is the first trace that is placed on the board. In the right image of Figure 14.10, you can see the back of the board. In the top right corner of the back of the board you can see that the area of the antenna has nothing on it. No components, and no traces. This is a 'keep out' zone, an area that we can define on a board to make sure that the autorouter or we cannot place anything there. Doing so would affect the way that the antenna works.

To learn how to create a keep out area you can refer to the relevant recipe.

Figure 14.10: The integrated antenna in this Micro:bit is a critical trace.

Once you have taken care of critical traces, if any, continue with power traces. Power traces travel throughout your circuit to feed it with power, and as such, they convey a higher current then signal traces. For this reason, it is appropriate to design power traces (GND, 5V, 3.3V etc) so that they are wider than normal signal traces. For low voltage and low power consumption boards, power traces should be around 0.30 mm to 0.40 mm in width. The trace width depends on a few variables and you can learn more about it in the Trace Width Calculator recipe. You can learn about using net design rules to automatically define the width of a trace in the custom net design rules recipe, or how to adjust the trace width manually in the custom design rules recipe.

After you have completed the routing of the power traces, you can continue with the rest of the traces. For larger boards, you can setup an autorouter that can save you some time. For smaller boards, you can finish routing manually. You can learn about the autorouter in the relevant recipe.

Before continuing to the next step, run the Design Rules Check process to make sure no defects have been introduced.

14.5. Step 5. Copper fills

This step is not necessary and often designers decide to skip it. Copper fill is an area on the board that is fully covered with copper.

Typically, copper fills (also known as 'copper pours') are used to create a ground plane, which is a contiguous mass of copper connected to electrical ground. Similarly to a ground plane, you can create copper planes connected to a voltage level. If your PCB draws power from a battery, then the ground plane is connected to the negative electrode of the battery, and a voltage plane is connected to the positive electrode of the battery.

When a copper fill is created, pads can be connected to the fill using a small number of very short traces called ‘thermal reliefs’ (or 'thermals' for short). You can see an example of this in Figure 14.11.

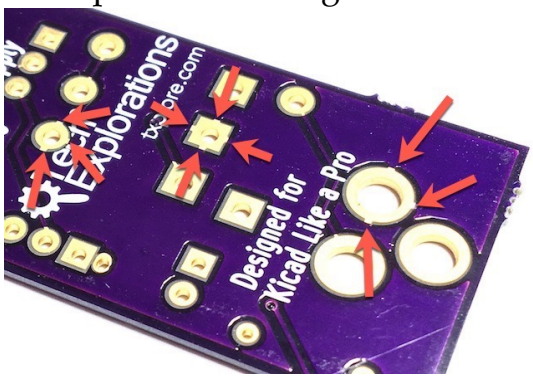


Figure 14.11: Arrows show where GND pads use thermals to connect to the GND plane.

In this example (the breadboard power supply board project from this book), you can see how multiple GND pads are connected to the ground plane in the back of the PCB using up to four short traces. The purpose of the thermals is to help with the soldering of the components on the pad. If the pads were connected to the plane with a full complement of copper, the heat of the soldering iron would dissipate into the copper fill too fast, and the pad would not be able to reach a temperature suitable for the solder to melt. To reduce the heat dissipation speed, the thermals are used to ensure electrical conductivity while managing the heat from the soldering iron.

The distance between the copper fill and traces that belong to a different net is termed 'backoff' or 'standoff'.

Copper fills may be made to be solid, or using a pattern like a 'cherry pie lattice'. Modern copper pours are almost always solid. In the past, cherry pie lattice or hatched patterns were used to prevent wrapping, but this does not seem to be a problem anymore. I have never experienced wrapping in my boards using a solid copper fill.

The benefits of using ground copper planes are:

1. They offer a degree of protection against electromagnetic interference
2. They help to dissipate heat produces by the board components
3. They enforce the discipline of placing signal traces on the top layer and connecting all ground pads to the bottom ground copper plane using vias.

To learn how to create a copper fill, please refer to the relevant recipe in Part 5. As you will see, the process is very similar to creating a keep-out zone.

A copper fill is created once all routing is completed. In a typical 2-layer board, a ground plane is created in the bottom of the PCB, and often a V+ (say, 5 V) copper fill is created on the top layer. If your board is using multiple layers and multiple voltages (like 3.3 V or 5 V), then another option is to use the bottom layer for the ground plane, one of the middle layers for the positive voltages, and the top layer for the signals.

Before continuing to the next step, run the Design Rules Check process to make sure no defects have been introduced.

14.6. Step 6. Silk screen

Step six of the PCB design process is the silkscreen artwork. Silkscreen artwork can be placed on the top and bottom layers. KiCad offers two special layers dedicated to the silkscreen: 'F.SilkS' and 'B.SilkS'. In general, the silkscreen artwork involves the following elements:

1. Descriptions of pads (i.e. what is the role of each pad). This is done using text characters.
2. A name and version number of the board, also in text characters.
3. Your logo, and other graphics you may want to include.
4. Other instructions that may assist the end user, like names, models and values of components, input and output voltage levels, and email address or a website where the user can look for more information.

Let's have a look at an example of these elements in the board you can see in Figure 14.12.

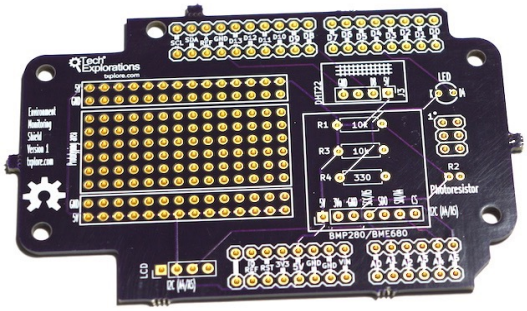


Figure 14.12: The silkscreen information on this board improve its usability.

The board in Figure 14.12 is one I designed for one of my Arduino courses. I designed it as an Arduino shield. It contains a prototyping board, and has provision for a DHT22 sensor, a BMP280 sensor, a photoresistor, and LED, and exposes the I2C interface so I can connect an LCD screen. When fully assembled and stacked on an Arduino Uno and an Ethernet Shield, it looks like the example in Figure 14.13.

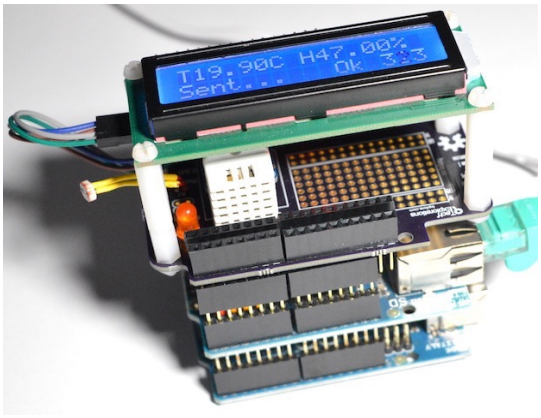


Figure 14.13: The Environment Shield in operation.

In Figure 14.12 you can see the contents of the top layer silkscreen in white ink. The Tech Explorations logo, with a URL, and the 'open-source hardware' logo appear on the left of the board. The name of the board and its version appear at the left edge of the board. Each pad has text that describes its purpose; the BMP280 and DHT22 pads are marked. There is also information on the values of the resistors, and the cathode of the LED is marked. All this information will help the end user to assemble the board without the need for a reference document.

The prototyping area is also marked. The row of pins that convey the GND and 5V levels are clearly marked. The bottom layer can also have a silkscreen where you can provide additional information. Because the bottom layer typically does not have components, there is more available real estate to use for this purpose.

Spending some time to design a beautiful and informative silkscreen adds significant value to your board, so it is worth the effort. Because there is no automated test, like the DRC, for ensuring that the information in the silkscreen is correct, you should take care to manually check and double check that there are no errors. Much of the silkscreen text and graphics belong to the footprints themselves. If the footprints come from a quality source, like the KiCad repository, the risk for errors is low, but it is still prudent to check your self. For example, in Figure 14.12, the LED footprint came with its own silkscreen graphics. The circle around the pads is part of the footprint. But to make the assembly process easier, I added the 'K' designator to indicate the cathode pad so that the end user would know how to connect the LED. The shield pad markings ('A0', 'A1', 'SCL', 'SDA', etc.') are all part of the Arduino shield footprints that I imported into the project. While I modified the footprint to add space for the prototyping area and the LCD screen, I left the silkscreen text in place.

To learn how to insert graphics, such as a logo, to the silkscreen, please refer to the relevant recipe. There is also a recipe that shows how to insert silkscreen text and simple graphics to the top or bottom layer.

14.7. Step 7. Design Rules Check

The seventh step of the PCB design process is the Design Rules Check (DRC). While it is a good habit to run the DRC frequently, and at least every time you complete major trace routing or adding a copper fill, you should always run it when you are satisfied that the design work is complete and before you send the board to your manufacturer.

To start the DRC, click on the DRC button which is located in the top toolbar (). In the DRC window (Figure 14.14), click 'Start DRC' to run the basic check. If it all goes well, the Problems and Unconnected tabs will remain empty.

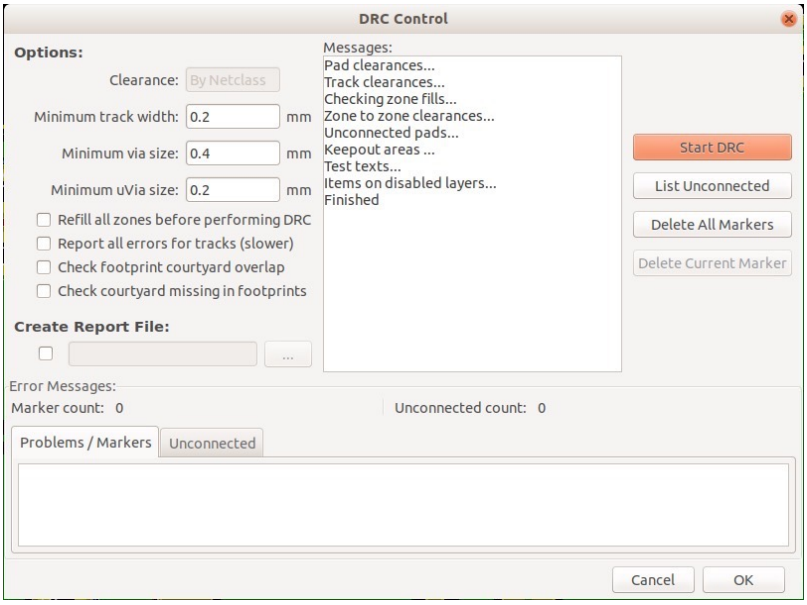


Figure 14.14: The DRC window; this checked returned no errors.

The DRC window allows several check options. You can get it to automatically refill zones before performing the DRC, to report all errors for tracks, and do a couple of courtyard checks, like courtyard overlap and footprints with missing overlaps. A footprint courtyard is defined by the KiCad documentation[11] as the smallest area that provides a minimum electrical and mechanical clearance around the component. Footprint courtyard layers are F.CtrYd and B.CtrYd.

[11] http://kicad-pcb.org/libraries/klc/F5.3/
Last accessed November 19 2018.

When I did the DRC again in my project board, with all options selected, as you can see in Figure 14.15, the DRC returned one problem that indicates that one of the board's footprints has no courtyard defined.

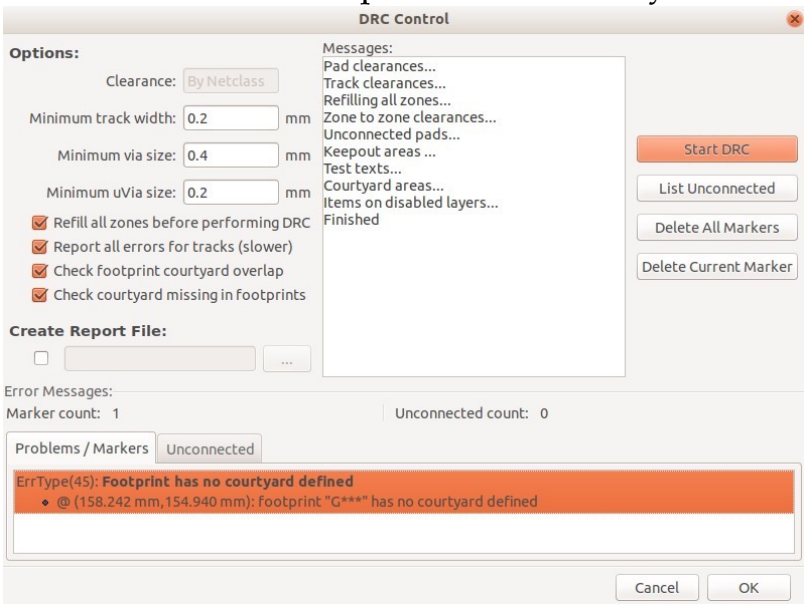


Figure 14.15: This DRC returned one problem.

To investigate the problem, I turned off the filled zones so I can see the board with more clarity. The DRC arrow in Figure 14.16 indicates the problem area.

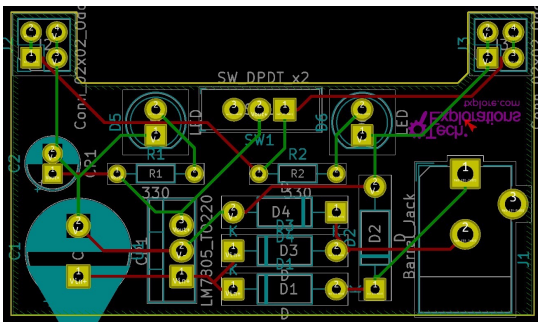


Figure 14.16: The logo footprint has no courtyard defined and triggered the courtyard check.

Because the logo only has silkscreen elements and no electrical or mechanical attributes, you can simply ignore this problem. If this was an actual electrical component, you would need to evaluate it and fixed if you determined that it could affect the operation of the board. Not every warning or problem needs to be dealt with; many can be ignored safely.

14.8. Step 8. Manufacturing

The goal of the PCB design process is turning the PCB from a set of files on your computer into a physical object. There are a few ways by which you can do that. At home, you can use a chemical etching process, or a CNC machine to carve out a circuit on a copper board. I personally find the

chemical etching process too messy, not to mention that you have to work with potentially toxic chemicals. If you already have a CNC machine, then you can certainly use it to make your boards. With a bit of patience and practice, you will be able to create two, or even four layer boards. In both cases, you will need to use a special process to create vias and holes, with copper-plated holes being more challenging.

I personally prefer the simplicity of using online manufacturers who can produce high-quality boards for a relatively small cost. You will need to plan ahead because the lead time (manufacturing plus shipping) can take a few weeks.

The standard method for ordering a PCB from an online manufacturer is by exporting Gerber files from KiCad and importing them to the manufacturer's website. Lately, companies like Oshpark make it possible to simply upload Pcbnew's '.KiCad_pcb' file. I find this development very encouraging because it is so simple. Exporting Gerber files has been the source of many mistakes and wasted time in my life. You have to be careful to export the correct files, with the correct file name extensions and units, and must not forget the drill files. With the ability to upload the '.KiCad_pcb' you automatically eliminate a big risk factor.

In this book, you will learn how to manufacture a PCB with an online service by using both the traditional Gerber files and Pcbnew's '.KiCad_pcb' file. To learn how to do this, please refer to the relevant recipes in Part 5 of this book (using the KiCad_pcb file and the traditional Gerber files method).

15. Additional design considerations

After reading the previous chapter, you should have a better understanding of the PCB design and layout process. Always keep in mind that there is no substitute for planning. Good planning not only will produce a better result, but it will allow you to get there faster and with fewer problems to deal with.

In this chapter, you will find common practices, advice, and simple things that you can do so that your PCBs come back from the manufacturer looking great and working perfectly. This chapter builds on the previous one. I advise you to not proceed unless you have already read the previous chapter.

I have grouped the Design Considerations into several topics. Let's begin with Shape and Size.

15.1. Shape and size

A printed circuit board is defined primarily by its shape and size. These attributes dictate what components will fit on its surface, where they will be placed, how they will be wired, and how to interface the board with other components of your project (like a project box). For these reasons, defining the outline of the board is one of the first design decisions you have to make.

Unless you have good reasons for doing so, keep your boards rectangular. This shape is easier to manufacture and results in less wasted space on the manufacturing panel. This results in a lower price. Rectangular PCBs also make it easier to place components and route the traces.

If you do have good reasons for doing so, KiCad and other modern tools, in conjunction with modern manufacturing techniques, allow you to implement almost any shape you can imagine. Round, triangular and other irregularly shaped PCBs are possible for both hobbyist and professional designers. In Figure 15.1 you can see the board of the Crazyflie 2.0 tiny quadcopter. The board's shape defines the actual quadcopter shape.

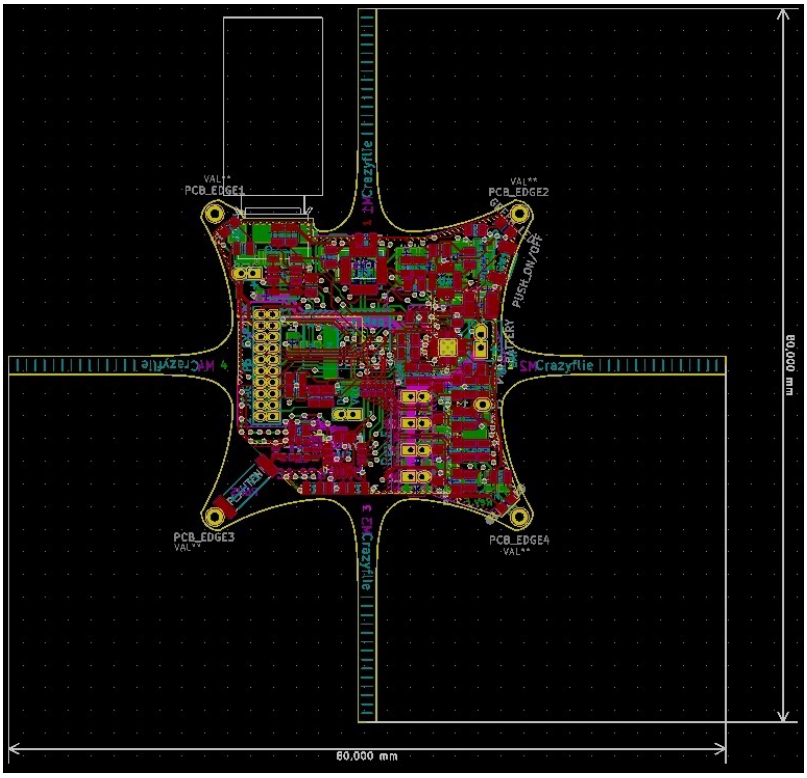


Figure 15.1: The Crazyflie 2.0 board is irregular (but awesome).

Modern design tools, like KiCad, also allows creating interesting features even if you go for a very simple board outline, like a rectangle. You can create a rectangular shape with rounded corners, for example, instead of sharp, 90-degree corners. Please read the relevant recipe to learn how to create a rounded corner.

Aside from the shape, two more considerations are important that relate to the mechanical aspects of your PCB: heat distribution requirements and mounting requirements.

To mount your PCB to a project box you can use screws or snap-on points. If you want to use screws, you must create screw mounting holes on your PCB that match the screw bosses in the box. You can see an example in Figure 15.2.

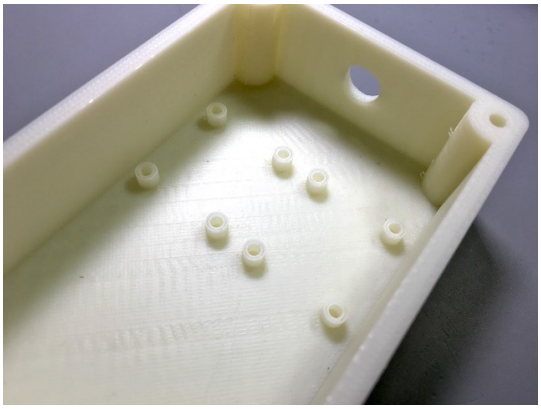


Figure 15.2: This project box has eight screw bosses.

If you are designing a PCB that is very small in size, screw holes will take up a disproportionately large amount of space. To avoid this, consider using snap-on point. A snap-on point is simply a location along the edge of the board where a small plastic flap that belongs to the project box will rest with a small amount of tension. This will keep the board in place. Creating a provision for snap-on points is just a matter of marking the area where the flap will rest as a keep out zone to prevent footprints and other elements from placed there.

The last consideration for this section is heat distribution requirements. While the board itself does not produce heat (unless traces are heating up because of a defect), the components on it can produce a considerable amount of heat. Transistors, voltage regulators, ICs and other components may need provisioning for distributing their heat away. Part of the planning for this can be done during the shaping of the board. For example, if you have a TIP transistor that drives a motor, you may need to provide it with a heatsink. The heatsink will require additional space, that you should either provide as cleared space around the transistor or perhaps by placing the transistor in the outer edges of the board so that it can be combined with an opening of the project box to facilitate ventilation. When you work with such components, take care to review their datasheet as they often contain advice on how to best implement heat distribution.

15.2. Layers

Modern PCB manufacturing methods have matured to the extent that two-layer boards are the norm. While you can still design and upload a single layer board, manufacturers will simply treat it as a two-layer board. On the other hand, if you are making your own boards at home, using the traditional chemical etching method or a CNC router, then a single board design is certainly much easier to manufacture than a two-board design.

The advantages of two-layer boards include:

1. They allow for easier component placement and trace routing. For most boards above the trivial level of complexity, the second layer allows traces to move from top to bottom and provide electrical connectivity without having to move components around or use jumper wires.
2. They are cheaper to make compared to multilayer boards, as manufacturers have optimised their processes for two-layer boards.

3. They provide ample space for a ground plane that can also be used for dissipating heat (useful, for example, in motor controller applications) and for countering interference from electromagnetic fields.

4. They provide additional space for silkscreen artwork, which can contain information about the board, graphics, and contact details for the designer.

15.3. Traces

A few simple design principles will ensure that your traces work flawlessly.

Length

Keep your traces as short as possible. As traces increase in length, more energy is lost on them as heat because of the material resistance. An additional benefit of shorter traces is controlling propagation time. This is especially important in applications that operate in high-frequencies and where data is transmitted in a bus through several traces (for example, RAM interfaces). The data must arrive at its destination within tightly controlled timeframes across all the traces of the bus. The main factor for this is the trace length. KiCad has tools that allow you to set specific trace lengths for applications such as that.

Angles

Electricity doesn't like flowing through sharp angles.

When a trace contains a sharp angle, like a 90-degree change of direction, undesired reflections and radio interferences can be generated. This is especially true for applications that operate at 200Mhz or more.

In slower applications, 90-degree angles don't make a noticeable difference, but it is still a good habit to use 45-degree angles in your traces.

Weight

As the designer, you must also decide on the thickness and copper weight of the traces. A standard thickness of a two-layer board is 1.6 mm. Various manufacturers, like PCBWay, offer other options, ranging from ultra-thin 0.4 mm to very thick 2.4 mm. A thicker board will be structurally stronger and can have interesting applications, such as the example of the Crazyflie board which provides the structural integrity for this tiny quadcopter.

Another consideration is the copper weight, which defines how much copper is used to create the traces. This number is separate to the trace width

value. Copper weight is defined in ounces (oz), as the weight of copper present in one square foot. The industry standard is 1 oz copper weight (300 g per square meter). Designing traces with more copper requires a more expensive etching process but improves the electrical characteristics of the board. In most applications, a 1 oz copper weight is appropriate. If you are unsure, you can use the Calculator app that comes with KiCad to confirm that.

Width

You can increase the amount of current that flows through a trace by increasing its width, instead of the copper weight. In particular, for traces that transport power to your circuit (ground, 5 V etc), it is a good practice to increase their width compared to signal traces.

You can learn how to calculate the ideal width of a power trace in the relevant recipe.

Proximity

Traces should be sufficiently separated from each other. This is good from a manufacturing point of view, but also for operational reasons.

Manufacturers have limits on how close two traces can be before they can no longer be manufactured without a high chance of defects. A typical separation between traces that is compatible with most manufacturers is 0.15 mm (6 mil). As an example, you can visit PCBWay's capabilities page.[12]

The second reason for which separation is important is for avoiding cross-talk interference between neighbouring traces. Again, this becomes important in high-frequency applications.

[12] https://www.pcbway.com/capabilities.html
Accessed November 19, 2018.

Part 4: Projects

16. About this Part

In this part, you will design three printed circuit boards on KiCad. Each circuit board project will give you the opportunity to consolidate and expand on your knowledge. You will become proficient in using KiCad features. You will also gain experience in schematic design and layout.

The topic of the first project is a breadboard power supply. This project will teach you how to design a board with specific mechanical constraints, on top of walking through the full design process.

The topic of the second project is a tiny HAT for the Raspberry Pi. In this project, you will learn how to use third-party board layouts and customised them to your needs. Board layouts can save you a lot of time since you do not need to take your own accurate measurements.

The topic of the third is to design an Arduino Uno clone, with features useful for building mobile, battery-powered gadgets. In this project, you will learn, among many things, how to shrink the size of your PCBs using SMD components, how to use the autorouter, and how to use hierarchical sheets.

Let's begin!

17. Project 1: Design a simple breadboard power supply PCB

17.1. Walk through a simple project

This project is the first complete and realistic one in this book. You will learn to use KiCad's most important functions, and experience a simplified version of the PCB design process.

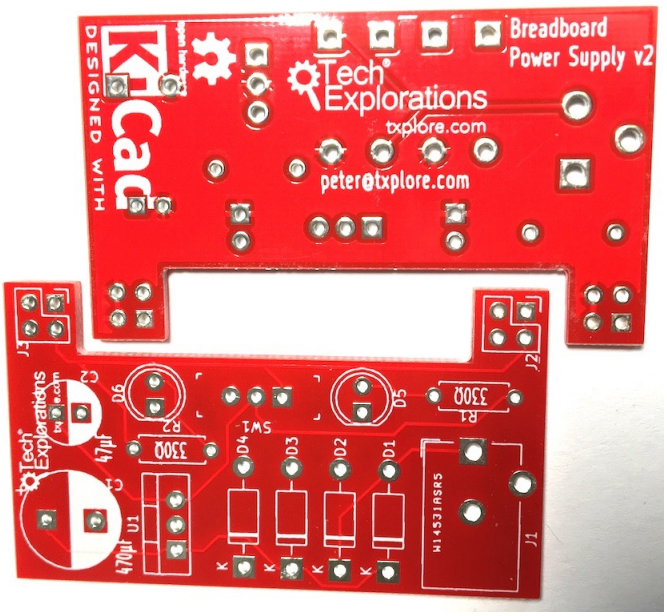


Figure 17.1: The outcome of Project 1.

In this project, you will build a double-sided PCB for a 5V power supply. You will be able to use this power supply to power your breadboard Arduino projects, or any project that requires a 5V power supply. As an input to your power supply, you can use an old DC or AC walled power supply, as long as its output voltage does not exceed 35V (I'll explain the reason behind this limit shortly).

In the process of designing your power supply, you will learn all the skills you need to create PCBs of comparable complexity. You will not learn everything there is to know in KiCad; you will not suddenly become an experienced PCB designer. But, you will have the opportunity to apply the principles you learned in Part 3, and create a respectable board.

What you will build and list of parts

In this first project, you will design the PCB for a 5V power supply for your projects. This power supply will use the LM7805 voltage regulator, a bridge rectifier composed of four 1N4007 (or 1N4004) rectifier diodes, a couple of smoothing capacitors, and a couple of indicator LEDs with their current-limiting resistors.

To draw power from mains safely, you will need a walled power supply from an appliance that you don't need anymore, that is capable of outputting AC or DC between 7V to 24V.

Here are the parts you will need:

1. One walled DC or AC power supply capable of outputting between 7V and 25V.
2. One LM7805 voltage regulator IC.
3. Four 1N4007 rectifier diodes.
4. One 470 μF capacitor, with a working voltage of at least 25V.
5. One 47 μF capacitor, also with a working voltage of at least 25V.
6. One red and one yellow LED
7. Two 330 Ω resistors to protect the LEDs

The exact capacitance values of the capacitors are not important. Larger capacitors will produce a smoother voltage in the output. What is important, however, is their working voltage. If you use a capacitor with a working voltage of 25V, be careful to not use a mains power supply that provides an output voltage of more than 25V, as this will exceed the maximum rated voltage of the capacitor. In Figure 17.2, you can see the parts that we'll be using in this project.

Figure 17.2 The components that we'll use in our power supply.

In regards to the schematic, our power supply is depicted in Figure 17.3:

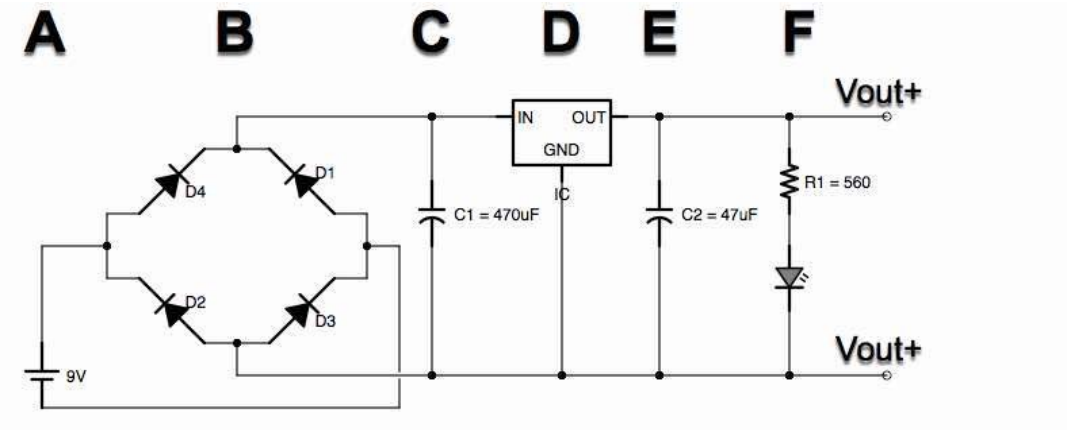


Figure 17.3: The power supply schematic diagram.

In the schematic, you can see the stages that lead to a stable 5V output voltage.

In stage A, we connect a step-down transformer mains power supply. In the schematic, I have marked it a 9V DC, but it can be any power supply that produces AC or DC between 7 and 25V.

The voltage from this mains power supply is taken through stage B, the rectifier stage. If the input is an AC waveform voltage, the output of the rectifier will be a DC waveform voltage, always positive. If the input is a DC voltage, then the output will be the same DC voltage.

To learn about rectifiers and how they work to produce DC voltage from an AC input, please read the relevant article on Wikipedia[13].

In the next stage, C, the voltage is further stabilised. The capacitor stores energy as the voltage rises and releases it as the voltage drops, and in effect, it 'bridges' the gaps in the waveform.

In stage D, we use the voltage regulator LM7805 to produce the 5V regulated voltage that we aim for. At this point, we can go ahead and connect the circuit that we want to power. However, we can use an additional smaller capacitor to smoothen out this voltage even further, in stage E.

Stage F is where we can connect an indicator LED that can tell us when the power supply is operating.

Figure 17.4 shows the assembled circuit on a breadboard. You can try this yourself and confirm that it is operating as expected.

[13] https://en.wikipedia.org/wiki/Rectifier, visited December 11, 2018.

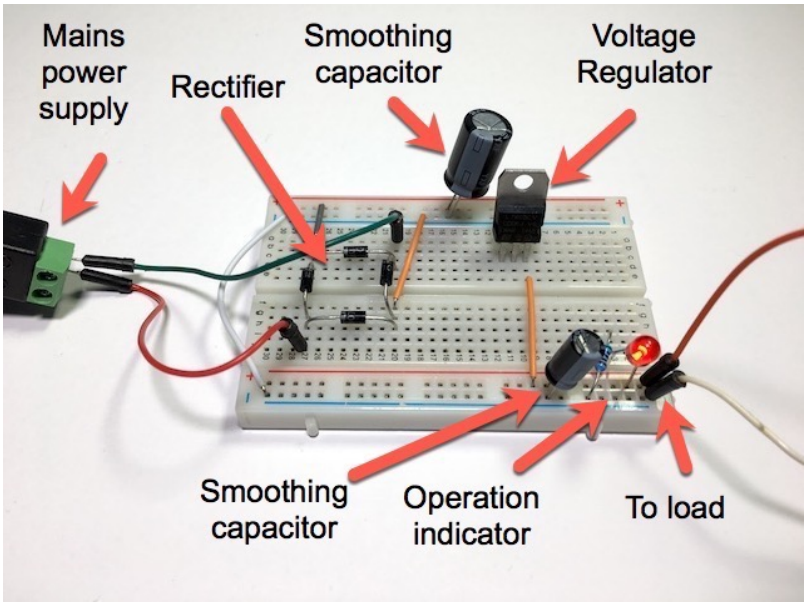


Figure 17.4: The power supply implemented on a breadboard for testing.

In my testing, I supplied the input with 9V DC using a spare mains power supply and tested the output. I used my multimeter to get the actual values of the voltage from the mains power supply, and the output from my breadboard power supply.

Figure 17.5: The actual voltage of the 9 V DC mains power supply I use to power the breadboard power supply.

In Figure 17.5 you see that the actual voltage of my 9 V mains power supply is slightly higher than what is advertised on the label. This is often done by design, to compensate from the voltage drop that occurs when a load is connected. In this measurement, I had only connected an LED as a load.

Figure 17.6: Actual output voltage, when input power comes from a 9 V DC mains power supply.

In Figure 17.6 you can see the actual output voltage of the circuit when I use my 9 V DC mains power supply. It is very close to 5 V. In another test, I applied a 16 V DC input voltage from my bench power supply to the breadboard circuit and measured the output voltage at 5.010 V. At 20 V input, the output voltage was 5.013 V. Based on these tests, we can be confident that this breadboard power supply works as expected.

Now that you have a good overview of the objective of this project, let's turn our attention back to KiCad.

What you will learn

In this project, you will use all of KiCad's main features that you learned about in Parts 2 and 3 of the book. As with the rest of the projects in this book, we will follow the design process outlined in Figure 17.7. You should download your copy of the graphic from the book's page,[14] print it, and use it as a quick reference guide.

To keep things simple, the process you will follow assumes that all of the components we need for designing this PCB are available in KiCad's

[14] The book page is at http://txplo.re/klpp. Valid as of December 11, 2018.

standard libraries[15] and that we are not interested in implementing any 'fancy' design and aesthetic features. You will have the opportunity to do both in later projects.

Nevertheless, you will learn a great deal. Here's a list of your learning objectives for this first project:

1. Learn how to design a schematic using Eeschema, KiCad's schematic editor.
2. Learn how to do an electrical test and confirm that the schematic is correct.
3. Learn how to associate the components in Eeschema with footprints, using KiCad's Cvpcb, KiCad's associations editor.
4. Learn how to create a file that contains the information that we need in order to do the PCB layout in Pcbnew. Pcbnew is KiCad's layout editor. The file that contains the information is known as the 'netlist'.
5. Learn how to use Pcbnew to design your PCB, place the components on it, and connect them by creating routes.
6. Learn how to prepare the PCB for manufacturing, including testing for defects.
7. Learn how to upload the files that contain the PCB design data to an online manufacturer.

While this process may feel like a lot of work, over time, it becomes just part of your prototyping process. Personally, designing a PCB is work that I enjoy immensely. Creating a PCB involves engineering and design/aesthetic elements.

Creating a PCB is an opportunity to not only create something useful, but also an object of art. Once all your electrical tests are done, once you have placed and routed all the components on the board, you will find yourself obsessed with small details, like the exact position of a silk screen label, or the size of a contour that defines the shape of a corner.

But to get to the art, you must first take the first step.

Let's do that now with the schematic design in Eeschema.

Project repository

This project has a Git repository. You can access it at txplo.re/klpp1.

---

[15] A library is a collection of components (the symbols that we will use to create the PCB schematic), and footprints (which we will use to design the layout of the PCB). KiCad comes with a lot of very useful standard libraries, but we can also import and use libraries created and shared by KiCad users.

This repository contains multiple commits at each step of the design process, as I was going through it. Feel free to clone this repository on your computer and make any modifications you feel like.

17.2. Schematic design: Eeschema

Let's begin the design process with the schematic diagram. The KiCad tool you will use is Eeschema, the schematic editor. You can follow the schematic design process in Figure 17.7.

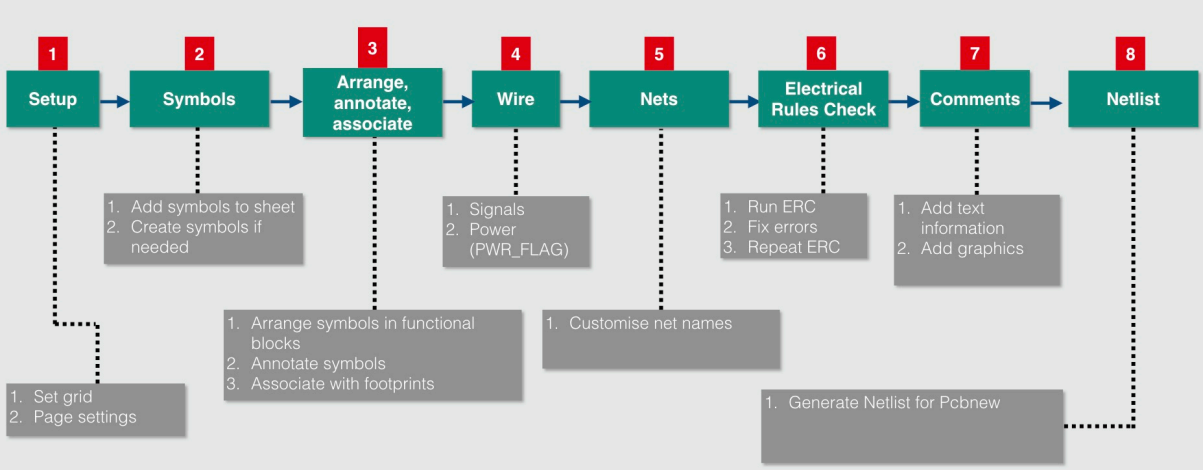


Figure 17.7: The schematic design process.

In Eeschema, you will replicate the schematic in Figure 17.3, with several changes and adaptations. Let's discuss those first.

As it appears, the schematic in Figure 17.3 is powered by a battery. The breadboard power supply will be powered by an AC/DC power transformer power brick. You will need some kind of connector to feed the voltage from the power brick to your board. In addition, the schematic does not contain a switch. From a usability point of view, it would be nice to have a switch on board so that we can use that to turn on power to the power board instead of having to unplug the cable from the board. Again from a usability point of view, if we choose to use a switch, it would be good to have an indicator LED that can inform us when the breadboard power supply is powered, in addition to when the breadboard itself is getting power.

I will show you how to implement those changes to the basic circuit in Figure 17.3, and create the schematic in Eeschema in accordance with those decisions.

Important: remember to save your work frequently. Although KiCad is very stable, it can still crash. Saving frequently will protect you from lost work.

Step 1: Setup

Start KiCad and create a new project. I called mine BreadboardPowerSupply_v2. From the KiCad main window, start Eeschema. You should now have an Eeschema blank sheet. The first step of the schematic design process is the setup of the page. Open the Page Settings window and fill in the project information. If you want to use a custom layout, you should do that now by selecting your custom page layout description file.

In Figure 17.8 you can see the settings of my schematic page. I am using a custom version of the page layout description file. You can learn how to do this by reviewing the recipe titled '42. How to design a custom page layout'.

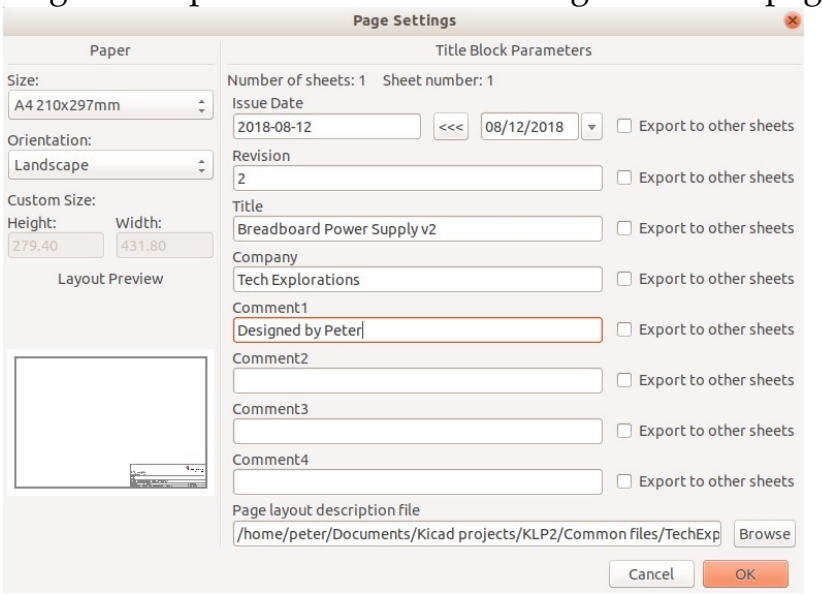


Figure 17.8: The schematic page parameters.

The new schematic page is ready. Let's add the circuit symbols in it.

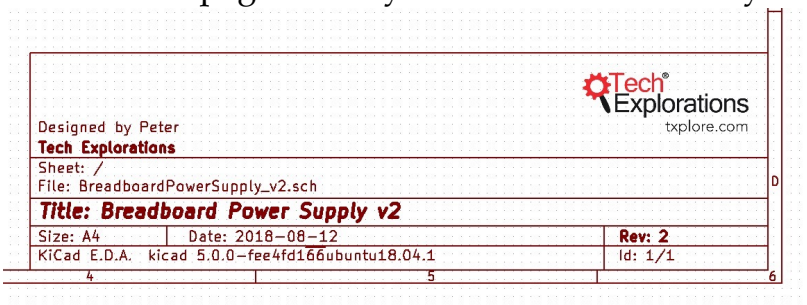


Figure 17.9: The sheet information.

Step 2: Symbols

In this step, you will place the symbols that represent the components in the circuit. You will do this by finding them in the symbols browser and adding them to the sheet, one at a time.

Start by creating a list of the symbols you need to search for in the symbols browser. Here is a list of the components that are part of the circuit. Each one will need a symbol. I have included the package information where applicable because in step 3 you will need to associate the symbols to a footprint.

1. One LM7805 voltage regulator IC, TO220
2. Four 1N4007 rectifier diodes
3. One 470 µF capacitor, with a working voltage of at least 25V
4. One 47 µF capacitor, also with a working voltage of at least 25V
5. A red LED to show power status on the output (breadboard)
6. A green LED to show power status on the input
7. Two 560 Ω resistors to protect the LEDs
8. One slide switch to use as the ON/OFF switch
9. Two 2x2 headers for the interface between the power supply and the breadboard
10. A barrel connector for the power input

Begin adding each symbol to the sheet. In Eeschema, click on the Place Symbol button from the right toolbar (), or type the 'A' hotkey. In the symbol chooser window, type 'lm7805' in the filter field. The symbol you are looking for is available from one of the symbol libraries that KiCad ships with, so it should appear, as in Figure 17.10.

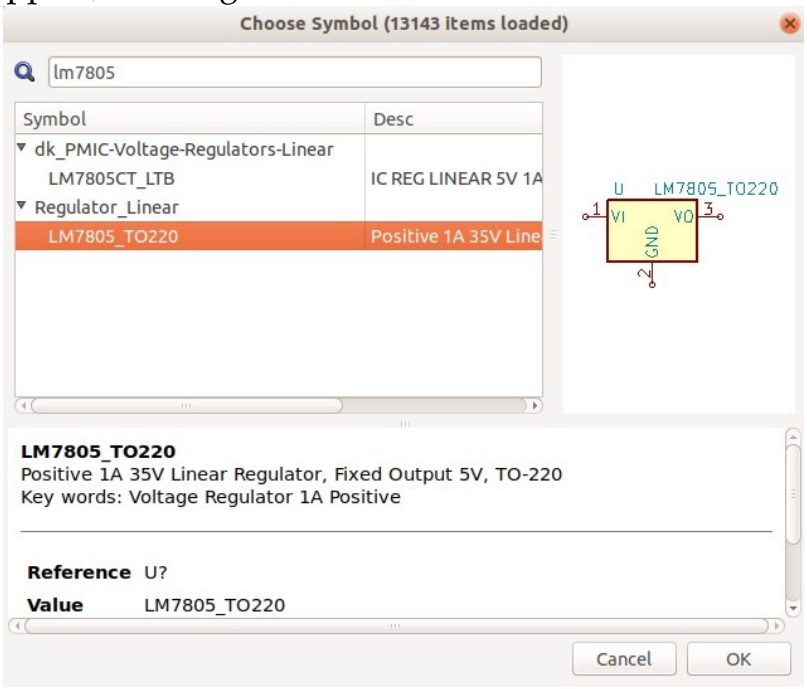


Figure 17.10: The symbol for the voltage regulator.

Select the version of the symbol with the TO220 package, and click 'OK' to add it to the sheet.

Continue with the rectifier diodes. The symbol for a rectifier diode is the same as for a normal diode. In the symbol chooser, type 'D' in the filter field. Click 'OK' to add the diode to the sheet.

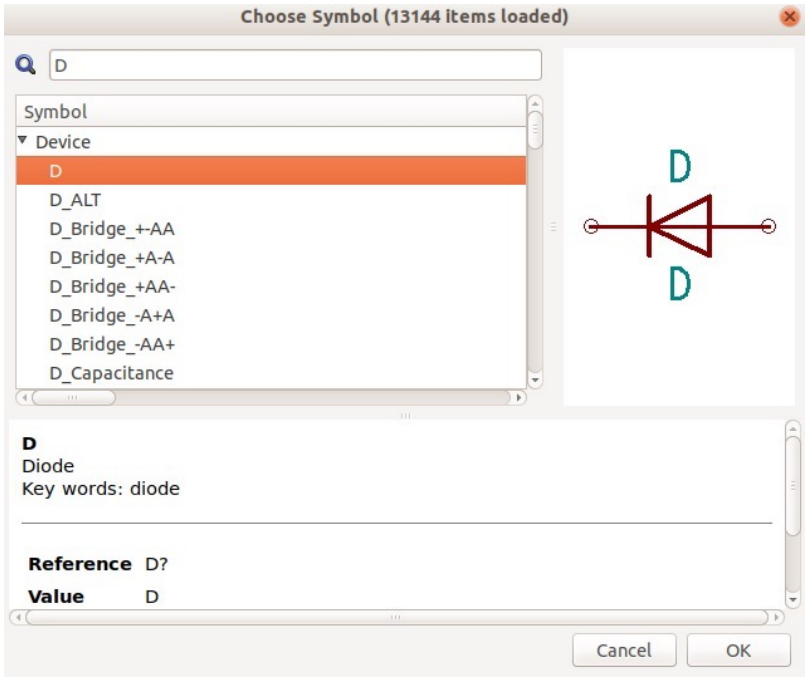


Figure 17.11: The symbol for the diode.

You need four diodes. Instead of repeating the process four times, duplicate the first diode three times using the 'C' hotkey. Simply position the cursor over the first diode and type 'C' to duplicate the symbol.

Several of the symbols in this schematic have values. For example, there are two capacitors with 470 µF and 47 µF capacitance values, and two 560 Ω resistors. This step is a good opportunity to set those values for each symbol, using the Symbol Properties dialogue box.

Let's set the value for the 470 µF capacitor.

Place your mouse pointer over the symbol, and type "E", or right-click and select Properties —> Edit Properties from the contextual menu.

In the Properties window that appears, type the value of the symbol; in this case it is "470 µF". Click Ok to commit this change.

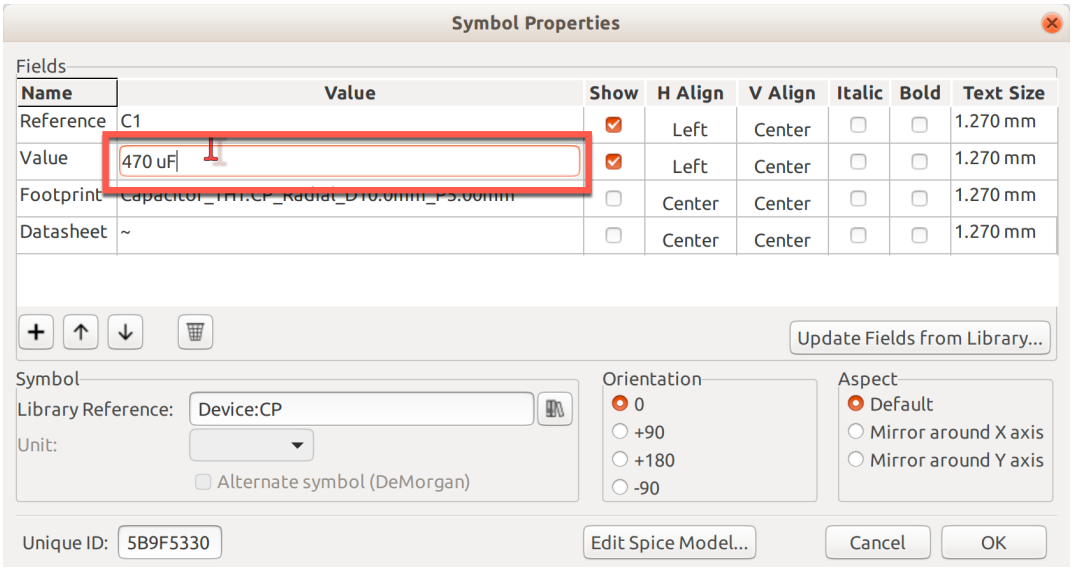


Figure 17.12: Set the value of a symbol.

Some symbols don't have specific value, like resistors and capacitors do. In similar cases, such as with the barrel DC power connector, you can use the Value field to provide a description of the symbol that will be helpful to the end user. Remember that the text that you enter in the Value property of a symbol is carried across to PCBNew and the layout. You will be able to make those values visible on the final PCB, for example in the silkscreen. By adding those values in the schematic, you will not have to type them in the layout step, and you will have a more detailed and descriptive schematic.

Continue with the rest of the items. The capacitors should be polarised, so ensure that you select the correct version of the symbol. In Figure 17.13 you can see the symbols that make up the schematic in this project. If you are having trouble locating these symbols, use their names as shown here to search for in the symbol chooser. For example, the barrel connector is discoverable in the symbol chooser under the name 'Barrel_Jack_Switch'. If you are not sure about the name of a symbol, you can always browse the symbol libraries manually. Remember to set the values of the symbols as you add them to the sheet.

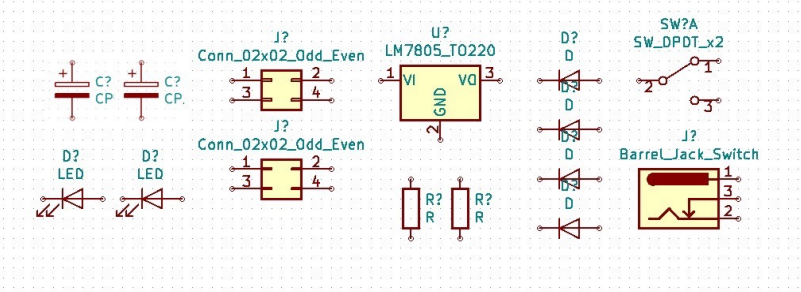


Figure 17.13: The schematic symbols.

With all the symbols that make up this schematic in the sheet, you can continue to step three, where you will do the three A's: 'Arrange, Annotate, Associate'.

Step 3: Arrange, Annotate, Associate

In step three, you will arrange the symbols in place, provide a unique designator for each one ('Annotate'), and finally will associate each symbol with a footprint. In the PCB design process depicted in Figure 17.14, associations are done in the Cvpcb step, but you can also do this at the symbol level, for each symbol individually.

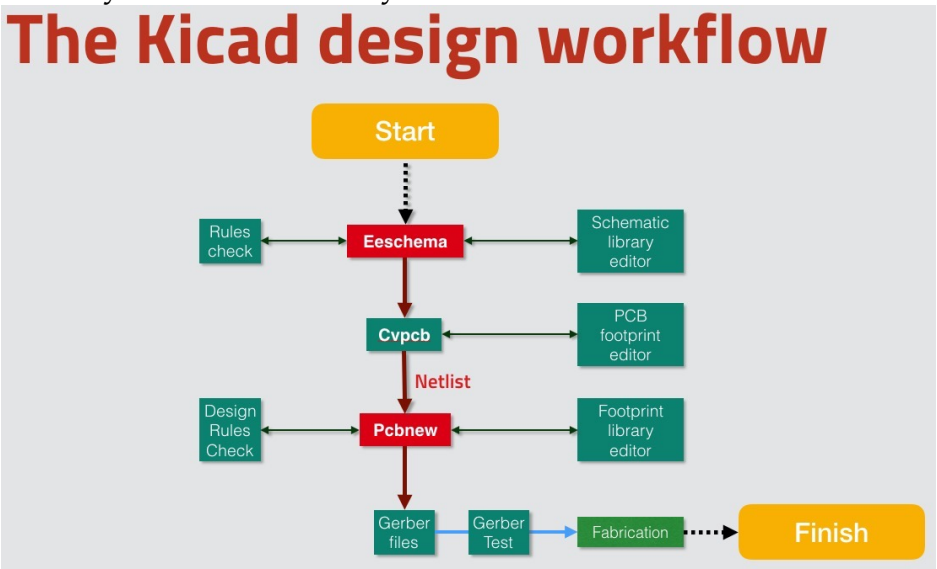


Figure 17.14: The KiCad design workflow.

In later projects, I will also show you how to use Cvpcb and do the associations in 'bulk'.

Arrange

We want symbols to have a logical arrangement so that they are both easy to read, and so that the risk of defects is reduced. What this means in practical terms depends on various factors, but in general:

1. Symbols should be arranged in functional blocks; those symbols that are directly connected should be placed near each other,
2. Inputs should be placed on the left side and outputs on the right.

This project lends itself well to this approach. The power supply is a linear combination of functional blocks. You can simply arrange them in place in accordance to these blocks.

Use the 'M' and 'R' hotkeys to move and rotate the symbols. I actually started with placing the voltage regulator first, as I regard this component the centre piece of the circuit. Then, I move to the input and then the output. Use

the grid to produce a placement that is symmetrical and balanced. In Figure 17.15 you can see how I have arranged the symbols.

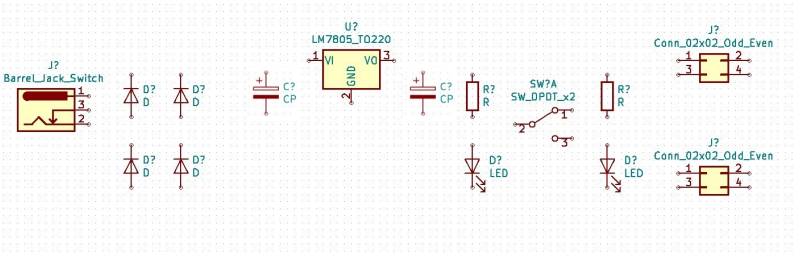


Figure 17.15: The symbols grouped and arranged according to functionality.

Annotate

Now that the components are in place, they need a unique designator. Think of the designator as an ID. The designator consists of a letter that depends on the type of the device it is attached to and a number. These two together must be unique for each symbol. You can learn more about designators on Wikipedia.[16] You can set the designator for each symbol yourself, through the Symbol Properties window. But this is a tedious and error-prone process. It is better to use the Annotator tool that Eeschema provides. Click on the Annotator button in the top toolbar (). You can use the default setting. Click on the 'Annotate' button to allow the annotator to do its work.

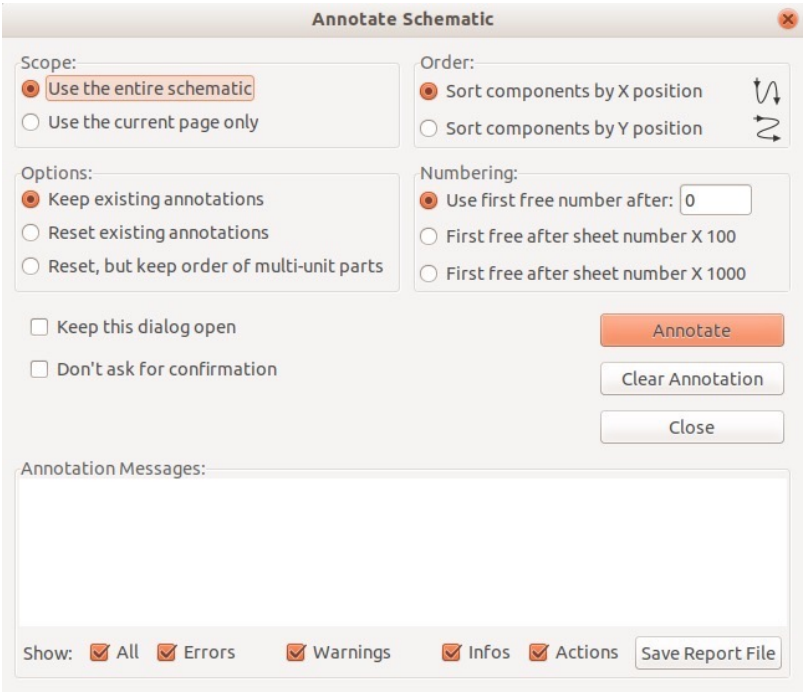


Figure 17.16: The Annotator.

[16] https://en.wikipedia.org/wiki/Reference_designator, visited December 11, 2018.

The tool will assign a unique designator to each symbol, and the schematic will now look like the example in Figure 17.17.

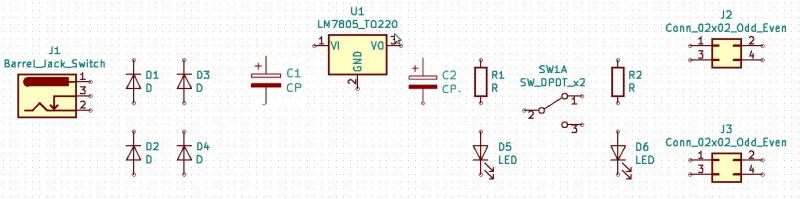


Figure 17.17: The components have a unique designator

Associate

Time to associate the schematic symbols with the layout footprints. This association is what helps, Pcbnew, the layout editor, to know which footprints to use. While you can create these associations at any time, up until the end of the schematic design process and just prior to exporting the Netlist, I have created a habit to do it when I have added all the symbols to the schematic sheet.

You can do the associations in two ways:

1. In bulk, using the Cvpcb tool
2. Individually, through the symbol properties

Let's try out both. We can start by doing the associations for a couple of symbols, like the voltage regulator and the switch. When you have multiple instances of the same symbol, like the resistors and the diodes in this project, the bulk associations tool (Cvpcb) is more efficient.

Place your mouse cursor over the voltage regulator symbol and type 'E' to bring up its properties. You can see the properties window in Figure 17.18.

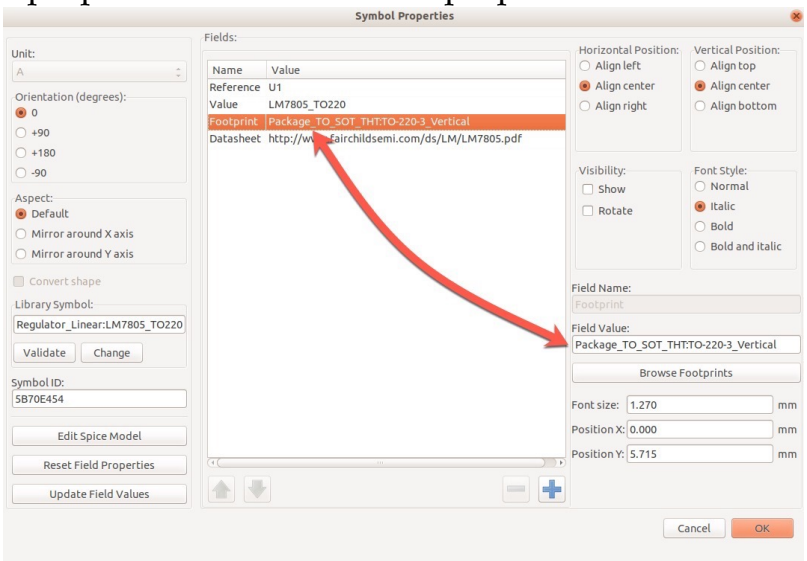


Figure 17.18: The properties for the U1 (voltage regulator) symbol.

The voltage regulator symbol already has a symbol assigned to it. You can see that the Footprint field has a value. You can verify this by selecting the

Footprint field to enable it, and then clicking on 'Browse footprints'. This will bring up the footprints browser. In the browser, the library and footprint for the TO220 package should already be selected.

Click OK or Cancel to get out of the symbol properties for the voltage regulator, and continue to edit the barrel connector (Figure 17.19). This symbol doesn't have a footprint so let's give it one.

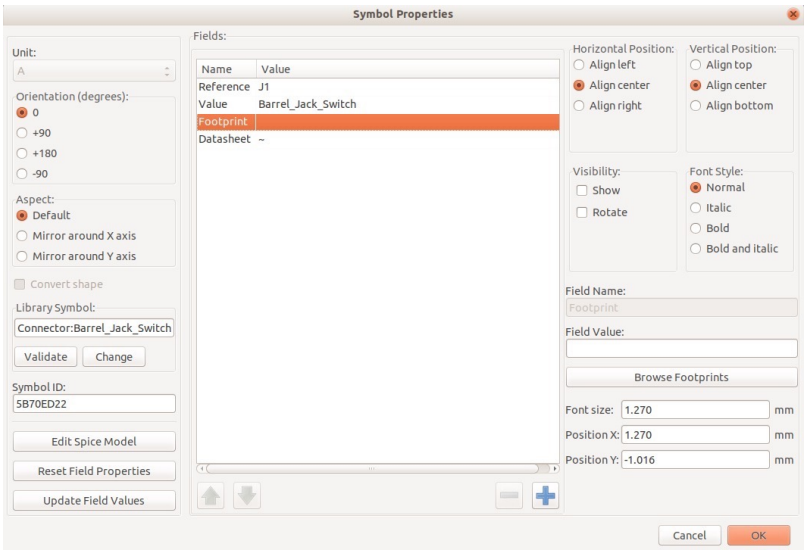


Figure 17.19: The properties for the barrel connector symbol.

In the symbol properties window for the barrel connector, click on the footprint field, and then the 'Browse Footprints' button. In the footprint browser window that came up, look for a library titled 'Connector_BarrelJack'. In it, there are several footprints that might work. This is where you need to look at the actual, physical part that you want to use as a component on the breadboard, and ensure that the footprint you choose here matches that component. In my case, I want to use a barrel connector like the one in Figure 17.20.

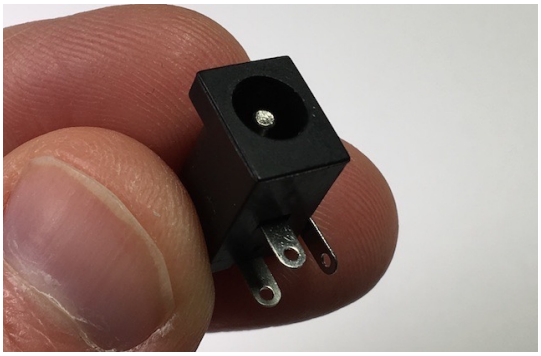


Figure 17.20: I want to use this connector.

There are a few possible matches for this component, but to be certain you should measure the physical component and cross-check those measurements against the footprint representation in the footprint browser. I

do this with my calliper, as in Figure 17.21. I can take a few measurements between the connector pins and then check for a footprint that matches those measurements.

Figure 17.21: The distance between two of the connector pins is 6.18 mm.

In the footprint representation that you can see in the footprint browser (Figure 17.22), I can confirm that the distance between pins 1 and 2 is about the same as what I measured with my calliper in the real component. As with Pcbnew, you can measure a distance by placing the mouse pointer on the centre of one of the pins, press the spacebar to reset the *dx*, *dy* and *dist* counters in the status line, and move the mouse pointer on the centre of the other pin.

To associate the footprint to the symbol, double-click on the footprint name (Figure 17.22).

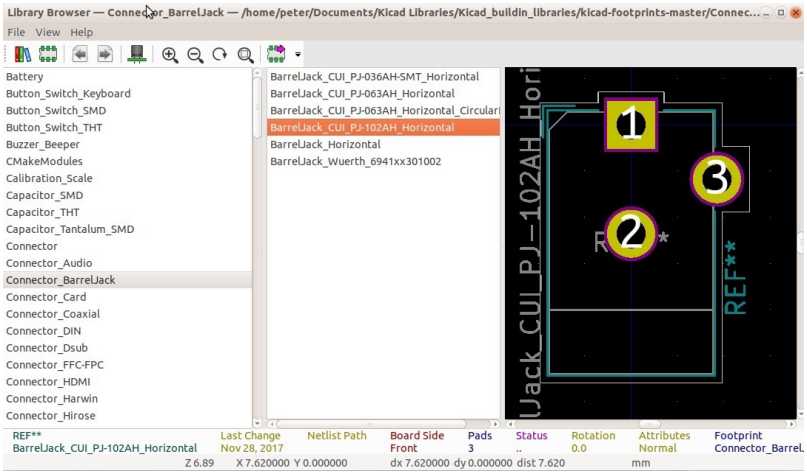


Figure 17.22: In the footprint browser, locate the appropriate footprint for the symbol.

You can confirm that the association is set in the symbol's properties window by looking at the Footprint field value (Figure 17.23).

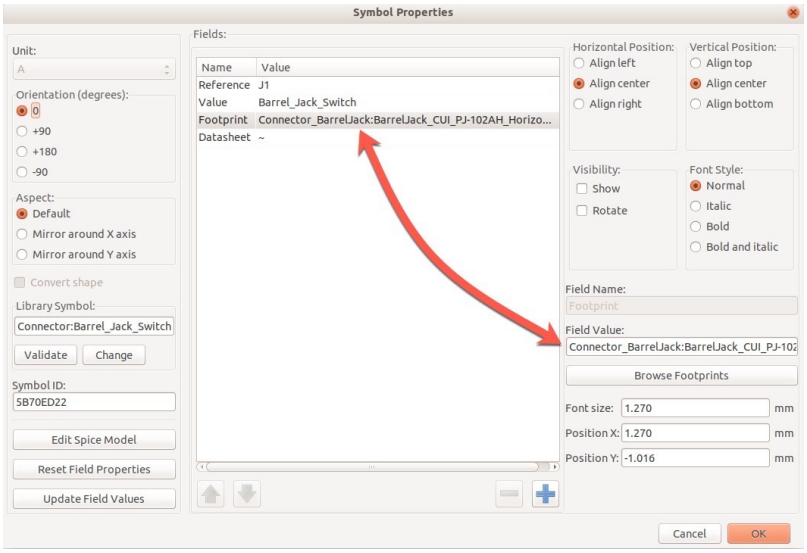


Figure 17.23: The footprint and symbol association is recorded in the symbol properties.

Let's continue this work using the Cvpcb. Click on the Cvpcb button from the top toolbar to start the app ( ). The Cvpcb window will open. You can see that the barrel connector and the voltage regulator symbols are already associated with a footprint. Go on to work on the rest.

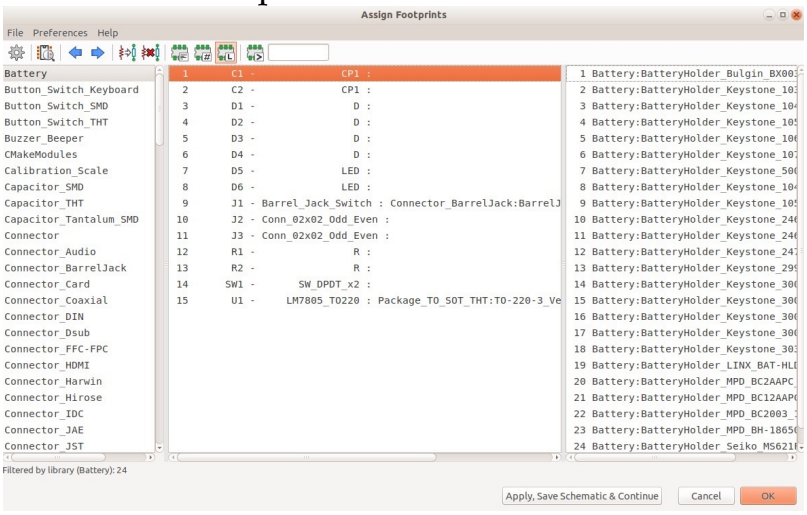


Figure 17.24: Cvpcb shows the remaining symbols for association.

Let's associate the two capacitors with their footprints. First, measure the diameter of their package, and the distance between the pins. I am using two cylindrical electrolytic capacitors. The larger has a diameter of 10.18 mm and pin pitch 5.19 mm, and the smaller a diameter of 6.46 mm and pin pitch 2.54 mm. In Cvpcb follow these steps:

1. In the middle pane, select C1. This will be associated with the large capacitor footprint.
2. In the left pane, find the Capacitor_THT library. This library contains through hole footprints.

3. In the right pane, search for a footprint of a capacitor with a diameter of around 10 mm.

In the top toolbar, Cvpcb offers various filters to help in finding footprints faster (Figure 17.25). You can experiment with them to understand how they work. The only one I am using at the moment is the one with the 'L' symbol on it. This is the library filter. It affects the items showing in the right pane. Only those that belong to the library selected in the left pane will show.

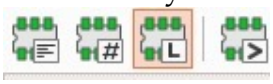


Figure 17.25: Cvpcb has footprint filters for faster browsing.

The one that best matches my particular capacitor has the string 'CP_Radial_D10.0mm_P5.00mm_P7.50mm'. This footprint has a diameter of 10 mm, and a pin pitch of 5 mm, exactly what I was looking for (Figure 17.26). Double-click on it to assign it to C1. To confirm you have the right part, you can also view the graphical representation of the footprint by clicking on the icon that looks like a microchip from the top toolbar (second from the right). You can then confirm these measurements.

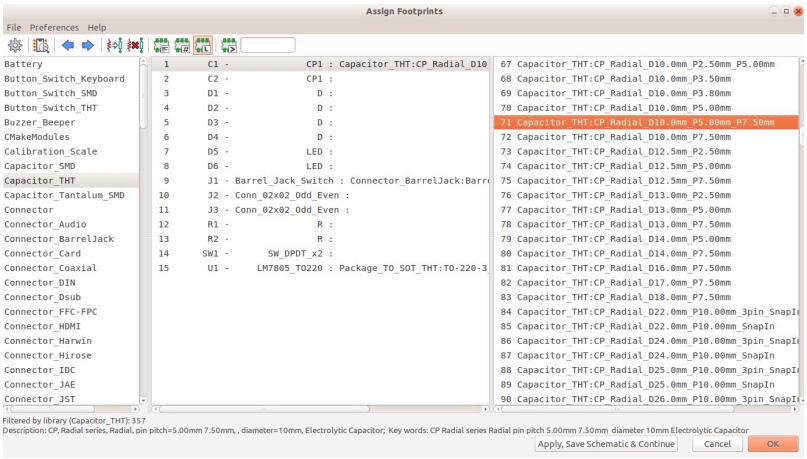


Figure 17.26: Found the footprint for the large capacitor.

Continue with the smaller capacitor, the diodes, LEDs and the resistors. Always measure and double check that you have the correct footprint. Although you can come back here and change the associations if needed, it really pays to get it right the first time.

In Figure 17.27 you can see the completed associations table. Click on 'Apply, Save Schematic & Continue' to save your work, and then 'Ok' to dismiss Cvpcb and return to Eeschema.

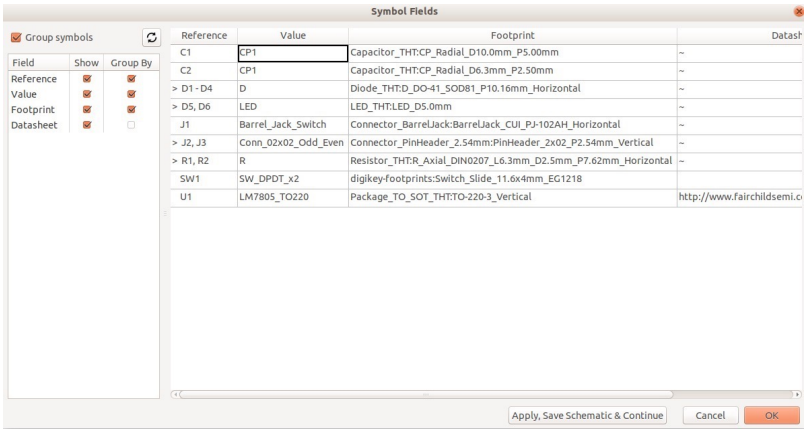


Figure 17.27: The completed symbol-footprint associations.

That was step 3 of the schematic design process. Next is the wiring.

Step 4: Wiring

Wiring is the part of the schematic design process I enjoy the most. It is also where most of the errors can happen. If not caught in time those errors can wreak havoc in later work, especially in the layout. While Eeschema can check for some basic errors like leaving a pin unconnected, or connecting a signal pin to a power source, it is up to the designer to ensure that it is correct.

To wire-up this project schematic, we will depend on Figure 17.3 for most of the guidance. The wiring of the additional symbols (like the LEDs and the barrel connector) are straightforward to do without a reference to guide us.

There is no particular method for the wiring. I typically start from the left of the schematic and make my way to the right. To the right of the schematic is the rectifier network (made up of the four diodes). I will wire that up first since the four diodes consist of a single functional block. I will then connect the barrel connector to the rectifier, and continue with the C1 capacitor and the voltage regulator.

In Eeschema, type 'W' or click on the wire tool button (). Wire the rectifier diodes and connect pin 1 and pin 2 of the barrel connector to the rectifier. Use the Unconnected tool () to mark pin 3 as unconnected (if you don't, the ERC will produce an error). In Figure 17.28 you can see the wiring progress at this point.

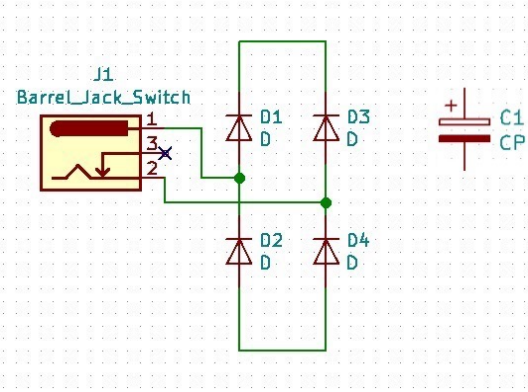


Figure 17.28: The progress so far.

Continuing towards the voltage regulator, connect the rectifier with the capacitors, the regulator, and the power indicator LED (Figure 17.29).

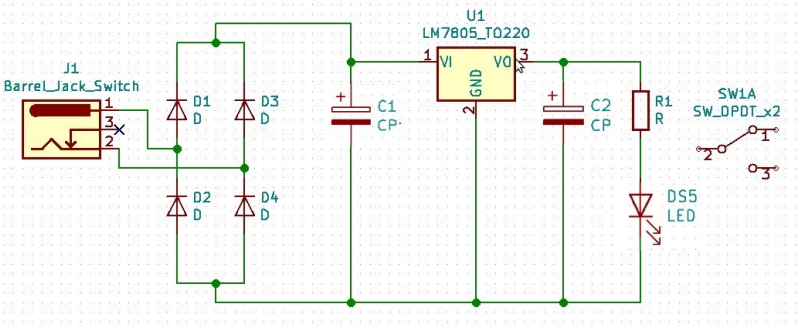


Figure 17.29: More progress.

Finish the process with the right side of the schematic, the output. This includes the switch, the on/off indicator LED and the pin connectors. You can see the completed wiring in Figure 17.30. Remember to mark pin 3 of the switch as unconnected. To improve the readability of the schematic, I also moved the switch symbol upwards.

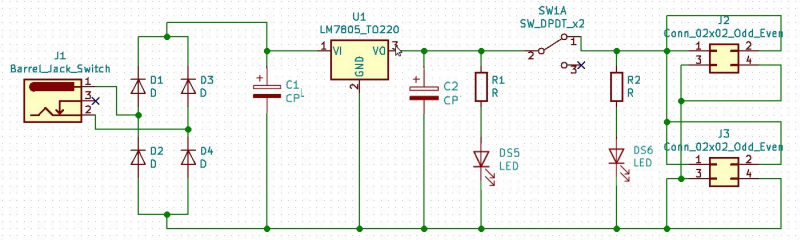


Figure 17.30: The completed wiring.

With the wiring complete, it is prudent to run an Electrical Rules Check now. In Figure 17.7, the ERC is shown as step 6, just before you will create the Netlist file. That is also a good time to do this check in order to detect problems that are introduced as you are tweaking the schematic. That is ok. Running the ERC now will help to actually complete the wiring correctly.

Click on the ERC button () to bring up the tool. Click on the Run button

to run it. The tool returns two common issues with the wiring, as you can see in Figure 17.31.

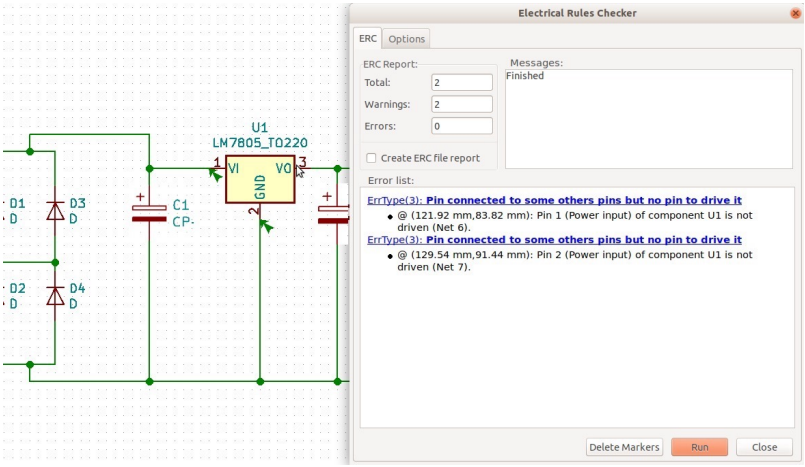


Figure 17.31: To common issues revealed by the ERC.

The ERC tool also marks the location of the two problems with green arrows. This is a common issue affect symbols with pins that are configured as power pins. Power pins must be connected to nets that are themselves marked with a power flag symbol. The power flag symbol informs KiCad that the marked flags convert power, not signals. To fix this type of error, attach the PWR_FLAG symbol to the nets that are connected to the pins marked with the green arrow. You can find the PWR_FLAG symbol in the symbols browser (Figure 17.32).

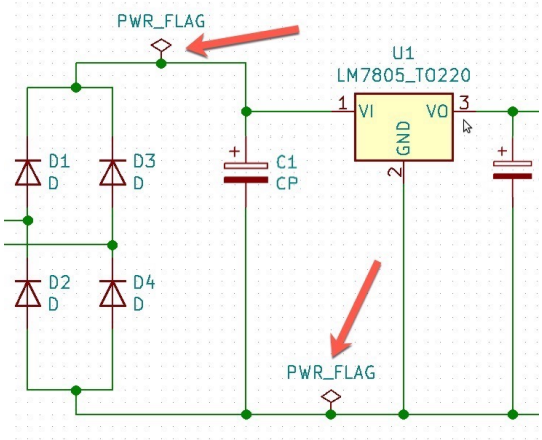


Figure 17.32: The PWR_FLAG symbols resolve the ErrType(3).

The PWR_Flag symbols can be connected anywhere in the net that is connected to the offending net; it is not necessary to place them right next to the pin. Run the ERC again to confirm that the two issues are resolved.

With all issues resolved, let's move on to the next step, naming the nets.

Step 5: Nets

Each wire (net) you created in the previous step is anonymous. Eeschema stores information about the nets in the '.sch' text file, which you can open and inspect with a text browser (Figure 17.33).

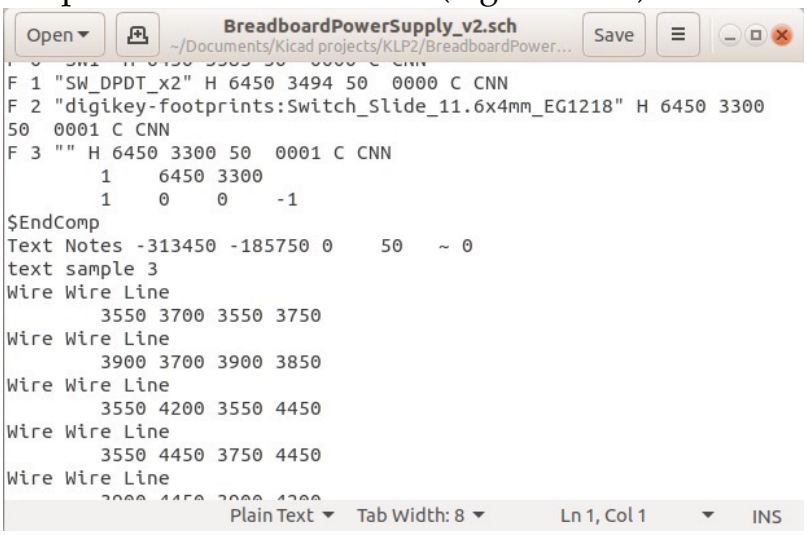


Figure 17.33: A segment of the project .sch file showing wire (net) information.

As you can see in Figure 17.33, each wire is defined by a set of coordinates. When you import the schematic information to Pcbnew with the use of the Netlist file, the nets are automatically given names such as 'Net-(D3-Pad2)'. Although they are human-readable, they are not very descriptive. Having descriptive names in Pcbnew make the layout process easier since you can know immediately what the purpose of a track is (power, signal) without having to follow it to the pads hunting for clues.

You can give nets human-readable names with the use of net labels. When you place a net label on a net, the value of the label becomes the net's name when you create the Netlist file. When you then import the netlist file to Pcbnew, the tracks are named after the nets, instead of the automatically generated names.

Let's go ahead to name the nets in this schematic. In particular, you should name the power nets (GND, 5V output, input voltage). The rest are not necessary to name, although you can if you wish.

Click on the net label tool from the right toolbar to select it (). Click on the net that conveys positive voltage from the rectifier network to the voltage regulator. A window will come where you can type in the name of this net. Since we don't know that input voltage (it will be within a wide range of values) let's call it 'Vin'. Type 'Vin' and click Ok (Figure 17.34).

Figure 17.34: Set the Vin net.

The schematic will look like the example in Figure 17.35 (partial, showing the 'Vin' label).

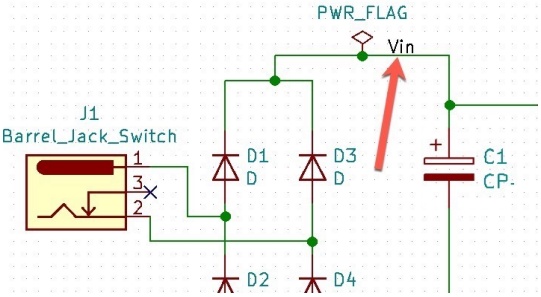


Figure 17.35: Added Vin net label.

To the same for 'V-'. I have also named nets V1 and V2 for the wires that come out of the barrel connector, and Vout1 and Vout2 for the positive voltage before and after the switch.

The result should be like the example in Figure 17.36.

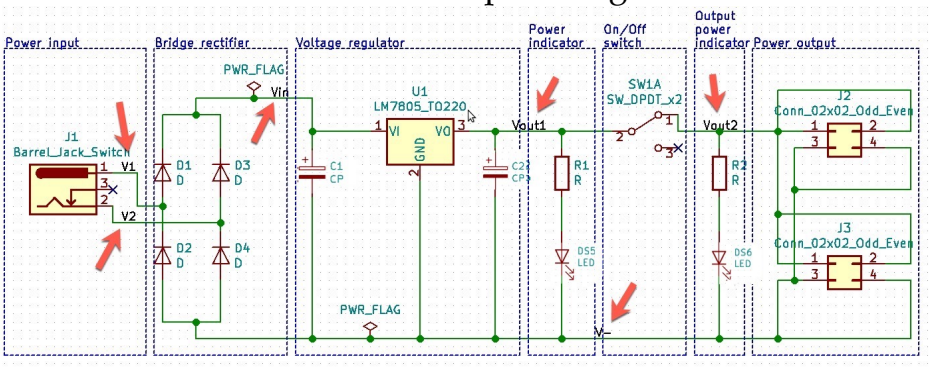


Figure 17.36: Added three net labels.

An added benefit of using net labels is that you can attach them to pins. Pins that are attached to labels with the same name are considered to be electrically connected even if there is no wire drawn between them. This is a technique that you can use in large schematics to de-clutter them. You will make use of this technique in the third project.

Step 6: Electrical Rules Check

With the wiring and the labelling complete, let's run the Electrical Rules Check to make sure that there are no defects. Click on the ERC button () to bring up the tool, and click on Run.

The result comes back empty, which is great! Let's move on to Step 7 where you will add comments and other graphical information to the schematic.

Step 7: Comments

In this step, you will add comments and other graphical information to the schematic. This information improves readability. This is especially important if you are working with other people, or are sharing your project.

Eeschema allows for text and rectangles only. Any such items are treated as comments are in a program: for the human reader only. They are not passed onto Pcbnew.

Start with the rectangle tool (). Use it to create rectangles around the functional groups of the circuit. The result is in Figure 17.37. I admit I have gone a bit overboard with this. Perhaps fewer boxes would be better.

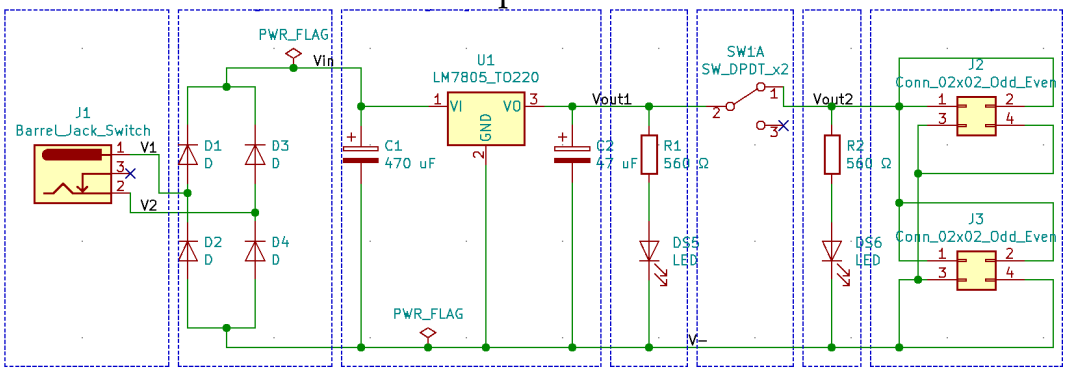


Figure 17.37: Functional groups marked with boxes.

Switch to the text tool () to annotate the boxes with a descriptive name. You can see the result in Figure 17.38.

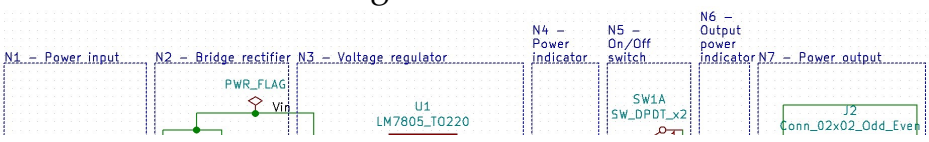


Figure 17.38: Each box has a name.

Your work to design this schematic is now complete. All that is left to do is to produce the Netlist file that you will use to import the circuit to Pcbnew and continue there with the layout.

Step 8: Netlist

The Netlist file contains the circuit information that Pcbnew needs in order to import the correct footprints and nets. Every time that you make a change to the schematic, you must create a new netlist file and import it again into Pcbnew. When you do this, Pcbnew is smart enough to be able to import only changes to the layout sheet. For now, we are going to work with a brand-new layout project, and generate the Netlist for the first time.

In Eeschema, click on the Netlist button in the top toolbar (

). This will bring up the Netlist window (Figure 17.39).

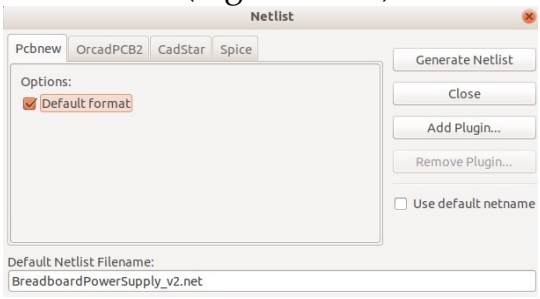


Figure 17.39: The Netlist window.

While the Netlist window provides the options to export into four different formats, we are only interested in KiCad. The 'Pcbnew' tab is selected. Leave the defaults as they are, including the file name, and click on the 'Generate Netlist' file. At the prompt, accept the default location for the file (the project folder). After the file is generated, you will be able to see it in the project's folder.

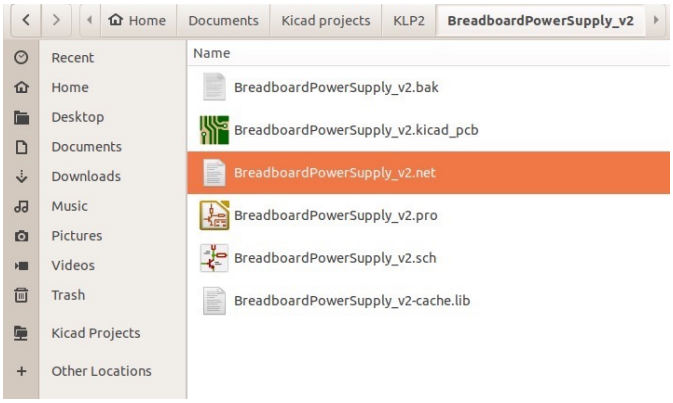


Figure 17.40: The netlist file has the '.net' extension.

The Netlist file has the '.net' extension. It is a text file that you can open using a text editor.

Time to move into Pcbnew and start work on the layout of your board.

17.3. Footprint layout in Pcbnew

Let's start the layout process. To do this, you will follow the steps illustrated in Figure 17.41.

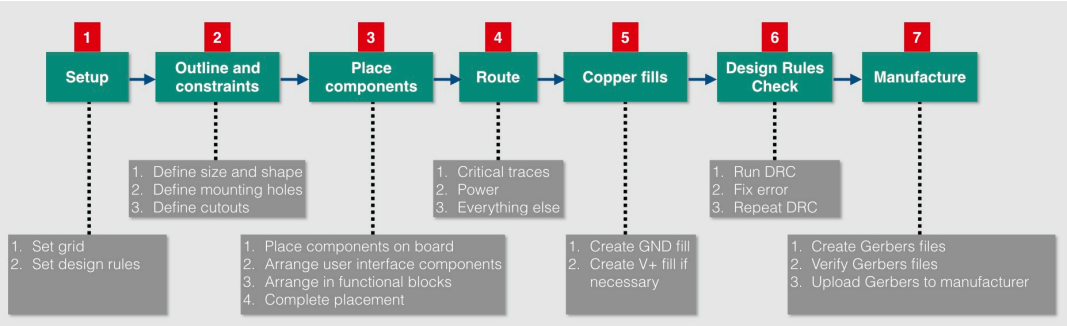


Figure 17.41: A serialised view of the PCB layout process.

Step 1: Setup

Begin by setting up your layout project. From the KiCad main window, click on the Pcbnew button.

First, set the page by opening the Page Setting window (Figure 17.42). You can access the window through the File menu.

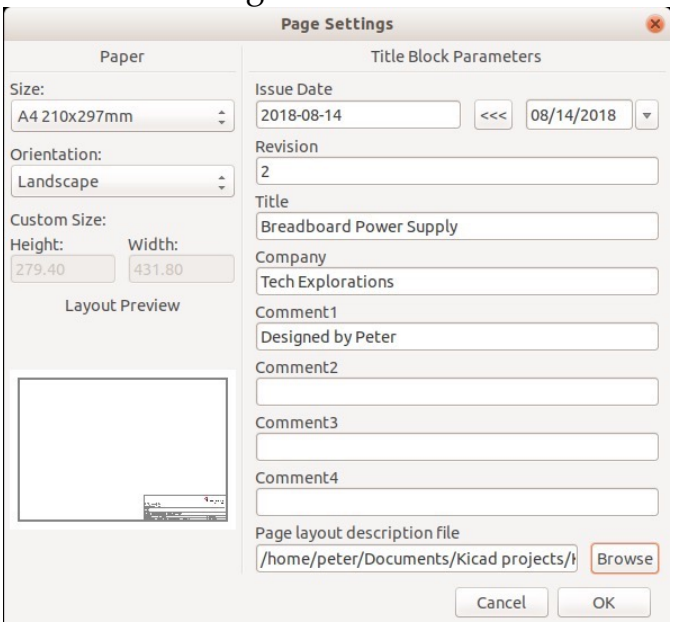


Figure 17.42: Setup the page layout.

Next, set up the grid size. You want a setting that is appropriate for the size of the board outline. The board's shape and size is tied to that of the mini-breadboard on which I want the power supply to plug. The two outmost rows of my breadboard have a distance of around 47.37 mm between them. I will set my grid size to a value that is a close fraction of this number. A reasonable value for the grid is 0.127 mm. I need 372.99 segments of 0.127 mm each to get a segment of 47.37 mm. I think that is accurate enough for this purpose (Figure 17.43).

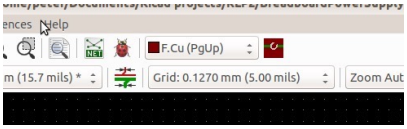


Figure 17.43: This grid looks appropriate for drawing the board border.

Next, let's decide on the number of layers for this PCB. The PCB is fairly simple, with a small number of through-hole components, which results in a small number of traces. A two-layer PCB will suffice for this. A new KiCad project contains two layers by default, but if you wish to confirm this, open the Layer Setup window from the Setup menu (Figure 17.44).

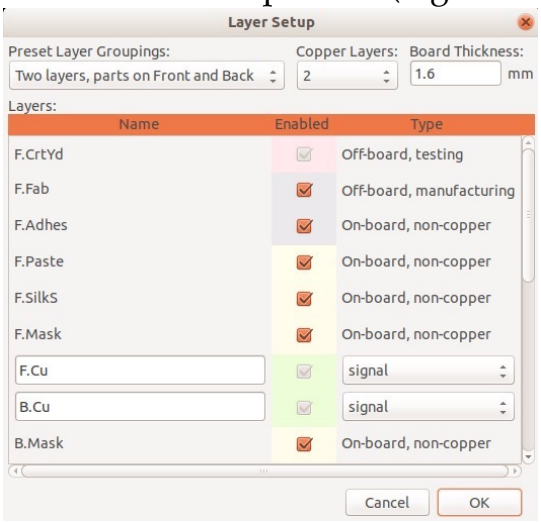


Figure 17.44: This board will contain two layers.

Before defining the design rules, you should import the Netlist file. This will allow you to set the net classes, in addition to the global design rules to satisfy the manufacturing requirements of your preferred manufacturer. Click on the Netlist import button (

). The Netlist window will come up. Since you are importing the Netlist to a new project, all the default settings are appropriate (Figure 17.45).

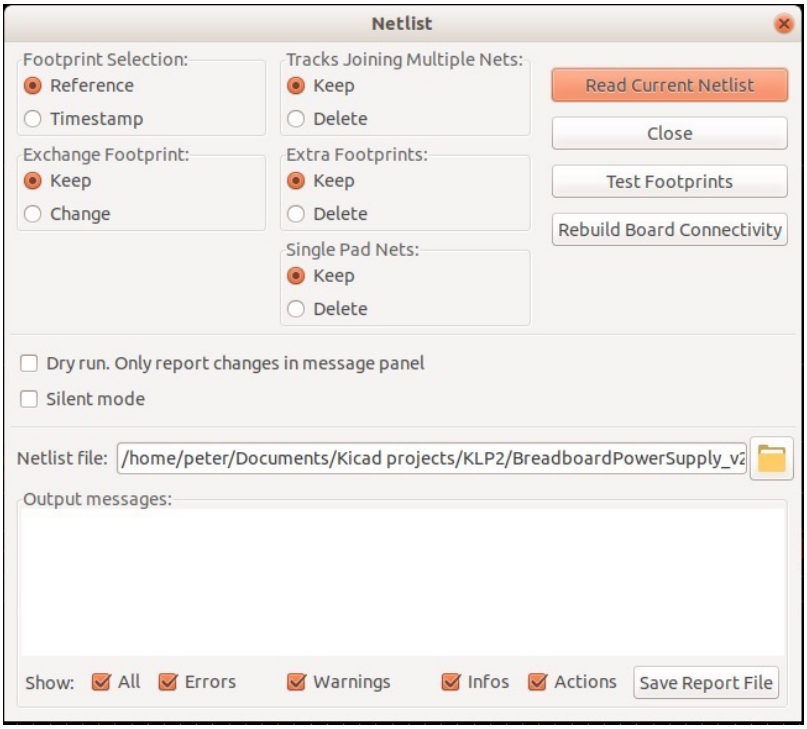


Figure 17.45: The Netlist window.

Click on the 'Read Current Netlist' button, and then 'Close'. The footprints are now in the layout sheet, as in the example in Figure 17.46.

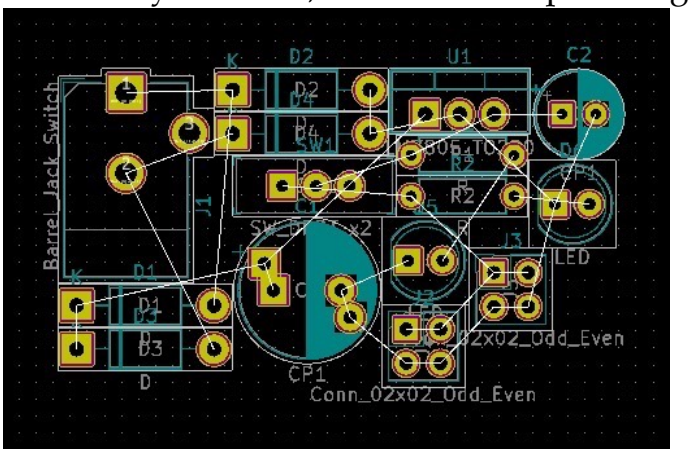


Figure 17.46: The footprints are arranged randomly in the layout sheet.

For now, leave the footprints as they are. Before arranging them in their final locations, you will need to draw the outline for the board. You will do that in Step 2. But before you do that, you should work on the design rules.

The manufacturer I am planning to use is Pcbway.com, and their PCB specifications are available on their website.[17] I have set my minimum design rules to match or exceed their specifications. I also took the opportunity to create a custom design rule for the power traces. I would like the traces and vias that belong to one of the power nets to be slightly larger than the signal

[17] https://www.pcbway.com/capabilities.html, visited December 12, 2018.

traces. You can learn how to create a custom net class, and then assign nets to classes in the recipe titled '27. Create custom net design rules'. You can also learn how to create custom via and track sizes in the recipe titled '26. Custom Global Design Rules and changing the width of a trace'. You can see my Net Class settings in Figure 17.47, and Global Design Rules settings in Figure 17.48.

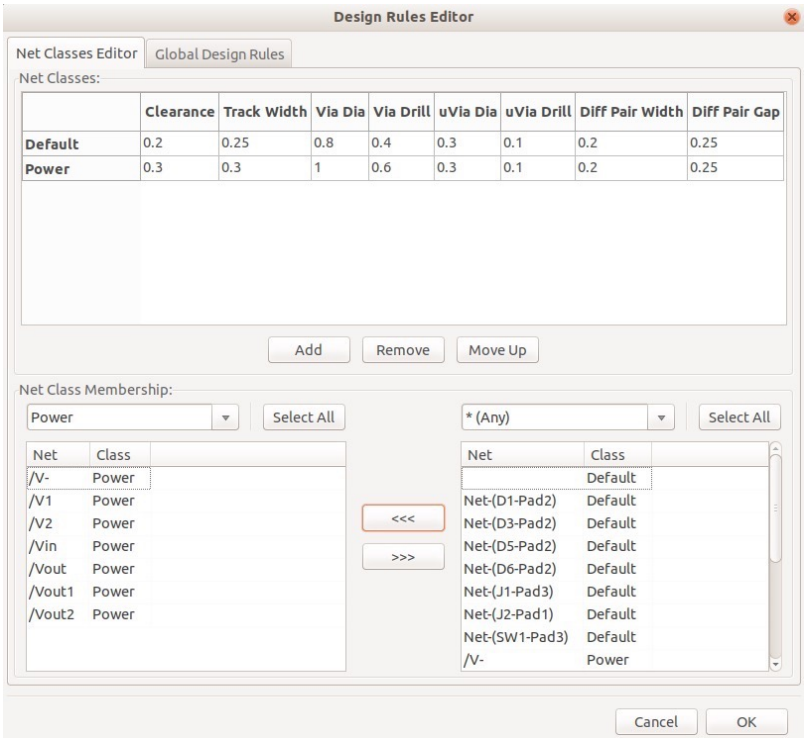


Figure 17.47: My net class settings for this project.

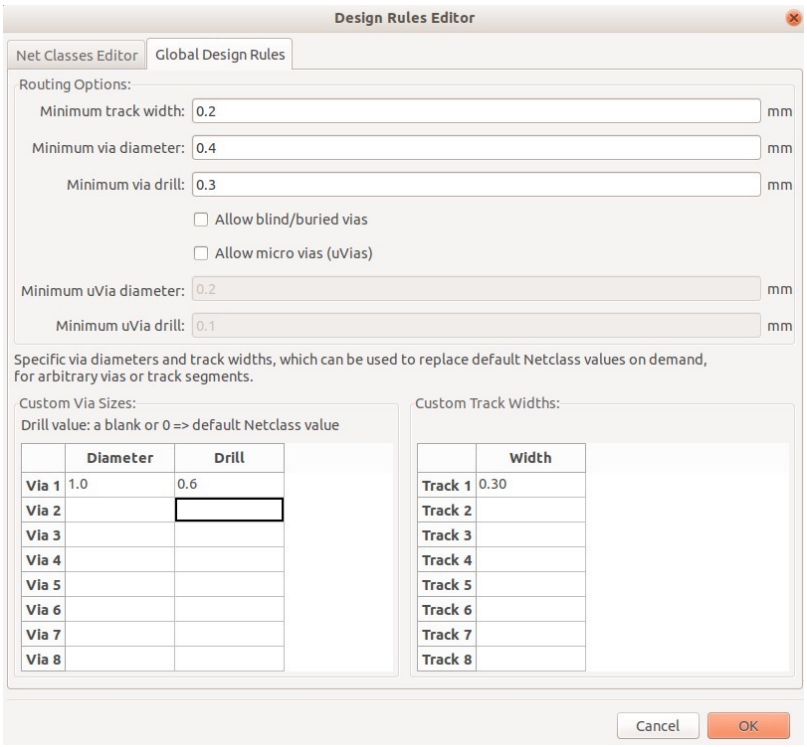


Figure 17.48: My Global Design Rules settings for this project.

Click Ok to exit the Design Rules Editor and save your project. Let's continue with Step 2, the outline and mechanical constraints.

