INNOVATIONS... MONTAGES FIABLES... ÉTUDES DÉTAILLÉES... ASSISTANCE LECTEUR

ELECTRONIQUE

ELECTRONIQUE ET LOISIRS magazine

www.electroniquemagazine.com

n° 145 HIVER 2018

3DRAG+, Imprimez en GRAND !

VoIP sur RaspberryPi

- Breakout Board NE555
- Une crèche avec Arduino
- Amplificateur BF 2 x 15 w
- Sapin de Noël électronique
- Détecteur de pluie
- Apprenons à utiliser RandA - 4
- Cours MPLAB X - 5
- CAO kiCad EDA - 5

N° 145 Décembre 2018

L 13270 - 145 - F: 8,30 € - RD

France 8,30€ - BEL/LUX : 8.80€ - PORT : 8.80€ - DOM/S : 8.30€ - NC/S : 990 CFP – POL/S : 1300 CFP – CAN : 12.70$ CAD

Sommaire

ARTICLES

Numéro 145
Hiver 2018
100 pages

IMPRIMEZ EN GRAND AVEC LA 3DRAG+

Capable de produire des impressions de 400 mm x 400 mm x 400 mm, elle intègre toutes les fonctionnalités de la 3DRAG, y compris la possibilité d'upgrade de la carte contrôleur basée sur Arduino.

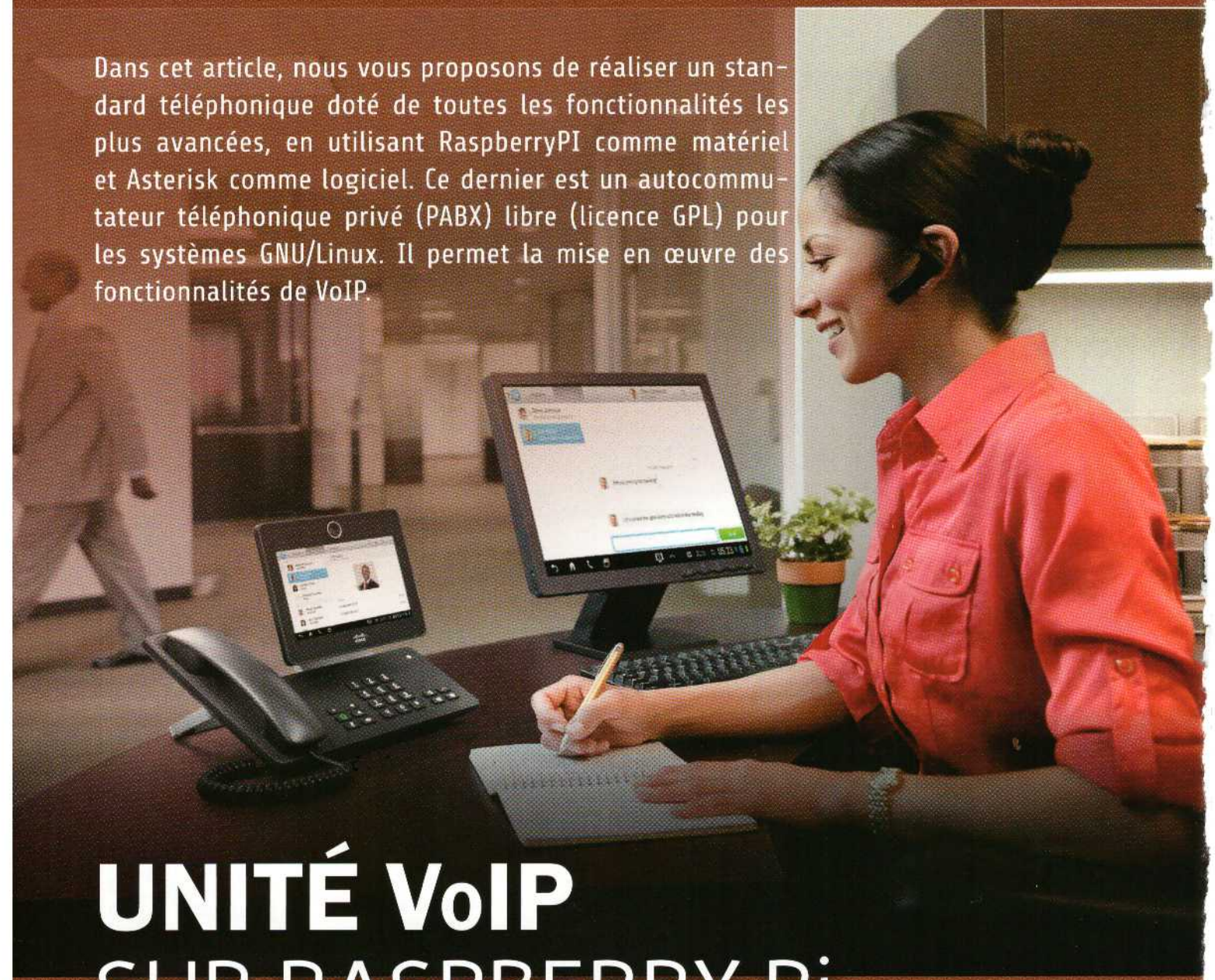
La Rédaction vous souhaite de Bonnes Fêtes de fin d'année et vous donne rendez-vous en 2019

Les typons des circuits imprimés et les programmes **lorsqu'ils sont libres de droits** sont téléchargeables à l'adresse suivante :

www.electroniquemagazine.com

Allez dans le sommaire de la revue 145 à l'onglet « Télécharger » (bas de la page web).

Dans cet article, nous vous proposons de réaliser un standard téléphonique doté de toutes les fonctionnalités les plus avancées, en utilisant RaspberryPI comme matériel et Asterisk comme logiciel. Ce dernier est un autocommutateur téléphonique privé (PABX) libre (licence GPL) pour les systèmes GNU/Linux. Il permet la mise en œuvre des fonctionnalités de VoIP.



UNITÉ VoIP SUR RASPBERRY Pi

Première partie

de Giuseppe Mazzucato

L'acronyme « **VoIP** », qui signifie « **Voice over IP** » ou « **Voix sur IP** », désigne un ensemble de technologies utilisées pour transporter la voix en exploitant les réseaux IP filaires (câble/ADSL/fibre optique) ou encore les réseaux satellites (Wi-Fi, GSM, UMTS ou LTE), qu'il s'agisse de réseaux privés ou d'Internet. En réalité, ces technologies ne se limitent pas au transport de la voix, mais peuvent également être utilisées pour transférer des fichiers, des vidéos, des SMS, etc. Il ne s'agit pas d'une technologie nouvelle, car elle est née avec l'arrivée d'Internet.

Cependant, alors qu'elle se limitait initialement à certains réseaux de communication, elle s'est développée ces dernières années sur le web grâce à l'apparition des box de plus en plus performantes. Les opérateurs téléphoniques utilisent désormais la VoIP pour les communications de leurs clients, ainsi que pour le réseau 4G. Toutes les communications sont acheminées sous la forme de paquets de données.

Le principal avantage qui pousse vers cette transition est ce que l'on appelle la « convergence », c'est-à-dire la possibilité d'utiliser un seul réseau constitué de paquets qui permettent d'acheminer toutes sortes d'informations, qu'il s'agisse de données ou d'autres contenus multimédias et donc la voix (voir la figure 1). La voix est donc convertie en signaux numériques puis regroupée en paquets qui sont ensuite transmis à l'aide du protocole IP.