Step 2: Outline and constraints

In Step 2, your objective is to draw the outline for your board. You can't start the placement of the footprints until you have defined its boundaries. For the board of this project, we have strict mechanical constraints because we want it to fit precisely with a mini-breadboard. Therefore, we will start the process of designing an outline for the board by taking a couple of simple measurements from the breadboard.

Because we want the breadboard power supply to connect to the power rails of the breadboard, the most important measurement is that one that gives us the distance between the two rows of holes at the top and bottom of the breadboard. I am using my calliper to measure this distance, as you can see in Figure 17.49.

Figure 17.49: The distance between the top and bottom rows is critical for this project.

My calliper measures this distance to be 47.50 mm. I tried to place the tips of the calliper tips in the centre of the holes. This measurement will help me place the output pins in the correct position in Step 3 of the process. I took one more measurement, to find out the width of the breadboard, 54.98 mm (Figure 17.50). This measurement will help me draw the outline of the board. I did not take a measurement of the length of the breadboard because that is not useful for the purpose of drawing the board outline.

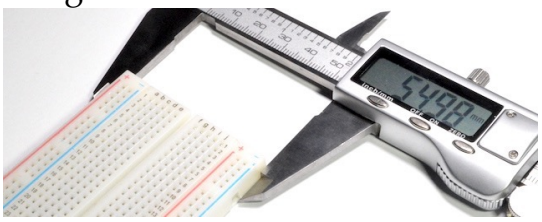


Figure 17.50: Measuring the width of the breadboard.

With the measurements complete, let's continue with Pcbnew. Select the Edge.Cuts layer from the layers manager and set the cursor shape to large. With grid at 0.127 mm and zoom level 3.05 (a good match for the board dimensions we are working with), click on the line tool to select it. I'd like the board to be slightly narrower than the breadboard so that its sides don't extend beyond the breadboard. So, I will draw the long side of the board around 52 mm. Place the mouse pointer in the sheet, press the space bar to zero the dx and *dy* counters, and draw a vertical line that is as close to 52 mm as you can get it. The *dx* counter should read 0, and *dy* should read 52.324 mm (Figure 17.51).

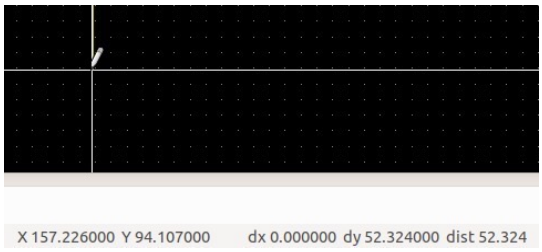


Figure 17.51: Drawing the first line of the outline.

Press the space bar to zero the counters, and continue horizontally with a 29.972 mm line (Figure 17.52.

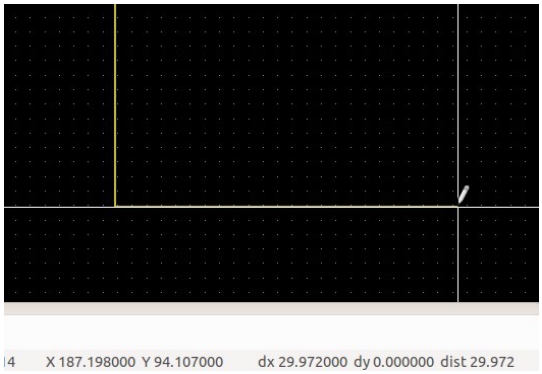


Figure 17.52: Drawing the second line.

Again, press the space bar to zero the counters (a habit you will soon develop is to zero the counters every time you draw a line), and move the mouse up by around 7 mm.

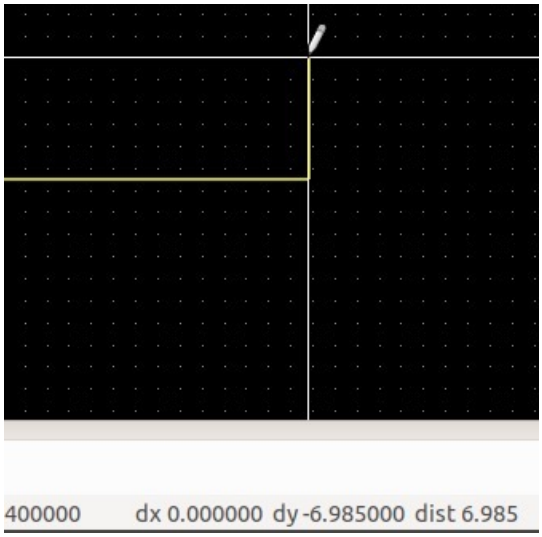


Figure 17.53: Drawing the third line.

Continue to draw the outline of the board, until you have the final closed polygon, as in the example in Figure 17.54. I have marked the dimensions within the layout sheet using the 'Add dimension' tool ( ).

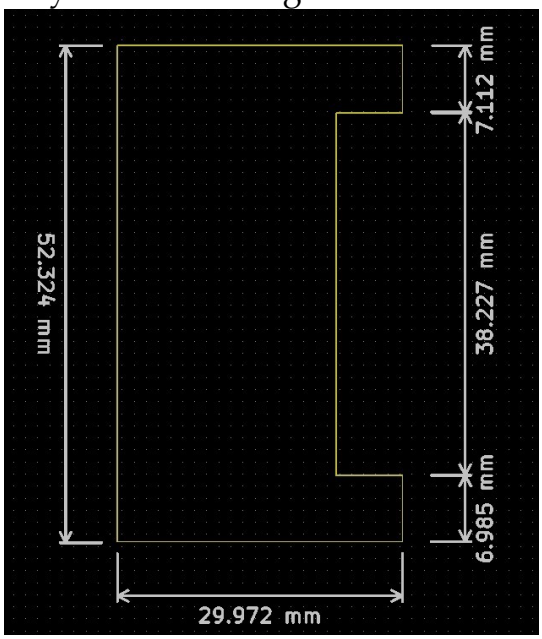


Figure 17.54: The final outline.

You can see the 3D rendering of the board by clicking on View, '3D Viewer'. If your outline is not a closed shape, you will get an error message from the 3D viewer. It will give you the coordinates of where the outline is open so you can fix it. You should not continue from this point if the outline is not closed.

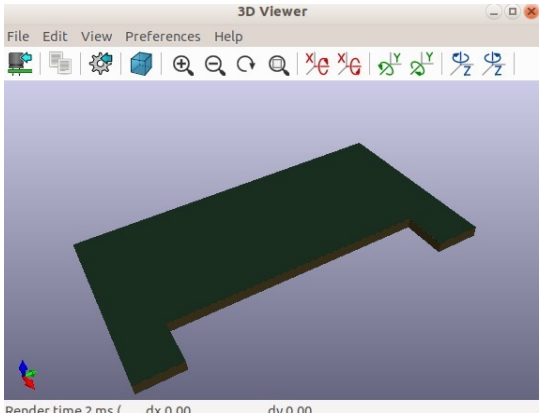


Figure 17.55: A 3D rendering of the final outline.

The outline is now complete. Because we don't plan to mount this board in a project box, there is no need for screw holes or other mounting provisions. We can continue with Step 3, and arrange the footprints within the board.

Step 3: Place components

Start the placement with those footprints that make up the board's user interface. Those are the barrel connector, the slide switch, the LEDs, and the pins that connect the board to the breadboard. Place the barrel connector along one of the corners of the board and is away from the breadboard so that the cable does not interfere with your prototyping. Place the slide switch and LEDs along the inside edge of the board, so that it is easier to reach. In Figure 17.56 you can see the placement of those footprints. I have also placed the headers in their final positions so that their pins fit precisely in the breadboard's power rails.

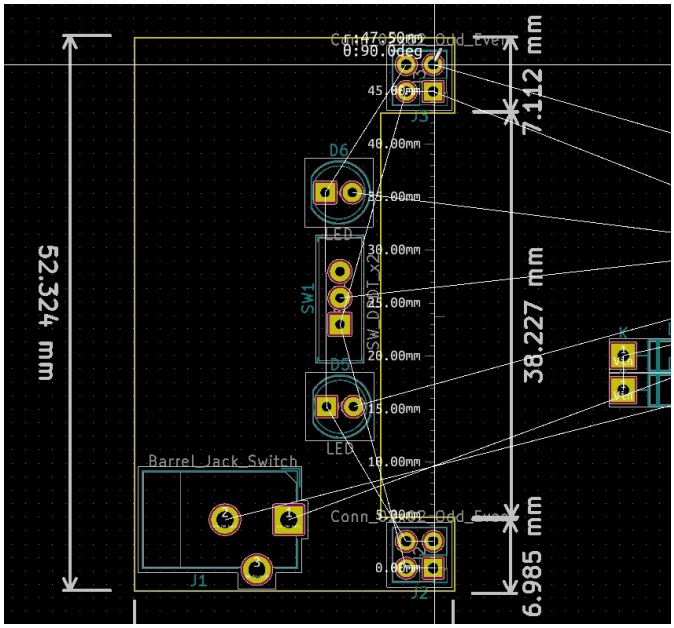


Figure 17.56: The user interface footprints in their final positions.

The exact positions of the barrel connector, LEDs and switch are not important in the sense that they can be placed in a variety of locations. However, the position of the connectors must be very accurate, else they will not align well with the breadboard. In Figure 17.56, I used the measuring tool to double-check that the distance between the outer pins is correct. You can also use the *dx* and *dy* counters in the status bar to help with measuring the distance during positioning. Finally, I have aligned the header footprints to the right and locked them in place using the context menu[18] (click on both footprints while holding down the Shift key to select them, and then right-click). Locking them in place will prevent them being moved by accident.

Figure 17.57: Align footprints to the right, and lock them in place.

Continue with the rest of the components. As much as possible, try to keep functional groups of footprints together, allowing for the available space. In Figure 17.58 you can see the final positioning of the footprints in my project. During placement, observe the thin ratsnest lines that represent the nets. Eventually, you will have to replace the ratsnest lines with traces, therefore you should try to position the footprints so that these lines are as close as possible. When a ratsnest line travels over a footprint, it is an indication that you will have a trace that travels over a footprint and that may need vias or complicated paths to make it possible. If you invest a bit of time to place footprints in a way that ratsnest lines are short and are not crossing over other footprints, your routing work will be significantly simplified.

You should also use the Align/Distribute context menu to place the footprint in precise positions in relation to each other. If you try to do this 'by

[18] The Align/Distribute context menu only appears if you have selected more than one footprints.

eye', you risk having components that look misaligned in the final populated PCB.

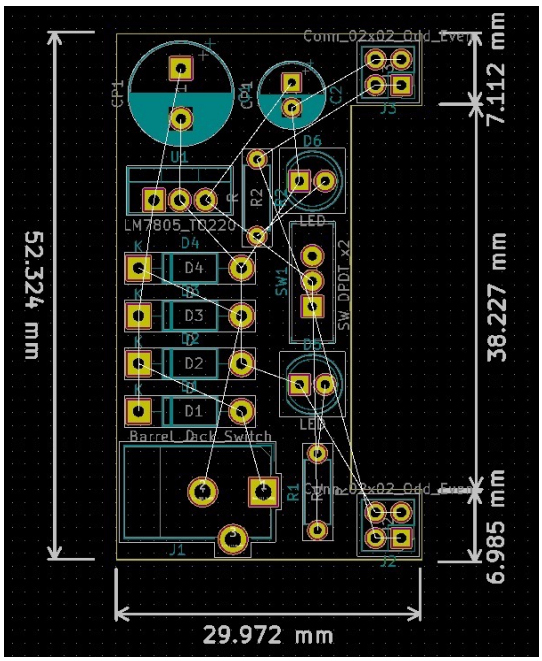


Figure 17.58: The final placement of the footprints on the board.

Step 4: Route

In Step 4, you will replace the ratsnets (that highlight the pads that should be connected with copper) with traces. While it is possible to use an autorouter to do this (as you can learn in the recipe titled '40. Using an autorouter'), it is beneficial to route at least the first few boards manually. This way you will gain valuable experience and skill that you will need to correct poor autorouter output or route unconnected pads.

As per the process in Figure 17.41, start by routing any critical traces, continue with power traces, and finish with everything else. Before you draw a trace, check that the trace width is correct.

In this board, there are no traces that I would call 'critical', such as an embedded antenna. Therefore, let's go straight into drawing the power traces. Start with the GND level power traces. Select B.Cu from the layers menu, the 'Route Tracks' tool from the right toolbar, and draw the traces that start from pad 2 of the barrel connector (Figure 17.59).

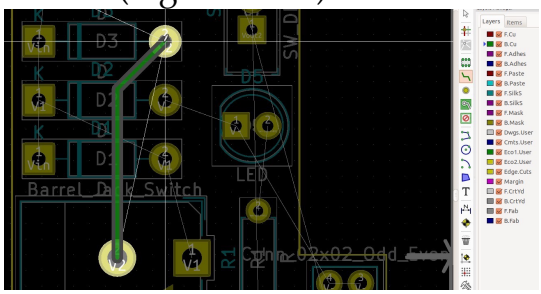


Figure 17.59: Creating a route that belongs to the Power net class.

Pause here for a minute to examine the trace you just created. Its net belongs to the Power net class, so according to the net class rules you defined in Step 1, its width should be 0.30 mm. You can confirm that by viewing the track's properties. Place your mouse over a segment of the trace, and type 'E'. Look at the Width property. Is it saying '0.3 mm'? Great!

Still working in the B.Cu layer, continue to create the traces for all the GND-level nets. You can take a look at the schematic diagram as you go drawing the traces. In Figure 17.60 you can see the current state of the board, which has all GND-level tracks completed, drawn in the B.Cu layer.

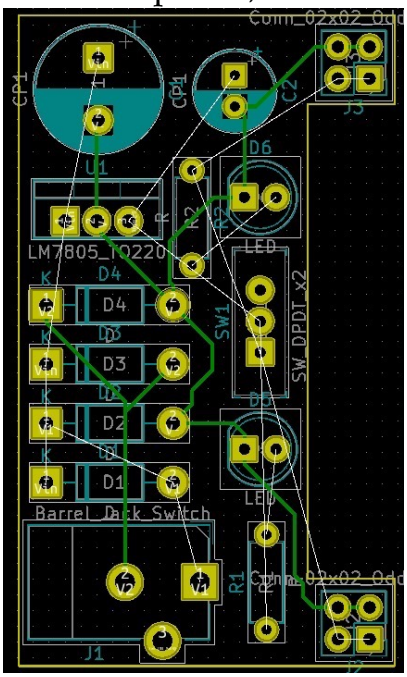


Figure 17.60: GND-level traces in the B.Cu layer are completed.

Continue with the F.Cu layer, and the rest of the nets in the power class. The result is in Figure 17.61.

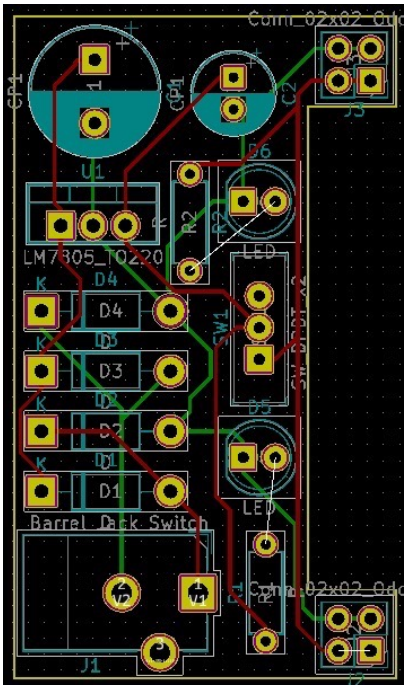


Figure 17.61: Non—GND level traces in the F.Cu layer are completed.

There are only a couple of traces left to do. Trace them to complete the routing step (Figure 17.62).

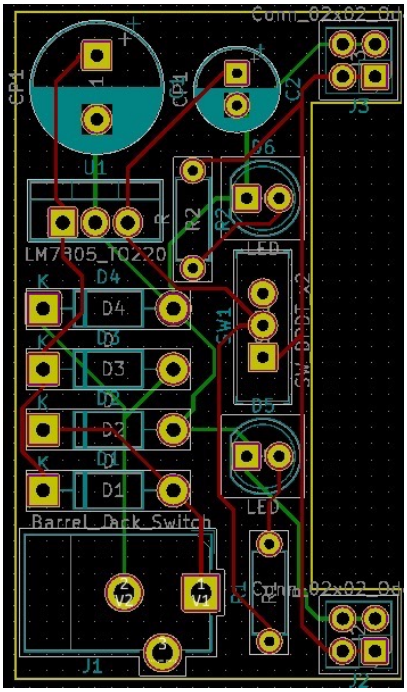


Figure 17.62: All tracing is complete.

Before you move on to Step 5, do an ERC to ensure there are no defects. My project looks good so far; no problems detected, and no unconnected pads (Figure 17.63).

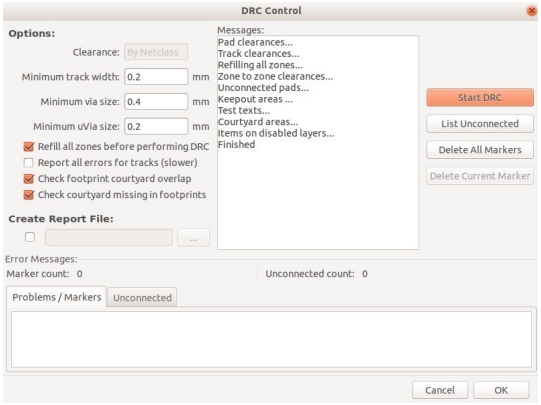


Figure 17.63: The ERC have given the all-clear.

Step 5: Copper fills

In this step, you will create a copper filled area in the bottom copper layer, and attach it to the GND-level net.

Start by selecting the B.Cu layer from the layers manager. Then, click on the 'Add filled zones' button from the right toolbar. Click on the top-left corner of the board to start the drawing of the polygon (Figure 17.64).

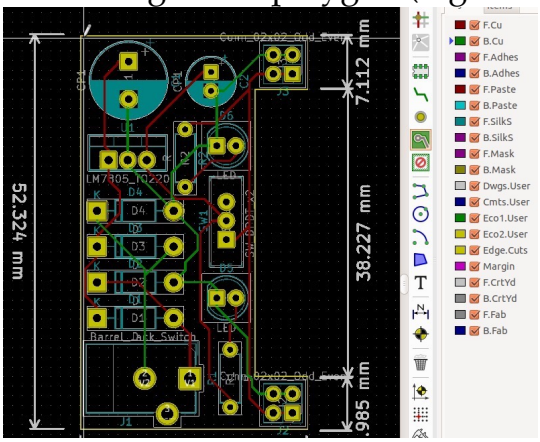


Figure 17.64: Start to draw the copper fill polygon.

The Copper Zones Properties window will appear (Figure 17.65). Ensure that the B.Cu layer is selected. Because you want this copper fill to be connected to the ground level net, select 'V-' from the Netlist. The rest of the properties are appropriate for our project. Feel free to experiment with them if you wish.

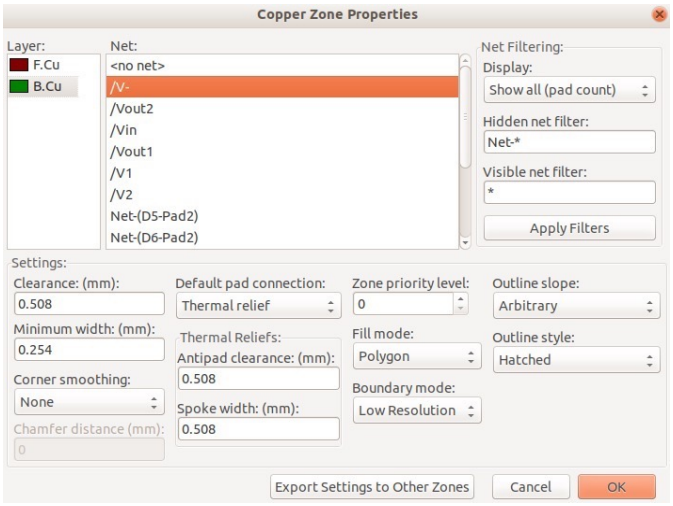


Figure 17.65: The Copper Zones Properties window.

Click on the 'Ok' button to dismiss the properties window, and continue with the drawing. You should see that a thin line connects the mouse pointer with the location of your last click. Place the mouse pointer to the next location to define a new edge for the copper fill polygon, and click again to draw it. Continue to point and click to draw new fill polygon edges until you have a closed polygon that follows the perimeter of the board. Eventually, you should have something like the example in Figure 17.66. The copper fill polygon appears in green. It is not filled yet.

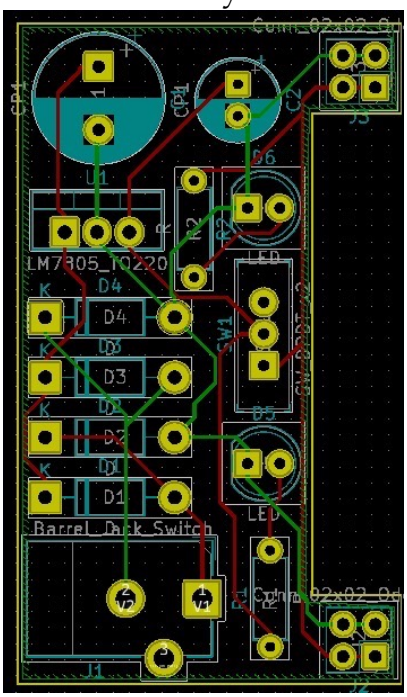


Figure 17.66: The copper fill polygon, unfilled.

To fill the area with copper, place your mouse over the green line and right click to bring up the context menu. Click on Zones, Fill, to complete the filling process (Figure 17.67).

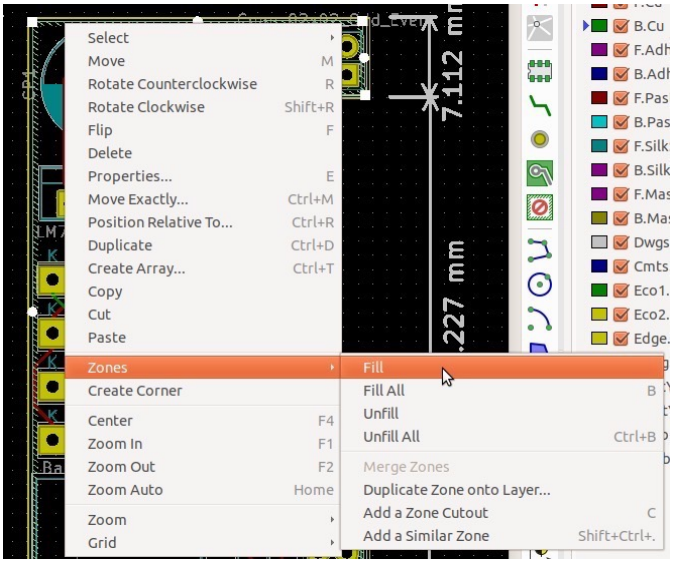


Figure 17.67: Fill the zone with copper.

To see the effect of this operation on your board, click on the 'Show filled areas in zones' button from the left toolbar. You can see the outcome in Figure 17.68.

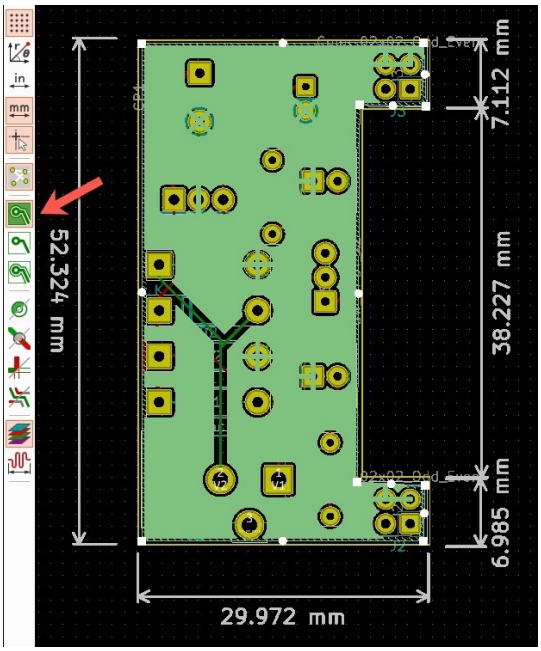


Figure 17.68: The copper-filled area in the B.Cu layer.

You will be able to see the copper filled zone in the 3D representation of the board (via the View menu, '3D Viewer').

Run another ERC to make sure there is no introduced defect, and then continue with Step 6, the silkscreen.

Step 6: Silkscreen

In Step 6, you will add text and graphics to the top and bottom silkscreen layers. This will improve the usability of your board by providing

usage instructions and other helpful annotations, and will also make your board look finished and professional.

I usually start by adding my logo, board name, and version, and continue with other text. I also take the opportunity to move, resize, or hide any silkscreen elements that the footprints contribute.

Let's begin with the logo. There is a recipe titled '29. How to add a custom logo to the silkscreen' that explains the process in detail. Here, I will assume that you already have an appropriate logo converted to a KiCad footprint, added to your footprint library and ready to use. I would like to place my logo on the F.SilkS layer so that it is visible when I am working with the board.

Select the F.SilkS layer from the layer manager, and then click on the Add Footprints button from the right toolbar (or just type 'O'). Bring up the footprint browser, and navigate through the libraries to find your logo. I selected a small version of my logo and placed it in between the two capacitors where there is a small amount of available space. In Figure 17.69 I have hidden the copper filled area to make it easier to see the silkscreen. The 3D Viewer gives a good perspective of what the logo will look like on the board.

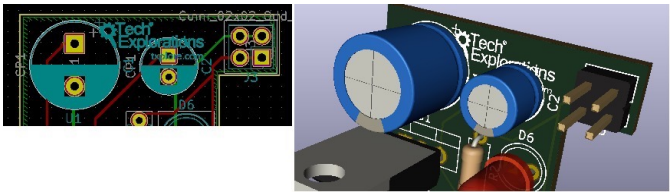


Figure 17.69: The logo, in position.

While I am working with the logo, I will continue to add a larger version on the bottom silkscreen, as well as some text with the board name, version, and contact information. I also added the KiCad and OSH logos. Select the B.SilkS layer before you do all this. When you add elements in the bottom silkscreen, remember to edit their properties and select 'Back' from the board side group (Figure 17.70).

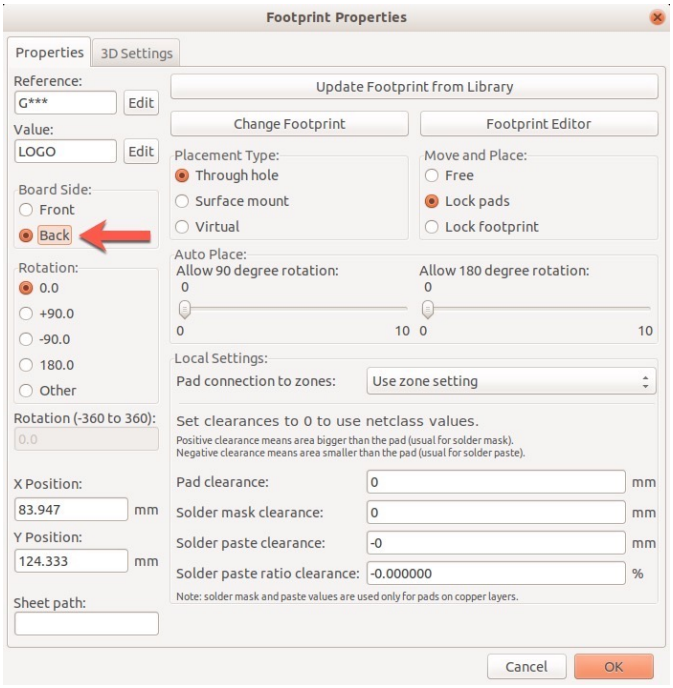


Figure 17.70: Remember to select 'Back' when you add elements to the B.SilkS.

The silkscreen (front and back) looks like the example in Figure 17.71. I have also provided the 3D rendering of the back of the board.

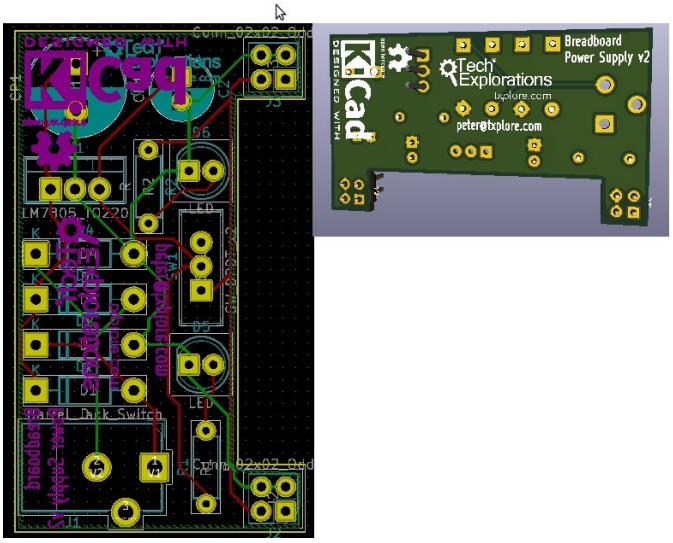


Figure 17.71: Done with logos and informational text.

Continue with tidying up the rest of the silk screen elements. I moved F.SilkS elements that belong to the various footprints so that they are more visible (some of them happened to be outside the outline of the board), and don't overlap.

Remember that during the schematic design, I added the values of the capacitors, resistors and other components using the symbol properties window. For example, I set "470 uF" for the value of one of the capacitors. These values have been carried across to the layout editor. If we want to make

them visible on the PCB, we can do so easily. Let's have a look at how to do that with the capacitor.

In Figure 17.1, the arrow points to the value of the symbol that was imported into PCBnew from EESchema. By default, PCBnew will place it in the F.Fab layer. This layer is typically used as part of the project's documentation, or to provide additional information to the fabrication house, but is not actually printed on the PCB for the end-user to be able to see.

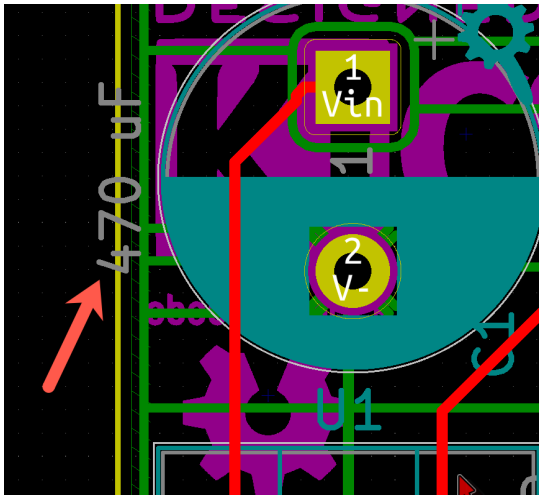


Figure 17.72: The value of the symbol is placed in the F.Fab layer.

If you wish to print at least some of the footprint values on the final PCB, you should change the layer on which these values exist. Let's do this for the 470 uF capacitor. Please your mouse over the value text, and type "E" to get the footprint text properties window. From the Layer dropdown, select "F.SilkS", and click Ok.

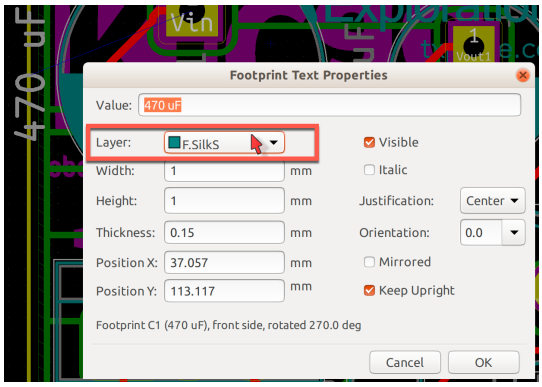


Figure 17.73: Set the Layer property of the footprint text to "F.SilkS" so that it is printed on the PCB and visible to the end user.

The value is now visible in the silk screen. Most likely, you will need to position it appropriately so that it exists inside the edge of the PCB. Use the "M" (move) and "R" (rotate) hot keys to do this.

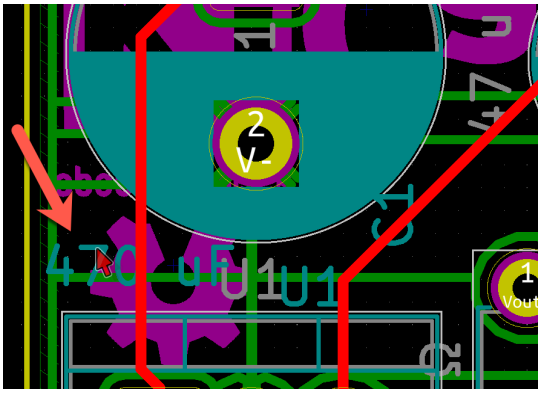


Figure 17.74: I have moved the footprint value text so that it is inside the PCB

You can see the final result in Figure 17.75. I have hidden the bottom silkscreen to produce a clearer view of the front.

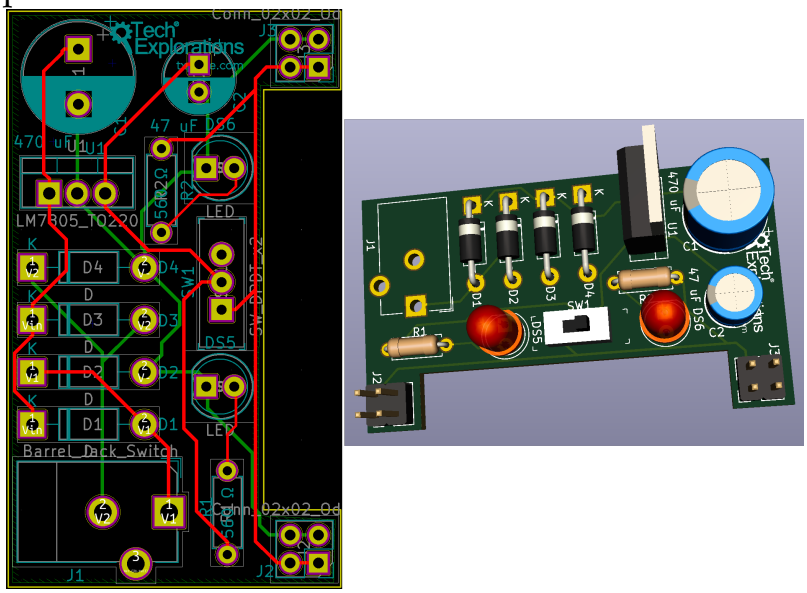


Figure 17.75: The work on the silkscreens is completed.

Step 7: Design Rules Check

We are now one step closer to manufacturing. Before we do that, let's make sure nothing is broken by running the DRC one last time.

The DRC does complain about a few things, as you can see in Figure 17.76. The DRC uses small red arrows to point to the offending locations.

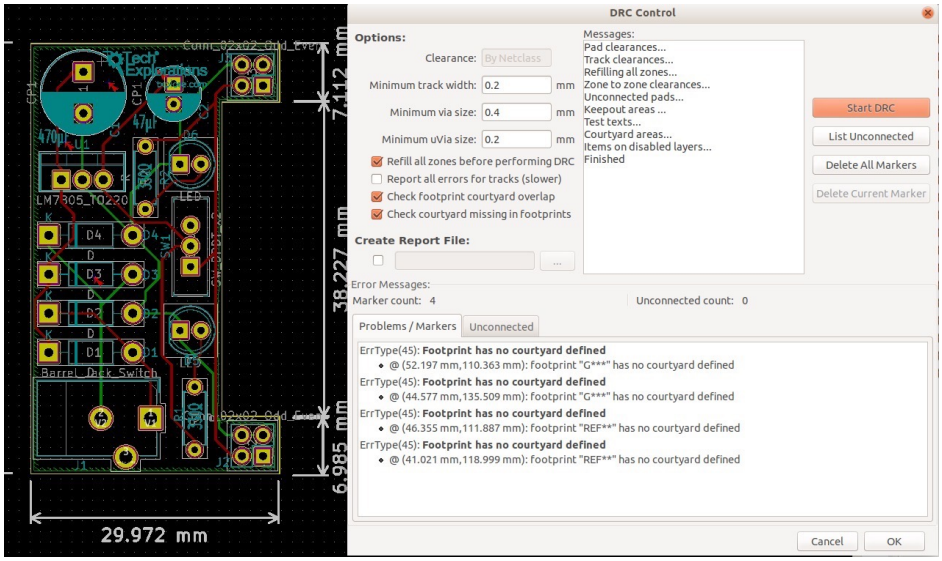


Figure 17.76: The DRC is complaining about missing courtyards.

The problem has to do with missing courtyards of the added graphics. In particular, the four logos I added are treated as footprints by Pcbnew, even though they only have a silkscreen layer. Because they are treated as footprints, a courtyard is expected, and not finding one triggers an error message.

You can safely ignore these error messages and move on to the next step.

Step 8: Manufacture

Let's order this board! I will Use Pcbway for this board. To complete this step you will need information available in recipes '31. How to make and test Gerber files' and '32. How to manufacture a PCB with PCBWay'.

In summary:

1. Generate the Gerber files, save them in a new folder
2. Create a Zip archive of the Gerber files folder
3. Upload the Zip archive to Gerblook for a final check
4. If no issues are detected, place your order with your manufacturer

The details of my order to Pcbway is showing in Figure 17.77. Take care to enter accurate size information so that you can receive an accurate quote. I used the Measure Distance tool for this purpose.

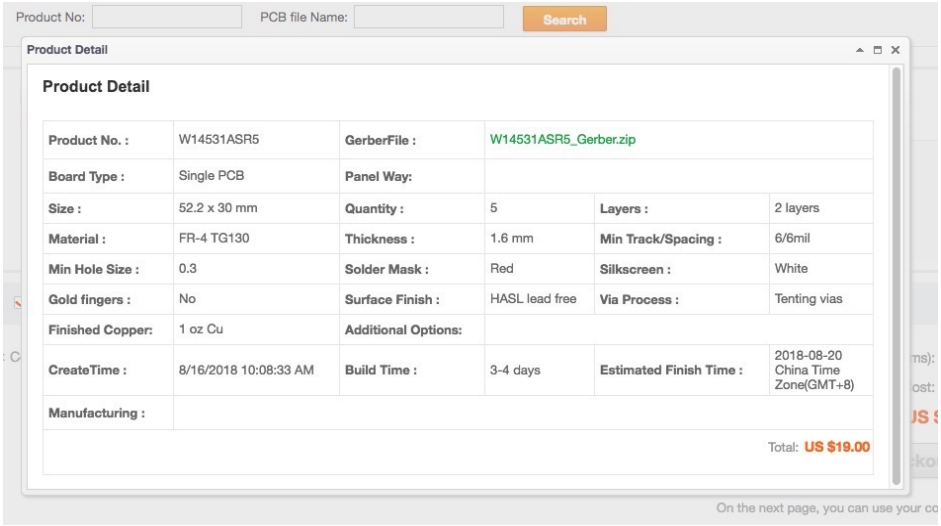


Figure 17.77: My order details to Pcbway.

17.4. Project extensions

Try these alterations to the first project:

1. Replace the 90-degree corners with rounded corners.

2. Replace the resistors, LEDs, diodes, and capacitors with surface-mounted components.

18. Project 2: Design a small Raspberry Pi HAT

18.1. What you will build and list of parts

My motivation for creating this simple Raspberry Pi HAT, is to make it easier for students of my course Raspberry Pi Full Stack to connect the course hardware to their Raspberry Pi. Using this HAT, students will be able to simply snap-on the board, and continue with their study, instead of spending time manually assembling the components on a breadboard.

In this project, you will create a tiny board for the Raspberry Pi. The board will contain four SMD components (three resistors and an LED) to help in reducing its size, a button, and a DHT22 sensor. We'll use a standard 2.54 mm pitch header with two rows to connect the board to the Raspberry Pi. To secure the board on the Raspberry Pi, we'll also include a mounting hole that matches the one that is located on the Raspberry Pi board (Figure 18.2).

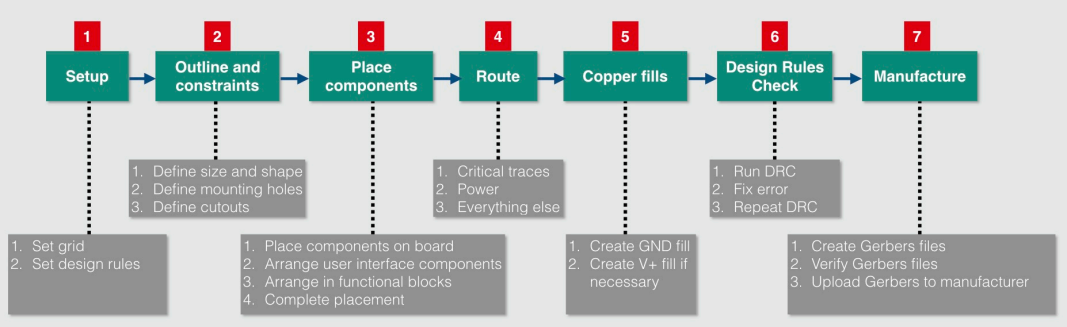


Figure 18.1: A serialised view of the PCB layout process.

An interesting aspect of this project is that instead of following precisely the layout process depicted in Figure 18.1, you will iterate between steps 2 and 3.

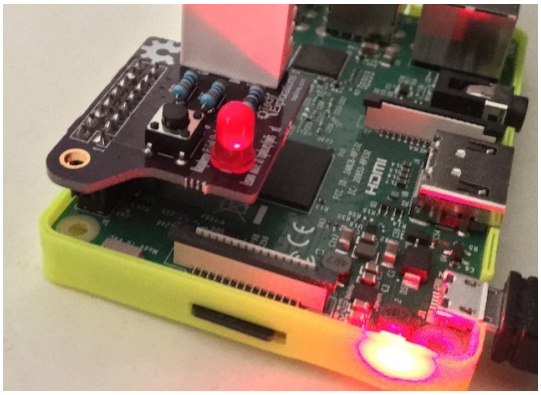


Figure 18.2: The board of this project will be similar to the one in the photograph, except that we will use SMD versions of the resistors and LED.

You will use your experience from the first project to speed up the process, and at the same time extend your skills.

The board will contain these components:

1. An 8x2 header, female, with the standard 2.54 mm pitch.
2. A TH momentary button.
3. Two 10 kΩ resistors, SMD 0805 package.
4. One 330 Ω resistor, SMD 0805 package.
5. One red LED, SMD 0805 package.
6. One DHT22 sensor.

For the SMD components, I have chosen the 0805 package because that is large enough to make hand-soldering possible.

Let's begin.

18.2. What you will learn

In this project you will learn and practice:

- Adding surface-mounted modules (SMD) to your board
- Minimising the amount of space needed for a board
- Creating non-rectangular shapes for the board outline
- Adding mounting holes
- How to revise the board outline after you place the footprints in order to conserve space

18.3. Project repository

This project has a Git repository. You can access it at txplo.re/klpp2.

This repository contains multiple commits at each step of the design process, as I was going through it. Feel free to clone this repository on your computer and make any modifications you feel like.

18.4. Schematic design in Eeschema

Start a new KiCad project, and store the project files in a new folder. I called mine 'RPi FS HAT.'

Step 1: Setup

Start by creating a new Eeschema file. Setup the page use the Page Settings window (accessible via the File menu). You can see my setup in Figure 18.3.

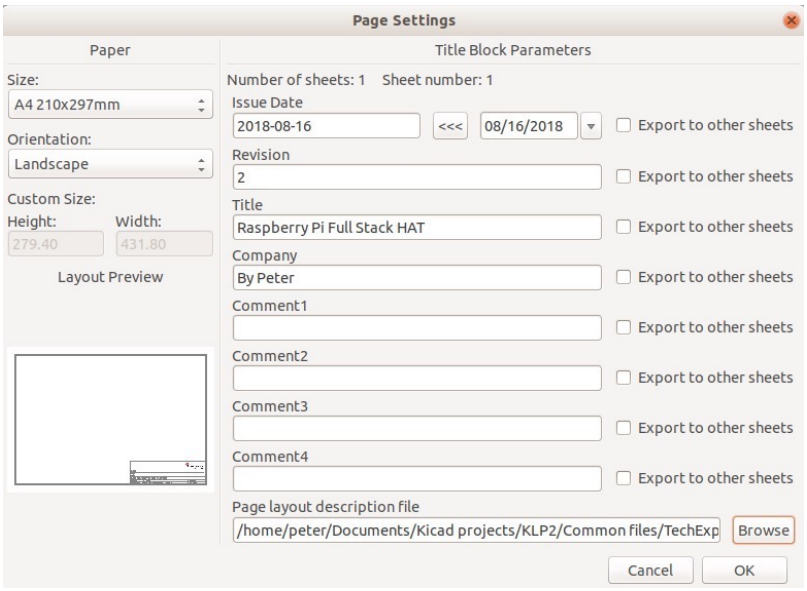


Figure 18.3: The page settings.

Continue to Step 2.

Step 2: Symbols

Find the symbols for the components listed below, and place them on the sheet.

1. A Raspberry Pi B+ (search for the Raspberry symbol).
2. A TH momentary button, (search for the DPST symbol).
3. Two 10 kΩ resistors, SMD 0805 package (search for the 'R' symbol).
4. One 330 Ω resistor, SMD 0805 package (search for the 'R' symbol).
5. One red LED, SMD 0805 package (search for the LED symbol).
6. One DHT22 sensor (search for the DHT11 symbol).

You might be wondering about the Raspberry Pi B+ symbol being listed above instead of the 2x8 header. And perhaps, why are we using a 2x8 header instead of a 2x20 header that the Raspberry Pi B+ has. The answer to the full question is that in the schematic, I would like to depict the wiring of the

circuit with an actual Raspberry Pi symbol that shows the full pin details, such as the GPIO and pin numbers. Because of KiCad's flexibility, I can then associate the symbol with any compatible footprints. Instead of going for the full 2x20 pin header, I opt to go for the 2x8 header. This is because the circuit only uses pins in the first half of the header, and since I want to minimize the size of the board, it is beneficial to use the smallest possible header than the full header.

The Raspberry Pi header is available from the Connector library. You can find it quickly if you search for it in the Symbol chooser window (Figure 18.4).

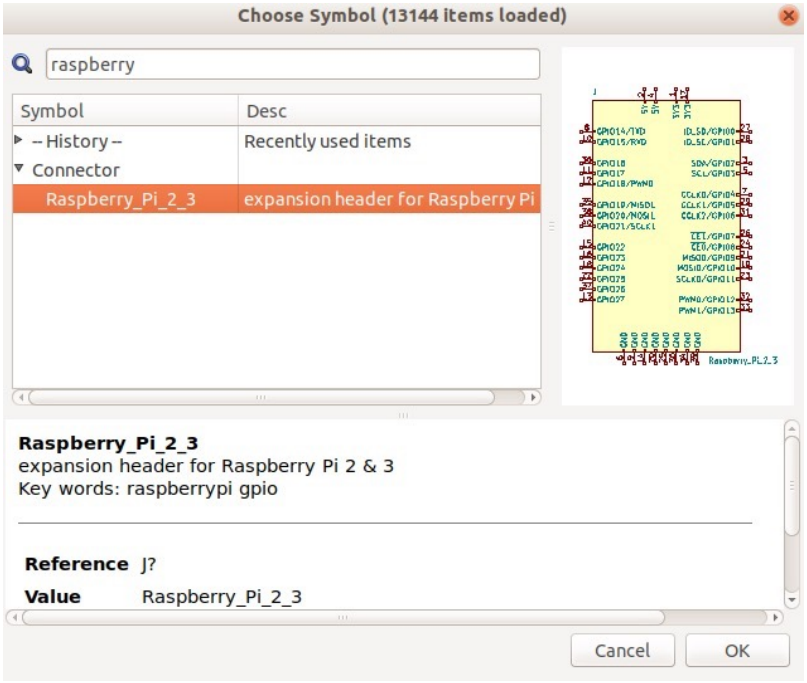


Figure 18.4: The Raspberry Pi symbol.

Go ahead and add the symbols on the sheet. For the resistors, remember to also set their values via the Symbol Properties window (hover your mouse pointer over the symbol, and type "E"). In Figure 18.5, you can see the Symbol Properties for R1, where I have set the Value property to 10 KΩ (I am not worried that I am using a non-ASCII character here because I do not plan to use the FreeRouting autorouter).

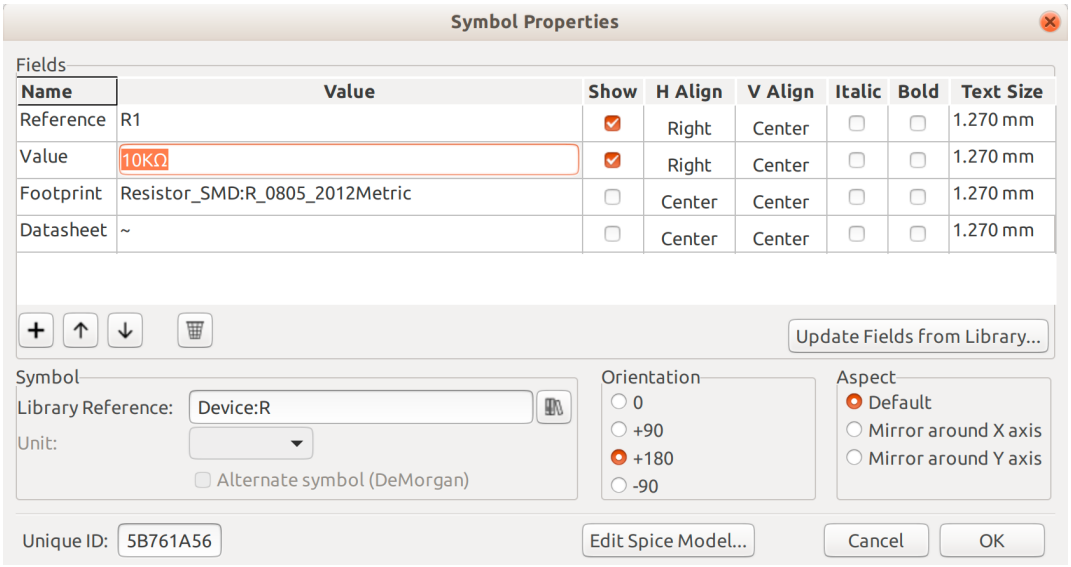


Figure 18.5: Set the value for R1.

Remember to set the Value property for the R2 (10 kΩ), R3 (330 Ω) and R4 (330 Ω) symbols.

Your sheet should look like the example in Figure 18.6.

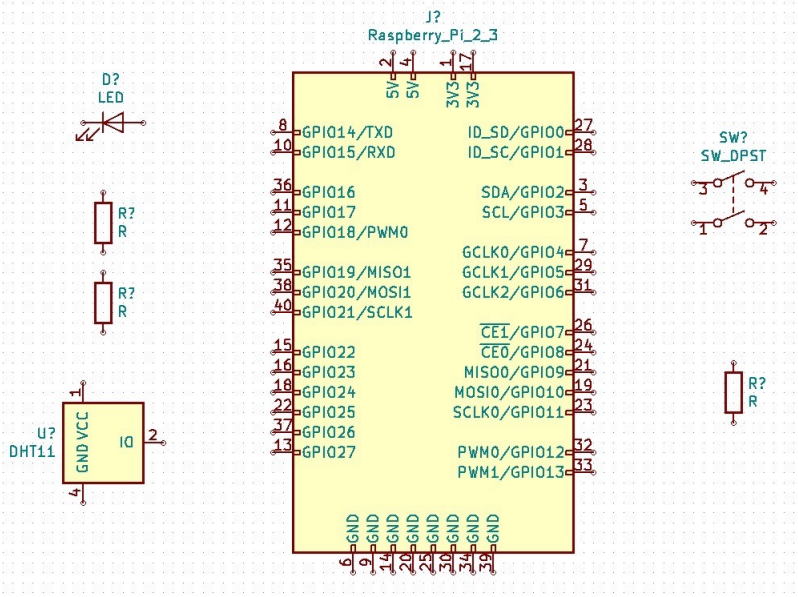


Figure 18.6: The symbols added to the Eeschema sheet.

Even though I plan to use the DHT22, the available symbol in the libraries is for the DHT11. Because the two sensors are pin-compatible, you can still use the DHT11 symbol, and rename it to DHT22. Do that by placing your mouse pointer over the DHT11 symbol, then type 'E'. In the Symbol Properties window, change the 'DHT11' value to 'DHT22' (Figure 18.7).

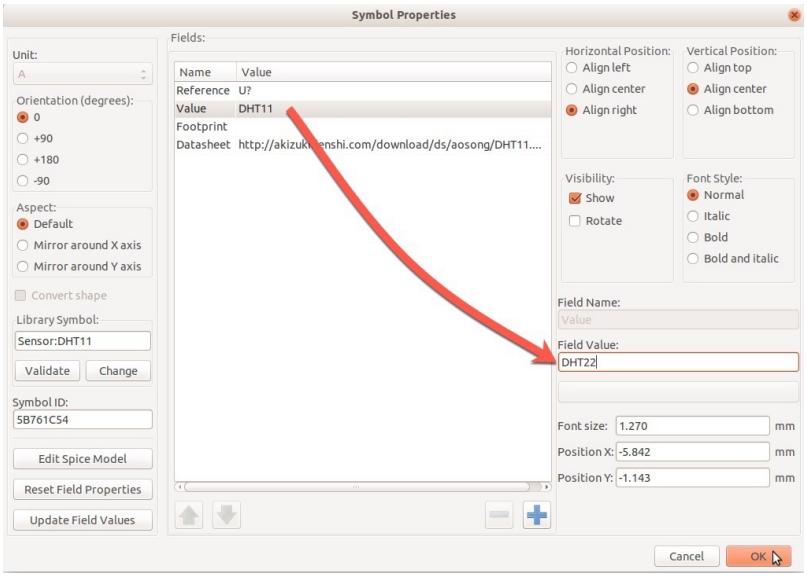


Figure 18.7: Change the field value of the sensor symbol.

Your schematic is now consistent with the reality of what you are designing. Let's continue with Step 3.

Step 3: Arrange, Annotate, Associate

In Step 3, you will work on the three 'A's: Arrange, Annotate and Associate.

Arrange

Unlike Project 1, in this project we don't have a natural flow of electricity from one end of the schematic to the other. Instead, we have a central component, the Raspberry Pi, to which we connect the peripherals. With this in mind, place the footprints on the side of the Raspberry Pi where the pins to which they will connect, are. The DHT22 sensor will connect to 3.3V, GND and GPIO17 (physical pin 11), so place it, and its pull-up resistor, on the left side of the Raspberry Pi symbol. The LED will be connected to GPIO4 (physical pin 7), so place it and its current limiting resistor to the right side of the Raspberry Pi. The button will be connected to GPIO14 (physical pin 8), so it, and its pulldown resistor should also go to the left of the Raspberry Pi (Figure 18.8).

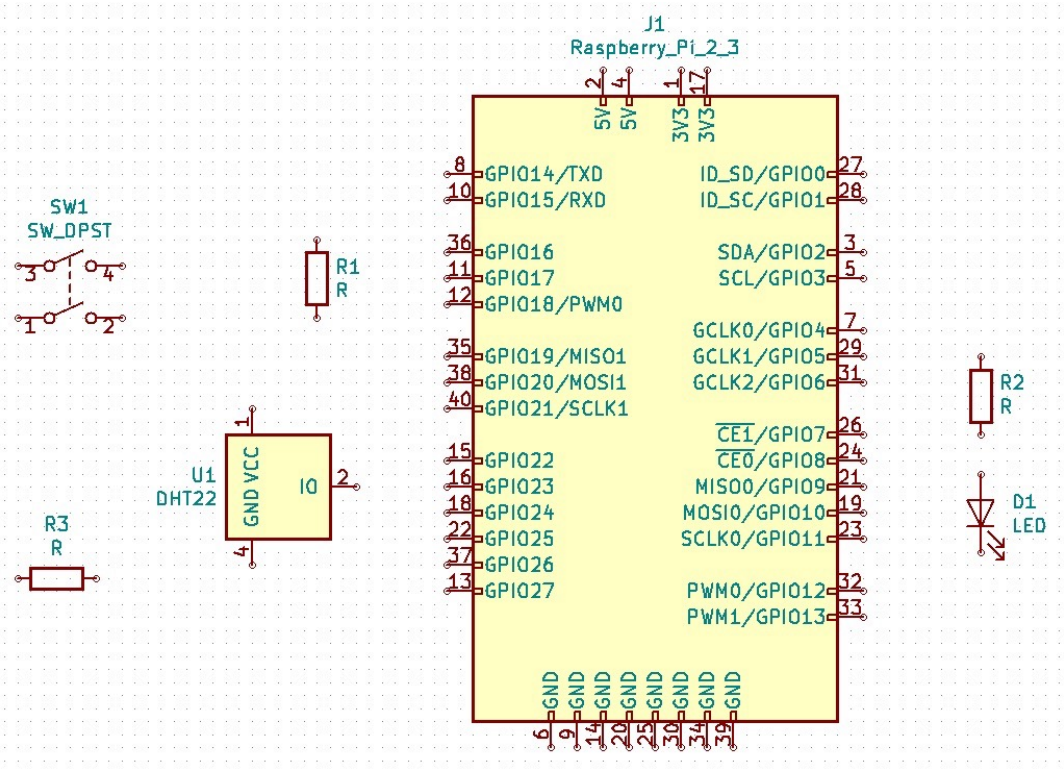


Figure 18.8: Final placement of the symbols.

Annotate

Next, annotate the symbols using the 'Annotate Schematic Symbols' tool. With the default settings in the annotator window, the result is shown in Figure 18.9.

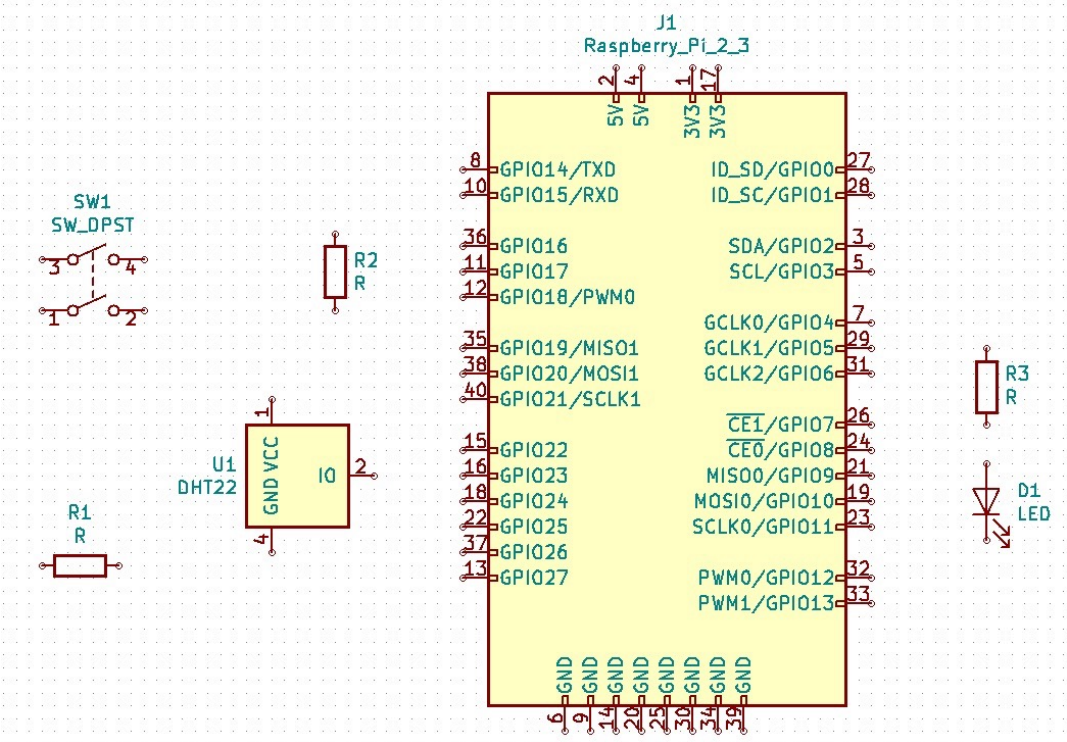


Figure 18.9: Annotated symbols.

Associate

This is the more interesting part of Step three. There's a few things to keep in mind:

1. For the resistors and the LED, instead of using TH (Through Hole) footprints, we will use SMD (Surface-Mounted Device). Because we want to attach the board components on the top side of the board, in the Pcbnew layout we will be routing the pads of those footprints exclusively on the top copper layer. It is certainly possible to attach components on the back of the board, but this is rarely done. Automated pick and place machines can only place components on one side of the board. If we need to place components on the other side, the operator will need to flip the panels over. In the footprint library, take care to choose footprints that have the '0805' package type. This package has a size of 2 mm x 1.2 mm.
2. For the Raspberry Pi, we don't want to use a full-sized HAT. As long as we choose a footprint with enough pins to accomodate for the first 16 pins of the Raspberry Pi (2x8) that we need for the circuit (even though we are only using 7 of those), and the pin numbers between the Raspberry Pi symbol and our footprint of choice match, we are free to pick and choose.

Let's start with assigning footprints to the resistors. If you want to do this individually for each symbol, open their properties window, click on the Footprint field, and click on the Browse Footprint button to bring up the footprint browser. I prefer to use Cvpcb and save some time. Open Cvpcb, and choose the Resistor_SMD library from the left pane, and the 0805_2012_Metric symbol from the right. Double click to associate this footprint with the three resistor symbols.

For the LED, choose the LED_SMD library, and the 0805_2012Metric footprint.

For the DPST switch, I have found a matching footprint in the Freetronics library. If you haven't installed it, download it from the Freetronics Github repository, and install it. Information on how to do this is available in the recipe titled '21. Adding a footprint library in Pcbnew'. I have chosen the SW_PUSHBUTTON_PTH because I have a lot of those pushbuttons in my

drawers. You can choose an SMD version of the pushbutton if you want to reduce the size of your board even further.

For the DHT22 sensor, you need a pin header with four pins. The actual DHT22 device is manufactured with four physical pins, although only three are electrically operational. Notice that the schematic symbol shows pins 1, 2 and 4. Those pins will match the footprint pads with the same numbers, leaving footprint pad 3 unconnected. From the library named 'Connector_PinHeader_2.54mm', select the footprint with name that ends with '1x04_P2.54mm_Vertical'.

Finally, for the Raspberry Pi footprint, choose the footprint with title that ends with '2x08_P2.54mm_Vertical', from the same library.

The associations now look like those in Figure 18.10.

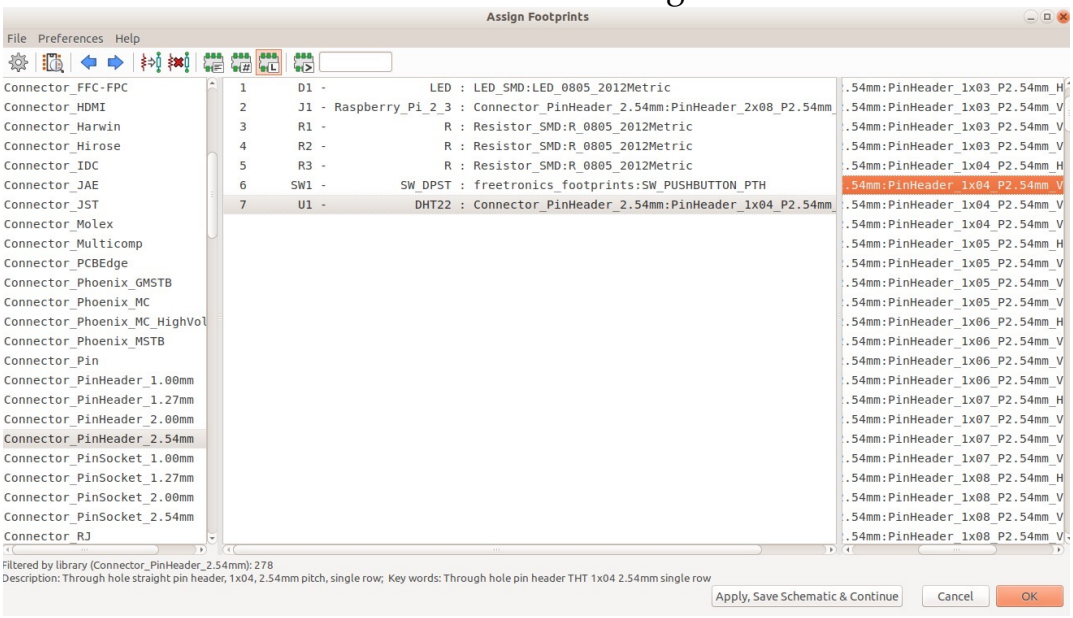


Figure 18.10: The final associations table.

Please note: In Figure 18.10, I have selected the PinHeader Connector "Connector_PinHeader_2.54mm". This footprint is electrically correct, and, as you will see in the layout pages, it works. However, the 3D model of the pinheaded model represents two twos of pins. Remember that the Raspberry Pi header consists of two rows of pins. By using a pinheaded as the footprint for the HAT, the HAT's 3D representation will have pins in place of its header, which don't "fit" with the header of the Raspberry Pi.

In Figure 18.11 you can see what the final PCB will look like with a pin header.

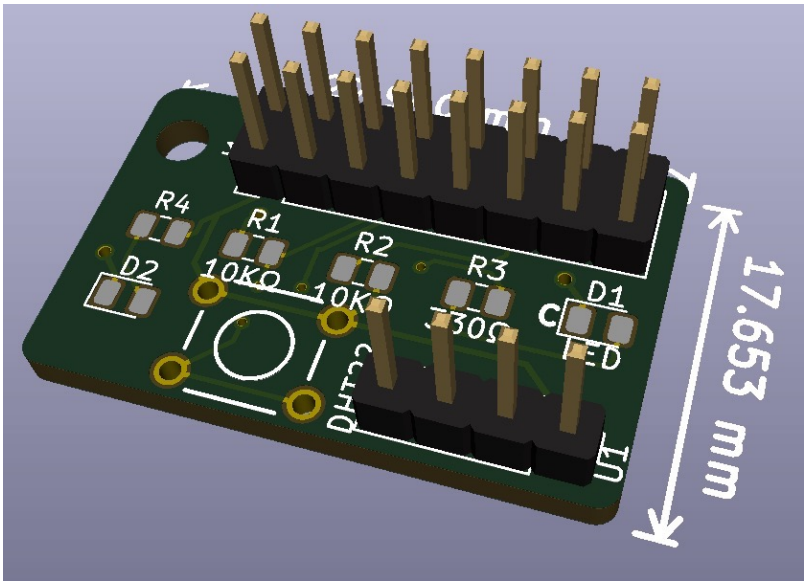


Figure 18.11: The completed PCB with a pin header

In Figure 18.12:, you can see what the final PCB will look like with a socket header.

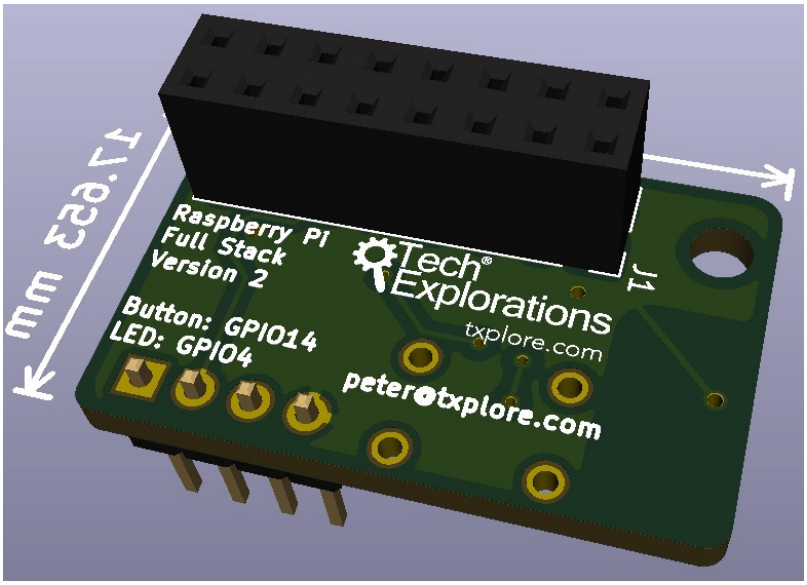


Figure 18.12: The completed PCB with a socket header

Conclusion: If you want the 3D representation of your HAT's header to "match" with the header of the real Raspberry Pi, then consider using a PinSocket_2x08_P2.54mm_Vertical footprint instead of the Connector_PinHeader_2.54mm I use in Figure 18.10.

Click on the 'Apply, Save Schematic & Continue' button to commit this work, and then click on OK to dismiss the window.

Let's jump to Step 4, wiring.

Step 4: Wiring

Step 4 is straightforward. In the Python scripts for Raspberry Pi Full Stack, I have configured the LED to be controlled by physical pin 7, the button to be read by physical pin 8, and the data from the sensor to be read by physical pin 11. Apart from those pins, the rest are the pins for the 3.3V supply (physical pin 1) and the GND (there are multiple of those, but I'll use physical pins 6 and 9).

Go ahead to do the wiring. The result is in Figure 18.13.

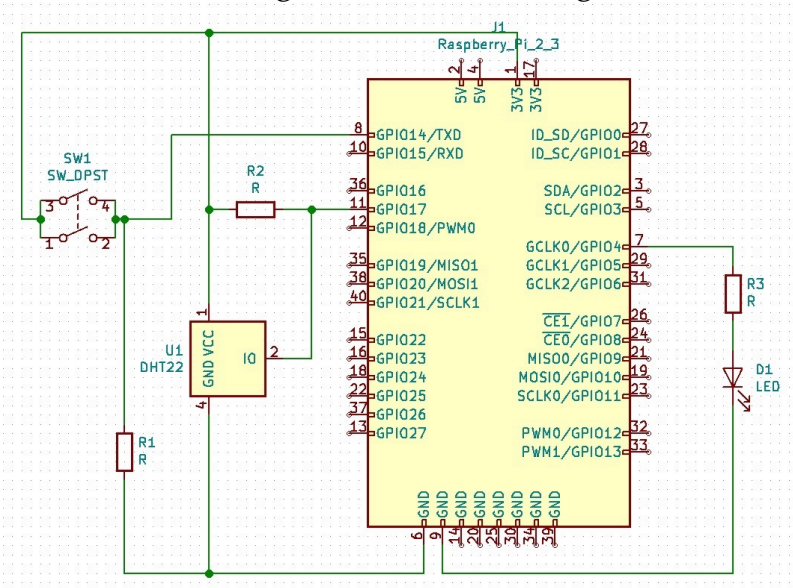


Figure 18.13: Wiring complete, but not elegant.

While the wiring, as shown in Figure 18.13, is complete and correct, it isn't elegant. It has wires crossing over other wires and too many junctions. We can do better than this, and produce a schematic diagram that is both correct, and good to look at. Consider these issues when you work on larger circuits. In Step 5, Nets, you will learn how to reduce the clutter by naming your nets and removing wires.

Before moving on, run an ERC. The results will contain multiple issues with the schematic. Specifically, it will complain about passive pins that are not driven. This has to do with the unused pins on the Raspberry Pi symbol. To resolve those issues, place the 'unconnected' symbol on each unused pin on the Raspberry Pi symbol. You will also need to use the PWR_FLAG symbol in any wire that conveys power. Those are the wires connected to the 3V3 and GND pins. You can see the updated schematic in Figure 18.14.

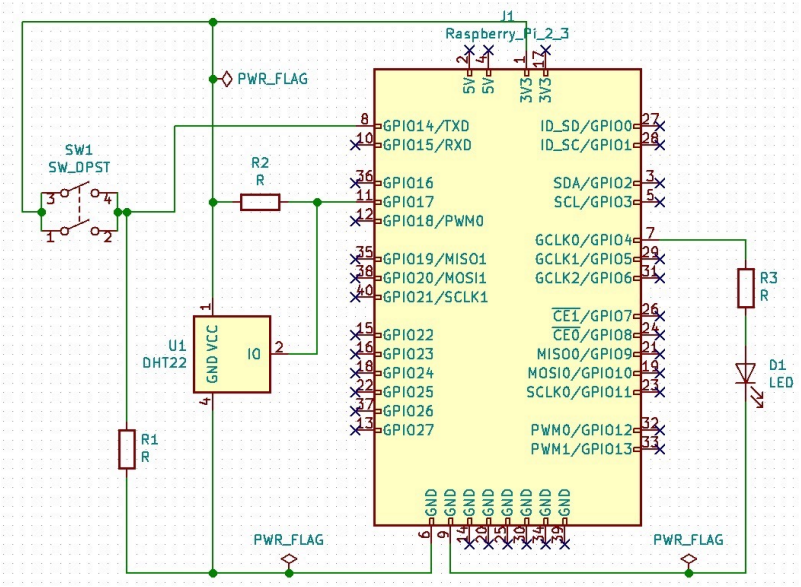


Figure 18.14: The updated schematic with PWR_FLAG and Unconnected symbols.

Time to continue with Step 5.

Step 5: Nets

You already know how to name nets from your work in Project 1, but in this project, you will also learn how to use named nets in place of wires. In Step 4, you wired the circuit, but the result is rather messy. What you will do now is to remove the wires and rely on net names and symbols for KiCad to about the electrical connections between pins.

Let's work on the first one. Remove all the wires that go to the GND pins and connect them to the GND symbol. You can see an example of this in Figure 18.15.

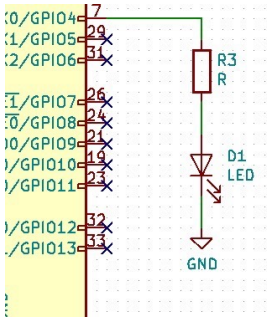


Figure 18.15: Using the GND symbol in place of a wire to the GND pin.

Another simplification you can do on the schematic is to replace the signal wire that connects the button to pin 8 on the Raspberry Pi symbol with net labels. Delete the wire, and place a net label on the Raspberry Pi pin 8 and the output of the button. I am using the name 'button_input' for this label (Figure 18.16).

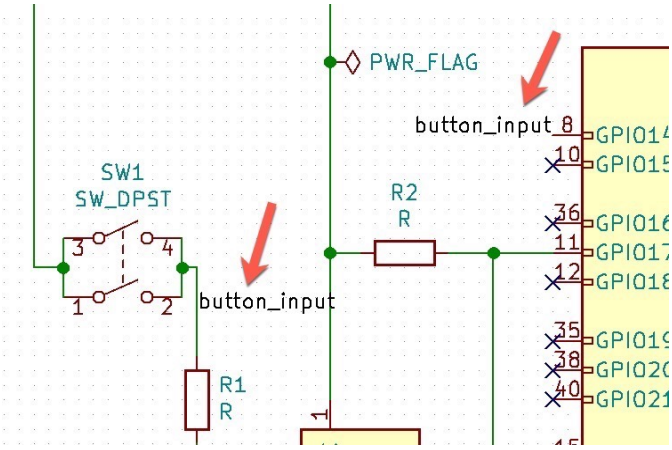


Figure 18.16: A wire replaced with net labels.

One more modification left to do is to replace the 3.3V wires with labels. You can see the result in Figure 18.17.

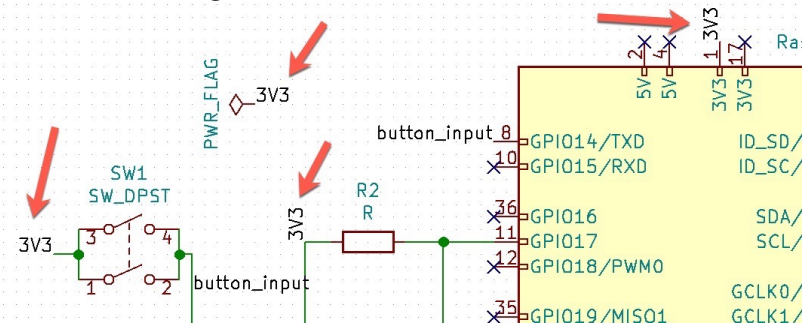


Figure 18.17:The 3.3V wires are replaced with net labels.

Before finishing this step, let's name the remaining of the wires. I used names like 'button_input', 'led', 'sensor_data' and 'GND'. The final schematic is in Figure 18.18.

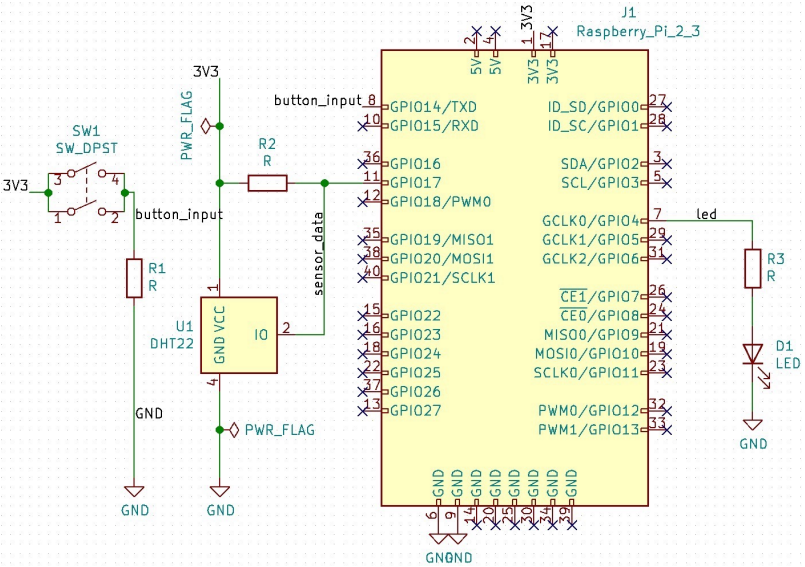


Figure 18.18: Final wires are labeled.

The net naming step is complete, let's move on to do the Electrical Rules Check.

Step 6: Electrical Rules Check

Although you have been doing electrical rules check during the wiring process, let's do one more to detect any last-minute issues.

Got the all-clear (an empty Error list)? Let's move on to Step 7.

Step 7: Comments

Let's add some informative comments. At the very least, I include component values, like that of the resistors. I used boxes to mark-up functional blocks and named them. The final schematic is in Figure 18.19.

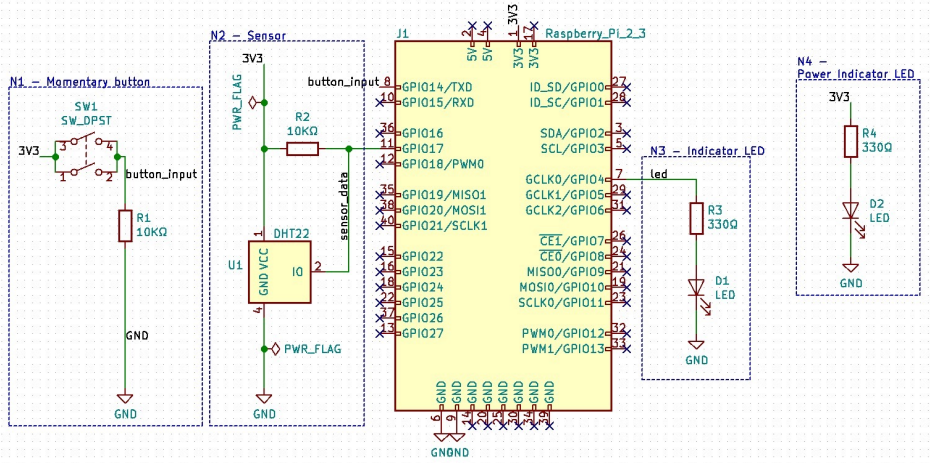


Figure 18.19: The final schematic.

Let's get into the next and final step where you will export the netlist file.

Step 8: Netlist

You have finished with the schematic, and the ERC is clear of errors. All that is left to do is to produce the Netlist file so you can continue with the layout process. Click on the Netlist button, accept the default settings in the Pcbnew tab, and click on 'Generate Netlist' to produce the file. Accept the default netlist name, for consistency. Close the windows.

You should have these 6 files in the project folder (Figure 18.20).

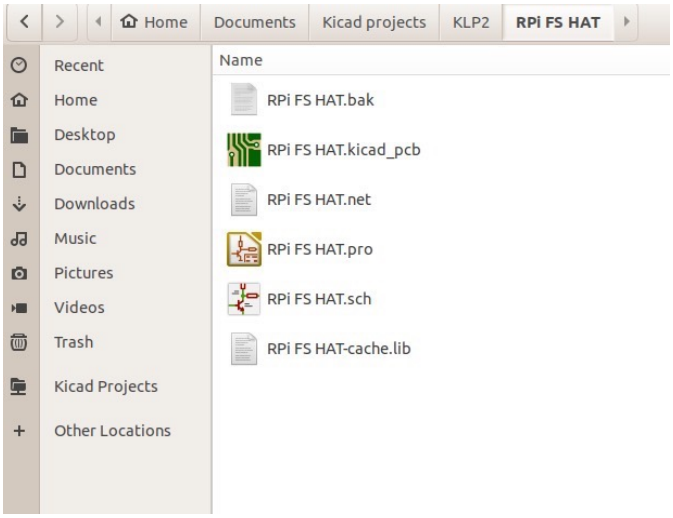


Figure 18.20: Checking the project files.

Let's continue with the layout process, where you will import the .net file into Pcbnew.

18.5. Footprint layout in Pcbnew

Let's start the layout process. To do this, you will follow the process illustrated in Figure 18.1.

Step 1: Setup

For this project, I will use the same grid, layer, and design rules settings as those in the first project. I'll be using the same manufacturer, so I do not need to make any changes to the minimum values for things such as the drill size and the track width. I will only customise the text of the page settings window since this project does have its own name (Figure 18.21).

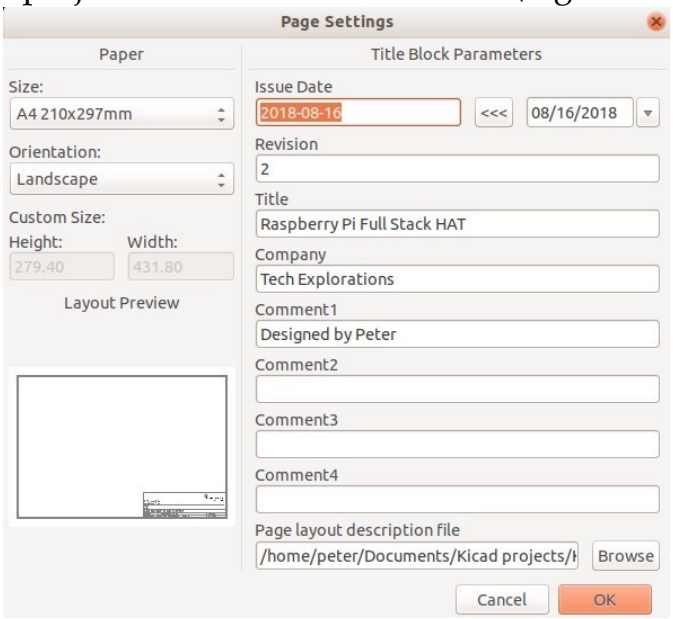


Figure 18.21: Project 2 Page Settings.

Before you set up the design rules, import your netlist file. This will allow you to allocate the nets to the appropriate net class. My setup is showing in Figure 18.22.

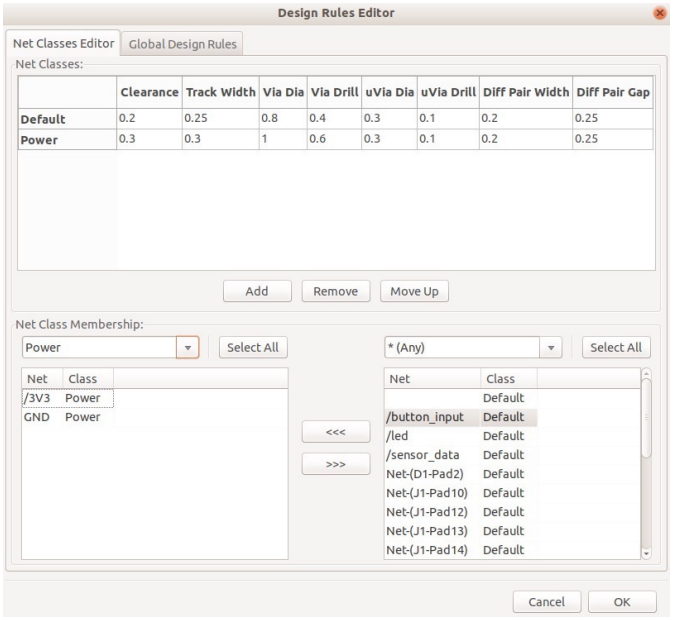


Figure 18.22: Design rules and net classes.

Continue with the outline and the mechanical constraints.

Step 2: Outline and constraints

The mechanical constraints that will influence the outline of this board depends primarily on the pin header, and the mounting hole. These two factors are constrained by the mechanical design of the Raspberry Pi. On top of that, we have to allow for enough room for the components, although this will be much easier to do compared to the first two.

After importing the netlist, the 8x2 pin header is already on the Pcbnew sheet, and it has the correct geometry. We can draw the outline of the board around this footprint and then continue with the mounting hole. We will have to make a precise measurement on a real Raspberry Pi to determine the exact position of the hole in relation to the pin header. My calliper says that the centre of the hole is 5.30 mm away from the centre of pin 1, and has a diameter of 2.95 mm.

Move the header away from the footprint bundle. You should also rotate it so that the footprint is in a horizontal position, with pin 1 (with the rectangular pad shape) placed towards the inside of the board. This is the side of pin 1 on the actual Raspberry Pi. Then, create the mounting hole. Proceed as follows.

1. Select the Edge.Cuts layer.
2. Click on the circle button from the right toolbar.
3. Place your mouse in the middle between pads 1 and 2.
4. Press the space bar to reset counters *dx* and *dy*.
5. Move the mouse towards the left until the *dx* shows a value as close as possible to 5.30.
6. Click to draw the circle.
7. Keep an eye on the left side of the status bar where the circle radius is displayed; draw the circle to a radius of approximately 1.47 mm (half the diameter I measured earlier).

With my grid setting of 0.127 mm, I was able to position the centre of the hole at 5.33 mm away from the centre of pins 1 and 2 and to create a hole with a radius of 1.397 mm. If you use the 3D viewer now, the circle you just created will appear solid instead of hollow. That is because there is not a closed outline around it. Let's create the actual board outline next. You can see the result of this work in Figure 18.23.

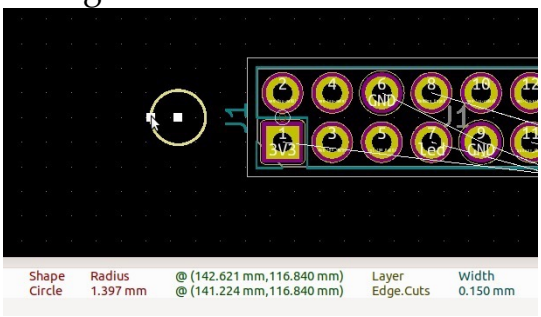


Figure 18.23: The mounting hole and pin header in position.

For this outline, I would like to draw rounded corners instead of 90-degree ones. To learn how to do this, please read the recipe titled '33. Rounded corners'. As my constraints are the header, the mounting hole, and the minimum space need to place the rest of the footprints within the outline, I will go ahead to draw the outline. I am open to the possibility that after I add the components inside the outline, I may need to redraw parts of the outline to improve the design. You can see the final outline in Figure 18.24.

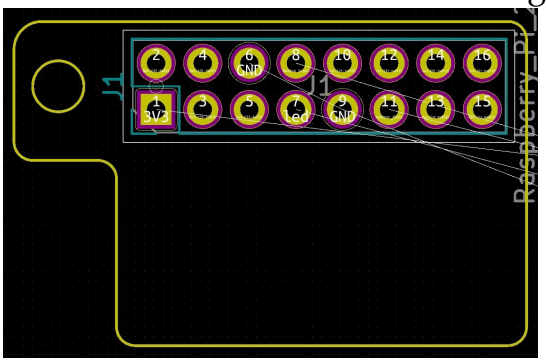


Figure 18.24: This outline looks ok for now.

With this outline, let's move to Step 3 and place the components within it. If needed, we can make improvements to it.

Step 3: Place components

The board only has a button and LED as a user interface. I placed the button first in one of the corners of the board where it will be easily accessible. Then, I placed the sensor header along the bottom edge and took care to orient pin 1 towards the corner so the when the sensor is installed, its grill is facing outwards. This will keep the space behind it free to place the LED and resistors (Figure 18.25).

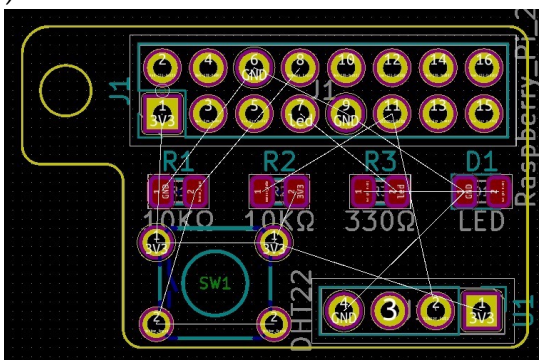


Figure 18.25: The board with the footprints positioned.

I selected the SMD components and used the context menu to distribute them horizontally.

My first drawing of the outline seems to be appropriate for accommodating the footprints. I will leave it as it is. If you wish to make changes, you can simply delete a segment of the outline and redraw it as you wish.

Step 4: Route

This is a small circuit, perfect for practicing your manual routing skills. Remember that you can use the FreeRouting autorouter if you wish. Learn how to do this in the recipe 'Using an autorouter'.

To route this board, start with any tracks that convey power since there are no tracks that we can legitimately call 'critical'. The power tracks are those with net names '3V3' and 'GND'. Then continue with the rest.

After this work, the board looks like the example in Figure 18.26.

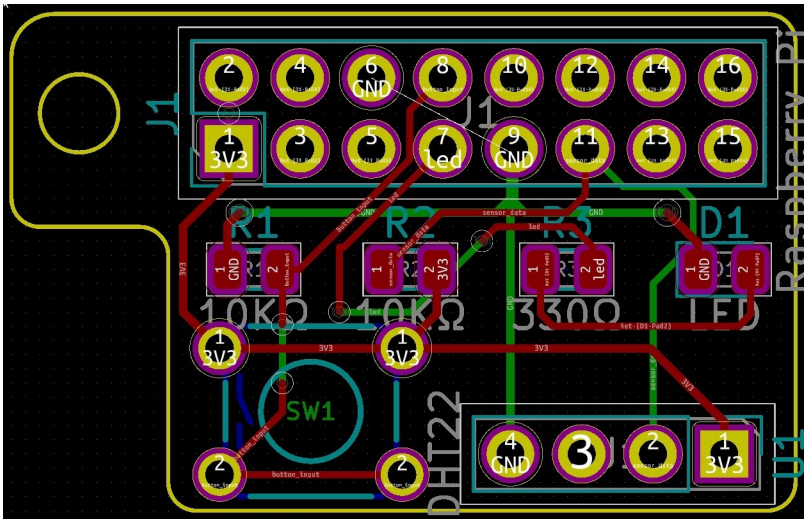


Figure 18.26: Routing complete, although one ratnest line remains.

The use of SMD components in place of TH means that we need fewer vias, and we have more of the board available for routing since board real estate is not lost.

The board in Figure 18.26 contains one remaining ratnest line (the one that connects pins 6 and 9). I left that on purpose. In the next step, you will create the GND copper filled zone. Based on the project's design rules, the copper fill will be connected to any GND pads through reliefs when we create the copper fills. By leaving this last ratnest line as it is, we don't need to create a new trace.

One last thing to do before continuing with Step 5, is to run the design rules check. When I ran mine, two errors were reported (Figure 18.27).

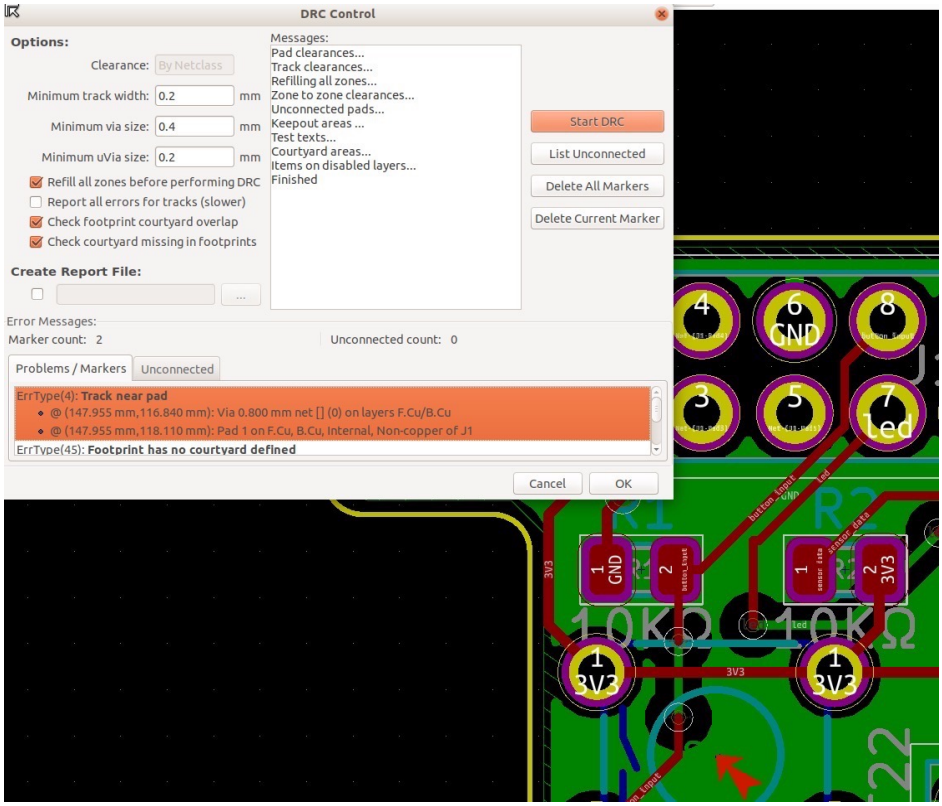


Figure 18.27: The 'Track near pad' error must be fixed.

One error had to do with the proximity of a via to a pad. This is marked with a red arrow. This is a violation of the design rules, so it must be fixed before continuing. I fixed the error by deleting the trace segments that connect to the via, moved the via downwards and away from pin 1 of the button, and then restored the traces.

The other error was about a missing courtyard for the button footprint. I can ignore this one because it does not affect the functionality of the board.

Let's move on to Step 5.

Step 5: Copper fills

In this step, you will create a copper filled zone in the back copper layer, and connect it to the GND net. Follow the process described in the recipe '25. Creating copper fills'. The result is shown in Figure 18.28.

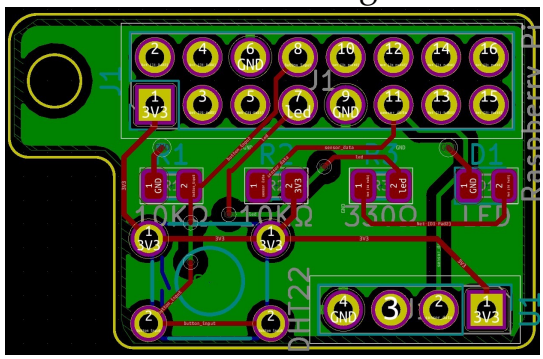


Figure 18.28: The GND copper filled zone, in green.

As I suspected, the filled zone is connected to the GND pins 6 and 9 of the 2x8 pin header. This removed the ratnest line that connected these pins earlier.

Run the design rules check one more time to ensure that there are no violations. Apart from the courtyard error for the button footprint, the list should be clear.

Let's work on the silkscreen next.

Step 6: Silkscreen

In the front and back silkscreen layer, I would like to:

1. Provide information about the values of the resistors.
2. Clearly mark the anode and cathode of the LED.
3. MarK the sensor pin names.
4. Mark the Raspberry Pi pin to which the button is connected to.
5. Mark the Raspberry Pi pin to which the LED is connected to.
6. Add the board name and version number.
7. Add contact information.
8. Add a logo and other graphics.

All this information will make assembly easier and will assist the user when she is working on her Python script without having to look at the board schematics and probing with a multimeter. To figure out which GPIO to use in their script to control the LED, they only need to look at the board.

Not all of this information will fit in the front of the board, so we will use the back silk screen.

Let's begin with the resistor values. We provided those values in the schematic, and they were carried over in the layout, but are available in the F.Fab layer. This is not visible in the final board, so you will need to change the layer of this text to F.SilkS. You can do this through the Footprint Text Properties window. Place your mouse pointer on the text and type 'E' to show the properties window. Then, change its layer to 'F.SilkS (Figure 18.29).

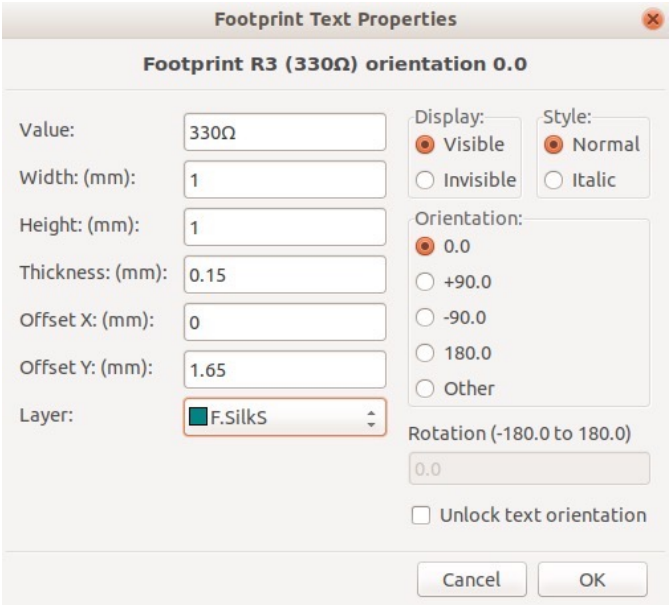


Figure 18.29: Changing the layer of the footprint text properties.

Do the same for the rest of the text labels of the SMD components (Figure 18.30).

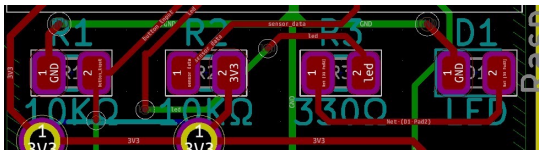


Figure 18.30: The SMD component text is in the F.SilkS layer.

These are the other changes or additions I made in relation to text labels:

1. I placed a 'C' character (for 'cathode') next to the cathode pad of the LED.
2. I changed the layer of the DHT22 text label to 'F.SilkS'.
3. I added labels 'GND', 'DATA', and '3V3' to the sensor pins.

The front silkscreen is complete (Figure 18.31).

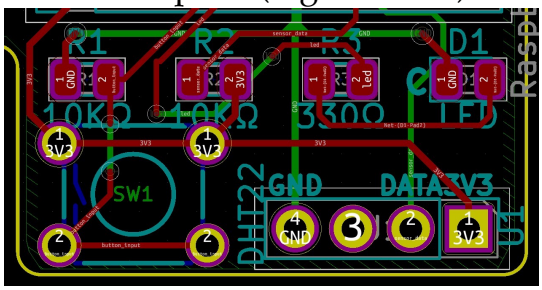


Figure 18.31: The F.SilkS text labels.

In the back silkscreen, I will place the rest of the elements. Those are items 4 to 8 from the list at the start of this chapter.

Click on the B.SilkS layer from the layers manager, and add:

1. The text 'Button: GPIO14, LED: GPIO4' (Figure 18.32)

2. The text 'Raspberry Pi Full Stack Version 2'
3. The text 'peter@txplore.com'
4. The Tech Explorations logo.

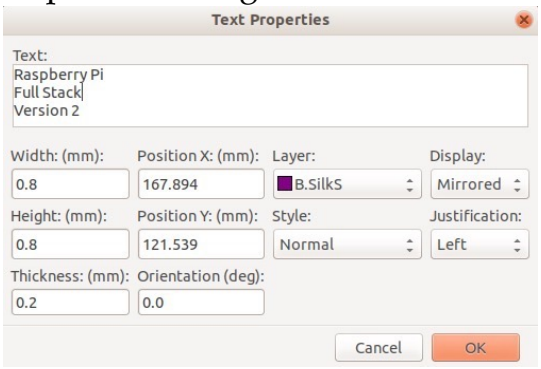


Figure 18.32: Reduce the size of the text to make it fit in small areas.

For the text, I reduced the size of each character to 0.8 mm width and height to make it fit in the small amount of space available.

You can see the end result in Figure 18.33.

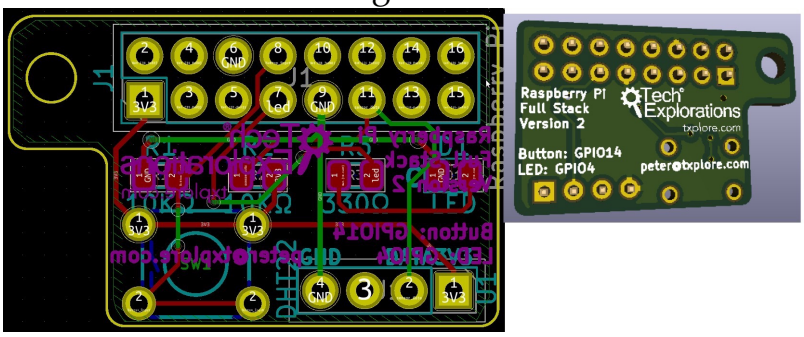


Figure 18.33: The final board.

Let's run a final design rules check, in the next step of the process.

Step 7: Design Rules Check

The DRC reports only two courtyard related issues, that you can safely ignore since they don't affect the operation of the board (Figure 18.34).

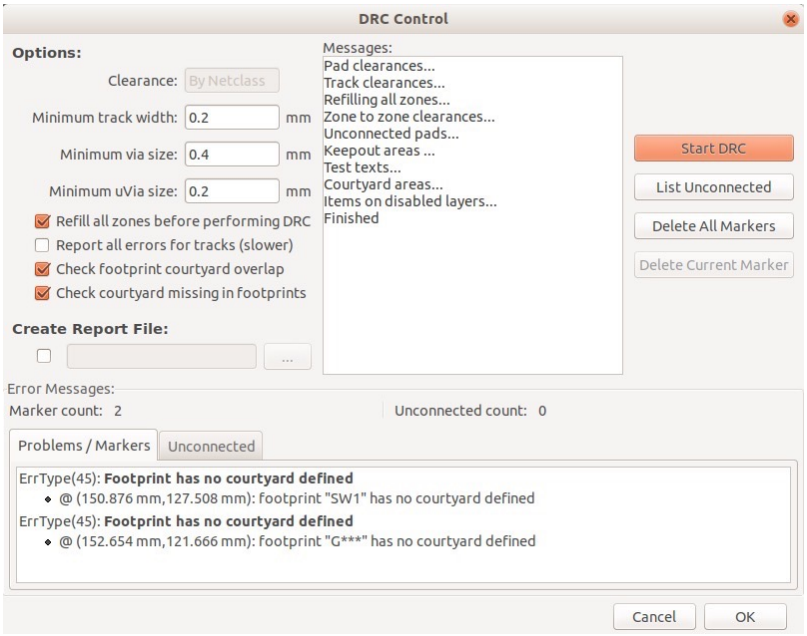


Figure 18.34: Courtyard issues don't affect the operation of the board.

The design work for this board is complete, so lets continue with the manufacturing step.

Step 8: Manufacture

In this final step, you will upload your board to your manufacturer's website. I will use PCBway, as usual, so I will need to:

1. Export the Gerber files.
2. Create a Zip archive that contains the Gerber files.
3. Test the archive is valid by uploading it to gerblook.org.
4. If the archive is valid, follow the ordering process on the manufacturer's website.

I exported the Gerber files as I did for project 1. The validation did not reveal any issues.

In Figure 18.35 you can see how I used the "Add Dimension" tool from the right tool bar to measure the horizontal and vertical size of the board. I manually copied the result (27.940 mm and 17.653 mm) to the manufacturer's order form so I can receive an accurate quotation. The dimensions that appear in Figure 18.35 are embedded in the Gerber files (in the bottom silk screen layer, though they will not be printed because they are outside the edge cuts), but are not parsed at by the manufacturer.

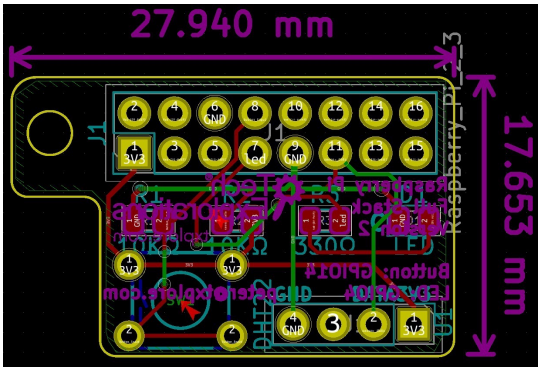


Figure 18.35: The board dimensions.

You can see my manufacturing choices in Figure 18.36.

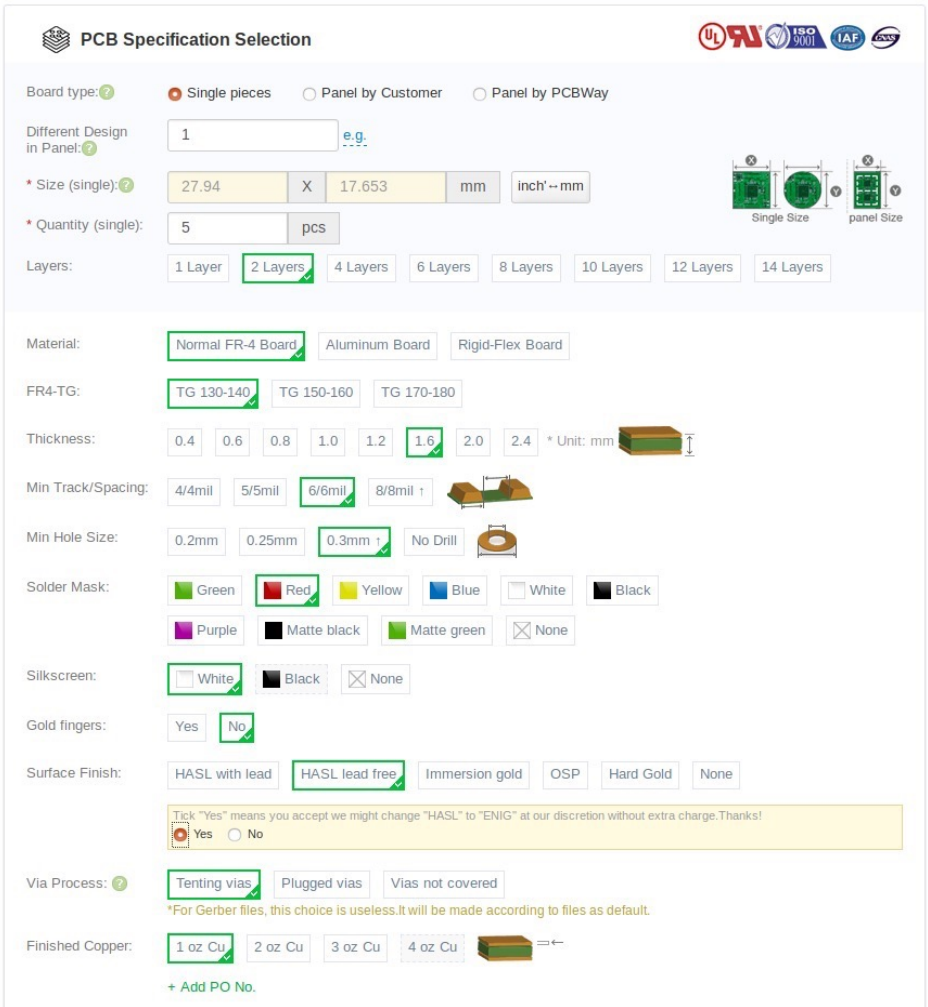


Figure 18.36: Pcbway.com provides multiple options; I particularly like the choices of solder mask and silkscreen colours.

And, finally, the order details:

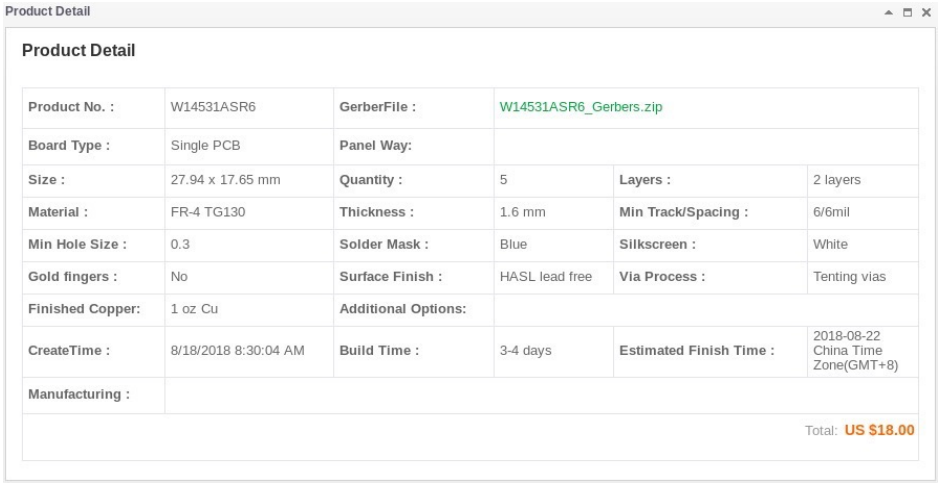


Figure 18.37: The order details.

19. Project 3: Arduino clone with build-in 512K EEPROM and clock

19.1. Project details

In this project, you will build on the knowledge and skill of the previous two projects and design an Arduino-compatible board using, mostly, surface-mounted components. The board will contain a microcontroller, two EEPROM modules, a clock and calendar module, two crystal oscillators, and several SMD-sized resistors, capacitors, and LEDs. In addition, it will include headers for connecting peripherals to the microcontroller.

All this, in a relatively small board, measuring 44x28 mm. Working on this board will give you the opportunity to explore some of KiCad's powerful features, such as splitting schematic diagrams into more than one sheet, using an autorouter, and converting a two-layer board into a four-layer board.

The name I have given to this board is 'Battery powered Arduino with clock and extended EEPROM'. Or, let's call it 'BACEE', for short.

Here is the list of components:

Component	Description	**Part**
1	Battery (multiple cells)	Battery
2	Polarised capacitor	CP1
3	Unpolarized capacitor	C
4	LED generic	LED
5	Generic connector, single row, 01x09, script generated (KiCad-library-utils/schlib/autogen/connector/)	Conn_01x09_Male
6	Generic connector, single row, 01x04, script generated (KiCad-library-utils/schlib/autogen/connector/)	Conn_01x04_Male

7	Generic connector, double row, 02x03, odd/even pin numbering scheme (row 1 odd numbers, row 2 even numbers), script generated (KiCad-library-utils/schlib/autogen/connector/)	Conn_02x03_Odd_Even
8	Generic connector, single row, 01x04, script generated (KiCad-library-utils/schlib/autogen/connector/)	Conn_01x04_Male
9	Resistor, for LED	R (330 Ω)
10	Resistor, pull-up for RESET	R (10 kΩ)
11	I2C Serial EEPROM, 1024Kb, DIP-8/SOIC-8/TSSOP-8/DFN-8	24LC1025
12	IC MCU 8BIT 32KB FLASH 32TQFP	ATMEGA328P-AU
13	Clock/timer IC	DS1337S+
14	Two pin crystal	Crystal
15	Two pin crystal	Crystal
16	Resistor, pull-up, for I2C	R (10 kΩ)

I designed this board because I wanted an Arduino-Uno compatible board, small in size, that I can power using alkaline batteries. I'd like this board to be able to keep the time, to have sufficient build-in memory for data logging, and to be able to connect a variety of sensors using standard headers. I also wanted a design that I can use as a basis for other similar projects. For example, I can design a variation of the base BACEE board with an integrated environment sensor, or wireless communications.

In Figure 19.1, you can see the four-layer 3D rendering of the board. The Atmega328P-PU MCU is in the centre of the board. The headers and the indicator LED are placed along the edges. Using SMD components helps to keep the board size to a minimum. I selected SMD packages that are large enough to be hand soldered.

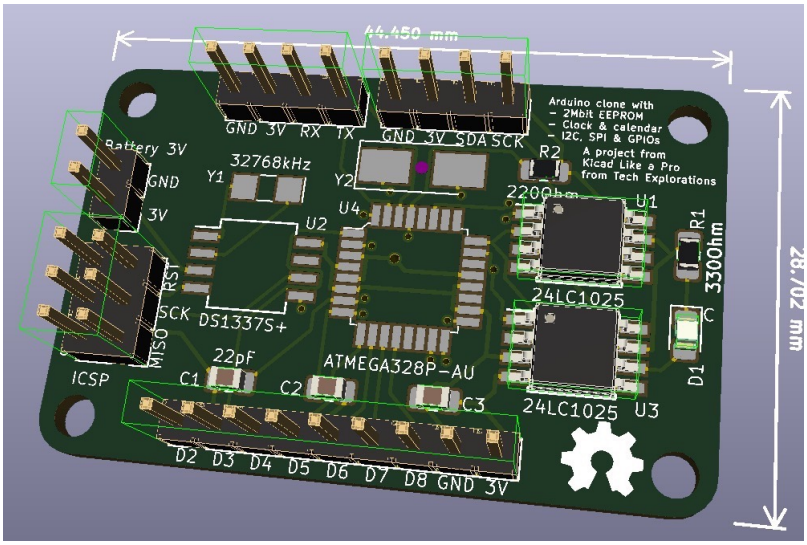


Figure 19.1: The outcome of this project.

For the most part, we will follow the schematic and layout design processes depicted in Figure 19.2 and Figure 19.3.

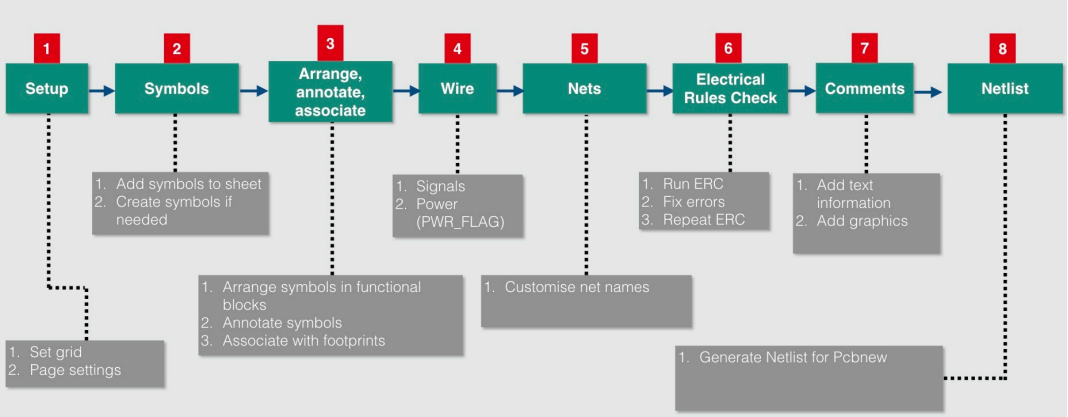


Figure 19.2: The schematic design process.

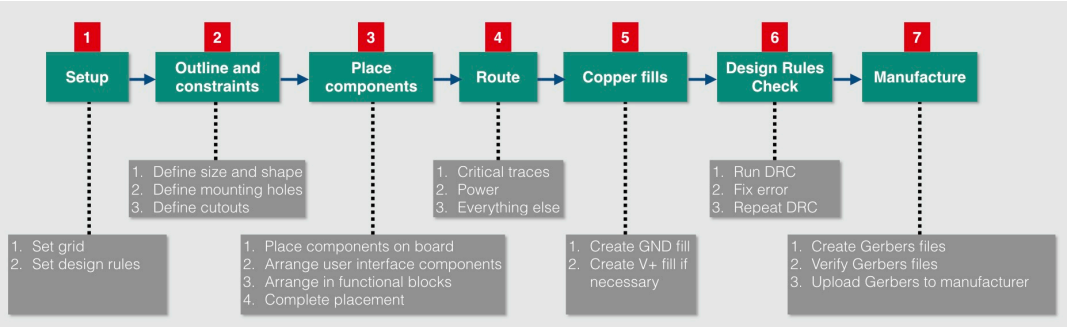


Figure 19.3: A serialised view of the PCB layout process.

However, as you are now more skilled in this type of work, you will feel free-er to iterate through the steps. For example, during the layout work, you may decide to redraw the outline of the board after you complete the footprint placement in order to accommodate a header. You can do that easily by adjusting your drawings in the Edge.Cuts layer.

Let's begin.

19.2. Project repository

This project has a Git repository. You can access it at txplo.re/klpp3.

This repository contains multiple commits at each step of the design process, as I was going through it. Feel free to clone this repository on your computer and make any modifications you like.

19.3. Schematic design in Eeschema

Start a new KiCad project, and store the project files in a new folder. I called mine 'BACEE'.

Step 1: Setup

Start by creating a new Eeschema file. Setup the page use the Page Settings window (accessible via the File menu). You can see my setup in Figure 19.4.

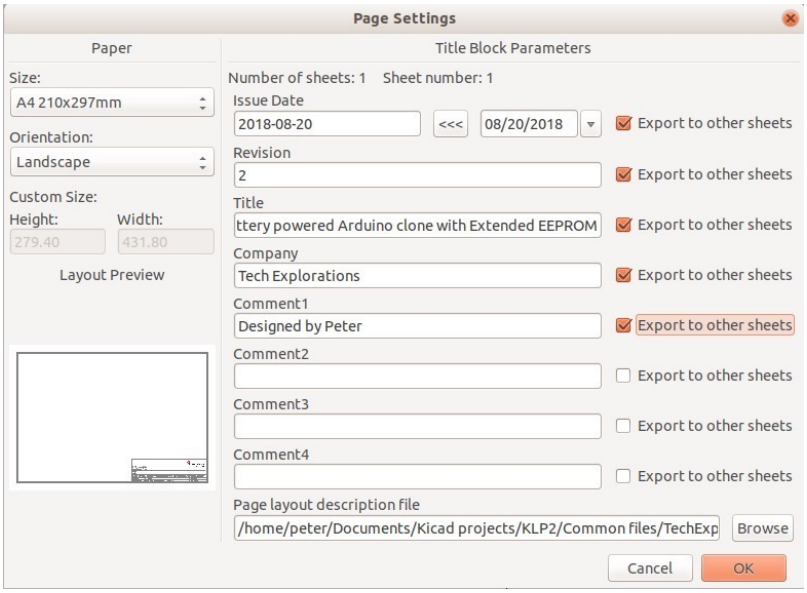


Figure 19.4: The page settings.

Notice that this time, I have checked the 'Export to other sheets' checkboxes. This is because I will be using hierarchical sheets to spread-out the schematic design. By checking the export boxes, the information of the page settings will be copied across the sheets.

Continue to Step 2.

Step 2: Symbols

Most of the symbols you will need for this schematic are available in the build-in symbol libraries, or in third-party libraries that you can install. Here is a list of the symbols, including the libraries in which you can find them:

1. The Atmega328p-AU symbol, in the Digikey library: dk_Embedded-Microcontrollers:ATMEGA328P-AU
2. The 24LC1025 EEPROM module x 2: Memory_EEPROM:24LC1025
3. Crystal oscillators x 2: Device:Crystal
4. Non-polarised capacitors x2: Device:C
5. Polarised capacitor: Device:CP1
6. Resistors x 2: Device:R
7. LED: Device:LED
8. Battery connector: Device:Battery
9. I2C connector: Connector:Conn_01x04_Male
10. Serial connector: Connector:Conn_01x04_Male
11. ICSP (SPI) connector: Connector_Generic:Conn_02x03_Odd_Even
12. Digital pins (GPIO) connector: Connector:Conn_01x09_Male
13. GND symbol
14. PWR_FLAG symbol

To keep the schematic clean and readable, eventually, you will place the connector symbols (9, 10, 11 and 12) in a separate hierarchical sheet. Learn how to create hierarchical sheets in the recipe with the title '43. How to use hierarchical sheets.'

For the DS1337S+, I made a custom symbol because I could not find one with the exact pin configuration for this module. Create one too. You can use the example in Figure 19.5 for guidance. You should also use information from the component datasheet to help with the design. The datasheet is available at this location[19] (footprint information) and this location[20] (full datasheet). To learn how to design a custom symbol, please read the recipe titled '35. Creating a new component (symbol)'.

[19] https://pdfserv.maximintegrated.com/package_dwgs/21-0041.PDF, visited December 12, 2018.

[20] http://www.farnell.com/datasheets/74348.pdf, visited December 12, 2018.

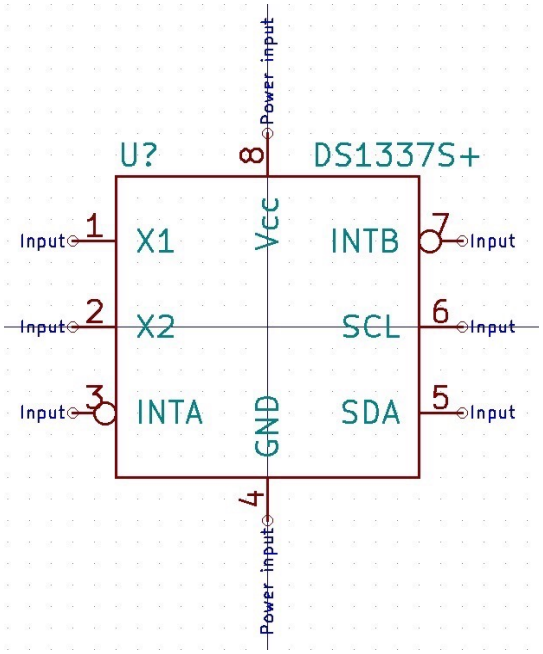


Figure 19.5:The only custom symbol for this project.

Start with the root sheet (the default sheet), and add items 1 to 8, 13 and 14. Remember to set the property values, especially for the capacitors and the resistors. R1 is 330 Ω, R2 is 10 kΩ, C1 and C2 are 22 pF, and C3 is 10 uF.

The sheet should look like this example in Figure 19.6.

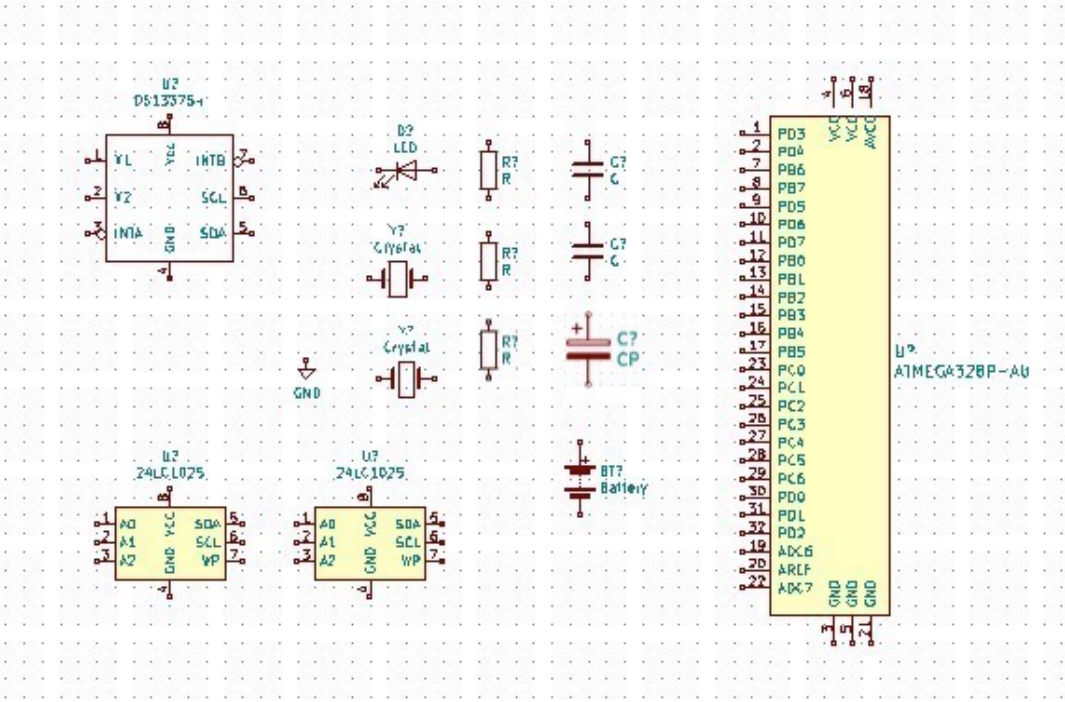


Figure 19.6: The symbols in the root sheet.

Then, create a new hierarchical sheet. To learn how to work with hierarchical sheets, please read the recipe titled '43. How to use hierarchical sheets'. Give the new sheet a reasonable name. Since it will contain connector

symbols, a good name is 'Connectors.'. In the new sheet, add symbols 9 to 12 (Figure 19.7).

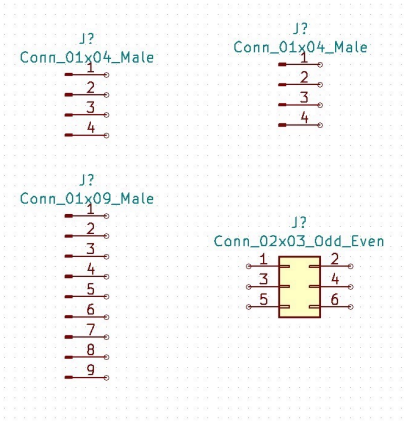


Figure 19.7: The contents of the Connectors sheet.

Continue with Step 3.

Step 3: Arrange, Annotate, Associate

In Step 3, you will work on the three 'A's: Arrange, Annotate and Associate.

Arrange

Unlike the previous projects, project 3 has a higher number of symbols and pins to connect. If we use only wires for the connections, the end result would be hard to read. Instead, we will use a combination of wires and, mostly, labels. We reserve wires only for those pins that are in near proximity. For everything else, we'll use net labels.

With this in mind, place the symbols in functional groups, allowing for sufficient space between the groups to make the schematic comfortable to work with and read.

In the root sheet, my schematic looks like the one in Figure 19.8.

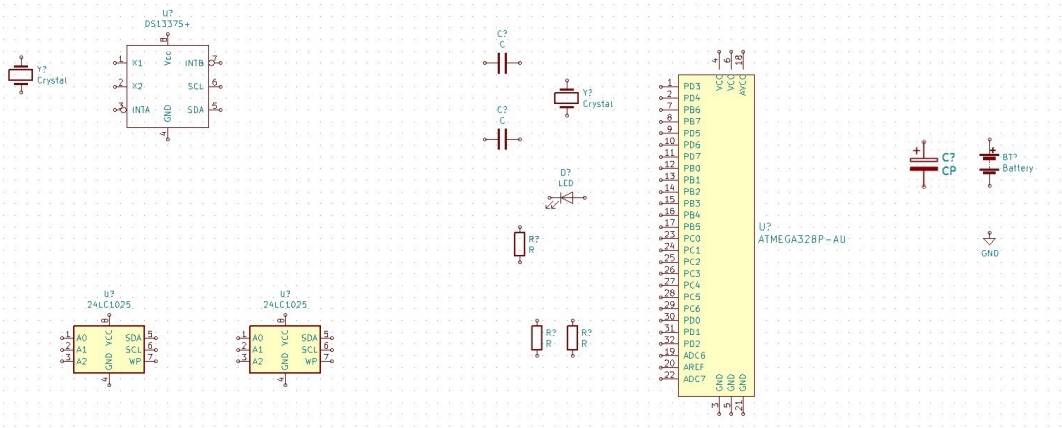


Figure 19.8: The current state of the root sheet.

In the Connectors sheet, things are much simpler since we only have the four connector symbols (Figure 19.9).

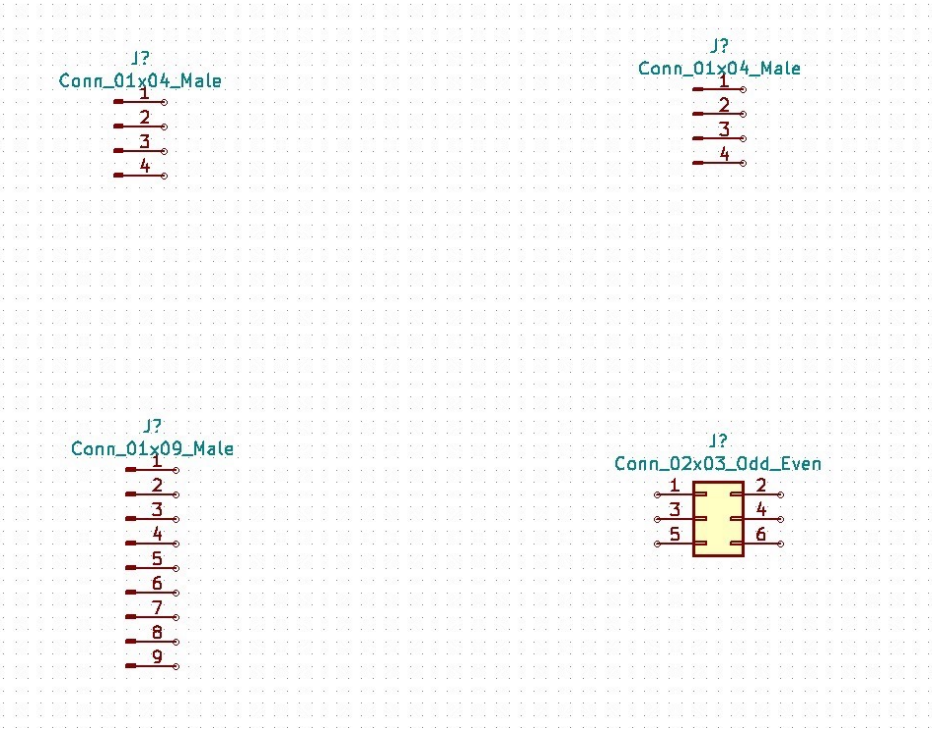


Figure 19.9: The current state of the Connectors sheet.

Annotate

Annotate all symbols by clicking on the Annotate Schematic button. The symbols in both sheets will get a unique designator.

Associate

The symbol to footprint association will require a lot of your attention. All of the components in this project are available in multiple packages. The crystal oscillators can come in a variety of sizes, and with two or four pins. The passive components, like the resistors and the capacitors, also come in a variety of widths and lengths. And the integrated circuits, like the clock/calendar module, can come in a multitude of packages, like DIP, MSOP, SOIC, and the ultra-small uSOP, with 8 or 16 pin configurations. You will need to spend a lot of time pouring over manufacturer datasheet and supplier websites to find the package that is available for purchase and that fits with your design requirement before you make the associations here.

In my design, I used SMD components that are reasonably easy to solder by hand. For the resistors, capacitors and the LED, this means 0805 packages. For the ICs, I chose the largest SMD packages available for each

one. You can see my choices in Figure 19.10. I used Cvpcb to make the associations in bulk.

Reference	Value	Footprint
BT1	Battery 3V	Connector_PinHeader_2.54mm:PinHeader_1x02_P2.54mm_
> C1, C2	22 pF	Capacitor_SMD:C_0805_2012Metric_Pad1.15x1.40mm_Han
C3	10 uF	Capacitor_SMD:C_0805_2012Metric_Pad1.15x1.40mm_Han
D1	LED	LED_SMD:LED_0805_2012Metric_Pad1.15x1.40mm_HandSo
J1	GND_rail	Connector_PinHeader_2.54mm:PinHeader_1x05_P2.54mm_
J2	I2C connector	Connector_PinHeader_2.54mm:PinHeader_1x04_P2.54mm_
J3	Digital pins	Connector_PinHeader_2.54mm:PinHeader_1x09_P2.54mm_
J4	Serial connector	Connector_PinHeader_2.54mm:PinHeader_1x04_P2.54mm_
J5	ICSP connector	Connector_PinHeader_2.54mm:PinHeader_2x03_P2.54mm_
J6	5V_Rail	Connector_PinHeader_2.54mm:PinHeader_1x05_P2.54mm_
> J7-J26	Conn_01x01_Female	Connector_Pin:Pin_D1.0mm_L10.0mm_LooseFit
R1	330 Ω	Resistor_SMD:R_0805_2012Metric_Pad1.15x1.40mm_Hand
> R2, R3	10 kΩ	Resistor_SMD:R_0805_2012Metric_Pad1.15x1.40mm_Hand
> U1, U3	24LC1025	Package_SO:SOIJ-8_5.3x5.3mm_P1.27mm
U2	DS1337S+	Package_SO:SO-8_5.3x6.2mm_P1.27mm
U4	ATMEGA328P-AU	digikey-footprints:TQFP-32_7x7mm
Y1	Crystal 32768 kHz	Crystal:Crystal_SMD_MicroCrystal_CC7V-T1A-2Pin_3.2x1.5n
Y2	Crystal 16MHz	Crystal:Crystal_SMD_5032-2Pin_5.0x3.2mm_HandSoldering

Figure 19.10: The final associations table.

Before you continue with Step 4, edit the values of the symbols as per Figure 19.10. For example, edit the C3 capacitor that is connected to the battery to be 10uF, and so on.

```
Important: use Latin characters for the values ('u' instead of
'µ' and 'Ohm' instead of 'Ω'), instead of the Greek ones.
This is because the FreeRouting autorouter which you will use
during the layout process has a bug which causes it to freeze
when symbol values contain non-ASCII characters.
```

Let's jump to Step 4, wiring.

Step 4 and 5: Wiring and Nets

In this project, you will use net labels to do almost all of the wiring in place of regular wires. You will also use hierarchical labels and pins to 'connect' the pins of the symbols in the two sheets that make up this project.

In the previous projects, in Step 4 we simply focused on doing the wiring, using wires. As a consequence, we allocated names for the various nets in Step 5. Because I have chosen to use named nets to do most of the wiring, I am combining Steps 4 (wiring) and 5 (nets) into a single step.

To improve readability and reduce the risk of defects, I choose to draw the schematic and do the wiring with the following attributes:

1. Group the components according to function. The micro-controller unit and its supporting symbols will consist one group. The EEPROMs, another. The clock, another. And the connectors, another group.
2. To reduce clutter in a single page, I will use hierarchical sheets. I will place the bulk of the symbols on the root sheet, and the connectors in the child sheet.
3. Also to reduce clutter and the risk of error, I will use named nets to connect pins, instead of wires.

I find it more logical to start the labelling process in child sheets, and then work my way up to the parent. This way you can be sure that once the parent (root) sheet is complete, the wiring is also complete.

Go over to the Connectors sheet. Because we want to connect the pins of those connectors to nets that exist in the root sheet, we will use hierarchical labels. For any connections that exist within the sheet (i.e. that don't make any references to nets outside the sheet), you can use regular net labels. Let's start with the ICSP connector. Click on the hierarchical label tool to select it and label the ICSP connector as in this example in Figure 19.11. If you don't know how to do this, review the recipe titled '43. How to use hierarchical sheets'.

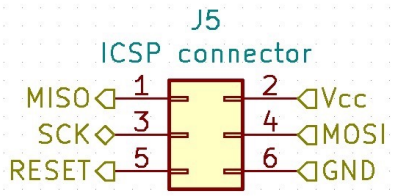


Figure 19.11: The hierarchical labels of the ICSP connector.

Continue with the rest of the connectors in this sheet. The finished sheets is in Figure 19.14.

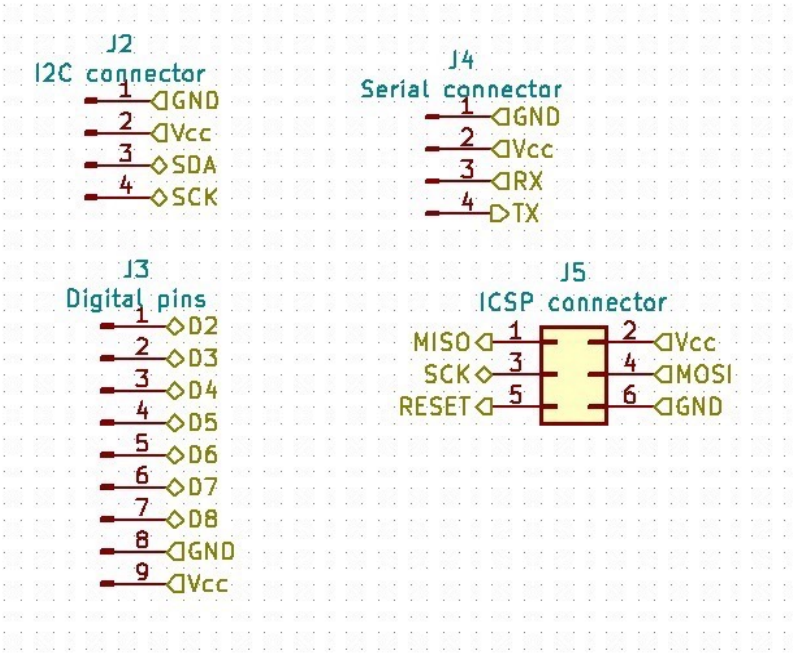


Figure 19.12: Completed labelling in the Connectors sheet.

Switch over to the root sheet. I suggest you start with the central symbol, the Atmega328 microcontroller. Use labels to mark all communications pins (I2S and SPI), as well as the GPIOs that you want to break out to the digital pins connector. Use wires only for the external 16MHz oscillator, led, reset line and battery. Don't forget to use the Unconnected symbol for any pins that are not connected to a net. You can see the current version of the root schematic sheet in Figure 19.13. You can download a high-resolution version of the schematic in PDF format by visiting txplo.re/klpr.

A few additional points to notice:

- Use wires to connect symbols that belong to the same functional group together.
- Use net labels to connect pins across functional groups.
- Use the PWR_FLAG to mark the Vcc net as a power net and prevent related ERC errors

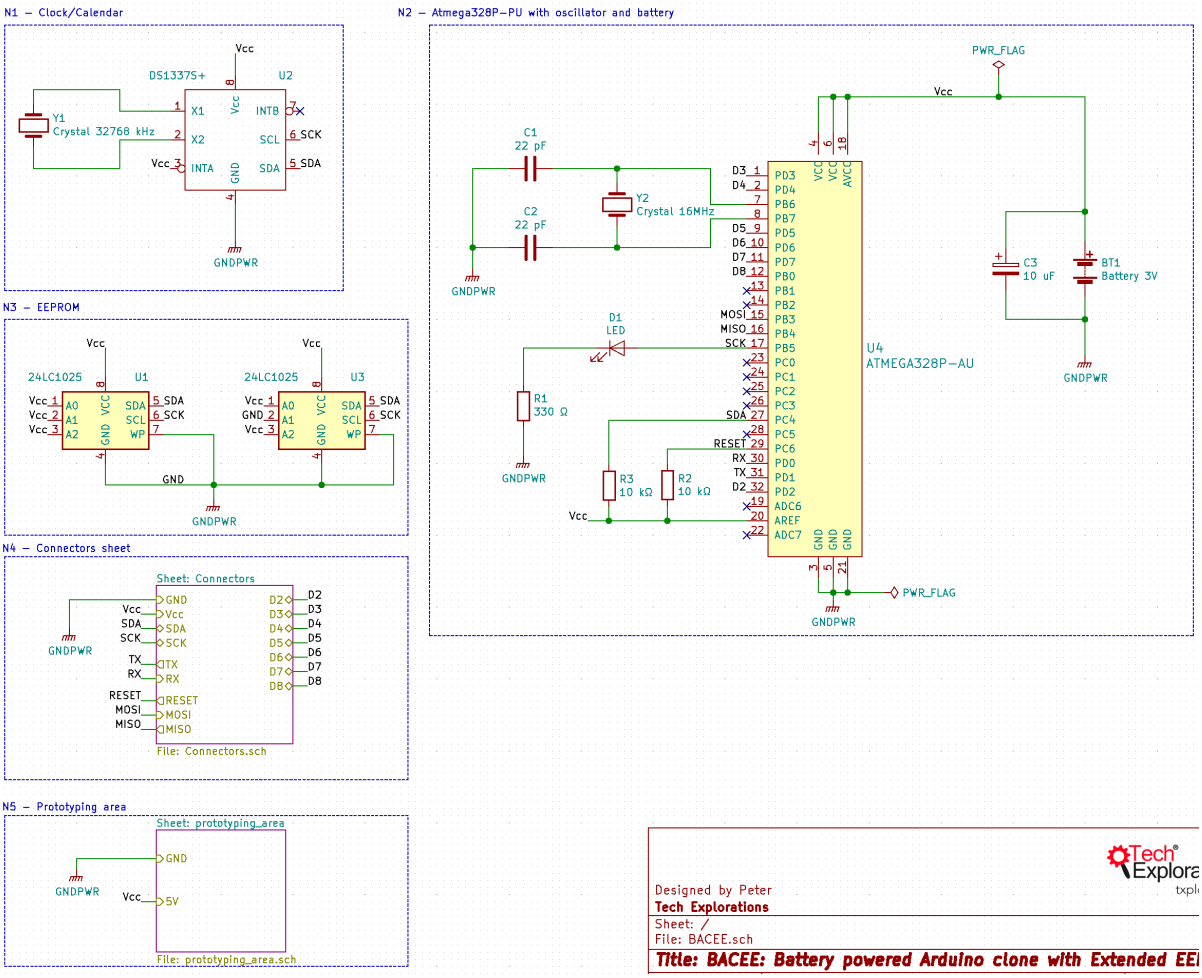


Figure 19.13: The root sheet that contains the bulk of the symbols, wired.

You can download a high-resolution PDF copy of the schematic in 19.13 from the book support page (txplo.re/kicadr).

Notice that in this schematic, I have used the GNDPWR symbol, instead of the GND symbol to denote ground. The GNDPWR symbol is more appropriate as per the IEC 60617 specification. Electrically, both work in exactly the same way, and the PWR_FLAG symbol is still necessary to allow the ERC to pass.

You still need to connect the nets of the root sheet with the nets of the Connectors sheet. Use the hierarchical pin importer tool to do this easily. The only 'challenge' is to decide on the location of the pins. My practice is to first export all the pins and then re-arrange them based on the symbol to which they belong. You can see my completed work on the Connectors sheet rectangle in Figure 19.14.

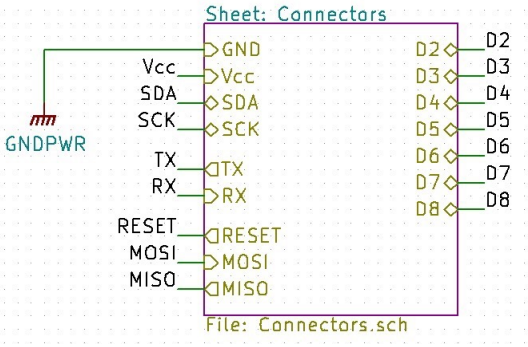


Figure 19.14: The Connectors sheet rectangle, labeled and wired.

The wiring and net naming steps are complete, let's move on to do the Electrical Rules Check.

Step 6: Electrical Rules Check

You should be doing frequent electrical rule checks as you work in the wiring and labelling steps (4 and 5), but you should always run it again before you continue.

Got the all-clear (an empty Error list)? Let's move on to Step 7.

Step 7: Comments

Let's add some informative comments and graphics that improve the readability of your schematic. I have includes component values, like that of the resistors, and boxes to mark-up functional blocks. I also spread out the elements of the schematic to better use the available space, and to make it easier to perform modifications and additions in the future.

The final schematic is in Figure 19.15 and Figure 19.16. You can download a high-resolution version of the schematic in PDF format by visiting txplo.re/klpr.

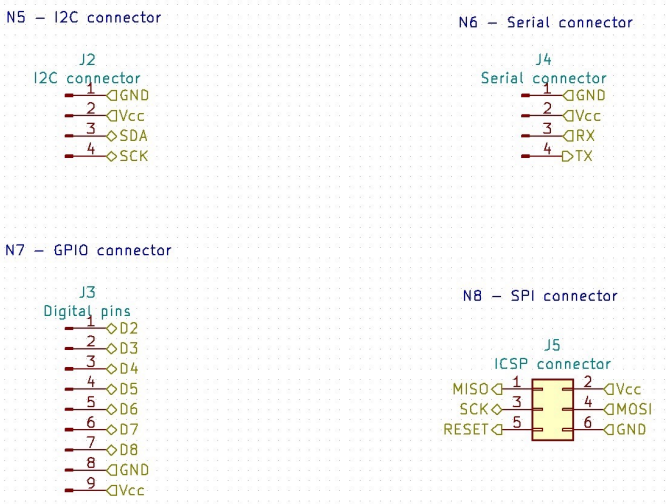


Figure 19.15: Connector sheet graphics added.

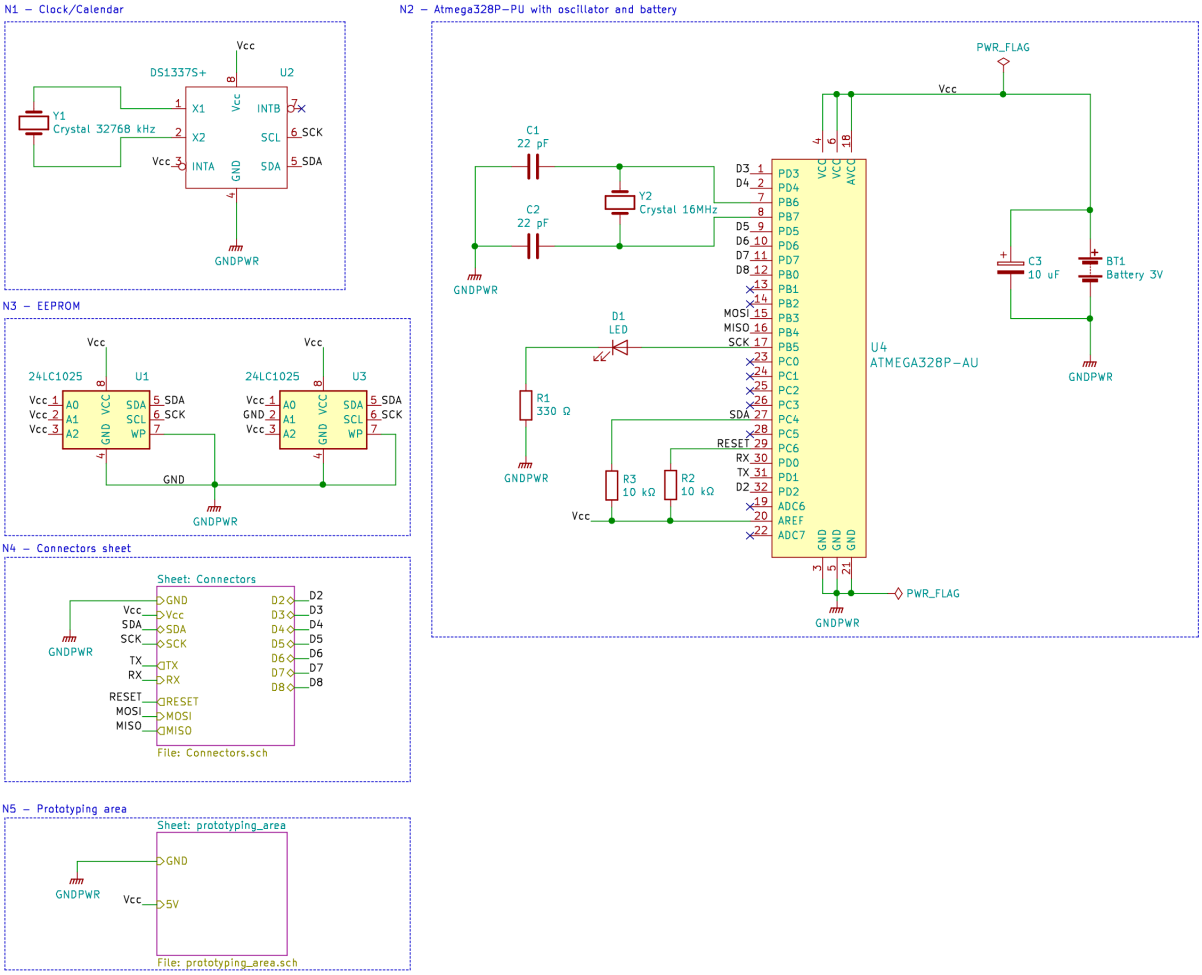


Figure 19.16: Root sheet graphics added.

Let's get into the next and final step where you will export the netlist file.

Step 8: Netlist

You have finished with the schematic, and the ERC is clear of errors. All that is left to do is to produce the Netlist file so you can continue with the layout process. Click on the Netlist button, accept the default settings in the Pcbnew tab, and click on 'Generate Netlist' to produce the file. Accept the default netlist name, for consistency. Close the windows.

You should have these 6 files in the project folder (Figure 19.17).

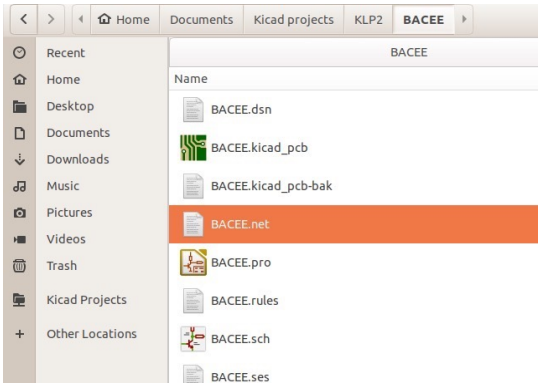


Figure 19.17: Checking the project files.

Let's continue with the layout process, where you will import the .net file into Pcbnew.

19.4. Footprint layout in Pcbnew

Let's start the layout process. To do this, you will follow the steps illustrated in Figure 19.3, with a small 'twist' as you will see soon.

Step 1: Setup

For this project, I will use the same design rules settings as those in the first project. I'll be using the same manufacturer, so I do not need to make any changes to the minimum values for things such as the drill size and the track width. I will only customise the text of the page settings window (Figure 19.18).

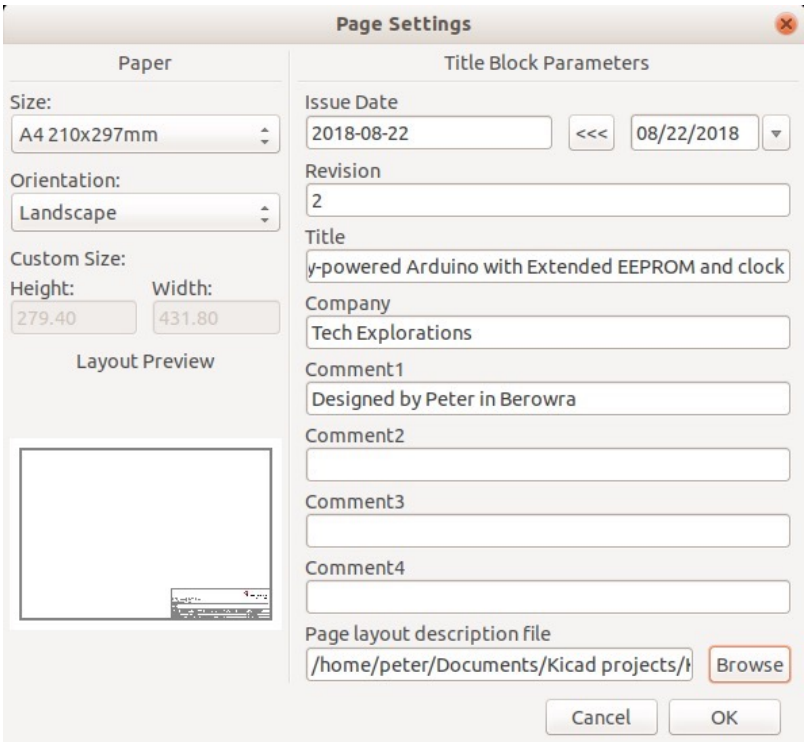


Figure 19.18: Project 3 Page Settings.

Before you set up the design rules, import your netlist file. This will allow you to allocate the nets to the appropriate net class. My setup is showing in Figure 19.19.

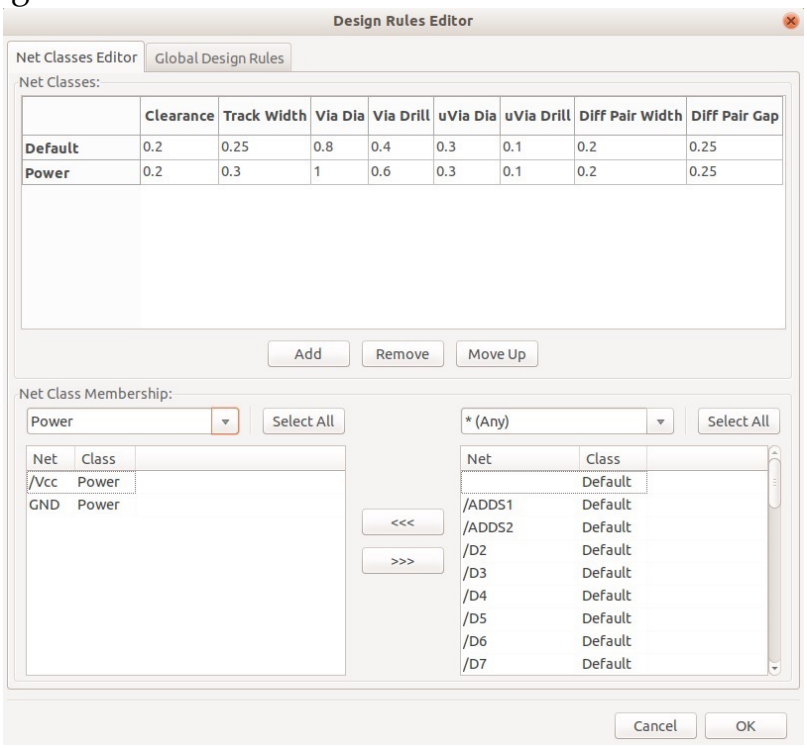


Figure 19.19: Design rules and net classes.

Continue with the outline and the mechanical constraints.

Step 2 + 3: Outline, constraints and component placement

For this project, my main design guideline is size. I want to make this board as small as possible so that it fits in small places, or it can become part of other projects. This is also why I opted to use SMD components.

In addition, I decided to use regular pin headers so that I can connect regular jumper wires during prototyping. This decision will increase the size of the board since these headers are relatively large. However, I can always solder peripherals directly to the pads if I want to reduce the height of the board, or even design a version of the board that uses smaller (including SMD-type) connectors.

I would like to make this board rectangular, to make it easier to fit inside a project box. I will make the corners round because they look awesome.

Lastly, I decided that having mounting holes is necessary to be able to secure the board inside a project box. I'd like to place the holes along the four corners of the board.

Because I don't have any measurements to go with, my process is to start with the components, and then create the outline around them. While in projects 1 and 2, I followed the process depicted in Figure 19.3, this time I combined steps 2 and 3 and iterated through them until I was happy with the outcome.

I started by placing the footprints as closely as I could, being mindful of the space needed to do the routing later. I placed the MCU at the centre of the board since everything has to connect to it. Then, I placed the EEPROM chips on one side, and the clock on the other.

Then, I placed the supporting components (resistors, crystals, capacitors) as close as I could to their connecting footprints.

Finally, I worked on the UI, the user interface. That is, the headers and the indicator LED. I placed those along the edges of the board.

At this point, the board looks like the example in Figure 19.20.

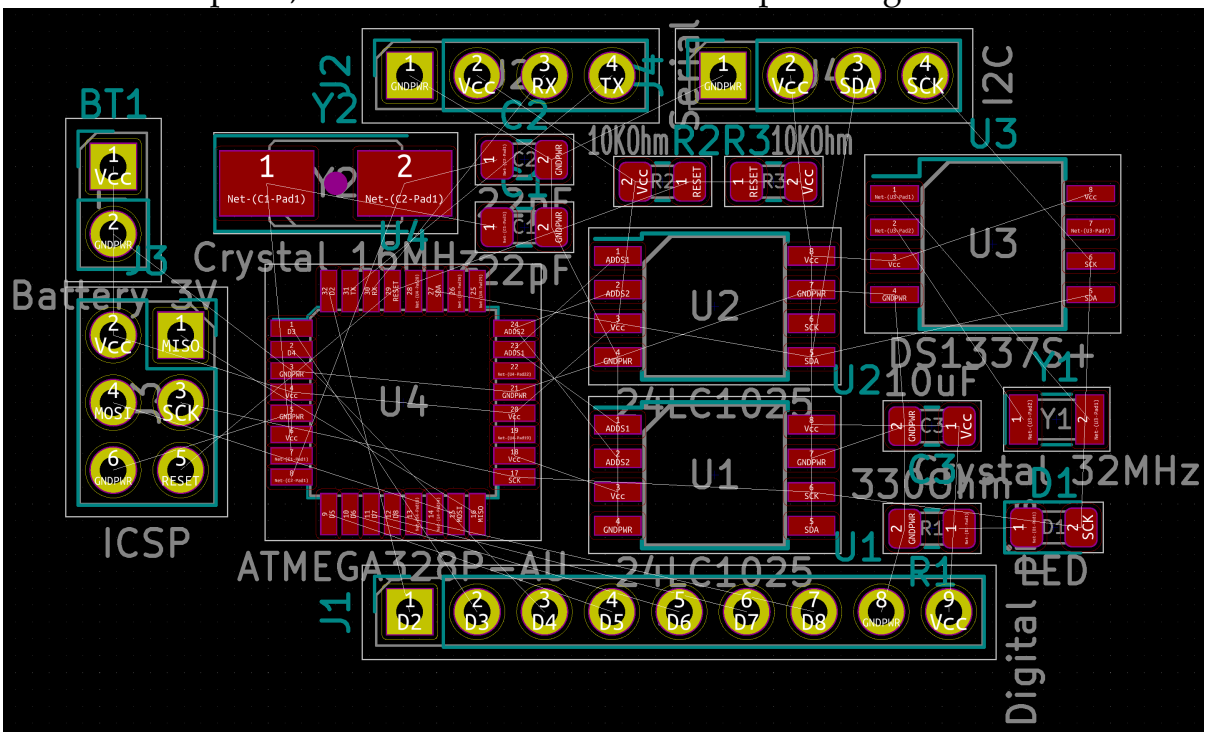


Figure 19.20: First iteration of the layout.

This layout can improve. For example, R2 is probably better placed on the right side of C1. The ratsnest lines indicate that the tracks will be shorter if I do place it there. I also think that I can reduce the distance between the IC footprints. You can try those ideas on your own if you like. However, for the first iteration, this placement is adequate. Time to draw the outline in the EdgeCuts layer, and create the mounting holes. To make drawing easier, I am using a grid size of 0.508 mm.

You can see the outline of the first iteration in Figure 19.21. I did move R2 to the left of C1 and used the alignment tools (available from the context menu) to align them horizontally. Try to reduce the distance between the footprints and bring the connectors inwards. This will produce a more compact board, which will cost less to manufacture.

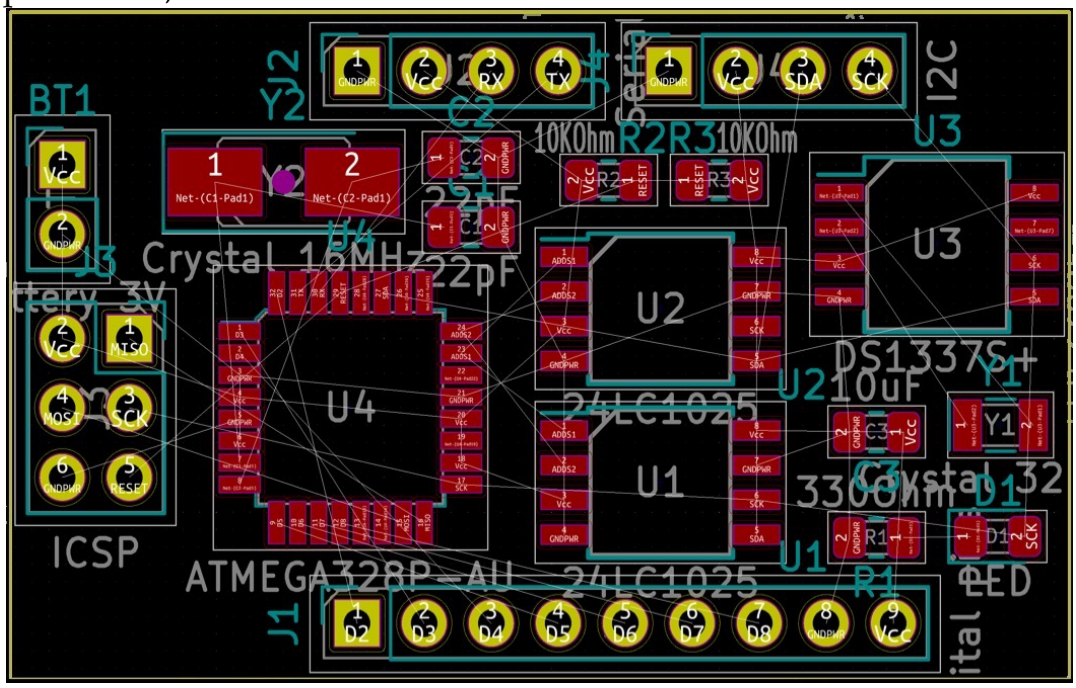


Figure 19.21: First iteration of the layout with its outline.

To make sure that the footprints were not too close to each other (making routing hard or impossible later), I did a test run of the FreeRouting autorouter. The autorouter was able to fully route this board. This gave me the confidence to commit to it, with a caveat: I must add the mounting holes and do another auto routing test before I fully commit to the design.

In Figure 19.22 and Figure 19.23 you can see the final board outline, with the mounting holes and the rounded corners. The radius of the mounting holes is 1.136 mm, and of the rounded corners 1.02 mm.

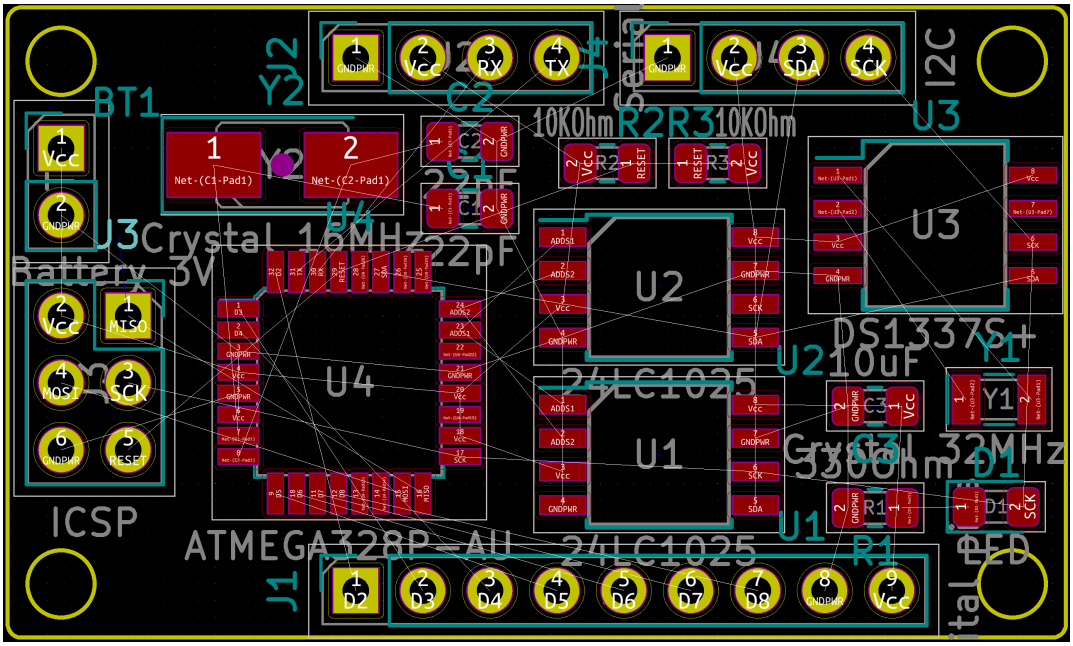


Figure 19.22: The final version of the placement and outline.

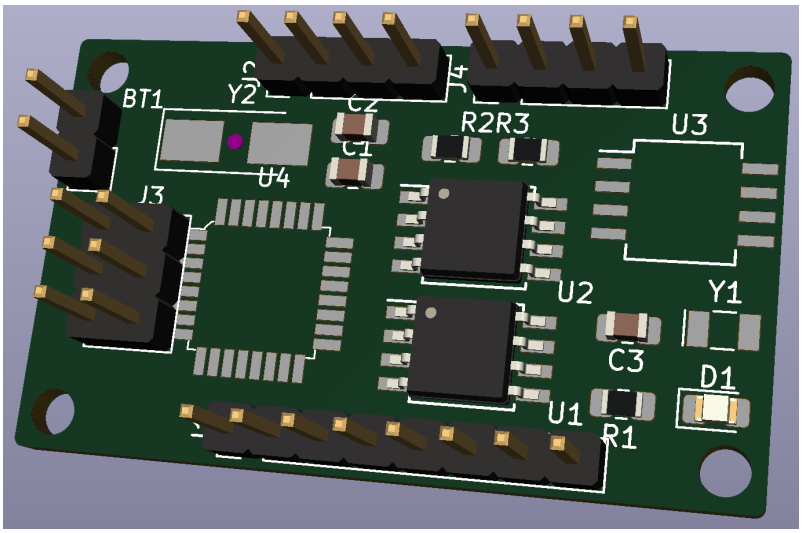


Figure 19.23: 3D rendering of the board.

I did a second test autoroute run and again I was able to fully route the board. With this information, I decided to commit to this placement and outline. Before you continue, consider if this placement and outline can be improved. For example, can the footprints be placed even closer to each other? How will that impact hand-soldering? Can you move the passive components in other locations in order to free up space above the J2 and J3 connectors so that you can further reduce the width of the board?

When you are ready, continue with step 4, where you will route the board in a 2-layer and a 4-layer configuration.

Step 4: Route

The board in this project 3 is denser than the ones in the two previous projects. Without any special provisions, and with the help of the autorouter, you can easily fully route a two-layer version. However, I think that this project presents a good opportunity to demonstrate how you can work in multiple layers. Therefore, in addition to routing the board in two layers, you will also route it in four layers.

Using the autorouter - two layers

Pcbnew is set to a two-layer board configuration by default. You can confirm that by accessing the Board Setup window, through the File menu item (see Figure 19.24).

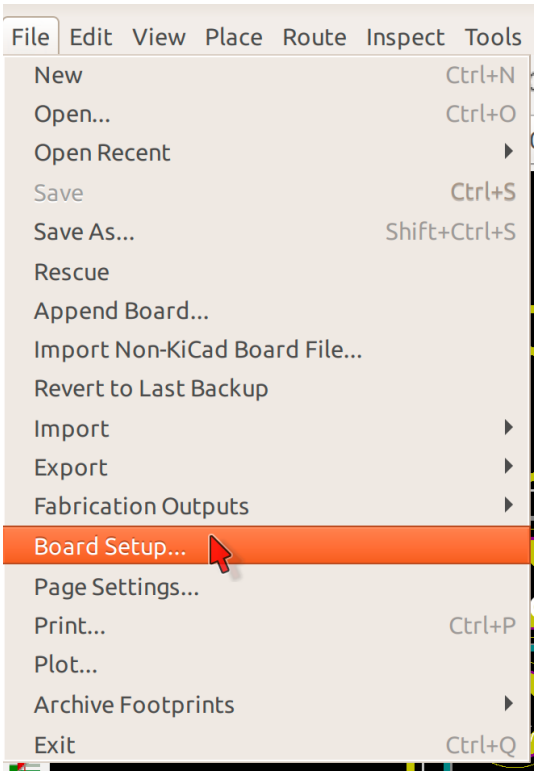


Figure 19.24: The Board Setup menu item.

This will bring up the Board Setup window, which looks like the one in Figure 19.25.

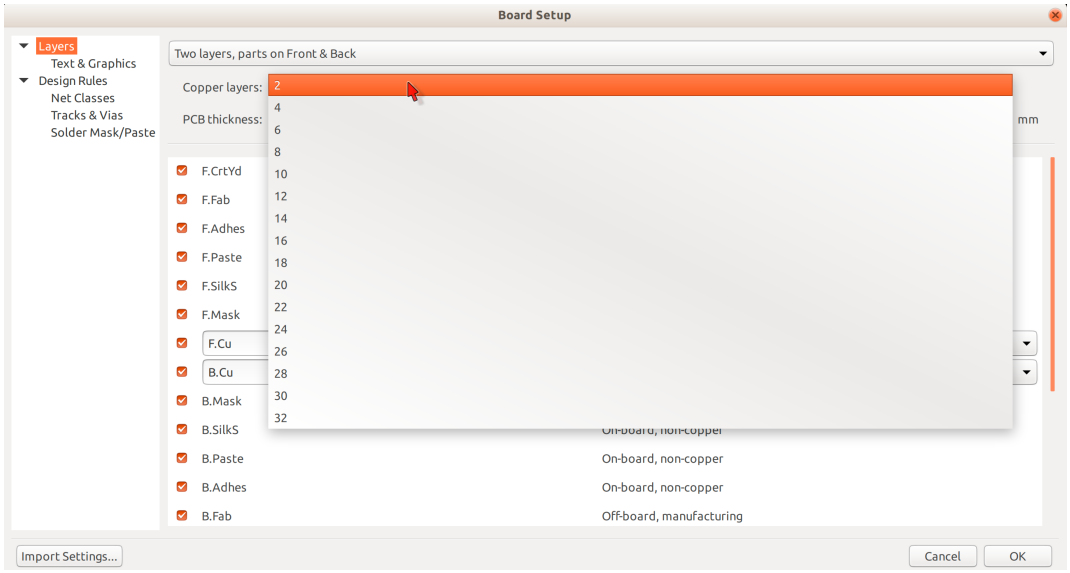


Figure 19.25: The two-layer configuration.

Focus on the configuration of the front and bottom copper layers, in the red box. You can use the drop-down menus to assign a specific type of track to each layer. You can choose between signal, power, jumper, and mixed. These types are treated as recommendations by the autorouter. In this example, leave them as they are, and you will see how FreeRouting will still route the power traces in the bottom copper layer, even though they are clearly not signal traces.

Use the instructions in the recipe titled '40. Using an autorouter' and perform the autorouting of this board. FreeRouting's result is in Figure 19.26.

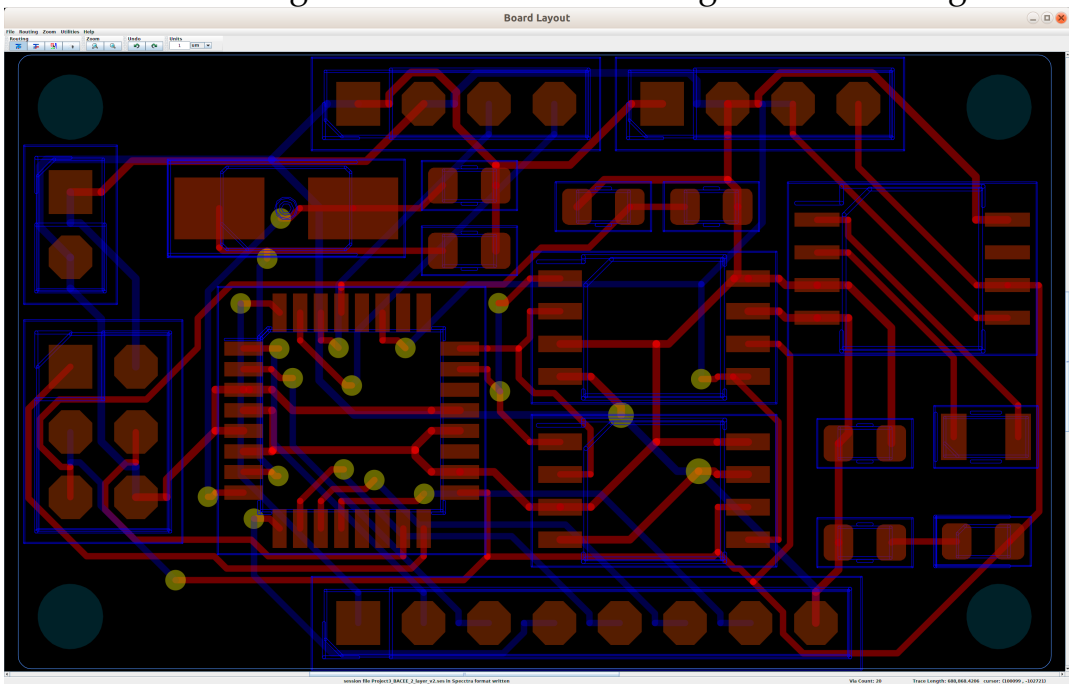


Figure 19.26: FreeRouting's completed routing of the two-layer board.

After you import the '.ses' file back into Pcbnew, your board will look like the example in Figure 19.27.

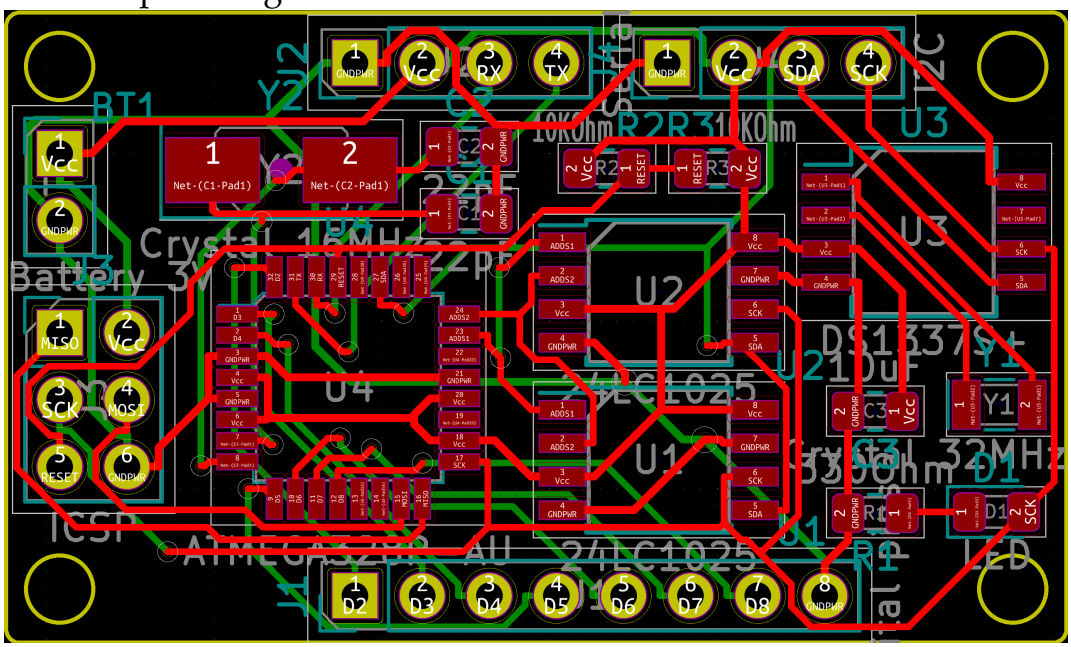


Figure 19.27: The two-layer board, fully routed, in Pcbnew.

Let's repeat the exact same process to create the four-layer version of this board.

Using the autorouter - four layers

To create the four-layer version of this board, start by clearing the tracks of the two-layer version. You can do this quickly through the Global Deletions menu item, under Edit (Figure 19.28).

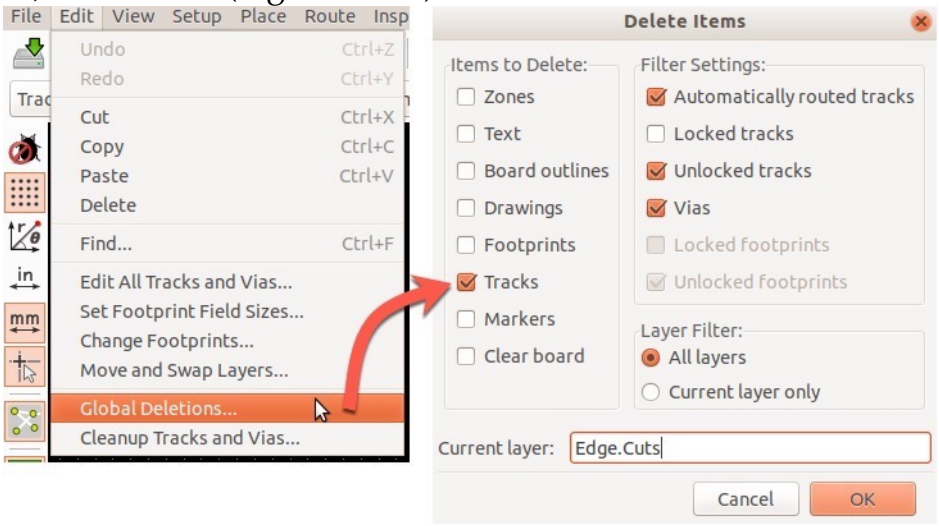


Figure 19.28: Delete existing tracks.

With your board clear of tracks, open the Board Setup window (under the File menu bar item), and switch to the four copper layers option from the Copper Layers drop down. There is also a drop-down at the top of the Board

Setup window which gives you presets for various board configurations. Because our board will only have parts mounted on the front side, choose "Four layers, parts on Front" in this drop-down.

In the second pane of this window, you can see the default board configuration for the current choice. Particularly interesting are the items in the red box, where you can see that the top and bottom layers are allocated to signals, and the two inner layers are allocated to power. These are sensible choices, so I'll leave them like that. You can see my four-layer board setup in Figure 19.29.

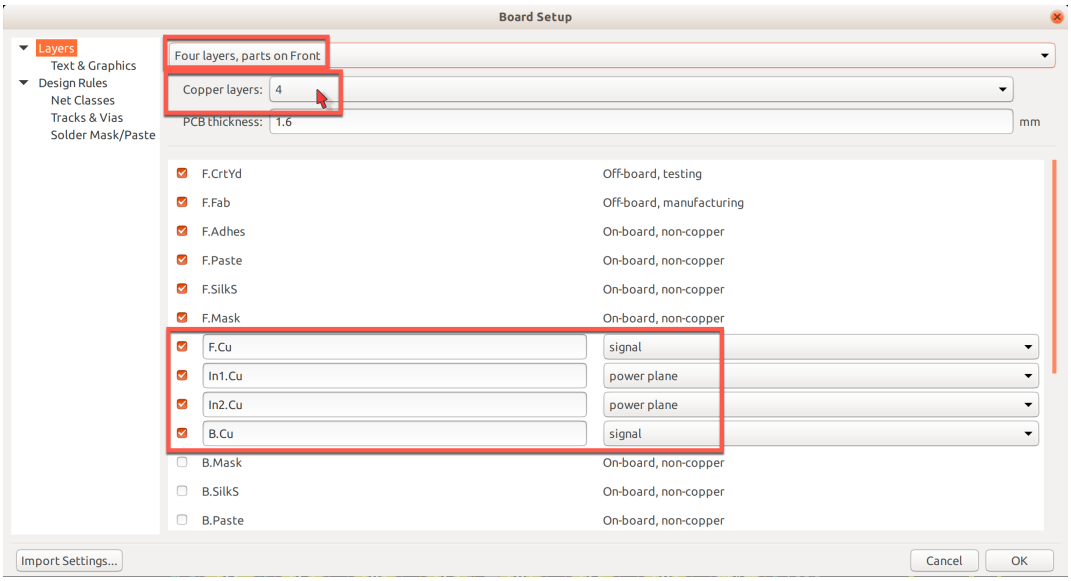


Figure 19.29: Configure the four-layer board.

Click OK to commit these changes. Repeat the FreeRouting process to have this board automatically routed. I saved the new DSN file with the file name 'BACEE_four_layer.dsn' to distinguish it from the two-layer version. There is nothing else you need to do in FreeRouting. All the information FreeRouting needs in order to route the board is in the Specctra DSN file.

You can see the FreeRouting result in Figure 19.30.

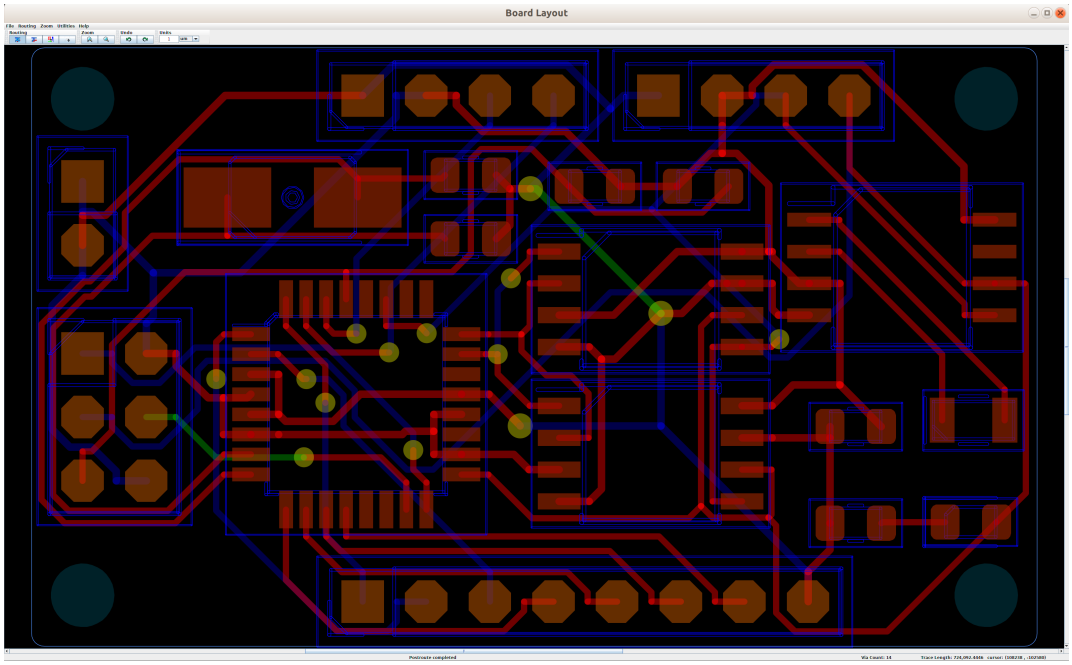


Figure 19.30: FreeRouting's completed routing of the four-layer board.

You can see the fully-routed four-layer board in Pcbnew, in Figure 19.31.

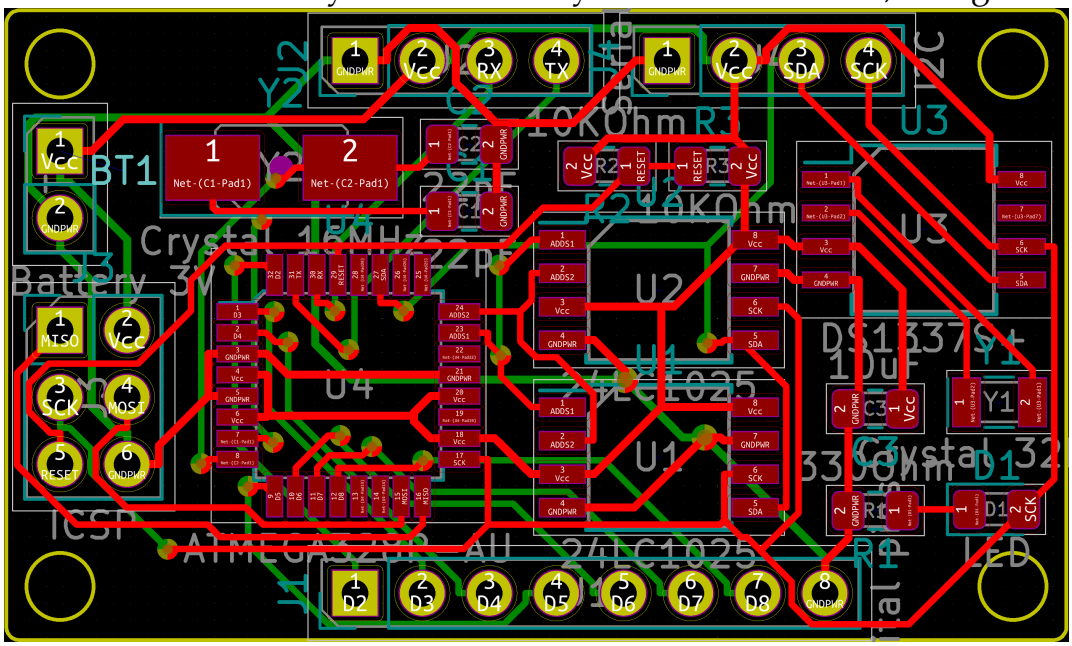


Figure 19.31: The four-layer board, fully routed, in Pcbnew.

Two layers, or more?

You may be wondering, is there any advantage of opting for a four layer board over a two-layer board?

In short, no.

If you can have your board routed in two layers, you will save money. I had both versions of the board if this project manufactured at Pcbnew. The

PCB cost (excluding shipping) for the four layer board was $55, and for the two-layer board was $18. That is a significant difference.

I would only consider four or more layer boards for very dense boards where routing in 2 layers is impossible without jumpers.

Step 5: Copper fills

I continue from the previous step using the two-layer version of the board. If you prefer to work on the four layer board, the process for creating copper fills is exactly the same. Because I kept the two-layer board Specctra Session file, I can simply delete all traces and load the Specctra Session file for the two-layer board.

In this step, you will create a copper filled zone in the back copper layer, and connect it to the GND net. Follow the process described in the recipe '25. Creating copper fills'. The result is shown in Figure 19.32.

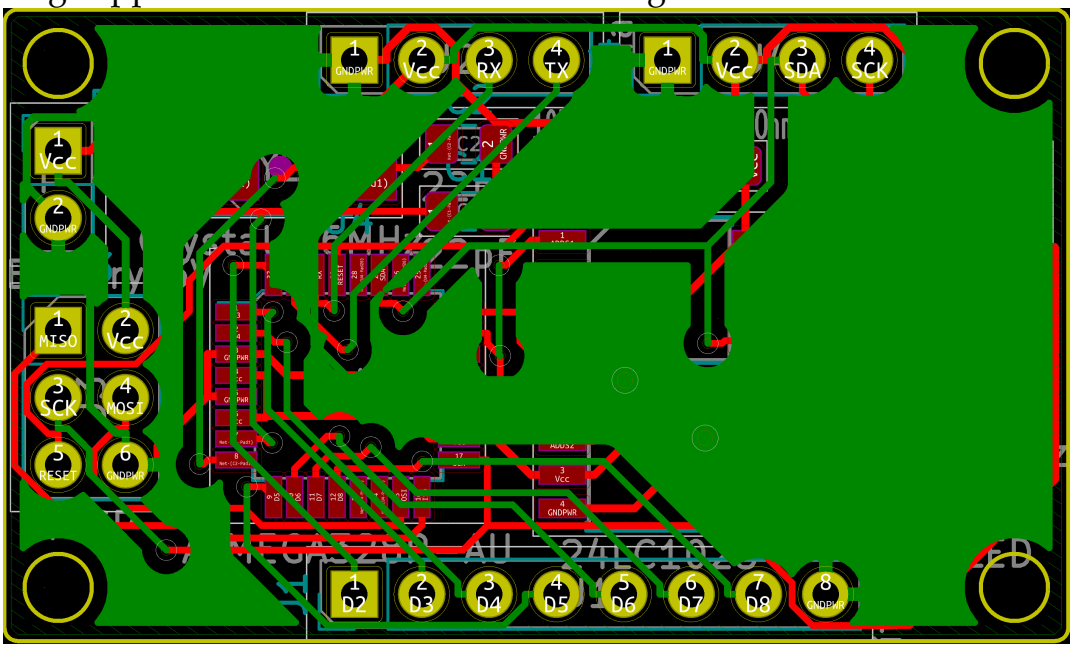


Figure 19.32: The GND copper filled zone, in green.

Run the design rules check one more time to ensure that there are no violations. The list should be clear of errors or warnings.

Let's work on the silkscreen next.

Step 6: Silkscreen

In the front and back silkscreen layer, I would like to:

1. Provide information about the values of the resistors.
2. Clearly mark the anode and cathode of the LED.
3. Label the GPIO pin names.
4. Add the board name and version number.

5. Add information about the board features.
6. Add contact information.
7. Add a logo and other graphics.

To make the text fit in the available space, you will need to reduce its size and width via the text properties window. I used 0.8 mm for width and height for connector names, and 0.6 mm for designator and pad names. Also, note that much of the information you would like to display in the silkscreen layer is available in the F.Fab layer. You can simply change that to the F.SilkS layer via the text properties window, and adjust their size to make them fit in the available space.

You can change the properties of text multiple text items (as well as those of tracks, vias and graphics) through the relevant bulk properties edit window. There are two such windows, under the Edit menu item in PCBnew. For example, say that you want to set the properties of all footprint reference text to have a text width and height of 0.6mm, and a thickness of 0.15, and to appear in the front silk screen layer. To do this, bring up the "Edit Text and Graphic Properties" window (under Edit), and set the values as per the example in Figure 19.33.

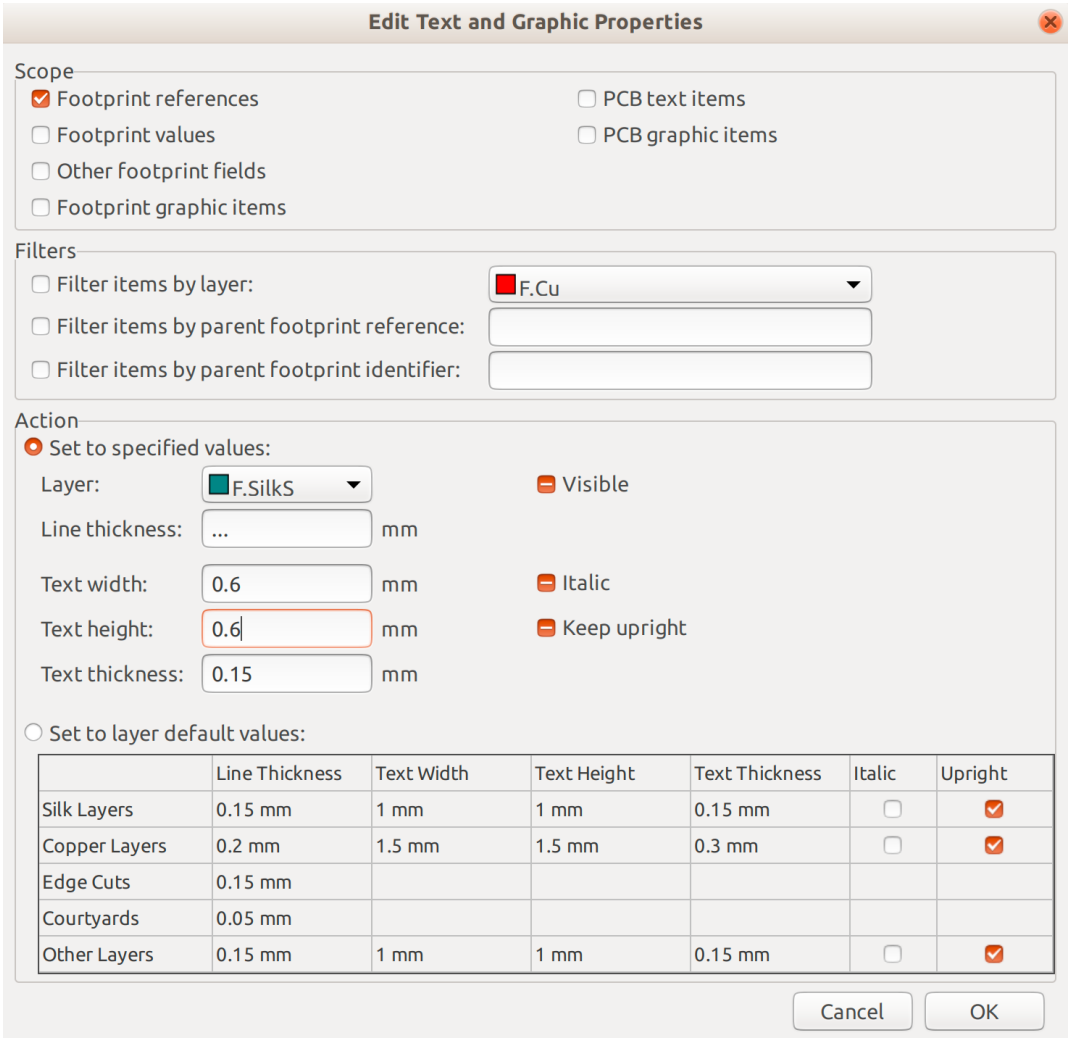


Figure 19.33: With the help of this Edit window, you can set text and graphic properties in bulk.

This window allows you to select other items for which you want to change their properties in bulk, but they all work the same way. Choose what you want to bulk-edit via the Scope and Filter widgets, and then set the values for the various properties. The bulk-edit windows are huge time savers.

As the board is crowded, I prefer to turn off all Cu, Adhes and Paste layers. This will make it easier to see the contents of the silkscreens.

You can see my version of the board, with the silkscreen completed in Figure 19.34.

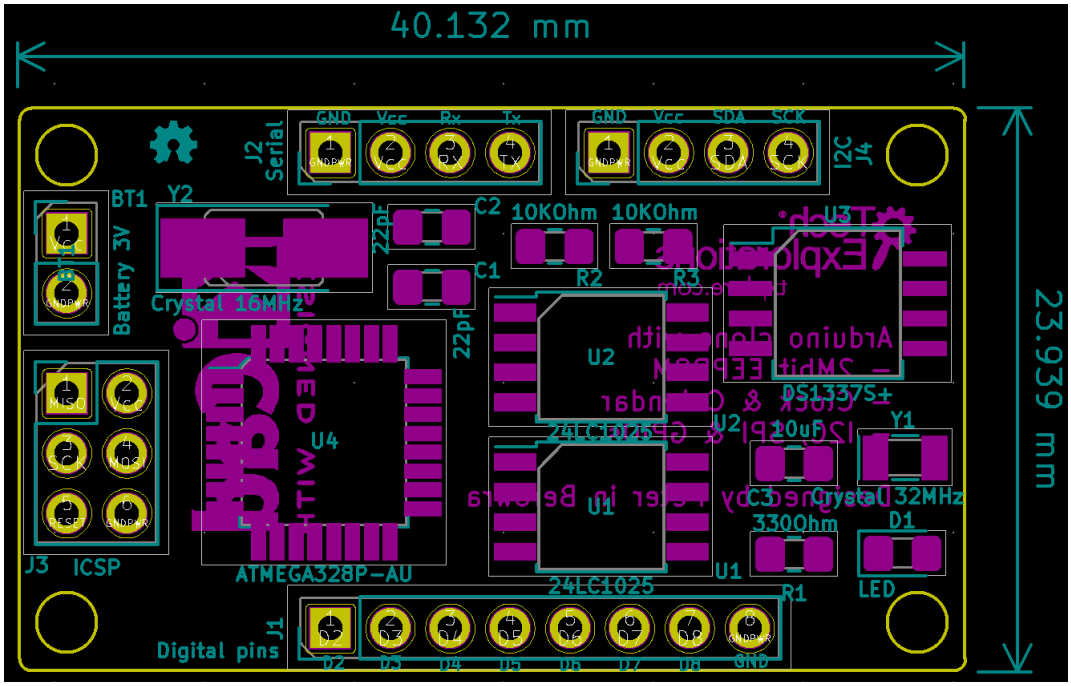


Figure 19.34: The board with the completed silkscreen, back, and front.

The 3D rendering is in Figure 19.35.

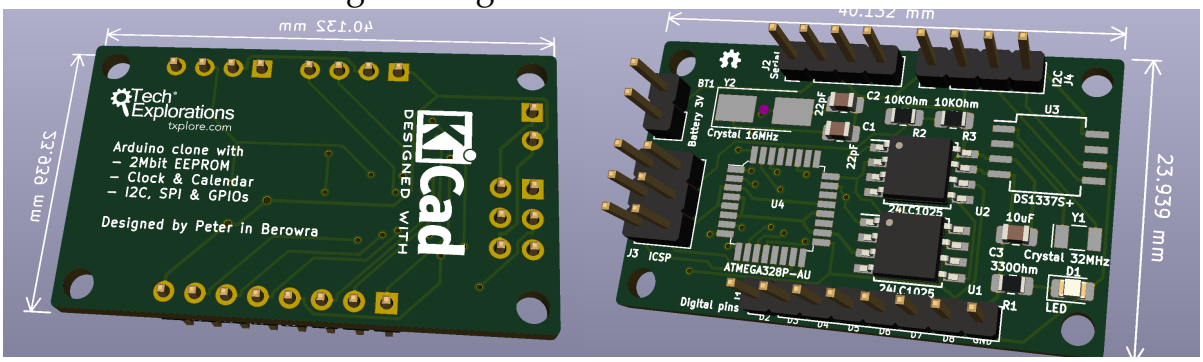


Figure 19.35: 3D rendering of the board, showing the back and front silkscreen.

Your work is almost finished, let's do a final DRC before you export the Gerber files.

Step 7: Design Rules Check

The DRC reports only three courtyard related issues, that you can safely ignore since they don't affect the operation of the board (Figure 19.36).

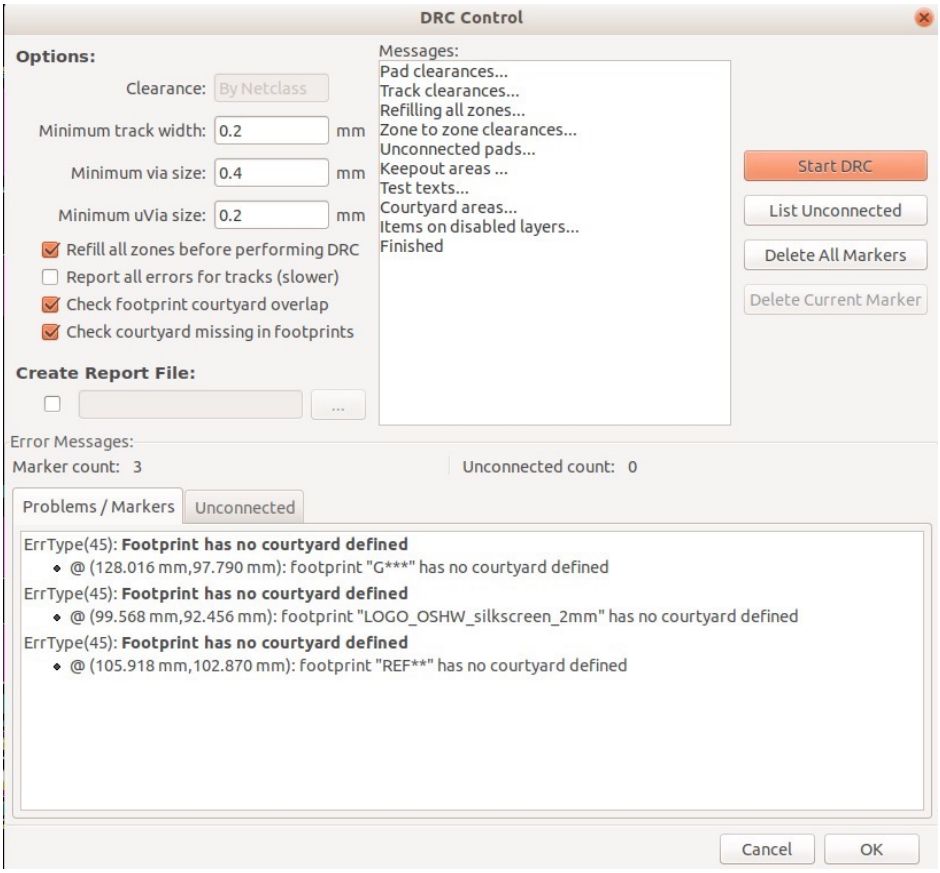


Figure 19.36: Courtyard issues don't affect the operation of the board.

The design work for this board is complete, so let's continue with the manufacturing step.

Step 8: Manufacture

In this final step, you will upload your board to your manufacturer's website. I will use Pcbway, as usual, so I will need to:

1. Export the Gerber files.
2. Create a Zip archive that contains the Gerber files.
3. Test the archive is valid by uploading it to gerblook.org.
4. If the archive is valid, follow the ordering process on the manufacturer's website.

I exported the Gerber files as I did for project 1 and project 2. The validation did not reveal any issues.

Use the distance measuring tool to get a precise measurement of the size of the board so that the price quote you receive is accurate.

Well done!

Part 5: Recipes

20. Adding a schematic symbol library in Eeschema

One of KiCad's great strengths is the sheer number of symbol and footprint libraries contributed by individual users and organisations. In this recipe, you will learn how to find, download and install symbol libraries to KiCad. Once you install a symbol library, you will be able to use its symbols in your schematics, precisely as you can with KiCad's build-in symbols.

You can find libraries for KiCad using Google, through searches like 'KiCad library download.' You may also know the location of contributed libraries as they are often shared in email lists and social media. Quality libraries are contributed by organisations like Digikey, Freetronics and Snapeda. Sources like Snapeda allow you to use a search engine to find individual symbols, footprints or 3D representations of a component and import it into KiCad. Others, like Digikey, allow you to download full repositories of symbols, and footprints and install them in bulk into KiCad. Either way, the process is the same.

In this recipe, you will learn how to import the symbol libraries published by Digikey.

First, use your browser to visit the Github repository from where you can download the libraries archive. The repository looks like the example in Figure 20.1.

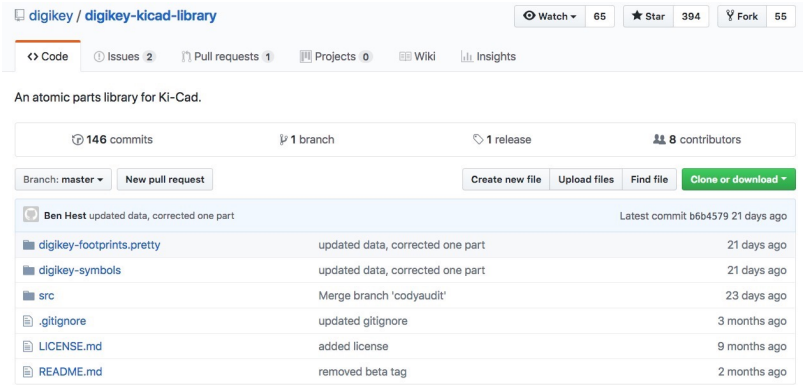


Figure 20.1: The Digikey KiCad libraries repository.

The folder titled 'digikey-symbols' contains symbols. The folder titled 'digikey-footprints.pretty' contains footprints. In this recipe we concentrate on the symbols, and in the next one on the footprints. Before you start the download, click on the symbols directory and have a look inside. You will see a long list of files with '.lib' and '.dcm' filename extensions. The files with the '.lib' extensions are the actual library files that contain the symbols. Those with

the '.dcm' extension contains descriptions, aliases, and keywords for the symbols.

Go back to the root of the repository, and click on the green button to download the ZIP archive of the repository. Expand the Zip file, and store its contents in a folder where you would like to keep third-party libraries. I placed mine in a folder titled 'KiCad Libraries' inside my Documents folder.

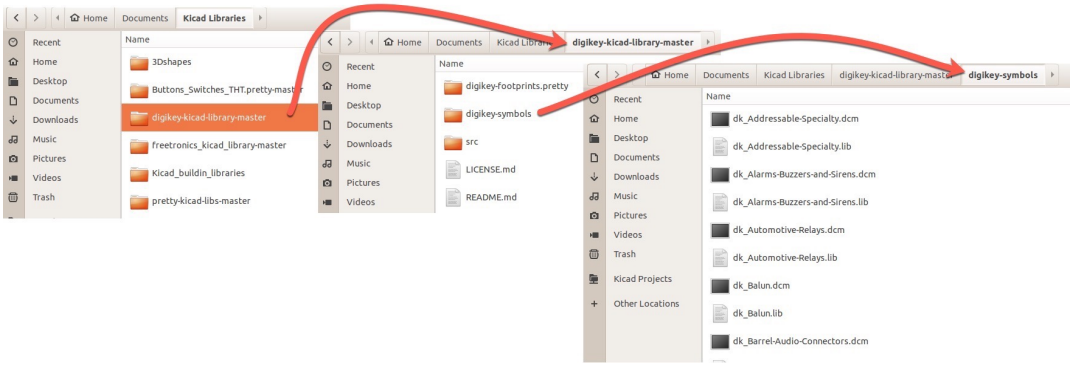


Figure 20.2: The Digikey libraries are now in the KiCad Libraries folder.

You can visit the contents of the Dikigey folder to confirm that it contains everything available on the Github online repository.

Continue by starting KiCad and Eeschema. Open the Library Manager by clicking on Preferences, 'Manage Symbol Libraries…' (Figure 20.3).

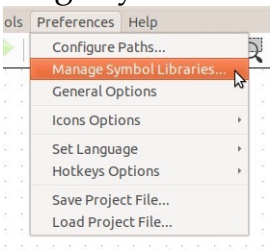


Figure 20.3: Start the Library Manager in Eeschema.

In the Library Manager, click on the 'Browse Library' button (Figure 20.4).

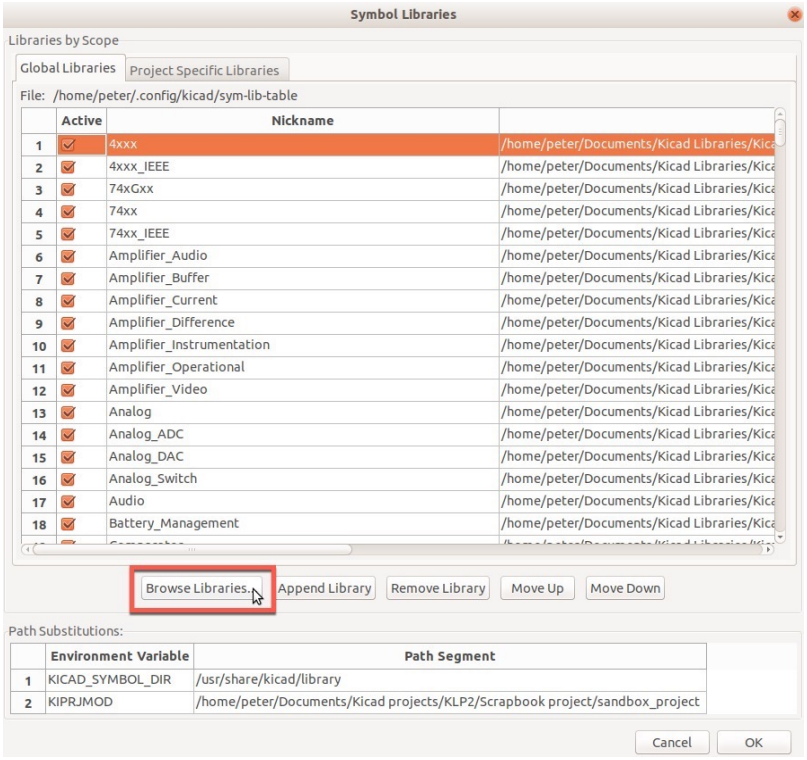


Figure 20.4: The Library Manager.