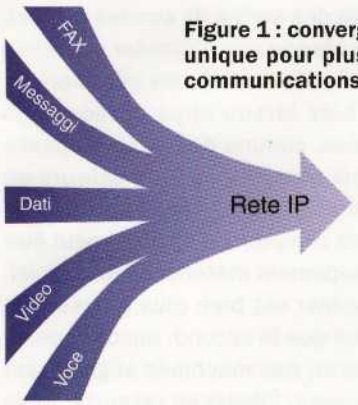


Figure 1 : convergence : un réseau unique pour plusieurs formes de communications.

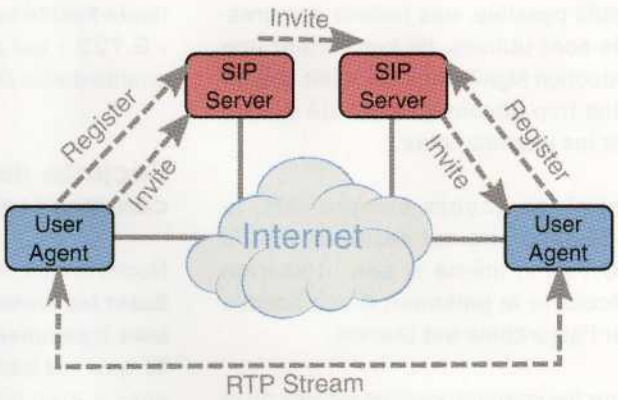


Figure 2 : architecture SIP.

Contrairement au transfert de contenus, tels que la musique ou les vidéos en streaming, une communication vocale doit s'effectuer en temps réel entre deux interlocuteurs ou plus.

Il n'est pas possible d'introduire plus de quelques dizaines de millisecondes de temps de latence, ainsi la taille du paquet ne peut donc pas être supérieure à une dizaine de millisecondes.

Protocoles utilisés

Pour la communication VoIP plusieurs protocoles ont été développés, à la fois libres (ouverts) et propriétaires. Ici, nous allons nous concentrer sur les protocoles **SIP** et **RTP,** qui sont les plus utilisés car ils ont l'avantage d'être complètement ouverts.

Les deux protocoles fonctionnent de connivence. Le protocole **SIP** (Session Initiation Protocol) **établit une connexion entre deux interlocuteurs** à travers les éléments du réseau, tandis que le protocole **RTP** (Real-time Transport Protocol) **assure le transport du contenu multimédia** entre les deux extrémités.

Le **protocole SIP ne transporte pas les données échangées** durant la session comme la voix ou la vidéo, il est indépendant de la transmission des données.

Le **protocole RTP** est un protocole de communication informatique permettant **le transport de données** soumises à des contraintes de temps réel, telles que des flux média audio ou vidéo. Dans le protocole SIP (voir la figure 2), deux types d'éléments sont définis : les « **endpoint** » ou « **UserAgent** », qui sont les terminaux d'où proviennent les connexions et les serveurs SIP, qui sont les éléments du réseau par lesquels ils transitent. Une des fonctions principales du serveur SIP est la fonction « **Register** » qui permet de **localiser les « UserAgent »**.

Périodiquement, chaque « UserAgent » s'enregistre sur son serveur SIP en communiquant l'adresse IP (ainsi que le port) à partir de laquelle il est accessible, afin que le serveur puisse le contacter en cas d'appel entrant.

Une autre fonction du serveur SIP est celle du « Proxy », qui permet le transfert de messages d'un « UserAgent » à un autre via un ou plusieurs serveurs SIP.

Le transfert RTP peut être dirigé entre les « UserAgent » à l'aide du serveur SIP qui agit comme une « passerelle multimédia » entrant dans la chaîne de transfert afin d'effectuer, par exemple, des conversions de formats.

Le codage de la voix

Pour être transmise, la voix doit être transformée en un flux de données sous la forme de bits (c'est-à-dire une suite de 0 et de 1) selon un processus appelé « codage », puis recomposée par le processus complémentaire qui est celui du « décodage ».

Ces processus sont réalisés par des dispositifs CODeurs/DECodeurs appelés « **CODEC** » pour plus de simplicité.

Un « CODEC » est donc un **dispositif matériel ou logiciel permettant de mettre en œuvre l'encodage ou le décodage d'un flux de données numériques**, en vue d'une transmission ou d'un stockage. Certains « CODEC » intègrent également une fonction de compression ou encore de chiffrement des données.

Il existe de nombreux types de « CODEC », chacun représentant un compromis entre la qualité du signal et la bande passante requise pour la transmission de ce dernier. Chaque type de « CODEC » est adapté à un domaine d'utilisation spécifique (son, vidéo, etc.). Le tableau 1 présente les codecs les plus couramment utilisés pour les communications VoIP.

Le codec « G.711 » a été le premier à être mis en œuvre, il utilise l'algorithme le plus simple. En fait, la voix est simplement échantillonnée à une fréquence de 8 kHz et quantifiée sur 8 bits selon une grille non linéaire permettant de diminuer le rapport signal/bruit et l'erreur de quantification pour les sons de faible amplitude. Il atteint un débit de 64 kbit/s.

Pour la transmission sur Internet, afin d'obtenir la bande passante la plus

Tableau 1 - Codecs les plus utilisés.

Codec	Débit min	Débit max	Qualité
G.711	64 Kbit/s	90 Kbit/s	Bonne
GSM	13 Kbit/s	45 Kbit/s	Suffisante
G.729	8 Kbit/s	35 Kbit/s	Suffisante
G.722	64 Kbit/s	90 Kbit/s	Très Bonne

faible possible, des codecs compressés sont utilisés. Ils permettent une réduction significative du débit binaire sans trop pénaliser la qualité perçue par les interlocuteurs.

Parmi les codecs compressés, le plus populaire est certainement le « G.729 », même si son utilisation nécessite le paiement d'une licence car l'algorithme est breveté.

Pour les communications locales où la bande utilisée n'est pas un problème, il est possible d'utiliser des codecs haute-fidélité ou HD, tels que le codec « G.722 » qui permet d'obtenir une qualité audio élevée.

Sécurité des communications VoIP

Normalement, les **communications utilisant les protocoles « SIP » et « RTP » sont transmises en clair**. Cela signifie qu'il est possible d'intercepter les appels avec n'importe quel renifleur (sniffer) capable de capturer des paquets transmis.

Ce sont des sortes de sondes placées sur un réseau afin d'écouter et parfois récupérer à la volée des informations sensibles lorsqu'elles ne sont pas chiffrées, comme des mots de passe (parfois sans que les utilisateurs ou les administrateurs du réseau ne s'en rendent compte). Le renifleur peut être un équipement matériel ou un logiciel. Le premier est bien plus puissant et efficace que le second, encore que, la puissance des machines augmentant sans cesse, l'écart se resserre. Mais le premier est surtout beaucoup plus cher que le second.