Using the browser, navigate to the location where the newly downloaded Digikey library files are. Since you want to import all of the symbols to KiCad, multiple select and highlight all files with a '.lib' extension. You can do this by selecting the first file in the list, hold down the shift key, scroll to the bottom of the list and click on the last file (Figure 20.5). When all lib files are selected, click 'Open'.

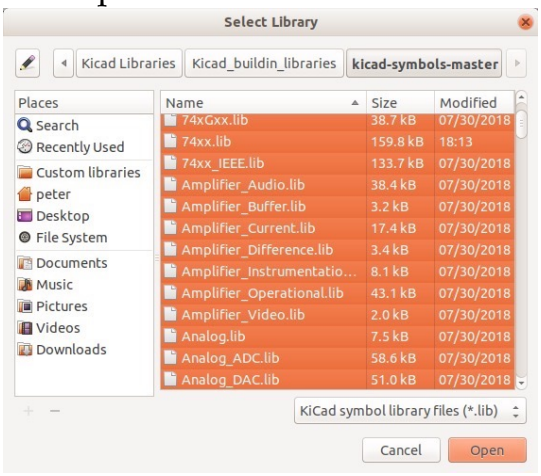


Figure 20.5: Select all '.lib' files.

All new symbol libraries are now in KiCad. You can verify this, first, by scrolling down the list in the Symbol Libraries manager window until you find the libraries with the 'dk' prefix (Figure 20.6). These are all the Digikey symbol libraries that you just imported.

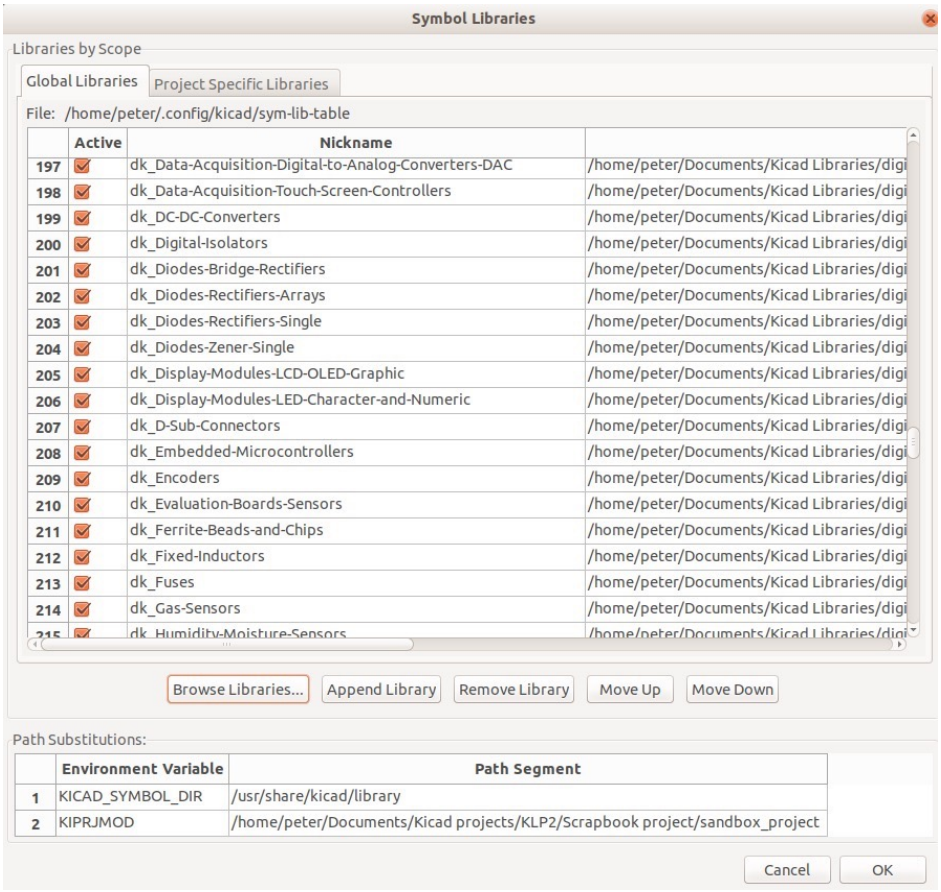


Figure 20.6: The newly imported symbol libraries have a 'dk' prefix.

Click Ok to dismiss this window and try to add a symbol from the new libraries to the Eeschema sheet. Click on the Place Symbol button (right toolbar) and click on the sheet, or type 'A' to bring up the symbol chooser. Scroll in the list, or type 'dk' in the filter to find the Digikey symbols (Figure 20.7).

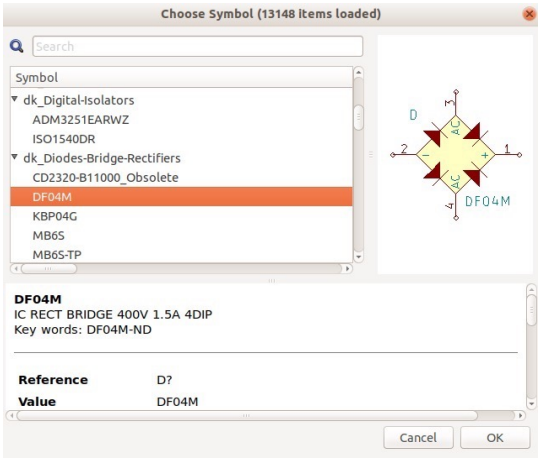


Figure 20.7: One of the symbols in the imported symbol libraries.

Select one of them, and double-click on it to add it to the Sheet. I selected the DF04M rectifier. At this point, you should have the symbol on the Eescema Sheet, like in Figure 20.8.

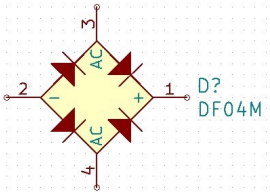


Figure 20.8: One of the new symbols, added to the Eeschema project sheet.

Using the exact same method, you can import individual symbols.

21. Adding a footprint library in Pcbnew

In this recipe, you will learn how to import third-party footprint libraries to your KiCad instance. In the example that follows, we will import the footprint libraries that are published by Digikey. In the previous recipe ('20. Adding a schematic symbol library in Eeschema'), you downloaded, expanded and save the library files to your computer. I assume that you have completed this step. If not, please go to the '20. Adding a schematic symbol library in Eeschema' recipe and complete the steps described there before you continue here.

Start KiCad, and Pcbnew. Open the Footprint Libraries manager through the Preferences menu (Figure 21.1).

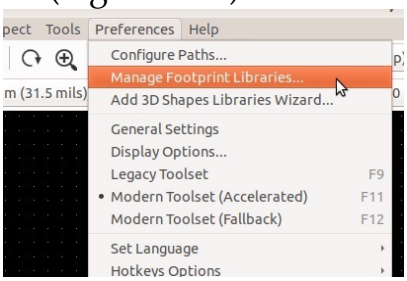


Figure 21.1: Start the Footprint Libraries manager.

In the Footprint Libraries, click on the 'Browse Libraries' button.

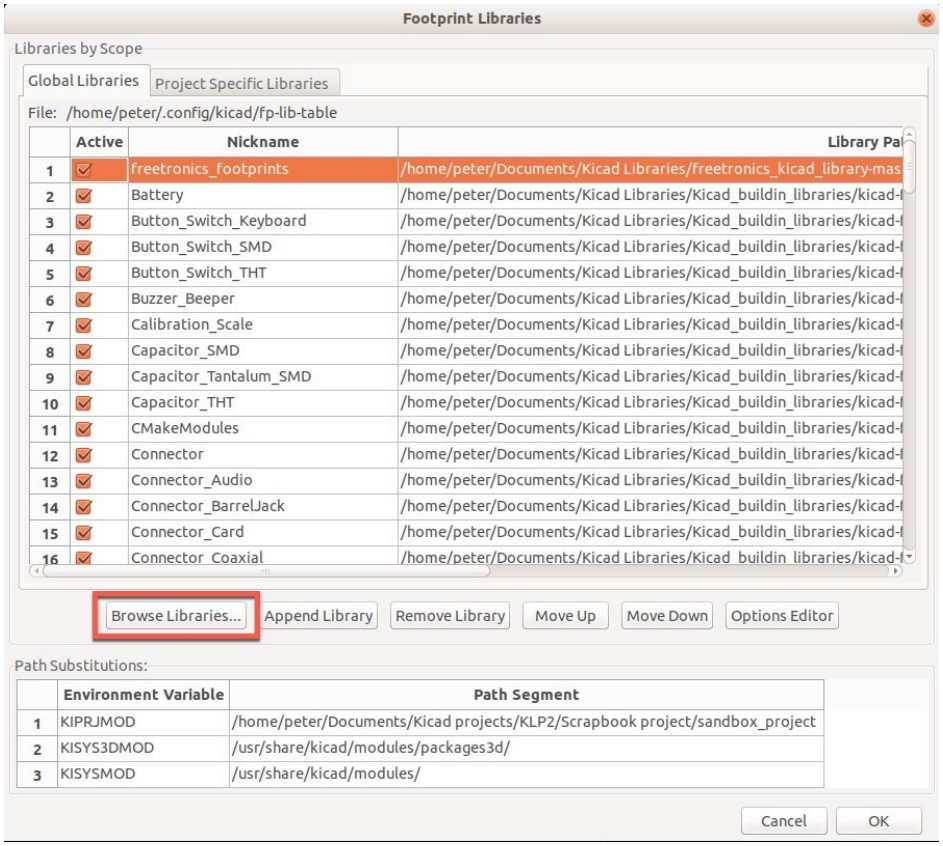


Figure 21.2: The Footprint Libraries manager; click on 'Browse Libraries' to add a new library.

Use the browser to navigate to the location where you stored the Digikey footprints library. This is the folder with the '.pretty' extension, inside the Digikey KiCad library folder. You should only select the .pretty folder, not browse to its contents. Inside this folder are multiple files with the '.KiCad_mod' extension. Each of those files contains one footprint (Figure 21.3).

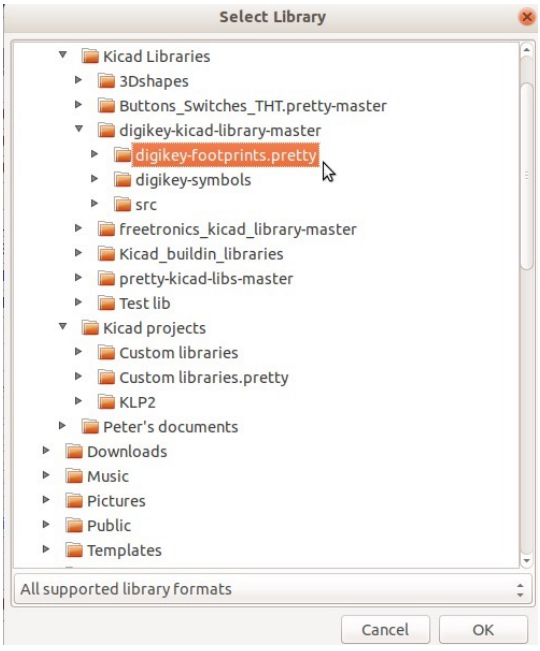


Figure 21.3: Find and select the .pretty folder that contains the footprints.

Click on the .pretty folder to select it, and then click on 'Ok'. Back in the Footprint Library manager, you will see a new row that contains the 'digikey-footprints' library (Figure 21.4). Click Ok to dismiss the manager window. The new library is now ready to use.

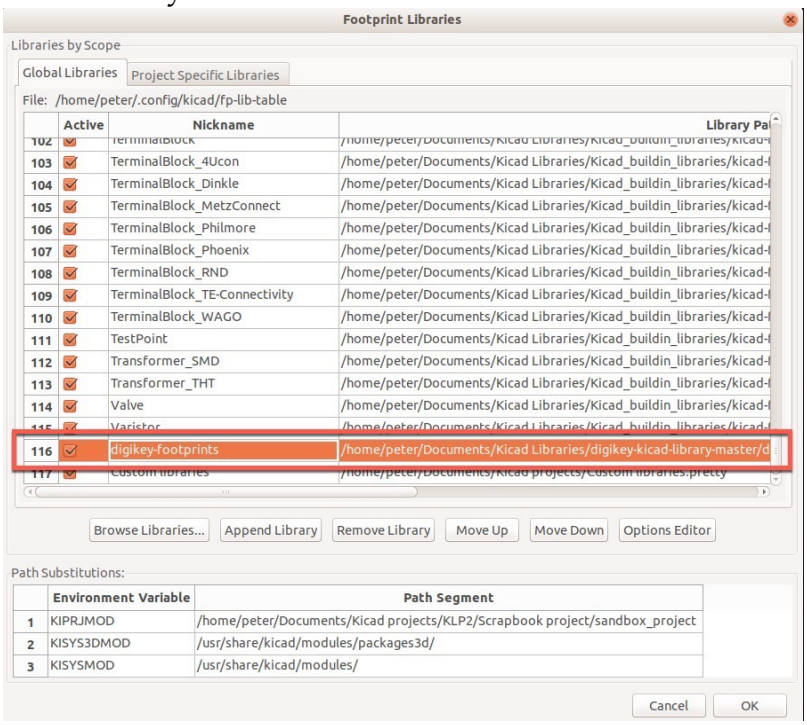


Figure 21.4: The new library appears in the Footprint Libraries list.

Back in Pcbnew, use the 'O' shortcut or select the 'Add footprint' button to add a new footprint. The footprint chooser will come up. Click on 'Select by Browser' button to bring up the browser. Let's find a Digikey footprint that matches the bridge rectifier symbol from the previous recipe. In the Browser, scroll down the left pane to find the digikey-footprints library, and then scroll down to find the DIP4_W7.62mm footprint. Click on this footprint to see it in the right pane (Figure 21.5).

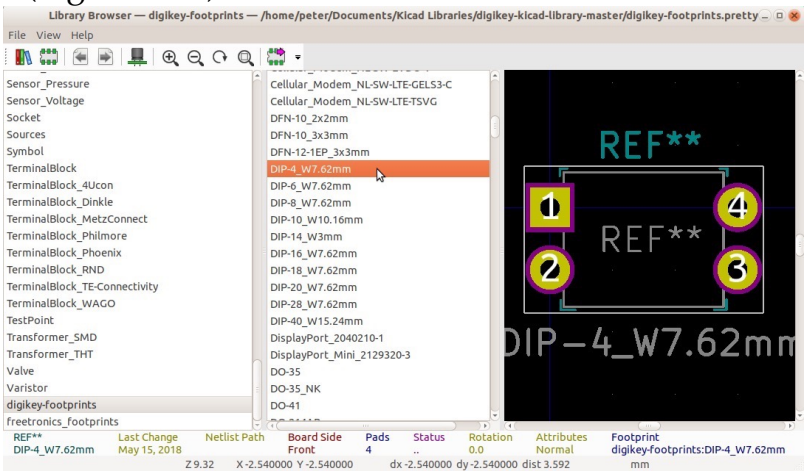


Figure 21.5: Browsing the new Digikey footprints library.

Double-click on it to select it and add it to the Pcbnew sheet. The selected footprint will now appear in the Pcbnew sheet (Figure 21.6).

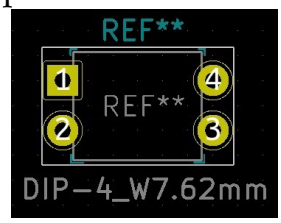


Figure 21.6: A footprint from the Digikey library in the Pcbnew sheet.

22. Using footprint libraries offline

You can opt to download Pcbnew footprints and use them locally instead of accessing the online Github repository. To do that, follow these steps:

1. Go to https://github.com/KiCad/KiCad-footprints
2. Click on the green 'Clone or download' button to download the repository.
3. Extract the downloaded Zip file into a folder of your choice. Consider creating a folder for all KiCad libraries in a reasonable place like the 'Documents' folder.
4. Start Pcbnew
5. Open the Footprint Libraries window, under the Preferences menu
6. Select all libraries and click on 'Remove Library' to delete them. If you have any third-party libraries that you want to keep, leave those unselected.
7. Click on the 'Browse Libraries…' button. The browser will appear.
8. Navigate to the location where you extracted the footprints, and open the folder that contains all the '.pretty' subfolders. Hold the Shift key pressed, click on the first .pretty folder on the list and then on the last one. This allows you to multiple-select all of the folders.
9. With all the .pretty folders selected, click the 'Ok' button to exit the browser.
10. The Footprint Libraries now contains a list of all the footprint libraries saved on your local machine, in the Local Libraries tab (Figure 22.1). This means that you can use these footprints in all your projects.
11. Click Ok to exit the Footprint Libraries window.
12. Test that Pcbnew can access the local footprints. While still in Pcbnew, click on the Add Footprints button from the right toolbar, and then anywhere in the page to add a footprint.
13. In the Choose Footprint window, click on 'Select by Browser.'
14. Use the footprint browser to select libraries and footprints, and ensure that all of them display a footprint representation in the right pane (Figure 22.2).

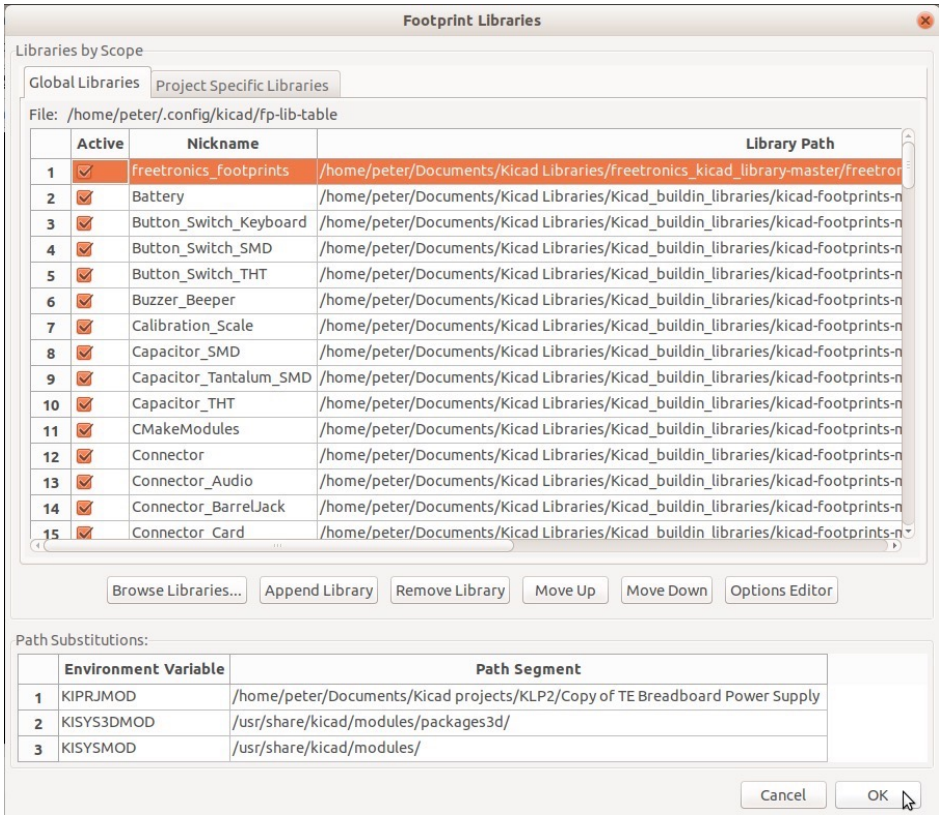


Figure 22.1: These footprints are stored locally.

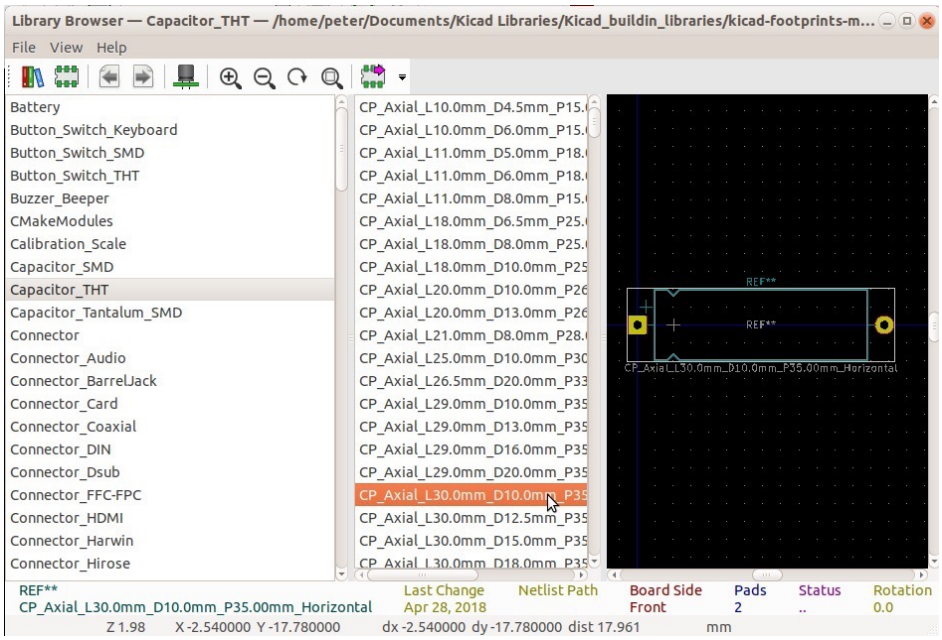


Figure 22.2: Testing local footprints.

As you are now using local footprints, remember to regularly download the latest version of the repository from GitHub, and repeat the process described above so that you always have the newest set of libraries.

23. Using symbol libraries offline

As with footprints, you can opt to download Eeschema symbols and use them locally instead of accessing the online Github repository. To do that, follow these steps:

1. Go to https://github.com/KiCad/KiCad-symbols.
2. Click on the green 'Clone or download' button to download the repository.
3. Extract the downloaded Zip file into a folder of your choice. Consider creating a folder for all KiCad libraries in a reasonable place like the 'Documents' folder.
4. Start Eeschema.
5. Open the Manage Symbol Libraries window, under the Preferences menu.
6. Select all libraries and click on 'Remove Library' to delete them. If you have any third-party libraries that you want to keep, leave those unselected.
7. Click on the 'Browse Libraries…' button. The browser will appear.
8. Navigate to the location where you extracted the footprints, and open the folder that contains all the '.lib' subfolders. Hold the Shift key pressed, click on the first .lib folder on the list and then on the last one. This allows you to multiple-select all of the folders.
9. With all the .lib folders selected, click the 'Ok' button to exit the browser.
10. The Symbol Libraries now contains a list of all the footprint libraries saved on your local machine, in the Local Libraries tab (Figure 23.1). This means that you can use these footprints in all your projects.
11. Click Ok to exit the Symbols Libraries window.
12. Test that Eeschema can access the local footprints. While still in Eeschema, click on the Place Symbol button from the right toolbar, and then anywhere in the page to add a symbol.
13. In the Choose Symbol window, expand the tree and click on any item. For each item you click on, you should see the symbol in the right pane (Figure 23.2).

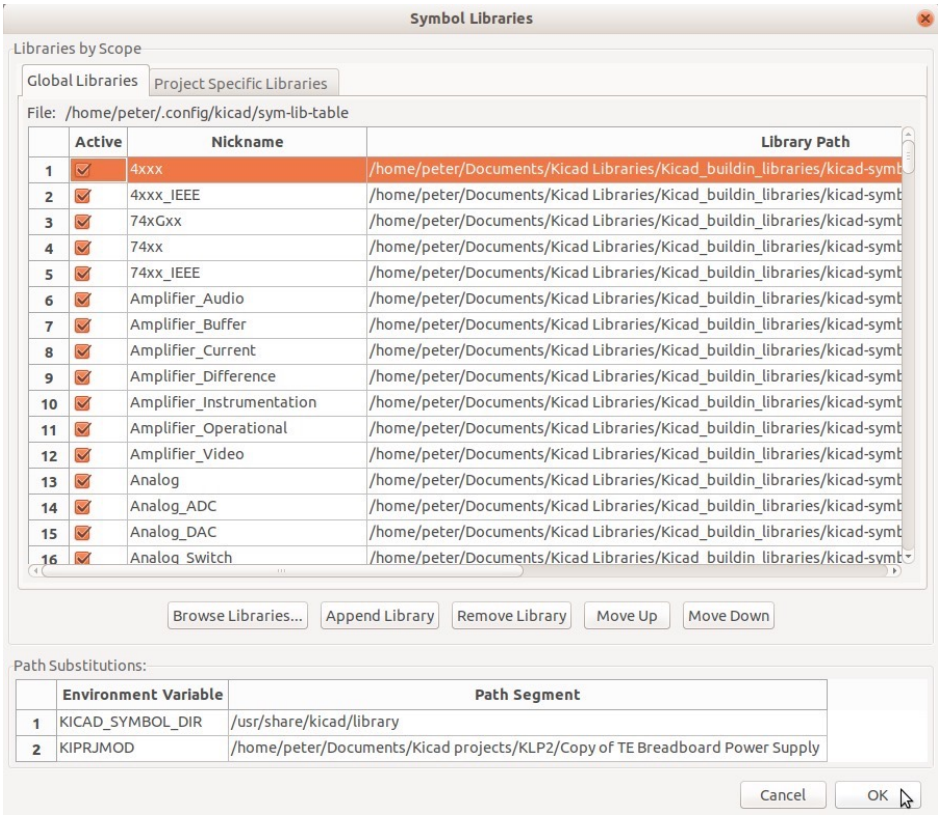


Figure 23.1: These symbols are stored locally.

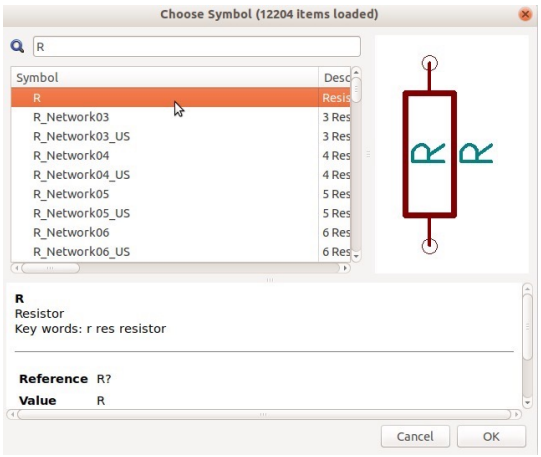


Figure 23.2: Testing local symbols.

As now you are using local symbols, remember to regularly download the latest version of the repository from GitHub, and repeat the process described above so that you always have the latest set of libraries.

24. Create a keep-out zone

A keepout zone is an area on a PCB layer that is marked to be free of footprints or traces. You may want to create keepout zones in response to mechanical, electrical or radio specifications. For an example of a case where a keepout zone is necessary, please go to the section on Routing, in Part 3.

To create a keepout area follow this process:

1. In Pcbnew, select the 'Add Keepout Areas' tool from the right menu bar. 
2. In the page, click on the location where you would like the first edge of the keepout area to be. The dialog box in Figure 24.1 will appear.
3. Select the layer for the keepout, the elements that you want to exclude from this area, and how the outline should look, then click 'Ok' to start drawing.
4. Draw the outline of the keepout area. The outline should be fully enclosed. To finish drawing, double-click.
5. To test that the keepout area is working, place one of the forbidden elements inside of it, and do a Design Rules Check. For example, I placed two vias in the keepout area, and I started the DRC. The DRC will contain 'Via inside a keepout area' messages, and the offending vias are marked with an arrow (Figure 24.1).

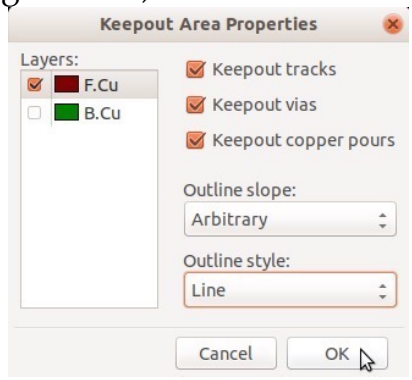


Figure 24.1: The Keepout Area Properties window.

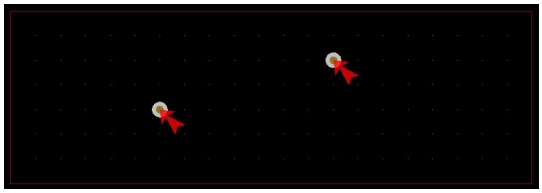


Figure 24.1: The keepout area is defined by a thin red line; the DRC has detected two vias that are not allowed in it.

25. Creating copper fills

In Part 3, you learned about copper fills, which is the fifth step of the PCB layout process. In this recipe, you will learn how to create a copper fill in KiCad. We'll be using the breadboard power supply board project for this demonstration. Before you create a copper fill, you must complete the routing of your board. In Figure 25.1, you can see the example board fully routed. The next step is to create a copper fill for the ground level (also known as 'ground plane').

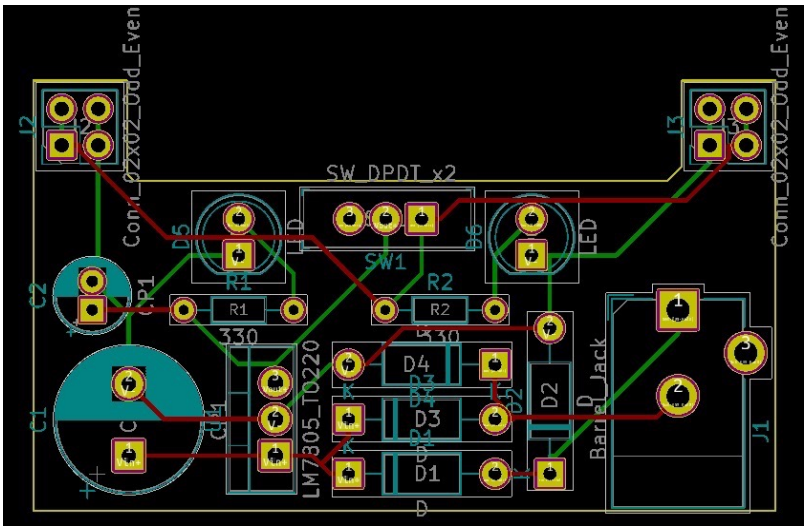


Figure 25.1:The fully routed board.

In Pcbnew, follow this process:

1. From the left toolbar, click on the 'Show filled areas in zones' button  . This will allow you to see the filled zone you are about to create. If this button is not pressed, the filled zones will be invisible. This is a useful option to use when you want to examine your board by reducing the amount of visual clutter.

2. From the right toolbar, click on the 'Add filled zones' button  .

In this example, you will create a ground plane. From the Layers Manager, click on 'B.Cu'.

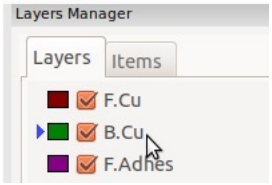


Figure 25.2: Select the back copper layer.

3. To start drawing the outline of the copper fill, place the cursor on the board location where you would like the first corner to be, and click. I am starting my drawing on the top left corner of the J2 connector.

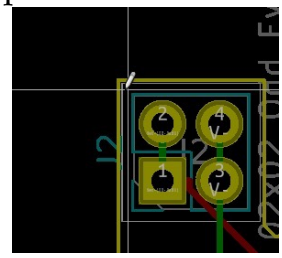


Figure 25.3: Click at the position you want to start drawing a copper fill.

4. The Copper Zone Properties window will appear. Most of the defaults are appropriate for this project. Draw your attention to selecting the correct layer ('B.Cu' since we want to create a ground plane in the bottom layer), the connected Net (in his example, the Ground net name is 'V-'), and the clearance of the copper fill to other nets (0.508 mm). Click 'Ok' to accept these settings, and continue drawing the outline of the copper fill.

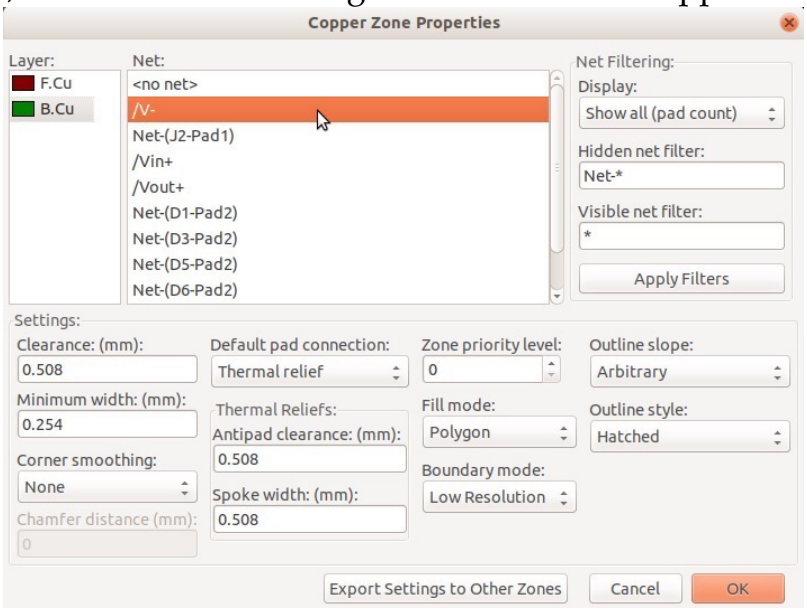


Figure 25.4: Connect the copper fill to a net.

5. Draw the outline of the copper fill. Click to create an edge, and try to draw each edge of the polygon as close as you can to the border to the board. To complete the outline, double-click on the exact location where you started the drawing. Figure 25.5 shows my example board on completion of the ground plane.

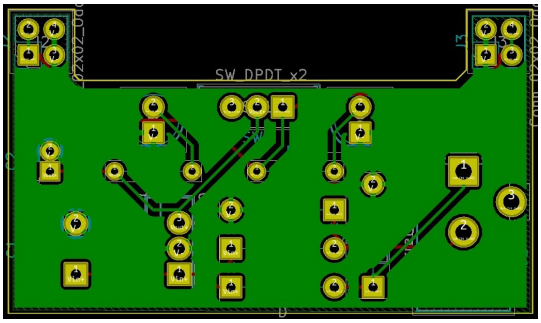


Figure 25.5: The completed ground plane in the bottom layer.

You can use the 3D viewer to inspect the new copper fill. Zoom in to one of the ground pads and notice that it is connected to the ground plane using thermals (Figure 25.6).

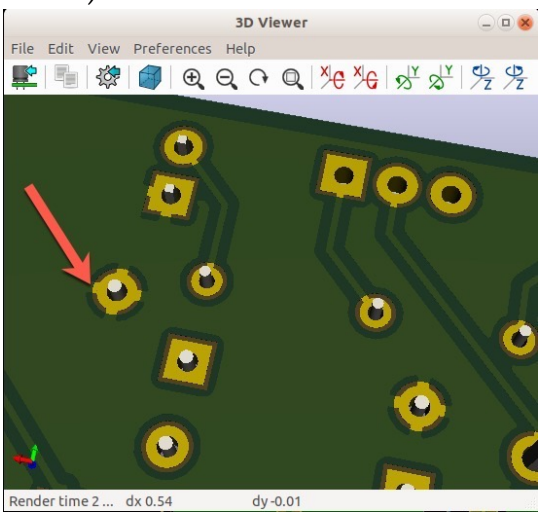


Figure 25.6: A partial 3D view of the ground plane, showing thermal traces between pads and ground plane.

If you want to re-create a copper fill, right click on the zone's border and click 'delete' from the context menu.

You can make the copper fill disappear (but not deleted) by clicking on the 'Do not show filled areas in zones' button from the right toolbar ( ).

26. How to calculate the width of a trace

KiCad includes a calculator that you can use to precisely work out what the width of a track should be based on various parameters, like the current you wish to convey through the trace, its total length, and the maximum temperature rise when that current is actually flowing through it. You can use this calculator to find out the minimum trace width, or you can rely on your experience and choose a width that is much larger than the standard width of signal traces.

To use the calculator, open the KiCad launcher window and click on the calculator icon (Figure 26.1).

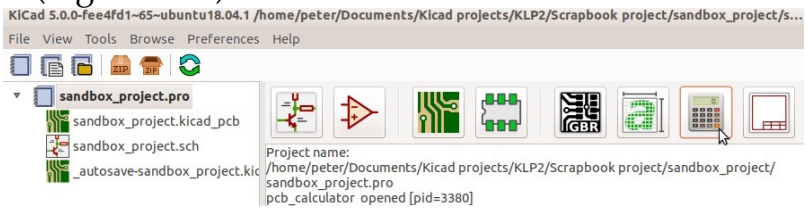


Figure 26.1:The calculator is available through the KiCad launcher window.

The calculator app actually contains multiple calculators. One of them is the Track Width calculator. Select it by clicking on its tab. Fill in the values that best describe your power track requirements. For a typical Arduino gadget, the values that you see in Figure 26.2 are reasonable. I have only altered the conductor length value to 30mm to better match the power trace length of one of my PCB projects. I tend to overshoot these values to ensure that the trace width that the calculator returns can comfortably cover the requirements.

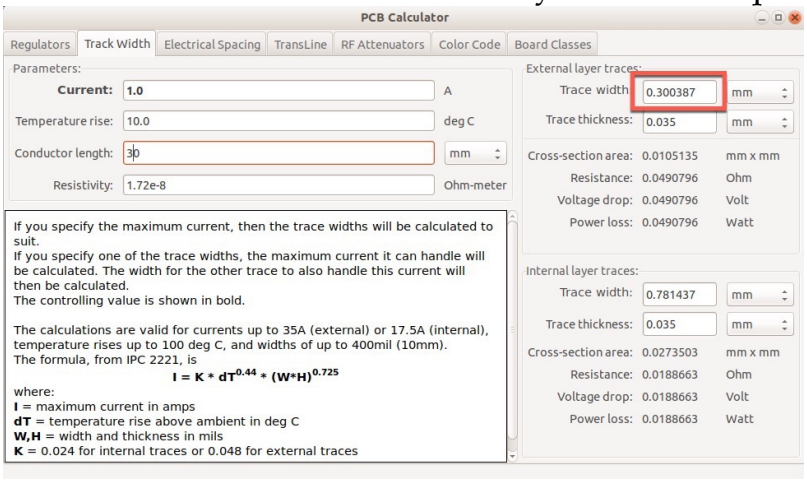


Figure 26.2:The Track Width calculator.

At the top right corner of the calculator, there is a field where you can provide the trace thickness. This is a value that you don't have control over and is defined by the manufacturer's specifications (some manufacturers allow you to select the weight of your copper trace, but for simplicity let's assume here that this is fixed). The default value, 0.035 mm, seems to be an industry standard. Manufacturers typically make their boards with that trace thickness. To be sure, either search your preferred manufacturer's website for their trace thickness or ask them.

As you type in the parameters, the calculator returns the suggested trace width. In the example of Figure 26.2, the suggested width is 0.30 mm.

26. Custom Global Design Rules and changing the width of a trace

This recipe is updated to cover Kicad 1.5 and newer.

When you start a new project in Pcbnew, the only trace width available is 0.25 mm (9.84 mils). If you need to create a trace with a different width, you will need to create a custom track width for that value.

For example, in the Trace Width Calculator recipe, I calculated that the minimum width for a power trace that can convey 1 A over 30 mm with a maximum of 10 degrees Celsius temperature rise is 0.30 mm. In Pcbnew, there is drop-down menu from where you can manually select the desired width of a track, titled "Track".

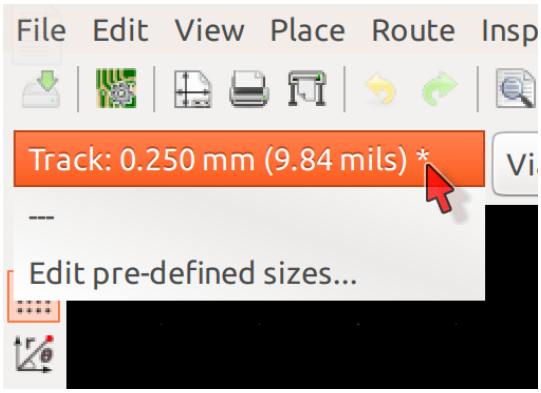


Figure 26.1: The Track drop-down menu; there is only one width available by default.

By default, this drop-down offer a single track width option: 0.250 mm. In this recipe, you will learn how to add '0.30 mm' as a track width option in the top toolbar drop-down menu. You can also use the exact same way to add custom Via sizes.

In Pcbnew, start by bringing up the Board Setup window. You can do this in three ways:

1. Click on File —> Board Setup
2. Click on the Track drop-down, and then "Edit pre-defined sizes…".
3. Click on the Via drop-down, and then "Edit pre-defined sizes…".

In any case, you will see the same result (Figure 26.2) .

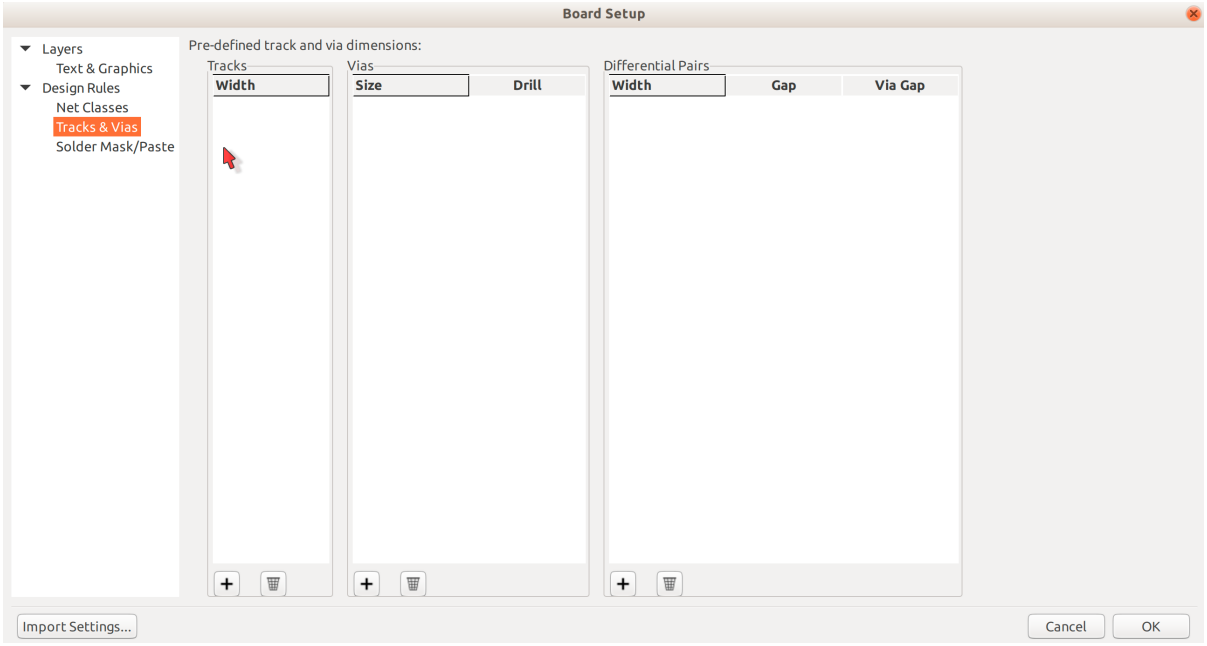


Figure 26.2: The Board Setup window with the Tracks & Vias pane selected.

Select the Tracks & Vias pane, under Design Rules. There, you will see three columns: Tracks, Via and Differential Pairs. There's a "+" button at the bottom of each column.

Let's add a few custom widths and Vias. To add each one, click on the "+" button and type in the numbers in the relevant row.

In Figure …, you can see my custom tracks and vias.

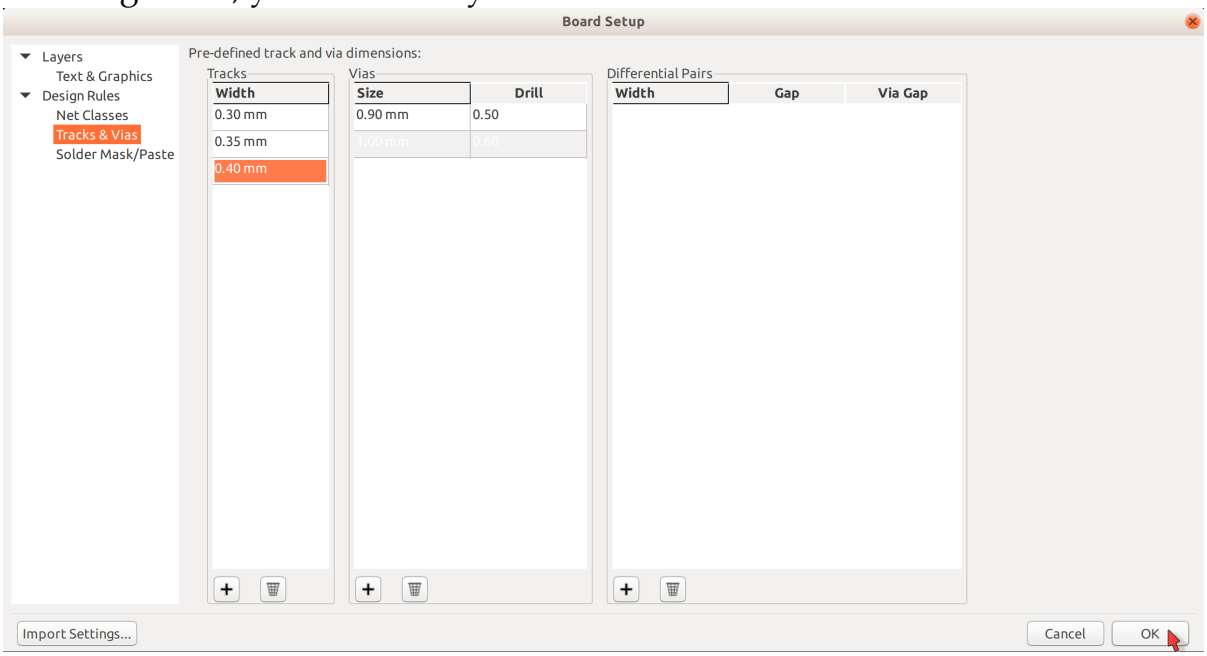


Figure 26.3: The Board Setup window with the Tracks & Vias pane selected; I have added several new custom track widths and vias.

Click 'Ok' to commit the changes, and confirm that the Track and Via drop-down menus show the new values (Figure 26.4).

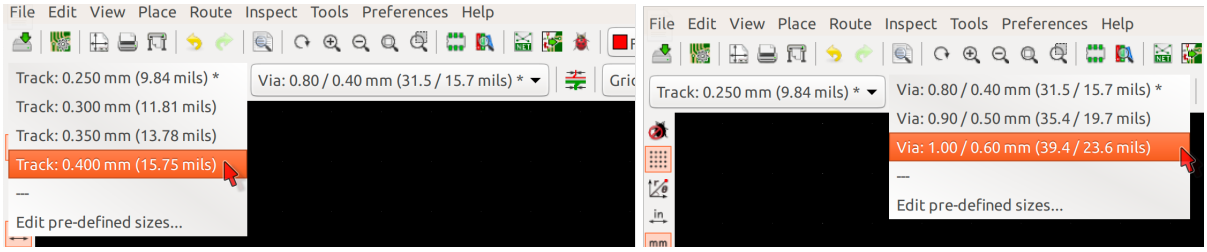


Figure 26.4: The Track Width drop-down contains a custom value.

Let's test that you can create a trace with the custom with. First, select the '0.30 mm' option from the track width drop-down menu. Then, use the 'X' shortcut or click on the Route Tracks button from the right toolbar to enter the track drawing mode. Draw a new track in the Page. Type 'Esc' to end the drawing, and choose the '0.25 mm' option from the track width menu. Draw another trace and compare its width to the first. I have also added a third trace, with 0.40 mm width. Notice that the traces have a different width?

Figure 26.5: Three traces with different widths.

You can change the width of a track at any time by using the trace's properties window. Place your mouse on the trace, type the 'E' shortcut (for 'Edit') and select the desired width from the Width drop-down menu (see Figure 26.6).

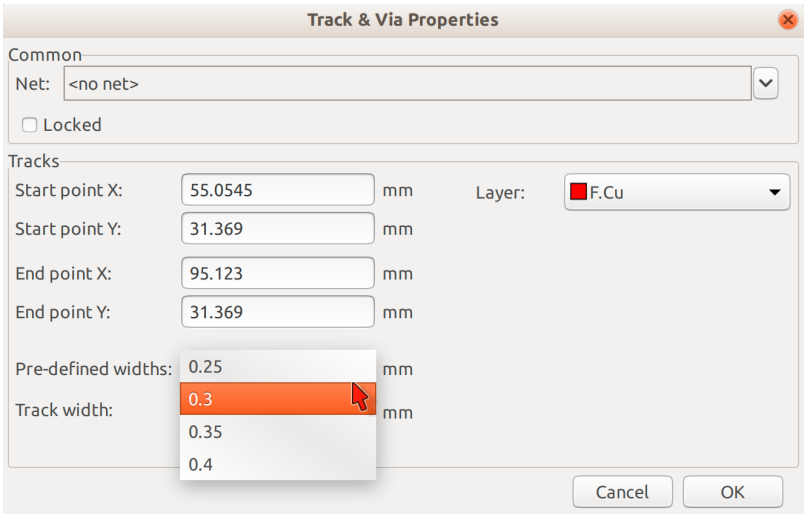


Figure 26.6: How to change the width of a track via the Properties window.

Click OK to make the change effective.

27. Create custom net design rules

This recipe is updated to cover Kicad 1.5 and newer.

Eeschema allows you to give custom names to any net. This makes your schematic more readable since you will be able to read its name and infer its purpose quickly. Named nets also allow you to speed up routing by defining design rules specific to named nets. For example, if you have a custom net design rule for nets named 'Power', this rule will be applied automatically, saving you the additional effort of manually setting things such as trace width and via diameters.

In this recipe, you will learn how to create a custom design rule for a net named 'Power'. The net name is arbitrary but should match the net names that you defined in Eeschema in order for the rule to be applied in Pcbnew.

In Pcbnew, open the Board Setup window from the File menu. Click on Net Classes under Design Rules. Click on the '+' button to add a new net class. At the prompt, give the new class a name, like 'Power'. A new row will appear, with default values for the various attribute fields (Figure 27.1). To update these values, click inside the field that you want to edit, type in the new value and hit Enter. In the example of Figure 27.1, I have set 0.30 mm for the Track Width, since this is the value that the calculator suggested as the minimum width. I adjust the rest of the values accordingly. For vias, I chose 1 mm for the diameter and 0.6 mm for the drill to closely match the increased power trace width. I don't plan to use micro-vias (uVia) so I left their values unchanged. For the differential pairs, I matched their width and gap with the single trace width and clearance.

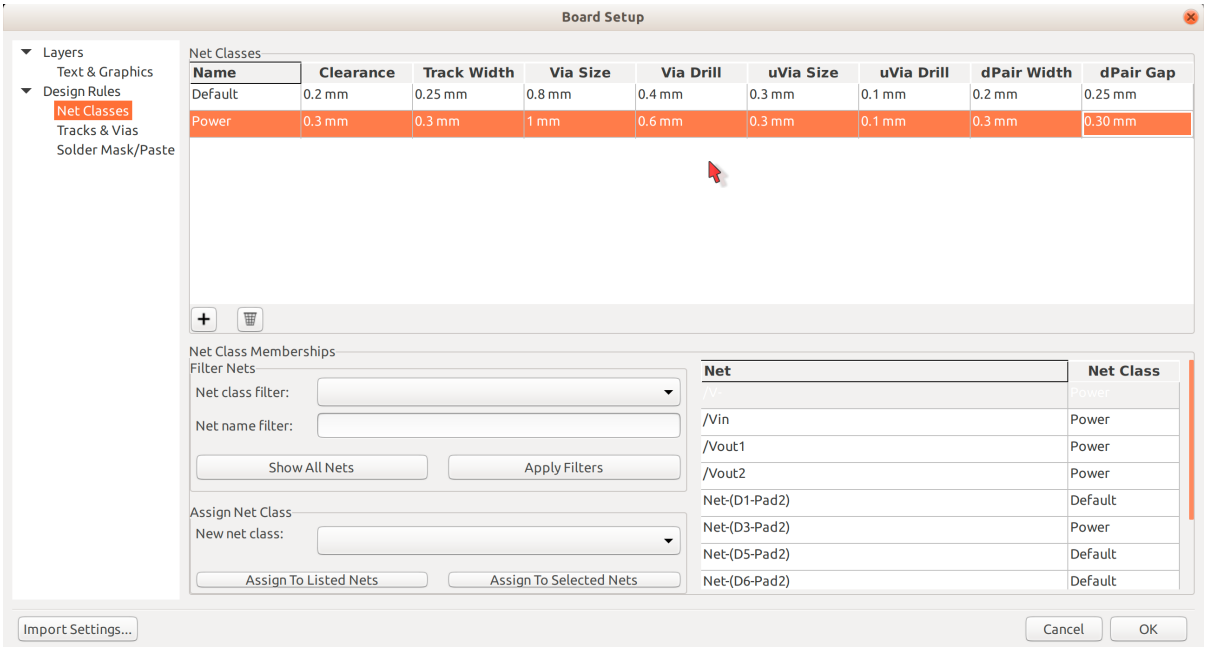


Figure 27.1: The new Power class.

With the new rule set for the Power class, when you draw the trace for a net named 'Power', the custom design rules will be applied. The Power traces will have a width of 0.3 mm (instead of 0.25 mm), and a via diameter of 1 mm (instead of 0.8 mm).

You can assign nets to a net class by using the Net Class Membership segment widgets of the Design Rules Editor. From the left side of the Net Class Membership segment, choose the net class you want to assign nets to. In the example of Figure 27.2, I have selected the Power class.

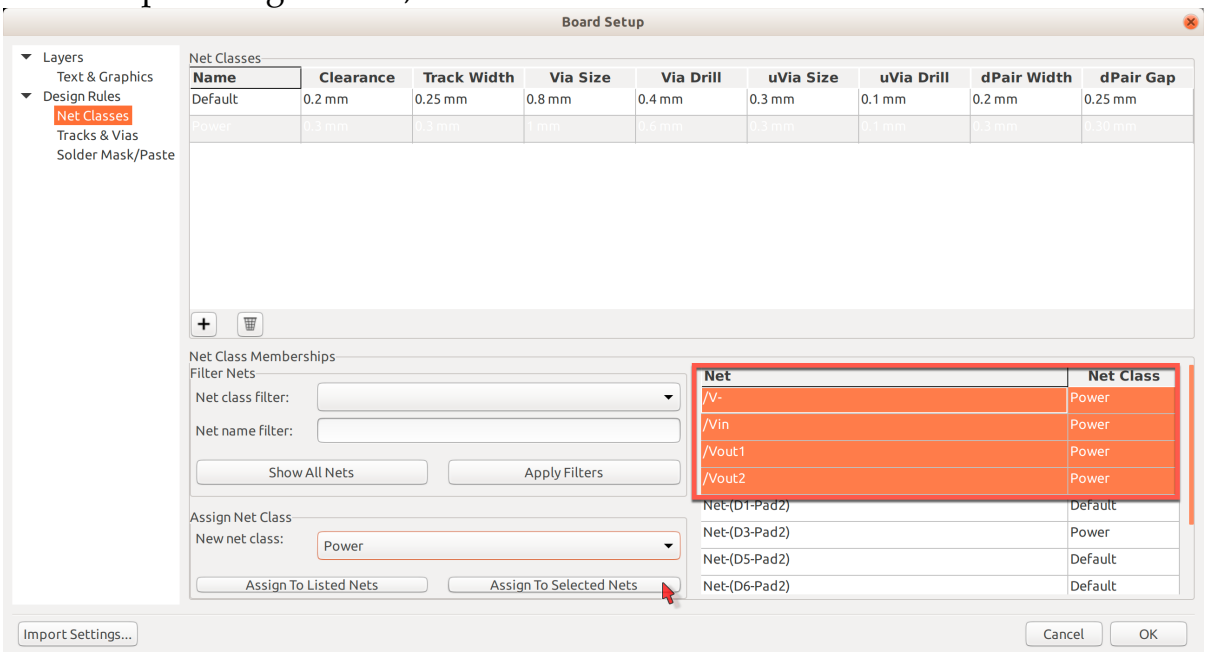


Figure 27.2: Assigning nets to a Net Class.

From the list of nets in the right side of the same segment, select the nets that you want to assign to the net class. I have selected the three nets you can see 'boxed-in' in Figure 27.2. You can select multiple nets by holding down the Shift key and clicking on a row. In the Assign Net Class box, select the Power class from the drop-down menu, and then click on “Assign To Selected Nets”. Click “OK” to dismiss the window and commit the changes. These nets are now members of the Power net class and will assume that Class’s attributes when you create their traces

28. How to add silkscreen text and simple graphics

In KiCad, adding silkscreen text and simple graphics to your boards is very easy because there are tools dedicated to this task. In this recipe, you will learn how to do it.

I will demonstrate using the breadboard power supply board (Figure 28.1). This board is already routed and contains a ground plane which I have made invisible in order to make it easier to work with the silkscreen.

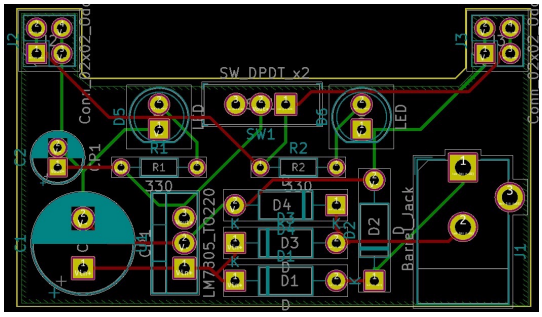


Figure 28.1: Let's add some custom silkscreen elements to this board.

The board already has several silkscreen elements on it, courtesy of the footprints we have used on it. The silkscreen elements are marked with a light-blue colour in Figure 28.1, and with white in the board's 3D representation (Figure 28.2).

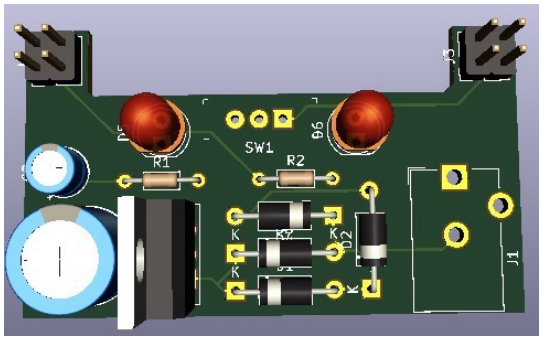


Figure 28.2: The existing silkscreen elements are shown in white in the 3D view of the board.

In the right bottom corner of the board, you can see the outline of the barrel connector footprint. There is a white line that marks the edges of the footprint, and the name of the footprint 'J1'.

Let's say that you want to make the text 'J1' to not be printed in the manufactured board. To do that, you must edit the text properties and make it invisible. Place your mouse cursor over the text, and type the 'E' shortcut (for 'Edit'). The Properties window in Figure 28.3 will appear.

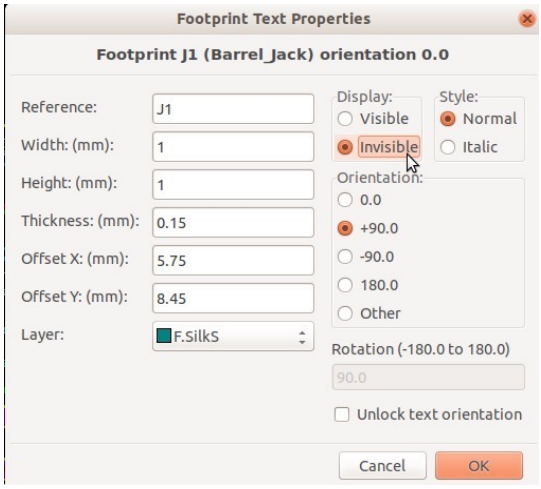


Figure 28.3: The Text Properties window.

Click on the 'Invisible' radio button to select it, and click on the Ok button. The 'J1' text should no longer be visible on the board, or in its 3D representation. To bring it back, either type 'Ctr-Z' (Windows, Linux) or 'Cmd-Z' (Mac OS) to undo the change. Or, edit the barrel connector footprint (place your mouse pointer on the footprint and type 'E'), then click on the Reference text 'Edit' button to bring up the text properties for the footprint reference text, and choose 'Visible' from the Display box.

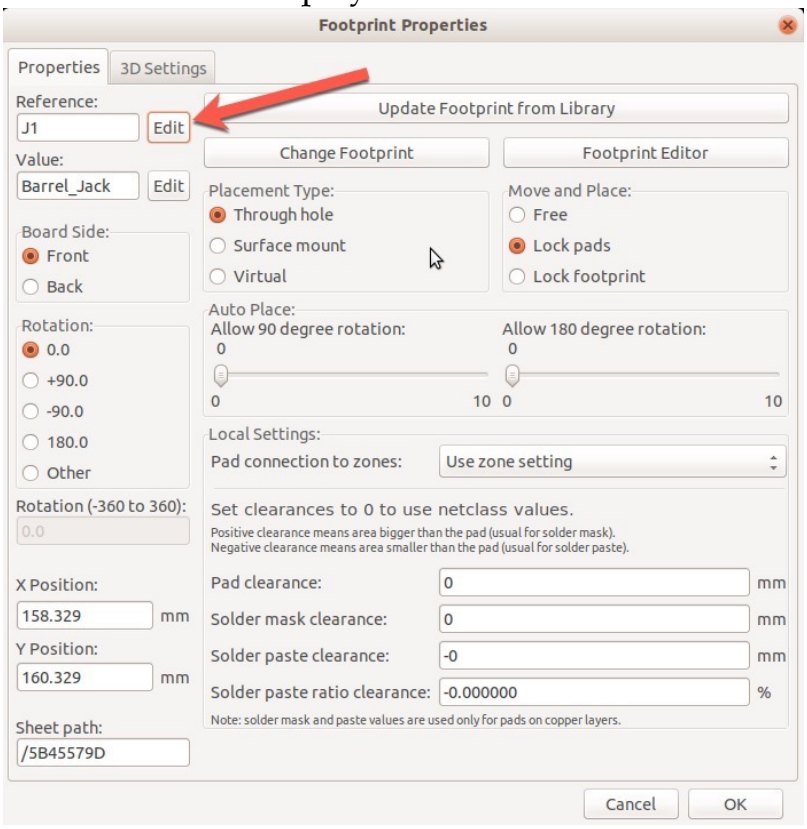


Figure 28.4: You can access the footprint reference text from the footprint properties.

Let's continue to add custom text and simple graphics to this board. First, because we want to insert silkscreen elements into the top layer of the

board, select 'F.SilkS' from the Layers Manager. If we wanted to do the same to the bottom layer, we would select 'B.SilkS' (we will do this later because there is an additional consideration for doing so).

To add text, click on the Text tool from the right toolbar (Figure 28.5).

Figure 28.5: These tools are used to create silkscreen elements.

Now click on the location on the board where you would like the text to appear. I will place mine over the barrel connector footprint. The Text Properties window will appear (Figure 28.6). Type your text in the text box. You can also control the size and other attributes of your text (I encourage you to experiment with them).

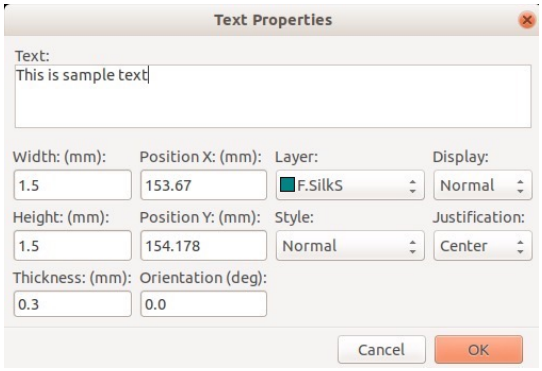


Figure 28.6: Text properties.

With the text and properties as it appears in Figure 28.6, click on Ok. The silkscreen should look like the example in Figure 28.7.

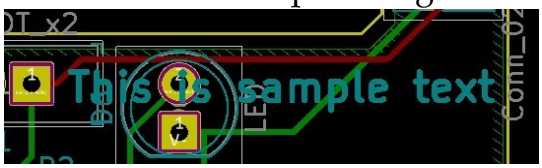


Figure 28.7: The new text appears in a single line.

The text appears in a single line, and it crosses with other elements, like the LED footprint and other text. It would be better to have this text appear in three lines, justified to the left, and moved to the right edge of the board. To do that, bring up the Properties window again (place your mouse pointer over the text and type 'E'), and break the text into three lines, as you can see in

Figure 28.1. Also, change the justification to 'Right'. You can reduce the text size by reducing the Width and Height to 1.0 mm.

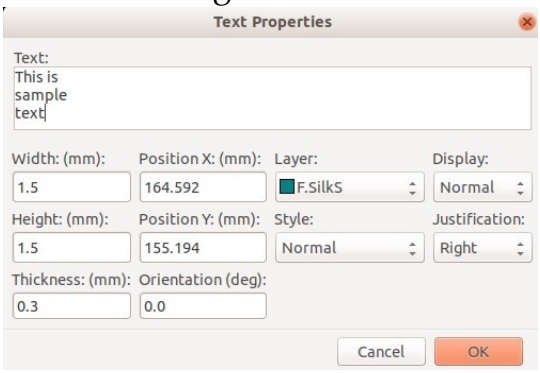


Figure 28.8: Text now appears in three lines, justified to the right.

Click Ok to commit the changes, and notice how this text appears on the board (Figure 28.9).

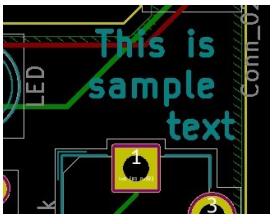


Figure 28.9: The text appears in three lines, justified to the right..

If you need to move the text, use the 'M' shortcut ('Move').

Next, let's add a box around the text. We'll use the polygon tool for this. If you wanted to draw a circle you would use the circle tool, and if you wanted to draw an arc you would use the arc tool. While you are still working on the F.SilkS layer, click on the polygon tool (the first from the top in Figure 28.5), and click on the four corners around the text to create the outline. Double click to exit the drawing mode. The result should look like the example in Figure 28.10.

Figure 28.10: Added a box around the text.

In 3D, the board looks like the example in Figure 28.11.

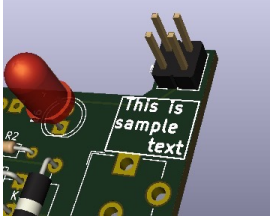


Figure 28.11: Added a box around the text, in 3D.

Now, let's say that you'd like to print the same text and box in the bottom silkscreen layer. Start with the text. Place your mouse pointer on the text and type 'E' to show the text properties. Change the Layer to 'B.SilkS' and the Display to 'Mirrored' (Figure 28.12).

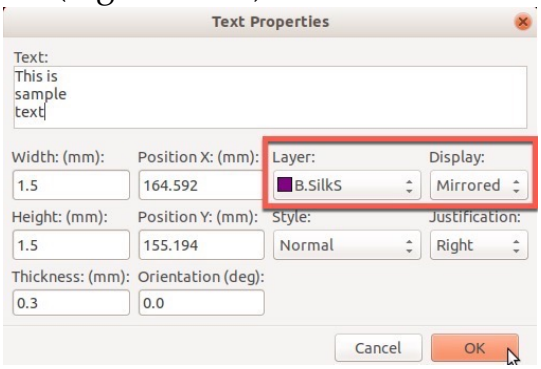


Figure 28.12: Moving the text to the bottom silkscreen.

Because the text is now placed in the bottom silkscreen, it is necessary to display it in mirrored orientation. Click Ok, and then used the 'M' key to move the text in position. Repeat the same process for each of the four lines that make up the box (Figure 28.13).

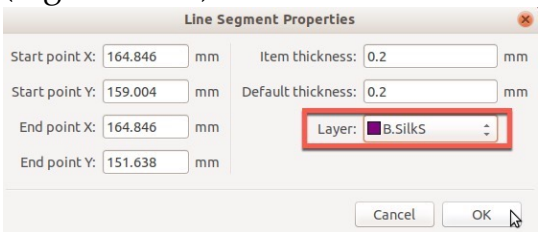


Figure 28.13: Moving the line to the bottom silkscreen.

You can see the result and its 3D representation in Figure 28.13.

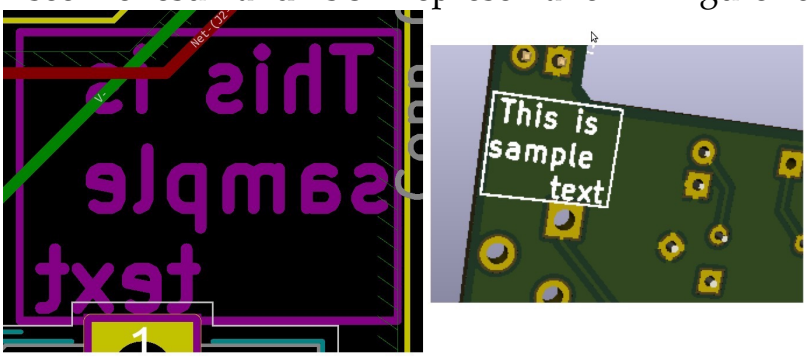


Figure 28.13: The text and box appear in the bottom silkscreen.

The bottom silkscreen elements appear in purple. In the 3D view, turn the board around to see the bottom layer, where the silkscreen elements appear in white.

To learn how to add a custom logo to the silkscreen, please continue with the next recipe.

29. How to add a custom logo to the silkscreen

To add your logo or other graphics on your PCB, you first need to determine the dimensions of the area on the PCB where the logo will be placed. An important consideration is that the graphics file must be a bitmap (BMP or PNG), and monochrome. For the end result to be a crisp and high-quality graphic, you must also use a file with a high DPI (Dot Per Inch) count, at least 500 DPI. This is something I discovered experimentally.

In this recipe, you will learn how to add a logo to your PCB. KiCad provides a tool that allows you to convert a bitmap file into a footprint that can be imported into Pcbnew. Because there is a plethora of image manipulation programs and this book is not about image manipulation, I leave it up to you to create a suitable image file.

For this example, I will use a BMP file that has a size of 1140 x 448 pixels to create a footprint that will fit in an area that is around 10 mm wide on the PCB. The bitmap file is far larger than what is necessary for our 10 mm end result, but after a lot of trial and error I have discovered that including more data in the raw file (a larger file) will result in a silkscreen graphic with better detail, that looks crisp and professional.

From this starting point, follow this process to create the graphics footprint and add the logo to your PCB in the front silkscreen:

1. Open the main KiCad window and click on the 'Bitmap to Component Converter' button.

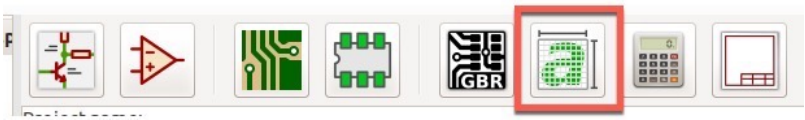


Figure 29.1: The 'Bitmap to Component Converter' button.

2. Click on the 'Load Bitmap' button and import the BMP file.

3. The image of the logo will appear on the left pane of the tool. Click on the 'Black&White Picture' tab.

4. The Bitmap to Component Converter is a very simple tool that allows you to do some basic manipulation of a bitmap image file. The most important modification you will need to do here is to match the size of the footprint you are about to create, with the area in which you want to place the logo. In this example, I would like to place the logo in an area that is around 10 mm in width. I'd like the height to be proportional to the width. In Figure

29.2, find the Bitmap Info box at the top right corner of the Converter. In it, you can see the raw dimensions of the image. The only item that you can edit is the resolution. Increase or decrease the resolution until the size matches your requirements. In my case, I had to increase the resolution to 2500 DPI (height and width) to get my images to around 10 mm. In fact, 11.6 mm across is still fine for the available space so I left the resolution at 2500 DPI.

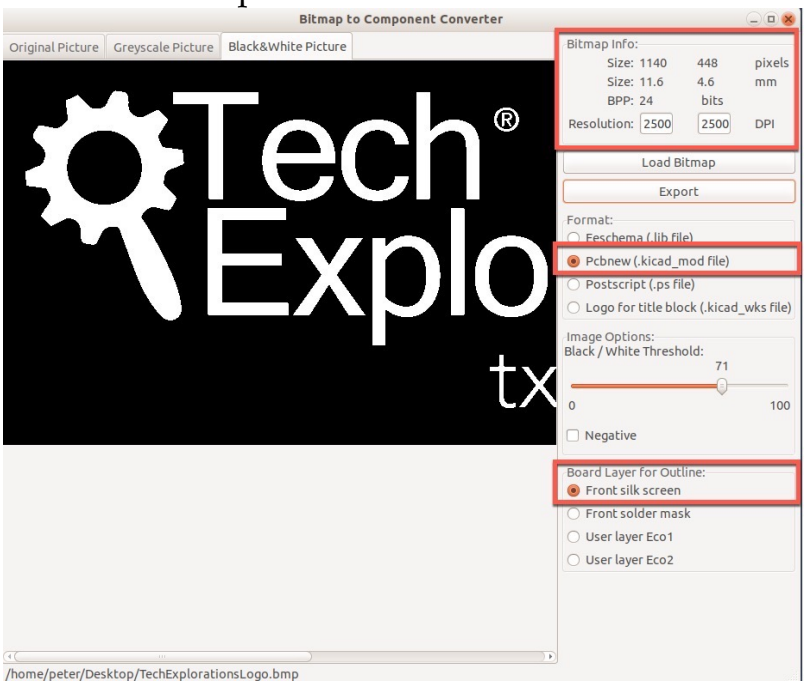


Figure 29.2: The Bitmap to Component Converter.

5. Ensure that Pcbnew and 'Front silkscreen' are selected, and then click on Export.

6. 'Export' will prompt you for a location for the new footprint library you are about to create. If you want to use this logo among all your projects, create and select a new folder outside of your current project folder. In my case, I created a new folder inside my current project directory, named 'ProjectLibraries', gave this new library a suitable name 'TechExplorationsLogo_2500DPI.KiCad_mod', and saved the file. You can now close the Converter.

7. Go into Pcbnew, and click on Preferences, ManageFootprintLibraries. Before you use the new footprint you must import it. You can choose to import the new library for the current project only, or for all projects. In either case, click on the preferred tab, and then on the 'Browse Libraries' button.

8. Navigate your file system and click on the folder that contains the new library. Click Ok to select it. In Figure 29.3 you can see the imported library in my 'Project Specific Libraries' tab.

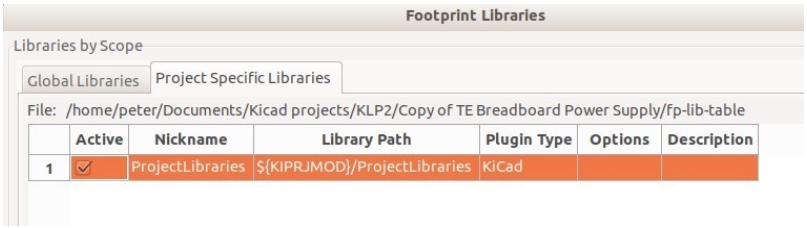


Figure 29.3: The new footprint library is specific to this project.

9. Select the front silkscreen 'F.SilkS' from the Layers Manager.

10. Press the 'O' key (to add a footprint) or click on the 'Add Footprints' button from the right toolbar, and then click on the location on the board where you would like to add your logo.

11. The 'Chose Footprint' window will appear. Click on the 'Select by Browser' button. The Library Browser will appear (Figure 29.3).

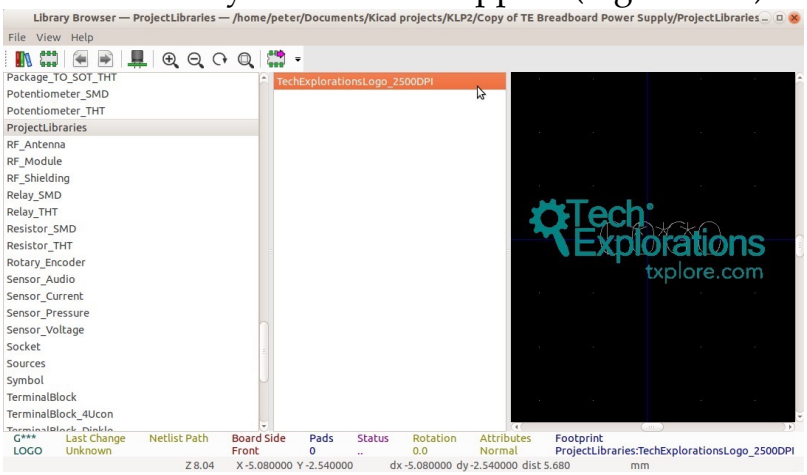


Figure 29.3: Find your new library in the Library Browser.

12. Look for your new library that contains your logo footprint. The library's name is the same as the name of the folder in which the logo footprint is.

13. When you find the library and footprint, double-click on it to select it and drop it on your board.

14. The logo will now appear in the front silkscreen layer of your board. You can verify that it looks as you want it by opening the 3D viewer (Figure 29.4).

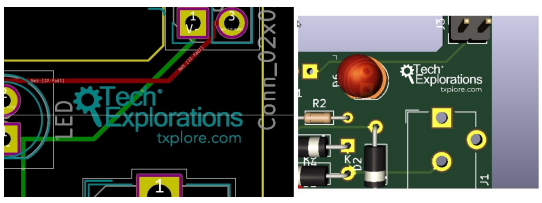


Figure 29.4: The logo appears on the front silkscreen.

What if you wanted to place this logo on the back silkscreen? All you have to do is to edit the logo footprint properties and select 'Back' as the board side (Figure 29.5).

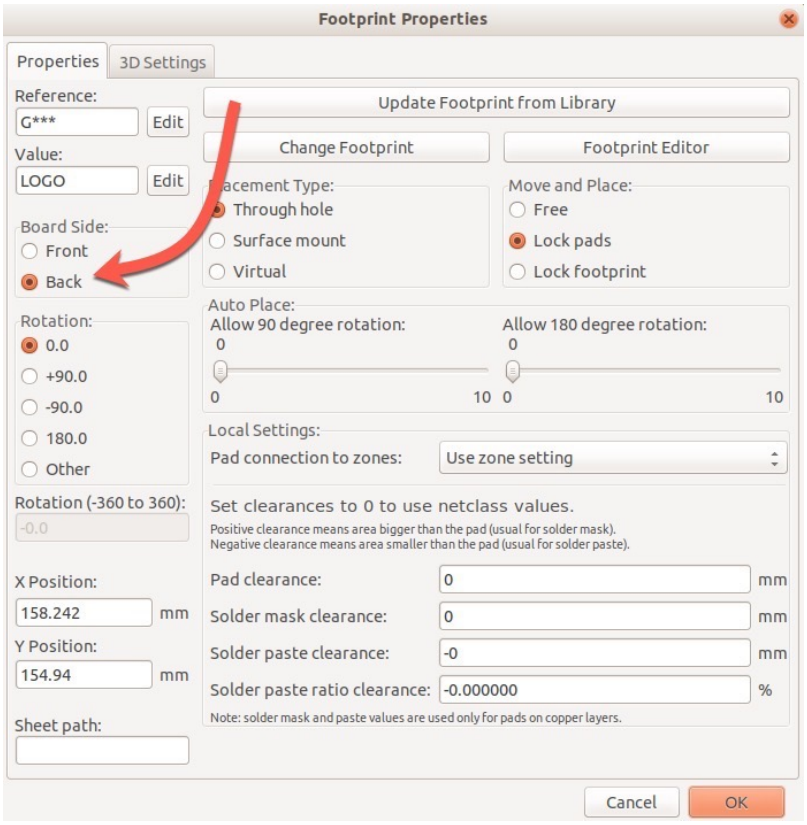


Figure 29.5: Switch your logo to the back silkscreen layer via the footprint properties window.

The PCB should now look like the example in Figure 29.6.

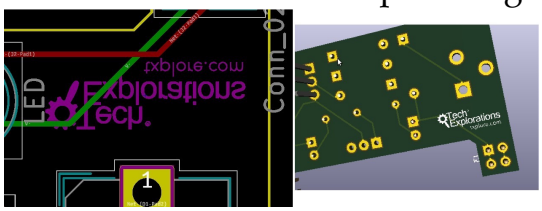


Figure 29.6: The logo now appears on the back silk screen.

As long as your original bitmap image is of decent quality, you will be able to decorate your boards with beautiful graphics.

30. How to manufacture a PCB with Oshpark

In this recipe you will learn how to order a PCB from Oshpark, using Pcbnew's '.KiCad_pcb file'. This method reduces a lot of the risk associated with the relative complexity of using Gerber files to do the same thing. Please note that very few online manufacturers can accept .KiCad_pcb files. If you want more control over the manufacturing details of your PCB, please consider using the traditional Gerber files method. You can learn more about this in the recipe "30. How to manufacture a PCB with Oshpark".

We'll use Oshpark.com as this is one of the very few services currently offering this option. Please follow this process:

1. Start by using your file manager to browse into your project's folder. In that folder, locate the file with the extension '.KiCad_pcb'. This is the file that contains your PCB's layout data, and it was created by Pcbnew.

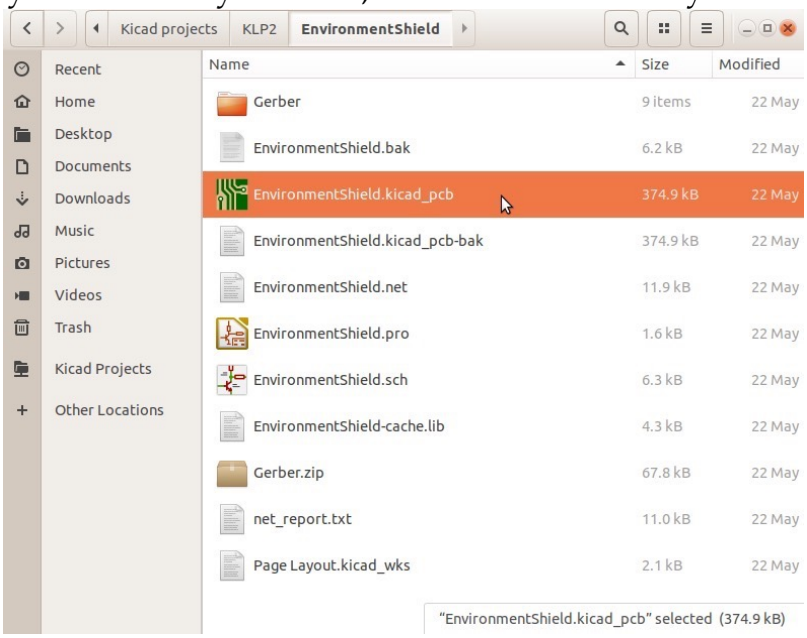


Figure 30.1: The KiCad_pcb file contains the layout of your PCB

2. Use your browser and navigate to www.oshpark.com.

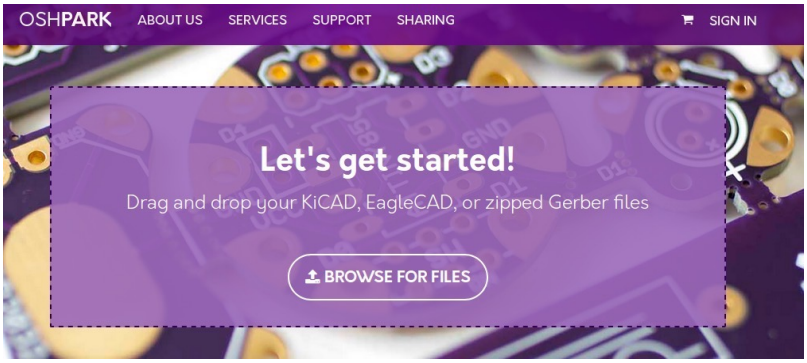


Figure 30.2: One of the attractions of Oshpark is it's simple user interface.

3. Drag and drop the KiCad_pcb file onto the purple box in the Oshpark home page.

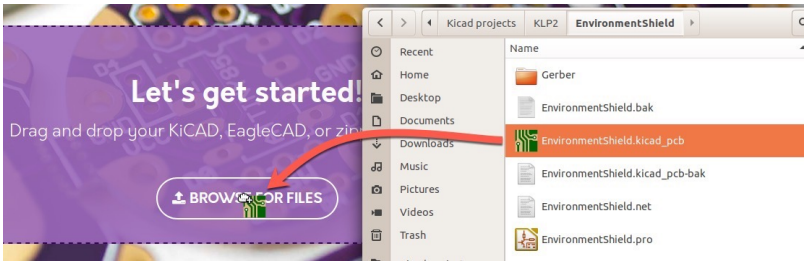


Figure 30.3: Uploads are as easy as drag and drop.

4. If you are using a development version of KiCad that is not yet supported by Oshpark, you will receive a message prompting you to upload Gerber files instead. In this case, please read the Gerber files recipe, create a Zip archive with the Gerber files, and upload the archive to Oshpark. If the upload of your KiCad_pcb file was successful, then let's continue.

5. If the upload succeeded, you will see a rendering of the front and back layers of your board, and a form where you can provide a name and description of your project, and your email address.

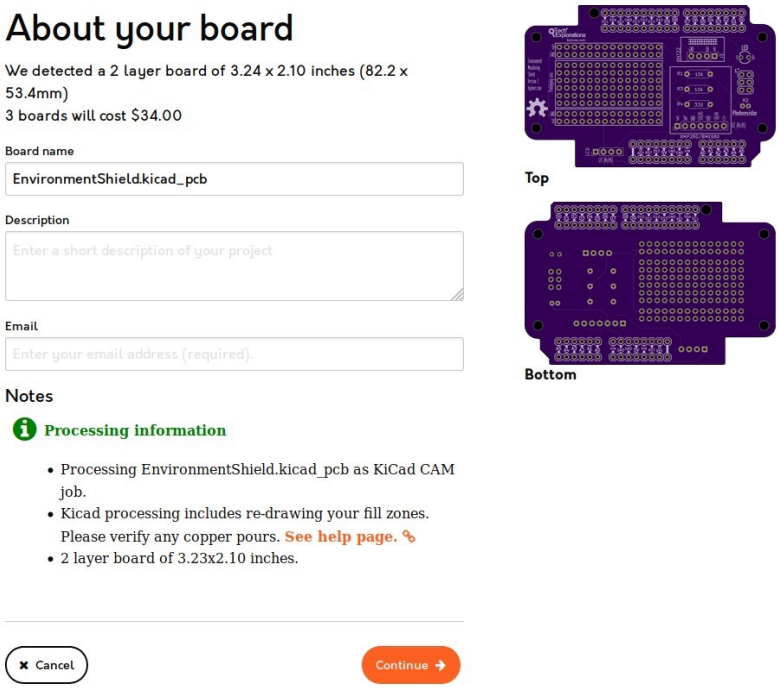


Figure 30.4: The upload was successful.

6. Fill in your project details, and click on Continue. The checkout process will take you through a verification step, where you can inspect that each layer of your board is correct, and a payment step. Oshpark's service is extremely simple. The only technical choice you have to make is whether you would like to opt for the 2 oz copper, 0.8mm thick PCBs. If you choose this option you PCB will have half the thickness of a 'normal' PCB, but with more copper in each trace. This is a good option for PCBs that will operate with higher currents in more confined spaces.

At this point, you can proceed with your order and a few weeks later you will receive your PCBs. Oshpark will notify you via email as your order is progressing.

31. How to make and test Gerber files

The vast majority of online PCB manufacturers accept Gerber files. Gerber files consists an industry standard, and virtually every PCB CAD application can produce them. In this recipe, you will learn how to export the necessary Gerber files, package them in a Zip archive, and upload them to a manufacturer. For the manufacturer, we will use pcbway.com, but the process is very similar across the industry.

PCBWay is a quality manufacturer located in China. Apart from competitive pricing, they offer a vast range of customisations. While Oshpark aims for simplicity, PCBWay aims to satisfy every conceivable PCB manufacturing desire.

In Pcbnew, to generate the Gerber files, follow this process:

1. Click on the Plotter button  in the top toolbar to bring up the Plot window.

2. PCBWay provides information about which Gerber files to produce and the various settings that you will need to enable. Other manufacturers should have similar information available on their website. There are two sets of files that you must create. The first set contains individual files for each layer of the PCB. The second set contains a file for the drill so that the manufacturer can know how to drill the holes and vias. You can see the main Plot window in Figure 31.1. If you are uploading your Gerber to PCBWay, you can set up your Plot window as in this example. Before you click on the Plot button to generate the files, click on the folder button on the top right corner of the window to change the output directory to a new directory. I usually name directories that contain Gerber files 'Gerber'. Once you set the output directory, click on the Plot button. In the output messages text area, you will see green text indicating that the files were created. Do not close this window yet!

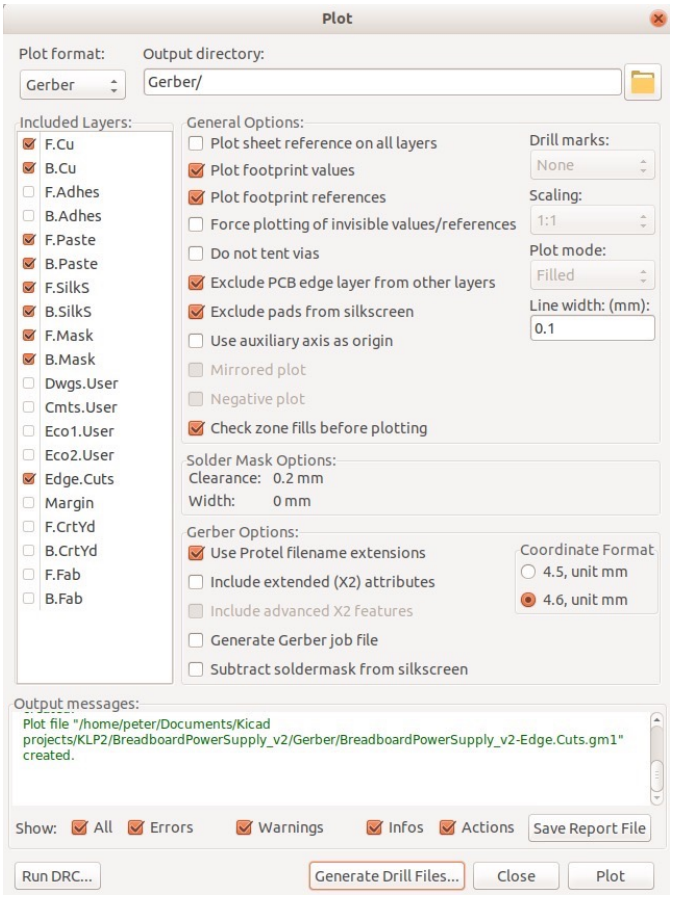


Figure 31.1: The Plot window, where you create the individual layer files.

3. You still need to create the drill files, unless your PCB has no holes. Click on the Generate Drill Files button in the bottom of the Plot window to do that. If you are uploading your Gerber to PCBWay, copy the setting from Figure 31.2, and click on Generate Drill File. This file will be stored in the Gerber directory along with the rest of the Gerber files. You can now Close both Drill and Plot windows.

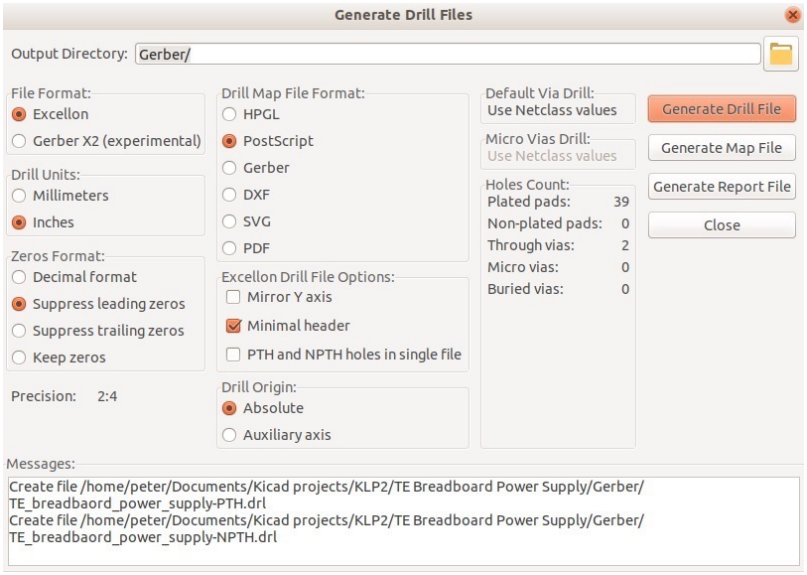


Figure 31.2: The Generate Drill Files window.

4. Use your file manager and go to your project's directory. Have a look inside the Gerber's directory and confirm that the Gerber files are there.

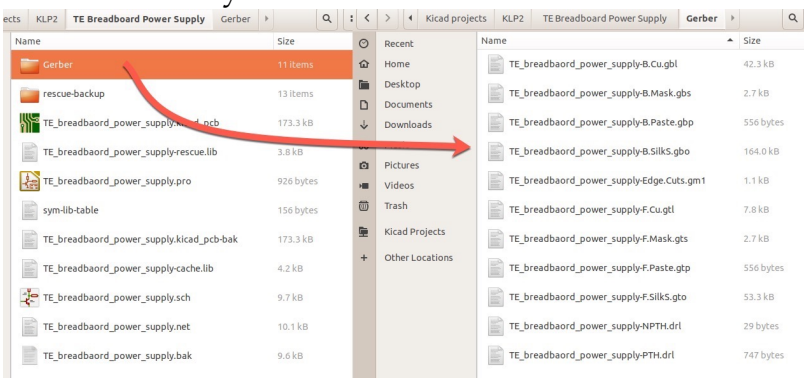


Figure 31.3: The Gerber files inside the Gerber directory.

5. Next, you must create a Zip archive that contains the Gerber directory with all the files in it. In Ubuntu, this can be done with a right-click, and by choosing the 'Compress…' option (Figure 31.4). The Gerber.zip files should be in your project directory (or wherever your archiving utility stored it).

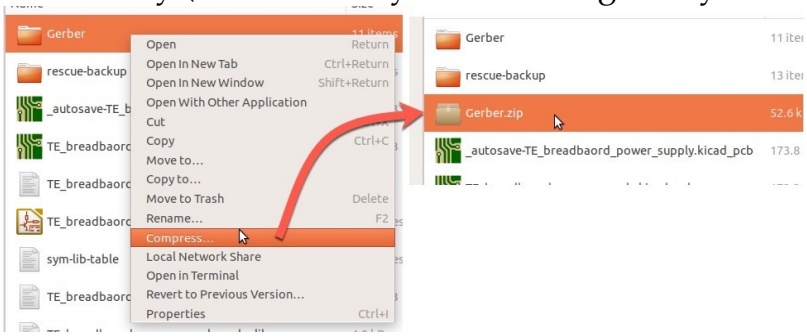


Figure 31.4: Create a Zip archive of the Gerber directory.

6. Upload the Zip file to a service like gerblook.org to ensure that they are correct. Gerblook will render the board layers, and you can visually inspect potential issues.

You are now ready to order your PCB from an online manufacturer using your Gerber files archive. To learn how to do this with pcbway.com please read the next recipe.

32. How to manufacture a PCB with PCBWay

Now that you have a Zip file that contains your project Gerber files, you can proceed with the upload process. PCBWay wants you to provide some information about your board first before you upload the Zip file. You will need to find out your board's dimensions, how many layers it has, and decide about the board thickness (Figure 32.1).

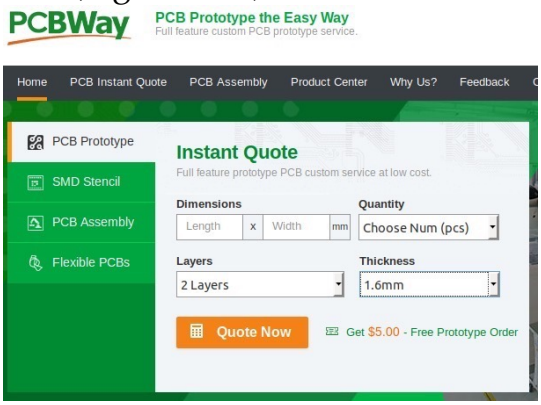


Figure 32.1: PCBWay requires some information before you upload your Gerber's.

Follow this process to find out the dimensions of your board and complete the ordering process:

1. Open your PCB project in Pcbnew.

2. Use the Measure Distance tool () from the right toolbar to measure the width and length of your PCB.

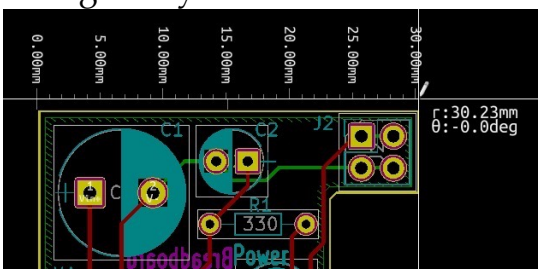


Figure 32.2: Use the Measuring tool to find out the dimensions of your board.

3. Type in the length and width values in the form (Figure 32.3). Choose your desired quantity, number of layers and thickness (1.6mm is a good standard value, only change it if you have a good reason for doing so). Click on Quote Now to go to the quotation page.

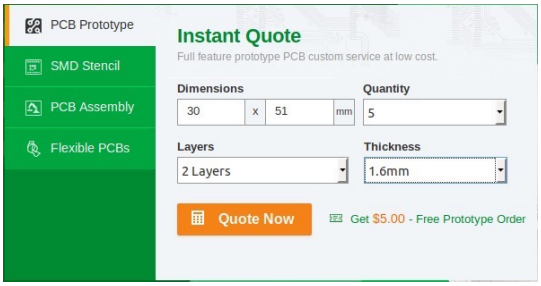


Figure 32.3: The PCB prototype form filled in.

4. The Quote page can be overwhelming at first (Figure 32.4). Most of the default settings are safe to keep. The only change that I make is to select the HASL lead-free surface finish since I try to work in a lead-free environment. This page does give you a good overview of the kind of customisations that are possible with PCBWay and why they are worth considering for your project. You can make boards on aluminium or rig-flex boards, with a large variety of thicknesses, solder mask colours, and surface finishes. They even support black silkscreen! When you are ready, click on the 'Add to cart' button (not showing in Figure 32.4, this button is on a side panel). This will take you to the upload page.

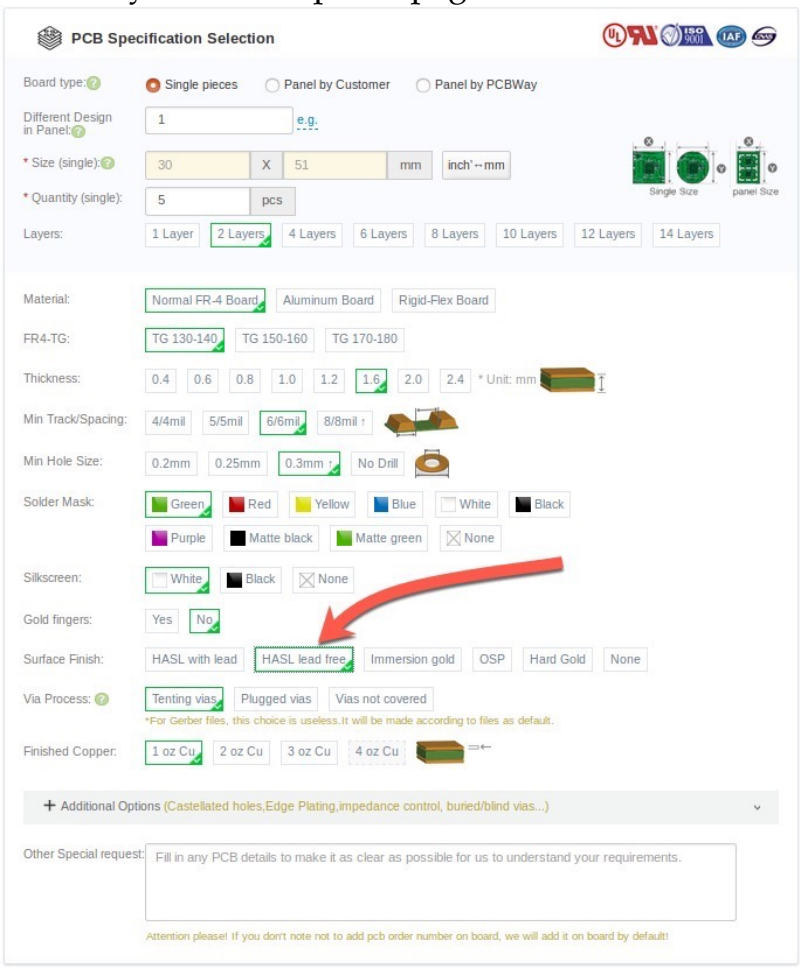


Figure 32.4:Most default options are safe to keep, except for the surface finish.

5. In the next step, you are prompted to upload the Gerber Zip archive.

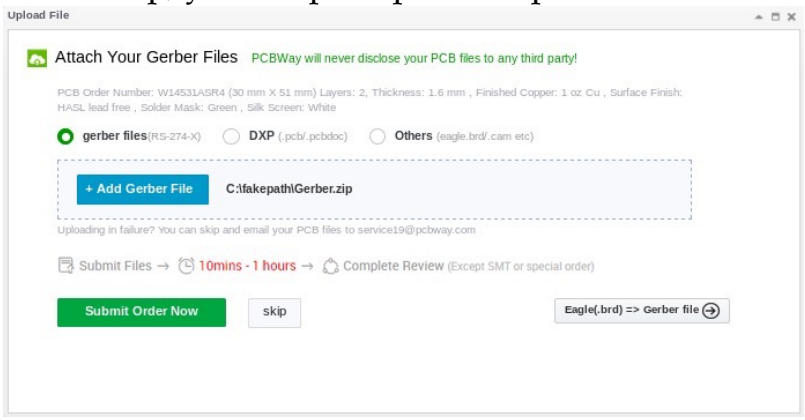


Figure 32.5: PCBWay upload facility.

6. When the upload is complete, PCBWay's engineers will check your project to ensure that it is technically valid and manufacturable. You will not be able to check out and pay for your boards until the audit is completed (Figure 32.6).

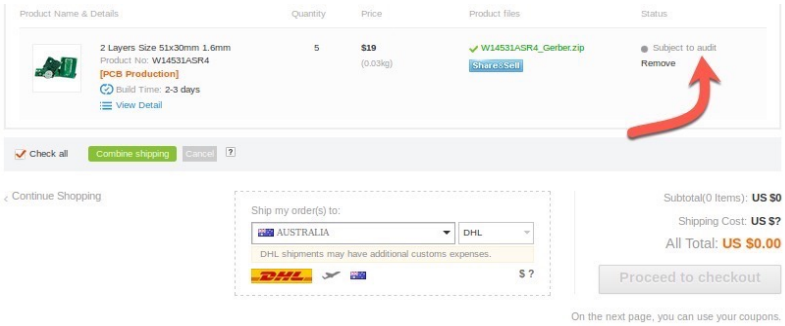


Figure 32.6: Your PCB project is subject to an audit by PCBWay's engineers before you can checkout.

The audit process usually takes a few hours to complete. If all goes well, you will receive an email advising you that you can complete the checkout. Once paid, you'll have your boards in a few weeks.

33. Rounded corners

In this recipe, you will learn how to create a rounded corner, like the one in the example in Figure 33.1. In this example, the arrows point to some of the rounded corners of the board. Although it does take additional work to design these corners, they do contribute to making this board look and feel 'quality'.

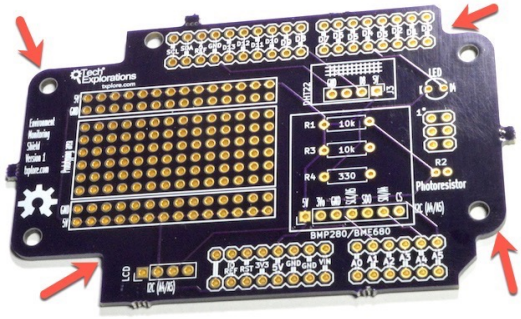
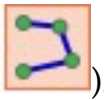


Figure 33.1: The arrows point to some of the rounded corners on this board.

To practice how to create an outer and an inner rounded corner, let's use an empty page in Pcbnew. Start a new KiCad project, and open KiCad. Since we will be designing the layout of a PCB, opt for a large grid and zoom factor. In my design, I use 1.27 mm (50 mils) for the grid size, and 11.46 for the zoom factor. Select the Edge.Cuts layer, and click on the Line tool (). In Figure 33.2 you can see our objective: our aim is to replace the 90-degree angled of this board with rounded corners.

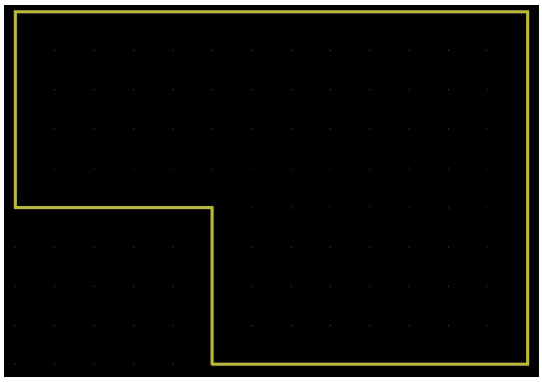


Figure 33.2: We will convert the angled corners into rounded corners.

We will need to re-draw parts of this outline, but if you wish to follow along with the practice operations, go ahead and draw the polygon.

To create rounded shapes, you will use the arc tool (). Click on its button to select it. Next, zoom in to the top left corner to practice using the arc tool. To draw an arc, you will go through a sequence of three clicks. First, click on the location where you wish to set the centre of the circle of which the arc is part of. Then, click at a location that defines the radius of the arc. Finally, click to define the length of the arc. In Figure 33.3, you can see this sequence.

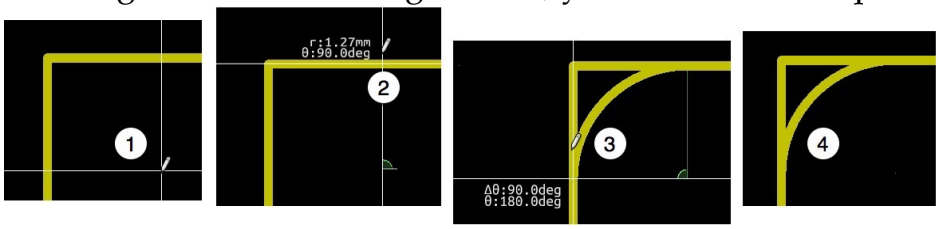


Figure 33.3: Creating an arc involves three clicks.

From left to right:

1. Using the grid as a guide, click inside the corner so that the centre of the arc's circle is across the corner. I am using a 1.27 mm grid, so I placed my mouse pointer to the grid dot that is located 1.27 mm horizontally and 1.27 mm vertically from the corner.

2. Move the mouse one grid segment upwards, to meet the existing outline line, and click. This defines the radius of the arc, which is 1.27 mm.

3. Finally, draw the 90 degree arc itself. Move the mouse towards the vertical line until it is over the grid dot, 1.27 mm below the corner. Click to complete the process.

In the fourth image of Figure 33.3 you can see the outcome of this process. This was only practice. The corner is still there, and to complete the work we will need to remove the lines that make it, leaving only the arc. But before we get to that, I'd like to show you one more thing.

Still working on the top left corner, create a new, longer arc, like the one in Figure 33.4 (the new arc is the longer one).

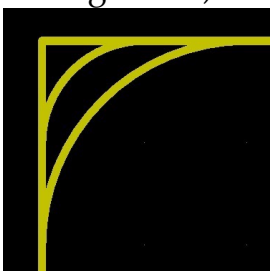


Figure 33.4: You can create arcs of arbitrary lengths.

To create this longe arc, place your mouse on the grid point that is 2 x 1.27 mm = 2.54 mm vertically and 2.54 mm horizontally away from the corner. Define the arc radius and draw the arc in the same way as the first arc. You

will end up with a longer arc than the first one. This is how you can create rounded corners of arbitrary size.

Let's continue to finish work on this corner (delete the longer arc, if you created it).

The next step is to remove the lines that create the 90-degree corner. This process is described visually in Figure 33.5.

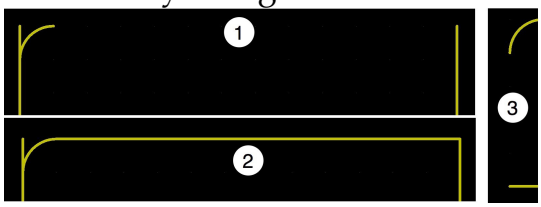


Figure 33.5:Delete and replace the lines, leaving the rounded corner in place.

Refer to Figure 33.5. Start with the horizontal line. Place your mouse over it and type the Del key to remove the line (1). Use the line tool to restore the horizontal line, but this time start it from the top-right corner to the edge of the arc (2). With the grid size at 1.27 mm, it should not be a problem creating a perfectly closed outline. If the outline is not closed, KiCad will show a warning message when you attempt to use the 3D Viewer. Continue with deleting the vertical line (3). Restore this line, and you will end up with an outline like the one in Figure 33.6, with a rounded corner.

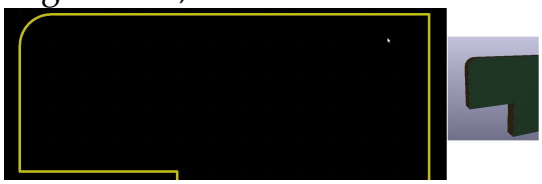


Figure 33.6:The top left corner is now rounded.

Follow the exact same process to replace the rest of the external corners with rounded corners. Each time, draw the arc in the size you want, and replace the straight lines, always taking care to produce a perfect closed outline (Figure 33.7).

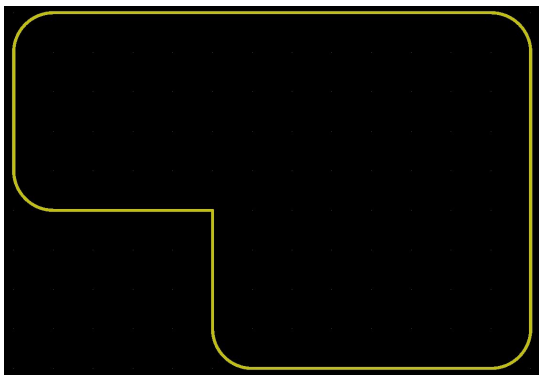


Figure 33.7: All external corners are rounded; let's work on the internal corner next.

For the remaining corner, the internal corner, we'll use a slightly modified process. Instead of placing the centre of the arc's circle inside the outline, we'll place it on the outside. The process is described visually in Figure 33.8.

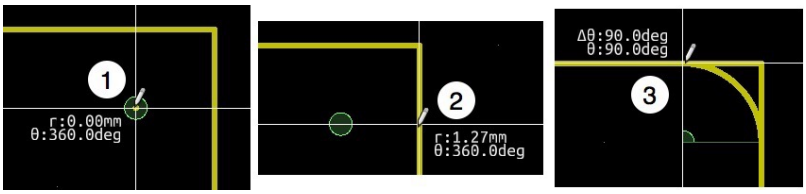


Figure 33.8: Creating an internal corner.

Start by clicking on the grid dot that is 1.27 mm horizontally and 1.27 mm vertically away from the corner, outside the outline (1). Move the mouse pointer right, on the existing line, and click to define the radius (2). Then draw the arc so that it meets the horizontal line (3). Finally, delete the two straight lines that make up the corner, and replace them with new lines that meet the arc. In Figure 33.9 you can see the completed board outline with rounded corners.

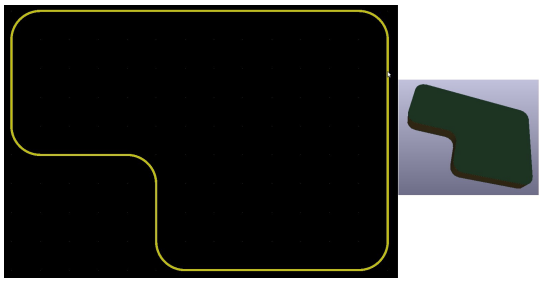


Figure 33.9: The board outline with completed rounded corners.

As you learned in the chapter 'PCB layout process', you should design the outline of the board before you place any footprints on it and do any routing. This will make it much easier to draw the outline since you will not need to negotiate obstacles. This is also the best time to create screw holes and other openings (as, for example, for passing wires). Please read the 'Creating mounting holes and openings' recipe to learn more about this.

34. Mounting holes and openings

The easiest way to create openings for requirements such as a screw or mounting holes, passing wires, or other component requirements, is to design them in the Edge.Cuts layer in the exact same way that you go about designing the board outline. Essentially, when you create a board internal outline, the manufacturer will treat it as a void, and carve material out of it.

If your manufacturer does not support this method, you can either choose to switch to a manufacturer that does or use pads or other footprints that have the shape you want. You can also create your own footprints for this purpose.

In this recipe, you will learn how to use the Edge.Cuts and pads methods.

The instructions that follow build a board similar to the one I describe in the 'How to create a rounded corner' recipe. You can see our starting point in Figure 34.9. The only difference is that the one in this recipe is larger (144 mm x 76 mm) so that there is enough space in it for the openings.

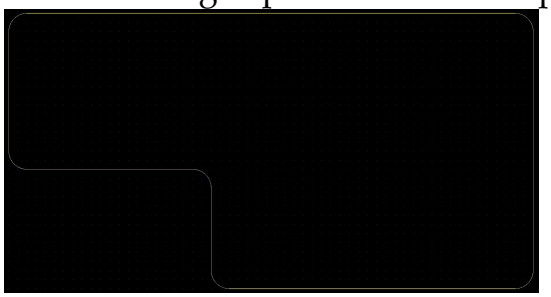


Figure 34.9: We will add four screw holes and an opening to this board.

To create mounting holes for the screws, start by measuring the size of the screws you wish to use. This is done easily with a calliper if you have one (Figure 34.1).

Figure 34.1: This screw will require an opening slightly larger than 2.88 mm.

The screw I measured in Figure 34.1 has a width of 2.88 mm, so it will require an opening that is slightly larger than 2.88 mm. I would like to place four screw holes on the board, along with its outer corners. To make it easier to achieve opening diameters of around 2.95 mm or 3.0 mm, I will change the grid size to 0.508 mm.

Let's start with the first screw hole at the top left corner of the board. Click on the Circle tool to select it ().

Place your mouse cursor across the top left rounded corner, inside the board outline, to a location that will allow you to draw a circle with at least a 3 mm diameter. This translates to 1.5 mm in radius. We will use the radius value because the KiCad status bars provide this as we draw the circle. The example in Figure 34.2 shows the process.

Figure 34.2: Drawing a circle with a radius of 1.524 mm.

Click to set the circle centre, and then drag the mouse and click again to define the circle radius. KiCad's status bar displays the current radius as you draw the circle. Because of the grid size, I have selected, the closest value I can get to 1.5 mm for the radius is 1.524 mm. That is close enough, so I will click there to complete the drawing. Using my calliper, I have measured that the head of the screw is 5.39 mm, so that gives me a lot of tolerance to work with as I design the holes.

You can look at the 3D rendering of the hole, in Figure 34.3.

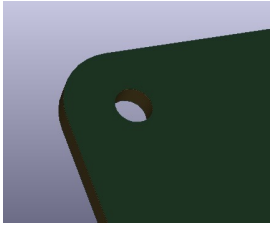


Figure 34.3: The new hole is rendered properly in the 3D viewer.

Repeat the same process to define the other three holes. Alternatively, because you want to create additional holes of the same size, you can duplicate the first hole by clicking on the hole outline to select it and typing Ctr-D. In Figure 34.4 you can see the board with all the screw holes in place.

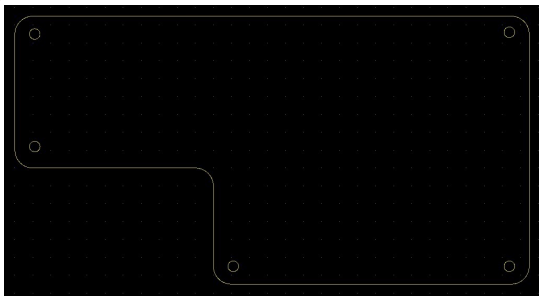


Figure 34.4: Five identical screw holes placed along the edges of the board.

You can verify that your board is as expected by using the 3D Viewer.

An alternative way to create mounting holes is to use a pad footprint, or even simpler, the via tool. Let's try that next to place a sixth hole adjacent to the inner corner. Click on the via tool to select it (). Place your mouse over the location where you'd like to via to go and click.

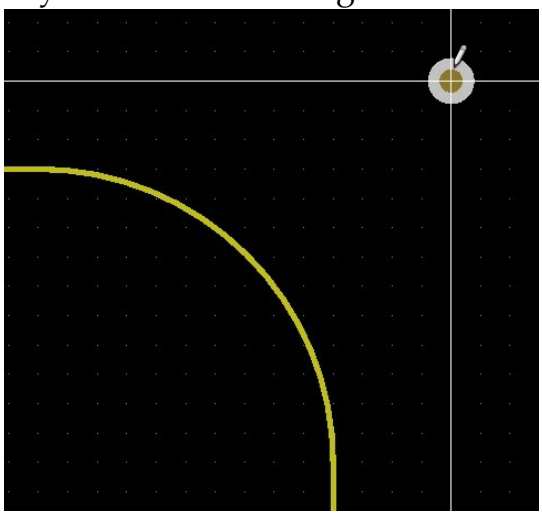


Figure 34.5: Placing a via.

Next, you must define its size so that the screw can fit through it. Place your mouse over the via and type 'E' to bring up its properties. You can see the properties window in Figure 34.6, with the values that match the screw I plan to use. Take care to check the 'Locked' checkbox to prevent the via/screw hole from being moved later when you work on the components and traces. I defined the drill size diameter to 3.1 mm to accommodate the size of the screw. The via diameter must be larger than the drill, otherwise, KiCad will display an error message. I set the diameter to 3.3 mm (Figure 34.6).

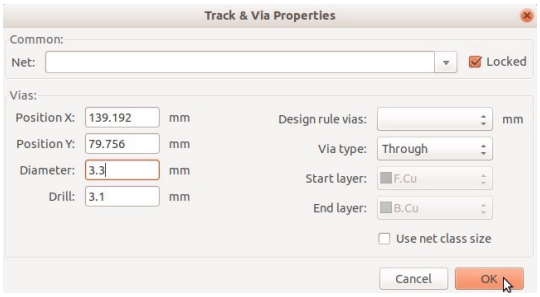


Figure 34.6: The via properties, appropriate for repurposing it as a screw hole.

Click Ok to commit these values. You can see the end result in Figure 34.7.

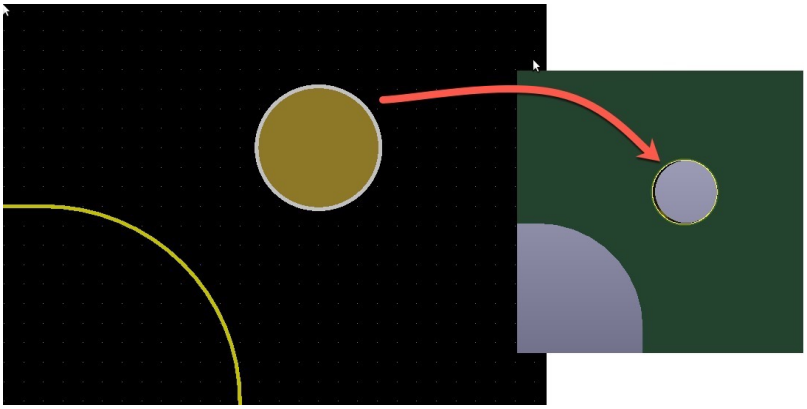
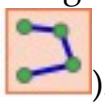


Figure 34.7: The via as a screw hole; as every via, it is plated internally.

As with all vias, the one you just created is plated internally. A small amount of tin is also present on the surface, just around the hole. In the 3D representation of Figure 34.7 you may be able to see a very thin light-coloured line. This is not a problem for the new role of this via as a screw hole.

Let's do one more alternation of this board as part of this recipe. You can make intents and opening on the board to facilitate the routing of wires, or as provisions for mechanical characteristics of components. Let's create a simple rectangular opening in the board. Working in the Edge.Cuts layer, use the line tool to create a rectangular opening (). Draw this rectangle on the right side of the board, as you can see in Figure 34.8.

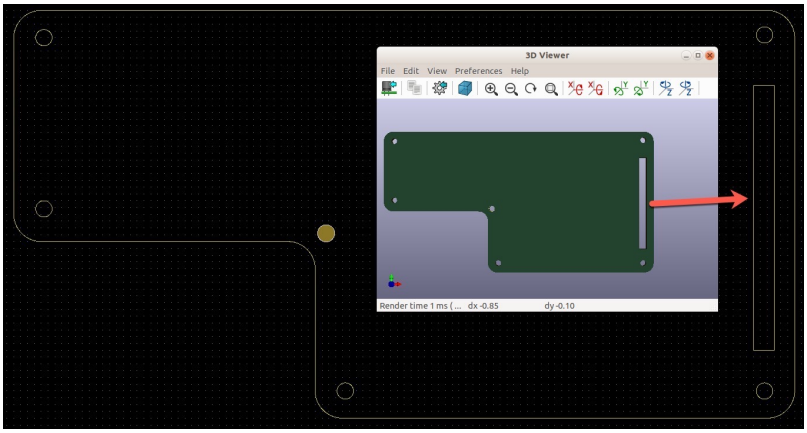


Figure 34.8: The board has a new rectangular opening, with its 3D rendering.

If your manufacturer is unable to work with internal cutouts in the Edge.cuts layer, you can still create them as custom footprints. Learn how to create footprints in the 'Creating a new footprint' recipe.

35. Creating a new component (symbol)

In this recipe, you will learn how to create a custom symbol. You use symbols in schematic diagrams that you can create in Eeschema. Much of what you will learn in this recipe you will be able to reuse for modifying existing symbols. There is a separate recipe in this book that explains how to do that.

Most likely, you want to create a custom schematic symbol because you have a physical component but can't find an existing symbol to represent it. Perhaps you have searched through the symbols that come with KiCad, and Googled for suitable third-party symbols, but couldn't find any.

It is also likely that you will want to associate your new symbol with a footprint. If your physical component has a standard package, like DIP, for example, then you will be able to associate an existing footprint with your custom symbol. If not, you will also need to create a custom footprint. You can learn how to do this in the relevant recipe, also available in this book.

In this recipe, you will learn how to create a custom symbol by creating a symbol for the 555 timer integrated circuit. There are plenty of libraries for KiCad that contain this symbol, but for the sake of learning, let's pretend that we can't find it.

What we want to create is a symbol like the one in Figure 35.1.

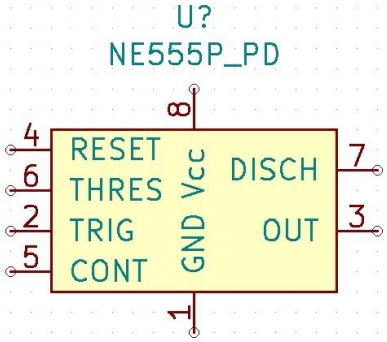


Figure 35.1: A custom-made symbol for the 555 IC.

Our objective is to create a symbol that complies with the convention. In regards to IC symbols:

1. We arrange pins around a rectangle.
2. We group pins according to function (like inputs, outputs, power etc.)
3. We place the Vcc pin on the top of the rectangle.

4. We place the GND pin on the bottom of the rectangle.
5. We choose an appropriate name and designator for the symbol.

Symbol designators are standardised; you can learn more about them here.

The physical component that we are working with is in Figure 35.2. This component comes in a standard DIP package with 8 pins.

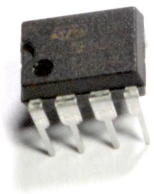
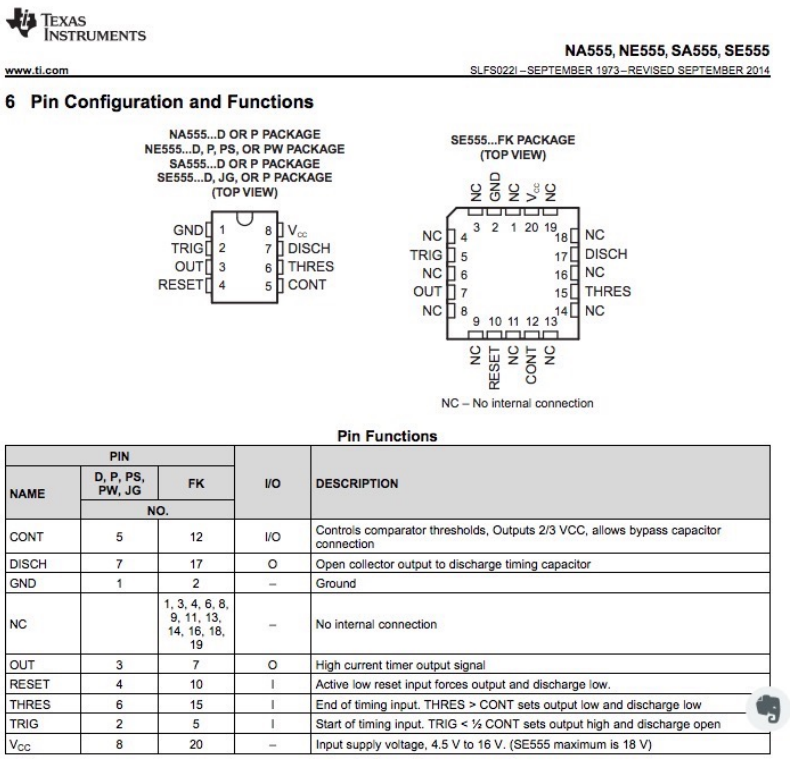


Figure 35.2: The physical component is a 555 timer in a DIP package with 8 pins.

Because we are working to create a symbol, we don't need to know anything about the physical characteristics of the physical component other than the total number of pins that its package contains. You need to know details about the physical characteristics when you are working on the footprint of the component.

However, it is beneficial to have access to the data sheet of the component. The data sheet contains information that you need: The names, numbers, and roles of each pin, whether they are input, output, bidirectional, power, signals, etc. All this is useful information, and the more you have on hand, the better.

The data sheet for the example component is available from its manufacturer. The information you need is on page 6, and I have included it in Figure 35.3 for your convenience.



TEXAS INSTRUMENTS
www.ti.com

NA555, NE555, SA555, SE555
SLFS022I – SEPTEMBER 1973 – REVISED SEPTEMBER 2014

6 Pin Configuration and Functions

NA555...D OR P PACKAGE
NE555...D, P, PS, OR PW PACKAGE
SA555...D OR P PACKAGE
SE555...D, JG, OR P PACKAGE
(TOP VIEW)

SE555...FK PACKAGE
(TOP VIEW)

NC – No internal connection

Pin Functions

NAME	PIN D, P, PS, PW, JG NO.	PIN FK NO.	I/O	DESCRIPTION
CONT	5	12	I/O	Controls comparator thresholds, Outputs 2/3 VCC, allows bypass capacitor connection
DISCH	7	17	O	Open collector output to discharge timing capacitor
GND	1	2	–	Ground
NC		1, 3, 4, 6, 8, 9, 11, 13, 14, 16, 18, 19	–	No internal connection
OUT	3	7	O	High current timer output signal
RESET	4	10	I	Active low reset input forces output and discharge low.
THRES	6	15	I	End of timing input. THRES > CONT sets output low and discharge low
TRIG	2	5	I	Start of timing input. TRIG < ½ CONT sets output high and discharge open
VCC	8	20	–	Input supply voltage, 4.5 V to 16 V. (SE555 maximum is 18 V)

Figure 35.3: Pin configuration and functions from the IC's datasheet.

Let's start the process of creating a new symbol. In the main KiCad window, click on the Symbol Library Editor button (Figure 35.4).

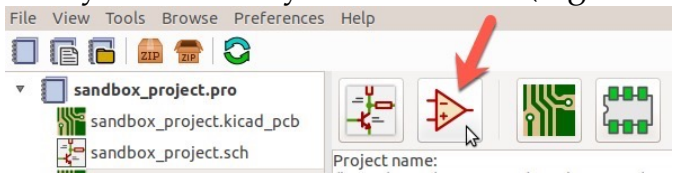


Figure 35.4: Start the Symbol Editor.

You must store each symbol inside a library file, so before you start creating the symbol create a new library. Create a new library by clicking on the 'New library' button ( ), or through the File menu.

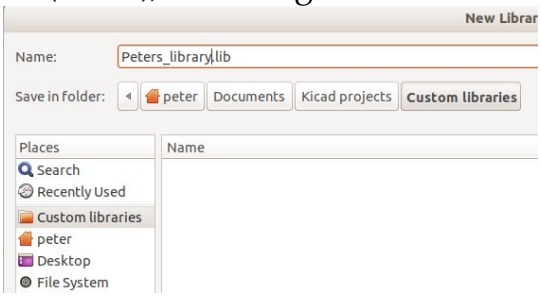


Figure library; I have placed mine in a directory named 'Custom libraries'.

KiCad will ask you if you would like this library to be available to all projects ('Global') or only to the current project ('Project'). Choose the most appropriate one for your circumstances (I chose 'Global'). You are now

working in the new library, and you will store the new symbol in it. You can confirm this by looking at the header of the Editor window. The path and name of the library you just created must be showing there.

Next, click on the 'Create new symbol' button in the top toolbar (). A prompt will ask you to select the library in which you will store the symbol. The library you just created should be listed. Click on it to select it and click 'ok' to continue. The Symbol Properties window will show up. The most important values that you need to complete are the symbol name and designator. The name typically consists of the physical component's model name and any other information that helps to identify it. When you use it later, you have to search for it in the symbol library and having a good name will help you find it quickly. In my example, because I want to differentiate my 555 symbol to those that exist in other libraries, I add my initials 'PD' at the end of the name.

For the designator, you should not guess. Visit Wikipedia to see the Reference Designators table (https://en.wikipedia.org/wiki/Reference_designator). You can see a section of this table in Figure 35.5. The designator for integrated circuits is 'U,' so type this in the Default Reference Designator field. For an authoritative discussion on designator and standards, please refer to a series of articles written by Lawrence Joy at https://txplo.re/96bc7.

RV	Varistor
S	Switch (all types, including push-buttons)
T	Transformer
TC	Thermocouple
TP	Test point
TUN	Tuner
U	Integrated circuit (IC)
V	Vacuum tube
VR	Variable resistor (potentiometer or rheostat)

Figure 35.5: A section of the reference designators standard IEEE 200-1975/ANSI Y32.16-1975.

Figure 35.6, shows the values I have entered in the Symbol Properties window. Other than the symbol name and the designator, everything else remains as per the default.

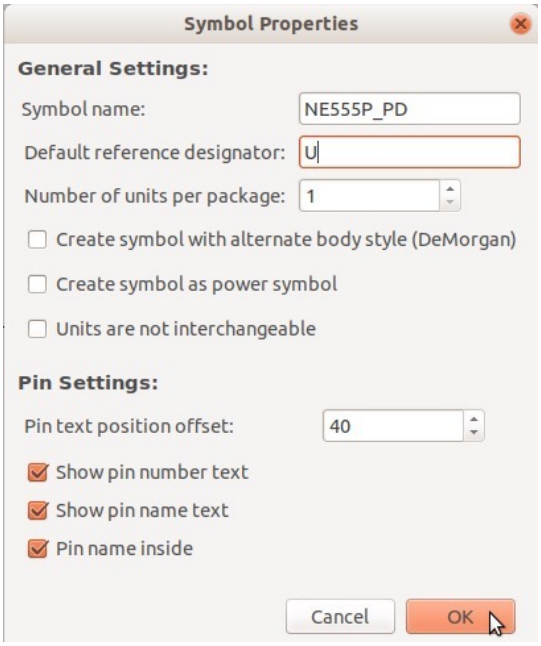


Figure 35.6:The properties for the new symbol.

Click Ok to commit and continue. KiCad will place the designator and symbol name in the middle of the sheet, on top of the other. Use the 'M' hotkey to relocate the two blocks of text. You should have something similar to the example in Figure 35.7.

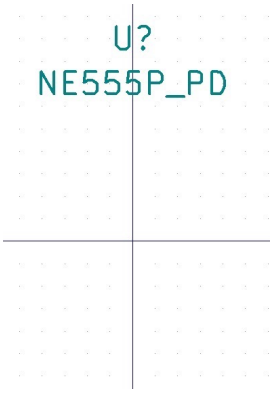


Figure 35.7: An empty new symbol.

Continue by drawing the outline of the symbol. You can either use the polygon tool ( ) or the rectangle tool ( ) to do this. Your outline should look like the example in Figure 35.8.

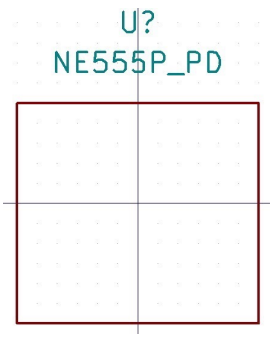


Figure 35.8:The outline of the footprint.

Add the background colour that is consistent with other IC symbols by opening the drawing properties window for the rectangle (place your mouse pointer on the rectangle line and type 'E'). Under 'Fill Style', check the 'Fill background' radio button (Figure 35.9).

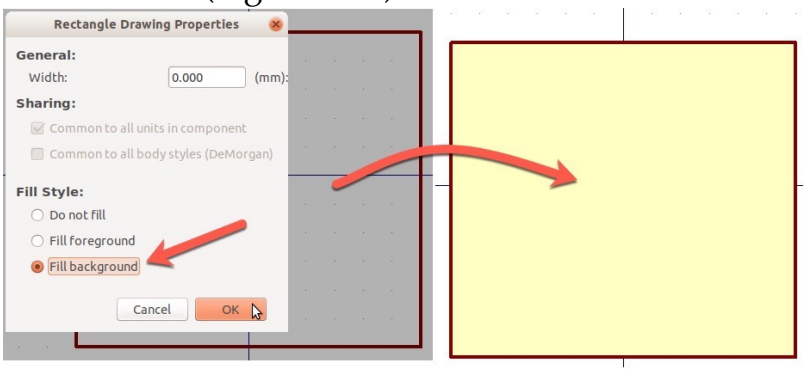


Figure 35.9: Fill the rectangle background.

Next, work on the pins. Keep the datasheet open because you need the information in it. For your convenience, refer to the excerpt in Figure 35.3.

Click on the pin button in the right toolbar ( ). Place the 8 pins around the perimeter of the symbol outline, as in the example of Figure 35.1. The thing to remember here is that the convention is to group similar pins together and place the two power pins to the top and bottom of the rectangle. Let's start with the Vcc pin. According to the datasheet, the Vcc pin is number 8, and according to the convention, it should go to the top edge of the rectangle. Click on the pin tool, and then click in the middle of the top edge. The Pin Properties window will come up. Fill it as you can see in the example in Figure 35.10.

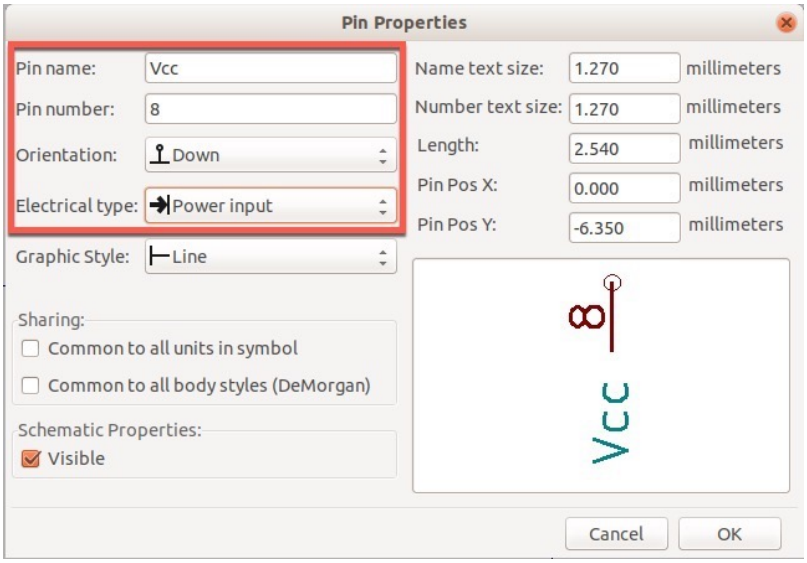


Figure 35.10:The Vcc pin properties.

In Figure 35.10, the fields inside the box are the ones that you need to focus on. The pin name is arbitrary, but of course, you should use a name that is appropriate. I usually use the same name that I see in the documentation for this pin. The Pin number, on the other hand, is very important. The Pin number is how schematic symbols and footprints can associate physical and symbolic pins. When you design the custom footprint for this physical component in the 'Creating new footprint' recipe, it is the number that you put in the Pin number field that dictates which net in the schematic diagram connects to the correct pad in the footprint. Take the pin number for the Vcc pin from the documentation ('8') and type it in this field.

In the orientation drop-down, select the option that matches the side of the rectangle where you are attaching the pin. The Vcc pin should go on the top of the rectangle and should have its circular connector pointing away from the rectangle. The horizontal line of the pin orientation icon represents the rectangle. If you wanted to place the pin on the left of the rectangle, you would choose the icon with the circular connector printing towards the left.

Finally, because the Vcc pin is a power pin, I have selected the 'Power input' electrical type. You should choose the same type for the GND pin.

Click Ok to commit the changes. Place the pin in the middle of the top side of the rectangle, as you can see in Figure 35.11. I have moved the text block so that they don't overlap with the pin.

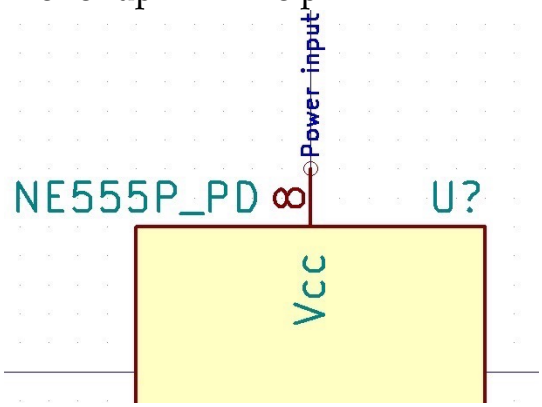


Figure 35.11: The Vcc pin, in place.

Follow the same process to add the GND pin (pin 1) in the bottom edge of the rectangle. Copy the pin name and number from the datasheet, and mark it also as a Power Input.

Continue with the left side of the rectangle where you should place the input pins. According to the datasheet, the input pins are RESET, THRES, TRIG. There is one bidirectional pin, 'CONT,' which you can place either on the left or the right of the rectangle. I have placed it on the left. Your symbol should look like the example in Figure 35.12.

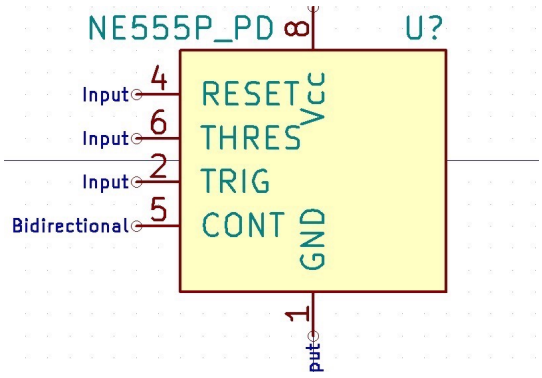


Figure 35.12: Input pins are placed on the left side.

The pin attributes for the input pins on the left of the symbol look like the example in Figure 35.13. Remember that pin 5 is bi-directional so its electrical type should be 'bidirectional'.

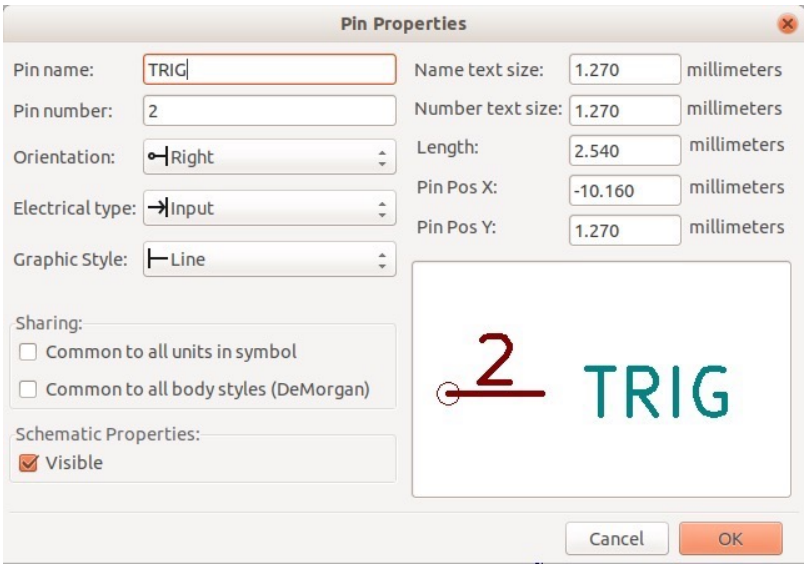


Figure 35.13: The input pin attributes.

Continue in the same way to create the last two pins. Those are output pins, as per the datasheet. Use the pin names and numbers as those appear in Figure 35.3 for pins 7 and 3. When completed, you should have a symbol that looks like the example in Figure 35.14.

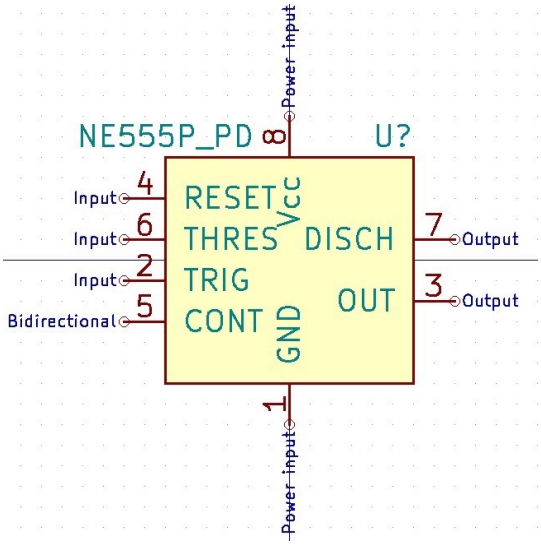


Figure 35.14:The completed custom symbol.

The last thing to do is to add the URL for the real-world component datasheet to the symbol properties. This will be useful to you for future reference. It will be surely useful when you go ahead to create a custom footprint to match the symbol. To add the datasheet URL, click on the Symbol, 'Fields…' to bring up the Field Properties window. Click on the Datasheet row to select it and then copy/paste the URL in the Field Value field (Figure 35.15). Click Ok to commit the changes.

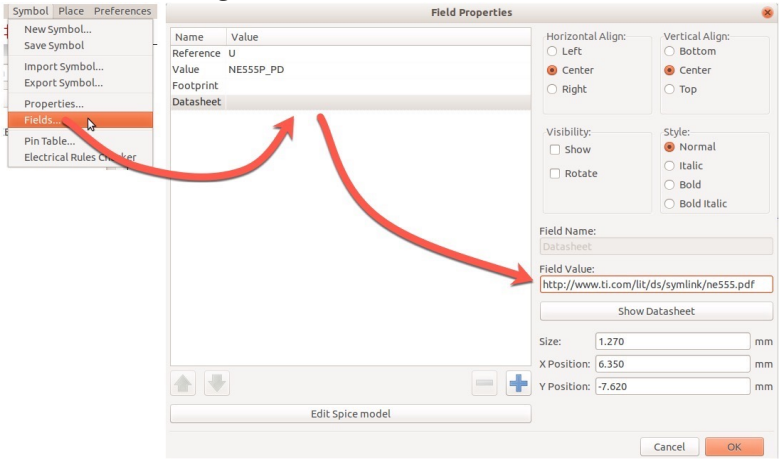


Figure 35.15: The datasheet is very useful to have readily available.

Your work is complete. Save the symbol to the selected library by clicking on the Save Current Symbol button (). Then, test that you can use it in Eeschema. Open Eeschema. Go into Symbol Libraries from the Preferences menu and add the new library (read the relevant recipe if you don't know how to do this). Place your cursor on the Sheet and type 'A' to add a new symbol. Search for the name of your library by typing part of its name

in the filter field (Figure 35.16). The library should appear. Double-click on the symbol to drop it on the sheet.

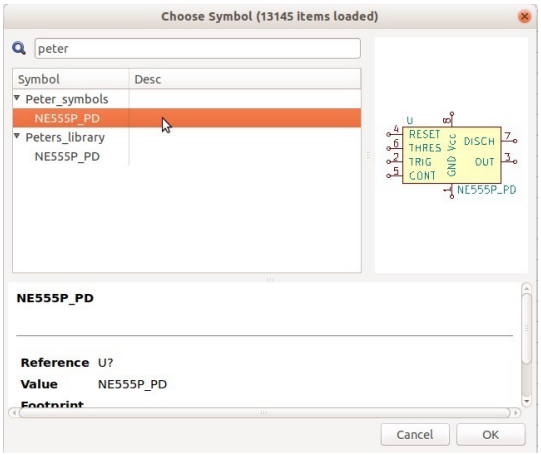


Figure 35.16: Find your new library and symbol.

The custom 555 symbol should now be in place inside Eeschema, and you can go ahead to use it as you do with any other symbol (Figure 35.17).

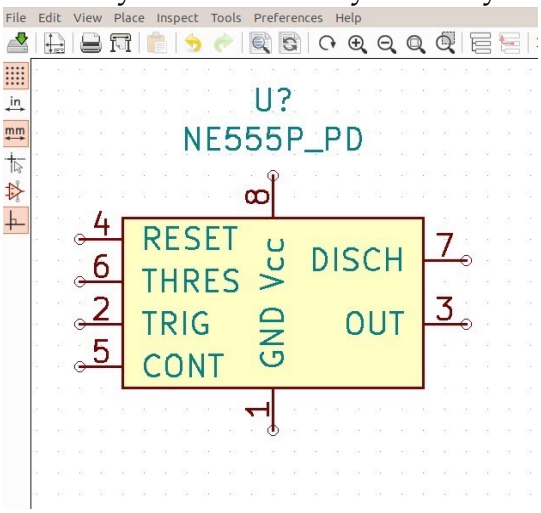


Figure 35.17: Your new custom symbol in Eeschema.

In this recipe, you learned how to create a brand-new symbol. What if you have found a symbol that is close to what you want, but could be perfect with a bit of tweaking? In other words, what if you want to modify an existing symbol? You can learn how to do this in the '36. Modifying an existing component (symbol)' recipe.

36. Modifying an existing component (symbol)

In this recipe, you will learn how to modify an existing symbol. It builds on knowledge that you have learned in the recipe on how to create a custom symbol. If you have not read that recipe (titled 'Creating a new component (symbol)'), please do that now and come back when you have.

The process of modifying an existing symbol starts with finding the symbol that you want to modify. Then, you create a copy of the original symbol, make the changes, and save it to an existing or new symbol library.

Let's begin. Your objective is to make a few cosmetic changes to a symbol that ships with one of the standard symbol libraries of KiCad 5. The symbol name is '74HC595'. This is a common 8-bit shift register that you will find in many Arduino projects. The symbol itself is complete as it comes out of the box, but for the sake of this recipe, let's say that you would like to re-arrange some of the pins, change the location of the text blocks, and add some descriptive text.

First, in Eeschema, find the symbol you want to modify and add it to the sheet. You can do this using the 'Place Symbol' tool from the right toolbar (). With the symbol on the sheet, right click on it to reveal the context menu and click on Properties, 'Edit with Library Editor'.

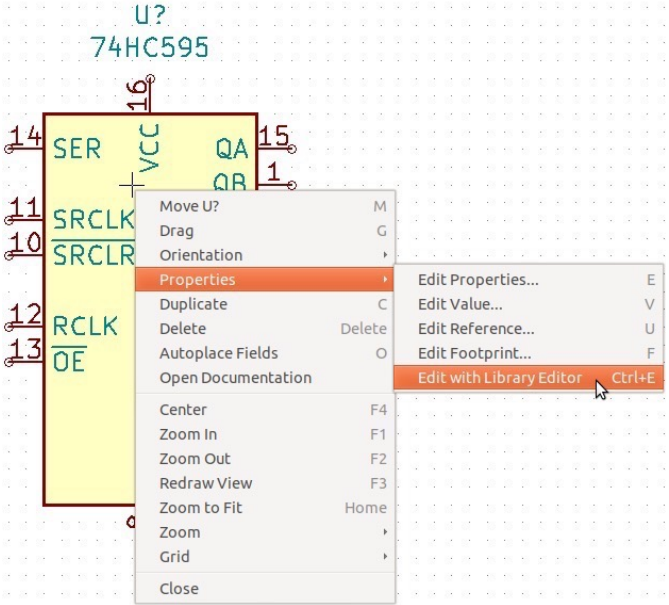


Figure 36.1: Edit an existing symbol with the Library Editor.

You have already used the Library Editor to create a new symbol in the '35. Creating a new component (symbol)' recipe, so the editor window should be familiar (Figure 36.2).

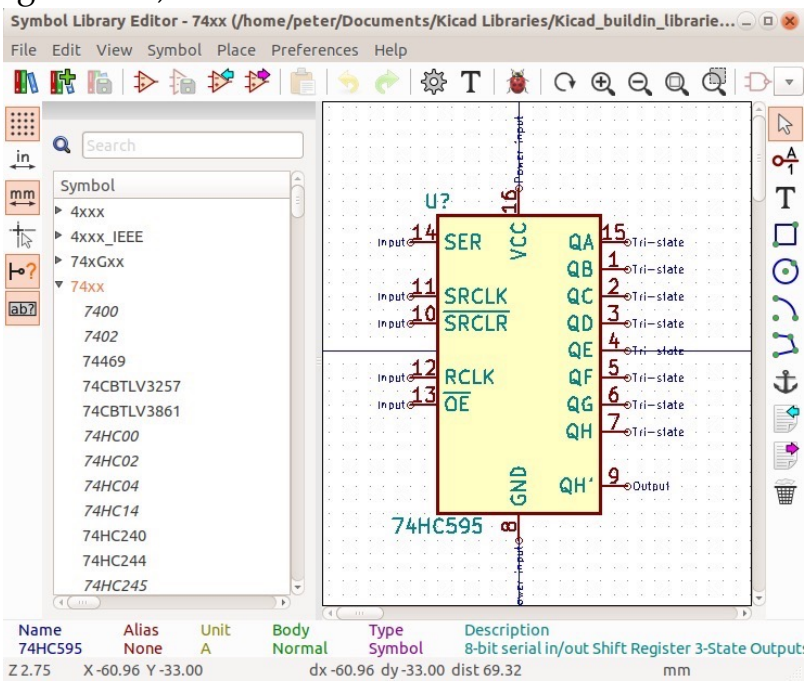


Figure 36.2: An existing symbol, waiting for your modifications.

Using the knowledge you gained in the '35. Creating a new component (symbol)' recipe, move the pins around, change their text, modify and move the text blocks, or add new text as you see fit. I modified the symbol to look like the example in Figure 36.3. I have added a block of text that describes the device and moved the input pins along the left side of the rectangle.

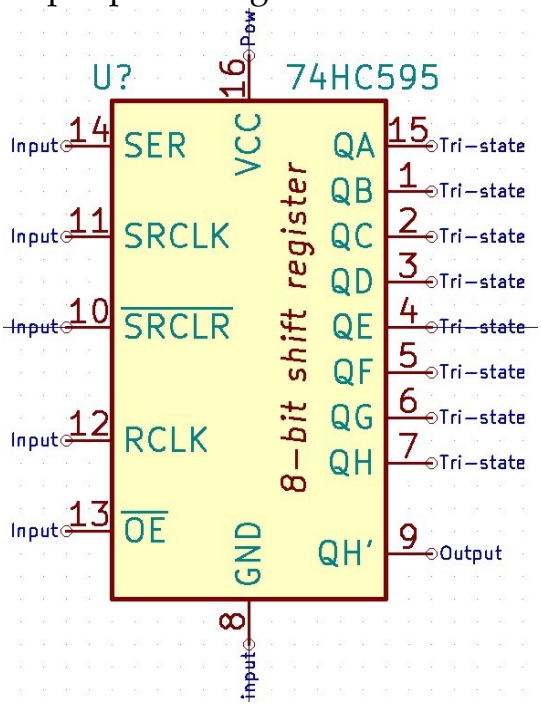


Figure 36.3: A slightly modified version of the build-in 74HC595 symbol.

When your modifications are complete, you must save the footprint in a library. If you click on the Save button ( ), then you will overwrite the original symbol with the modified version. A better option is to save the modified symbol in a new library file. I prefer this option. Click on the Export button ( ), or select Symbol, 'Export Symbol' from the menu. Navigate to your libraries directory, type in a new name for the new symbol and click Ok. I simply added my initials after the original name of the symbol (Figure 36.4).

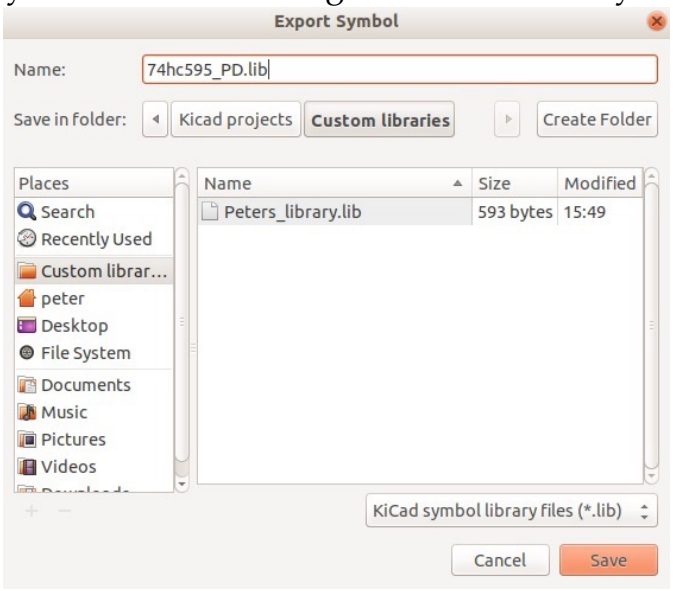


Figure 36.4: Saving the modified symbol.

Finally, let's test your modified symbol. In Eeschema, add the new symbol library to your libraries table if it isn't there already. Add a new symbol to the sheet, which will reveal the symbol chooser window. Use the filter to search for the filter, by typing '74HC595' in the field (Figure 36.5).

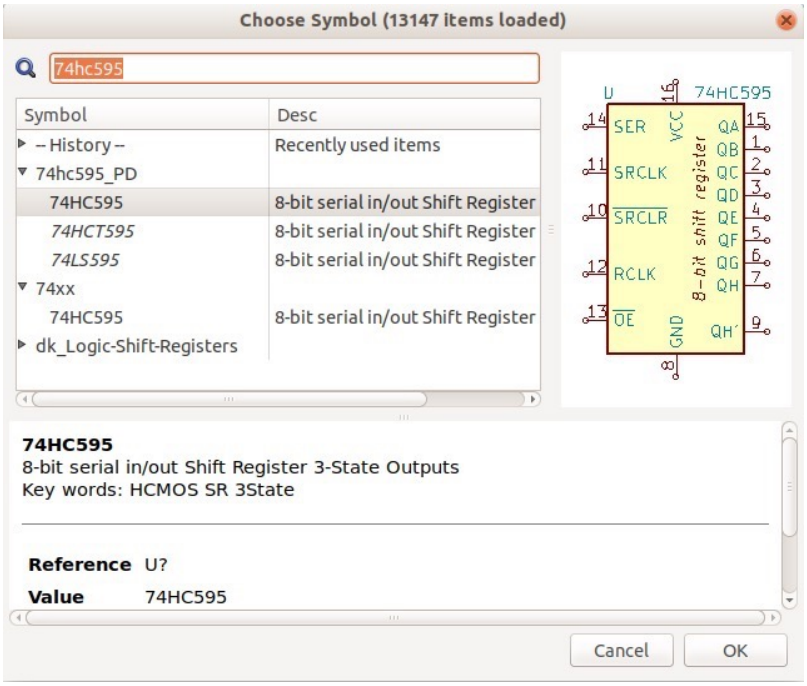


Figure 36.5: The modified symbol appears in the symbol chooser results.

Notice that there are three items inside the '74hc595_PD' library. Apart from the actual symbol name, there are two more: '74HCT595' and '74LS595'. These are aliases, pointing to the same symbol. To edit the aliases, use the symbol properties window from the Library Editor's 'Symbol' menu.

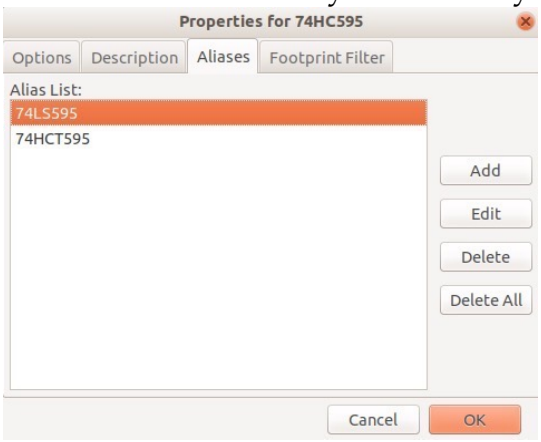


Figure 36.6: Symbols can have aliases, configurable from the Properties window.

Double click on any of the symbol aliases to add the modified footprint to the sheet. The example in Figure 36.7 shows the original symbol on the left, and the modified version on the right.

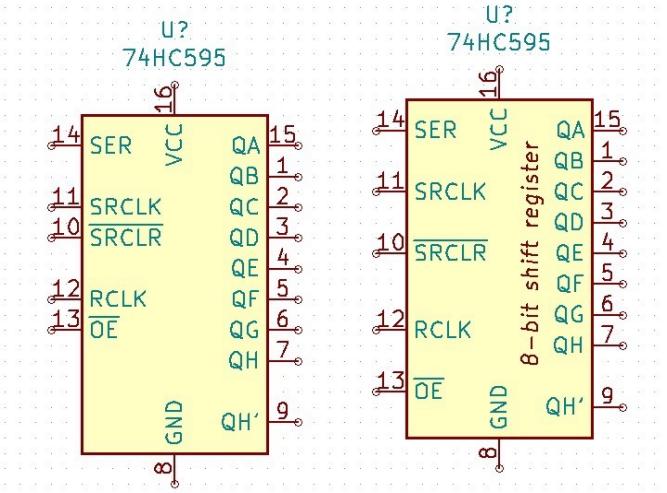


Figure 36.7: The original (left) and modified (right) symbols.

37. Creating a new footprint - manually

After creating a new schematic symbol, you may need to create a matching footprint. In this recipe, you will learn how to do this manually. For some types of footprints, KiCad has introduced a wizard that can accelerate the process. There is a separate recipe that covers that option.

As with creating a new symbol, to create a new custom footprint you will need some mechanical information from the real-world component's datasheet. If you don't have a datasheet, you can use a calliper to take measurements. I usually use both datasheet and calliper together. The datasheet gives me the mechanical values I need for the drawing of the component (width and height of the package, pins and their positions, pin attributes etc.), and then I use the calliper to confirm the dimension and distance values.

In recipe 'Creating a new component (symbol)' you created a custom symbol for the 555 integrated circuit. In this recipe, you will create a footprint for that symbol.

Creating footprints is a slightly more involved process than creating a symbol because we have to consider the physical characteristics of the device, and the manufacturing requirements. We have to think about how to place information about the footprint in the various physical and design layers that KiCad uses for this purpose. For example, information about the boundary of the footprint is placed in the courtyard layer, pads are placed in the copper layers (front and back), the outline of the footprint goes in the fabrication layer, and any artwork (text and graphics) goes to the silkscreen. In Figure 37.1 you can see the elements that make up a typical KiCad footprint, and the layers where those elements are placed. The footprint in Figure 37.1 is what you will work towards creating in this recipe, even though it already exists in a library that you can import.

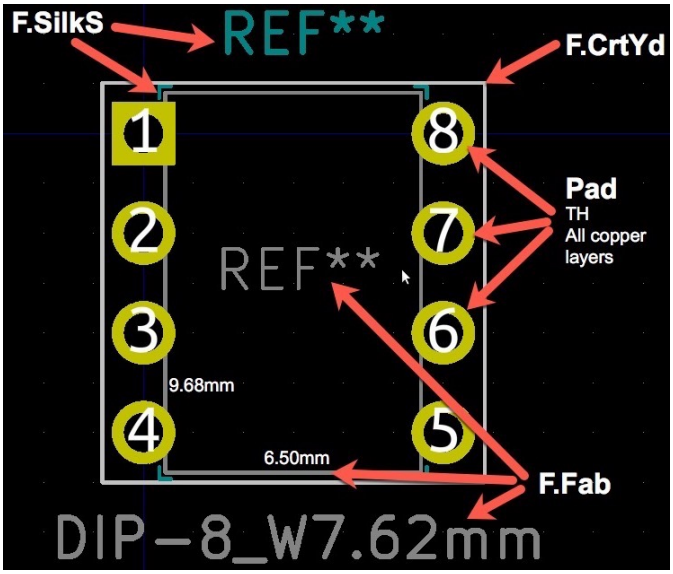


Figure 37.1: The elements and layers of a typical footprint.

To create the footprint in Figure 37.1, you will work through a process in which you place the elements in one layer. The real component for which you will design this footprint is shown in Figure 37.2.

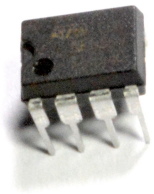


Figure 37.2: You will create a footprint for this component, the NE555N timer integrated circuit.

The component in Figure 37.2 has a standard DIP package with 4 pins on each side. The data sheet is available from its manufacturer. From the datasheet, you will need the mechanical data illustration towards the end of the document. For your convenience, I have included this illustration in Figure 37.2.

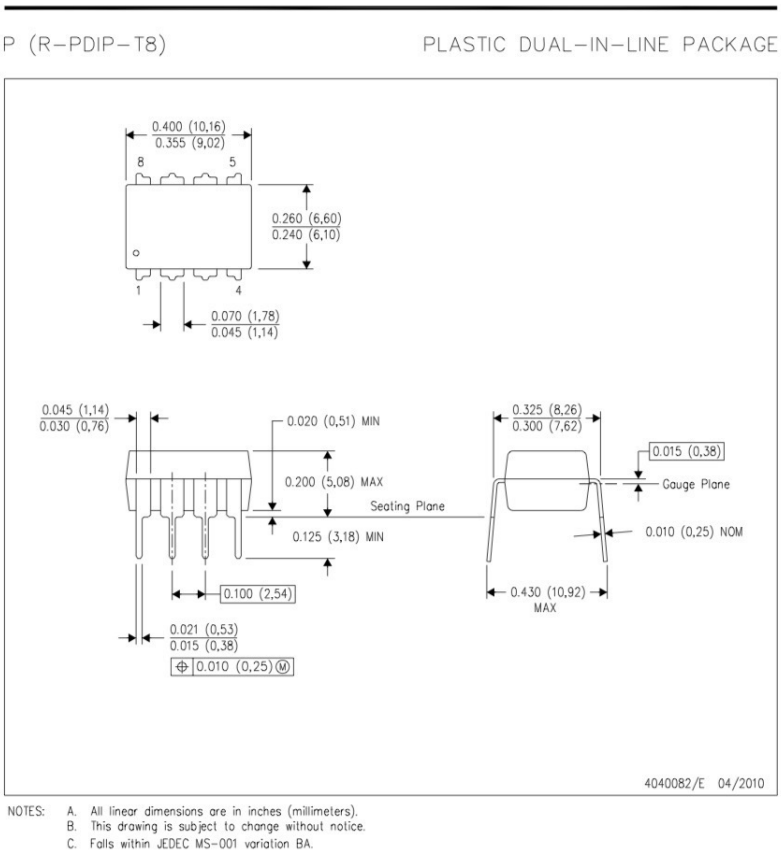


Figure 37.2: The mechanical characteristics of the R-DIP-T8 package.

You will create this footprint by following this process:

1. In the front fabrication layer ('F.Fab') you will draw the outline of the footprint. The fabrication layers (front and back) are used by the manufacturer. Their elements do not appear in the end result.
2. In the top and bottom copper layers, you will draw the pads.
3. In the front courtyard layer ('F.CrtYd'), you will draw the external outline of the footprint. No other footprint will be allowed within this outline.
4. In the front silkscreen layer ('F.SilkS'), you will add text and graphics, such as the corners of the DIP package and the side where pin 1 is facing.

Let's begin. From the main KiCad window, click on the Footprint Library Editor button (Figure 37.3).

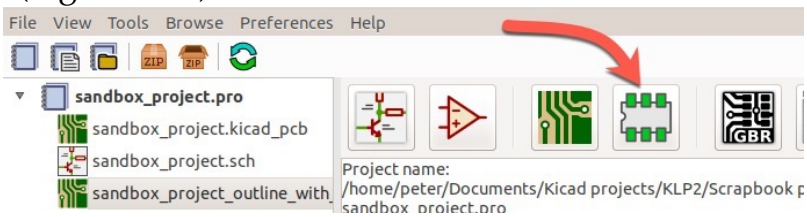


Figure 37.3: Start the Footprint Library Editor.

The Footprint Editor window will appear. From the File menu, select 'New Footprint' (Figure 37.4).

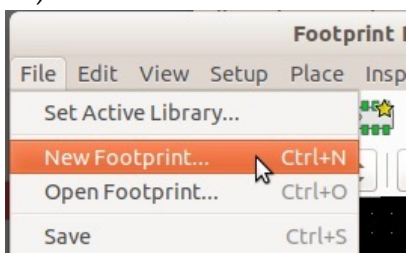


Figure 37.4: Create a new Footprint.

The Editor will prompt for a name for the new footprint. Since we are creating a footprint for a DIP component with 8 pins and 7.62 mm width (including the pins), type in this name: 'DIP-8_W7.62mm_PD'. I added my initials, to differentiate from other DIP-8 footprints that might be available in other libraries. Of course, use your own initials. Click on Ok to dismiss the dialog. The Editor will show an almost blank sheet. Its only content is two text blocks, one in the front fabrication area (the name of the footprint), and one in the front silkscreen area ('REF**').

Front Fabrication layer ('F.Fab') - Outline

Continue with the front fabrication layer. Select 'F.Fab' from the layers manager, and use the polygon tool to draw a rectangle in the dimensions of the DIP package. Those dimensions appear in the documentation. Working in millimetres, you should draw a rectangle that is 6.60 mm in width and 10.16 mm in length. You will need to adjust the grid to make it easier to achieve those dimensions, or at least as close to them as you can get. I set my grid to 0.1270 mm and I was able to create a rectangle with the exact 6.60 mm x 10.16 mm dimensions. Also, change the cursor shape so that the crosshairs extend to the edges of the sheet, and use the *dx* and *dy* values in the status line to know the length of each segment of the rectangle as you draw it. In Figure 37.5, the arrows point to the tools that you use and functions you should enable to draw the outline of the footprint. The exact point where you start drawing is not important.

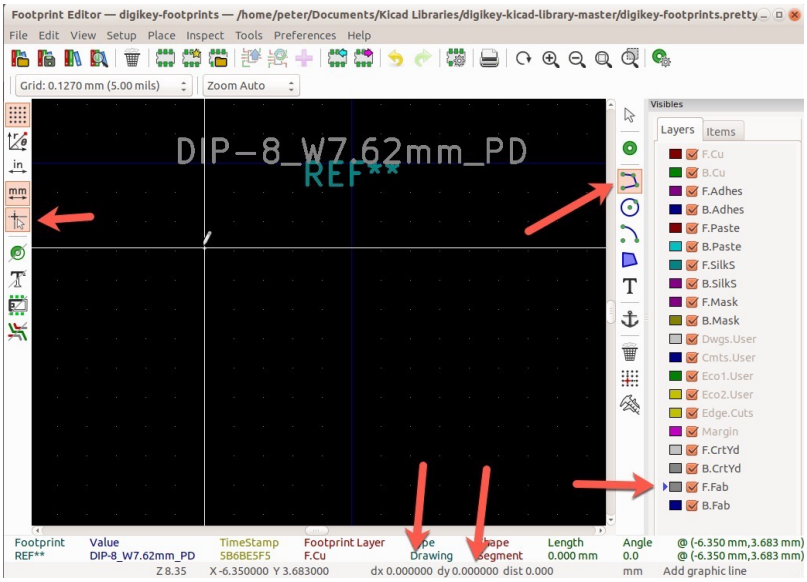


Figure 37.5: Using the polygon tool, start drawing the rectangle.

Draw the first horizontal line of the outline to a length of 6.60 mm. In Figure 37.6 notice that the *dx* value is 6.60 mm. Click at that point, and continue with the first vertical line.

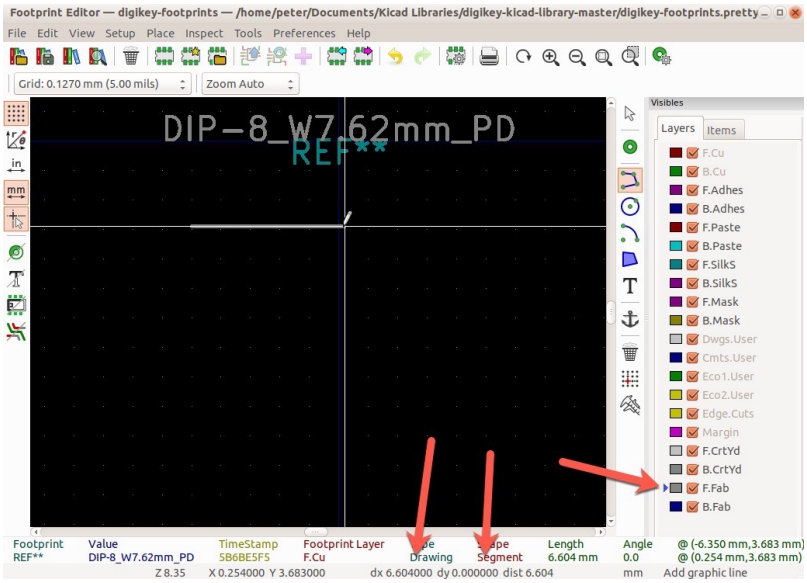


Figure 37.6: The first line, at 6.60 mm, is complete.

Extend the vertical line to 10.16 mm. Use the crosshairs to help you create a perfect 90-degree angle between the two lines (Figure 37.7).

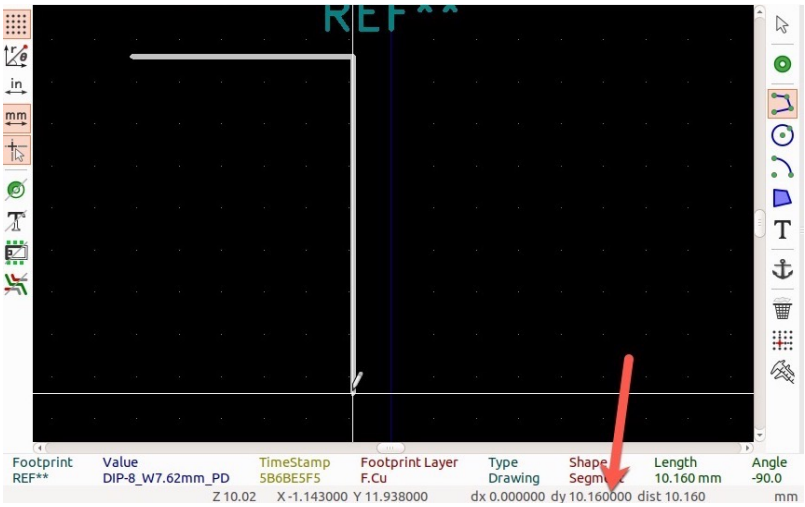


Figure 37.7:The first vertical line is 10.16 mm in length.

Complete the rectangle so that in the end you have something like the example in Figure 37.8.

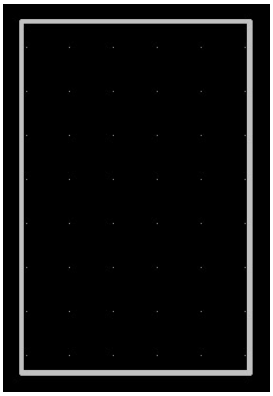


Figure 37.8:The completed outline in the F.Fab layer.

Pads

Continue with the placement of the pads. As per the data sheet's mechanical specifications, the pad centres must be 2.54 mm apart. The pin diameter is 0.25 mm, so the pad drill must be slightly larger than that. The specification doesn't give us the exact offset of the pins from the body of the device, so we will have to use our judgement or a calliper. I used my calliper to find that the offset is around 1 mm. With this information, let's go ahead and place the pads. Keep the grid size to 0.127 mm since this works well with the 2.54 pad pitch we are working with. That is because 2.54 is a multiple of 0.127.

From the right toolbar, click on the pad tool (). Follow these steps:

1. Move the pad to the top left of the outline.
2. Align the crosshair centre exactly on the vertical line, around 1 mm from the corner.

3. Press the space bar to zero the *dx*, *dy* and *dist* values of the status bar.
4. Move the pointer slightly to the left, to separate the pad from the outline. At *dx* at 0.63 mm, I think this distance is good. It is smaller than the 1 mm that I measured with my calliper, but since the device pins are flexible, I opt to a narrower rather than a wider footprint.
5. Ensure that dx is still at 0.63 mm and click to commit the pad in place. After clicking your mouse, do not move it until you press the space bar again to reset *dx* and *dy*. This will allow you to measure the distance between this pad and the next. Don't worry about the negative sign in the dx value, we are only interested in absolute values when we measure distances for the purposes of creating this new footprint.

The result is similar to what you can see in Figure 37.9.

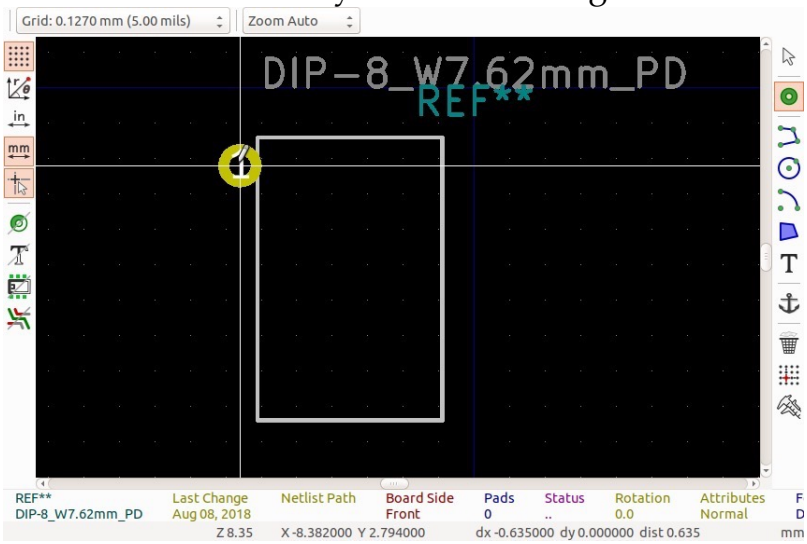


Figure 37.9: The first pad is in place.

The pad tool is still enabled, and the second pad is attached to the cursor, waiting to be placed. Hopefully, you reset *dx* and *dy* as soon as you clicked to commit pad 1. If you didn't, move the cursor over pad 1, and press the spacebar to reset the counters. Then, follow this process:

1. Move the mouse downwards making sure that *dx* is always zero.
2. Look at the *dy* counter, and stop moving the mouse when it reads '2.54 mm'.
3. When *dy* becomes 2.54 mm, click your mouse to place pad 2 in position.
4. Press the space bar to reset *dy* to zero.
5. Repeat this process until you have all four pads on the left of the footprint's outline.

When you place pad 4 in position, you must move the mouse across to the other side of the outline. To ensure that pad 5 is exactly horizontal from pad 4, place your mouse pointer exactly over pad 4, and press the spacebar. Then move the mouse to the right, making sure that *dy* is always zero. Place pad 5 in position so that it is exactly 0.63 mm from the right vertical line and click to commit it in place. Your footprint should look like the example in Figure 37.10.

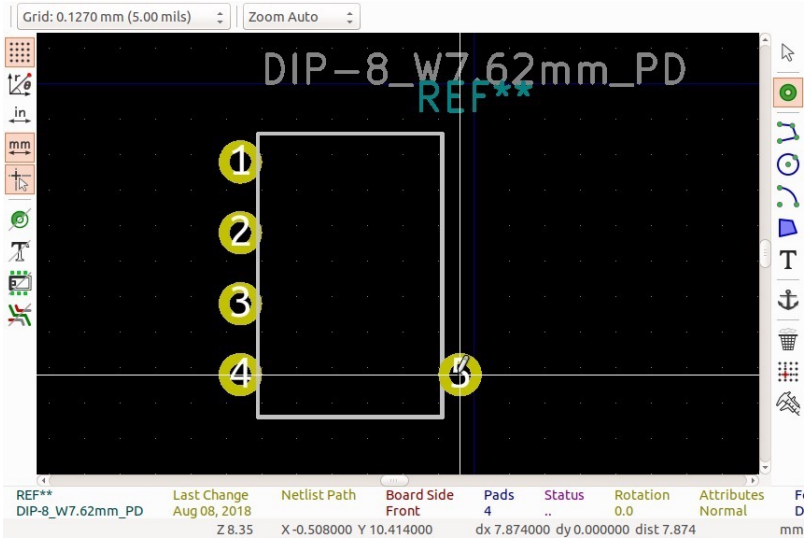


Figure 37.10: Left side has four pads, continue on the right side.

Continue to place pads 6, 7 and 8 in the same way. Eventually, your footprint will look like the example in Figure 37.11.

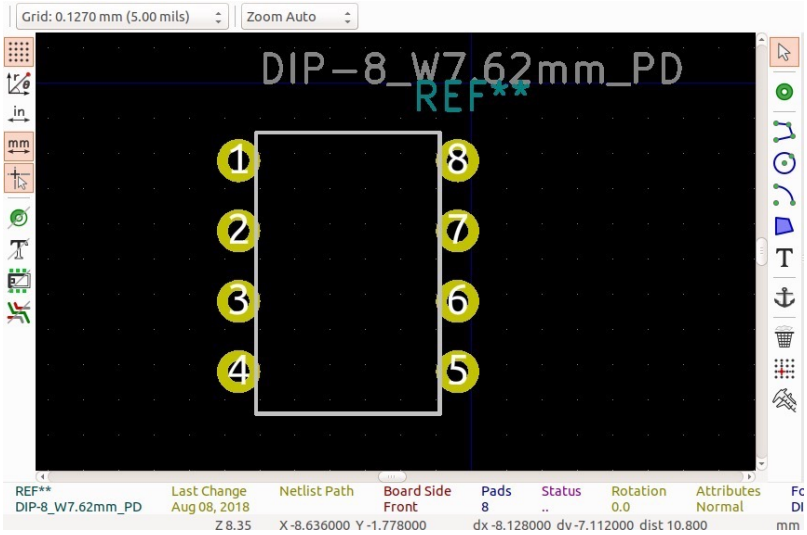


Figure 37.11: All footprints in place.

Before you continue work in the other layers, you should check that the footprint drills are appropriate for the pin size of the physical device. You should also change the pad type of pad 1 to rectangular instead of circular. This is due to a convention that holds that the first pin of an integrated circuit should be square to make it possible to identify the correct orientation of the

chip during assembly. We will do this now, but we will also add graphics in the silkscreen to reduce the risk of error in assembly.

To configure a pad, use the pad's Properties window. Place your mouse cursor over the first pad, and type 'E'. This will bring up the pad's Properties window. As you can see in Figure 37.12, you should change the Pad shape to rectangular. You can also confirm that the hole size is 0.762 mm, which is sufficiently larger than the pin diameter, so there is no need to change this. You can also confirm that for this pad, copper will be poured in all copper layers. If you have a good reason for doing so (perhaps you are designing a single layer board that you want to etch at home?), then you can change this setting to a bottom copper layer. If you were creating an SMD pad, you could specify the top or bottom copper layer.

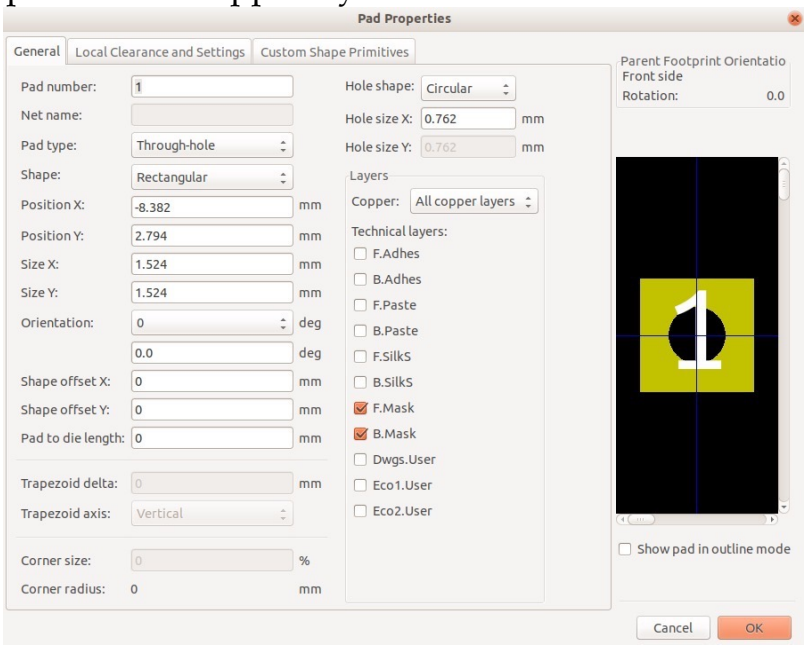


Figure 37.12: Pad 1 is now a square.

You can leave the rest of the pads as they are, no changes are needed. Your footprint should look like the example in Figure 37.12.

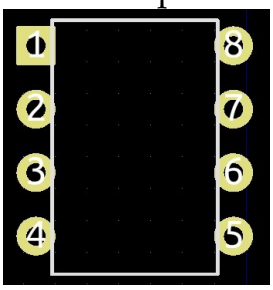


Figure 37.12: The footprint, with fabrication layer and pads completed.

Front Courtyard layer ('F.CrtYd')

Let's continue with the front courtyard layer ('F.CrtYd') where you will create the outline of the boundary for the footprint. From the Layer Manager,

select 'F.CrtYd'. Your objective is to draw a rectangle around the footprint that encloses the pads and the footprint outline. Allow sufficient space around the pads to ensure that they cannot overlap with other footprints. Select the polygon drawing tool and start drawing from the top right corner of the footprint (Figure 37.13).

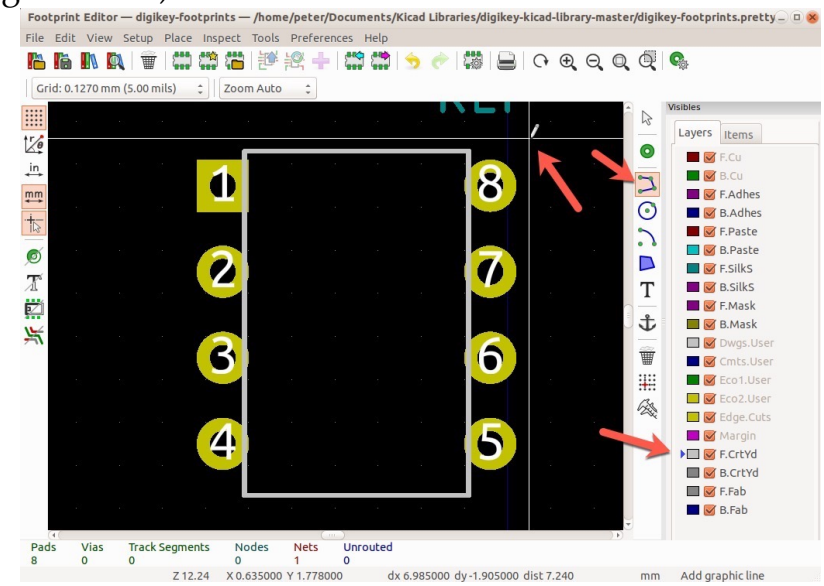


Figure 37.13: Drawing in the front courtyard layer.

Draw the rectangle around the footprint until the polygon is fully enclosed. In the end, your footprint should look like the example in Figure 37.14. The rectangle around the footprint is the boundary as defined in the front courtyard layer.

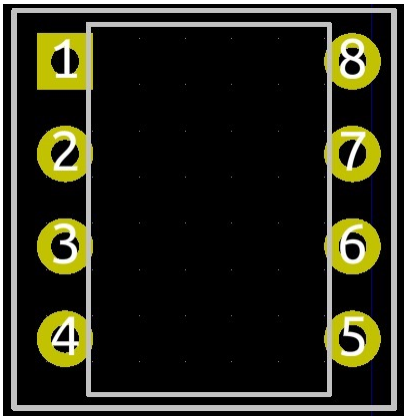


Figure 37.14: The line around the footprint is the front courtyard outline.

Front Silkscreen

While your footprint is now functionally ready, you should spend a few more minutes to add informational text and graphics in the front silkscreen ('F.SilkS') layer. Since you are working on the footprint of an integrated circuit, at the very least you should mark the location of pin 1. Let's do that now. Select 'F.SilkS' from the Layer Manager. Select the polygon tool. Draw four

lines to mark the outline of the IC on the board, like in the example of Figure 37.15.

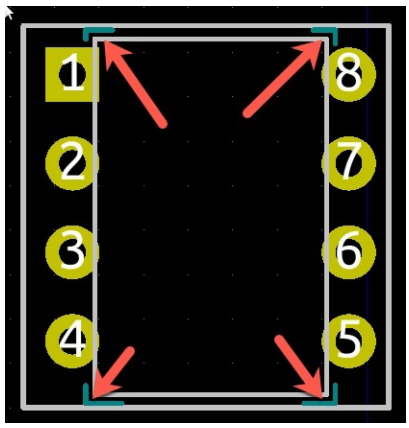


Figure 37.15:In the F.SilkS layer, mark the edges of the footprint.

Also draw a circle that indicates the position of pin 1, using the circle tool. The end result is in Figure 37.16.

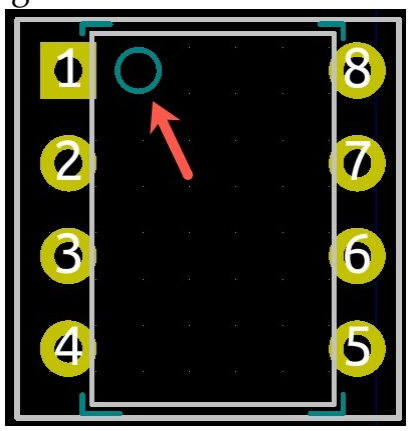


Figure 37.16: A circle in the F.SilkS layer indicates the position of pin 1.

Tidy up

You are almost finished with this design. Time to tidy up before saving the new footprint. Use the 'M' hotkey to move the text blocks over and below the footprint. Select the complete footprint, including the text block, and move it over the center of the axes. The final footprint is in Figure 37.17.

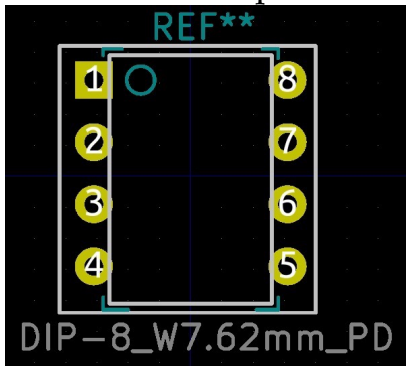


Figure 37.17: The final custom footprint.

Save the footprint

The last thing you must do before you can use your custom footprint is to save it. If you already have a folder for your custom footprints with the '.pretty' extension, you can use that to store the new footprint. If not, create this folder now. Then, click on the 'Export' button from the top toolbar (Figure 37.18).

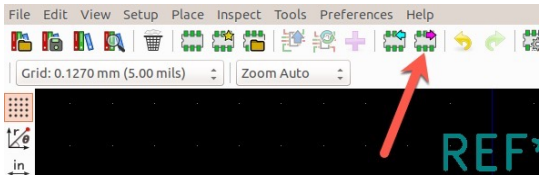


Figure 37.18: Use the Export button to save the new footprint.

This will bring up the Export Footprint window. Navigate to your .pretty folder, give your new footprint a name (or accept the default, as I do), and click on Save (Figure 37.19).

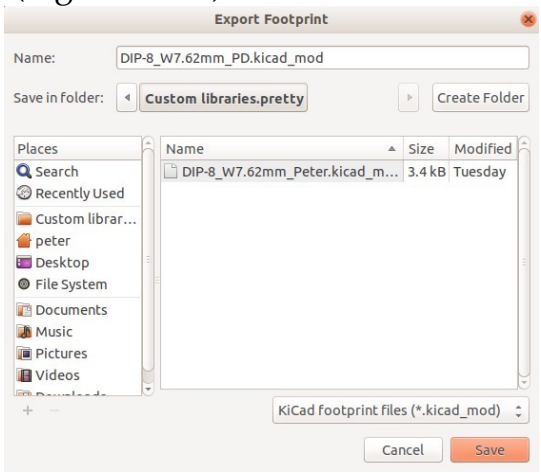


Figure 37.19: Save the new footprint.

Test the footprint

Let's go back to Pcbnew to test your new footprint by adding it to the sheet. If the .pretty folder that contains your new footprint is not in the libraries table, add it now (Figure 37.20). You can learn how to do this in the relevant recipe.

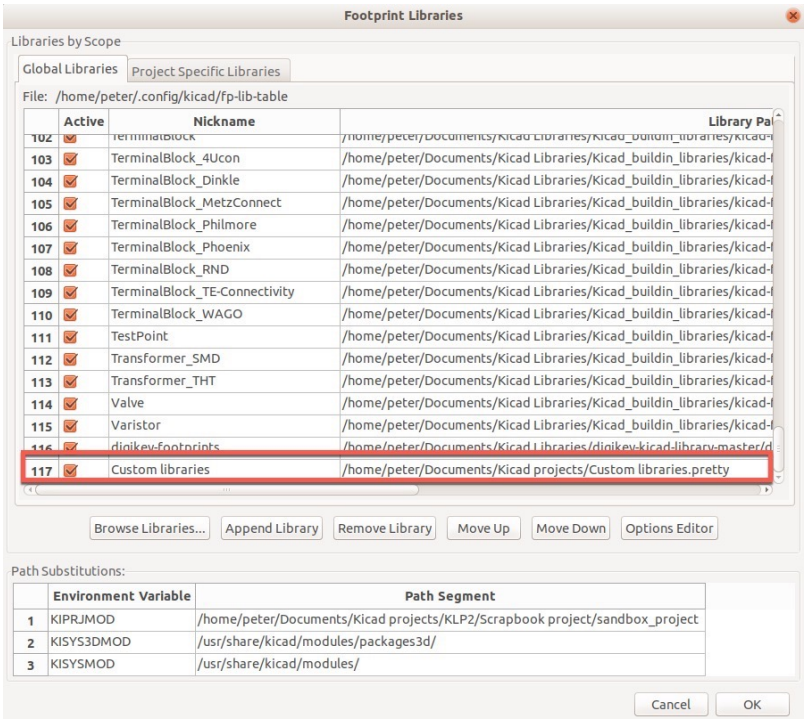


Figure 37.20: My custom .pretty folder is in the libraries table.

Once that is done, use the 'O' hotkey to add a new footprint. Use the browser to navigate the list of libraries. Find your custom library, and in it, you will find your new footprint (Figure 37.21).

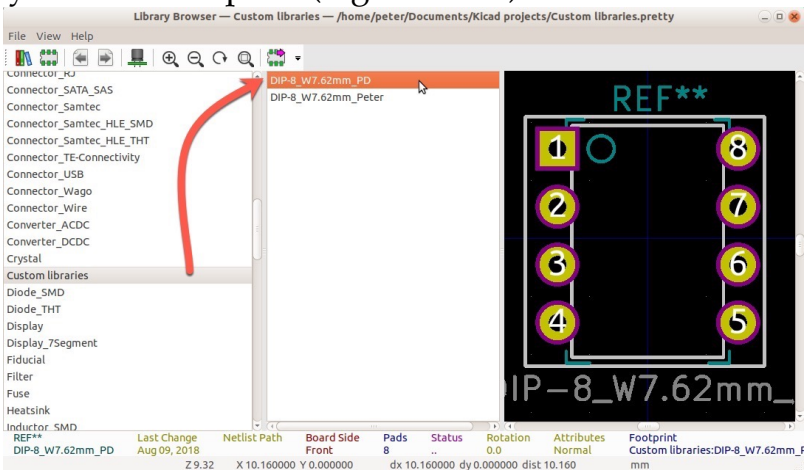


Figure 37.21: Select your new footprint from the library browser.

Double click on the footprint, and Pcbnew will place it on the sheet (Figure 37.22).

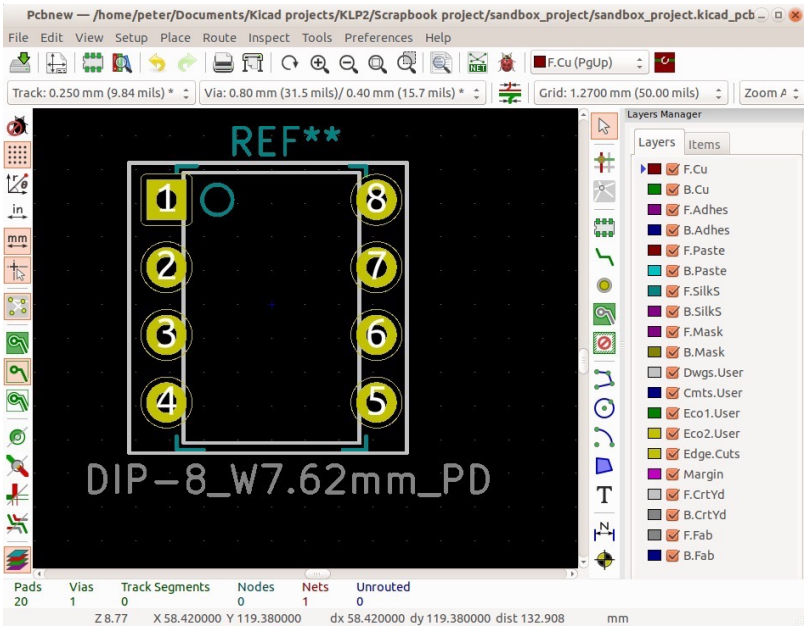


Figure 37.22: Your new footprint in the Pcbnew sheet.

Well done, this was a long process. You now have a custom footprint that you can use in your layouts as you do with any other footprint.

38. Creating a new footprint - using the footprint wizard

In the previous recipe, you learned how to create a custom footprint. The footprint you designed was for a standard 8-pin DIP integrated circuit. As you will probably agree, designing this footprint required a significant amount of effort. For devices like that, and several others, the KiCad Footprint Editor offer a wizard that can cut the total time needed to create something like the footprint from the previous recipe by 90%. In this recipe, you will learn how to use this wizard to create the same DIP-8 footprint as the one from the previous recipe which just a few keystrokes and clicks.

To make the most out of this recipe you should first read the previous one.

From the main KiCad window, start the Footprint Manager. In the Footprint Manager, click on the 'New footprint using footprint wizard' button (Figure 38.1).

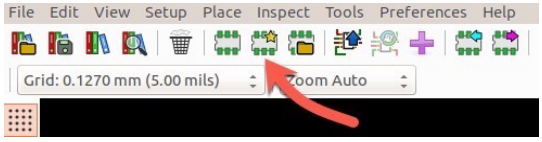


Figure 38.1: Start the footprint wizard.

The footprint wizard window will appear. At the moment it contains no information because we have not selected one of the available footprint generators. To do so, click on the 'Select wizard script to run' button (Figure 38.2).

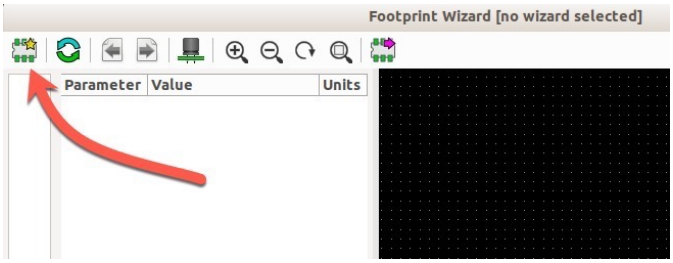


Figure 38.2: Select a wizard script to run.

The Generators window will appear. It contains a list of the available wizard scripts. There is a good variety of scripts, and you can expect more in the future. Feel free to experiment with each one to gain an understanding of how they work. For the purposes of this recipe, select the S-DIP script, and click 'Ok' to select this script.

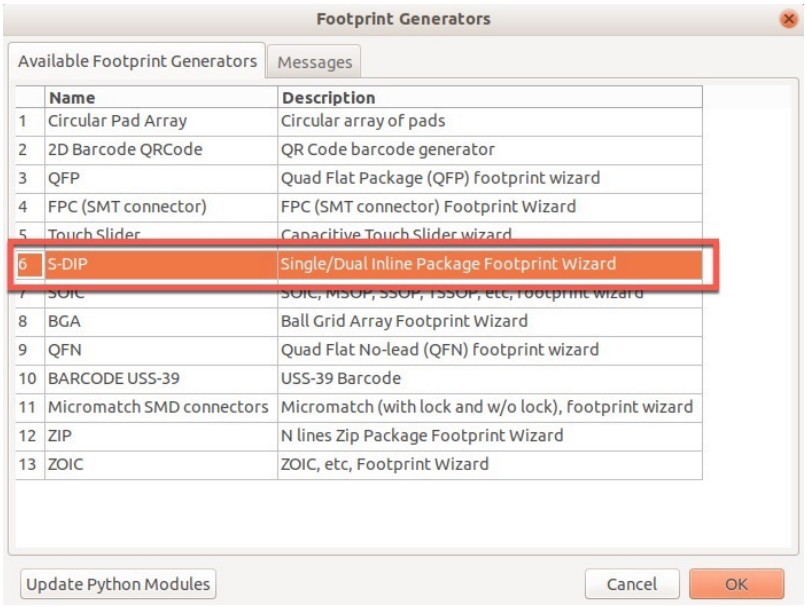


Figure 38.3: Select one of the wizard scripts.

The wizard window is now populated with the footprint parameters and its preview. All the parameter values are editable. To create the 8 pin package, simply type '8' in the pad count field. The rest of the parameters are appropriate as they are, but you should check again. The drill size is adequately large at 0.8 mm to accommodate the pins. The pad pitch is also good at 2.54 mm (Figure 38.4).

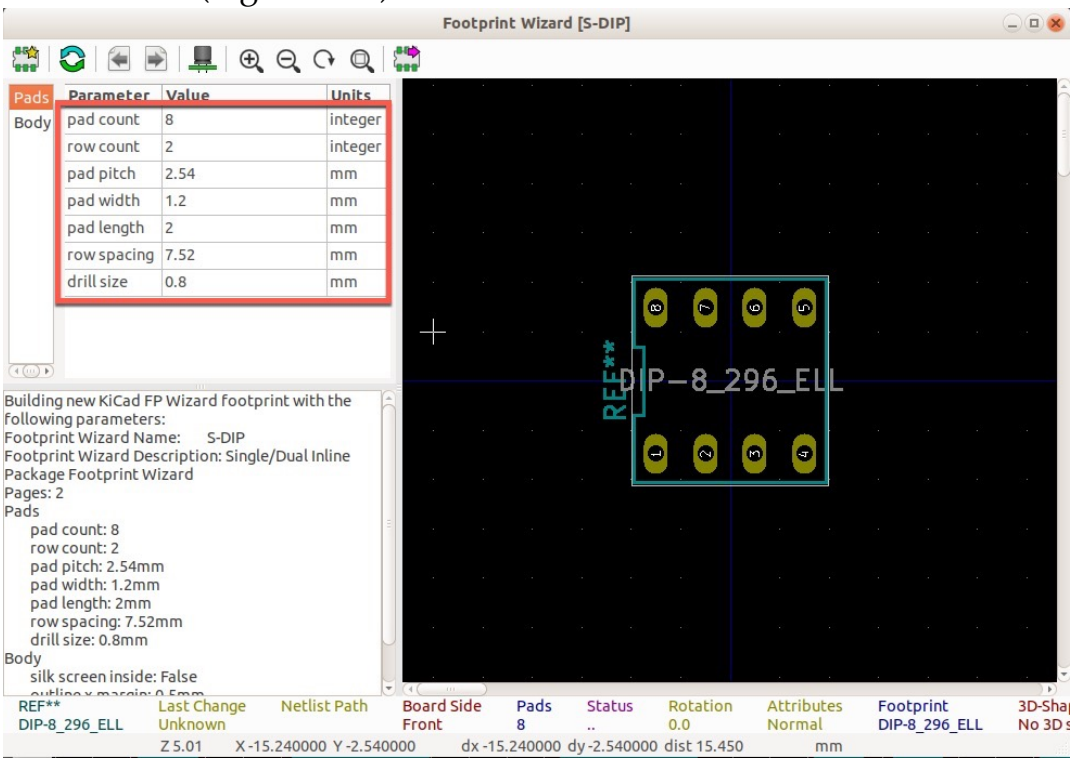


Figure 38.4: The footprint parameters in the wizard.

Click on the Body tab to inspect the footprint body parameters. The default values are appropriate. Feel free to change them to see their effect on

the footprint. When you are ready, click on the 'Export footprint to editor' button (Figure 38.5).

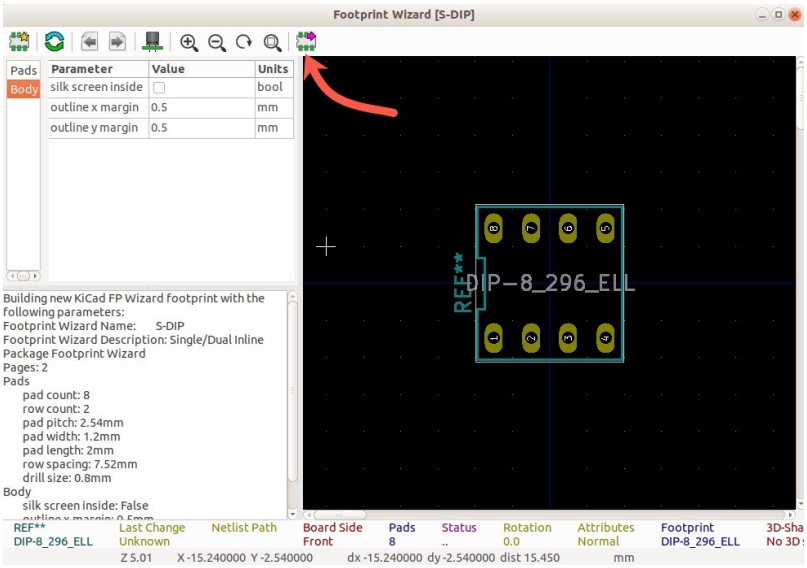


Figure 38.5: The Body parameters.

The footprint that you just created with a few clicks in the wizard is now in the footprint editor (Figure 38.6). You can continue with other customisation tasks there.

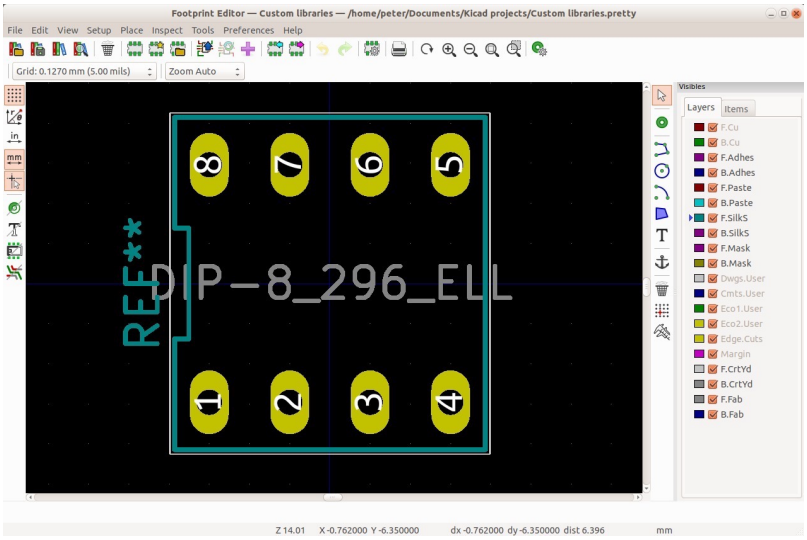


Figure 38.6: The new footprint is in the editor for additional work.

A couple of things you can do:

1. Change the type of pad 1 from oval to rectangular
2. Change the type of the rest of the pad from oval to circular (I did not do that)
3. Rotate the footprint by 90 degrees.
4. Move the text blocks.

To change the pad types, bring up the properties window for each pad. You can find instructions on how to do this in the previous recipe (hint: use the 'E' key).

In Figure 38.7 you can see what my wizard-generated custom footprint looks like after I completed changes 1, 3 and 4.

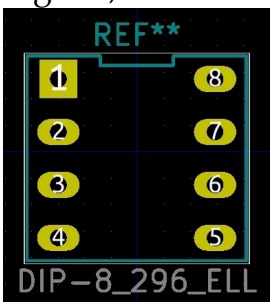


Figure 38.7: The final footprint; most of the work was done by the wizard.