Échange de messages

Les séquences possibles d'échange de messages sont très nombreuses et varient en fonction de l'évolution de l'appel. Nous allons voir ici deux exemples simples qui nous permettent d'expliquer les aspects fondamentaux. Les exemples illustrés ont été capturés avec l'outil gratuit « Wireshark », très utile également pour le débogage. « Wireshark » est un analyseur de paquets libre et gratuit. Il est utilisé dans le dépannage et l'analyse de réseaux informatiques, le développement de protocoles, etc.

Chaque « User Agent », pour pouvoir être suivi par le serveur SIP et pouvoir ainsi recevoir les appels entrants, doit « s'enregistrer » en communiquant son identité et son adresse au serveur SIP. Dans le diagramme ci-contre, nous notons que, pour des raisons de sécurité, le serveur d'adresse 192.168.88.100 n'accepte pas la première tentative d'enregistrement de l'agent ayant l'adresse 192.168.88.253, en répliquant avec le message « 401 Unauthorized » (voir la figure 3). Ce message contient le mécanisme utilisé avec les informations d'identification pour composer la deuxième tentative d'authentification, qui dans ce cas a abouti. L'enregistrement reste valable pour une période limitée et doit donc être périodiquement actualisé en répétant la procédure à intervalles réguliers.

Nous voyons ici le cas d'un appel entre deux postes : le poste 200 à l'adresse 192.168.88.253 qui appelle le poste 201 à l'adresse 192.268.88.254. Le processus commence par le message « INVITE » qui, une fois authentifié, est transféré du serveur SIP à l'« User Agent » destinataire. Le message contient l'identité de l'appelant (from: 200@192.168.88.100) et de l'appelé (to: 201@192.168.88.100) dans un format similaire à une adresse email, c'est-à-dire « nom_utilisateur @ nom_serveur » ainsi que de nombreuses autres informations que nous ne détaillerons pas ici (voir la figure 4).

En attendant que l'appelé réponde, des messages sur l'état d'avancement, tels que « Trying » et « Ringing », sont transmis à l'appelant sous la forme d'indications visuelles ou de tonalités sonores. Lorsque l'appel est accepté, le message « OK » est envoyé, ce qui active le flux audio qui est transmis par le protocole « RTP ». L'appel prend fin lorsque l'un des deux interlocuteurs raccroche et que les messages « BYE » s'affichent, libérant ainsi les ressources utilisées par chaque section de la communication.

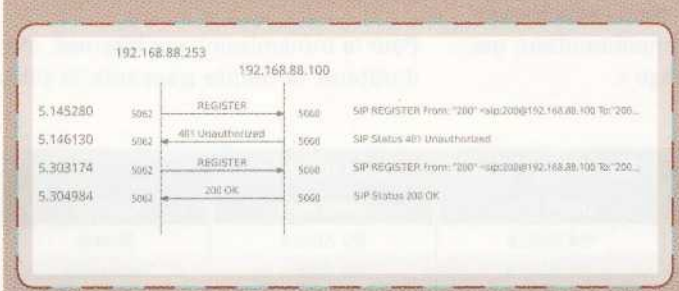


Figure 3 : processus d'enregistrement.

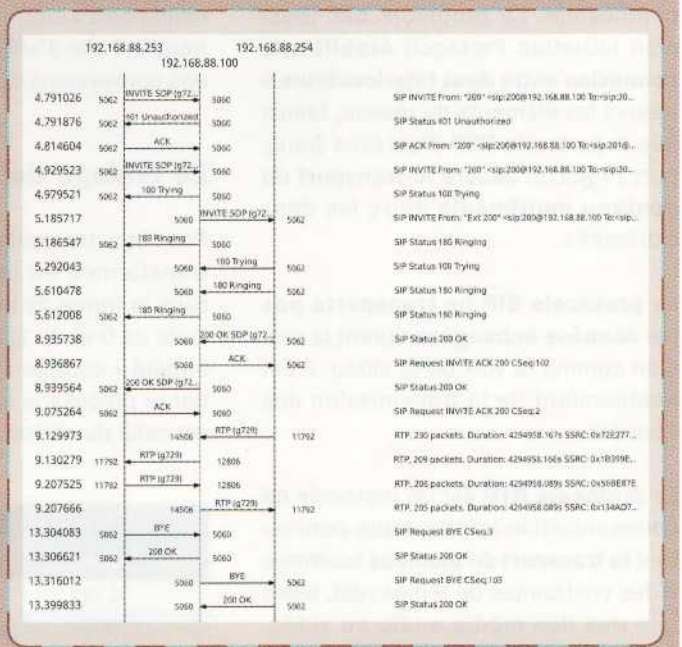


Figure 4 : appel type.

La technique de translation d'adresses ou « NAT »

Lorsque la norme « IPv4 » a été définie en 1981, les adresses étaient codées sur 4 octets (32 bits). À cette époque, cela semblait largement suffisant pour identifier tous les ordinateurs du monde, mais cette limite a été dépassée depuis longtemps. Avec l'arrivée d'internet au début des années 2000, il y avait peu d'adresses disponibles en comparaison du nombre croissant de machines connectées sur le web. Il fut donc décidé de réserver des intervalles d'adresses à des usages privés uniquement, en conséquence ces adresses n'étaient plus routables sur Internet.

La technique NAT (Network Address Translation) a été introduite. La translation d'adresse est utilisée pour permettre aux machines du réseau privé d'accéder à Internet, et de façon générale à d'autres réseaux. Le principe de base est simple puisqu'il s'agit de remplacer à la volée les champs d'adresses dans les paquets qui sont destinés à un autre réseau, ce qui implique que la NAT soit effectuée entre les 2 interfaces réseau, entre le réseau privé et les autres.

Cette technique permet donc à plusieurs ordinateurs de partager une seule adresse publique en attendant que le nouveau standard « IPv6 » prenne effet avec ses adresses de 128 bits. Les ordinateurs d'un réseau interne se voient attribuer des adresses privées identifiées par des adresses classiques « 192.168.x.y » (mais aussi « 10.x.y.z » ou « 172.16.x.y ») qu'ils utilisent pour communiquer directement les uns avec les autres, tout en communiquant avec l'extérieur à l'aide de la même adresse IP publique. Le routeur situé entre le réseau privé et le réseau public transforme les adresses et les ports des messageries et met à jour une table de conversion lui permettant de gérer les connexions entre les interlocuteurs externes et les correspondants du réseau interne.

Cependant, la technique NAT peut créer des problèmes pour la VoIP en raison du manque de correspondance entre les adresses introduites dans les messages SIP (messages privés) et celles insérées dans les « header IP », c'est-à-dire les adresses publiques. Pour éviter de tels problèmes, il existe différentes techniques, qui consistent toutes essentiellement à faire connaître aux machines du réseau privé quelle sera leur adresse publique, par exemple via un serveur STUN. Un serveur STUN permet de passer des appels vers un opérateur VoIP hébergé hors du réseau privé (ou local). Ainsi, la technique NAT présente à la fois des inconvénients et des avantages au niveau de la sécurité pour les machines du réseau privé.

En terme d'inconvénient, la mise en œuvre de la technique NAT n'est pas une opération anodine et ce bien qu'elle ait pour vocation d'être transparente. En effet, la NAT modifie les paquets IP et cela a pour conséquence directe de perturber le contrôle d'intégrité au niveau IP et même aux niveaux supérieurs TCP.

Un des avantages du NAT est de protéger les machines du réseau privé d'attaques directes puisqu'elles ne sont pas accessibles de l'extérieur.