As you can see, using the wizard radically reduces the amount of time needed to create a footprint.

39. Modifying an existing footprint

In this recipe, you will learn how to modify an existing footprint. You may want to do something simple, like relocate silkscreen graphics, or change the drill size of the pads. Whichever the case may be, the process is the same.

Start by finding a footprint to edit. Let's assume that you know which library this footprint is in, and you know its name. If you don't, use the footprint browser in Pcbnew to find it first.

From the main KiCad window, start the Footprint Manager. In the Footprint Manager, use the 'Import footprint' button to find and import the footprint (Figure 39.1). If the Editor already contains a footprint, you will get a prompt asking for confirmation to discard it. Consider saving it if you haven't done so already, and continue.

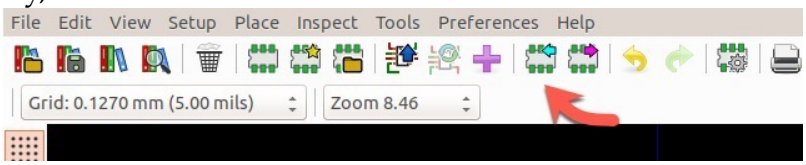


Figure 39.1: Import the footprint you want to edit.

In the browser, navigate to the .pretty folder where the footprint is, and select it. In my example, I navigated to the location of the Digikey footprints directory, and I have selected to modify the LED_3mm_Radial footprint (Figure 39.2).

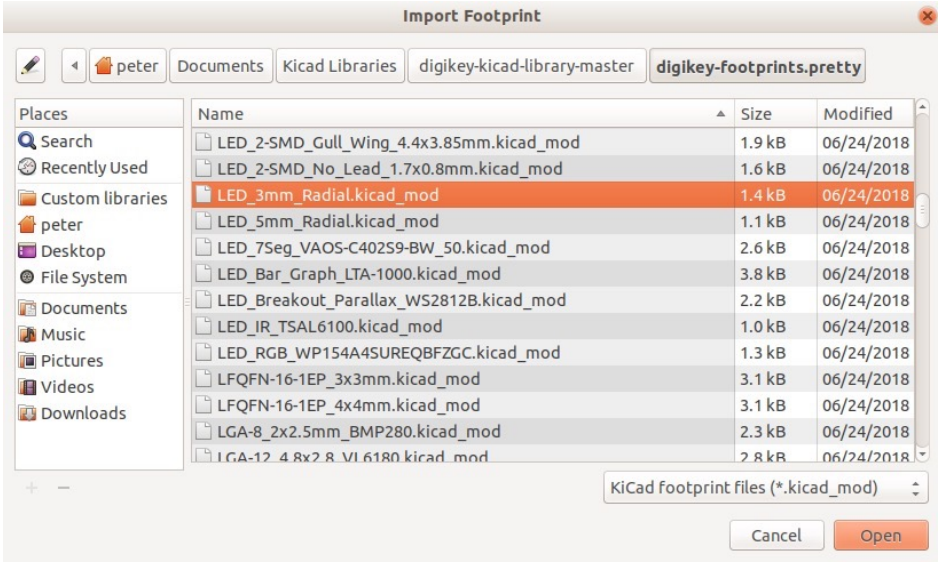


Figure 39.2: Navigate to the footprint's location and select it.

Click 'Open' to import the footprint. The footprint will appear in the Editor sheet. Let's make a couple of simple modifications:

1. Add the block text 'LED' in the silkscreen.
2. Mark the cathode pad with the letter 'C' in the silkscreen. Note: According to IEEE 315, Clause 8.4, the letter “K” should be used to mark the Cathode, while the letter “C” is reserved for the Collector.[21]
3. Reduce the drill size to 0.8 mm for both pads.

Let’s start with the text. Select the 'F.SilkS' layer from the Layers Manager. Select the Text tool. Click on the right side of the LED footprint; the text properties will appear. Type 'C' in the text box and click on 'Ok'. Position the text block right next to the symbol cathode (the straight section of the circle), and then click below the footprint to create the second text block. Type 'LED' in the text box, and click 'Ok' to create the new block. Click on the sheet to commit the new block.

Continue with the pad drill size. Hover the mouse pointer over pad 1 and type 'E' to bring up the pad properties. In the pad properties window,

---

[21] For your convenience, I have included the text of IEEE 315, Clause 8.4.

8.4 Rules for Drawing Style 1 Symbols

To draw a device symbol, start at an electrode whose polarity is known (usually an emitter) and proceed along the

device, showing all of its regions individually. Finally, indicate ohmic connections where required.

NOTE — 8.4A: Numbers, letters, and words in parentheses are to correlate illustrations in the standard; they are not intended to

represent device terminal numbering or identification and are not part of the symbol as shown in items 8.5, 8.6, 8.10, and

8.11.

Name of Terminal

Letter

Anode A

Base B

Collector C

Drain D

Emitter E

Gate G

Cathode K

Source

Main terminal

S

*

Substrate (bulk)

T

U

*Used with bidirectional thyristors. The terminals are differentiated by numerical subscripts 1 and 2, T 1 being the terminal to which the gate trigger signal is referenced, if applicable.

change the hole size X value to 0.8 mm (Figure 39.3). Then click 'Ok' to make the change effective.

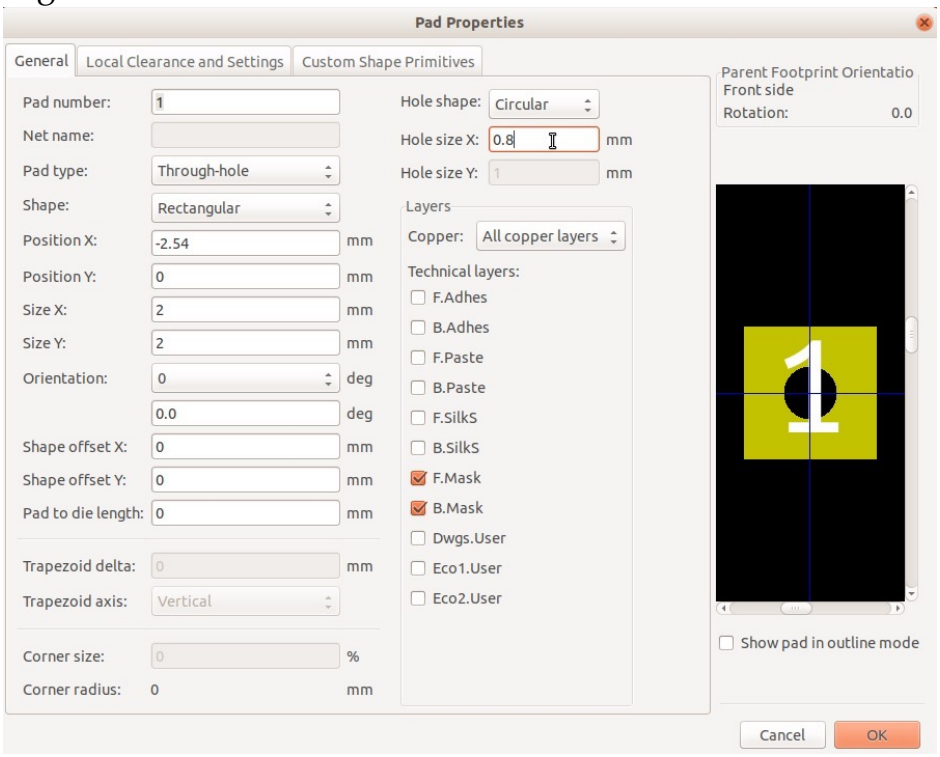


Figure 39.3: Modify the drill size for pad 1.

Do the same for pad 2. The modified footprint should look like the example in Figure 39.4.

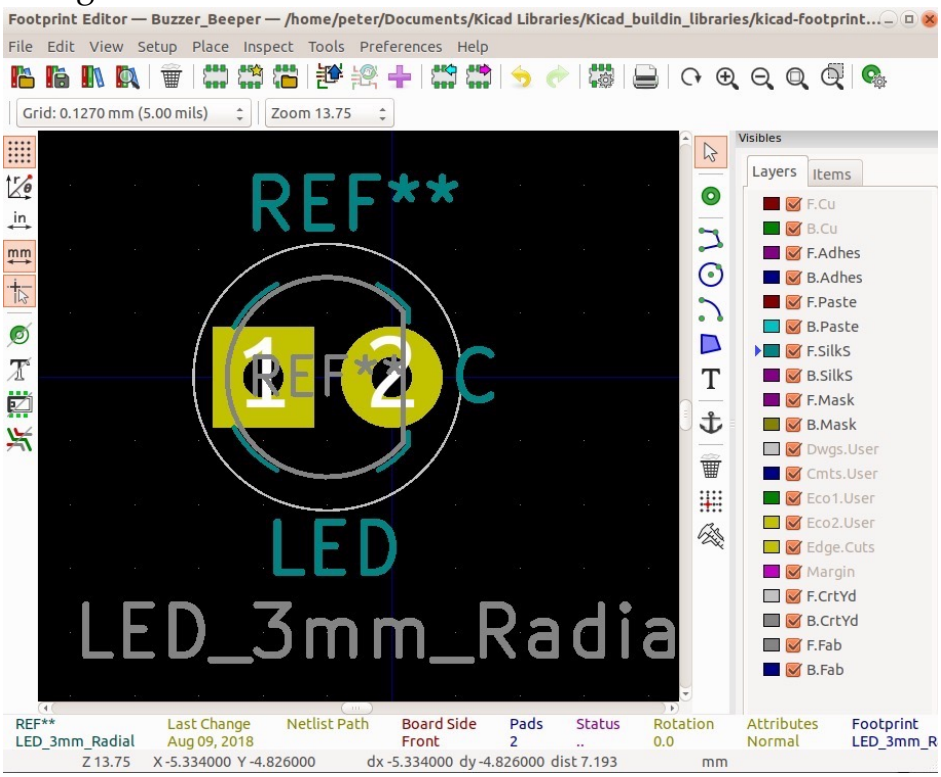


Figure 39.4: The modified footprint.

Before you can use the modified footprint you must save it. Use the Export Footprint button (the one with the red arrow) to do this. Navigate to the location of your custom footprints .pretty folder, and store the modified footprint there (Figure 39.5).

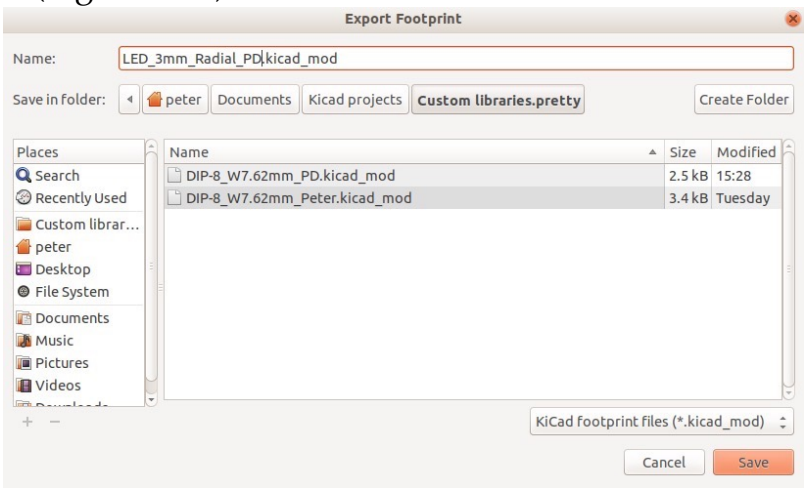


Figure 39.5: Store the modified footprint in the custom footprints folder.

You are now able to use your modified footprint in your layouts.

40. Using an autorouter

The autorouter that was included in KiCad 4 has been removed in KiCad 5. It is possible that a new autorouter will be added in the future. Until then, you can use an external autorouter. In this recipe, you will learn how to use FreeRouting, an open-source autorouter.

The online project home for FreeRouting is at freerouting.org. There, you will find information on how to install it on your computer and usage documentation. FreeRouting is a stand-alone program with many capabilities. If you are planning to design a large PCB, it may be worth spending some time to study the FreeRouting documentation. In this Recipe, you will learn how to use FreeRouting to autoroute a simple PCB that you are working on in Pcbnew.

In my experience, the easiest way to install FreeRouting is by installing yet another application, LayoutEditor. LayoutEditor is a general editor that is used in micro and nano-electronics. It happens to include a binary version of FreeRouting and installers for many operating systems. This means that you will not have to worry about downloading the source code of FreeRouting and compiling it for your operating system. For this reason alone, it is worth downloading and installing LayoutEditor.

Start by going to the LayoutEditor download page, and download the version for your operating system. Once you have it on your computer, install it. When the installation is complete, go to the installation directory of the LayoutEditor. On my Ubuntu computer, this directory is /opt/layout. Navigate to the bin folder and find a file named 'freeRouting.jar'. This file contains the FreeRouting application. You can copy the file to a location that is convenient for you to access, or create a shortcut.

FreeRouting is a Java application, so if you don't have the Java Runtime Environment, you should install it now. On Ubuntu, you can install the JRE through a terminal window by typing:

```
$ sudo apt install default-jre
```

Now that you have access to FreeRouting, let's use it.

Open an unrouted project in Pcbnew. You can see my example in Figure 40.1.

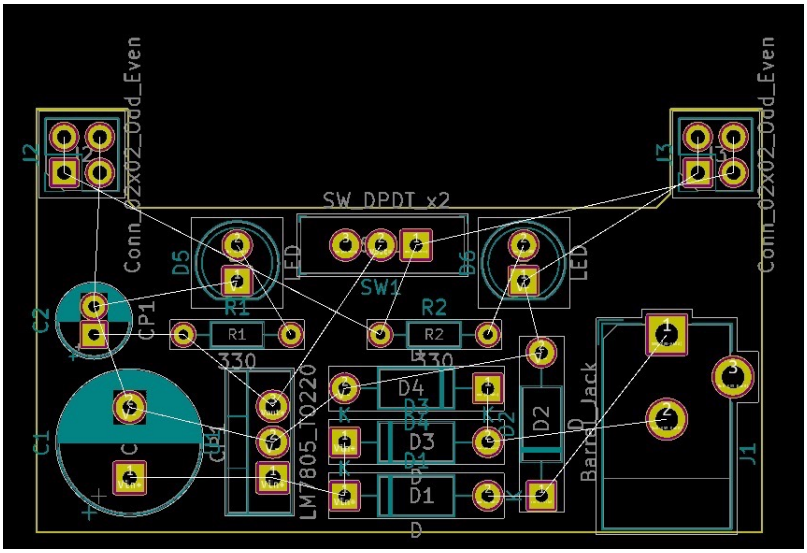


Figure 40.1: We will use FreeRouting to autoroute this PCB.

You must export a Specctra DSN file that contains the information that FreeRouting needs in order to do the routing. Create the DSN file by clicking on File, Export, 'Specctra DSN...' (Figure 40.2). A dialog box will ask for a location and file name for this file.

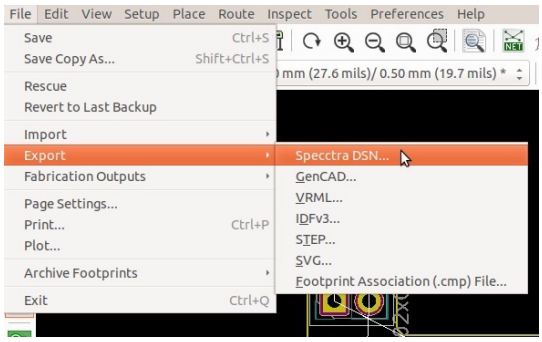


Figure 40.2: Export the DSN file for FreeRouting.

Proceed with FreeRouting. Run freeRouting by double-clicking on the .jar file. The Java Runtime Environment should execute the program assuming it is installed correctly. FreeRouting will ask you to load the DSN file, so locate it and load it. Eventually, FreeRouting will display your board with the various footprints and its layout exactly as you see it in Pcbnew. To run the autorouter, click on Routing, followed by Autorouting (Figure 40.3).

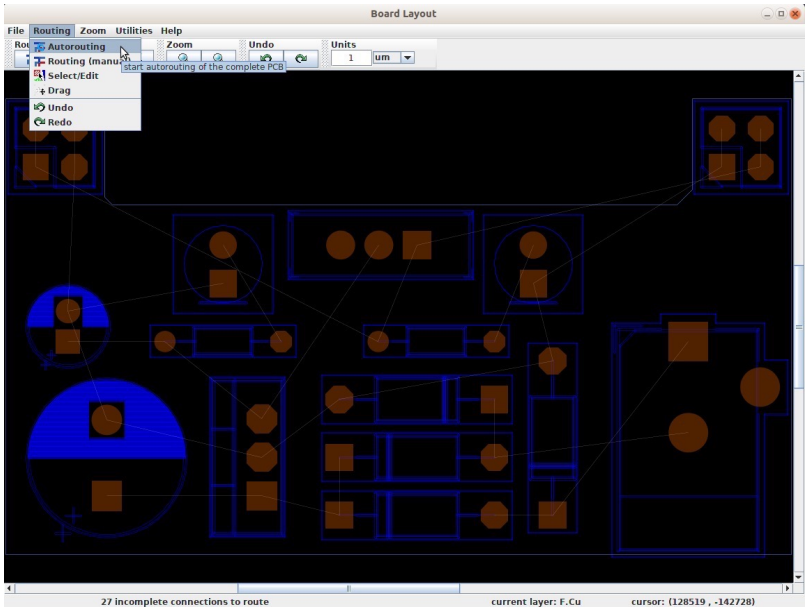


Figure 40.3: FreeRouting showing an unrouted board.

After a few seconds, FreeRouting will create the routes and show them in red and blue colour, depending on the layer that the route exists in. You will need to import this version of the board back to Pcbnew to continue with the work there. To export from FreeRouting, click on File, Export, Export Specctra Session File. You can close FreeRouting and return to Pcbnew.

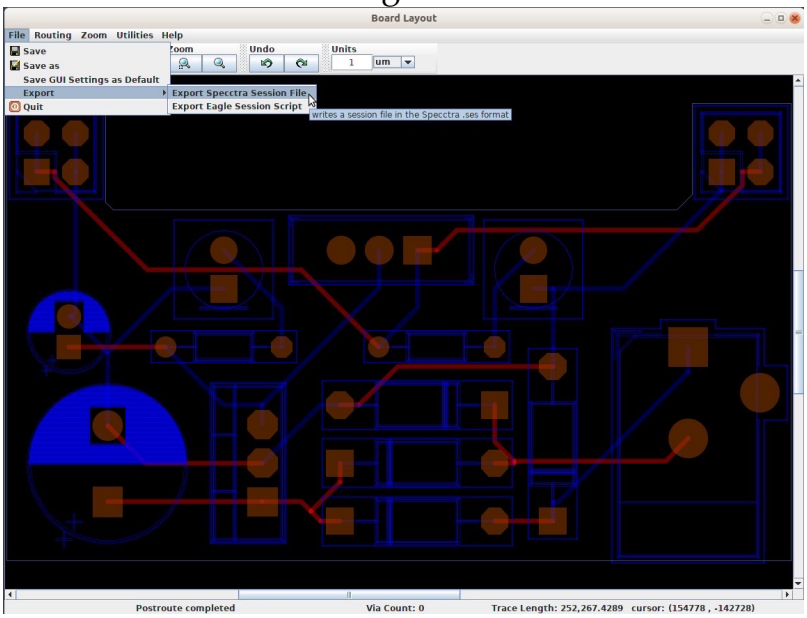


Figure 40.4: Exporting the routed PCB from FreeRouting.

In Pcbnew, go to File, Import, 'Specctra Session...' and select the file with the .ses extension that FreeRouting created (Figure 40.5).

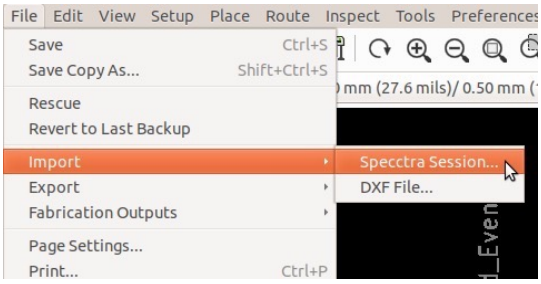


Figure 40.5: Importing the SES file.

Pcbnew will display the fully routed board, as you can see in Figure 40.1.

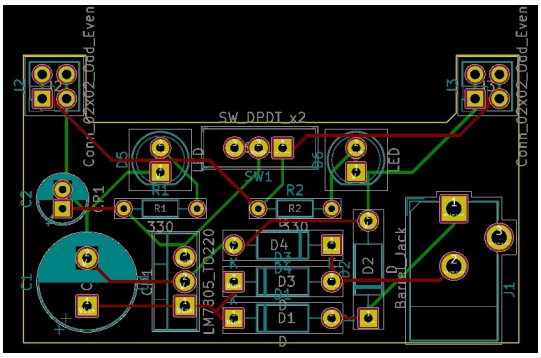


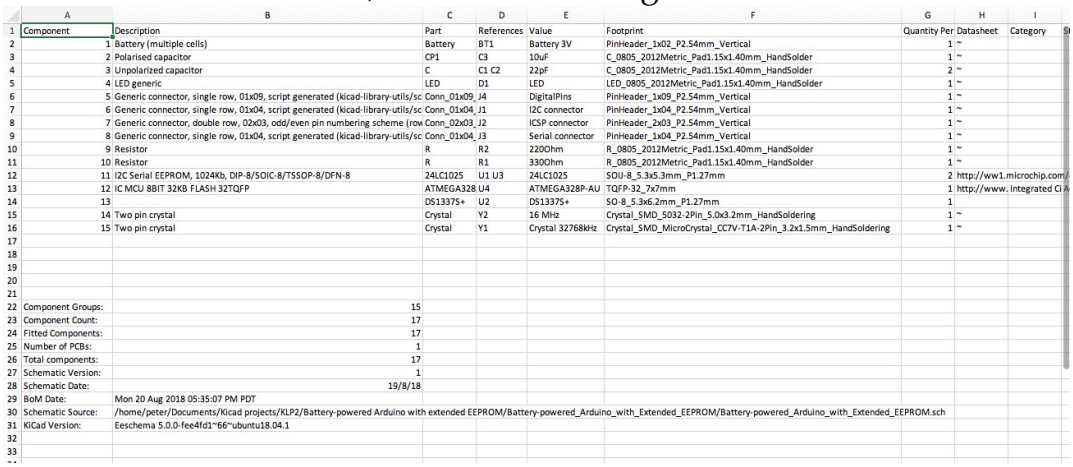
Figure 40.1:The fully autorouted board.

While the autorouter did its job, there is still work to do. For example, you will need to pay attention to power traces and confirm or edit their widths. You must also run the Design Rules Check to ensure that none of the design rules are violated.

Should you use an autorouter, or route everything manually? It depends. There does not seem to be a consensus on this topic. Many people opt to use it; others prefer manual routing. The newer versions of KiCad, starting with KiCad 4, introduced powerful interactive routing tools which make manual routing fast. I prefer to use the autorouter for larger boards because of its speed, and then use interactive routing to make any changes necessary.

41. How to create a bill of materials (BoM)

In KiCad, you can create a Bill of Materials[22] (BOM) report via third-party plugins. In this recipe, you will learn how to use one of the plugins to create a BOM in CSV format, like the one in Figure 41.1.



	A	B	C	D	E	F	G	H	I
1	Component	Description	Part	References	Value	Footprint	Quantity Per	Datasheet	Category
2	1	Battery (multiple cells)	Battery	BT1	Battery 3V	PinHeader_1x02_P2.54mm_Vertical	1	~	
3	2	Polarised capacitor	CP1	C3	10uF	C_0805_2012Metric_Pad1.15x1.40mm_HandSolder	1	~	
4	3	Unpolarized capacitor	C	C1 C2	22pF	C_0805_2012Metric_Pad1.15x1.40mm_HandSolder	2	~	
5	4	LED generic	LED	D1	LED	LED_0805_2012Metric_Pad1.15x1.40mm_HandSolder	1	~	
6	5	Generic connector, single row, 01x09, script generated (kicad-library-utils/sc	Conn_01x09_	J4	DigitalPins	PinHeader_1x09_P2.54mm_Vertical	1	~	
7	6	Generic connector, single row, 01x04, script generated (kicad-library-utils/sc	Conn_01x04_	J1	I2C connector	PinHeader_1x04_P2.54mm_Vertical	1	~	
8	7	Generic connector, double row, 02x03, odd/even pin numbering scheme (row	Conn_02x03_	J2	ICSP connector	PinHeader_2x03_P2.54mm_Vertical	1	~	
9	8	Generic connector, single row, 01x04, script generated (kicad-library-utils/sc	Conn_01x04_	J3	Serial connector	PinHeader_1x04_P2.54mm_Vertical	1	~	
10	9	Resistor	R	R2	220Ohm	R_0805_2012Metric_Pad1.15x1.40mm_HandSolder	1	~	
11	10	Resistor	R	R1	330Ohm	R_0805_2012Metric_Pad1.15x1.40mm_HandSolder	1	~	
12	11	I2C Serial EEPROM, 1024Kb, DIP-8/SOIC-8/TSSOP-8/DFN-8	24LC1025	U1 U3	24LC1025	SOIJ-8_5.3x5.3mm_P1.27mm	2	http://ww1.microchip.com/	
13	12	IC MCU 8BIT 32KB FLASH 32TQFP	ATMEGA328	U4	ATMEGA328P-AU	TQFP-32_7x7mm	1	http://www. Integrated Ci A	
14	13		DS1337S+	U2	DS1337S+	SO-8_5.3x6.2mm_P1.27mm	1		
15	14	Two pin crystal	Crystal	Y2	16 MHz	Crystal_SMD_5032-2Pin_5.0x3.2mm_HandSoldering	1	~	
16	15	Two pin crystal	Crystal	Y1	Crystal 32768kHz	Crystal_SMD_MicroCrystal_CC7V-T1A-2Pin_3.2x1.5mm_HandSoldering	1	~	
17									
18									
19									
20									
21									
22	Component Groups:	15							
23	Component Count:	17							
24	Fitted Components:	17							
25	Number of PCBs:	1							
26	Total components:	17							
27	Schematic Version:	1							
28	Schematic Date:	19/8/18							
29	BoM Date:	Mon 20 Aug 2018 05:35:07 PM PDT							
30	Schematic Source:	/home/peter/Documents/Kicad projects/KLP2/Battery-powered Arduino with extended EEPROM/Battery-powered_Arduino_with_Extended_EEPROM/Battery-powered_Arduino_with_Extended_EEPROM.sch							
31	KiCad Version:	Eeschema 5.0.0-fee4fd1~66~ubuntu18.04.1							
32									
33									

Figure 41.1: A BoM created using the KiBoM Python script.

The plugin you will install is a Python script, available from Github. To install it, follow these steps:

1. Download the ZIP archive of the Github repository from https://github.com/SchrodingersGat/KiBoM (Look for the green 'Clone or download' button and select 'Download ZIP')
2. Extract the contents of the ZIP archive.

---

[22] An alternative to the term “Bill of Materials” (“BOM”) is “Parts List” (“PL”). This term is described in ANSI/ASME Y14.34 as:

Parts List (PL): a tabulation of all parts and bulk materials used in the item(s), except those materials that support a process and are not retained, such as cleaning solvents and masking materials.

3.20.1 Integral Parts List integral parts list: a parts list prepared and revised as part of an engineering drawing (see Fig. 4).

3.20.2 Separate Parts List separate parts list: a parts list prepared as a document separate from the engineering drawing to which it is associated, and one that may be revised independently of the drawing (see Figs. 9A and 9B).

When a separate list is prepared and a cover sheet is used, it shall include the mandatory information shown in Fig. 1; other information may also be included.

NOTE: Other terms previously used to describe a parts list are list of materials, bill of materials, stock list, and item list.

3. Copy the folder named 'KiBoM-master' to your KiCad documents folder, or anywhere you prefer to keep your work files.

4. Start KiCad, and open Eeschema.

5. Click on the BOM button from the top toolbar (Figure 41.2)

6. The Bill of Material window will appear. Click on the 'Add Plugin' button.

7. Navigate to the location of the KiBOM_CLI.py file and double-click on it to select it (Figure 41.3). You can accept the default name for the plugin.

8. The plugin is now installed. The Bill of Material window will show relevant information in the 'Plugin information' text box, and the command line path (Figure 41.4).

The command line path is of particular interest because you can control the output of the plugin by manipulating the command line arguments in it. You can find information about the valid arguments and ways to customise the plugin output at the plugin Github page.

Using the default configuration, however, produces a complete BoM. Go ahead and try it out. Load a schematic into Eeschema, and click on the BoM button. Click on the 'Generate' button. The plugin will produce the BOM file, and provide information about the work it did in the 'Plugin information' text box.

Browse to the project directory, and you will find the newly-created '.csv' file, that contains the BoM (Figure 41.5). You can open this file with a text editor or a spreadsheet application. In Figure 41.1, you can see the BoM for project 3 of this book as it appears in Microsoft Excel.

Figure 41.2: The BoM button.

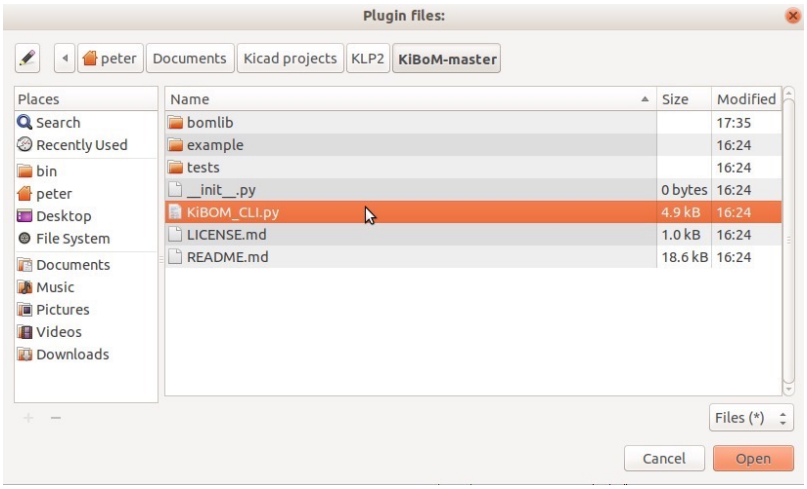


Figure 41.3: Navigate to the location of the KiBOM_CLI.py file.

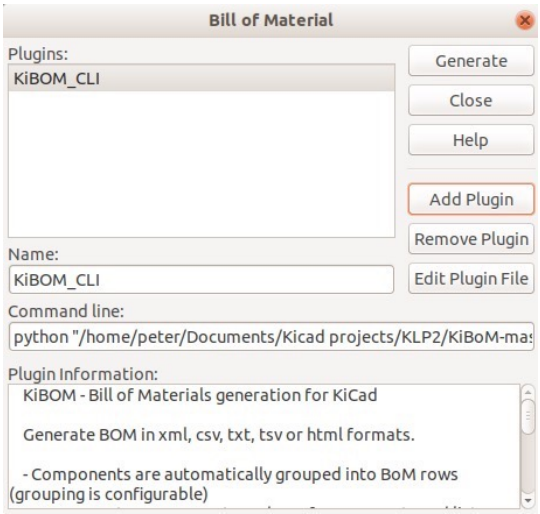


Figure 41.4: The BoM plugin is installed.

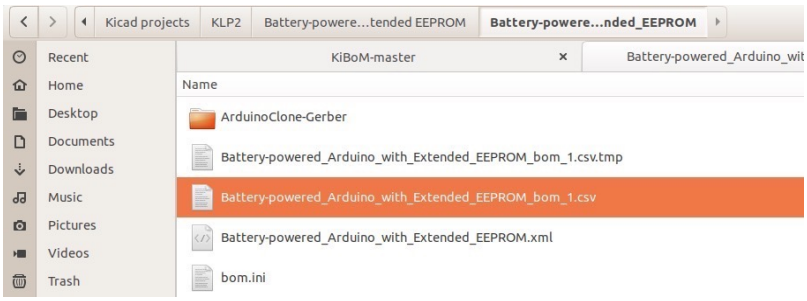


Figure 41.5: The newly-created CSV file that contains the BoM.

If you are interested in an alternative plugin to implement the bill of materials feature in KiCad 5, I encourage you to have a look at KiCost (https://pypi.org/project/kicost/). As per the KiCost documentation, this plug-in generates part-cost spreadsheets for boards. This is very helpful if you want to make an accurate cost calculation for your project. The plugin can also generate an XML file that you can upload to Digi-Key or Mouser to quickly order the project parts.

42. How to design a custom page layout

You can create a custom version of the page layout. A popular reason for doing that is to include a logo in every project page.

In this recipe, you will learn how to create a simple custom sheet layout. The only difference between the default layout and the one we are about to create is that the custom one contains a logo graphic. Once you have your custom sheet layout file, you will be able to use it in all your schematics.

To create and edit sheet layouts, KiCad provides a helper application called Pl_Editor. To start Pl_Editor, go back to the main KiCad application (which is really an application launcher), and click on the Pl_Editor button on the far right of the toolbar (Figure 42.1).

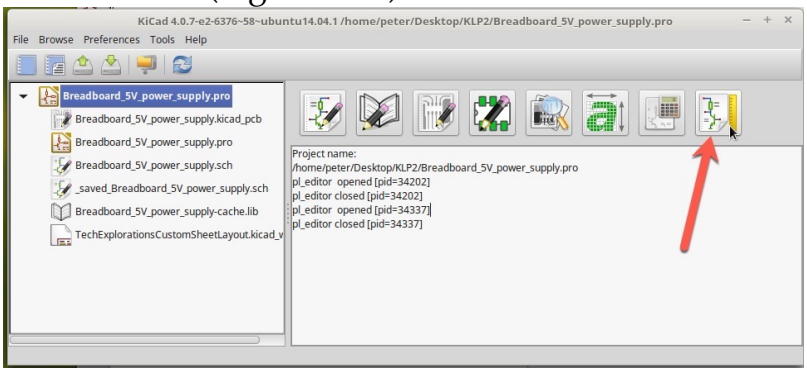


Figure 42.1: Starting PI_Editor.

The editor will start, showing the default layout. You can choose to work and edit the default layout, or create a new one from scratch (right-click on File and then New Page Layout Design).

Let's keep this task as simple as possible, and go with the first option. Your Pl_Editor should look like the example in Figure 42.2.

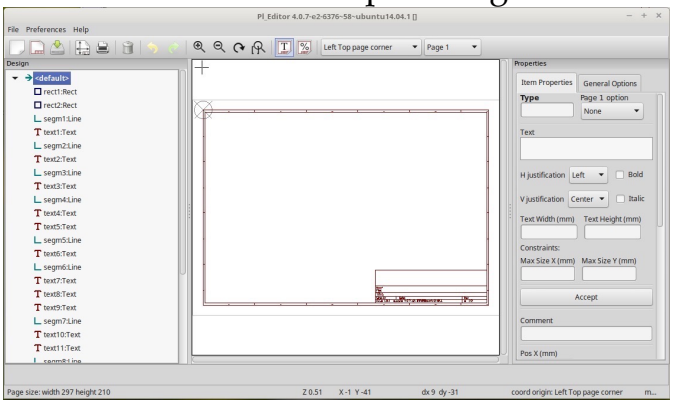


Figure 42.2: Pl_Editor showing the default sheet layout.

On the right side of the editor, you can see the design elements. They consist of lines, rectangles and text fields. To add new items in the designer, place your mouse cursor at the location where you would like to add it and do a right-click. The contextual menu that appears gives you the option to add lines, rectangles, text or bitmap (graphics) object.

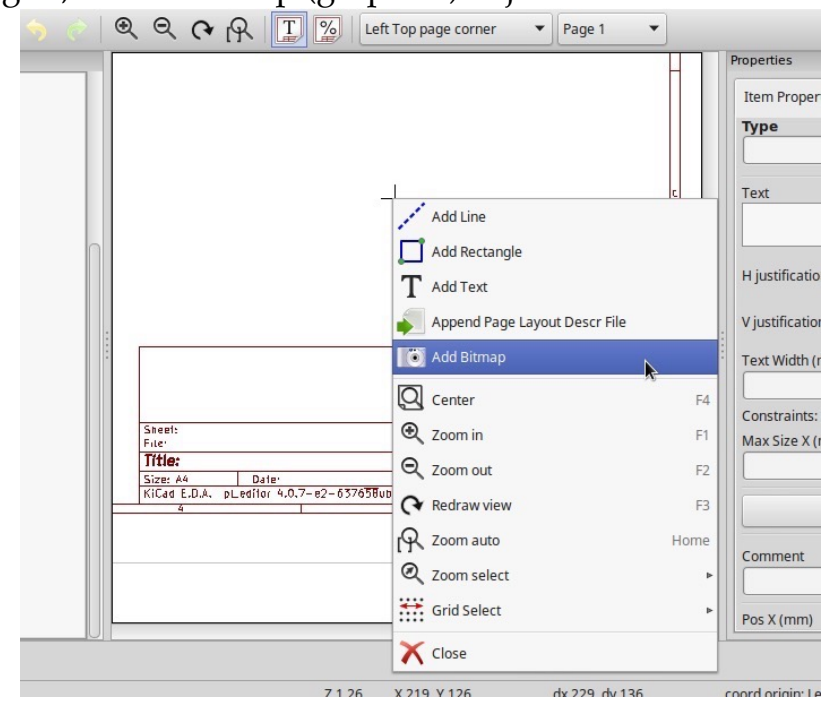


Figure 42.3: The Pl_Editor context menu.

Experiment with all of those. Lines and rectangles are simple to use, just click and draw. With text, you have two options. You can simply drop in a text object that contains the fixed text that you type in the text object's 'Text' field in its properties (as in Figure 42.4), or use one of KiCad's format symbols.

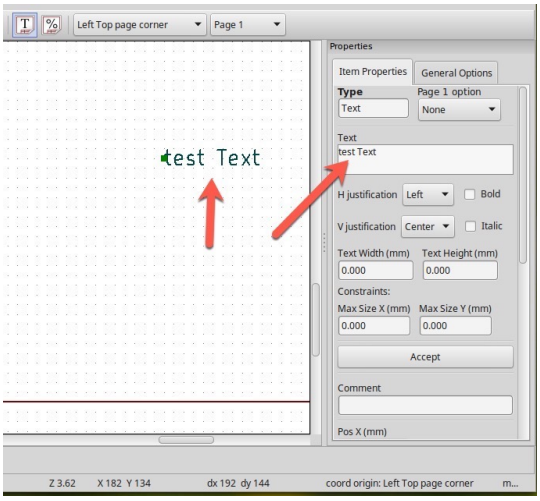


Figure 42.4: You can insert fixed text in the sheet layout.

The default sheet layout uses format symbols to display information such as the KiCad version and the date in the title block. For example, if you

click on the Title text object, you will see the content 'Title: %T' in the Text field of the object's properties (Figure 42.4).

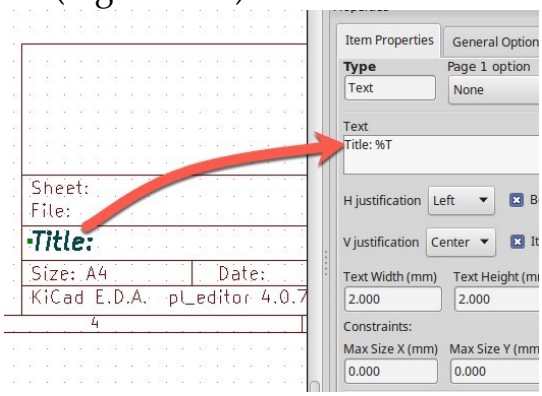


Figure 42.4: The Title format symbol '%T'.

The content of the '%T' symbol is sourced from the Title field of the Page Setting dialog box (Figure 42.4). The table below contains all the available formal symbols that you can make reference to in Pl_Editor:

Symbol	Description
%K	KiCad version
%Z	Paper format name
%Y	Company name
%D	Date
%R	Revision
%S	Sheet number
%N	Number of sheets
%Cx	Comment (x = 0 to 9 to identify the comment)
%F	Filename
%P	Sheet path
%T	Title

Table 42.1: A list of format symbols available in KiCad.

Since we are working on modifying the default sheet layout, we will not need to do anything with the format symbols. Instead, I'd like to add the Tech Explorations logo in the title block.

The process starts with creating a graphics file in PNG format that contains the graphic you would like to add to the layout. Take care to create one with the appropriate dimensions because Pl_Editor has a very limited ability to resize graphics files. An image of around 500 pixels width and 300 pixels height fits nicely within the header of the title block.

Next, right-click somewhere in the title block header, and select 'Add Bitmap' (see Figure 42.3). This will bring up the image chooser. Navigate to the location of your graphics file and double-click to select it (Figure 42.5).

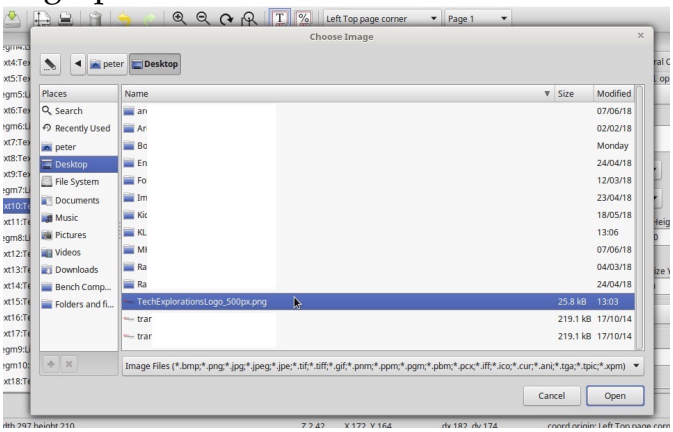


Figure 42.5: The image chooser.

A small window titled 'New Item' will appear. This window allows you to specify the position of the new item, but since the new item will be attached to the mouse pointer once you click the 'Ok', and therefore you will be able to position it accurately, there is no need to make any changes in the New Item window. Click 'OK' to insert the logo graphic to the layout.

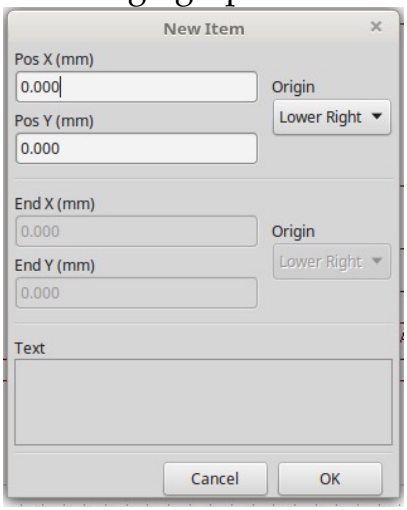


Figure 42.6: Confirm the addition of the new item to the layout.

When you click 'OK', the New Item window will disappear, and the logo graphic will appear attached to your mouse pointer. Move the mouse to find the exact location where you'd like the logo to be, and left-click to commit it.

If the location or the size of the logo is not exactly as you would like them, there's a couple of things you can do.

First, you can move the logo by right-clicking on it and selecting 'Move'. Move the logo to a new location and left-click to commit it.

Second, if you need to reduce the size of the graphic logo by a small amount, you can do this within Pl_Editor by increasing the Bitmap PPI setting. The default is 300. If you increase this value to, say, 500, the graphic will become smaller. Don't forget to click on the Accept button after you change the Bitmap PPI value. For larger changes or increases in size, it is better to use a proper graphics editing tool to prevent reducing the quality of the graphic.

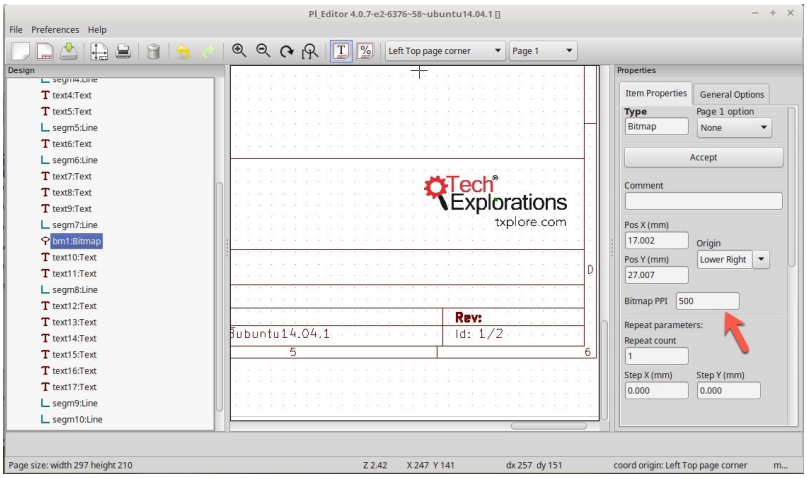


Figure 42.7: Changing the Bitmap PPI will change the size of a graphic.

In Figure 42.7 you can see what my logo looks like after I changed the value of the Bitmap PPI field to 500.

If you are ready to save your new layout and apply it to your schematic sheet, go ahead. Click on File and Save, and give it a reasonable name in the Save dialog box that appears. I also like to save my layouts in a location that is easy for me to find. In this case, I save the layout file in the project directory (see Figure 42.8).

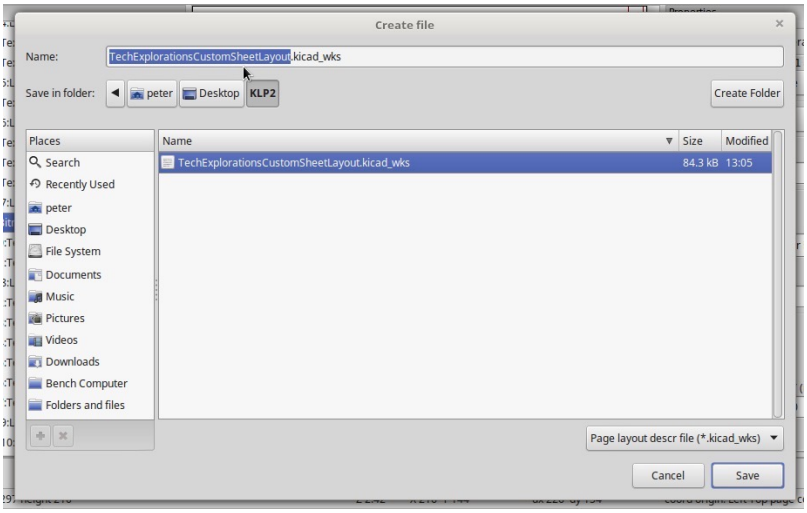


Figure 42.8: Save your layout file in a location that is easy to find later.

That's all there is to do in Pl_Editor. Close this application and return to Eeschema.

In Eeschema, open the Page Setting dialog box (File, Page Settings). In the 'Page layout description file' field, click on the browse button to find the layout file you just created. Click Open to accept your selection.

You will get a warning about the effect that changing the layout will have on your project. Accept the warning to dismiss the box, and you will see the file name of the custom layout file in the 'Page layout description file' field (Figure 42.9).

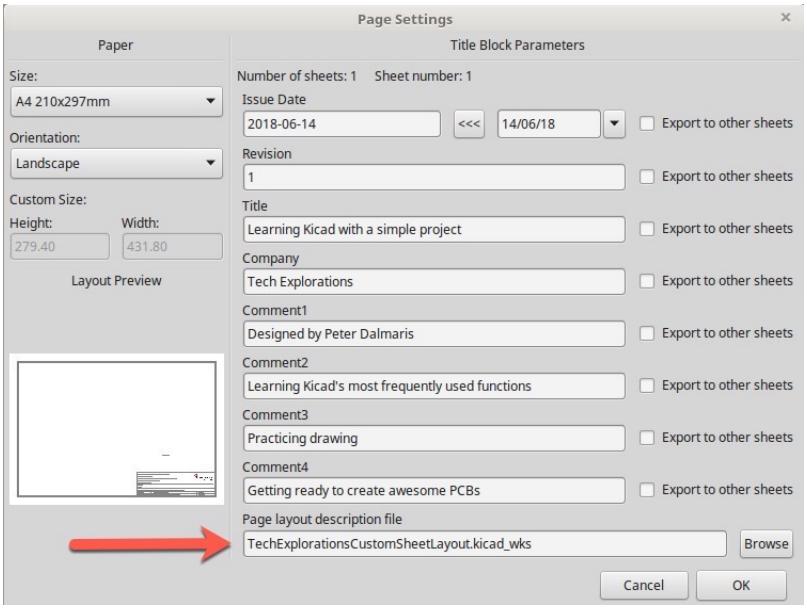


Figure 42.9: We have selected our custom sheet layout file.

When you click the OK button to close the Page Settings dialog box, KiCad will ask you to save the project file. It does that because you have made a change to the project, and this change must be committed to the project file. Click on the Save button to do so and overwrite the existing project file with the updated version.

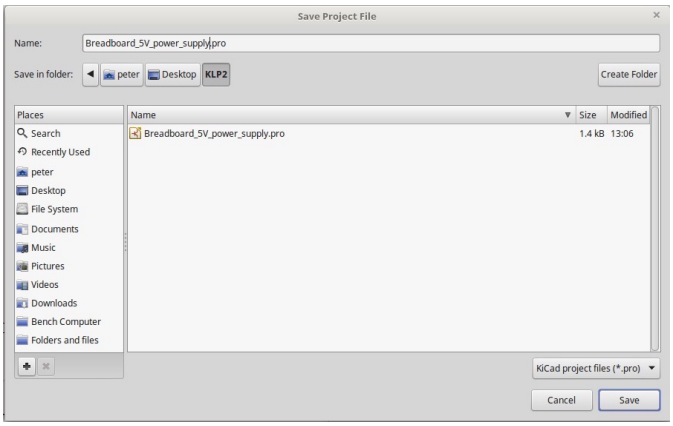


Figure 42.10: Save the updated project file to disk.

The process is now complete. Your logo should appear in the title block, with all the text blocks populated as expected. Figure 42.11 shows what my custom sheet layout looks like.

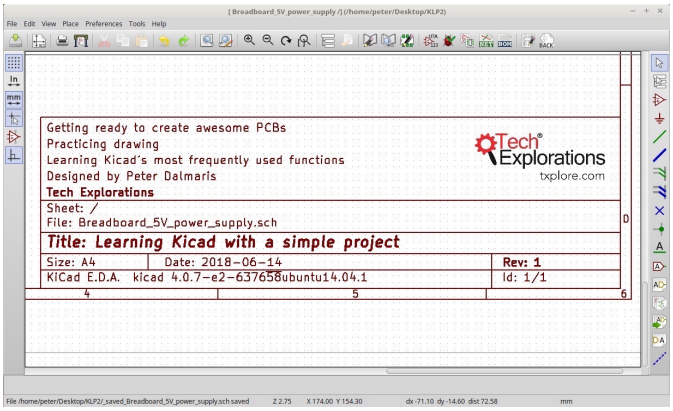


Figure 42.11: Peter's custom sheet layout.

You can use the same sheet layout in any of your projects simply by loading the sheet file to the project through the 'Page layout description file' field in Page Settings.

43. How to use hierarchical sheets

As your schematic diagrams grow, your sheets will become cramped and harder to read. When that happens, you need to split your diagrams into multiple sheets. In KiCad, this is done using hierarchical sheets. Hierarchical sheets are regular schematic sheets with hierarchical relations between them. When you start Eeschema, the sheet that you see is known as the 'root'. You can create a new sheet as a 'child' of the root, in which you can continue your work.

Sheets communicate with each other by means of hierarchical and global labels, and hierarchical pins. These allow nets that exist in different sheets to become logically connected. This way, when you import the netlist to Pcbnew, the layout will show footprints that represent symbols in all sheets that are part of the same project, with their nets connecting their pins seamlessly.

In this recipe, you will learn how to use hierarchical sheets. Open Eeschema. To add a child sheet to the root sheet, click on the 'Create hierarchical sheet' button on the right toolbar (). With your mouse, click at an empty location of the root sheet to create one point of a new rectangle, drag and click again to commit the rectangle, as in Figure 43.1.

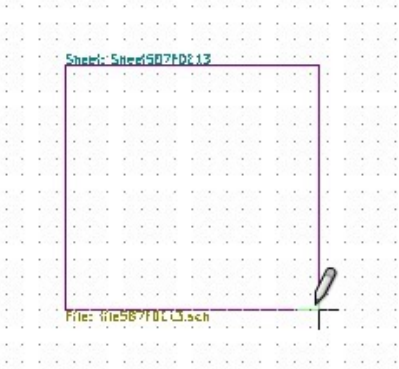


Figure 43.1: This rectangle represents a hierarchical sheet.

In the properties window that appears, type in a reasonable name, like 'Connectors'.

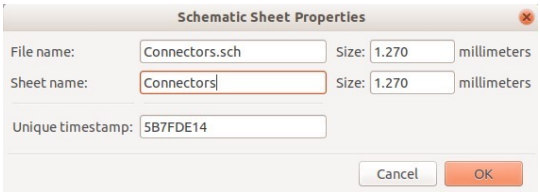


Figure 43.2: The hierarchical sheet properties needs a good name.

Your new sheet is now created. The Sheet rectangle will display the sheet name and the new file that is created to contain its data. You can navigate through sheets using the Navigator. Its button is in the top toolbar (Figure 43.3).

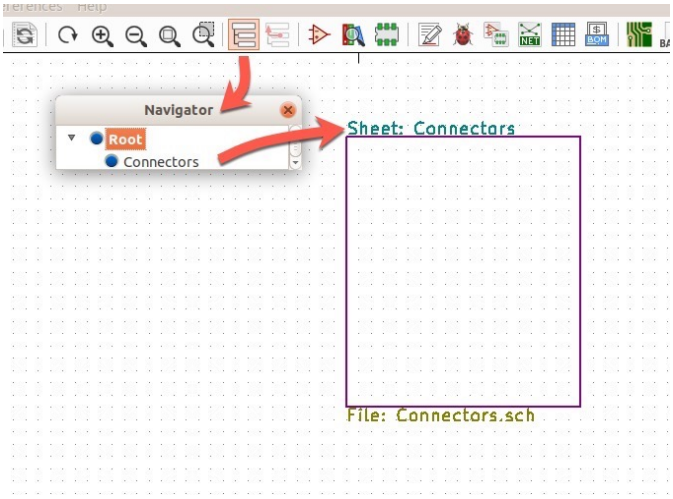


Figure 43.3:Use the sheet navigator to access a sheet.

In the navigator, double-click on the Connectors sheet to access the sheet by that name. You will see a new, empty sheet. To go back to root, click again on the Navigator button and then double-click on the Root sheet, or click on the 'Leave sheet' button to jump to the parent of the current sheet ().

In the Connectors sheet, add a simple symbol, like the Conn_01x04_Male connector (Figure 43.4).

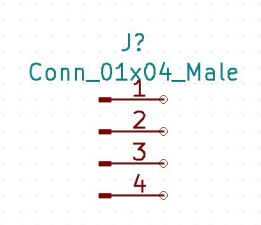


Figure 43.4: A new connector in the child sheet.

What you will do next is to use hierarchical labels to connect the four pins of this connector to nets that exist in the root sheet. To do that, you must use a specialised type of net label, the 'hierarchical label'. You will then place its counterpart, the 'hierarchical pin' to the corresponding sheet rectangle in the parent sheet.

Start with the first pin (although you can do multiple pins in bulk once you understand the process). In the Connectors sheet, click on the hierarchical

label button from the right toolbar ( ). Then click on the first pin of the connector to create a new hierarchical label. Call it 'GND'. The label properties allow you to set its orientation and shape. The connector now looks like the example in Figure 43.5.

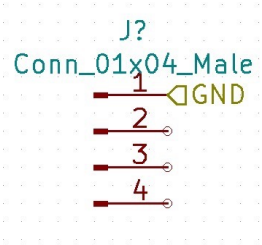


Figure 43.5: A hierarchical label in pin 1.

Next, you must expose this label to the parent sheet by placing a hierarchical pin to the sheet's representative rectangle. Jump to the parent sheet ( ). The easiest way to create a hierarchical pin is to use the hierarchical pin importer tool from the right toolbar ( ). You can also use the hierarchical pin tool ( ) but then you will have to remember the exact name of each hierarchical label in the child sheet. Let's go with the importer.

Click on the importer button, then place the mouse cursor on the right or left edge of the hierarchical sheet rectangle. Click to create the hierarchical pin. A new pin will appear, with the name of the hierarchical label you created in the child ('Connector') sheet (Figure 43.6).

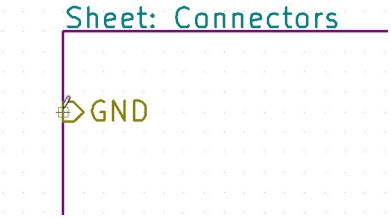


Figure 43.6: A hierarchical pin connecting the two sheets.

If you had used the hierarchal pin tool, a dialog box would ask for the name of the pin. The name you type must match exactly the name of a hierarchical label in the child sheet.

If you had created multiple hierarchical labels in the child sheet, then with the importer tool still selected, you could continue clicking, and with each click, another pin carrying the name of its label counterpart will appear.

Try that now. Return to the Connector sheet (use the Navigator or double-click to the sheet representative rectangle), and using the hierarchical

label tool create three more labels, as you can see in Figure 43.7. In pins 3 and 4 I have configured the labels as 'bi-directional'.

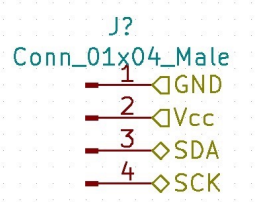


Figure 43.7: Four hierarchical labels.

Return to the root sheet, and select the pin importer tool. Click three times below the existing hierarchical pin, to create the remaining four pins. The result is in Figure 43.8.

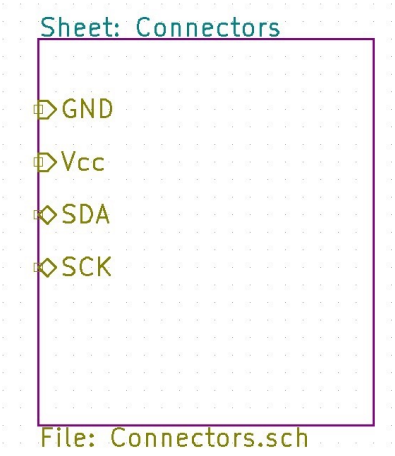


Figure 43.8: Imported hierarchical pins.

From here, you can connect these pins to other parts of your schematic in the root sheet. For example, you can use wires, or local labels, as in Figure 43.9.

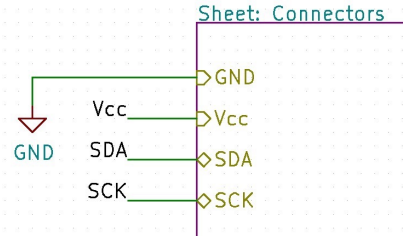


Figure 43.9:Connecting hierarchical pins to the rest of the schematic.

44. How to use differential pairs

Differential pairs allow you to route two wires at the same time. This capability is useful in those cases where two adjacent wires are complementary and have to be treated as such. By routing the wires together, we can ensure that the travel time of the signal from origin to destination will be practically the same, as well as that they will receive an almost identical amount of interference.

Differential pairs are important in applications such as audio, digital singling, data communications (RS-422, Ethernet etc). The technique helps to minimise electromagnetic interference and crosstalk. It also helps to minimise the noise that these wires emit to their environment.

KiCad, apart from allowing to draw differential pairs in the layout, also provides tools to configure them. You can set the total length of the pair, ensuring matching lengths of traces where that is important (like in high-speed memory applications).

To enable the differential pair feature in Pcbnew, you need to label the participating nets in Eeschema with names that end with '+' and '-' or with '_N' and '_P'. Let's have a look at a very simple example, depicted in Figure 44.1.

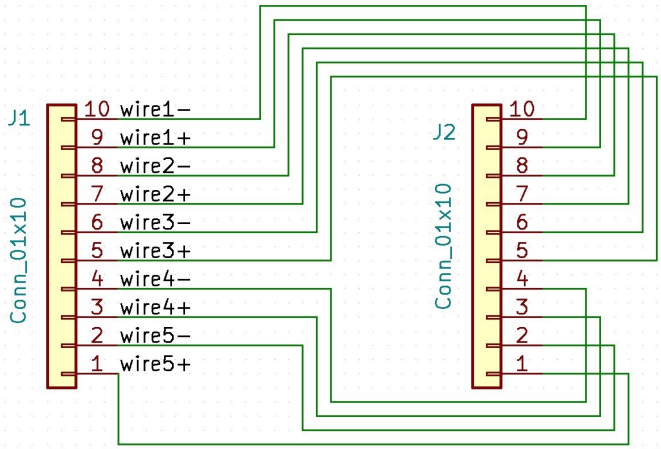


Figure 44.1: The wires that belong to a pair have net names that end with '-' or '+'.

In this example, I have two connectors, with wires connecting their pins. I have labeled the wires with net labels, and for each pair, the name of the label ends with either '+' or '-'. In place of '-' or '+' I could have used '_P' or '_N'.

When you export the netlist for this schematic and import it into Pcbnew, you will have a layout like the one in Figure 44.2.

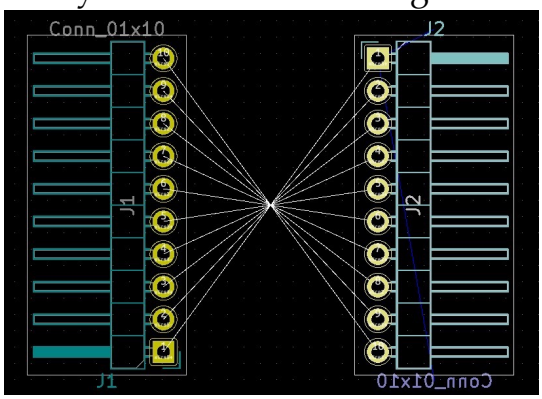


Figure 44.2: The unconnected layout view.

You can route each net using individual routes, or use differential pair routing. To enable differential pair routing, choose this option from the Route menu (Figure 44.3).

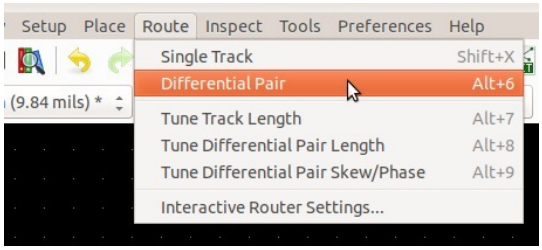


Figure 44.3: Enable differential pair routing.

Then, click on one of the pads to begin drawing. You will see that you are now drawing two traces: one is connected to the pad you clicked on, and the other to its pair (Figure 44.4).

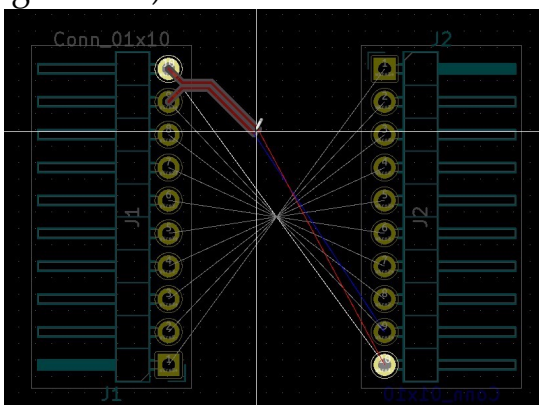


Figure 44.4: Started drawing a differential pair.

Click on the pad that is highlighted in the J2 connector to complete the drawing. The result is in Figure 44.5.

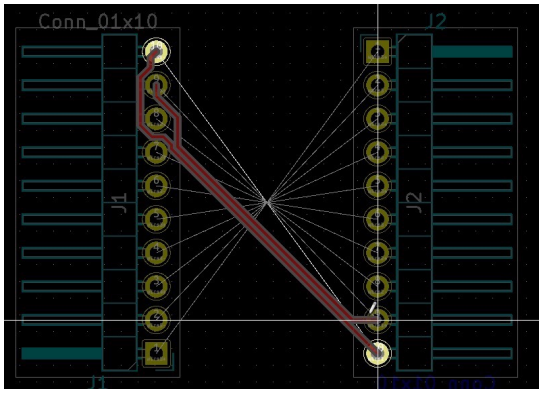


Figure 44.5: The first differential pair is routed.

In the next example, try to route a differential pair but this time add vias, like in Figure 44.6. As you can see, by typing 'V' you create two vias, one for each trace member of the pair.

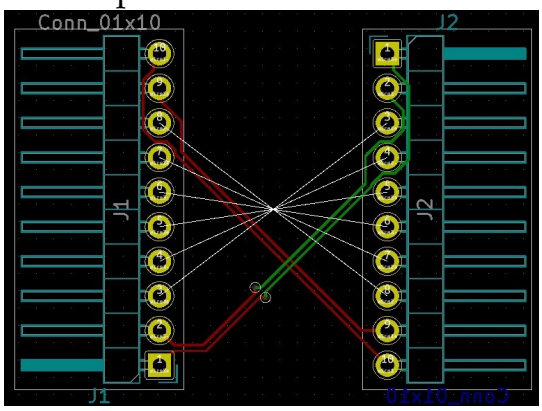


Figure 44.6: The second differential pair is traced in the B.Cu with the help of vias.

KiCad allows you to fine-tune the length of a trace. This allows you to control the signal propagation through the trace, precisely. Let's try this out. Start by clicking on the 'Tune Track Length' option from the Route menu. Then, click on a member trace of one of the differential pairs that you want to tune the length for. This will give you information about the current length of that trace (Figure 44.7). In this example, the trace I clicked on is shorter than the target.

Figure 44.7: The current length of this trace is 37.056 mm; the target is 40 mm.

To configure the target length, right click on the trace and click on the 'Length Tuning Settings…' option. In the window that comes up, define the track length target. There are other options you can experiment with, that define the characteristics of the meanders that KiCad uses to adjust the trace length. I have changed my target length to 40 mm (Figure 44.8).

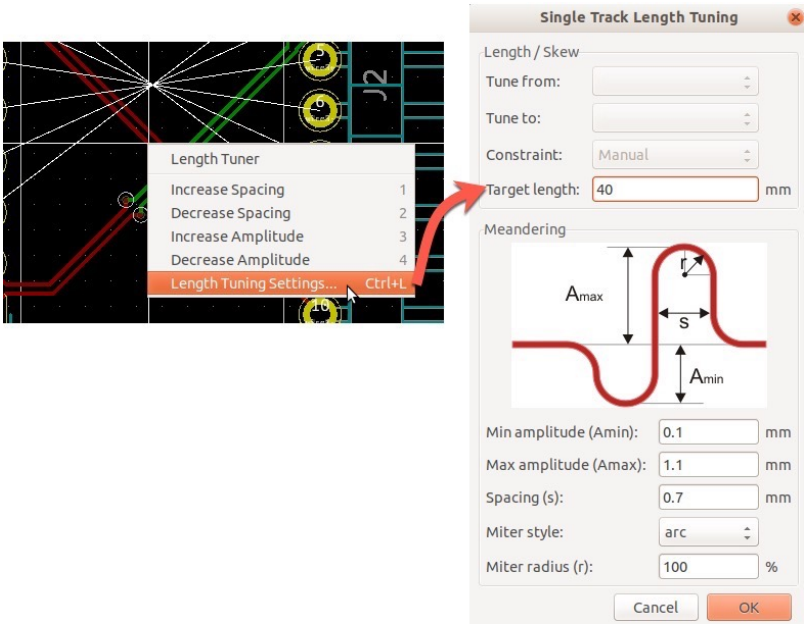


Figure 44.8: You can control the characteristics of differential per traces.

To adjust a trace so that its length becomes equal to the target, click on a trace to select it and move your mouse to the direction where you want to meanders to appear. KiCad will keep you informed about the current length of the trace. When you reach the target, click to commit the changes.

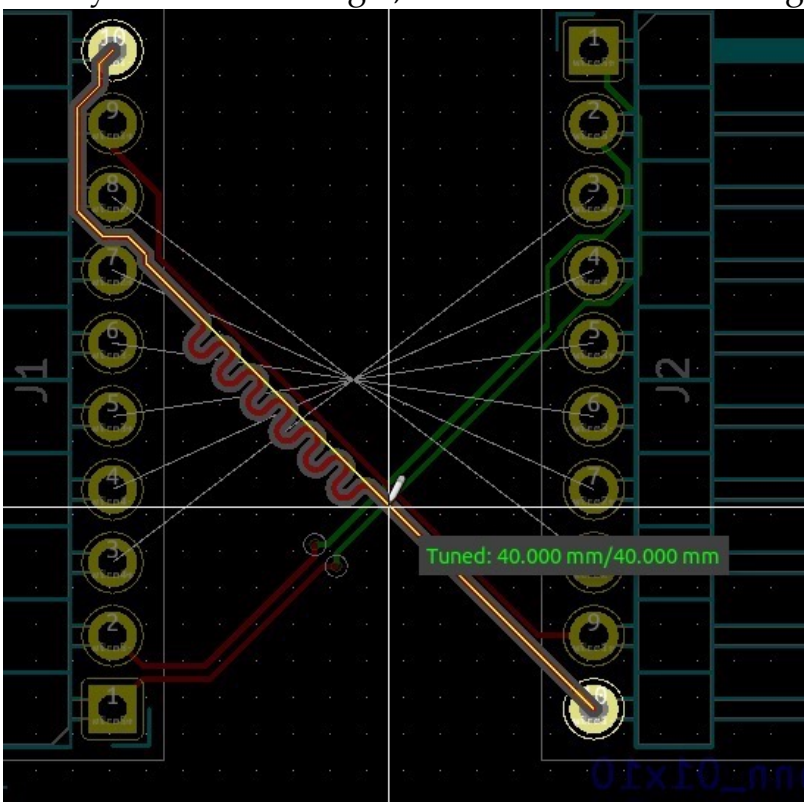


Figure 44.9: Adjusting the length of a differential trace.

Repeat the same process for the rest of the differential traces. If the length adjustment is blocked by a via, continue on the other side of the via.

You can tune the length of both traces in a pair at the same time by selecting the 'Tune Differential Pair Length' option from the Route menu.

45. Interactive router

The interactive router is an advanced tool that can make a big difference in the effort required to layout a board. With the interactive router, you can create and modify traces in three modes, and you can configure your mouse drag behaviour. Let's experiment with the interactive router now.

Load one of your layouts in Pcbnew. From the Route menu, open the 'Interactive Router Settings…' window (Figure 45.1).

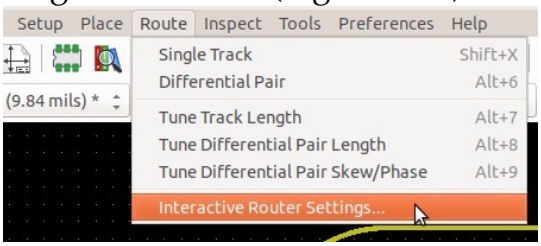


Figure 45.1: Open the Interactive Router Settings.

My interactive router is set up as you can see in Figure 45.2.

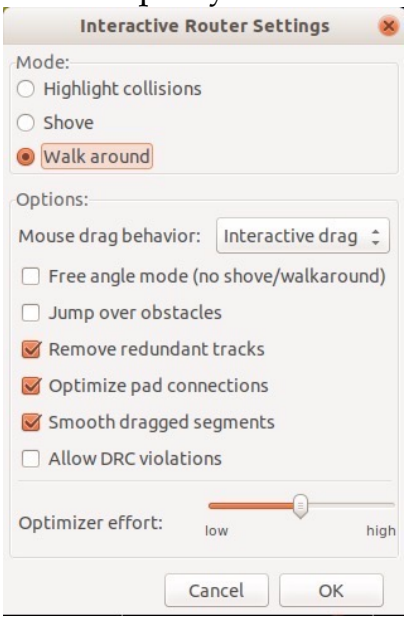


Figure 45.2: Current setting of my interactive router.

The 'Walk around' mode will ensure that no elements of my layout (such as vias and traces) can be moved while I am drawing a new trace. My mouse behaviour is set to 'drag', which means that I can select a trace or a via, and move it around without disconnecting it for traces that are already connected to it.

Let's try an experiment. In Figure 45.3, I have deleted the trace from pad 8 of the pin header.

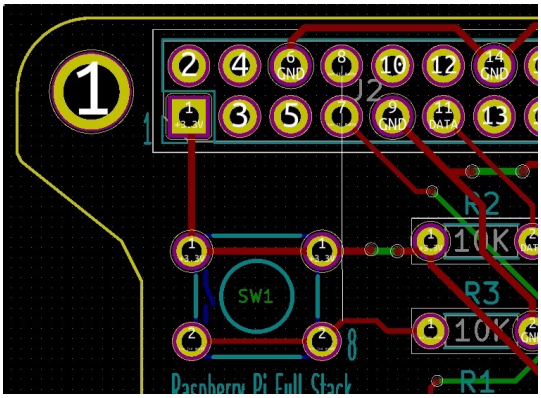


Figure 45.3: One pad is not connected.

With the interactive router in 'Walk around' mode, I will start drawing a new wire. As I move my mouse towards pad 2 of the SW1 footprint, the trace is routed around other objects, likes footprints and wires (Figure 45.4).

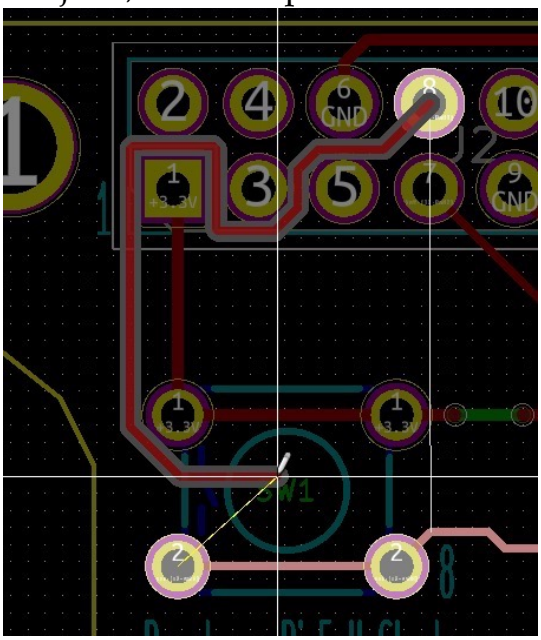


Figure 45.4: Example of the 'Walk around' mode.

You can influence the route of the trace using your mouse. Because in the interactive router setting I have not selected the option 'Allow DRC violations', the trace will always be legitimate.

Next, switch the router mode to 'Shove'. Try the same experiment, but now try to 'force' your way through an existing trace. The interactive router will obey, and move traces around as you are directing it with your mouse (Figure 45.5). In this case, this kind of behaviour is not very useful.

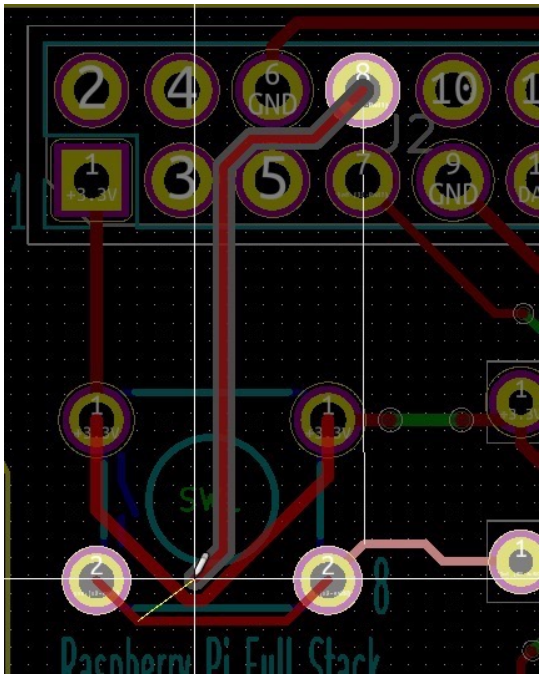


Figure 45.5: Example of the 'Walk Shove' mode.

Next, try the 'Highlight collisions' mode. The router will not move or go around anything. Instead, it will simply highlight all the violations that you are causing with your manual routing (Figure 45.6).

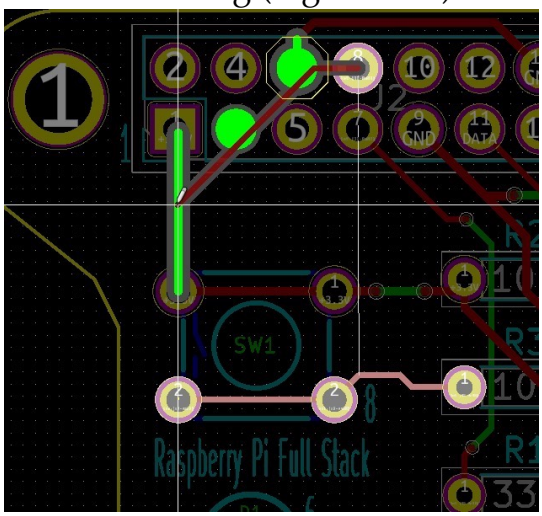


Figure 45.6: Example of the 'Highlight collisions' mode.

Let's experiment with the mouse. The default setting for mouse drag (when you click and hold on an item like a via or route) is 'Interactive drag'. Click on a via and move the mouse and try this out. As in the example of Figure 45.7, the traces remain connected to the via, and the router moves other items aside as it is trying to accommodate your instructions.

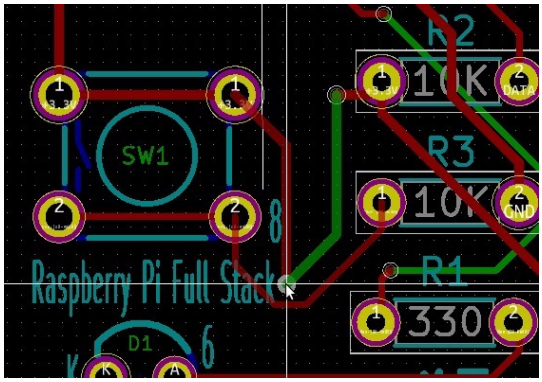


Figure 45.7: Interactive drag when using the mouse.

Now, switch the mouse mode to 'Move item' and repeat the same process. As you can see in Figure 45.8, while I was able to move the via, the traces did not follow along.

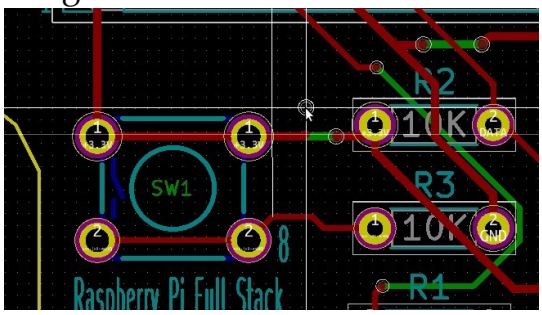


Figure 45.8: 'Move item' mode when using the mouse.

It is worth your time to learn the interactive router. Its usefulness really becomes obvious in complex projects.

46. Creating unique board edge cuts

If you need a PCB with a complex outline, you will find it very challenging to design with the available tools in Pcbnew. For such cases, it is possible to create the board outline in a dedicated drawing program, and then import the design into Pcbnew.

There are significant limitations in using external drawing programs that relate to the kinds of graphics file formats supported for KiCad, and the complexity of the design that you want to import. Nevertheless, you can create very interesting boards using only open source software and some basic artistic skills.

The current version of KiCad can import DXF R12 graphics files. You can create such files using a variety of tools, like Adobe Illustrator and the open-source Inkscape. Because of the limitations of KiCad's import function, any DXF file you import must not contain polygon lines and ellipsis. If it does, the import will fail, without KiCad providing any feedback. If your drawing does have such elements, you can use another tool to convert them into primitive components. Such a tool is LibreCAD, also open source.

In this recipe, I will show you how to create a complex board outline using Inkscape and LibreCAD.

Inkscape is available on virtually any computer platform. Go to inkscape.org to download your copy. Similarly, you can download your copy of LibreCAD from librecad.org.

Start by creating a design of your PCB in Inkscape. Take care so that your design has the dimensions you need. In my case, I would like to draw a fancy-looking PCB that is at maximum 60 millimetres in length and 50 millimetres in width. To make my design easier, I set the grid to be the same as the one I plan to use in Pcbnew. I have also configured the Inkscape page in a similar way to the drawing settings in Pcbnew. To access the properties window, click on the cogwheel icon in the top bar (Figure 46.1).

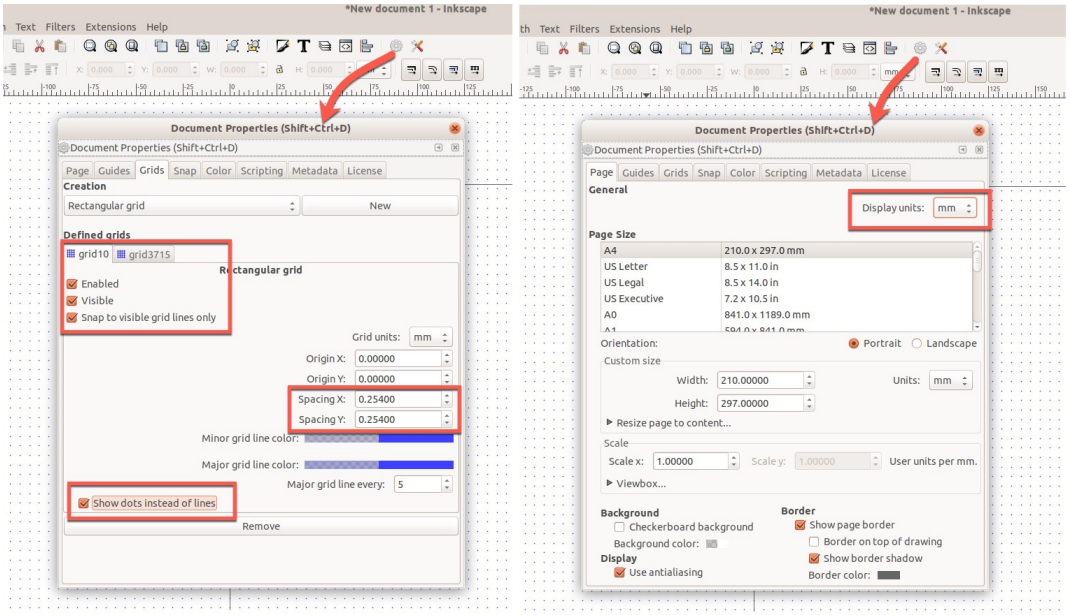


Figure 46.1: In Inkscape, set up the drawing area to resemble the Pcbnew sheet.

Go ahead and create a drawing. You can see my fancy drawing in Figure 46.2. I know. My drawing skills are terrible.

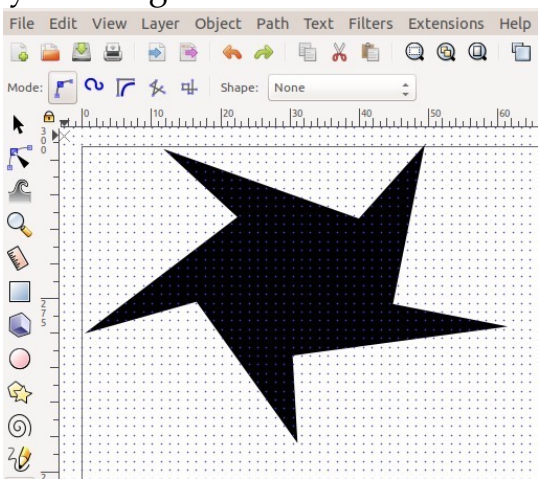


Figure 46.2: This drawing will become a PCB.

When you are finished, click on File, Save As. In the window that appears, choose the 'Desktop Cutting Plotter (AutoCAD DXF R14)(*.dxf)' file format (Figure 46.3).

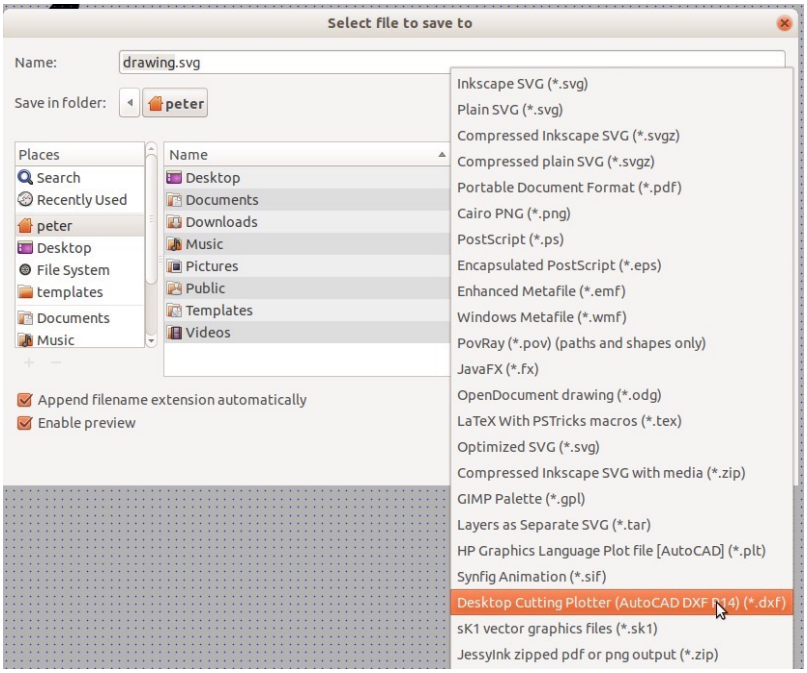


Figure 46.3: Save in DXF R14 format.

In Figure 46.4 you can see my export settings.

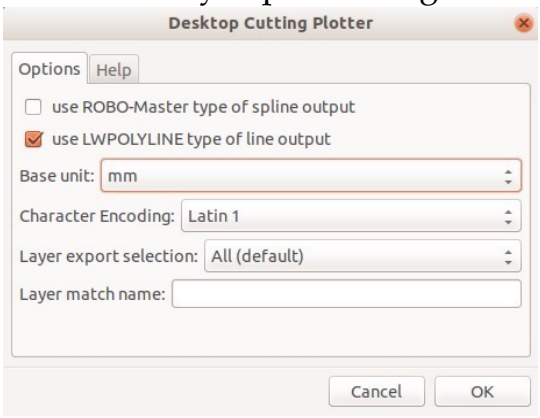


Figure 46.4: The DXF R14 export settings.

Notice that this format, DXF R14, is not compatible with KiCad's import feature. If you try to import it to KiCad now, you will not get what you expect. You will need to convert the DXF file to a compatible format using LibreCAD.

In LibreCAD, open the File menu and click on Open. Find the file you just saved with Inkscape and open it.

Click on the outline of the shape, and from the Tools menu select Modify, Explode (Figure 46.5).

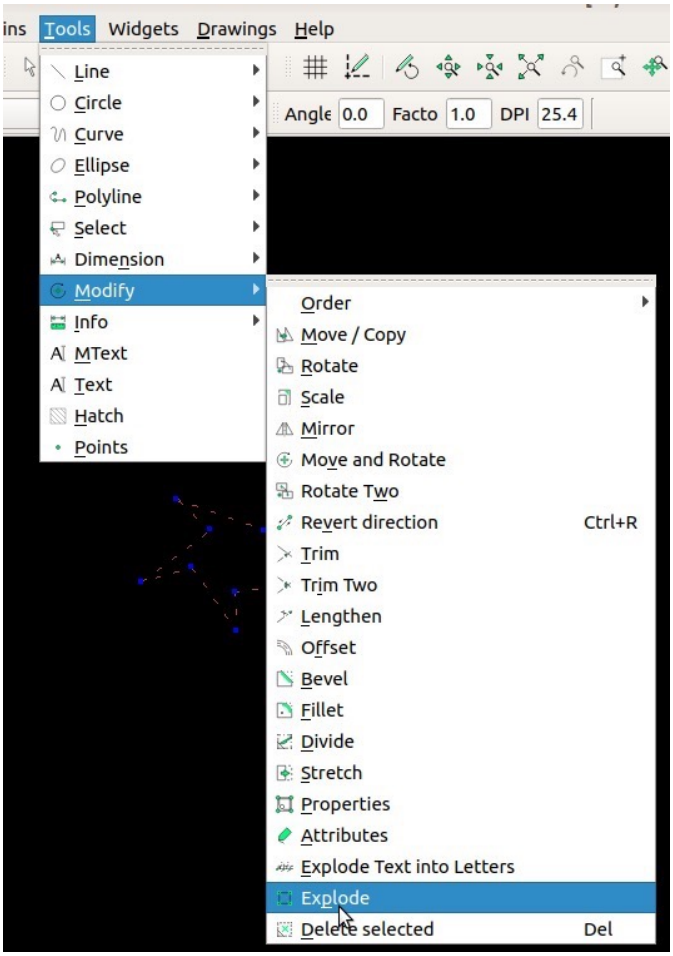


Figure 46.5: The Explode tool will convert all drawing elements into primitives that KiCad can import.

Next, save the file in DXF 12 format. From the File menu in LibreCAD, choose Save As. In the window that appears, name your new file and choose the DXF R12 file format from the dropdown (Figure 46.6).

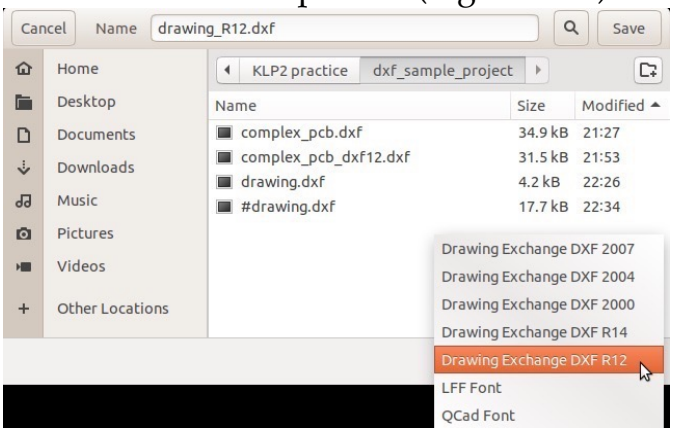


Figure 46.6: Save the exploded drawing in the DXF R12 file format.

Back in Pcbnew, choose Import, 'DXF File…' from the File menu (Figure 46.7).

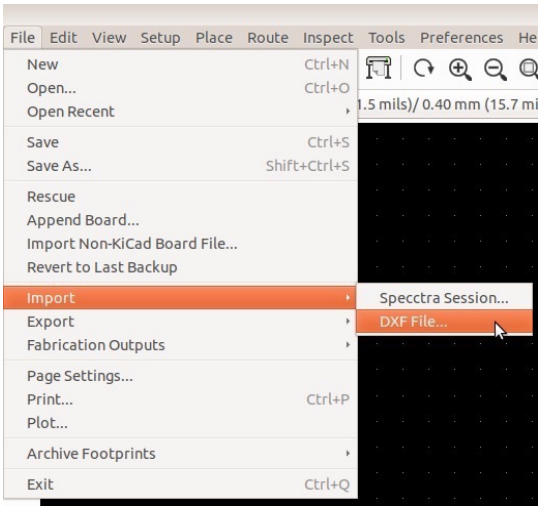


Figure 46.7: Import the DXF R12 file into Pcbnew.

The Pcbnew importer will ask you for the import settings. Select the Edge.Cuts graphics layer. I prefer to use 1 mm as the default line width for the outline. Your new board outline will appear in Pcbnew. Try viewing its 3D representation in the 3D Viewer (Figure 46.8).

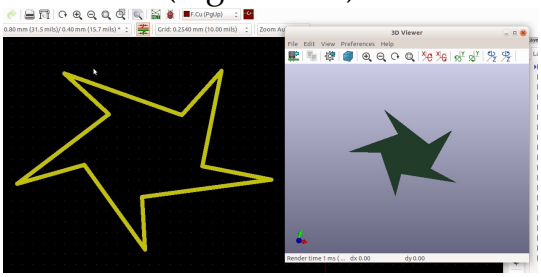


Figure 46.8: Your new, fancy board outline in Pcbnew.

From this point, you can continue working on the board layout as normal. Place footprints, trace pads, create copper zones, all within your fancy-looking PCB outline.

47. Using Git for version control

KiCad projects consist of files that contain plain text. You can use any text editor to open them and have a look inside. For example, in Figure 47.1 you can see a section of the schematic '.sch' file for one of my projects, opened using the Atom text editor.

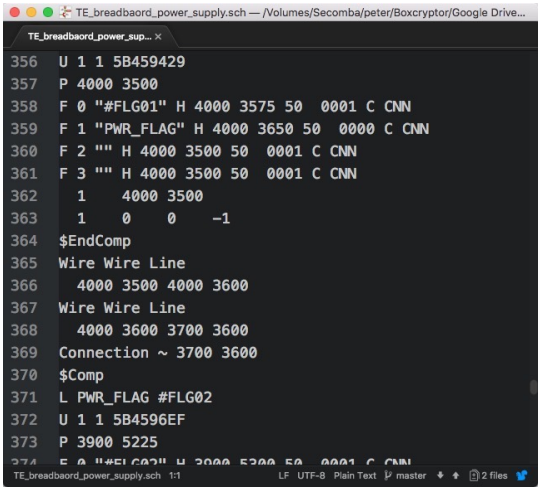


Figure 47.1: Import the DXF R12 file into Pcbnew.

The fact that KiCad's projects are text files makes it easy to use a versioning system, like Git, to keep track of all changes made across the project. There are several popular versioning systems available, but my personal preference is Git. It is open source, fast, very popular, and extremely versatile. As you will see in this recipe, it is also very easy to use.

What you will learn in this recipe is how to use Git, and the Github online repository, to maintain your project's history. Doing so will allow you to:

1. Preserve your project's history. This will allow you to access past versions of any file in your project.
2. Create experimental branches. This is useful if you want to experiment with alternate design options, or design different versions of the same board. Each one can be stored in a separate branch of the same repository.
3. Merge or discard different branches. This Git function allows you to merge (unify) two branches into one. For example, you may have the main branch of your project, and go on to work on an experimental branch as you are investigating a special board feature. If the experiment succeeds, you can merge the experimental branch to the main branch, and continue from there.

If not, you can just discard the failed experiment branch, and continue with the intact project in the main branch.

These are just three of the many possible scenarios. Those are the three scenarios that I use most often.

If you use Git alongside an online repository, like Github, you will be able to use this versioning system to collaborate with other people on the same project. You will also be able to share your project and its history with other people. In Figure 47.2, you can see part of the history of one of the projects in this book, as it appears in the publicly-accessible repository on Github.

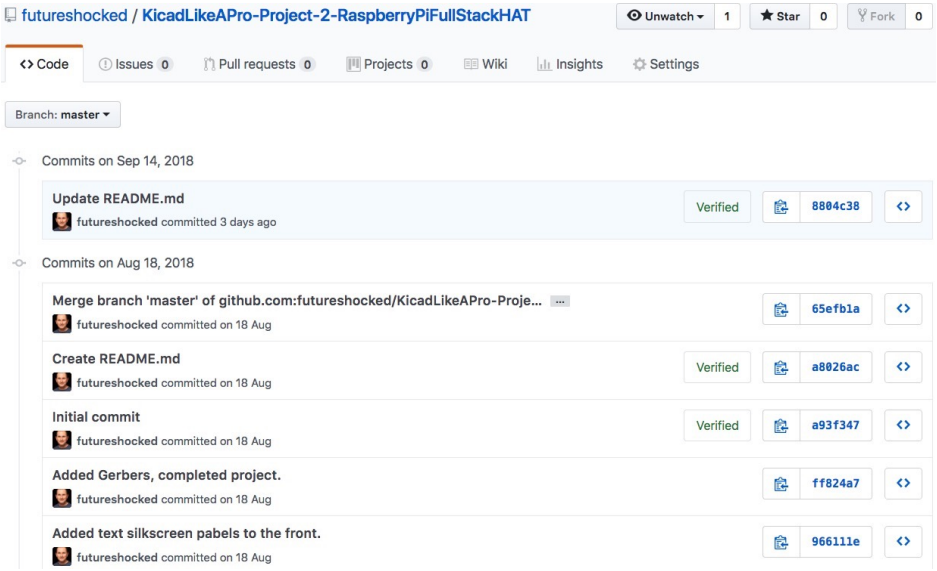


Figure 47.2: Part of a history of one of the projects in this book on Github.

In Figure 47.3, you can see the history of the schematic file for the same project.

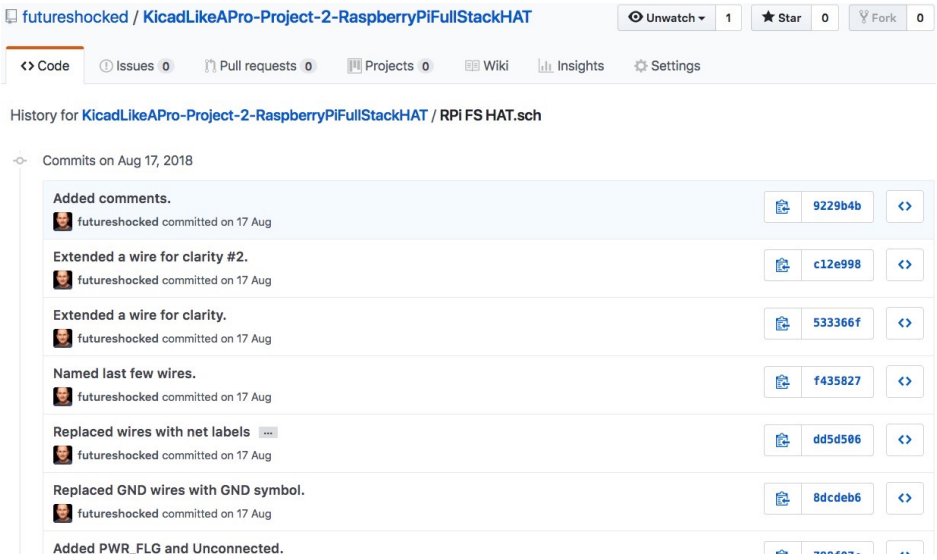


Figure 47.3: Part of the history of the schematic file on Github.

The numbers in the right side of each column are called 'commits'. Each number is an ID that refers to a commit. The full ID of a commit is a long alphanumeric, like 'f0266d1343a446d068e874f46cf9cc29'. What you see in Figure 47.3 is a short hash of that ID. A commit may contain changes in multiple files, or additions and removals of files. To get detailed information about a commit, you can click on the commit number. The result is a side-by-side comparison of the changes detected in each file, as in the example of Figure 47.4.

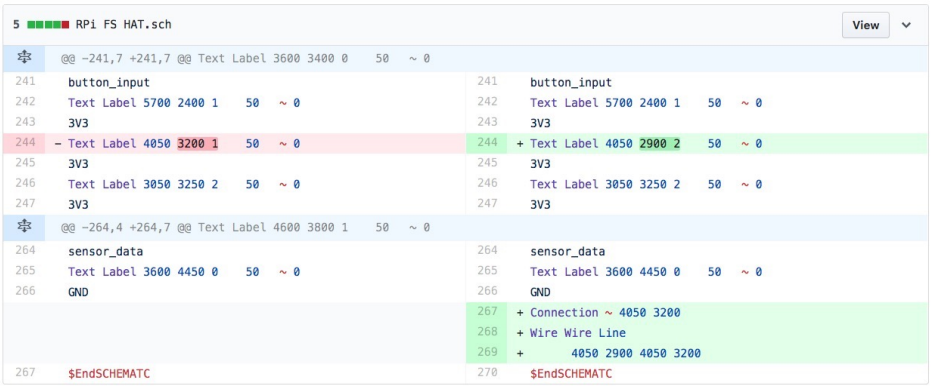


Figure 47.4: The changes detected in file 'RPi FS HAT.sch' at commit.

In this tutorial I will give you a basic introduction of Git and Github. Git is a big topic, and I do encourage you to learn more about it by using a specialised source, like this Getting Started guide from Github.

Read on to learn how to use Git and Github in the context of a KiCad project. You will learn how to:

1. Create the repository on your computer.
2. Commit changes of your project to the repository.
3. See those changes in the log.
4. Checkout past commits.
5. Create branches.
6. Upload your project to Github so you can share it with other people.

Start by installing Git on your computer. For each operating system, you can find detailed instructions on the Git web site. Here, I will show you how to do this in Ubuntu. To begin, start the terminal and type the following instructions at the command prompt:

```
$ sudo apt install git
```

Once the installation is complete, configure your git user settings:

```
$ git config --global user.name 'testuser'
$ git config --global user.email 'testuser@example.com'
```

From this point onwards, the instructions and the code is the same for all operating systems as long as you have access to the command line. Most of

these operations are also possible using the Git desktop app, if you prefer a graphical user interface.

You are now ready to use Git with your KiCad project. Use the terminal to browse into your KiCad project. At the root of the project, initialise the Git repository:

```
$ git init
```

You will get confirmation that your new repository is ready to use, something like this:

```
   Initialized empty Git repository in /home/peter/Documents/KiCad
projects/KLP2/BreadboardPowerSupply_v2/.git/
```

Add the current project files into the repository:

```
$ git add .
$ git commit -am 'First commit.'
```

Again, Git will respond with confirmation. If you want to know what the current status of the repository is, use the 'status' command:

```
$ git status
On branch master
nothing to commit, working tree clean
```

Now lets say that you have made a change to the schematic diagram and that you wish to store this in the Git repository. Save your Eeschema file, and check the status of the repository:

```
$ git status
On branch master
Changes not staged for commit:
  (use 'git add <file>...' to update what will be committed)
  (use 'git checkout -- <file>...' to discard changes in
working directory)

   modified:   BreadboardPowerSupply_v2.bak
   modified:   BreadboardPowerSupply_v2.sch

no changes added to commit (use 'git add' and/or 'git commit
-a')
```

Git knows about the change, but it is not yet committed to the repository. To commit it, use the commit command:

```
$ git commit -am 'Added text label.'
[master 2f77b7b] Added text label.
 2 files changed, 183 insertions(+), 181 deletions(-)
 rewrite BreadboardPowerSupply_v2.bak (68%)
```

Continue to do some more work in Eeschema, and save and commit this new work:

```
$ git commit -am 'Added 2nd text label.'
[master 74cabdc] Added 2nd text label.
 2 files changed, 4 insertions(+)
```

Git is keeping track of these commits in its log. If you want to know what is in the log, you can check with the log command:

```
$ git log
commit 74cabdc120196a703eb71c2290f016fa3b4b65eb (HEAD ->
master)
Author: Peter Dalmaris <peter@txplore.com>
Date:   Sun Aug 12 19:52:07 2018 -0700

    Added 2nd text label.

commit 2f77b7b68037436ed169797e9dbb8bd7678bd69a
Author: Peter Dalmaris <peter@txplore.com>
Date:   Sun Aug 12 19:50:23 2018 -0700

    Added text label.

commit 06d8909c726ae8f57ca456c4f9e0fc46e7b37fdd
Author: Peter Dalmaris <peter@txplore.com>
Date:   Sun Aug 12 19:44:37 2018 -0700

    First commit.
```

The newer commits appear on top. Each commit has an ID, that is useful if you want to operate on it. Let's say, for example, that you changed your mind about the latest change you made, and that you want to go back to a previous version of the Eeschema file. As long as you have the commit ID, you can do that using the checkout command:

```
$ git checkout 2f77b7b68037436ed169797e9dbb8bd7678bd69a
Note: checking out '2f77b7b68037436ed169797e9dbb8bd7678bd69a'.

You are in 'detached HEAD' state. You can look around, make
experimental
changes and commit them, and you can discard any commits you
make in this
state without impacting any branches by performing another
checkout.

If you want to create a new branch to retain commits you
create, you may
do so (now or later) by using -b with the checkout command
again. Example:

  git checkout -b <new-branch-name>

HEAD is now at 2f77b7b Added text label.
```

As an attribute to the checkout command you must provide the commit ID of the commit that you want to retrieve. Git will confirm that it retrieved this commit, and that in the case of this example, the Eeschema files have been restored to the earlier version. Close and reopen Eeschema and you will see that the schematic is, indeed, at its earlier version. The changes you committed since then still exist in the repository, its just that you are now at an earlier time in its history.