De plus, dans la majorité des cas, les requêtes de connexions ne peuvent provenir que des machines privées. Cela permet également de se prémunir contre un monitoring du trafic qui viserait à scruter les communications entre 2 machines particulières, un serveur sur Internet par exemple et une machine du réseau privé.

Comme cette dernière n'est plus identifiable, l'opération devient impossible à moins de remonter au niveau applicatif (d'où l'utilité d'utiliser une protection/chiffrement à ce niveau également).

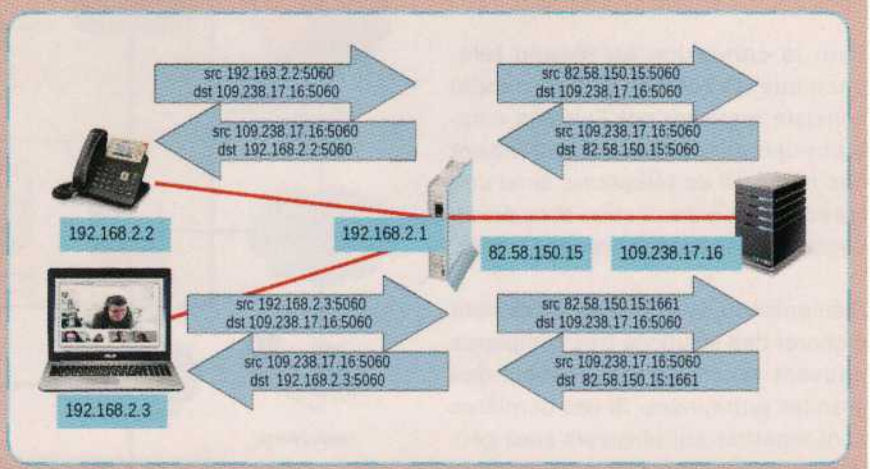


Figure 5 : La technique de translation d'adresses ou « NAT ».

Pour ceux qui ont des besoins de confidentialité, **il est possible de crypter le canal audio** à l'aide du **protocole « SRTP »**, qui, comme indiqué par la lettre S de l'acronyme, n'est rien d'autre que la version sécurisée du protocole « RTP ».

De même, pour protéger les messages, c'est-à-dire les informations sur les numéros appelant et appelés, ainsi que l'heure et la durée de la conversation, il est possible de chiffrer les messages avec le protocole TLS (Transport Layer Security).

Le degré de protection obtenu avec ces mesures est très élevé.

En fait, les navigateurs utilisent le même protocole lorsqu'ils utilisent le protocole « **HTTPS** » (avec les mêmes algorithmes de cryptage).

Au niveau sécurité, il est important de noter que pour éviter des intrusions dans le système de communication, **il est nécessaire d'utiliser des mots de passe suffisamment complexes** pour qu'ils ne puissent pas être « crackés » (piratés).

Dans tout système VoIP exposé sur le réseau public, il y a plusieurs tentatives d'intrusions par jour. Le système consiste à essayer quelques centaines de paires d'utilisateur et de mot de passe les plus courants avant d'abandonner la tentative.

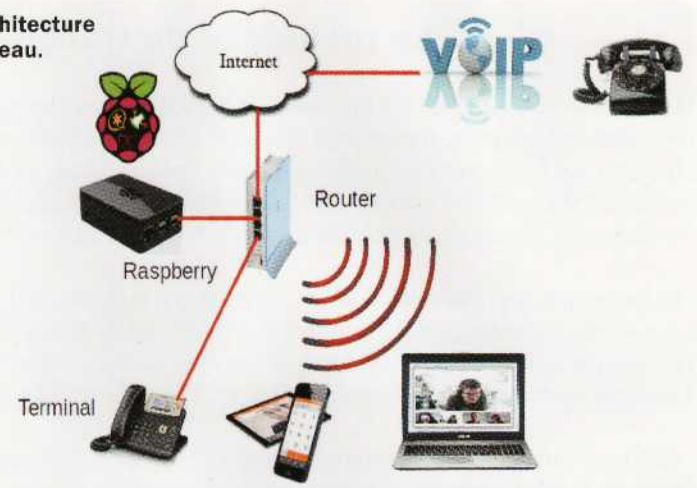


Figure 6 : architecture d'un petit bureau.

Architecture d'un système de communication

Le système de communication pour un petit bureau peut être très simple. Il suffit de connecter un mini serveur, tel qu'un RaspberryPi, sur le réseau local auquel sont connectés des appareils téléphoniques utilisés par des personnes (utilisateurs). L'architecture est représentée en figure 6.

Comme terminaux téléphoniques, il est possible d'utiliser des téléphones spécialement conçus à cet effet ou des logiciels (logiciels de téléphonie) à installer sur un PC, sur un smartphone ou encore sur une tablette.

Pour la connexion au réseau téléphonique, la solution la plus simple consiste à passer par l'un des nombreux opérateurs VoIP qui fournissent des numéros de téléphone ainsi que les échanges à destination et en provenance du réseau téléphonique public.

Bien entendu, vous pouvez également élaborer des solutions très complexes pouvant répondre aux besoins des grandes entreprises. Si ces dernières sont réparties sur plusieurs sites géographiques, il est possible de créer un système intégré dans lequel plusieurs serveurs peuvent être installés dans les principaux bureaux afin de diviser la charge.

Les plus petits sites peuvent toutefois être configurés en tant que satellites de sites plus importants et utiliser les services à distance. Il est possible d'arriver au cas extrême des « télétravailleurs » qui peuvent facilement travailler de chez eux ou lorsqu'ils sont en déplacement (voir la figure 7).

Panorama des appareils de communication

Le marché propose une large gamme de périphériques pouvant être utilisés en tant que composants d'une solution VoIP. Les fabricants d'appareils certifient les plates-formes de communication avec lesquelles leurs appareils sont compatibles, en privilégiant évidemment les plates-formes ouvertes et les plus populaires.

Les téléphones IP sont en effet des mini-ordinateurs spécialisés dans la

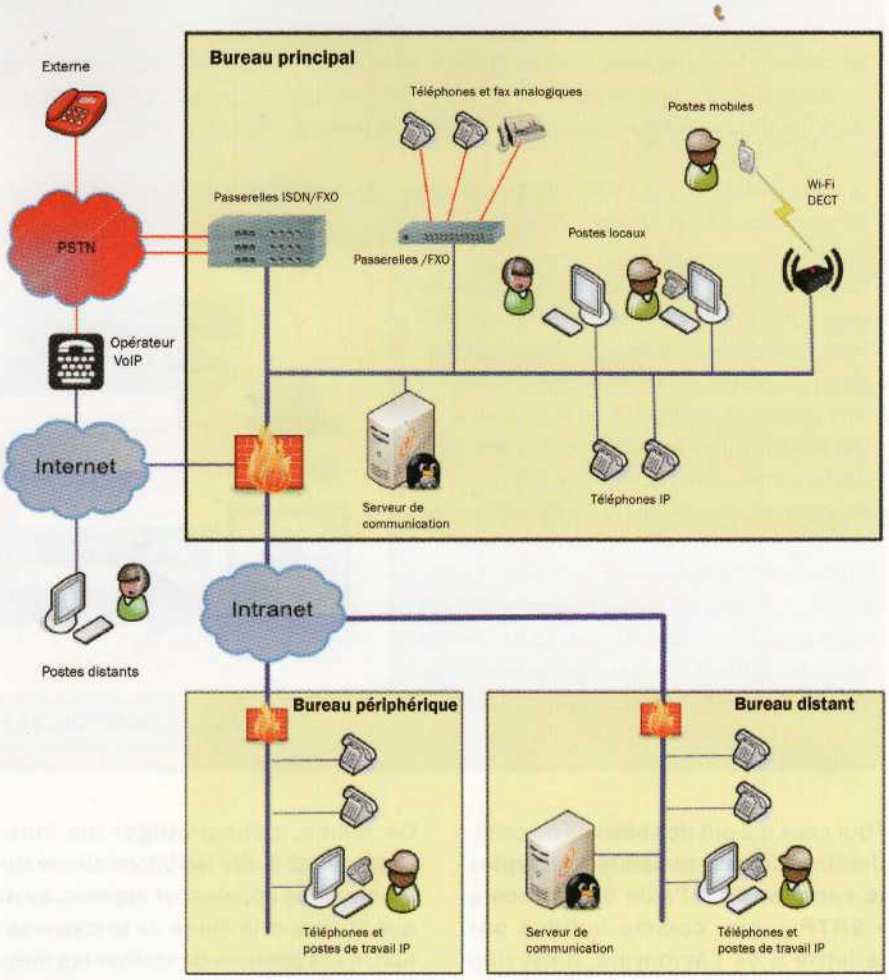


Figure 7 : architecture multi site.

fonction téléphonique VoIP avec une structure physique similaire à celle des téléphones classiques.

Il existe de nombreux modèles ayant des prix très différents, avec des caractéristiques variables en termes de taille et d'affichage, de nombre de touches de fonction, de nombre de lignes, de design et de qualité.

Des téléphones sans fil sont également disponibles pour les utilisateurs pouvant utiliser le réseau Wi-Fi comme une radio, ou la norme « DECT » plus performante en termes de portée et de durée de vie de la batterie.

En plus des téléphones physiques, des logiciels de téléphonie sont disponibles pour chaque système d'exploitation. Ils peuvent être installés sur un PC, un smartphone ou une tablette et connectés via le réseau Wi-Fi. Ces terminaux ont généralement des fonctionnalités plus avancées que les téléphones physiques, telles que la messagerie instantanée ou encore la communication vidéo. Il existe également des téléphones IP à usage spécifique, tels que des interphones équipés d'actionneurs ouvre-porte, d'interfaces pour les systèmes d'annonce, des téléphones résistant aux produits chimiques et conçus pour un usage intensif.

Une partie importante des appareils est représentée par la famille des passerelles, ces dernières convertissent les communications VoIP vers d'autres formats de communication traditionnels.

Il existe donc des passerelles pour les lignes analogiques, pour les lignes RNIS et pour l'interfaçage des réseaux mobiles GSM/UMTS/LTE.

Ces dernières années, plusieurs opérateurs téléphoniques VoIP ont vu le jour, fournissant un accès au réseau téléphonique public directement via le protocole SIP.

Passons à la pratique

Jusqu'ici, nous avons abordé les bases de la technologie VoIP sur lesquelles repose le mini-standard que nous allons construire.

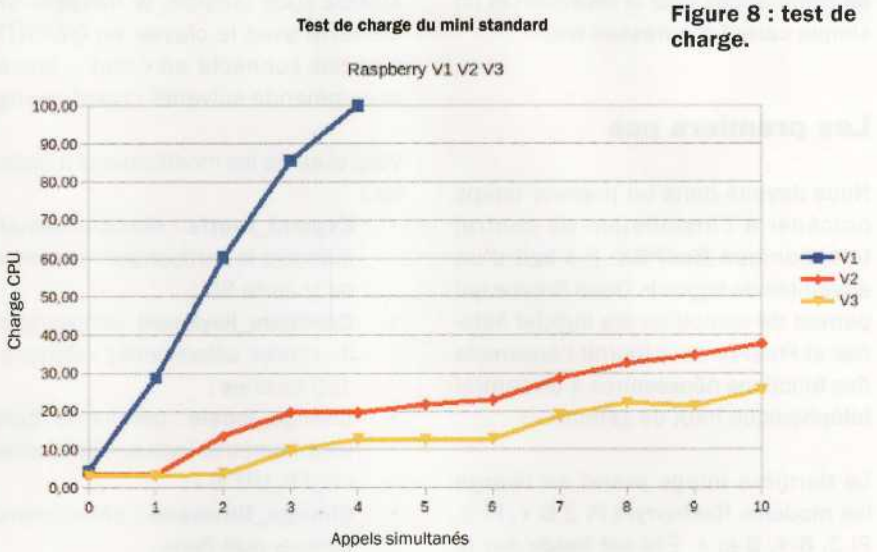


Figure 8 : test de charge.

Nous pourrons ainsi appliquer ces concepts dans la pratique. Le standard que nous vous proposons de mettre en œuvre est complet et dispose de toutes les fonctionnalités les plus avancées.

Il est basé sur un RaspberryPi qui fait office de plate-forme matérielle. Vous connaissez déjà le RaspberryPi, car nous avons publié de nombreux articles dans la revue. L'une de ses applications les plus fréquentes est celle de l'implémentation d'un serveur basé sur le système d'exploitation Linux.

Dans notre projet, nous allons installer le logiciel Open Source **Asterisk** afin de réaliser notre application cible, à savoir un central téléphonique VoIP, petit mais fonctionnel. Le logiciel Asterisk, qui est devenu au fil des ans le standard le plus utilisé dans le domaine professionnel, fournit toutes les fonctions téléphoniques imaginables et est également très « léger » en termes d'occupation des ressources matérielles.

Ce logiciel fonctionne également avec la première version du RaspberryPi, mais avec toutefois une limitation à 2 ou 3 appels simultanés, une limite qui ne conviendrait pas pour les petits bureaux.

L'augmentation des performances introduite avec les versions 2 et 3 du RaspberryPi permet facilement de gérer plusieurs appels simultanés, ce qui permet au système de répondre aux besoins des utilisateurs de petite et moyenne taille (voir la figure 8).

Dans la distribution du RaspberryPi, outre le système d'exploitation Raspbian et le moteur de communication Asterisk, l'interface graphique **FreePBX** est incluse, ce qui permet de **configurer le central simplement à partir du navigateur**, sans devoir modifier des fichiers de configuration qui sont complexes. Certains paquets ont été ajoutés, tels que des messages audio en français, le panneau

Figure 9 : la fenêtre de login de FreePBX.

de commandes pour la réception et un simple carnet d'adresses web.

Les premiers pas

Nous devons dans un premier temps procéder à l'installation du central téléphonique **RasPBX**. Il s'agit d'un ensemble de logiciels Open Source qui permet de compléter les logiciel Asterisk et FreePbx pour fournir l'ensemble des fonctions nécessaires à un central téléphonique haut de gamme.

La dernière image prend en charge les modèles RasberryPi Pi 3 B +, Pi 3, Pi 2, B +, B et A. Elle est basée sur la distribution Raspbian Stretch avec des améliorations et modifications par rapport aux images précédentes basées sur Raspbian Jessie.