As Git is telling you, you are now in detached HEAD state. This simply means that you have rewound your project to an earlier time. The commit on which you are working on now is not the latest. You can work on this version and decide what to do next. If you are just looking around, then you can checkout HEAD when you are ready and continue work from the latest version:

```
$ git checkout master
```

If you choose to make changes, you can create a new branch and continue work there. You can create a new branch of your project like this:

```
$ git checkout -b new_branch
```

Where 'new_branch' is the name of the new branch. Continue to work in this branch as normal, saving your files and committing them to the repository. When you decide to return to the master branch, simply check it out:

```
$ git checkout master
```

Close and reopen your KiCad apps and you will see that their contents are updated accordingly. Continue to work in the new branch. The convention is that the master branch contains completed work. When you are ready, you will want to merge your experimental branch with the master branch. To do this, checkout the master branch, and then merge the experimental branch into the master.

```
$ git checkout master
$ git merge new_branch
```

Sometimes, there are conflicts between the two versions of a file, and Git will not be able to automatically merge them. In that case, Git will let you know, like this:

```
Auto-merging BreadboardPowerSupply_v2.sch
CONFLICT (content): Merge conflict in
BreadboardPowerSupply_v2.sch
Auto-merging BreadboardPowerSupply_v2.bak
CONFLICT (content): Merge conflict in
BreadboardPowerSupply_v2.bak
Automatic merge failed; fix conflicts and then commit the
result.
```

You will need to resolve the conflict by using a text editor and manually merging the files in question.

Git is a powerful tool for both saving your work at multiple points, and for experimenting. If used correctly, there is essentially no risk of loosing work. Knowing that you can return to any point in the history of your project is truly liberating.

Another important benefit of using Git in your KiCad projects is that it is made for collaboration. You can share your repository with other people

using a tool like Github.com. If you work in a team, then the team members will be able to work on the same project at the same time, in their own branch, and then merge their work into the main branch. This makes it possible to use KiCad in teams, even though the KiCad itself does not have any collaboration capabilities.

Uploading your repository to Github

Suppose you would like to publish your project online so that other people can access it. You can do this easily by creating a remote Git repository on a service like Github. The repository on Github is a remote copy of your local repository. Every time you make a change to your local repository, you can push it to the remote so that your collaborators can access the updates. And vice-versa: if a change is accepted in the remote, you can pull it to your local repository so that you can use the changes.

Start by creating a repository on Github (Figure 47.5).

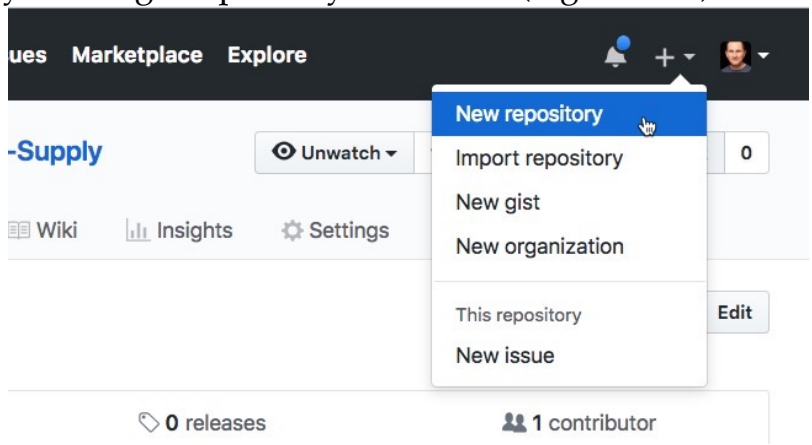


Figure 47.5: Create a new repository on Github.

Give it a name, a license type (I prefer the MIT license for sharing projects), and a description.

Next, go to your command line, and browse to the directory where your project is saved. I assume that you already have a local Git repository for your project in this directory. If not, follow the instructions in the first part of this chapter to create and populate one.

Go ahead to create the remote for your local project. A common name for the remote is 'origin', so let's use that in this example.

Using the command line, working inside the directory of your project, type:

```
$ git remote add origin git@github.com:futureshocked/
KiCadLikeAPro-Project-Breadboard-Power-Supply.git
```

Replace the remote URL for the one of your project. You can find your remote URL from the repository page on Github (Figure 47.6).

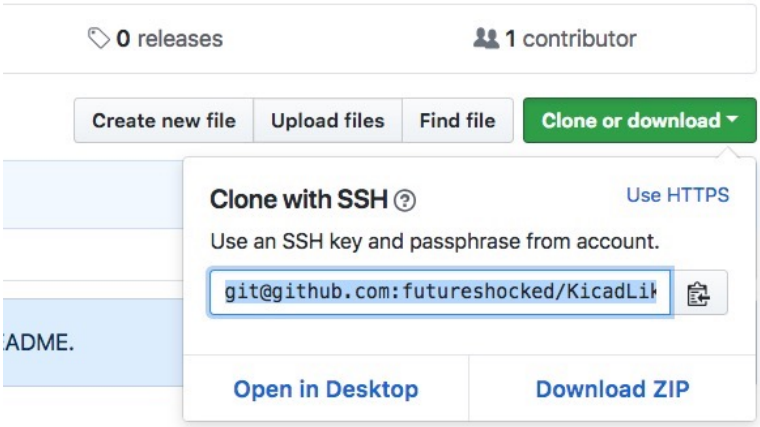


Figure 47.6: Find out your repository URL on Github.

You can confirm that the remote is set by typing:

```
$ git remote -v
origin  git@github.com:futureshocked/KiCadLikeAPro-Project-
Breadboard-Power-Supply.git (fetch)
origin  git@github.com:futureshocked/KiCadLikeAPro-Project-
Breadboard-Power-Supply.git (push)
```

Before you can 'push' (i.e. upload) your local repository to the remote for the first time, you will need to 'pull' (i.e. download). Pushing and pulling is not the same as uploading and downloading because both also include merging. With merging, the contents of the local and remote repositories are synchronised.

Go ahead to pull the remote to your local repository like this:

```
$ git pull origin master --allow-unrelated-histories
From github.com:futureshocked/KiCadLikeAPro-Project-
Breadboard-Power-Supply
 * branch            master     -> FETCH_HEAD
Merge made by the 'recursive' strategy.
 LICENSE | 21 ++++++++++++++++++++
 1 file changed, 21 insertions(+)
 create mode 100644 LICENSE
```

Git will ask you for a comment using your default text editor. Type in something like 'First pull' and save the file to allow Git to complete the pull. For the first pull only, you will need to use the —allow-unrelated-histories since the two repositories are not related. From this moment onwards, you will be able to use 'git pull origin master' to pull from the remote.

Finally, go ahead to push your local repository to the remote. Do this:

```
$ git push origin master
Counting objects: 117, done.
Delta compression using up to 2 threads.
Compressing objects: 100% (116/116), done.
Writing objects: 100% (117/117), 332.61 KiB | 2.75 MiB/s,
done.
```

```
Total 117 (delta 61), reused 0 (delta 0)
remote: Resolving deltas: 100% (61/61), done.
To github.com:futureshocked/KiCadLikeAPro-Project-Breadboard-
Power-Supply.git
   9428bca..30dfd14  master -> master
```

The local repository is now merged with the remote on Github. Refresh your web browser to see your project files on Github (Figure 47.7).

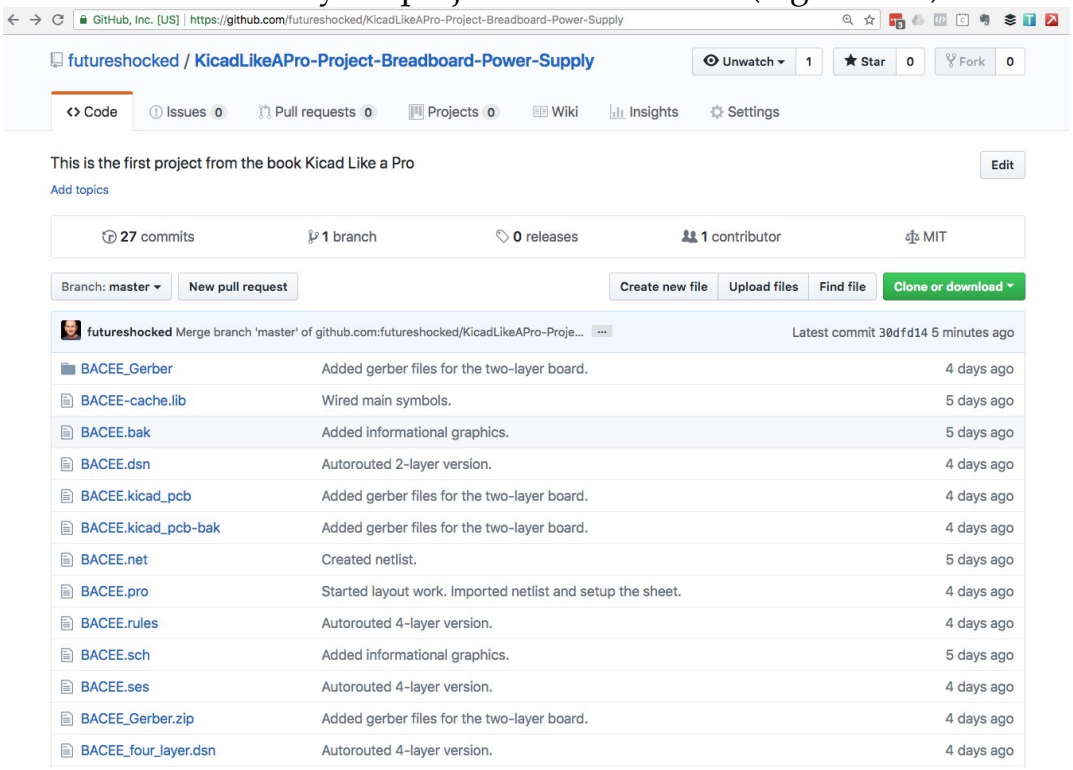


Figure 47.7: After the push, the project files are now on Github

Authentication

To avoid having to type in your Github credentials every time you pull or push, you should setup your SSH keys in your Github account. Each of your computers has a unique SSH key (or you can generate one). Once setup, Github will use that key to authenticate it, in place of a user name and password. My Github SSH keys setup looks like the example in Figure 47.8.

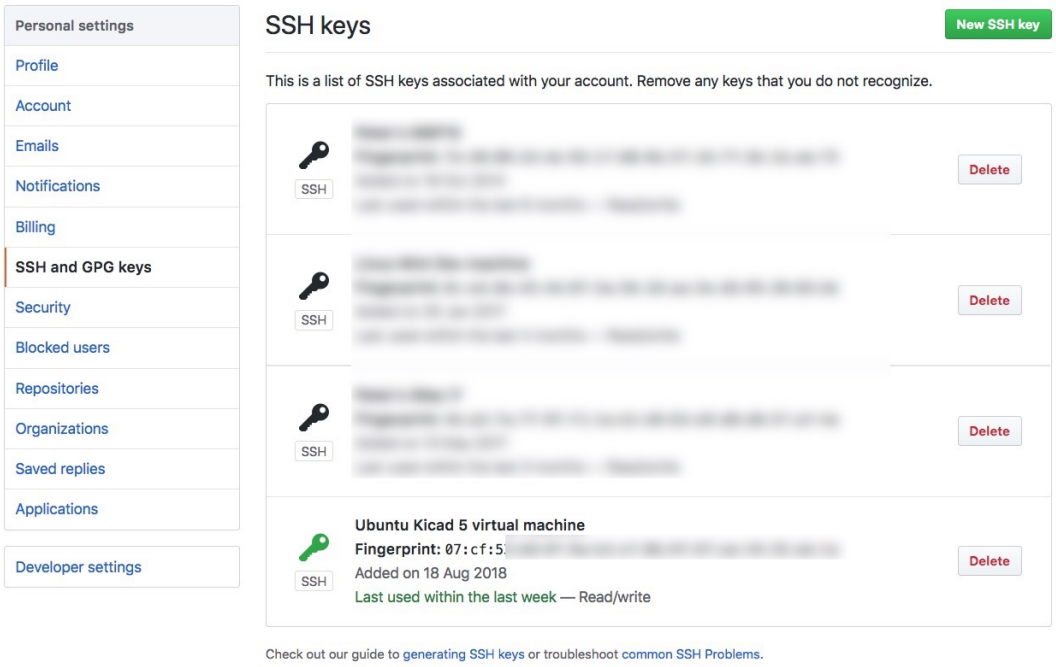


Figure 47.8: SSH keys speed up interactions with Github.

To learn how to setup your SSH keys, please follow the instructions that Github provides in its documentation, at https://help.github.com/articles/connecting-to-github-with-ssh/

48. Creating a multi-layer PCB

In KiCad, you can create PCBs with up to 32 layers. You can set the number and configuration of layers in Pcbnew, using the Layer Setup window (under the Setup menu). The layers number and configuration is independent of your schematic.

Before getting into the 'how' for multilayer boards, let's ask 'why' would you want to design one.

An obvious answer is that multilayer PCBs make it possible to reduce the size of a board by providing more space for the routing of the traces. If you have a small board and find yourself unable to fully route it (and the autorouter also being unable to complete the routing task), even after repositioning the components, then increasing the number of available layers is a good option.

On the other hand, multilayer PCBs are expensive. A 4-layer PCB can cost around twice the cost of a 2-layer PCB. It may be better to use a jumper wire for that last un-routable trace.

Let's look at the 'how', now. Start by setting the number of layers you would like to have on your PCB by bringing up the Layer Setup window, in Pcbnew, Setup (Figure 48.1).

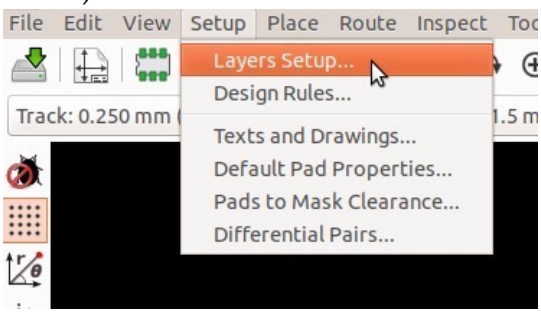


Figure 48.1: Setup the number of layers and their configuration.

Next, select the number of layers for your board, under the Copper Layers drop-down (Figure 48.2).

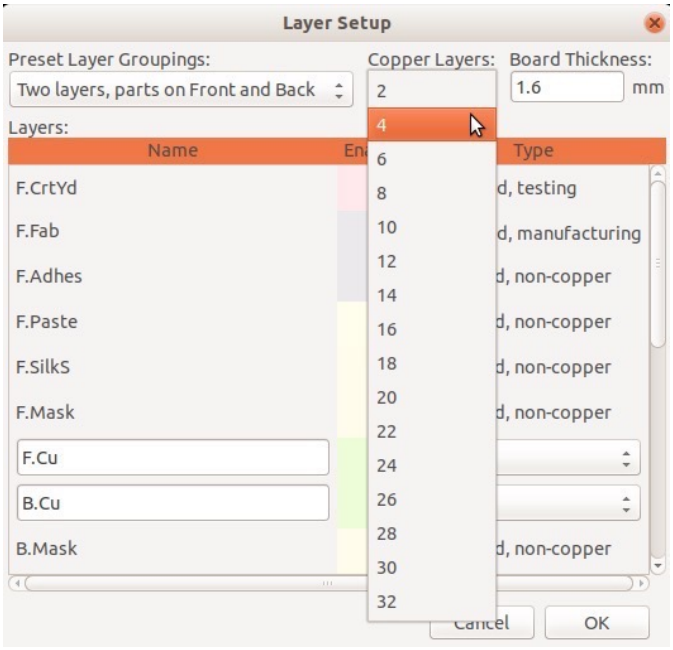


Figure 48.2: Let's make this a four-layer board.

Finally, configure the usage type of each layer by selecting an option from the layer's drop-down menu (Figure 48.3). These settings are used by the autorouter. If you are routing your board manually, these settings have no effect in your work.

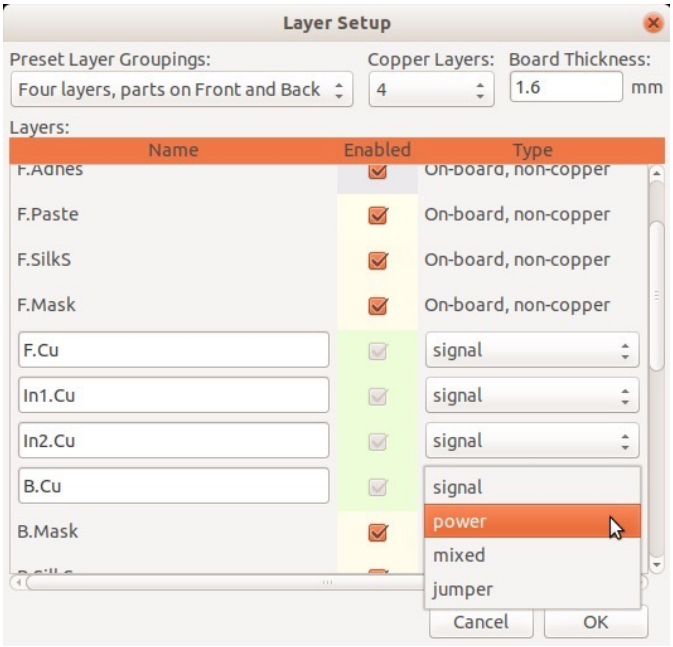


Figure 48.3: Select the usage type of each layer.

Except in rare cases, you will assign signal, power, or mixed usage types to each layer. Here are some guidelines that you can use when you decide how to configure these layers:

1. When you use SMD components in the front copper layer, set the F.Cu to signal. This ensures that the traces that connect the components stay

on this layer, thus avoiding vias (except for this that connect the components to power and ground).

2. If you place components on the back copper layer, you should also set B.Cu to signal.
3. If there are no components on the back copper layer, you can set B.Cu to signal or mixed. This will allow for the autorouter to use B.Cu to route as needed.
4. Choose at least one of the inner layers for Power (usually GND). You can use blind vias to connect SMD components to that inner layer. A blind via does not go through the entire width of the board; it only connects an outer layer to an inner layer, directly.
5. You can choose to have both inner layers set for power. In a four-layer board, you can assign In2.Cu to Power for GND, and In1.Cu to Power for 3.3V. In high-frequency applications, this arrangement also has electromagnetic interference benefits.

A common configuration for four-layer boards is this:

1. F.Cu: Signal
2. In1.Cu: Power
3. In2.Cu: Power
4. B.Cu: Signal

A common configuration for two-layer boards is this:

1. F.Cu: Mixed
2. B.Cu: Mixed

If you are designing a 4-layer board or over, you should also enable blind and micro vias. To do that, open the Design Rules Editor and check the boxes for 'Allow blind/buried vias' and 'Allow micro vias (uVias)' (Figure 48.4).

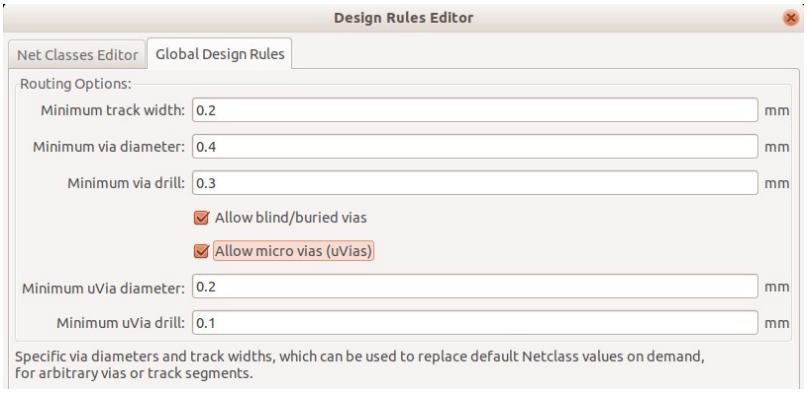


Figure 48.4: Enable blind and micro vias.

Let's have a look on how to manually route a four-layer board. From the Layers Manager, select the layer where you would like to start drawing a

new wire, say F.Cu. Start drawing a trace. To create a via and continue drawing in a different layer, right-click to reveal the context menu, and select one of the appropriate via menu items (Figure 48.5).

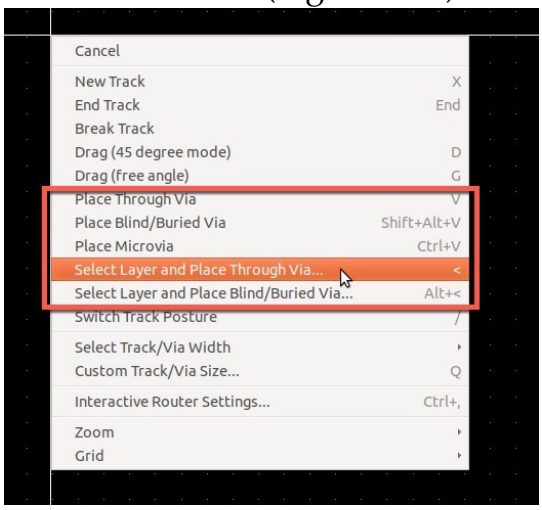


Figure 48.5: Select a via option.

Each of those options has a keyboard hotkey. For example, you can use 'V' to create a through via, and 'Shift-Alt-V' for a blind or buried Via. If you use one of the Place commands, KiCad will create vias to a layer based on your previous selections. If you want select a specific layer to connect, opt for one of the 'Select Layer' options. This will bring up the Select Layer chooser (Figure 48.6).

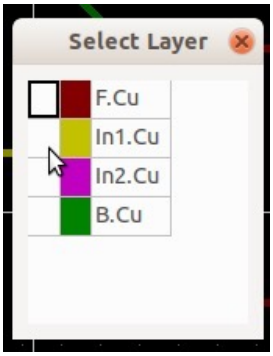


Figure 48.6: Select the layer to connect using a via.

Take a bit of time to experiment with these options and 'get the feel' of using vias to connect specific layers. In Figure 48.7 you can see my practice session. Vias allow a trace to continue between layers. Each layer is represented by a different colour.

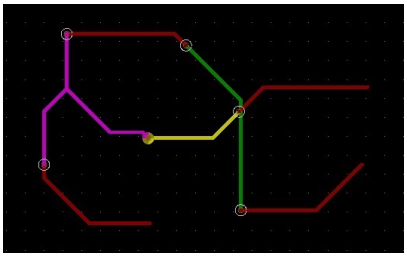


Figure 48.7: Practicing via placement.

49. How to use buses

In cases where you have several wires that belong to the same functional group, buses can help reduce clutter and risk of errors.

Let's look at an example. Say you want to connect a Z80 microprocessor to a memory chip. This connection requires a lot of wires for the address and data. In Eeschema, you would start with an arrangement like the one in Figure 49.1. Our objective is to connect pins A0 to A15 from the CPU to the pins with the same name on the RAM module and do the same for the data pins.

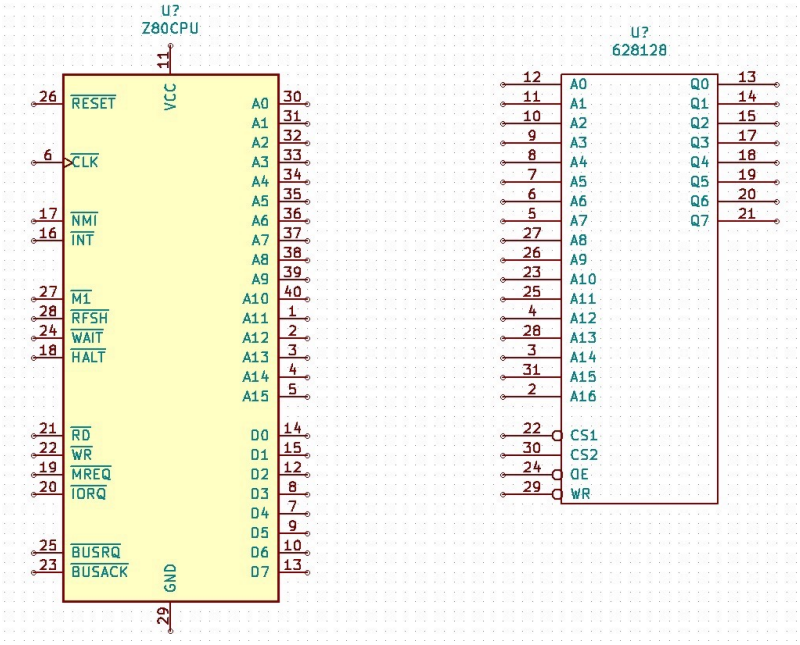


Figure 49.1: Let's connect the CPU to the RAM using buses.

You can use normal wires, and the schematic would be correct, although very (visually) busy. Instead, we will use the bus option, and create two busses, one for the address pins and one for the data pins. To do that, you will use two tools from the side menu, the Bus tool ( ), and the Bus Entry tool( ). Use the Bus tool first to draw a bus line in between the CPU and the RAM. In Figure 49.2 I have drawn a bus with this particular shape. It doesn't have to look like this; the exact shape is up to you.

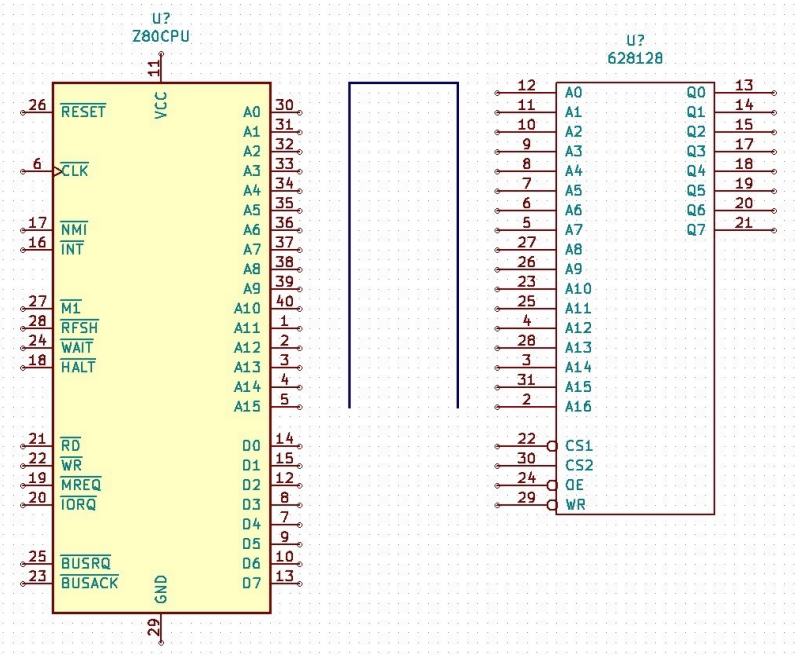


Figure 49.2: A bus is depicted by a thick blue line.

Next, use the Bus entry tool to create entry points from the pins to the bus. Attach the bus entries to the bus line (Figure 49.3), and then use a normal wire to connect the bus entry lines to the CPU pins if the bus entry lines themselves are not long enough (Figure 49.4). You can also move the bus entry line if needed to align them better.

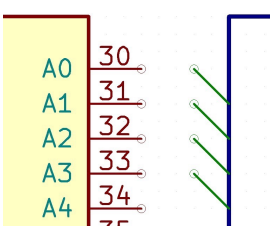


Figure 49.3: Bus entry lines for pins A0 to A3.

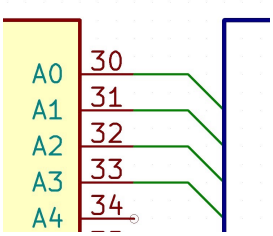


Figure 49.4: Using normal wires to connect the pins to the bus entries.

Do the same thing on the RAM module side. Your schematic will look like the example in Figure 49.5. In this example, I have grounded A16 of the RAM module since the CPU address bus can only drive 16-bit addresses (not 17 bits, as the RAM module is capable of).

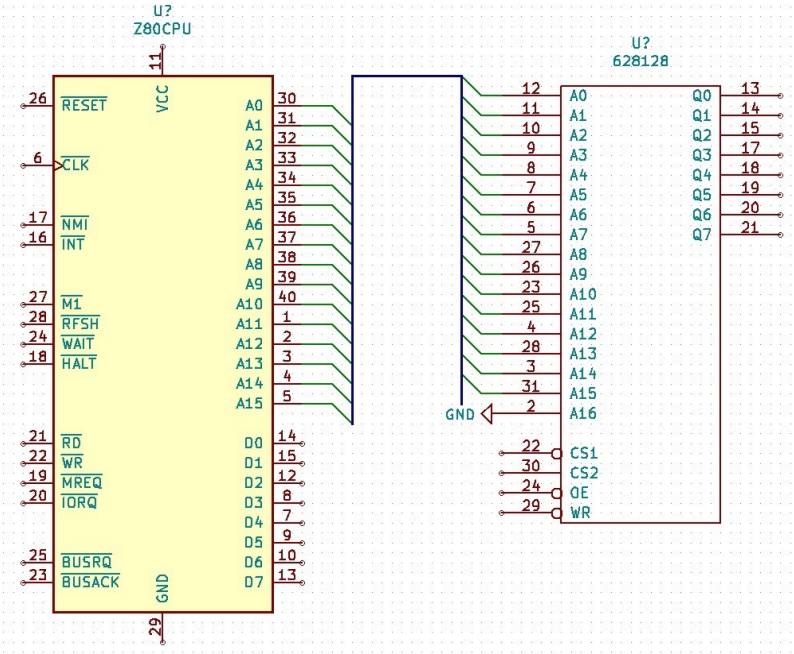


Figure 49.5: The Address bus, unlabelled.

To complete the bus wiring, you need to label each bus entry point. The labels allow Eeschema to know which pins within the bus are electrically connected. To quickly set create the labels, use the net label button, label the first net with 'A1', and then continue by using the Insert key to insert the rest of the labels automatically. Each time you press Insert, the next label, properly numbered, will appear. When you complete all labels on the CPU side, up to A15, manually create the next label as 'A0' and continue using the Insert key. The final result looks like the example in Figure 49.6.

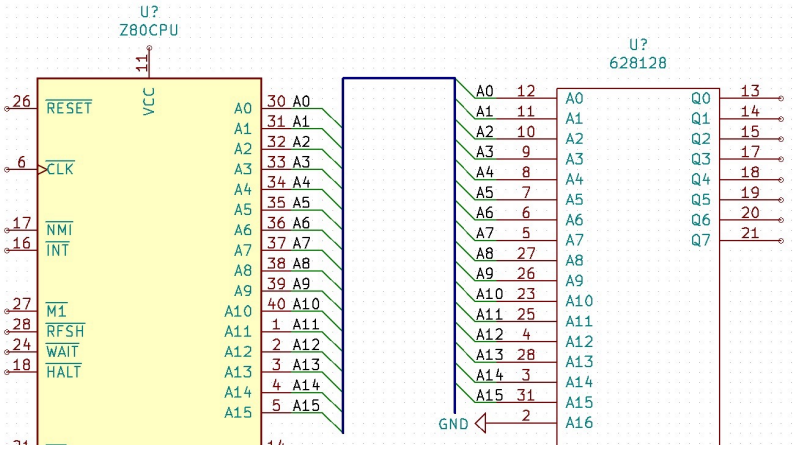


Figure 49.6: The address bus is now labeled.

Repeat the same process to wire the data bus. The result is in Figure 49.7.

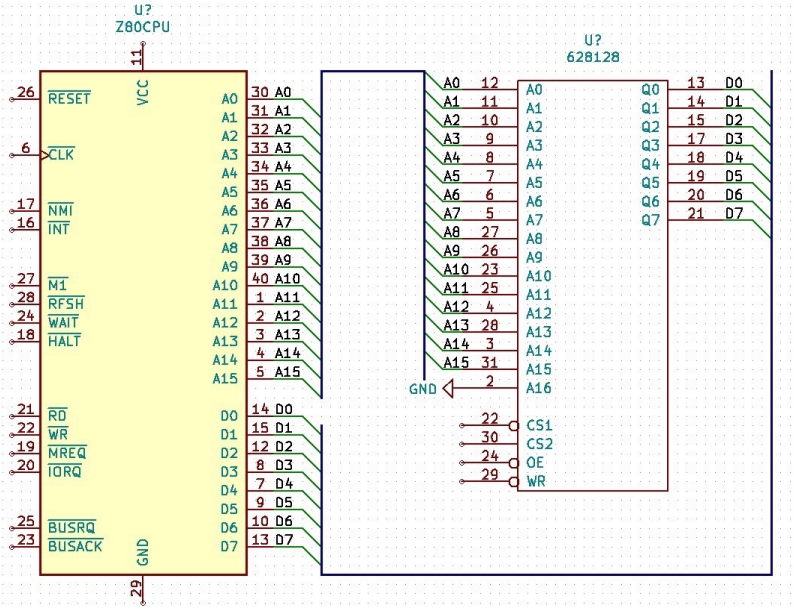


Figure 49.7: The data and address buses are fully wired.

As you can see, the resulting schematic is clear and easy to read. The two buses contain invisible but real electrical connections. When you import the netlist file into Pcbnew, these connections will be converted into individual tracks.

50. How to update your schematic and layout (with Git)

Printed circuit board design is an iterative process. You'll work on a schematic, continue with the layout when you decide to make a change and have to go back to the schematic. In the projects of this book, the process you experienced was mostly linear. When you work on your own projects, you will be working in a more iterative process.

In this recipe, you will learn how to make a change in the project schematic and carry this change over to the layout. To demonstrate, I will use the second project of this book.

At the end of this project, the board looked like the one in Figure 50.1.

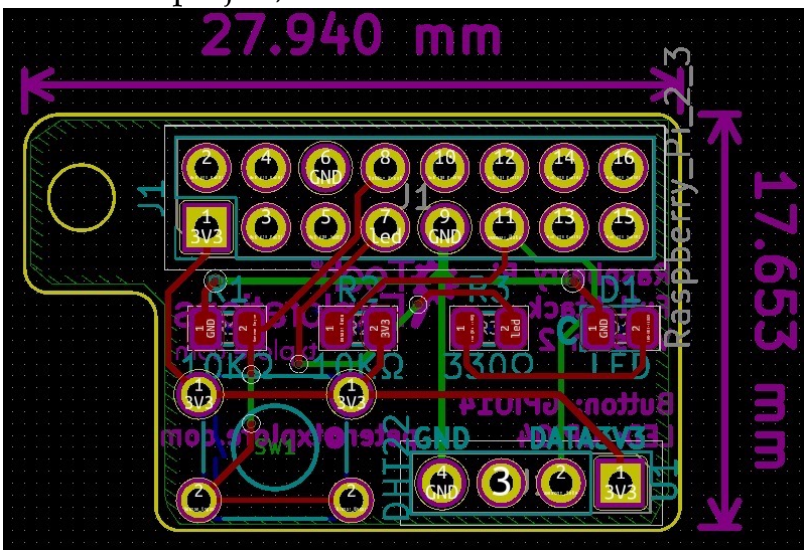


Figure 50.1: You will make a change to the schematic of this board.

Say that you'd like to add a second LED, connected to pin 1, to indicate that power on the Raspberry Pi is on. This requires a change to the schematic. Then you have to carry this change over to the layout so that you can update the board.

As an aside, I will do this process in a new Git branch of this project. You can safely ignore this if you simply want to learn how to make a change to the board. If you want to know how to use Git to maintain the history of a project, please read the recipe titled '47. Using Git for version control'.

Using the terminal, browse to your project:

```
$ cd /home/peter/Documents/KiCad\ projects/KLP2/RPi\ FS\ HAT/
```

Create a new Git branch for the project:

```
$ git checkout -b add_power_led
```

```
Switched to a new branch 'add_power_led'
```

Make sure that you are working in the new branch:

```
$ git branch
* add_power_led
  master
  net_names
```

The indicator '*' is next to the new and active branch name. This means that any new commits will go to this branch, which is what you want.

Start KiCad and open your KiCad project. Start Eeschema. Add the LED with a resistor connected to the GND and 3V3 nets (Figure 50.2).

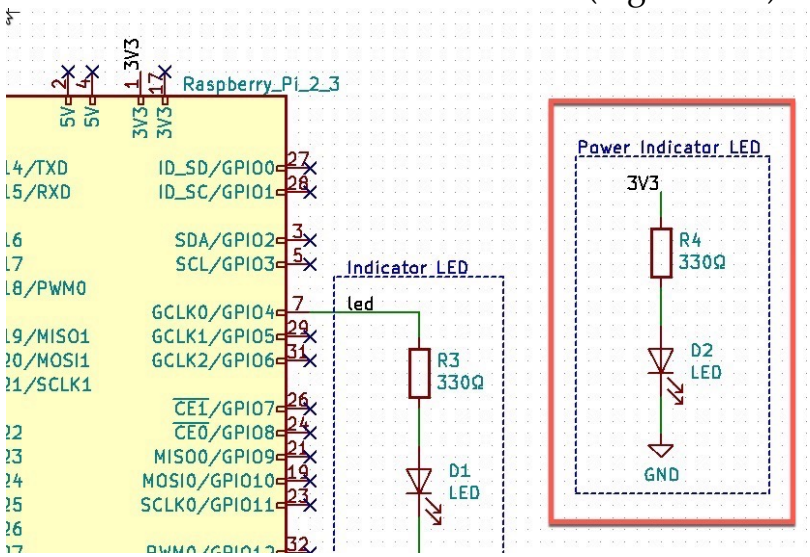


Figure 50.2: The new elements are indicated by the box.

Of course, remember to annotate the new symbols. Assign footprints to the new symbols using Cvpcb (Hint: if you copy the existing resistor and LED, the new components will preserve their associations with the footprints so you will not need to associate them manually). You can see the associations of the updated schematic symbols in Figure 50.3.

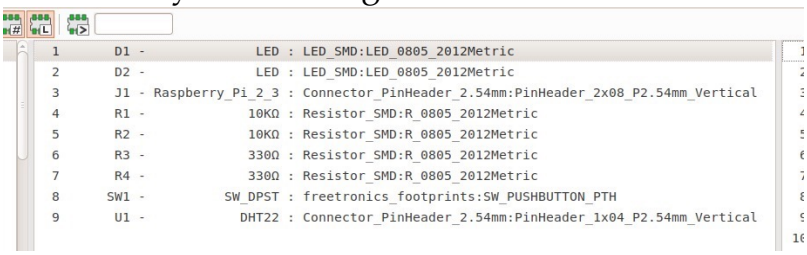


Figure 50.3: The associations of the updated schematic symbols.

Do an Electrical Rules Check to make sure nothing is broken. Then, export a new netlist file (you may overwrite the old netlist), and close Eeschema.

At this point, commit the changes to Git:

```
git commit -am 'Added new symbols.'
[add_power_led 54ade43] Added new symbols.
```

```
 2 files changed, 117 insertions(+), 36 deletions(-)
```

Start Pcbnew and import the new netlist file. It is worth spending a bit of time to carefully select the appropriate options from the Netlist import window (Figure 50.4). In this example, we have added two new symbols. We have not deleted any symbols, and we have not made any changes in associations. This is a fairly simple scenario, and the default options in Figure 50.4 will work well.

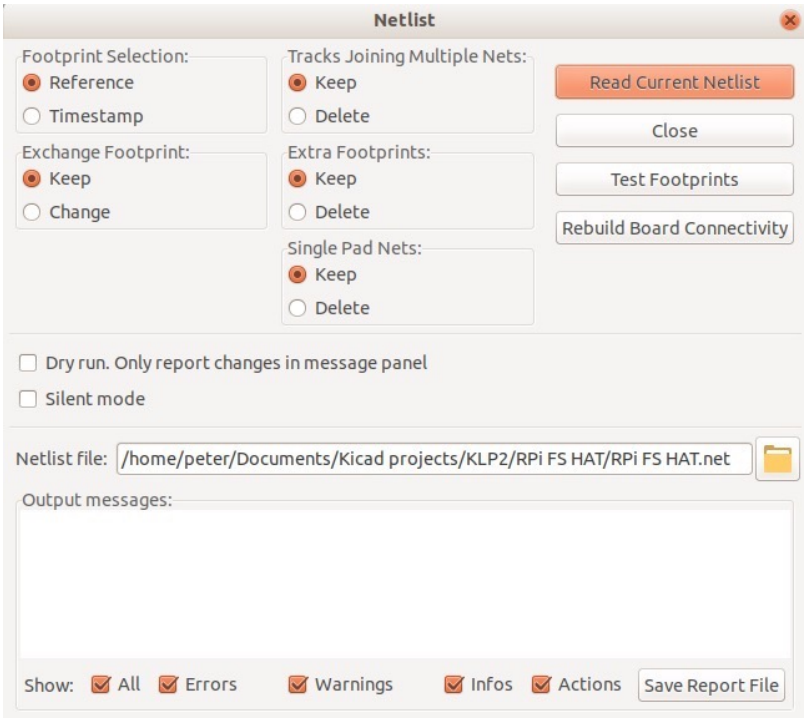


Figure 50.4: Select the appropriate import options.

If you had deleted a symbol, you should opt for the 'Delete' option under 'Extra Footprints'. This will delete any footprints that exist in the layout but not in the schematic. If you made changes to the associations, you should opt for the 'Change' option under 'Exchange Footprint.' As you expect, this would change the footprint of an existing symbol.

When you submit the importer, the new footprints will appear where your mouse pointer happens to be (Figure 50.5).

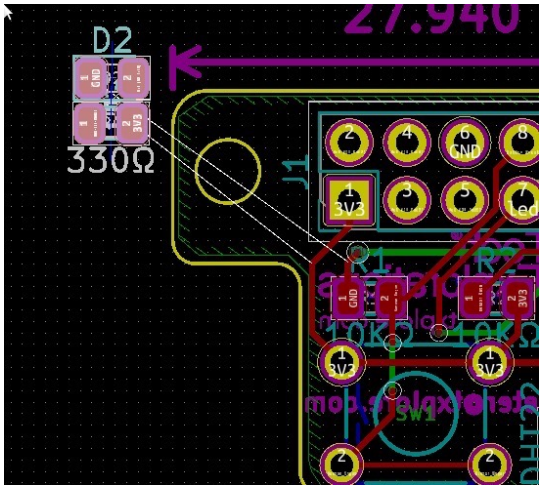


Figure 50.5: The new footprints.

Place the footprints as usual. You may need to make more space by adjusting the Edge.Cut drawings. You can see my final version of the updated board in Figure 50.5.

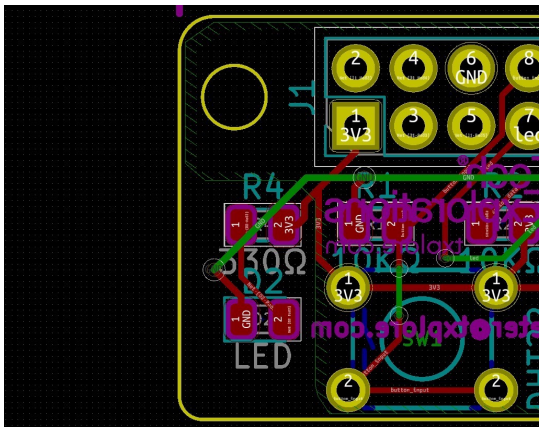


Figure 50.5: The final version of the board.

Remember to update the Git repository with the latest version of the board:

```
$ git commit -am 'Completed update of the board.'
[add_power_led fe394cb] Completed update of the board.
 2 files changed, 946 insertions(+), 20 deletions(-)
```

51. Starting KiCad apps individually

KiCad is a collection of apps that are bound together through the project '.pro' file and the main KiCad window. Despite that, you can use the individual apps in stand-alone more.

Say, for example, that you would like to have a quick look in the schematic file of one of your projects. You don't need to start the main KiCad window and then start Eeschema. You can start Eeschema directly.

Try this now. Go into the directory for one of your projects, and locate the file that ends with '.sch'. Right click on the file, and select on the 'Open with Eescheam (Standalone)' option (Figure 51.1).

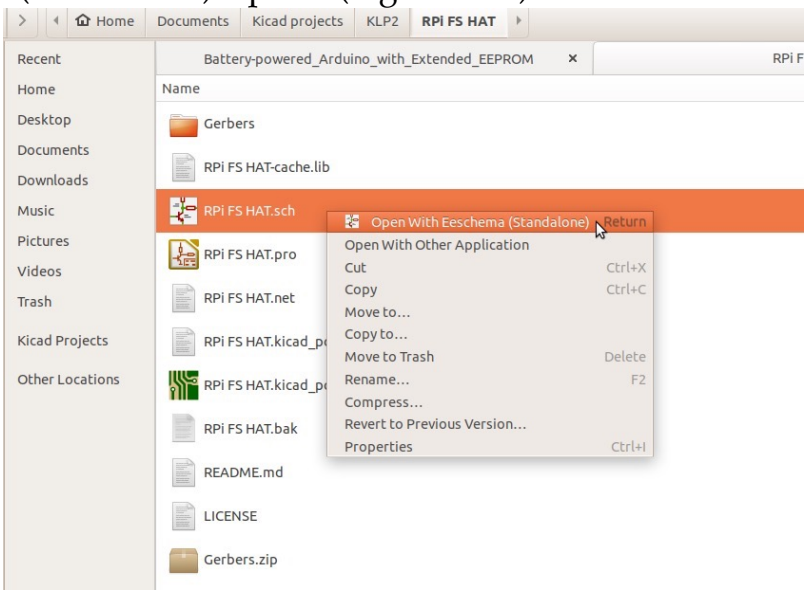


Figure 51.1: Start Eeschema in stand-alone more.

Eeschema will start, and contain the schematic (Figure 51.2).

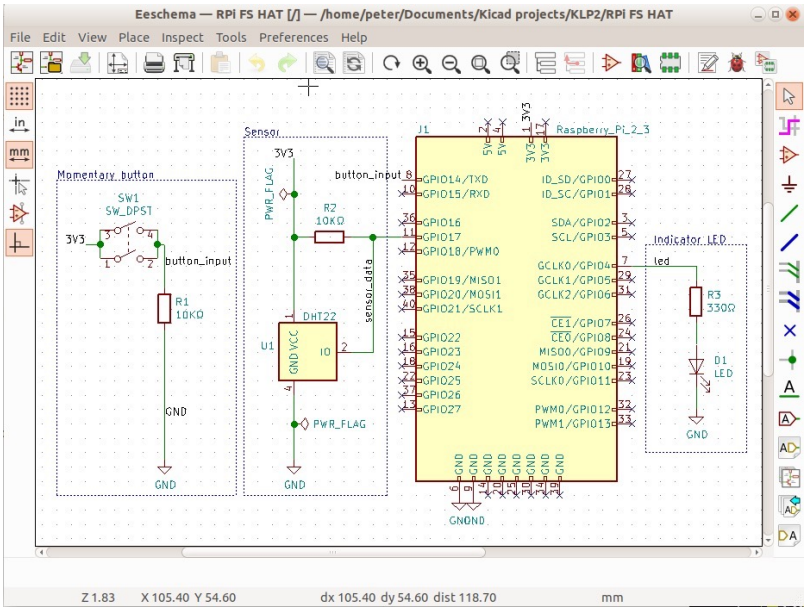


Figure 51.2: Eeschema in stand-alone more.

You can do the same thing with Pcbnew; look for the file with the '.kcad_pcb' extension.

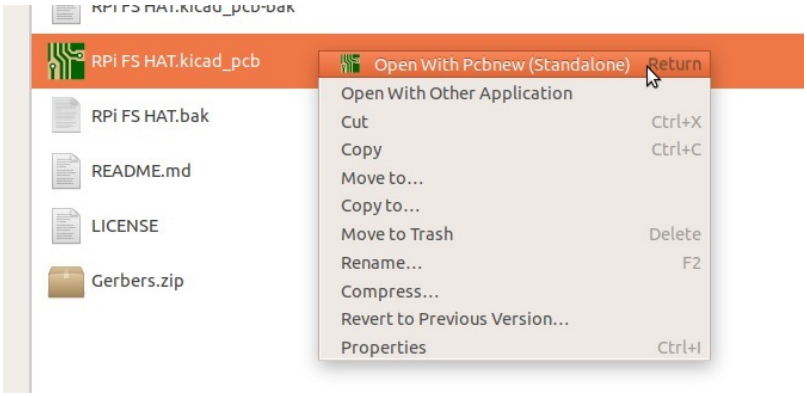


Figure 51.3: Start Pcbnew in stand-alone more.

52. Creating a new version of a PCB without altering the original

Once you have your schematic ready, you have a wide range of options available to you in regards to how to design the actual PCB. You may choose to go for an SMD version so that the final board is small in size. You may choose to go for a TH version so that the assembly is easier (although with a bit of practice, you can solder SMD components just as easily). You may want to experiment with different shapes to accommodate different end-usage scenarios.

Whatever you choose, it is possible to have a single project, with a single schematic, but with multiple layouts. Assuming that the schematic between the various versions of your final PCB remains the same, the only differences will be in the associations and the layout. Still, those differences affect most of the project files:

- The schematic '.sch' file will store the associations for the board-specific footprints,
- The layout 'KiCad_pcb' file will store the board-specific layout,
- The netlist 'net' file will also store the board-specific net and footprint information
- and, of course, the Gerber files will depend on the board-specific layout

The best way to go about doing this is by using Git. With Git, you will be able to store these board-specific versions of the project files safely, without affecting the other board versions. For each board version, you can create a separate branch. For example, if you have three layouts as your target, 'small', 'medium' and 'large', you can implement them by creating three branches named appropriately.

If you haven't read the recipe titled '47. Using Git for version control' yet, please do so now.

Let's go ahead and look at an example. Let's create an extended version of the board from Project 3, the Arduino clone. I would like to modify that board so that it contains a prototyping area. This will involve enlarging the board outline, adding pads and a couple of power rails for 5V and GND.

Use your terminal window to navigate to your project folder. Check that all your latest work is committed to Git:

```
$ git status
On branch master
nothing to commit, working tree clean
```

If you have uncommitted changes, commit them now so that your working tree is clean.

Continue to create a new branch. I will call my new branch 'proto-area' to indicate that the new board version will have a prototyping area.

```
$ git checkout -b proto-area
Switched to a new branch 'proto-area'
```

You can use the next command to see the branches in your project, and double-check that you are working in the new branch before you continue:

```
$ git branch
  master
* proto-area
```

The proto-area branch is active, so we can proceed. Start KiCad and open the project. I will add the prototyping area in a new hierarchical sheet in the schematic, export the new netlist, and adjust the layout in Pcbnew. I will not change any existing footprints, only add the footprints for the prototyping area. The new footprints will consist of individual pads and a couple of pin connectors for the rails.

In Figure 52.1 you can see the sheet symbol in the Eeschema root sheet, and in Figure 52.2 the new prototyping area sheet.

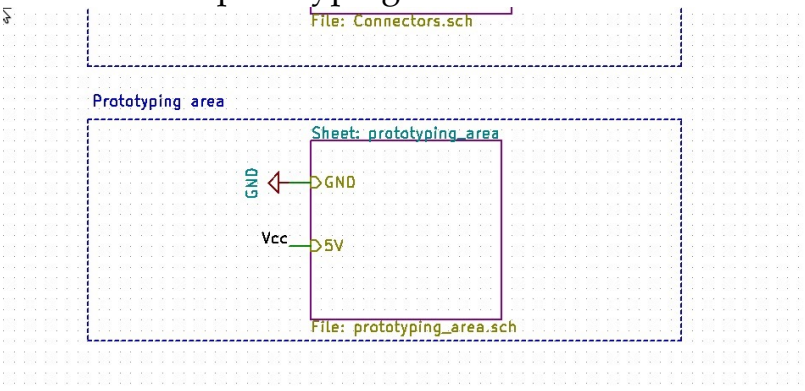


Figure 52.1: The new sheet symbol in the project root sheet

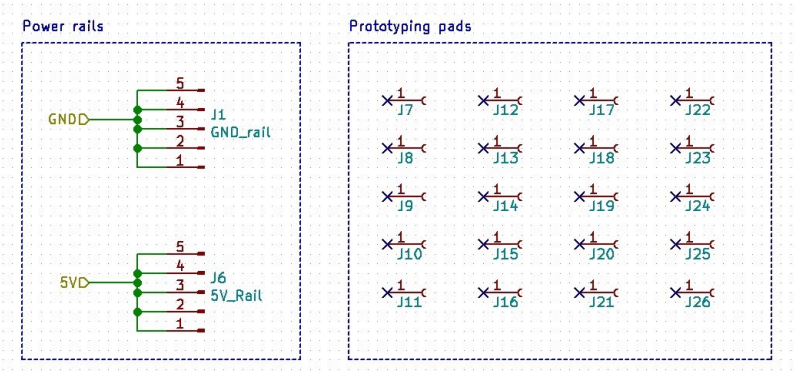


Figure 52.2: The prototyping schematic in the prototyping sheet.

Save the changes in Eeschema. This is a good point in time to commit these changes to the proto-area Git branch. In the terminal, type this:

```
$ git status
On branch proto-area
Changes not staged for commit:
  (use 'git add <file>...' to update what will be committed)
  (use 'git checkout -- <file>...' to discard changes in
working directory)

   modified:   BACEE-cache.lib
   modified:   BACEE.bak
   modified:   BACEE.sch
   modified:   Connectors.bak
   modified:   Connectors.sch

Untracked files:
  (use 'git add <file>...' to include in what will be
committed)

   prototyping_area.bak
   prototyping_area.sch

no changes added to commit (use 'git add' and/or 'git commit
-a')
```

Git is telling you that there are several changed files and two untracked files that belong to the new sheet. Add the new files to the repository:

```
$ git add .
And commit all new changes to the current working branch:
$ git commit -am 'Created schematic for prototyping area.'
[proto-area ec43d9a] Created schematic for prototyping area.
 7 files changed, 961 insertions(+), 151 deletions(-)
 create mode 100644 prototyping_area.bak
 create mode 100644 prototyping_area.sch
```

Continue to do an electrical rules check, and then use Cvpcb to associate the new symbols with footprints. You can see the new associations in Figure 52.3.

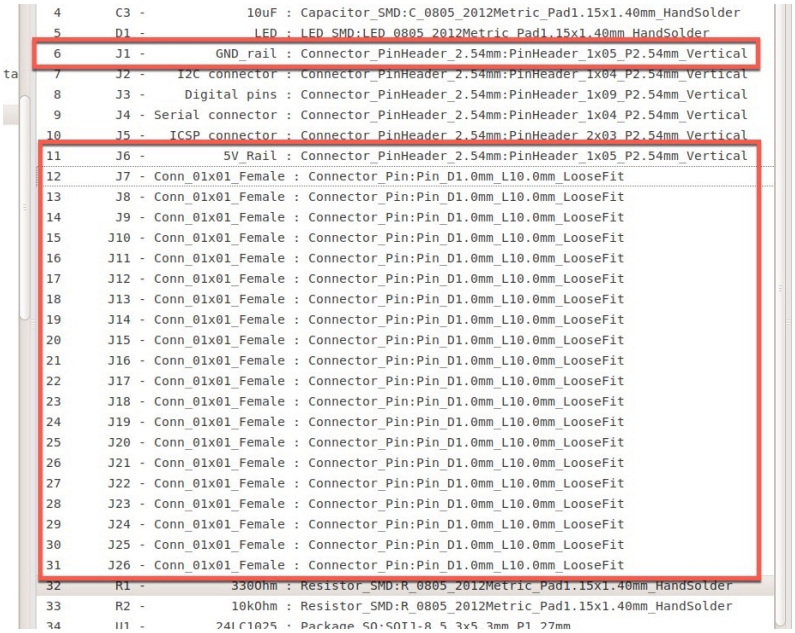


Figure 52.3: The associations for the new symbols.

Time to generate the netlist file, and continue with Pcbnew. I used the title 'BACEE_extended.net' for the new netlist file, to differentiate it from the existing board file (Figure 52.4.

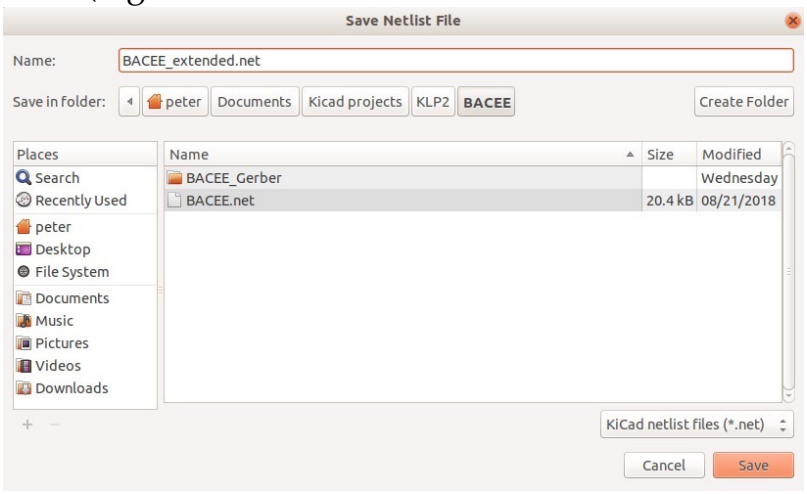


Figure 52.4: Use a descriptive name for the board-specific netlist file.

In Pcbnew, open the new netlist file. You will have to browse for it if you used a custom name, as I did (Figure 52.5).

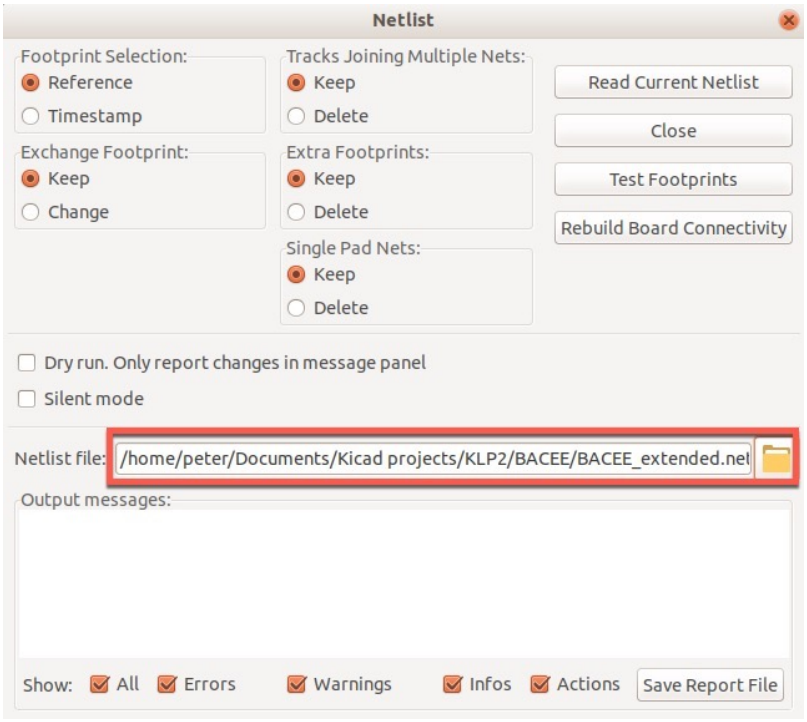


Figure 52.5: Load the new netlist file.

The new footprints will appear in the sheet (Figure 52.6).

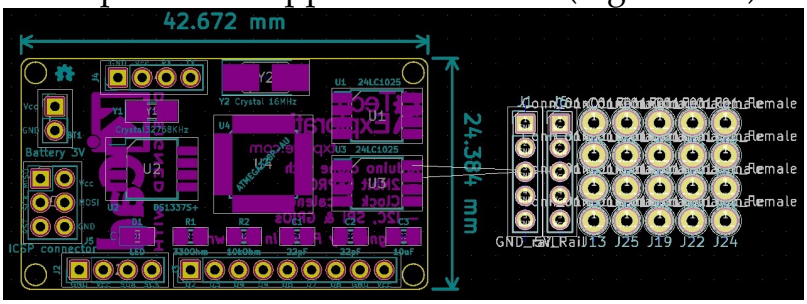


Figure 52.6: The new footprints appear in the sheet.

This is a good time to commit the changes to Git. The changes include the new net file and an updated project file. Use 'git add .' and 'git commit -am 'Created new netlist file' to do so.

Next, work on changing the outline of the board to accommodate the new footprints, and then placing and wiring them. I decided to keep the board at the same width but make it longer.

You can see the result in Figure 52.7 and Figure 52.8.

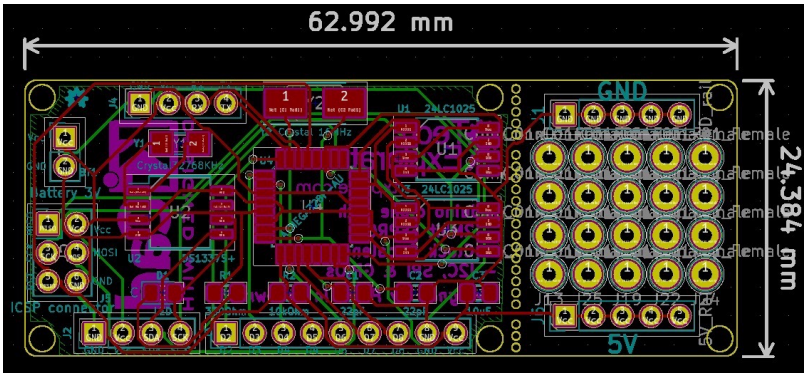


Figure 52.7: The extended board.

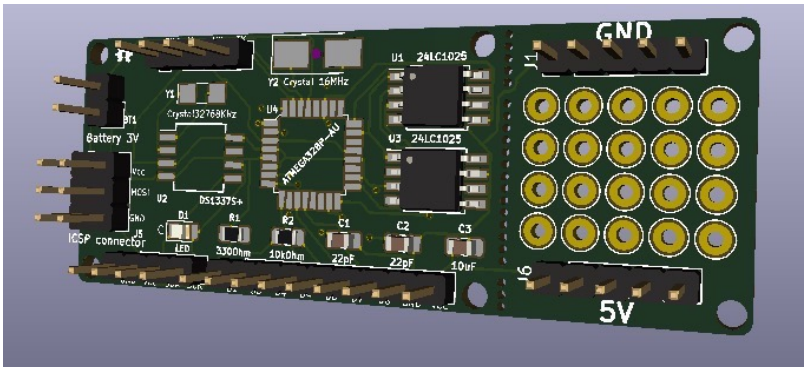


Figure 52.8: The 3D rendering of the new board version.

In the new board version, you can see the prototyping area pads and the power rails. I have added additional holes with a 0.35 mm radius in between the prototyping area and the rest of the board to make it easy to snap-off the prototyping area if it isn't needed. I also added a couple of additional mounting holes.

You can proceed to do a DRC check and then export the Gerber files. Then, save the project files (especially the Pcbnew file), and do one last commit:

```
$ git commit -am 'Completed design of extended board.'
[proto-area ccb1cd7] Completed design of extended board.
 2 files changed, 940 insertions(+), 136 deletions(-)
```

All your work is safe in the Git repository, inside the proto-area branch. Let's say you want to go back to the original version of the board, which is stored in the 'master' branch. First, close KiCad and all its applications. Then, type in this:

```
$ git checkout master
Switched to branch 'master'
```

Go back to KiCad, and open Eeschema and Pcbnew. Your original version of the board is still there, intact. Want to go back to the extended version? No problem, close KiCad and type this:

```
$ git checkout proto-area
Switched to branch 'proto-area'
```

Isn’t this awesome?

53. Making a PCB without a schematic

Sometimes you just want to 'hack' a PCB together. You are so confident that you know what you are doing that you are willing to forgo the schematic step, the ERC, and the design rule checks in Pcbnew, and just produce a PCB. You may be copying an existing PCB.

In that case, you can use Pcbnew autonomously, and disable design rules checking. Here's how it works.

Start by launching Pcbnew directly, not through the main KiCad window. You can do this by double-clicking on the Pcbnew application icon, or by searching for 'Pcbnew' in your operating system's launcher. In Ubuntu, I search for 'pcbnew' in the Activities tool (Figure 53.1.

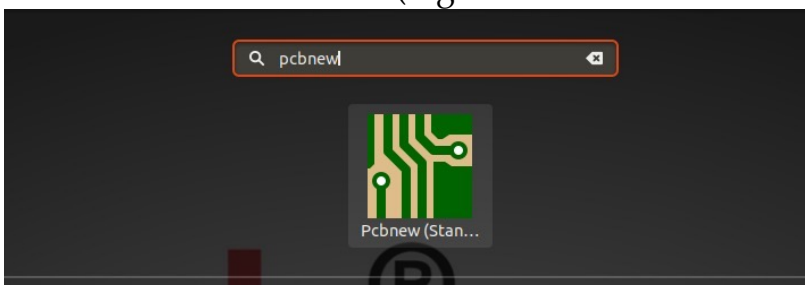


Figure 53.1: Start Pcbnew directly.

When Pcbnew starts, disable the design rules check by clicking on the first button in the left toolbar.

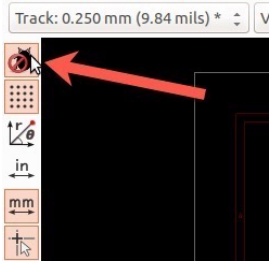


Figure 53.2: Disable design rules check.

You are now free to create your board; free from all checks and restrictions. You can add footprints, traces, outlines. You will not be able to use the autorouter since there is no information available for pad and net relationships.

You will not be able to assign traces to nets, but you will be able to control trace width by manually setting custom trace widths in the global design rules (Figure 53.3).

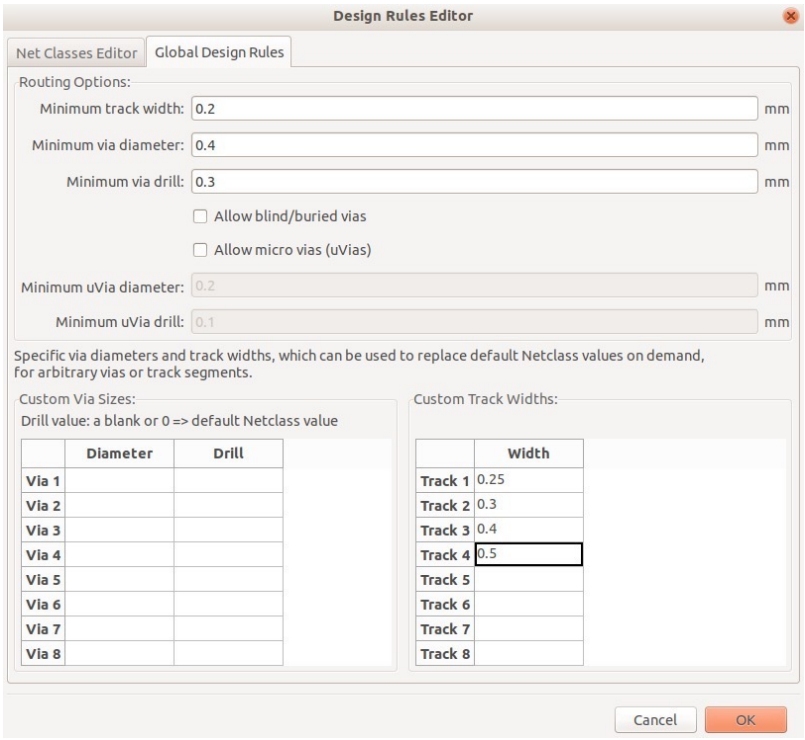


Figure 53.3: Your traces can have one of the custom widths.

In Figure 53.4, you can see a very rough board I designed without a schematic. Although I created a copper pour in the back copper layer, I did not connect it to a net, since there are no nets. However, you can easily connect the copper pour to any pad using a small trace segment.

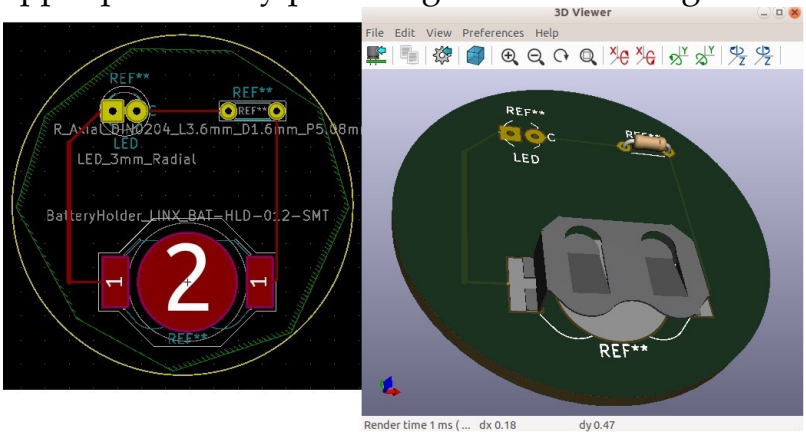


Figure 53.4: A board, without a schematic.

54. How to set a text editor and why

All of the data that makes up a KiCad project is stored in plain text files. You can open those files and edit them manually using a text editor. If you have strong programming skills with experience in text processing, you are also able to edit the project files programmatically. Kicad power users often do this, and the results are close to magic.

In the main KiCad window, you can set your preferred text editor. Doing so gives you a shortcut for opening project files quickly, through the KiCad window. You can see this in Figure 54.1.

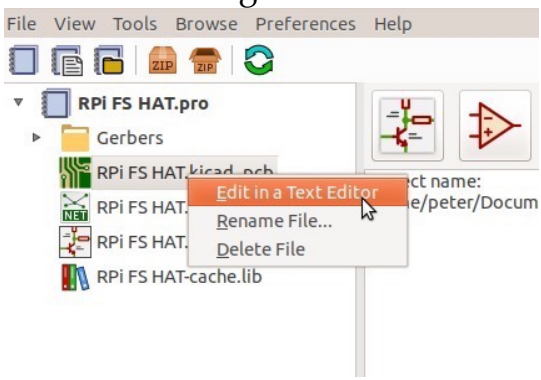


Figure 54.1: Once set up, you have a shortcut for viewing project files in a text editor.

When you open, say, the layout data files in a text editor, its contents look like the example of Figure 54.2.

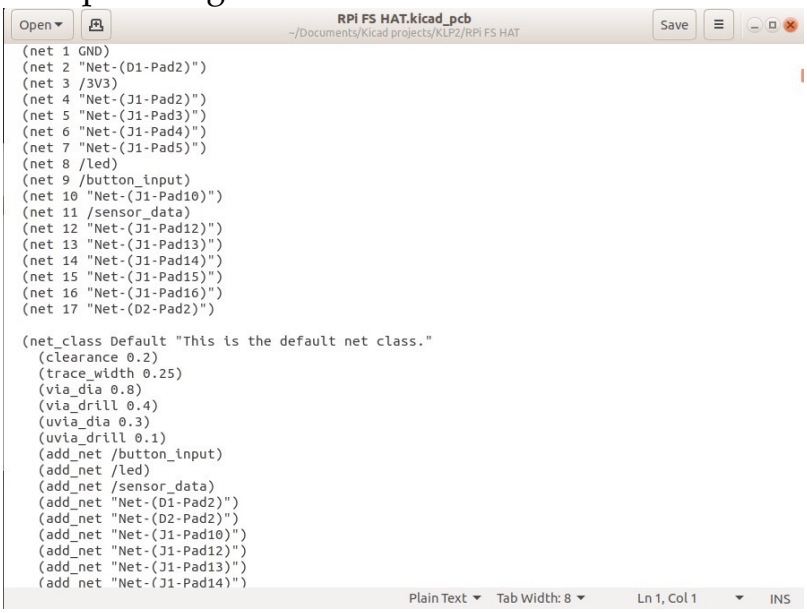


Figure 54.2: An example of a layout data file, in clear text.

To set your preferred text editor, open the KiCad main window and click on 'Set Text Editor…' (Figure 54.3).

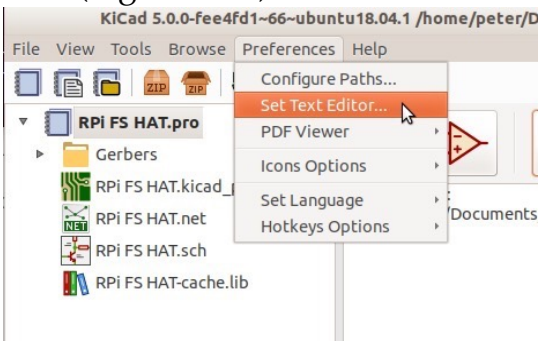


Figure 54.3: Set Text Editor.

The file browser will appear. Navigate to the location of the text editor program file, and select it. In my example, I am setting gEdit as my preferred text editor (Figure 54.4).

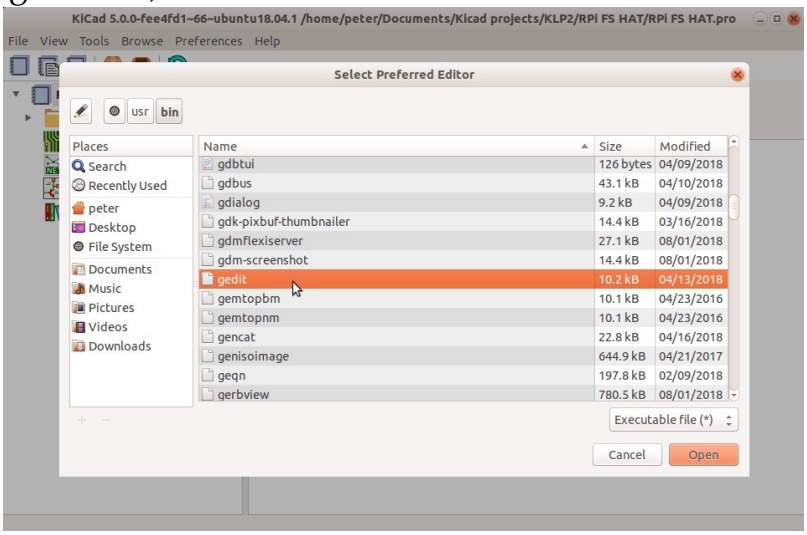


Figure 54.4: My preferred text editor.

Now, you will be able to right-click on any file in the Kicad project that is listed in the KiCad main window, and select “Edit in a Text Editor” to open it.

55. How to install 3D shapes

KiCad's 3D viewer can display 3D versions of footprints as long as they are installed. Because of the file size of those 3D shapes, they are not installed by default but are available for later installation through a wizard.

To install the 3D shapes, start Pcbnew, and start the Wizard by clicking on Preferences, 'Add 3D Shapes Libraries Wizard…'.

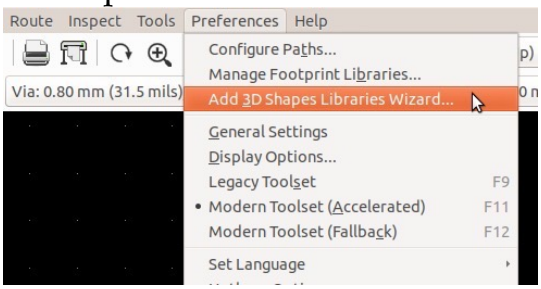


Figure 55.1: Start the 3D install wizard.

The wizard will ask you a few questions about which 3D shape libraries you want to install, and where. I choose to install them in my KiCad libraries directory (Figure 55.2).

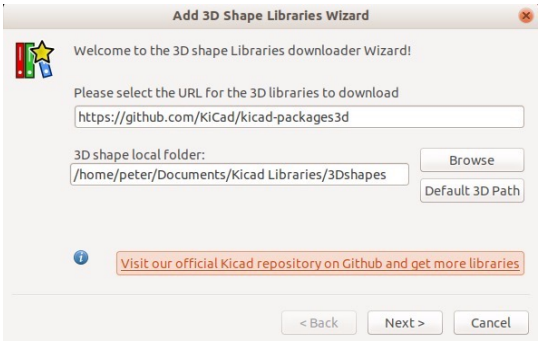


Figure 55.2: Define the 3D shapes library location.

Next, the wizard will ask you to select the libraries you want to install. I select them all so that I can have any existing libraries updated, and new ones downloaded (Figure 55.3).

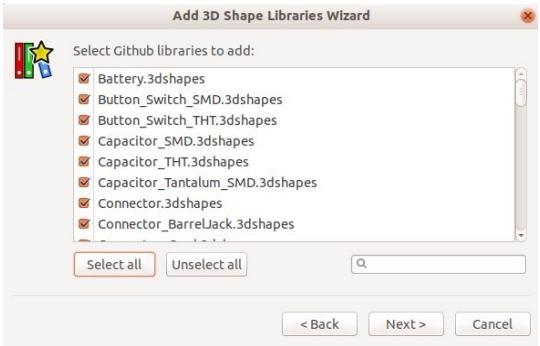


Figure 55.3: Select the libraries to install.

Next, the wizard will give you a review of the work that it will do. Existing libraries will be updated, and new ones will be installed fresh.

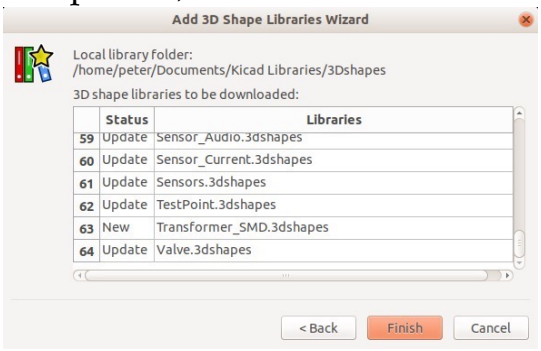


Figure 55.4: Review the libraries that will be downloaded.

Click on Finish to start the download. For slow Internet connections (like mine), you may want to consider doing this overnight. It may take a few hours to complete (Figure 55.5).

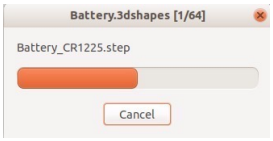


Figure 55.5: The download process will take up to a few hours.

In the end, the 3D viewer will be able to use 3D shapes for many of the most common footprints, but not all. In the example of Figure 55.6, you can see a board with some of its footprints represented using available 3D shapes. Not all of the footprints do.

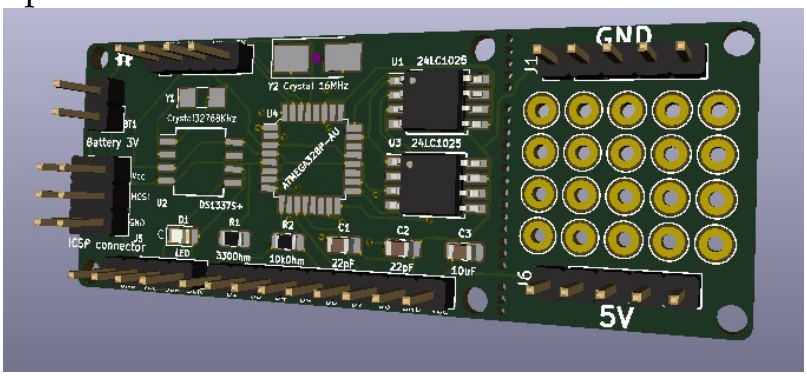


Figure 55.6: Many, but not all, footprints have 3D shapes.

56. How to change footprint in Pcbnew and back-import footprint symbol associations to Eeschema

This scenario is bound to happen before long: You have created your schematic, completed all the symbol and footprint associations, and created your layout in Pcbnew. With the layout complete, you realise that one of the footprints you chose is just not right, and that you should replace it with a better one.

In this recipe, you will learn how to change a footprint in Pcbnew, and update your schematic so that both schematic and layout files are in sync.

There are a couple of ways to deal with this situation. In any case, the end result of the footprint change process involves:

1. Changing the footprint for a symbol.
2. Updating the layout to use the new footprint.
3. Update the schematic symbol and footprint association.
4. Update the schematic in Eeschema with the new association.

If you don't do all that, your project will be out of sync, and new work will inevitably introduce confusion errors.

The best way to go about changing a footprint for a component after you have completed your layout, is to do the necessary work in Pcbnew and then use the back-import process to update the associations file and Eeschema. Let's have a look at how this works through a simple example.

Let's begin with the LED and resistor circuit from Part 2 of this book. The current iteration of this project and PCB, looks like this (Figure 56.1):

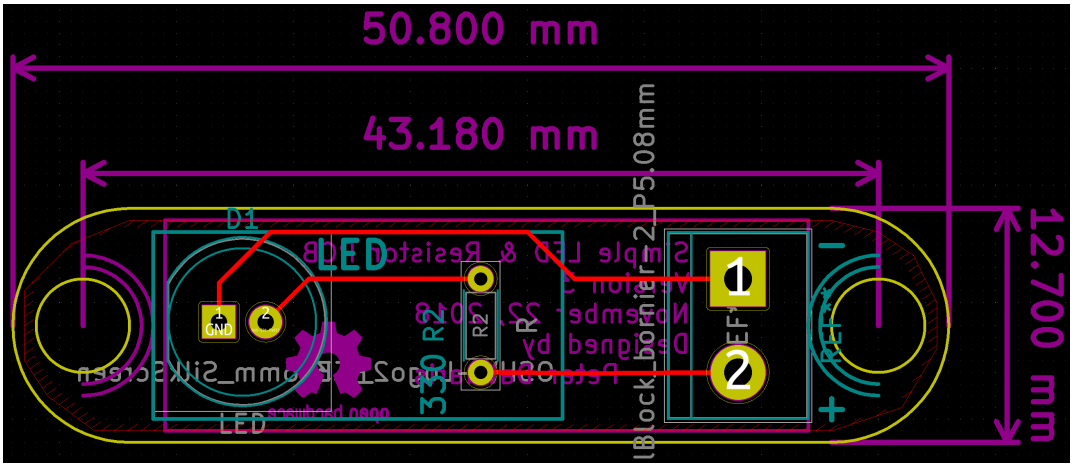


Figure 56.1: This is the current version of the example PCB.

Let's change the footprint of the LED so that we can use a surface mounted component, instead of the current through-hole component.

To change a single footprint, you can right-click on the footprint you want to change, and select "Change footprint". If you wanted to change several footprints at once, you can click on the "Change Footprints…" menu item, under Edit. Either way, the window that will come up is almost identical. For our example, we'll go with the single footprint change.

Right-click on the LED footprint and select "Change Footprint" from the context menu.

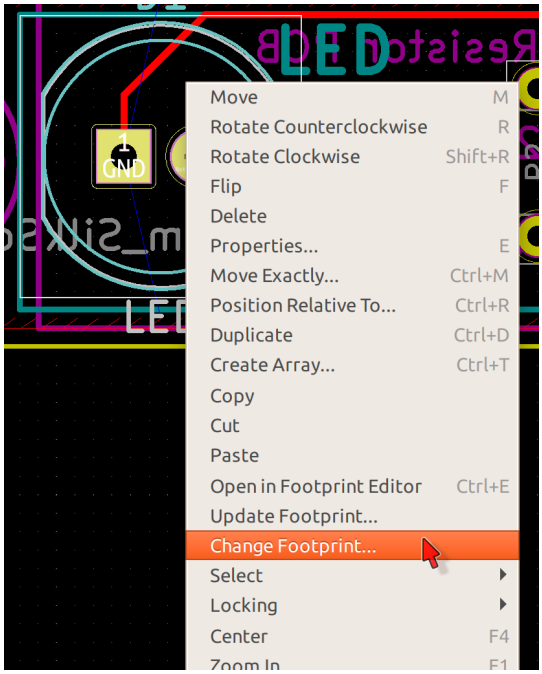


Figure 56.2: Change a single footprint.

The Change Footprints window will come up. From the available options, click on the "Change selected footprint" radio button (1), and then click on the library icon to select the new footprint from the library (2).

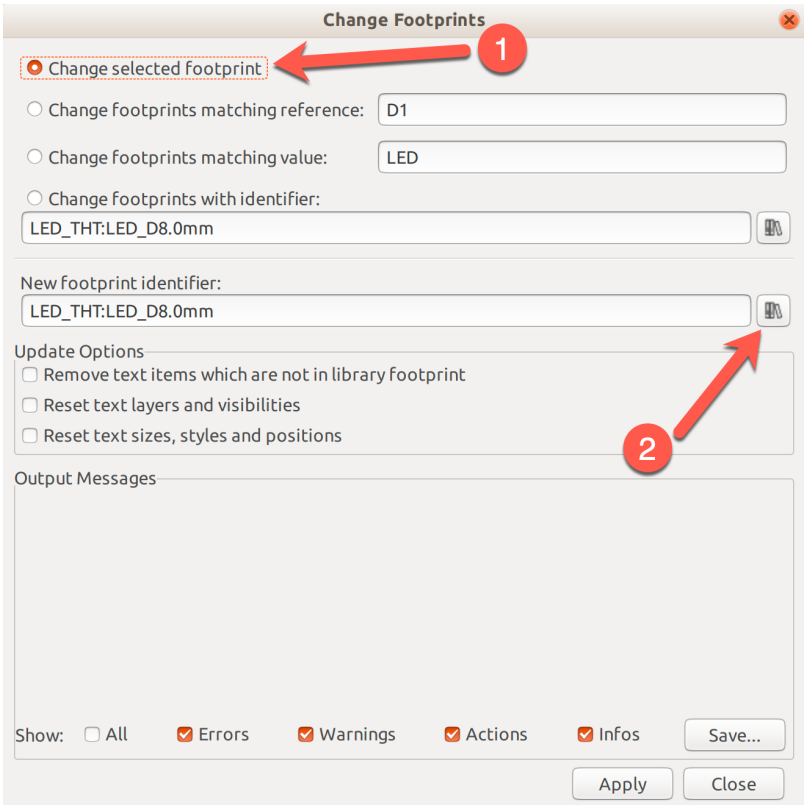


Figure 56.3: The Change Footprints window.

In the Footprint Library Browser window, find the SMD component you want. I will use the LED_0805 footprint since I have a lot of those in my drawers.

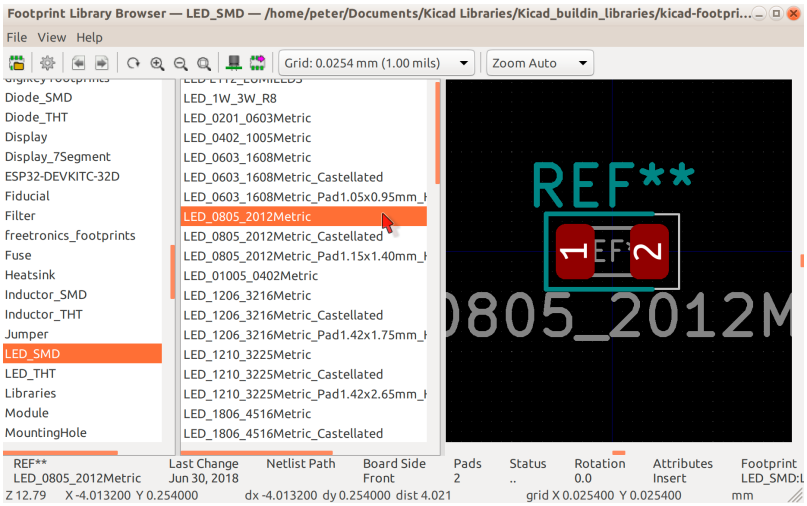


Figure 56.4: Choose the new footprint from the library browser.

Double click on the new footprint to select it. The new library identifier will appear in the relevant field in the Change Footprints window. Click Apply to commit the change, and Close to dismiss the window (Figure 56.5).

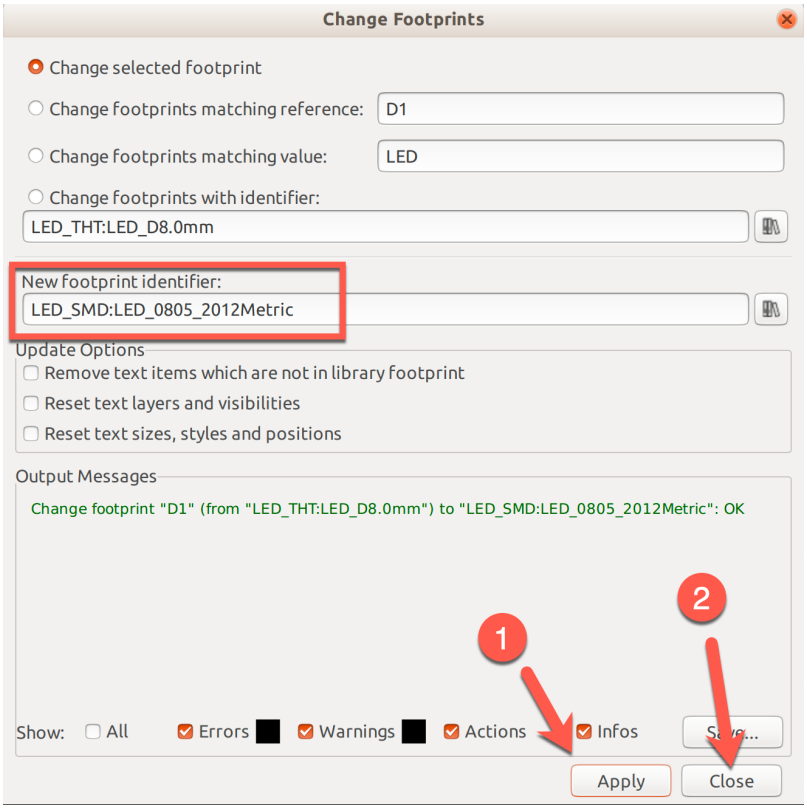


Figure 56.5: The new footprint identifier is listed.

Because the geometry of the new footprint is different to that of the original, we must re-draw the traces.

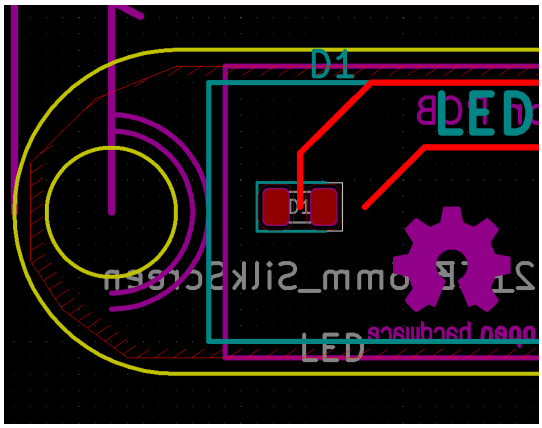


Figure 56.6: We must redraw the traces to work with the new footprint.

Remove the last couple of segments from the tracks that connected the old TH LED, and redraw them so that the new SMD LED is connected to the rest of the circuit.

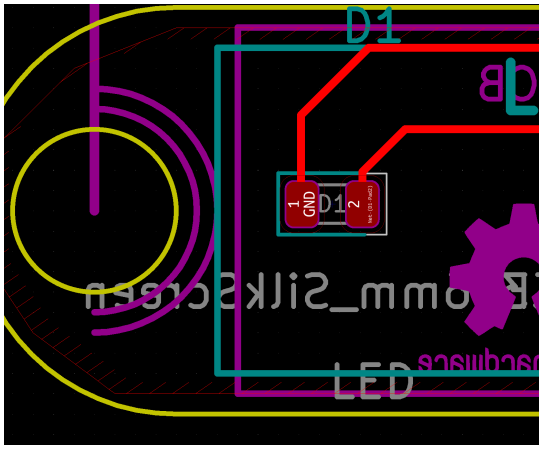


Figure 56.7: The new SMD LED is connected to the circuit.

Save your work in Pcbnew.

The new layout, that contains the new footprint/replacement of the old TH LED is ready. The last thing that remains to do is to update the schematic.

To do this, first export the Footprint Association File (.cmp), by clicking on File, Export, Footprint Association (.cmp) File. Give the new file a reasonably descriptive name. This is the file that contains the new associations information, which we will import in Eeschema.

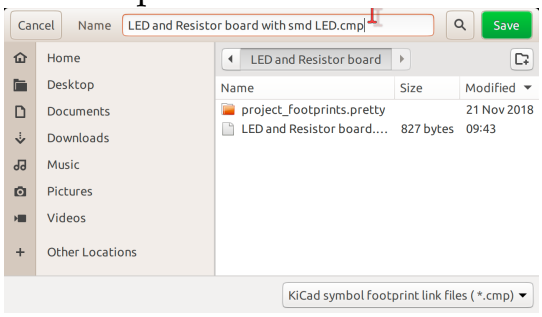


Figure 56.8: Save the Footprint Association File.

Now, we can continue to Eeschema. Open Eeschema by clicking on the Eeschema icon in the Pcbnew toolbar, or via the main Kicad window.

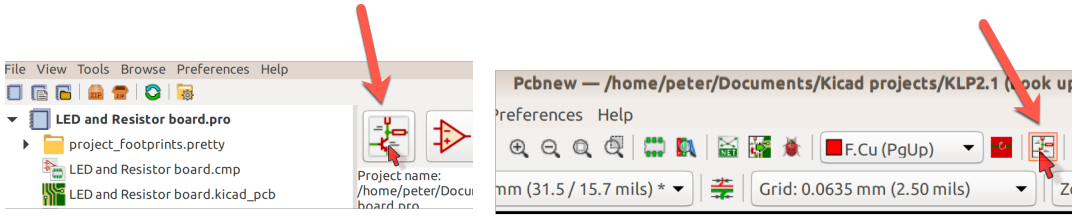


Figure 56.9: Open Eeschema to import the cmp file.

Before you do the import, have a look at the current association of the LED symbol with its footprint. Place your mouse over the LED symbol, and type "E" to bring up its properties (or, right click and select Properties, Edit Properties.

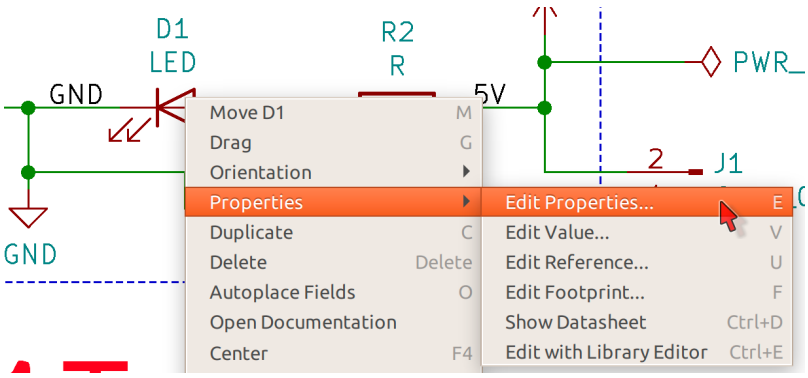


Figure 56.10: See the current properties of the LED symbol.

The LED symbol is currently associated with a through-hole footprint (LED_THT:LED_D8.0mm).

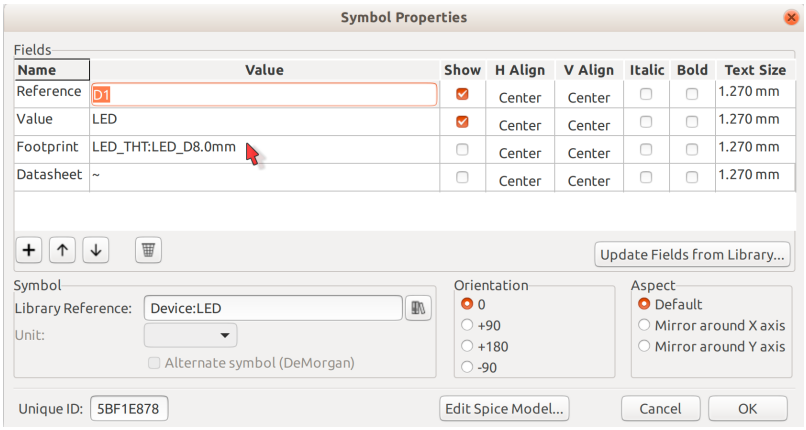


Figure 56.11: The current footprint association of the LED symbol.

Let's update the symbol association. Click OK to close the properties window. From the Eeschema top tool bar, click on the back-import button (first from the right), or File, Import, Footprint Association File…

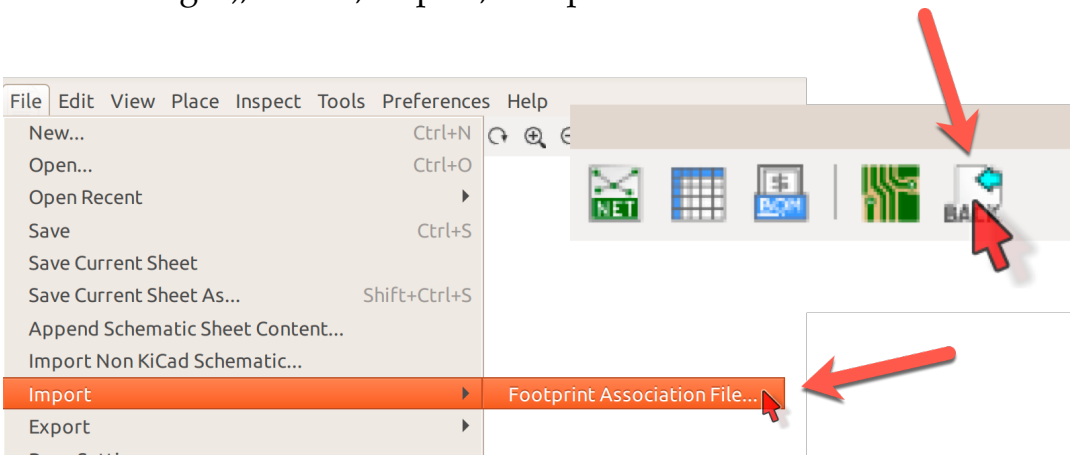


Figure 56.12: The current footprint association of the LED symbol.

Next, select the .cmp file you created earlier, and click on Open.

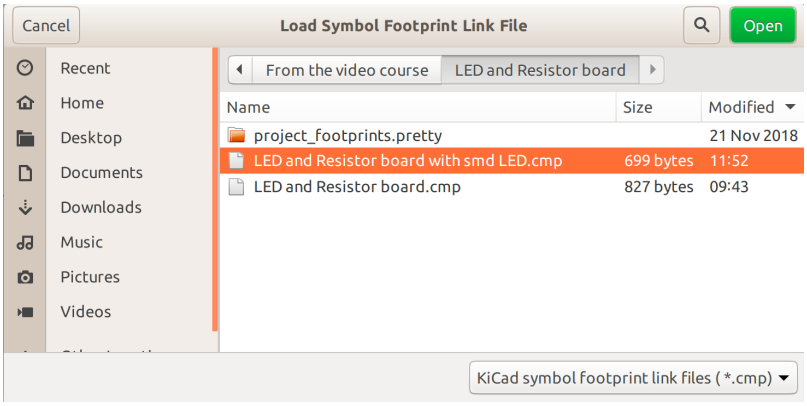


Figure 56.13: Select the .cmp file that contains the new association.

A new dialog box will ask you to keep the footprint visibility as-is, or show/hide all footprint field. I prefer to keep visibility as-is, but if you want to experiment, try "show all". To hide the fields, you will need to re-import the .cmp file and choose "hide all footprint field".

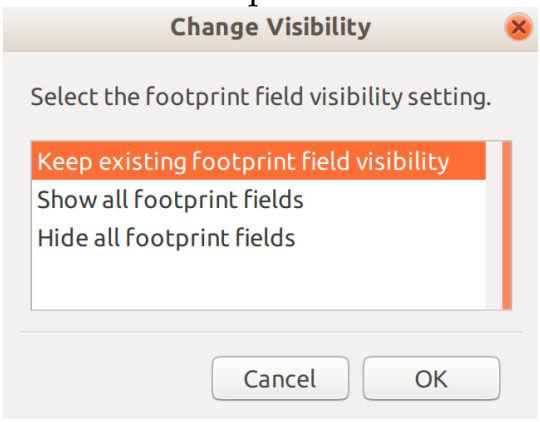


Figure 56.14: Choose your footprint field visibility settings.

Click Ok to commit the changes. Your schematic looks the same as before:

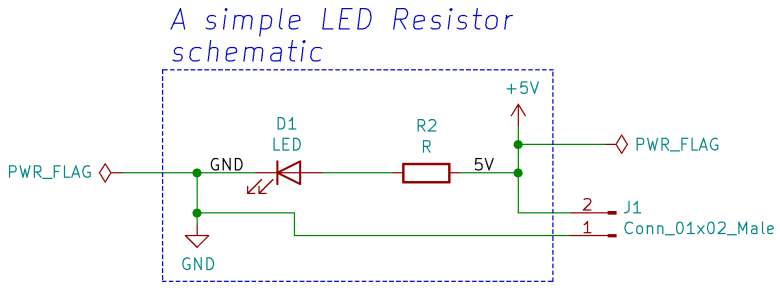


Figure 56.15: Your updated schematic; looks the same as before.

To verify the new association, bring up the properties window for the LED. It should look like this:

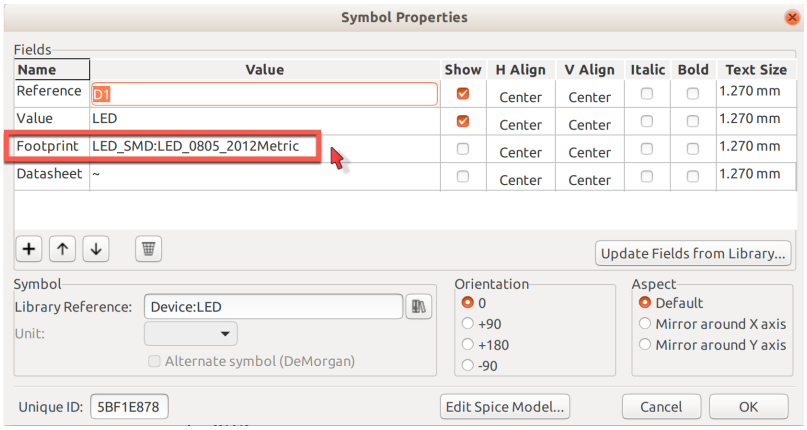


Figure 56.16: Verify the new symbol-footprint association.

As you can see, the LED symbol is now associated to the LED_SMD:LED_0805 footprint.

With this, the process of changing a footprint is complete.

What if your project contained more than one footprints of the same kind, that you wanted to change in a single step? Say, for example, that you have 5 TH LEDs, and you want to change them into SMD footprints?

In that case, you can use the bulk-change option that is available from the Edit menu in Pcbnew.