Notamment, nous trouverons la version FreePBX 14 avec de nombreuses améliorations et de nouvelles fonctionnalités, la distribution Stretch qui apporte des mises à niveau et des modifications majeures, telles que MariaDB, qui remplace MySQL. MariaDB offre de meilleures performances que MySQL.

Les utilisateurs de RasPBX devraient remarquer une interface graphique plus rapide que les versions précédentes, le serveur PJSIP qui est la pile SIP par défaut et qui « écoute » le port 5060. Le canal SIP des versions antérieures est toujours disponible sur le port 5160.

La dernière image disponible en téléchargement (raspbx-04-04-2018.zip) sur notre site comprend :

- **Asterisk 13.20.0**
- **FreePBX 14.0.2.10**

L'image compressée au format ZIP a une taille d'environ de 1 Go, qui, une fois décompressée, devra être installée sur une carte SD d'**au moins 8 Go** à l'aide de « **Win32diskmanager** », comme nous l'avons déjà vu dans la revue. La vitesse de la carte SD est importante pour les performances du système, nous vous recommandons donc d'utiliser un modèle ayant une classe 10. Insérez la carte SD dans le RaspberryPi, puis connectez le clavier, la souris, le câble Ethernet.

Mettez sous tension, le RaspberryPi démarre avec le clavier en QWERTY. Une fois connecté en « root », lancez la commande suivante : **raspi-config**

Voici ci-après les modifications à apporter :

- **Expand_rootfs** : étend automatiquement la partition sur l'ensemble de la carte SD ;
- **Configure_keyboard** : configuration du clavier, sélectionnez « Generic » 105 touches ;
- **Change_locale** : permet de régler les paramètres locaux, sélectionnez « fr_FR_UTF-8 » ;
- **Change_timezone** : sélectionnez Europe, puis Paris.

Cliquez sur le bouton « **Finish** », puis n'oubliez pas de **redémarrer le RaspberryPi** afin que les modifications prennent effet.

Il est nécessaire de disposer d'une adresse IP fixe, pour cela vous devez entrer dans l'administration de votre box et configurer une adresse IP fixe. En effet, si votre serveur se voit attribuer une autre adresse différente par le DHCP, il faudra reconfigurer tous les téléphones IP.

Pour connaitre l'adresse MAC, tapez la commande : **ifconfig**, notez l'adresse IP de votre serveur.

L'adresse IP peut être modifiée en accédant aux dernières lignes du fichier « /etc/dhcpcd.conf » et en entrant les paramètres de votre réseau. Dans notre cas, nous utiliserons l'adresse par défaut 192.168.88.100, mais pour votre réseau elle doit être différente et vous devez la remplacer par l'adresse réelle définie dans votre réseau. Ci-après, vous pouvez voir les dernières lignes du fichier « **/etc/dhcpcd.conf** » (dans notre cas) :

```
interface eth0
static ip_address=192.168.88.100/24
static routers=192.168.88.1
static domain_name_servers=192.168.88.1
```

Une fois connecté au réseau, après quelques dizaines de secondes nécessaires au démarrage, le RaspberryPi est accessible via le shell « **SSH** » en utilisant le nom d'utilisateur « root » et le mot de passe « raspberry ». Reportez-vous aux anciens numéros (à partir du n°123) traitant des connexions SSH à l'aide du logiciel « **PUTTY** ».

Il va sans dire que l'une des premières choses à faire est de **remplacer le mot de passe par un mot de passe personnel** et sécurisé.

La séquence des opérations sous la forme de lignes de commandes, pour la modification du mot de passe, est la suivante (vous pouvez le faire aussi à partir de la fenêtre « Rasp- config » puis « change pass », assurez-vous que le clavier soit bien en AZERTY) :

```
Prompt $ ssh root@192.168.88.100
root@192.168.88.100's password:
*****
.....
root@raspberrypi:~# passwd
Entrez le nouveau mot de passe UNIX : ******
Ressaisissez le nouveau mot de passe UNIX : ******
passw: mot de passe mis à jour correctement
root@raspberrypi:~#
```

Eventuellement, vous devez intégrer les dernières mises à jour disponibles, pour cela utilisez comme d'habitude les commandes suivantes :

```
apt-get update
apt-get upgrade ou raspbx-upgrade
```

Attention toutefois, les mises à jours de certain fichiers (Networking, Motd, Apache2) **ne sont pas conseillées**.

Lors de l'installation, une série de questions apparaissent pour savoir si vous souhaitez installer les mises à jour de certains fichiers.

Nous vous conseillons de répondre « **N** » (non) pour ces 3 fichiers (la mise à jour d'Apache 2 est vraiment déconseillée), et de conserver la version actuelle de ces fichiers.

Une fois la mise à jour effectuée, vous pouvez afficher la console Asterisk à l'aide de la ligne de commande suivante, afin de vérifier qu'il soit correctement installé : **asterisk –r**

Vous devez voir une ligne s'afficher qui ressemble à celle-ci :

Connected to Asterisk 13.20.0 currently running on raspbx (pid=8014)
raspbx*CLI>

Pour quitter la console, il suffit de taper la commande : **quit**

Une fois ces opérations de configuration terminées, nous allons nous connecter à l'interface FreePBX qui est un outil de configuration graphique très convivial pour Asterisk.

Pour se connecter, nous tapons notre adresse IP de notre RaspberryPi dans un navigateur à l'adresse suivante : http://192.168.88.100 (elle doit être différente dans votre cas, tapez votre adresse IP).

Pour commencer, entrez le login et le mot de passe :

login : **admin**
mot de passe : **admin**

que vous changerez bien entendu par la suite. En haut à droite de l'écran, vous pouvez modifier la langue de l'interface en cliquant sur le bouton « **Language** », choisissez le français (voir la figure 9).

Depuis la page principale, vous pouvez accéder à quatre applications qui couvrent différents besoins :

- **Administration FreePBX** : c'est la zone de configuration du central à partir de laquelle tous les paramètres peuvent être modifiés. Les possibilités de configuration sont nombreuses, nous ne verrons dans cet article que les principales et dans le prochain certaines plus importantes. Pour les autres, reportez-vous à la documentation disponible sur le site https://frwiki.peoplefone.com/wiki/Configuration_de_FreePBX ;
- **Panneau utilisateur** : c'est l'espace de travail attribué par l'administrateur et à partir duquel chaque utilisateur peut personnaliser son réseau interne ;
- **Répertoire téléphonique** : c'est l'application qui vous permet de mémoriser les numéros de téléphone appelés le plus souvent et qu'il est possible de numéroter en un clic ;
- **Panneau de commande** : c'est l'application qui vous permet d'afficher le statut de disponibilité libre/occupé, cela est pratique pour les réceptionnistes et les assistants qui doivent trier les appels téléphoniques vers leurs collègues.

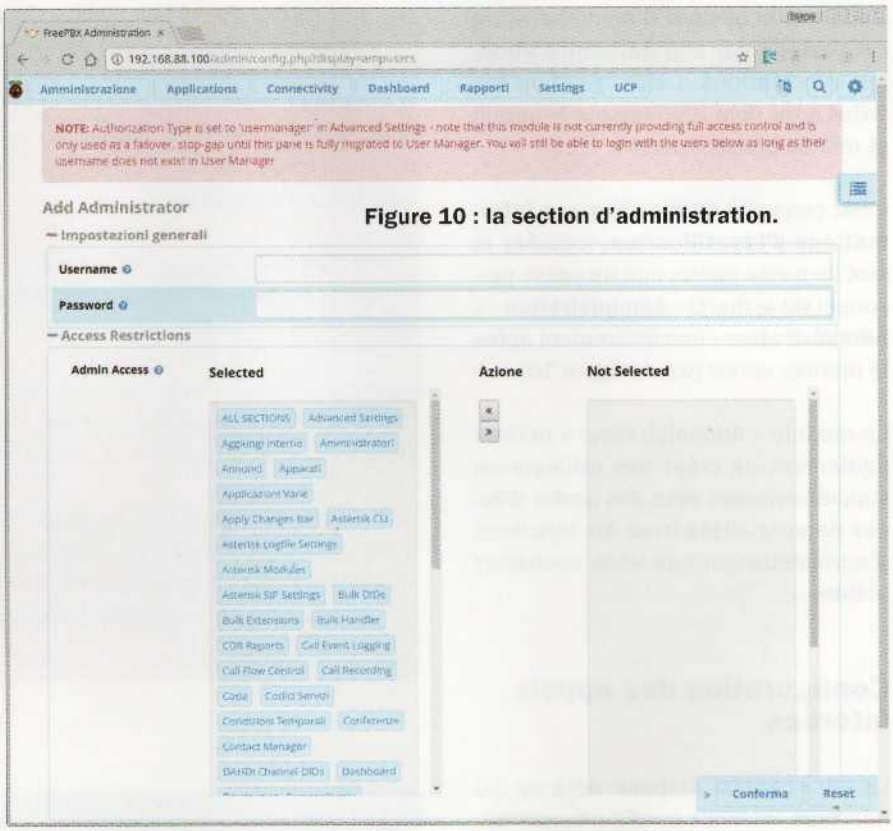


Figure 10 : la section d'administration.

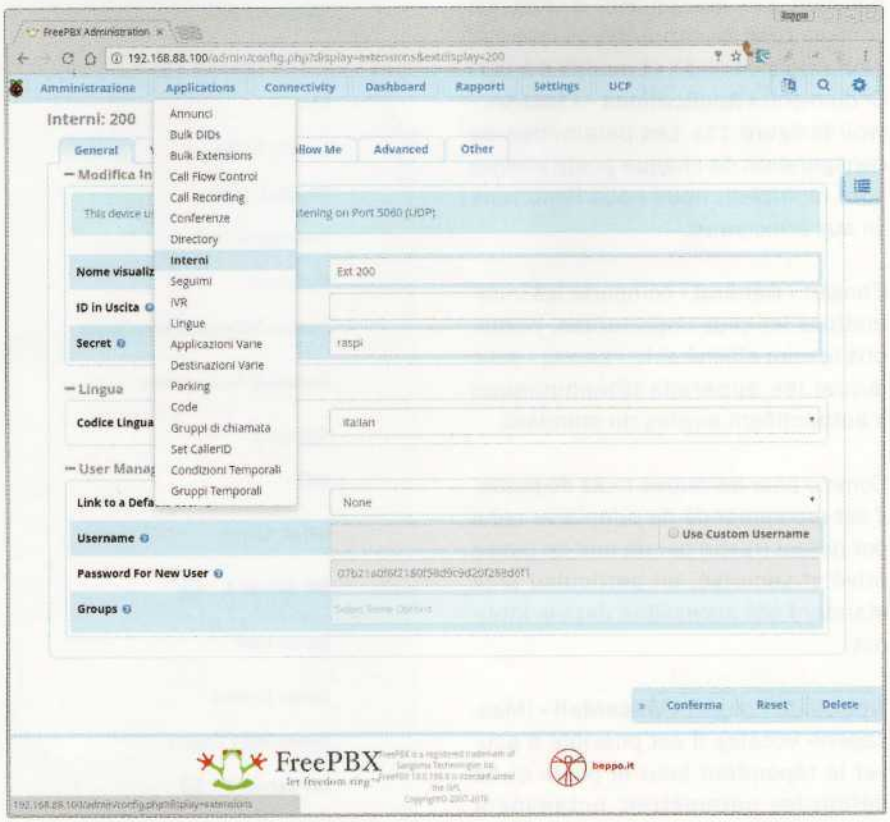


Figure 11 : configuration des postes internes à partir de l'onglet « Applications ».

L'accès à la section d'administration est protégé par une authentification, les informations d'identification initiales sont nom d'utilisateur = admin et mot de passe = admin.

Il est conseillé de **modifier ces informations d'identification**, modifiez le mot de passe par un mot de passe personnel via le menu « **Administration → Administrateur** » immédiatement après le premier accès (voir la figure 10).

Le module « Administrateur » permet également de créer des utilisateurs supplémentaires avec des profils d'accès personnalisés avec les fonctions d'administration que vous souhaitez activer.

Configuration des appels internes

Le mini central dispose déjà de 10 numéros internes configurés correspondant aux postes 200 à 209. La configuration de ces numéros internes peut être visualisée et modifiée à partir du menu « **Applications → Interne** » (voir la figure 11). Les paramètres de configuration de chaque poste interne sont multiples, nous nous limiterons ici aux principaux.

L'onglet « **Général** » comporte les informations les plus importantes, y compris le nom affiché et le « secret » avec lequel les appareils téléphoniques s'authentifient auprès du standard.

Comme pour les autres mots de passe, il est recommandé de remplacer celui par défaut (raspi) par un mot de passe privé et sécurisé, en particulier si le standard est accessible depuis Internet.

À partir de l'onglet « **VoiceMail** » (Messagerie vocale), il est possible d'activer le répondeur pour le poste et de définir les paramètres, notamment le mot de passe et l'adresse email à laquelle envoyer les messages.

L'onglet « **Find Me/Follow Me** » permet de définir une liste de numéros à contacter si le poste interne ne répond pas, comme par exemple un numéro mobile, ou celui d'un assistant.

FreePBX et Asterisk

Le moteur téléphonique Asterisk s'appuie sur une série de fichiers textes de configuration contenus dans le répertoire « /etc/asterisk ». La syntaxe de ces fichiers n'est pas simple et il est facile de commettre des erreurs s'ils sont modifiés manuellement.

C'est pour cette raison que l'interface de FreePBX est devenue très populaire, permettant de définir tous les paramètres de fonctionnement via une interface web pratique et de générer automatiquement les fichiers de configuration d'Asterisk.

Lorsqu'un ou plusieurs paramètre(s) est(sont) modifié(s) à l'aide de l'interface FreePBX, la configuration n'est pas mise à jour immédiatement. Un bouton rouge « Apply Config » (Appliquer la configuration) apparaît en haut de l'interface.

Un clic sur ce bouton met à jour les fichiers de configuration et demande à Asterisk de les télécharger. De cette manière, il est possible d'appliquer plusieurs modifications à un système, même lorsque ce dernier fonctionne, sans impacter le fonctionnement du central téléphonique.