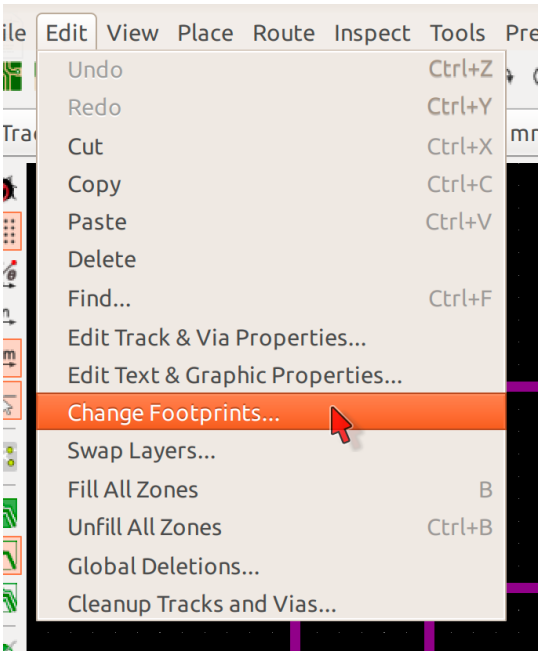


Figure 56.17: You can change multiple footprints via the Change Footprints menu item.

The dialog box that will come up allows you to choose multiple footprints that match various criterial, such as the footprint identifier or a reference. You can apply the new footprint to any existing footprints that match the criteria you have chosen.

57. How to import a 3D shape from Grabcad.com

The 3D viewer in Pcbnew provides an excellent way to visualise your board before you send it out for manufacturing. Out of the box, the 3D viewer will show you what your board will look like, and can render the PCB itself, the traces, copper fills and all pads and holes.

The 3D Viewer can also visualise the components that will eventually be attached on the board, so that you can have a complete picture of the final product.

KiCad ships with a comprehensive library of 3D shapes as a seperate download (because of its size). See recipe "55. How to install 3D shapes" for more information on how to do that.

However comprehensive this library may be, there will always components on your board that don't have a matching 3D shape available in KiCad's library.

In those cases, you have two options:

1. Look for a matching shape in a 3rd party provider.
2. Make a matching shape yourself.

In this recipe, you will learn how to find and import a 3D shape from one such provider, Grabcad.com. You can follow the same process to import shapes from other providers. In a separate recipe, you will also learn how to create your own shapes.

To demonstrate how to import a 3D shape, we'll work on the PCB from Project 1 of this book. Our starting point is the end of Project 1, when the final PCB looks like this:

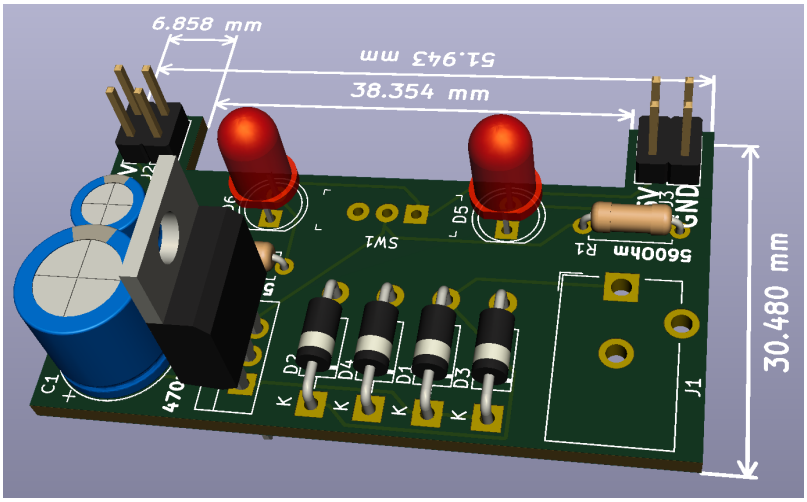


Figure 57.1: This is the current version of the example PCB; two 3D shapes are missing.

It already looks good, with most of the components having a match in the KiCad 3D shape library. But, two components don't. The barrel connector and the slide switch have no match.

There are many contributed 3D shapes and shape libraries available on the Internet. In this recipe, we'll use grabcad.com to find the missing shapes for this PCB.

Let's start with the barrel connector. Go to grabcad.com, and use the search bar to search for this shape. It may take a few attempts, because the naming of many components are not standardised or using. The barrel connector can be referred to as "DC power plug" or "DC power supply connector" or "EJ5035 connector", etc. After a few attempts, I found the exact shape I was looking for. You can also find it at https://grabcad.com/library/ej503b-connector-1.

Once you found the shape you need, look for the downloadable file with the ".stp" extension. An .stp file, pronounced "STEP", is a commonly use data exchange format for representing 3D objects. The 3D viewer in KiCad can work with this file format.

Figure 57.2: The .stp file contains the data for our 3D shape

Download the EJ503B-ND.stp file. I save it in a 3D shapes folder in my project directory, so it is easy to find later.

Next, start Pcbnew if not already started. Hover your mouse over the barrel connector footprint, and bring up the footprint properties window by

pressing the “E” key (or right click to get the context menu, and select Properties).

In the Properties window, click on the 3D Settings tab.

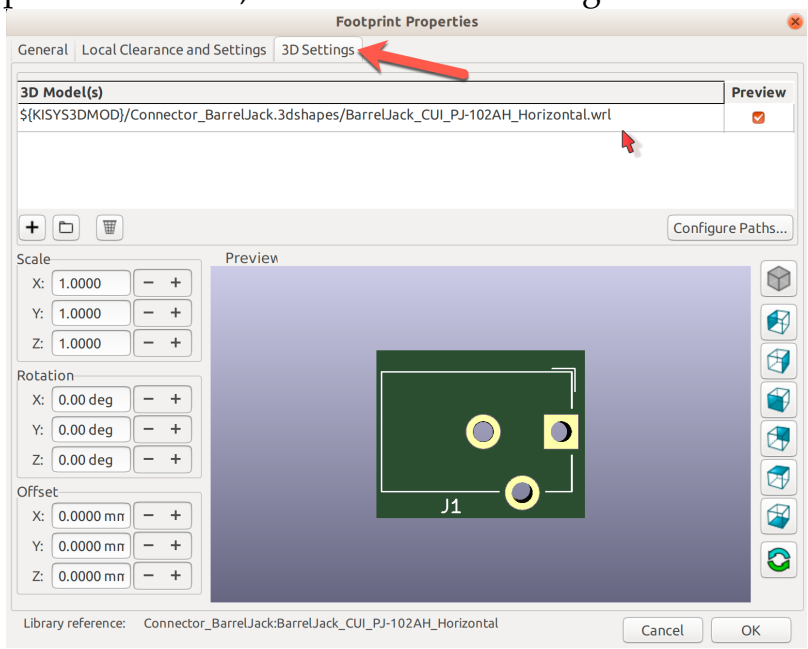


Figure 57.3: We’ll assign a 3D shape for this component in the 3D Settings tab.

Before we assign a new 3D shape, delete the existing one by selecting the only row in the 3D Models pane, and clicking on the rubbish bin button. Even though there is an assignment, this specific wrl file does not exist on my computer.

Let’s add the new 3D shape. Click on the folder button right under the 3D Models pane, and navigate your file system to the location where you saved the STP file earlier. Click on the STP file, and the viewer will show the 3D shape in the right pane.

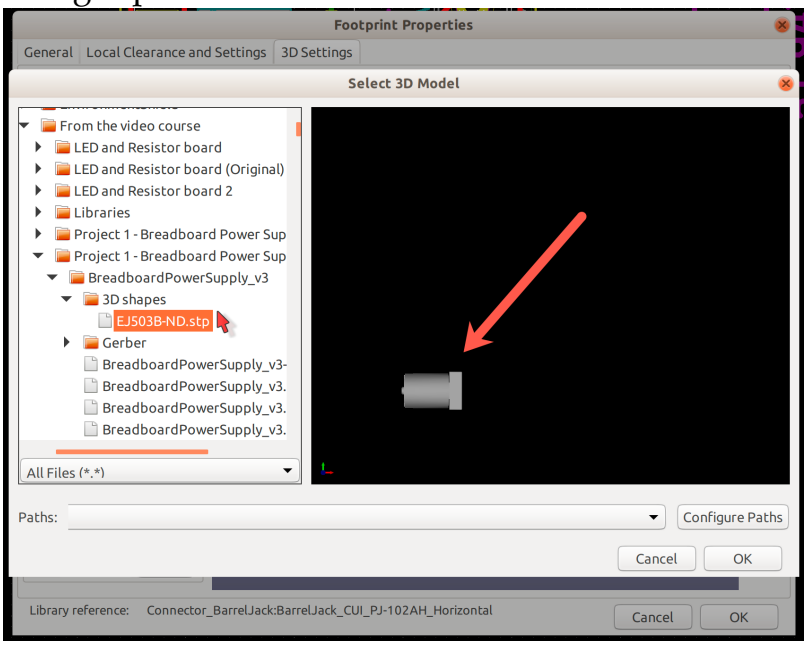


Figure 57.4: We'll select this 3D shape; its preview confirms this is the model I need.

Click OK to accept this selection, and you'll go back to the Footprint Properties window. As you can see, the 3D shape is not visible in the preview, but it is not in its correct place.

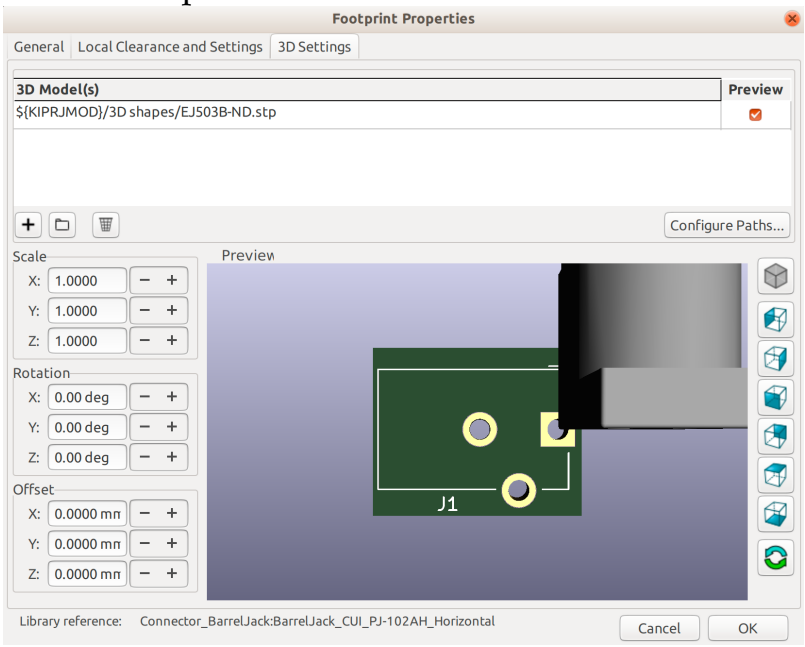


Figure 57.5: The new 3D shape is part of the model, but it is out of place.

Now, we must manually position the 3D shape over the PCB in the way we want it to appear in the final board rendering. Use the Scale, Rotation and Offset controls to do that. With a bit of effort, the final 3D model for this footprint will look like this:

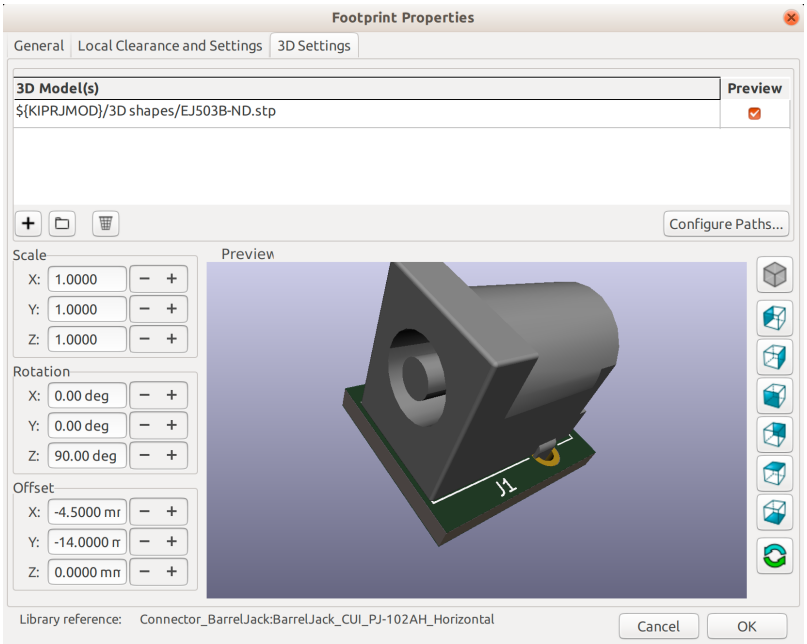


Figure 57.6: The new 3D shape is part of the model, but it is out of place.

Be sure to inspect the model from various angles to make sure everything is aligned. Also look at it from the bottom to ensure that the pins

and the holes are aligned. When the alignment is complete, click on “Ok” to commit the changes.

Let’s have a look at the 3D rendering of the full PCB. Bring up the 3D viewer. It should look like this:

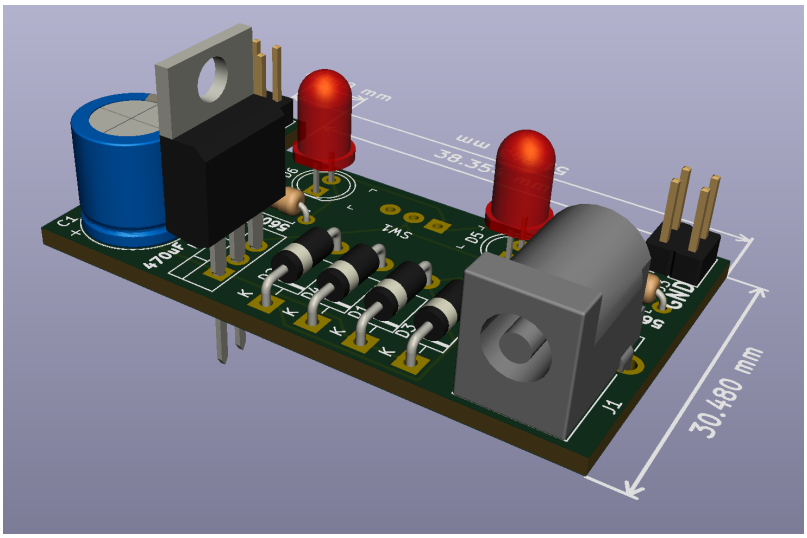


Figure 57.7: The barrel connector 3D shape appears in the PCB 3D viewer.

Much better, ins’t it?

There one more footprint without a 3D shape, the slide switch. You can find a shape for this footprint also in grabcad.com. Go to this page: https://grabcad.com/library/small-slide-switch-1, or search for “small slide switch”. Follow the exact same process to associate the slide switch footprint with this shape. To make the shape fit well, I also had to adjust the X-scale value.

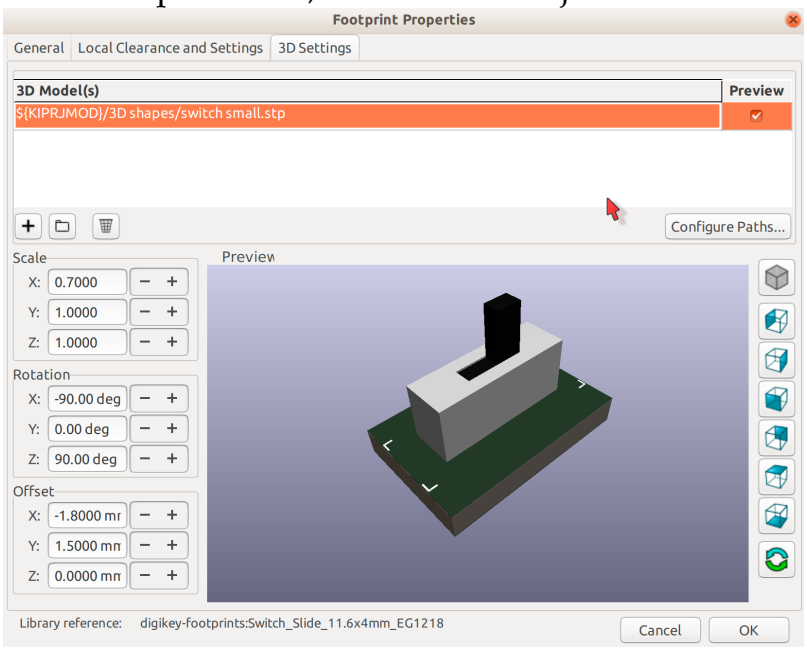


Figure 57.8: The slide switch 3D shape appears in the PCB 3D viewer.

Click OK to commit these changes, and bring up the 3D viewer for the PCB.

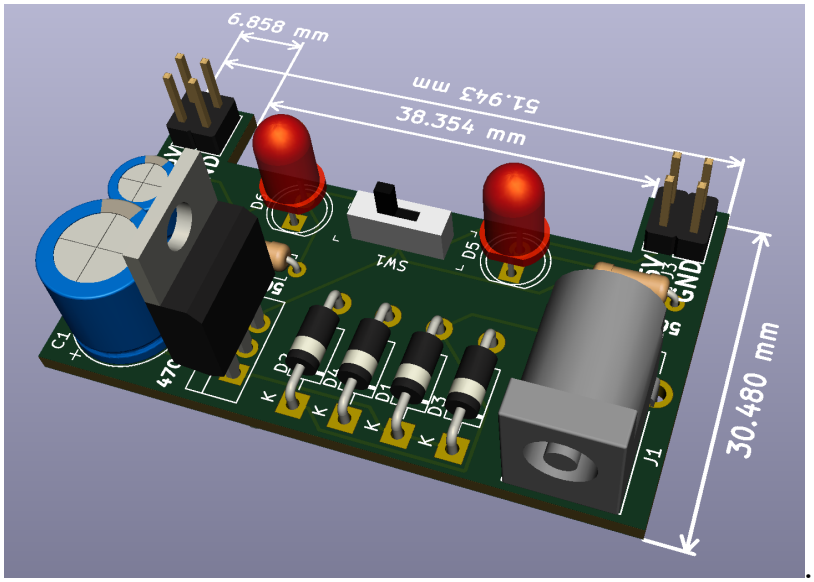


Figure 57.9: The completed PCB, rendered in the 3D viewer, with all component models.

This is much better. All of the footprints now have their 3D shapes appearing in the 3D rendering of the PCB. For most footprints, you should have no trouble finding an existing 3D shape that can fit your footprint perfectly after a small amount of tweaking its position and scale.

58. How to import symbols, footprints and 3D shapes from Snapeda.com

Snapeda (snapeda.com) is a popular repository of electronics designs. If a symbol, footprint or 3D shape exists, chances are that you will find it at Snapeda. Apart from the comprehensive designs library, Snapeda also supports many EDA tools other than KiCad.

In this recipe, you will learn how to use Snapeda to import a symbol, footprint and 3D shape for a component of one of the project boards. Specifically, you'll use Snapeda to quickly add a 2-pin screw terminal to the breadboard power supply that you designed in project 1 of this book. With this screw terminal, your power supply will be even more useful, allowing you to power circuits that are not on a breadboards.

Let's begin.

You can see the starting point iin Figure 58.1:

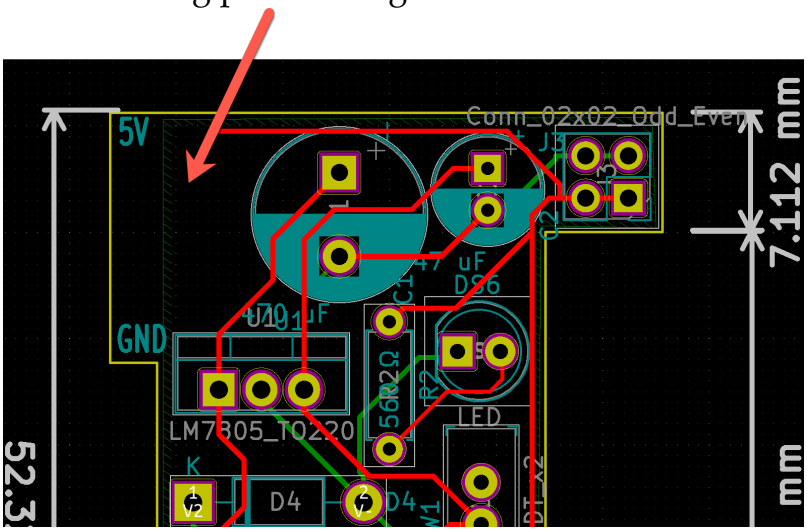


Figure 58.1: We'll start our work with Snapeda and enhancing this board here; let's add a screw terminal to the board.

This is the top section of the breadboard power supply from project 1 in the book. I have repositioned the two capacitors towards the right in order to make room on the left for the screw terminal. I have also changed the edge cut of the board to ensure that there is enough space on the PCB for the terminal.

Next, we need to find a symbol and footprint for the new component. Ideally, we'll be able to also find a matching 3D shape. Let's go to snapeda.com and do a search. Here is my complete search URL: https://

www.snapeda.com/search/?q=board+terminal+block+2+pin&search-type=parts

This search returns several hits, many of which look like exactly what I need. But one of them also has a 3D shape, as indicated by the box icon for item 1935161. So, I'll chose this specific device.

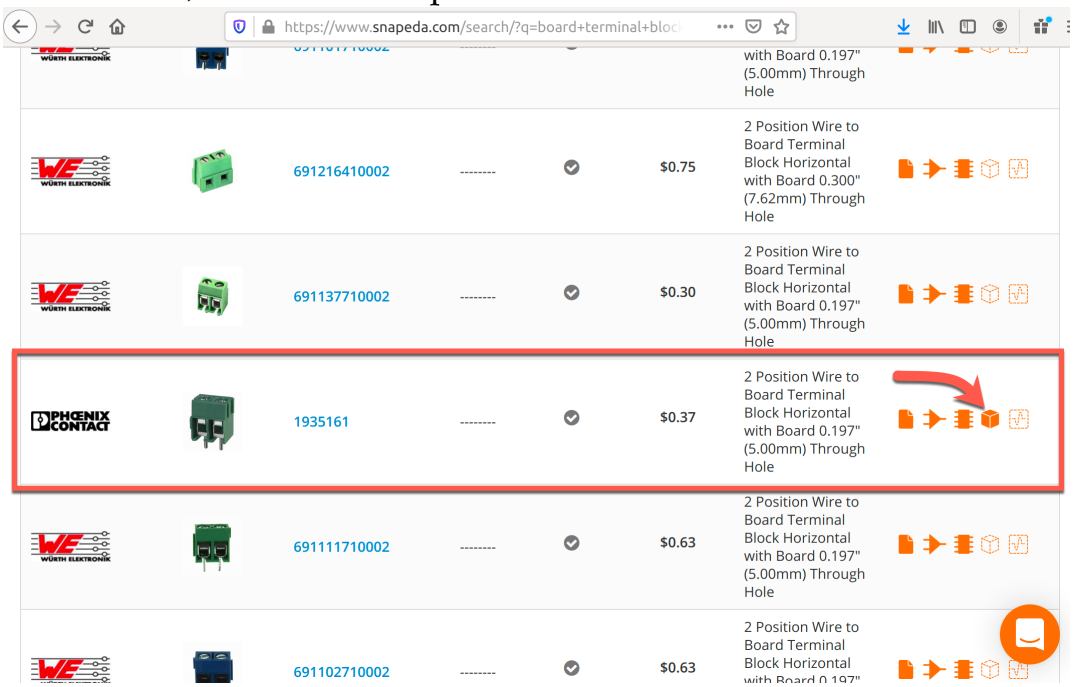


Figure 58.2: Several of these devices look perfect for my project, but the one in the box also has a 3D shape.

Click on the Phoenix Contact product row (in the red box) to access the device page. This page provides a preview of the symbol and footprint in one tab (2D Model), and the 3D shape in the second tab ("3D Model").

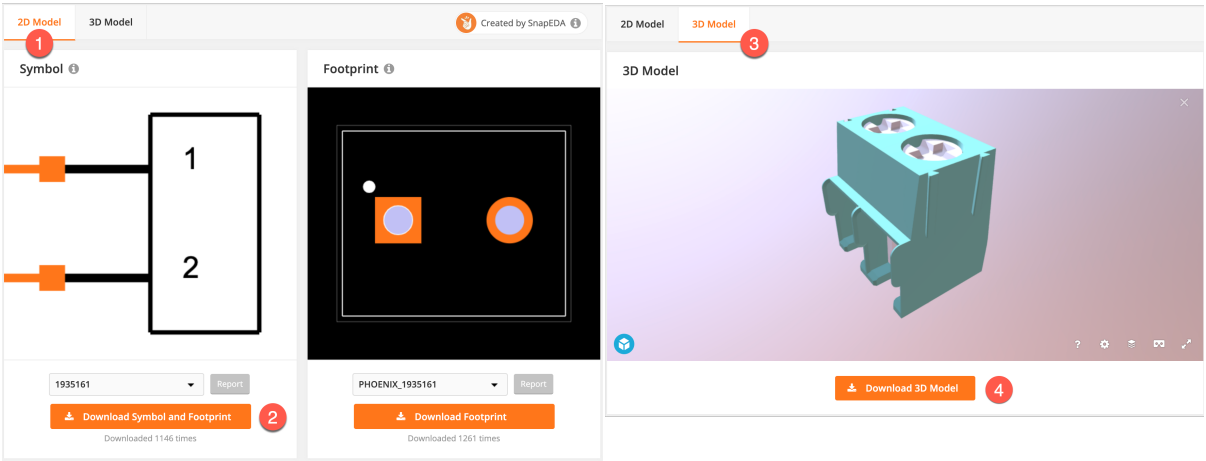


Figure 58.3: The device page provides previews for the symbol, footprint, and 3D shape, plus the download links.

To download the symbol and footprint, click on "Download Symbol and Footprint" button. This will download the two files in a ZIP archive. To download the 3D model, click on the "3D Model" tab, and then on the

"Download 3D Model" button. This will download the file in the STEP format (same as the file the you get from grabcad.com, as you can see in the relevant recipe.

Download these files, extract them from the ZIP archive, and save them in a convenient location. I have created three new directories in my project folder for "Symbols", "Footprints" and "3D Shapes", and save the files there.

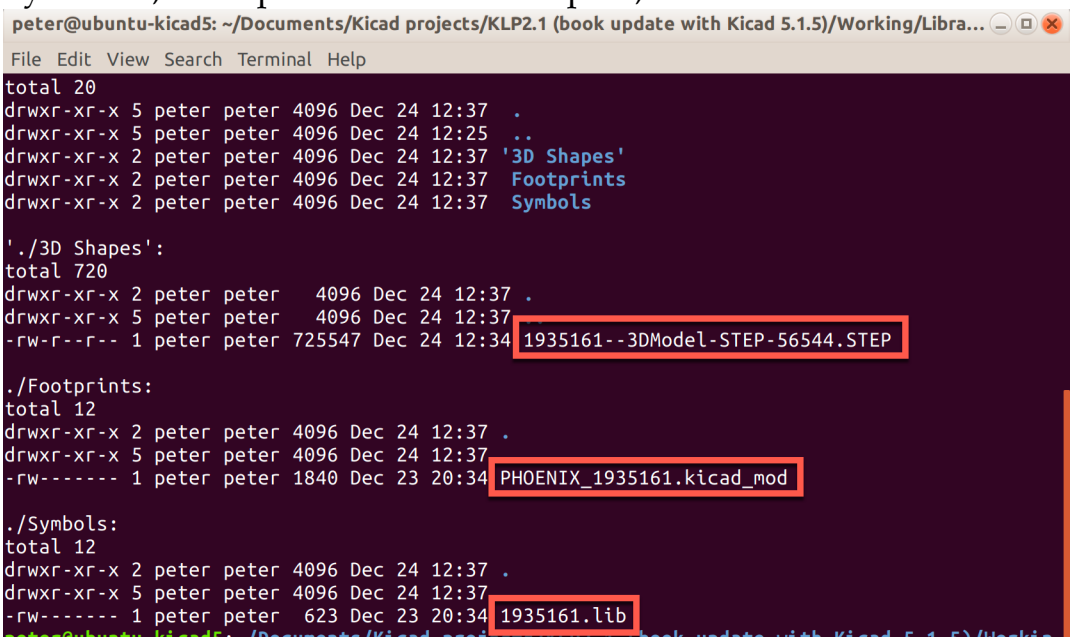


Figure 58.4: I saved the three device files in these directories: Symbols, Footprints, 3D Shapes.

This orgnization makes it easy to keep my imported device files tidy, especially in larger projects.

Next, let's make use of the new library files.

Start with Eescheema.

In Eeschema, click on Preferences, Manage Symbol Libraries to bring up the library manager.

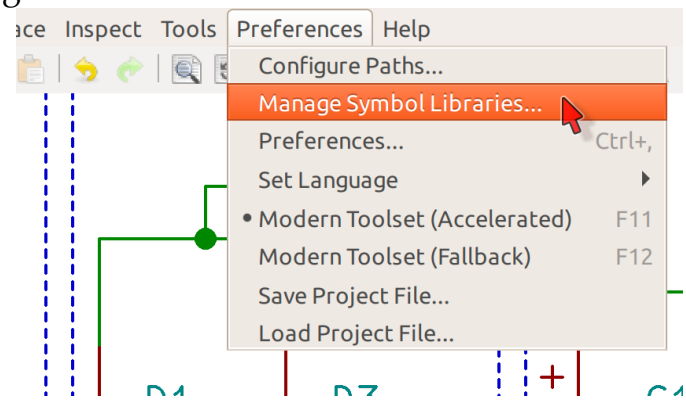


Figure 58.5: Bring up the library manager

In the Symbol Library Manager, click on the folder icon.

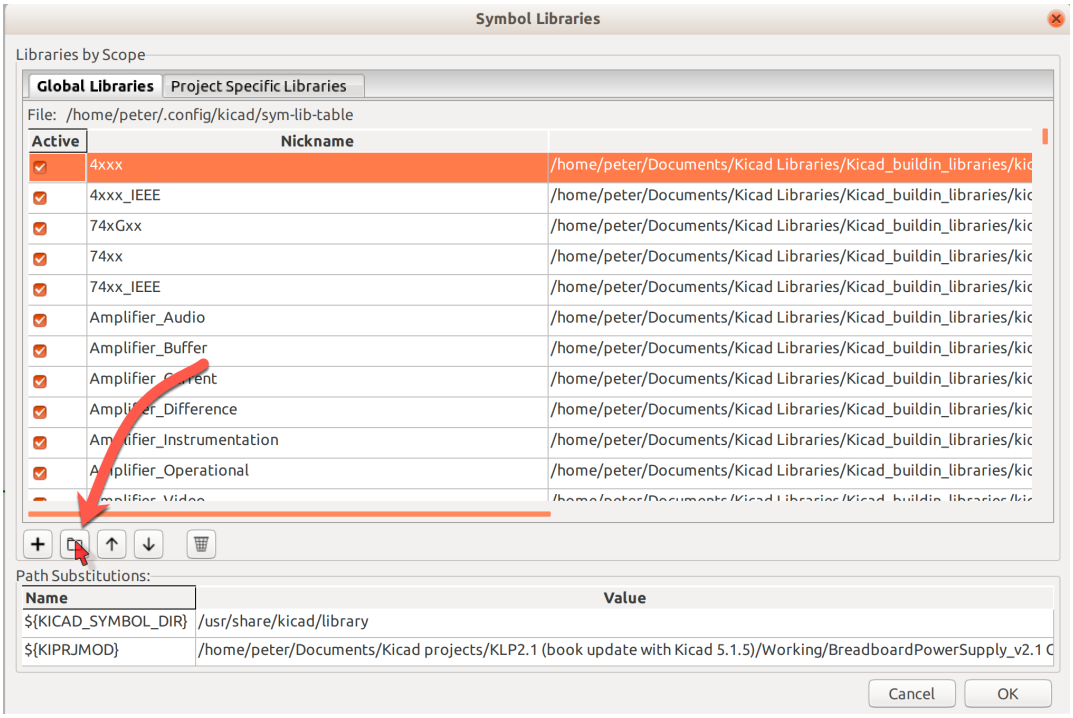


Figure 58.6: Click on the folder icon to reveal the file system browser.

This will bring up the file system browser. Go to the location of the lib file that you downloaded from Snapeda, select the file, and click Open.

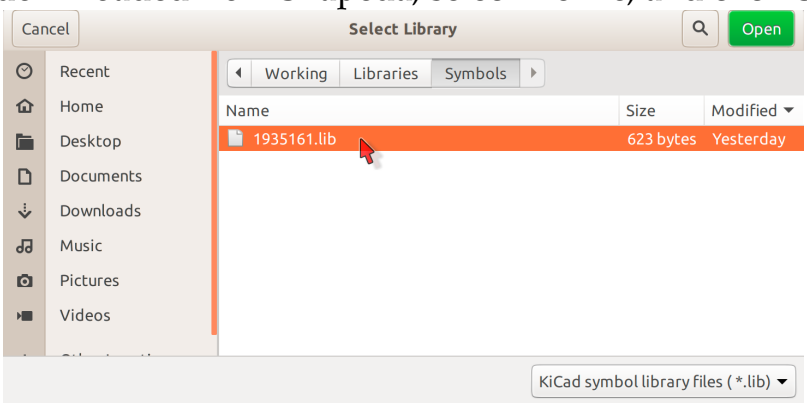


Figure 58.7: Find the lib file and click Open to import it.

The new symbol is added in the bottom of the Global Libraries list. To make it easier to find later, I recommend you change its nickname to something more sensible. White space is not allowed in nicknames. I have nicknamed mine to “Screw_terminal_2_pin”, as in Figure ……

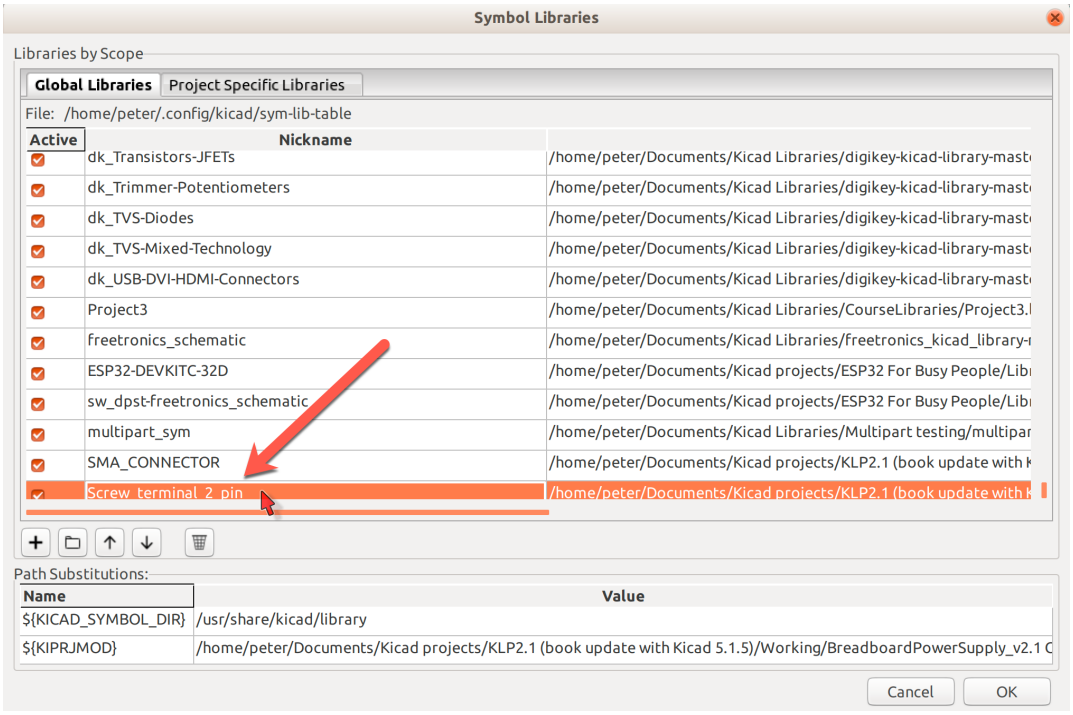


Figure 58.8: Provide a reasonable nickname for your new symbol.

Click OK to dismiss the Library Manager. The new symbol is now ready to use.

Let's place the new screw terminal symbol below the existing headers in the schematic. Place your mouse pointer blow the N7 block, and type "A" (for "add symbol"). The Symbol selector will appear.

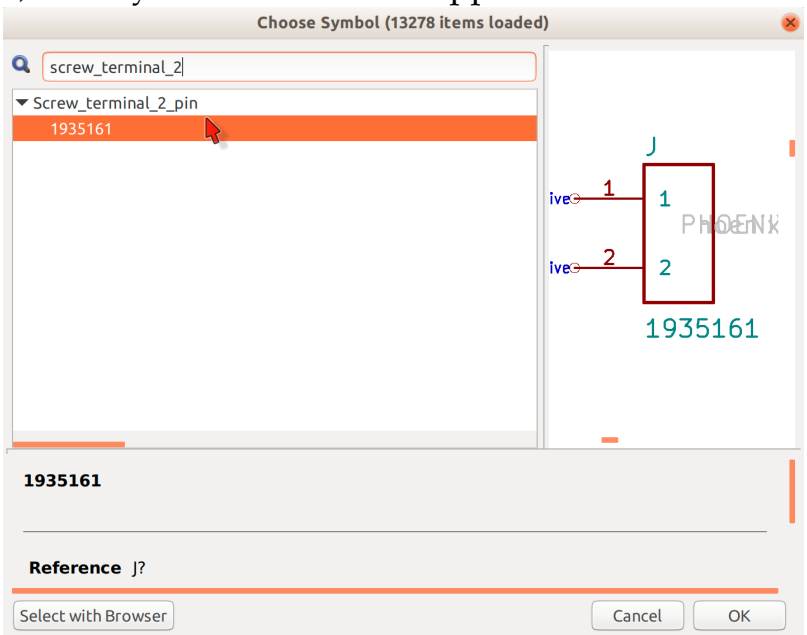


Figure 58.9: Type the first few letters of its nickname to find the new symbol.

The Symbol Chooser contains a listings of hundreds, or thousands of symbols. Instead of browsing, it is much more efficient to use the search box. Type in the first few letters of the symbol nickname to narrow down the listing, until you find the new symbol. Double-click on it to select it and drop

it in the sheet. Connect the two pins of the symbol to the V- and Vout2 nets. I use labels to make the final schematic easier to read.

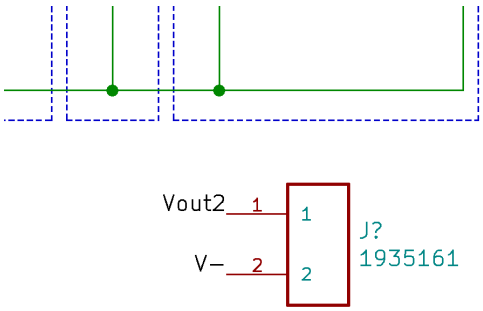


Figure 58.10: The new symbol is connected to the circuit via net labels.

Before you continue with Pcbnew, have a look at the new symbol properties window, and notice that the Footprint field is already populated. The symbol cam from Snapeda with the right footprint association, however we have not yet imported this footprint in Pcbnew. If you try to update the layout in Pcbnew with the updated schematic from Eeschema, you will get an error message because Pcbnew is not aware of the new footprint.

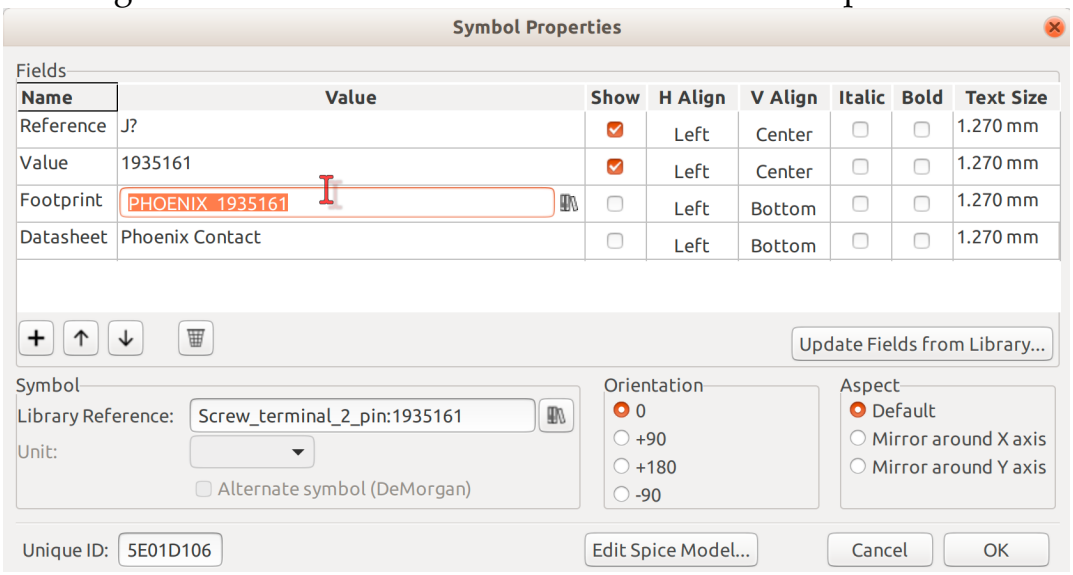


Figure 58.11: The symbol is already associated with its matching footprint, but Pcbnew is not aware of this footprint yet.

So, lets import the footprint.

Go to Pcbnew, and click on Preferences, Manage Footprint Libraries.

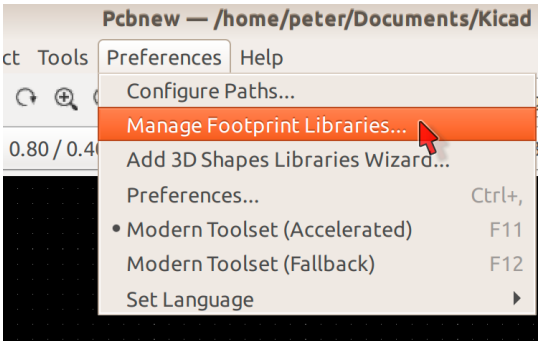


Figure 58.12: In Pcbnew, bring up the footprint library manager.

Under the Global Libraries tab, click on the folder button to bring up the file system browser.

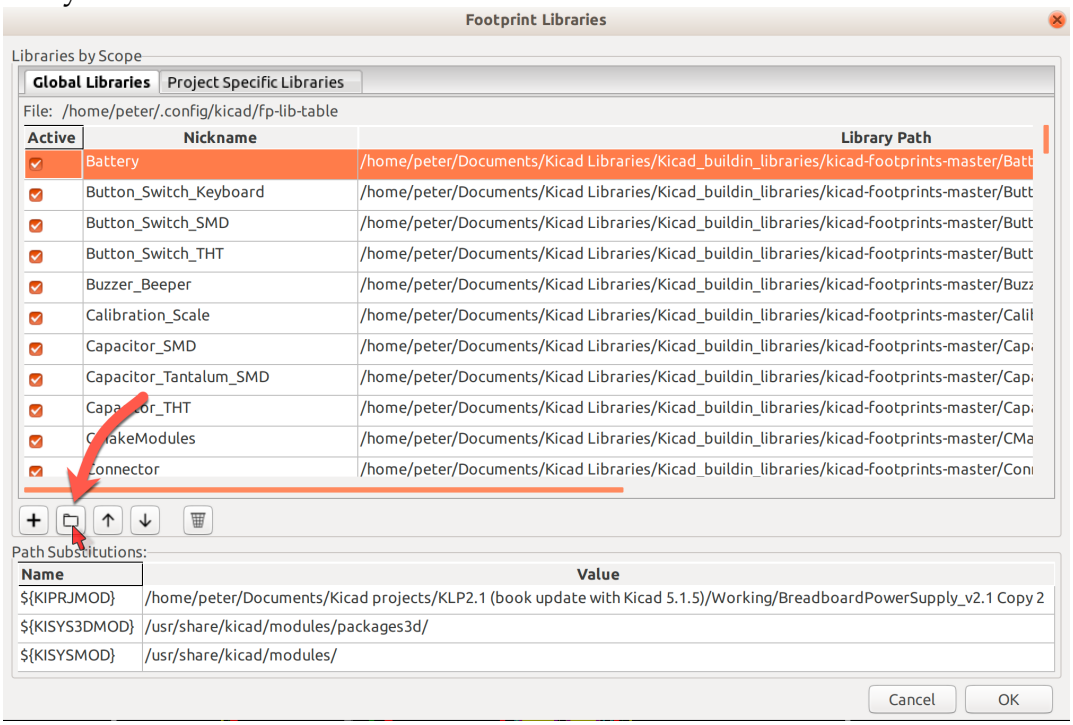


Figure 58.13: Click on the folder button to bring up the file system browser.

Select the folder in which you have saved the new footprint, and click OK. Unlike with the symbol, the footprint manager works at the folder level. Once you select the folder that contains your footprints, any new footprints you add to it will be usable in Pcbnew without having to add them to the Footprint library manager.

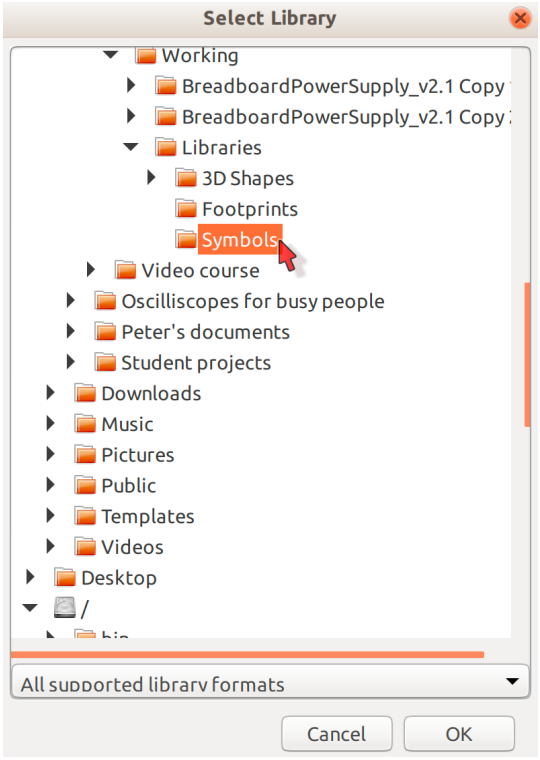


Figure 58.14: Select the Symbols folder and click OK to import all footprints contained within it.

Now you'll go back to the Footprint Library Manager, and you'll find the new footprint folder at the bottom of the list. I suggest you change the nickname to something reasonable, then click on OK.

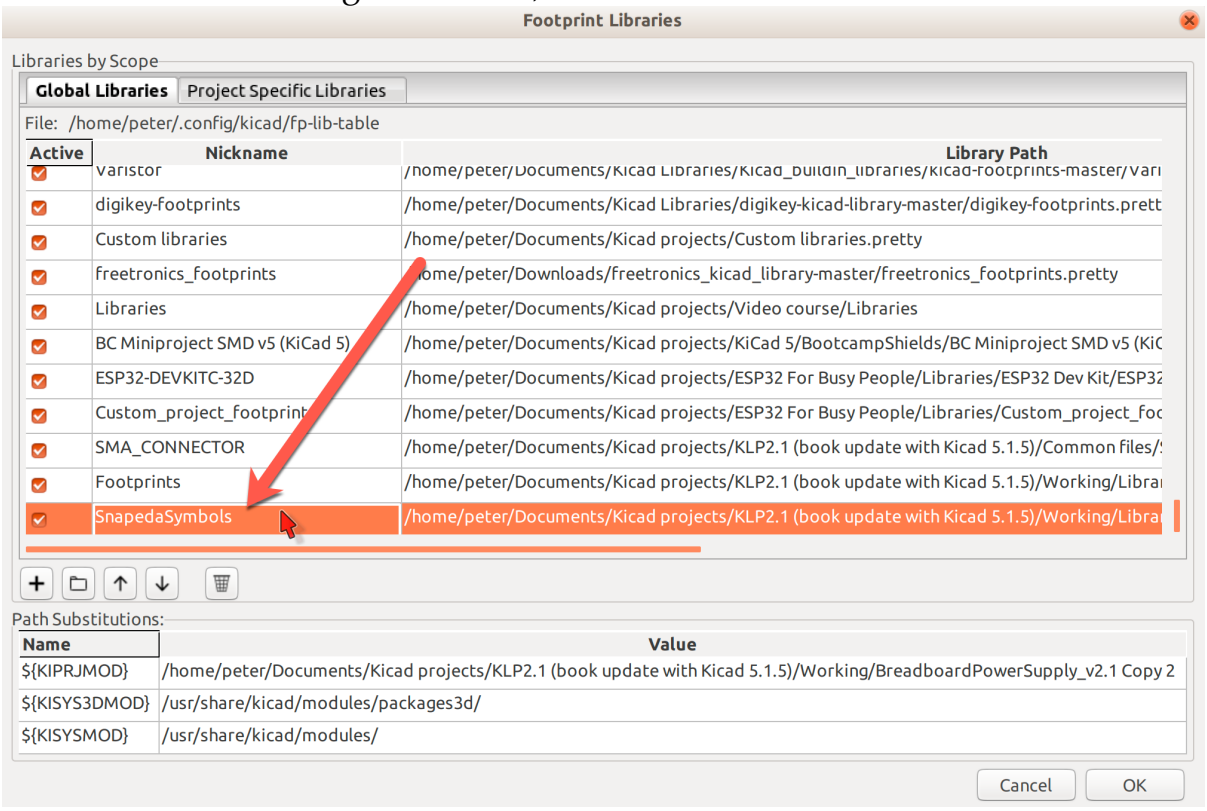


Figure 58.15: The new footprint is added to Pcbnew so we can now use it.

Ok, we are close to completing the addition of the screw terminal.

Ensure that you have saved your changes in Eeschema. In Pcbnew, click on the “Update PCB From Schematic” button.

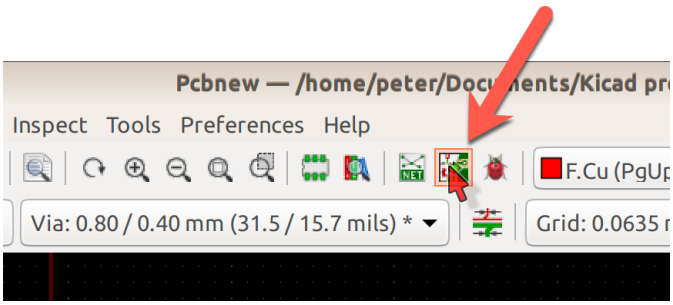


Figure 58.16: Import the changes of the schematic into the layout.

Kicad will detect that there is one symbol without annotation, so it will offer to annotate it for you. Click on Annotate.

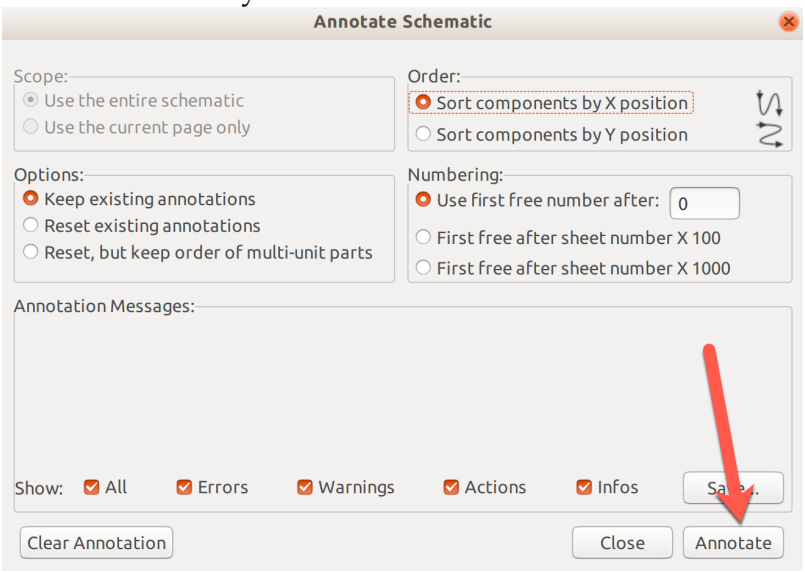


Figure 58.17: Kicad will annotate a symbol without a unique reference. This dialog will only come up if there is at least one symbol without annotation.

The annotation dialogue will be replaced by the PCB Update dialog. You can keep the defaults as they are, and click on the Update PCB button to complete the update.

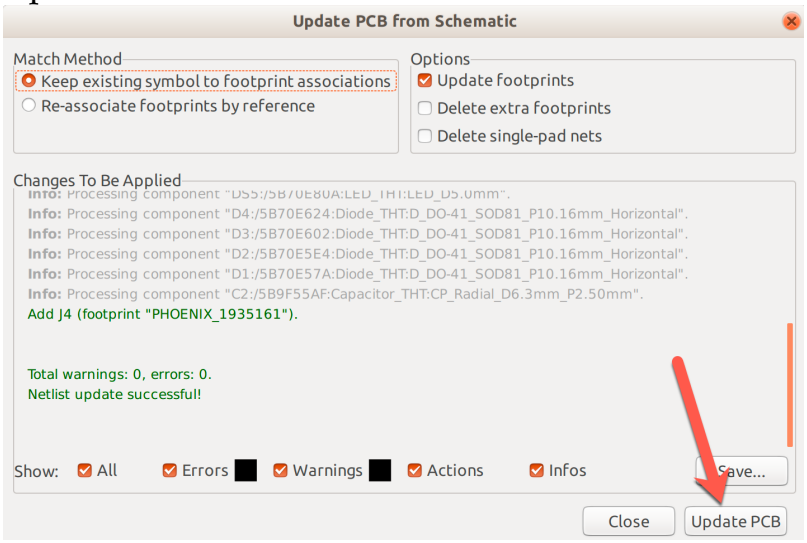


Figure 58.18: Click on the Update PCB button to complete the update process.

You will see a list of changes to the PCB layout in the Update window. We only have one change, the screw terminal footprint and symbol from Phoenix, so it all looks good. Click on Close to dismiss this dialog box.

Kicad will take you back to Pcbnew, where you can see the new footprint on the sheet, but out of place. Place the footprint at the correct location, and complete the wiring. The PCB should now look like the example in Figure 58.19.

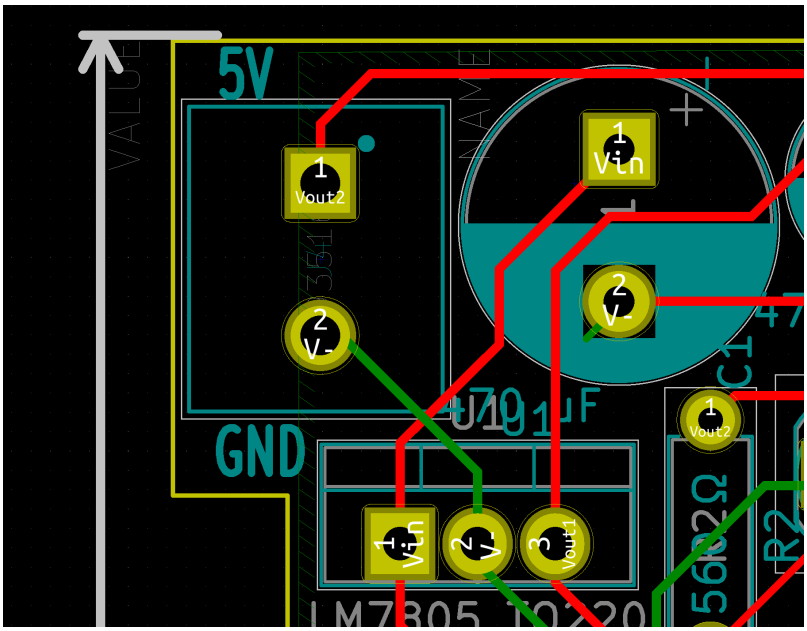


Figure 58.19: The new screw terminals placed on the PCB.

There is one more thing to do: assign a 3D shape to the footprint. Remember, while you have already downloaded the 3D shape for this device from Snapeda, you have not yet associated it with this footprint. The process to do this is identical to the process I document in the Grabcad recipe.

Let's go through it very quickly.

Bring up the properties for the footprint, and click on the 3D Settings tab. Browse to the location you saved the STEP file, and then place it nicely over the footprint. You can see my settings in Figure 58.20

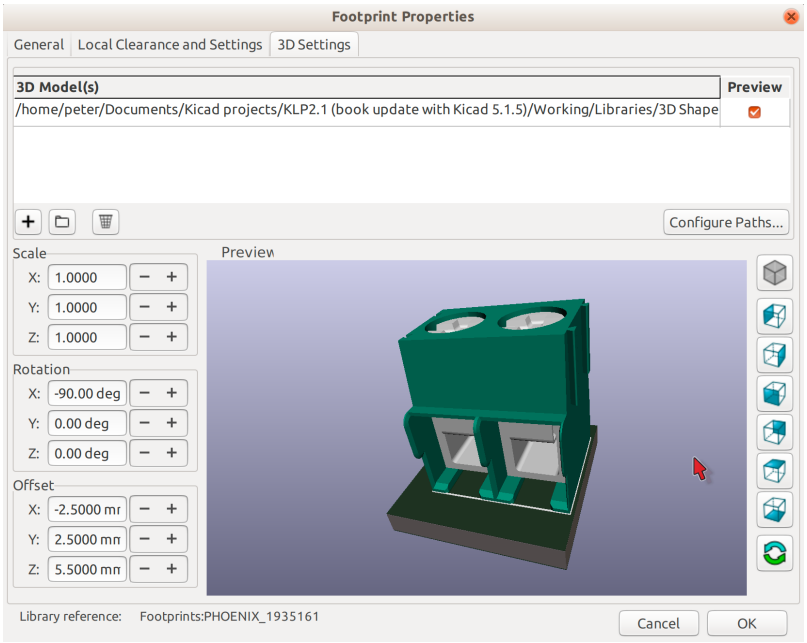


Figure 58.20: The position settings for the 3D shape of the new footprint.

Click OK to dismiss this window, and then bring up the 3D render of the PCB.

It should look like this:

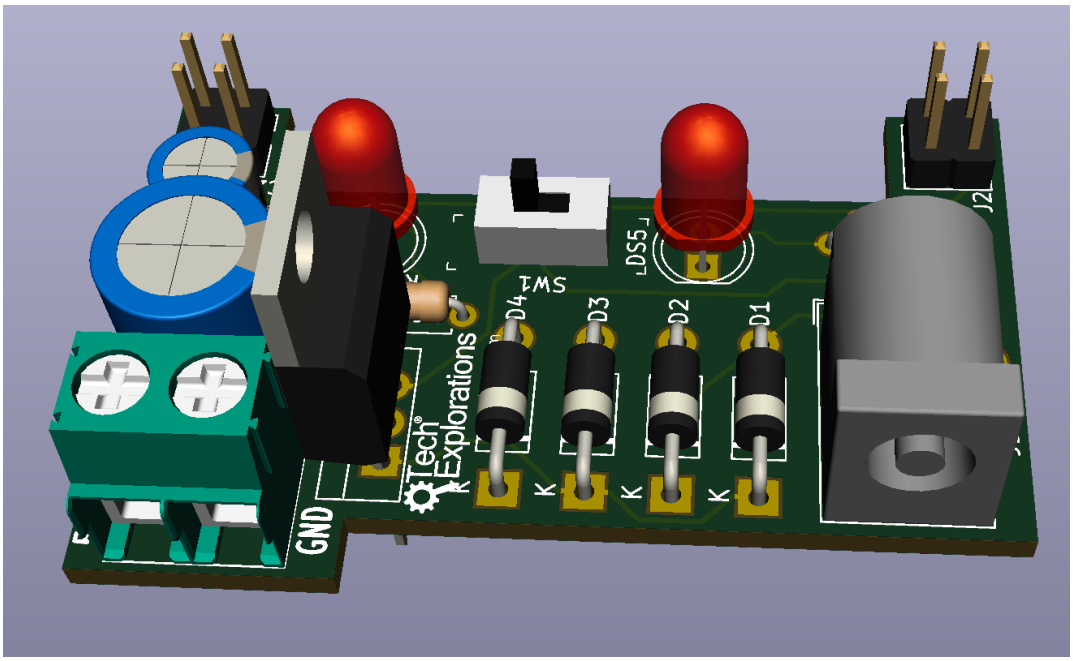


Figure 58.21: The final 3D render of the PCB, with the new device from Snapeda.

Isn't this neat?

59. How to change text and graphic properties in bulk (Pcbnew)

KiCad provides user interface tools to help you make multiple changes in a single step. For example, you can change all silkscreen layer text items to have a uniform height and width. You can also be specific in terms of scope, and change text attributes of all footprint values or reference text items at once.

Apart from text properties, you can also change graphics, track and via properties in bulk. With the help of special filters, you can be very specific. For example, you can change the width of all power tracks on the top copper layer to a new value.

The global change tool you will learn about in this recipe works in Pcbnew.

In this recipe, you will learn how to change all footprint values text item properties so that they are uniform. To do this, I will use the breadboard power supply project, which is Project 1 in this book.

To begin with, we have a PCB that looks like the one in Figure 59.1.

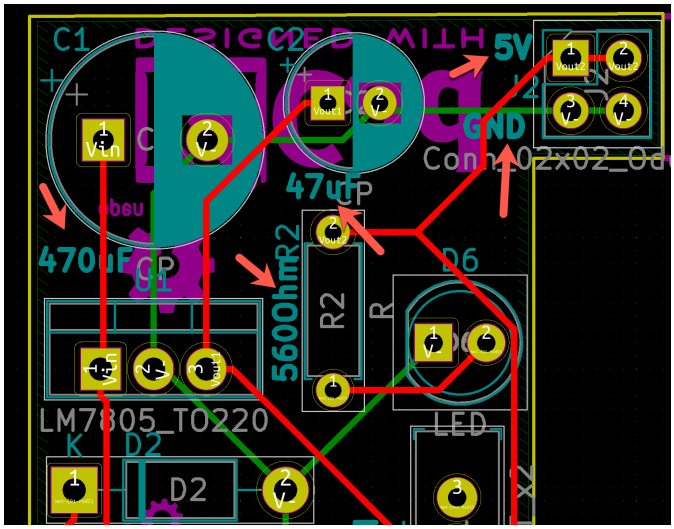


Figure 59.1: The starting point of this recipe: the arrows show the text items that I will modify their attributes in bulk.

I only show a segment of the full PCB to make it easier to distinguish the text items that I will be changing the attributes of. Those are indicated by the red arrows.

I would like to reduce the size (height and width) of those items, just slightly, to make it easier to fit in the available space on this PCB. What these

items have in common, is that they are independent PCB text items. I created them using the text tool in Pcbnew, and are not attached to a footprint. Of course, it is possible to make bulk changes to any other text item on the board, as long as you know what type of item it is (such as “footprint values” or “footprint reference” etc.) so that you can target them with the appropriate filter.

Let’s continue with the independent PCB text items.

Currently the size of each character in these items is 1mm x 1mm (width x height) with a thickness of 0.25mm. You can verify this by getting the item properties (hover you mouse over the item, and type “E”).

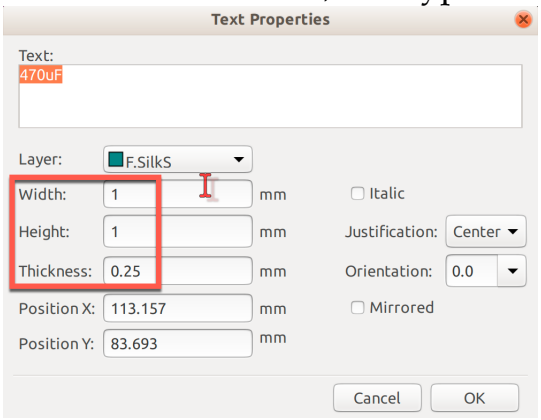


Figure 59.2: Currently, the text item character size is 1mm x 1mm.

Let’s change these dimensions to 0.8mm x 0.8mm, and the thickness to 0.2mm.

Click on Edit, Edit Text & Graphic Properties…

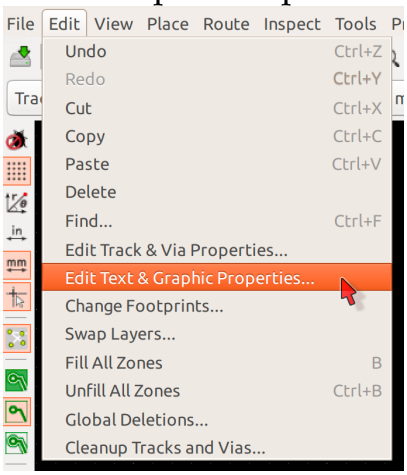


Figure 59.3: Bring up the Edit Text properties window

This will bring up the edit window, as in the screenshot in .

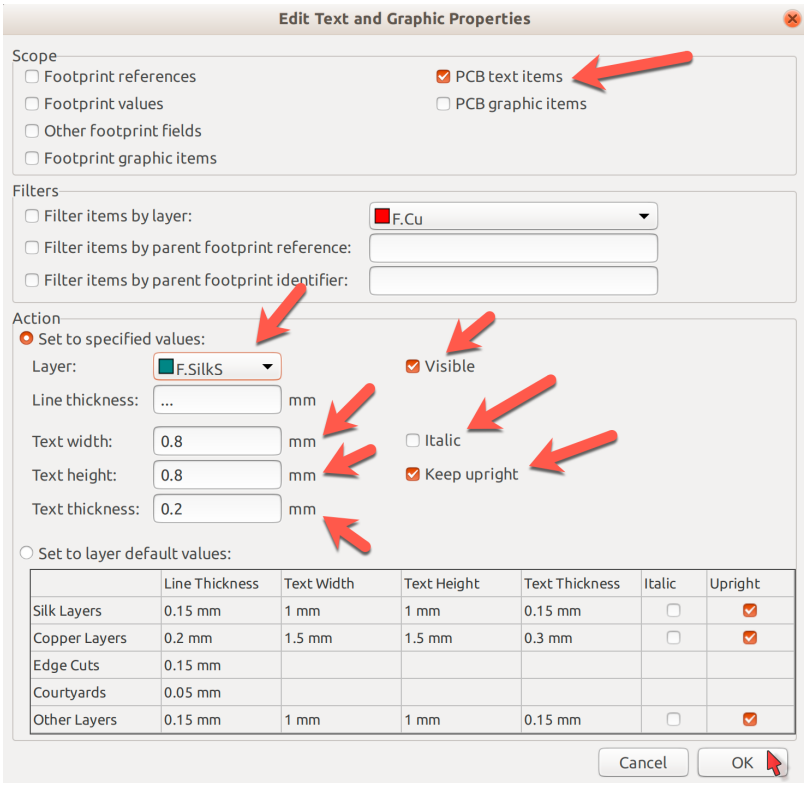


Figure 59.4: This edit window gives you a lot of flexibility to change text item properties.

This edit window gives you a lot of flexibility as to what is it that you want to change, and what change to introduce. Under "scope" and "Filters" you can select the kind of text or graphic item that you want to change. To keep this example simple, I have chosen to change independent PCB text items, and that is why I have clicked on this specific checkbox. Note that these are checkboxes, not radio buttons, so you can choose multiple kinds of items and have changes applied on all of them.

Under "Filters", you can choose text of graphics items based on the layer that they exist, or a parent footprint reference or identifier. So, for example, you can choose all text items that belong to the LED footprints, etc.

Once you have selected the kind of text or graphic item you want to change, go to the "Action" group to provide the details of the changes. In this example, again, lets keep it simple. I simply want to make the text smaller, so specify text width and height to 0.8mm. I also checked the "Visible" and "Keep upright" boxes, and unchecked "Italic". I can also set my chosen text items to preset values, listed in the "Set to layer default values" box. For any values that you want to keep unchanged, leave the "…" in the respective field.

Click OK to accept the changes.

The result is Figure …

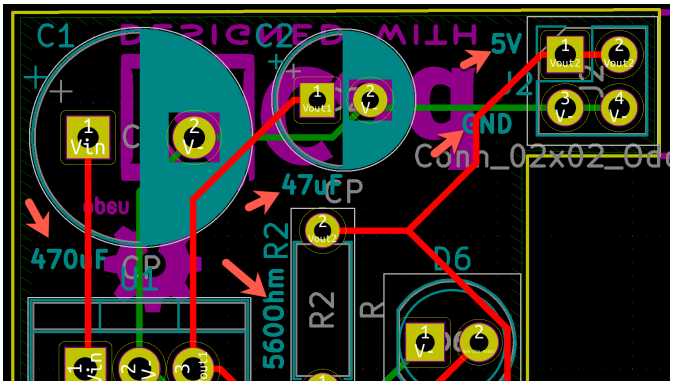


Figure 59.5: The text items marked with the red arrow have the new attributes applied.

Compare the “before” and “after” images of the PCB, and you will notice that the text items marked with the red arrow are slightly smaller then the originals. It only took a single interaction with a dialogue box to make this change, which affected multiple text items.

You can use the exact same technic to bulk-change the attributes of graphics (like lines and circles), but also those of vias and traces.

For example, try this little experiment. In this board, the power traces in the top copper layer is 0.3mm. Can you change the thickness of all top layer power traces to 0.35mm? (Hint: use the “Edit Text & Graphics Properties” window).

Part 6: Content by external authors

60. What is the meaning of the layers in Pcbnew and in the footprint editor? (by Rene Pöschl)

This chapter is sourced from the KiCad forum, and was written by Rene Pöschl.

You can find the original at: https://txplo.re/34c33

Accessed December 30, 2019, and included here with permission from the author.

A PCB is a 3-dimensional object with different materials stacked on top of each other.

If a designer says they want to design a 2 layer board, then they usually mean that this board will have two copper layers. A 4 layer board has 4 copper layers and so on. In addition to these copper layers there are other technical and documentation layers available.

A good introductory read is this post by @dchisholm : Any way to make the screen more understandable[23]?

In KiCad the following layers are available:

- Layers that have a front and back version start with F. (for front) and B. (for back).
- The **F.Cu** and **B.Cu** layers are the copper layers.
    - If there are additional copper layers they use the names In[number].Cu by default.
    - The names for copper layers can be changed by the designer.
- **F.Silk** and **B.Silk** define artwork on the silkscreen layers.
    - Typically this is the white artwork printed on the board.
    - If there is space then it typically has a body outline, polarity marker and reference designator.

[23] Available at https://txplo.re/ac64f.
Last accessed: December 20, 2019

    - Be aware of minimum clearances to exposed copper and minimum line width requirements. (Resulting in minimum text size requirements.)
- **F.Mask** and **B.Mask** define the area free of soldermask.
    - It is the negative of the resulting film that covers the board.
- **F.Paste** and **B.Paste** define the area that will be covered with solder paste (In datasheets often called stencil).
    - Used for reflow soldering of surface mounted devices.
- **Edge.cuts**: This layer is used to communicate with the manufacturer what the final board shape should look like.
    - The edge-cut cannot contain self intersections.
    - Polygons on the edge-cut must be continuous and closed.
    - It is allowed to have internal cutouts.
- **F.Adhes** and **B.Adhes** are layers to define adhesive (=glue) areas.
    - Only needed if components are on the bottom side during reflow soldering. (And even with components on the bottom it is not always needed. Check with your manufacturer if you need to define it for your board.)

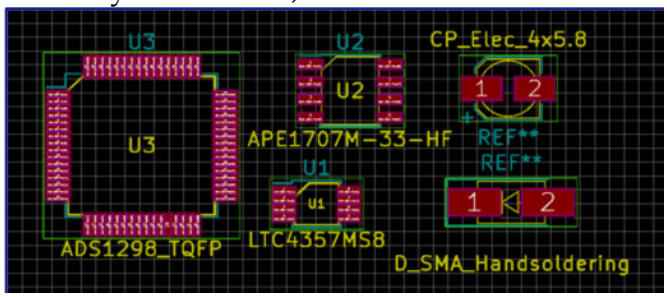
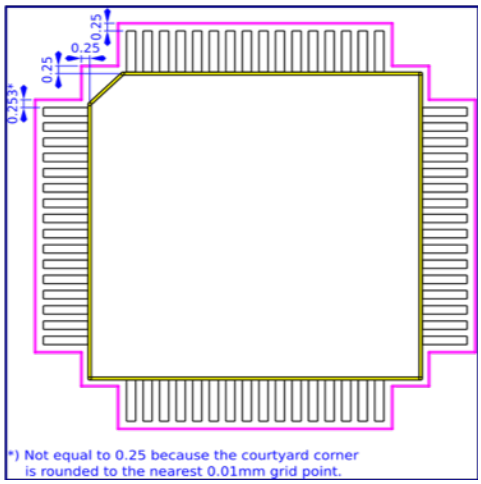


Figure 60.1: Example courtyard clearances for different packages.[24]



[24] The original of this image exists at https://txplo.re/32bd4.
Last accessed: December 30, 2019.

Figure 60.2: Example courtyard clearance for a QFP-64 package.[25]

- **F.CrtYd** and **B.CrtYd** are used to define a courtyard area.
  - The courtyard defines where no other component should be placed.
    - The size of this area depends on your manufacturing capabilities.
    - It also depends on your needs. (If you want the possibility to rework the pcb, you might need a larger area compared to when you do not plan to do this.)
    - The rules used in the official library are defined in the KLC RULE F5.3 and are closely aligned to industry standards.
  - This layer is checked for violations since version 5 of KiCad.
    - KiCad requires every drawing on this layer to represent a closed polygon (drawn with the normal line drawing tool.)
    - Internal cutouts are allowed.
- **F.Fab** and **B.Fab** are documentation layers.
  - These are intended to be used for communicating with board assembly houses and for user documentation.
  - It typically has the outline of the part body (nominal or maximum dimensions, depends on your particular needs.)
  - It is common to have a polarity marker on this layer.
  - Reference and value fields are often found on this layer. (This might be the only place where these are found for high density boards as there is not enough space for silkscreen text. Fab text does not come with the same minimum size requirements as silk.)
- **Dwgs.User** and **Cmts.User** are used for user drawings and comments.
  - In the official lib this layer is used to communicate with the user of footprints. (Example to tell them where to place keep-out areas as they are not directly supported by KiCad in footprints)
- **Eco1 and Eco2** are layers with no specific defined purpose. (The user can use them for whatever they want. They are not used in footprints supplied by the official lib.)
- **Margin layer**: Is there to define a margin relative to the edge cut.
  - There is no DRC check to verify that no copper feature violates the margin definition.

[25] The original of this image exists at https://txplo.re/32bd4.
Last accessed: December 30, 2019.

Limitations of the footprint editor

SMD pads in the footprint editor can only be placed on a top or bottom layer.

THT pads typically use *.Cu to communicate that these interact with all copper layers.

The use of layers that have no defined bottom or top layers is undefined. (Dwgs, Cmts, Eco1/2 layers).

For more information, see this wishlist bug: https://bugs.launchpad.net/kicad/+bug/1746279. Please note that this bug is fixed in KiCad 5.1.

The use of the edge cut layer in footprints can result in problems. As there is no snapping for the drawing tool other then to the grid this might result in polygons that are not closed.

This layer is unsupported by the footprint editor (KiCad 5 and earlier). There is no way to define drawings on that layer from within it. KiCad 4 even moved drawings on that layer to another layer when opening it in the footprint editor. This non feature has luckily been removed from version 5.

For more details about edge cut support, see this wishlist bug: https://bugs.launchpad.net/kicad/+bug/1251393. Please note that this bug is marked to be fixed in KiCad 6.0.0-rc1.

61. Power pins in multi unit symbols (Marc Nijdam)

This chapter is contributed by Marc Nijdam (marc@auditeon.com). It is provided here with only grammatical and syntactical improvements.

In Eeschema multi-unit symbols are symbols which consist of multiple identical units which have common power supply pins. Despite the fact that the KiCad symbol library,[26] included along with the KiCad installation, still contains multiple solutions for drawing power pins, according to the KiCad Library Convention[27] (KLC), the recommended way is to draw a separate power unit.[28]

KiCad symbol library status quo

As of May 2019, the following implementations for power pins in the KiCad symbol library can be found:

[26] As stated on the following page (http://kicad-pcb.org/libraries/download/), the official KiCad libraries are available at https://kicad.github.io (Pages visited at 26th of May 2019)

[27] The KiCad Library Convention (Revision 3.0.18, from 2019-02-15) can be found here: http://kicad-pcb.org/libraries/klc/ ((Page visited at 26th of May 2019)

[28] This is explained here: http://kicad-pcb.org/libraries/klc/S3.8/ (Page visited at 26th of May 2019)

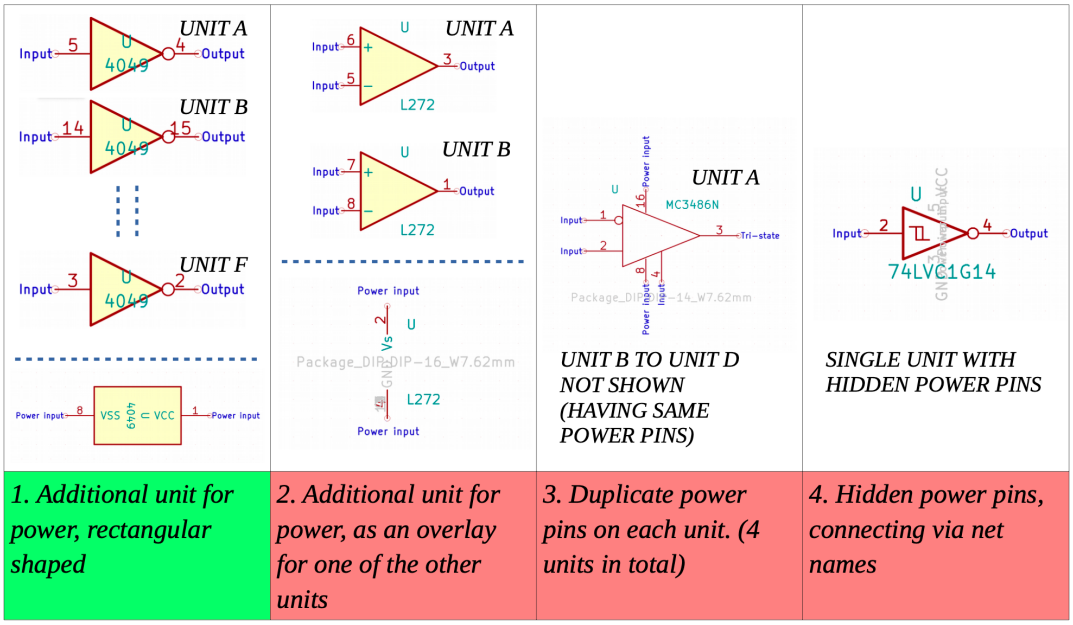


Figure 61.1: KiCad - Implementations for power pins

Maybe a fifth method is also present, where one of the units has power pins and the others haven't. This is problematic for swapping gates. Also no example could be found. Maybe it is not in use in the official library.

The first implementation uses one additional unit consisting of power pins connected to a block. This is the way which can be found in the KLC. The power pins as shown in implementation 2 could be placed on top of one of the other units. Implementation 3 has power pins on all of the units. The rationale with implementation 4 (hiding power pins completely) was to prevent clutter with logical gates. Hidden power pins are then implicitly connected to net names "VCC" and "GND".

Creating a new symbol

The KiCad Symbol Editor is used to create new schematic symbol. Now before we continue and start delving into creating a new symbol, we need to be really sure that this is really necessary. Often the symbol alias feature,[29] may offer a much more sound alternative for our goal. Specifically if the new symbol has an exact similar pinout of another already existing symbol. Then, instead of creating a new symbol, we just create an alias: In the alias tab of the library symbol properties we add the differences of the new part (These are: Name, datasheet, description and keywords). Many symbols in the existing

[29] See here http://docs.kicad-pcb.org/5.1.4/en/eeschema/eeschema.html (at "Symbol Library Overview") and here: https://forum.kicad.info/t/what-does-alias-do-in-edit-component-properties-in-the-part-library-editor/8279/2 (last visited on 15-09-2019)

libraries are defined this way. For example, the 74LS00 has multiple aliases like the 7400, 74HC00, 74HCT00 and 74LS37. Another example is the TLC272 op amp, defined as an alias of the LM2904. Aliases greatly simplify libraries with a lot of duplicate symbols with only small differences.

One good reason though to create a new symbol from scratch, is that that there is no write access to KiCad libraries[30] which are included along with the KiCad installation, not even for creating an alias. If so, creating our own symbol in a library where we can write our symbols and aliases into is a good workaround.

When creating a new symbol, it needs to be placed inside a symbol library, a placeholder for one or more symbols. When no symbol library is selected, it will ask for a library in which the new symbol should be placed. Alternatively a new library can be created via Menu File → New Library... . Also good to know, deleting a library can be done via Menu Preferences → Manage symbol libraries... . Please note that for deleting a library, the symbol editor may need to be restarted in order to become effective.

Getting a library first

When creating a new library, the symbol editor will ask if the library is available to the project only or globally to all projects.

If the library only contains your personal collection of parts and is available for your project only, you could save the library in a subdirectory of your project and use for example the following arbitrary name:

```
Custom_Various.lib
```

Observe that the library is created accordingly:

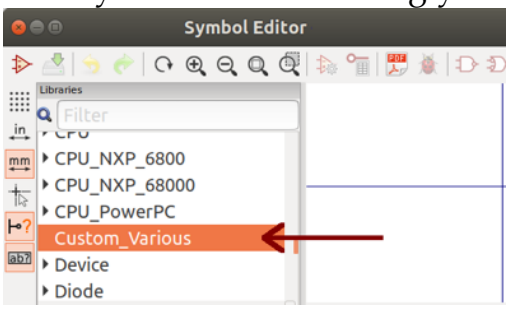


Figure 61.2: Create New Library in KiCad Symbol Editor

The library is empty and symbols can be added to it.

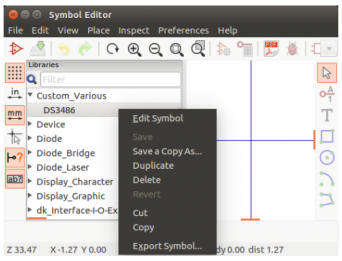
[30] Per default these (kicad-footprints, kicad-libraries, kicad-packages3d, kicad-symbols and kicad-templates) are installed as user root in: /usr/share/kicad/library/

It is also possible to add a symbol to an existing library as long as the library did not come along with the KiCad installation itself.

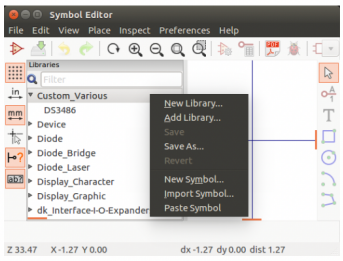
For correct naming of symbols and libraries, KiCad has conventions which may help to keep names consistent[31].

Adding a new symbol to a library

Symbols can be added to a library by creating them from scratch, copying over or by importing. Depending on where you right click, you can edit/copy/export/duplicate symbol, create a new library, add a library or create/import/paste a new symbol:



Right clicking on a symbol



Right clicking on a library

Figure 61.3: Context options: Symbol vs. Library

Often copying over a symbol from another library is the fastest solution. For adding, right click on the library of your choice and select “New Symbol...” . For copying a symbol from another library, right click on the symbol and select “Copy”. Then right click on your library of choice and choose “Paste Symbol”.

After copying, it may be necessary to change the number of units. Use the Library Symbol Properties to change:

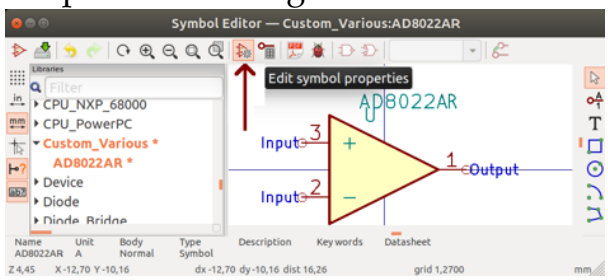


Figure 61.4: Library Symbol Properties

Properties can be alternatively found via Menu -> Edit -> Properties... The Library Symbol Properties window looks like the following:

[31] See http://www.kicad-pcb.org/libraries/klc/#G1.1 (last visited on 20-09-2019)

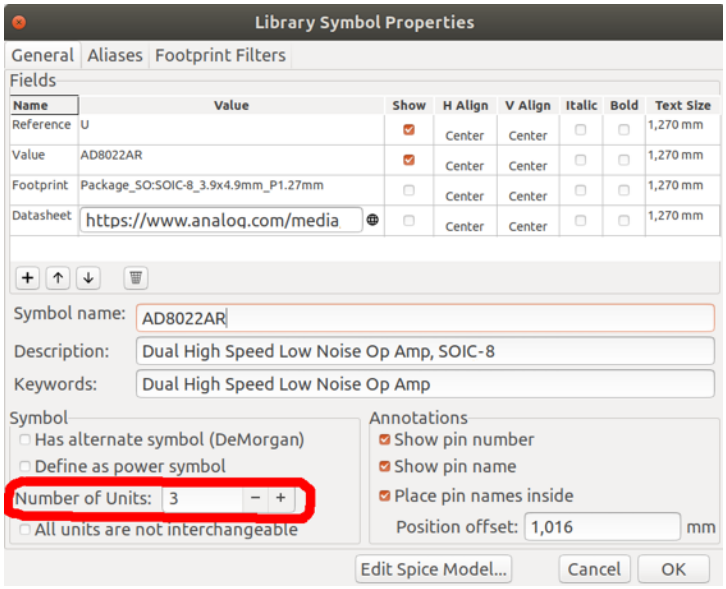


Figure 61.5: Number of units

Add one unit for the dedicated power supply: If there are 2 identical units, the number of units will then be 3.

For editing, the following two options may be very confusing at first and should be verified at all times:

Synchronized pin edit mode

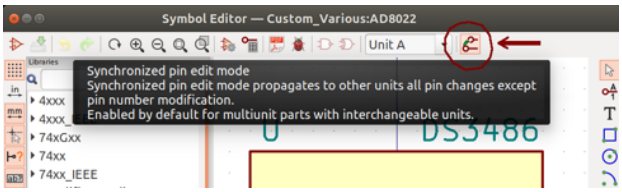


Figure 61.6: : Synchronized pin edit mode

If selected, both additions and deletions are propagated to the other units. Make sure that this button is not selected if specific editing is done on only one of the units.

Line options common to all units in component

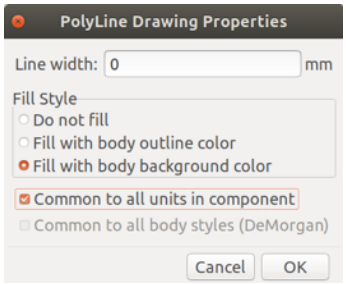


Figure 61.7: Common to all units in component

When drawing a line on a specific unit, make sure this setting is disabled. Otherwise it may propagate to other units as well and even worse, remove the drawing from the other units.

The last unit is usually dedicated for power only. For drawing this symbol, we draw a rectangle, using the "Add graphic rectangle to symbol body" drawing tool.

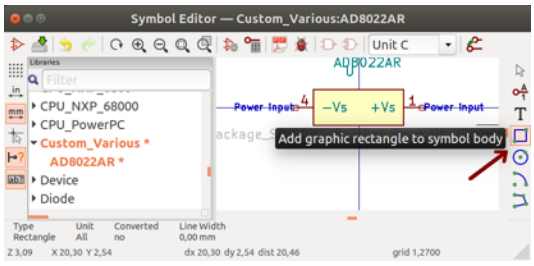


Figure 61.8: Function "Add graphic rectangle to symbol

After drawing the rectangle, fill it with the background color by right clicking somewhere on the line and selecting the "Edit Rectangle Options..." context menu.

Examples

For more practical understanding of creating multi unit symbols, two examples are given. The first example is dedicated to the Analog Devices AD8022AR[32] (dual op amp) and the second example to the Texas Instruments DS3486[33] (Line receiver, similar to the MC3486N). In the first example we create the part from scratch. In the second example we copy from an existing library and change the newly created symbol: From a multi unit symbol with four seemingly identical comperators into two pairs of similar units which share a common control input.

Analog Devices AD8022AR

The functional block diagram of the AD8022AR can be found in the datasheet[34] of this component:

[32] https://www.analog.com/media/en/technical-documentation/data-sheets/AD8022.pdf (last visited on 09-09-2019)

[33] https://www.ti.com/lit/ds/symlink/ds3486.pdf (last visited on 09-09-2019)

[34] Per default these (kicad-footprints, kicad-libraries, kicad-packages3d, kicad-symbols and kicad-templates) are installed as user root in: /usr/share/kicad/library/

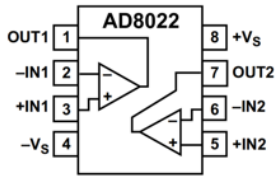


Figure 61.9: Functional Block Diagram

The component contains two similar units and a positive and negative power supply. For KiCad this means we need to create a multi unit symbol consisting of in total three units.

From the main window we go straight into the symbol editor, create a new library and give it a useful name. When clicking on save, a dialog will appear, asking for: “Choose the Library Table to add the library to:”. We can choose between Project or Global. Choose whatever serve your needs.

Then right click on the newly created library and choose the option New Symbol...

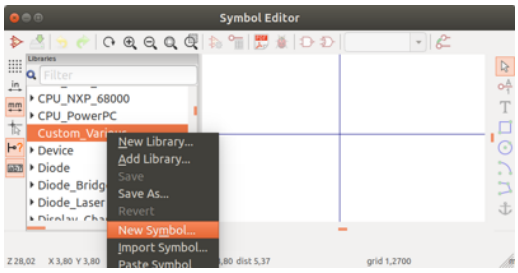


Figure 61.10: Create a New Symbol...

Now the Symbol Properties window appears. Enter a symbol name, as it appears in the datasheet and change the number of units per package. In our case the number of units per package is 3.

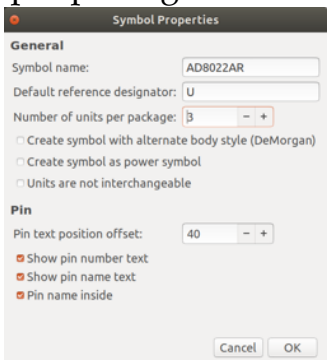


Figure 61.11: Symbol properties

After clicking on OK, we should see now the following in our symbol editor:

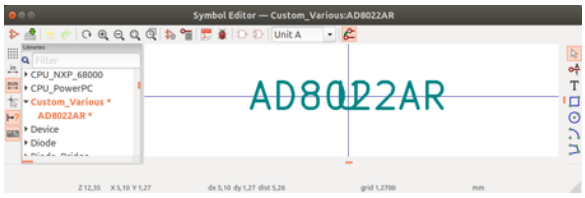


Figure 61.12: Symbol Reference and Value

Reference (U) and value (AD8022AR) are in their default position shown as overlapping text.

It is expected at this point that the user has some experience to use the regular drawing elements for housing, pin number, pin name and fill color. We kept the synchronized pin mode on during the adding of the pins to help at least the positioning to the same place at the other units. When done it should look like the drawing in Figure 61.13.

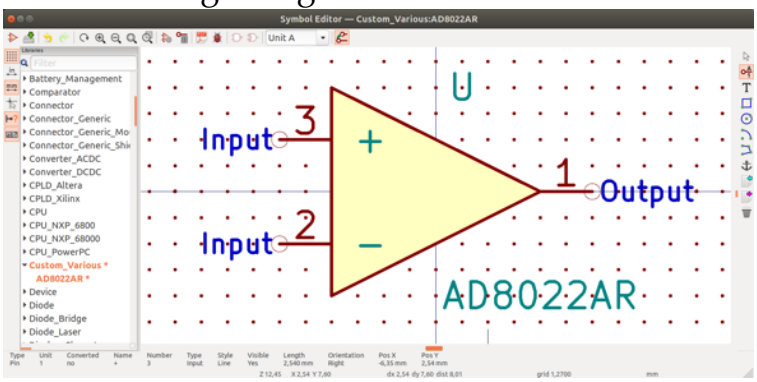


Figure 61.13: Symbol editor

Please note that when adding pins, KiCad automatically assigns a different name for the same pins at the other units to prevent duplicate names. Pin 1 on unit B is for example called 1-UB and on unit C, 1-UC. This is different to what the datasheet shows and needs to be corrected. To prevent further changes, first switch off the synchronized pin mode. When done, draw on unit A a triangular shape with background fill like in the illustration above. Per default this triangle will be also automatically duplicated to the other units. However this mechanism is not very intuitive as explained further below. Therefore on unit A, if not done already, edit the line options and make sure to disable the “Common to all units in component” (Figure 61.14) in the “Edit line Options...” context menu.

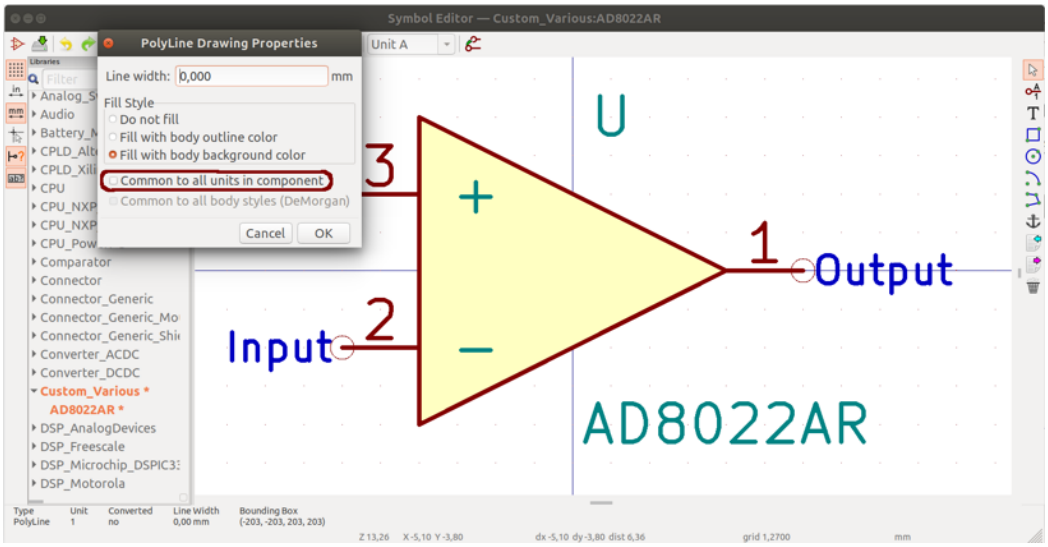
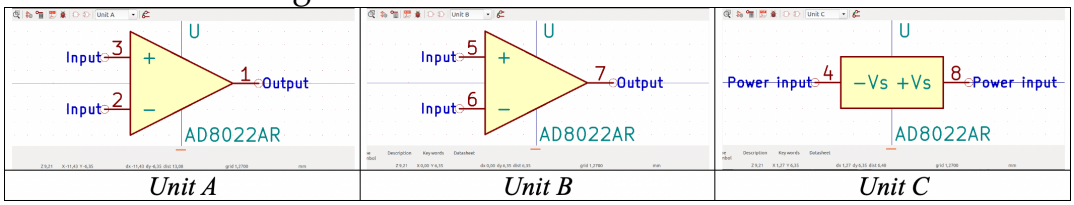


Figure 61.14: Disable "Common to all units in component"

Note that this option does not duplicate line and background to other units, but let appear line and background simultaneously on all units. Additionally it can be used also as a way to transfer line and background to another unit while removing it from elsewhere. If not properly used, this may cause unexpected effects. Likely the best method is to draw another triangle at unit B and a rectangle at unit C. We should then have all three units ready:



Unit A *Unit B* *Unit C*

Figure 61.15: AD8022AR, completed pins and drawings

To finalize the symbol, add in the library symbol properties (See Figure 61.4) the remaining information and save accordingly:

Item	Change
Footprint	Package_SO:SOIC-16_3.9x9.9mm_P1.27mm (Selected from footprint library)
Datasheet	https://www.analog.com/media/en/technical-documentation/data-sheets/AD8022.pdf
Description	Dual High Speed,Low Noise Op Amp, SOIC-8
Keywords	Dual High Speed,Low Noise Op Amp

Figure 61.16: Additional symbol properties for AD8022AR

Texas Instruments DS3486

The DS3486 line receiver does not exist in the KiCad libraries, included along with the KiCad installation. Since it is a replacement for the MC3486N, which is in one of the libraries, we could in principle add the DS3486 as an

alias. But because, as we have seen above, the KiCad libraries are not writable, we need to at least add the DS3486 in one of our own libraries and use the MC3486N as base.

If we look at the MC3486N we see that it each unit has a set of power pins.

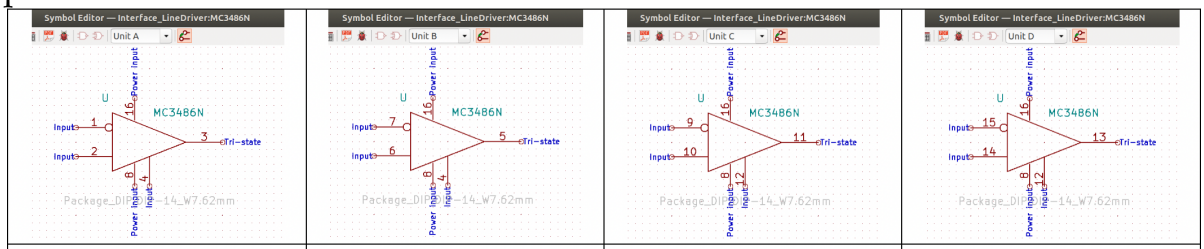


Figure 61.17: MC3486N, completed pins and drawings

This is not according to the KLC. Problematic are also redundant pins for TRI-STATE control input pins 4 and 12. To overcome this, we will copy the MC3486N into our own library and improve it from there. Start with searching for the MC3486N in the symbol editor. Then right click on it and select Copy, navigate to your own library (if not present, create it before) and select Paste symbol:

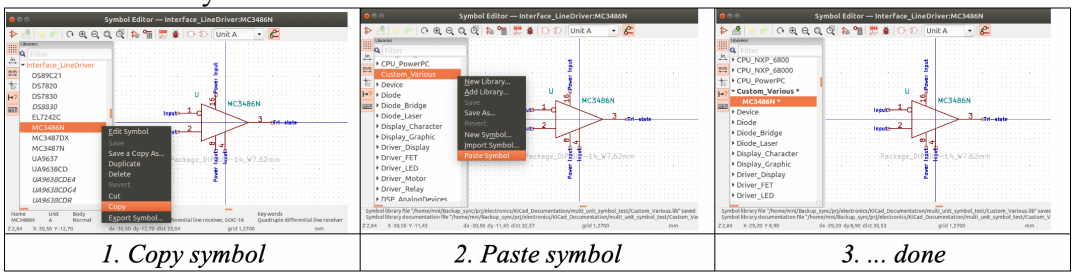


Figure 61.18: Copy symbol between libraries

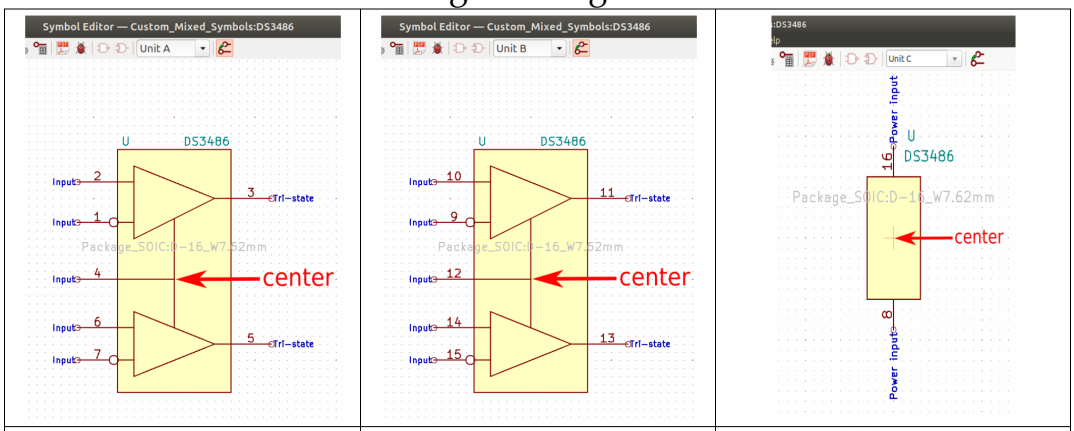
Then make it active by double clicking on the newly created symbol. We need to change among others the value from MC3486N into DS3486. Click on Edit Symbol Properties (Figure 61.4) and modify the following:

Item	Change
Value	DS3486
Footprint	Package_SO:SOIC-16_3.9x9.9mm_P1.27mm (Selected from footprint library)
Datasheet	https://www.ti.com/lit/ds/symlink/ds3486.pdf
Description	DS3486 Quad RS-422, RS-423 Line Receiver, SOIC-16
Keywords	DS3486 Quad RS-422, RS-423 Line Receiver
Number of Units	3

Figure 61.19: Changes for MC3486N to DS3486

When done, click on OK.

Further changes are needed, but to prevent unexpected changes between units, make sure to disable the “synchronized pin edit mode”(Figure 61.14) and the “common to all units in component” in the “Polyline drawing properties” (Figure 61.15). Now using the provided tools, change each unit that it resembles the following drawings:



Unit A: Input A and C	*Unit B: Input B and D*	*Unit C: Power supply*

Figure 61.20: DS3486 drawing of each unit

Unit A and B are centered at the junction of the vertical and horizontal line of pin 4 respectively pin 12. Alternatively unit C can be drawn horizontally as can be seen in table 3. Make sure the center is located in the middle between the two pins. When done, press Ctr-S to save your work.

Library component validating

If planning to share library components with the community, it is recommended to validate these with a set of Python scripts[35] which are developed by the KiCad library team. Without going into details, please find more information at the corresponding link in the footnote.

[35] See: https://github.com/kicad/kicad-library-utils (last visited on 16-09-2019)

61. KLC RULE F5.3

This chapter is sourced from the KiCad library documentation.

You can find the original at:
https://www.kicad-pcb.org/libraries/klc/F5.3/

Accessed December 30, 2019.

Courtyard layer requirements

The component courtyard is defined as the smallest rectangular area that provides a minimum electrical and mechanical clearance around the combined component body and land pattern boundaries. It is allowed to create a contoured courtyard area using a polygon instead of a simple rectangle. (IPC-7351C)

The courtyard should include any extra required clearance for mating connectors (for example).

KiCad refers to the courtyard layers as:

- F.CrtYd - Front courtyard layer
- B.CrtYd - Back courtyard layer

A fully enclosed Component courtyard must be drawn on the F.CrtYd layer, with the following parameters:

1. Courtyard uses 0.05mm line width
2. All courtyard line elements are placed on a 0.01mm grid
3. If the component requires a courtyard on the back of the PCB, a corresponding courtyard must be provided on the B.CrtYd layer.
4. Where the courtyard depends on the dimensions of the physical part body, clearance is calculated from the nominal part dimensions.

Courtyard clearance should adhere to the following requirements:

5. Unless otherwise specified, clearance is 0.25mm
6. Components smaller than 0603 should have a clearance of 0.15mm
7. Connectors should have a clearance of 0.5mm, in addition to the clearance required for mating of connector
8. Canned capacitors should have a clearance of 0.5mm
9. Crystals should have a clearance of 0.5mm
10. BGA devices should have a clearance of 1.0mm

Example for courtyard clearance when applied to the contoured courtyard outline of a QFP-64 package. (Shown are courtyard, fab-outline and copper pads.)