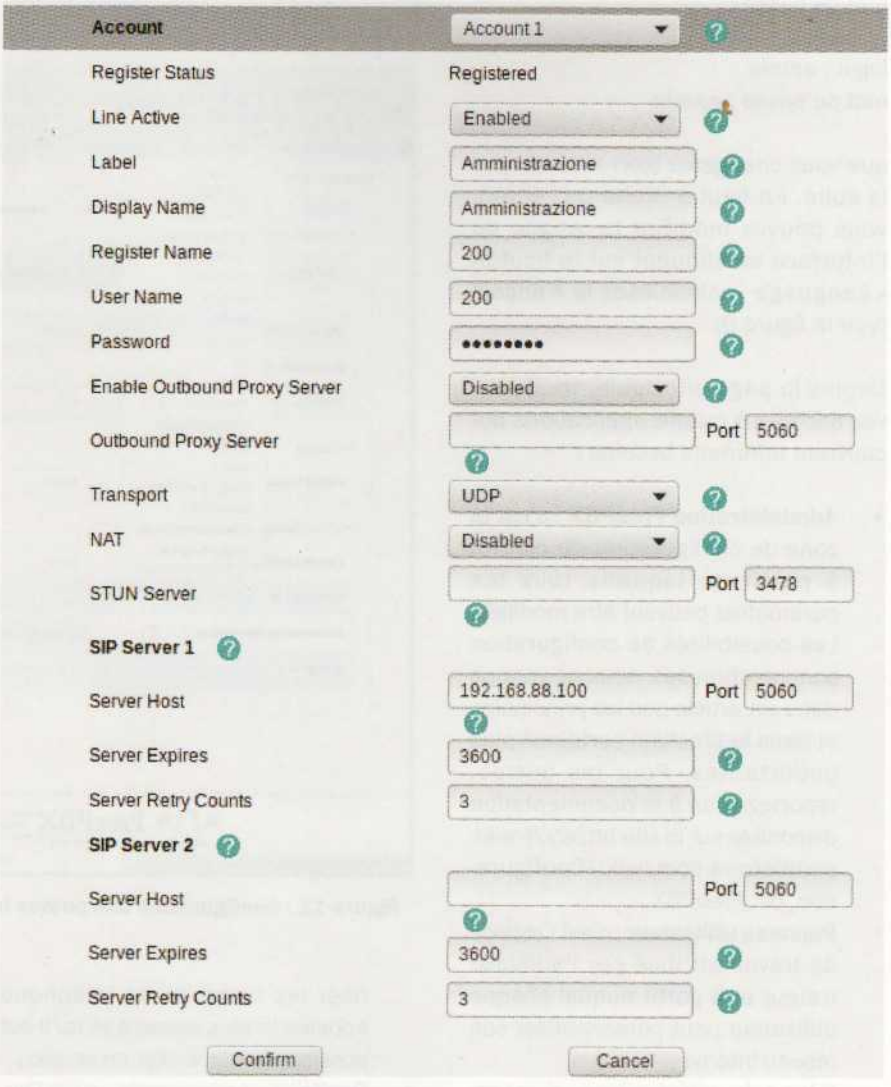


Figure 12 : configuration d'un téléphone (« endpoint ») SIP.

Enfin, l'onglet « **Advanced** » propose beaucoup de paramètres parmi lesquels nous ne mentionnerons que celui du mode NAT. Il doit être réglé sur « No » si l'appareil est sur le réseau avec un standard et sur « Yes » s'il est masqué derrière un NAT.

Le premier appel

Une fois les postes configurés, il est possible de connecter les appareils téléphoniques au central et de commencer les premiers appels internes.

Comme périphériques, nous pouvons utiliser des téléphones IP matériels ou des softphones, c'est-à-dire un logiciel à installer sur un PC ou encore un smartphone. Il existe plusieurs modèles, la seule chose importante est qu'ils doivent supporter le protocole SIP.

La configuration d'un « endpoint » SIP dépend bien évidemment du fabricant et du modèle de téléphone, mais **les paramètres de base à configurer sont toujours au nombre de trois** (voir la figure 12) :

- le **nom d'utilisateur**, parfois aussi appelé « register name », qui correspond au numéro de poste (par exemple 200, 201, 202, etc.) ;
- l'**adresse du serveur SIP** qui doit coïncider avec l'adresse IP du central (dans notre cas : 192.168.88.100) ;
- le **mot de passe**, parfois appelé « secret » défini dans le menu « Applications → Interne ».

Nous allons expliquer comment configurer le softphone « **Jitsi** » qui est un programme Open Source écrit en Java et qui peut être utilisé sur différents systèmes d'exploitation. Ce softphone inclut, outre les fonctions de téléphonie normales, de nombreuses fonctionnalités avancées telles que le chat, la communication vidéo et le partage d'écran.

Le programme peut être téléchargé gratuitement sur le site https://jitsi.org/downloads/ où, en plus des sources, les packages d'installation pour les systèmes d'exploitation les plus répandus sont disponibles.

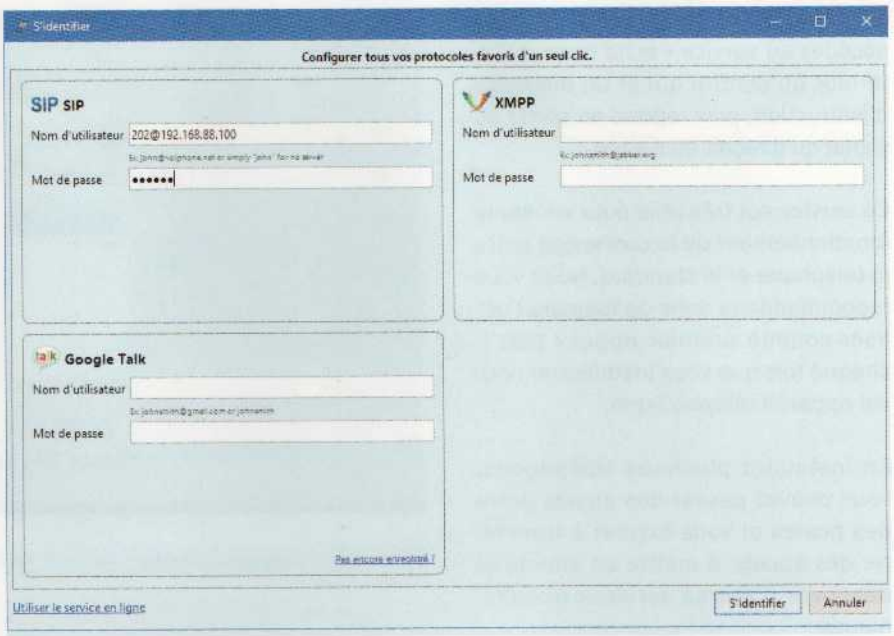


Figure 13 : fenêtre de login de « Jitsi » .

Les canaux SIP « chan_sip » et « chan_pjsip »

Lorsque vous créez un nouveau poste interne ou un nouveau Trunk (c'est un lien qui permet de relier le central téléphonique à l'extérieur), vous remarquez qu'il existe deux types de canaux SIP différents. Pourquoi et quand utiliser l'un ou l'autre ?

« Chan_sip » est le driver historique pour les canaux SIP présent dans Astérisque depuis ses débuts, car à partir de la version 12 il prend en charge un nouveau driver « chan_pjsip » basé sur la librairie gratuite « pjsip » également utilisée dans de nombreux autres projets.

Les avantages de « pjsip » sont essentiellement destinés aux développeurs qui recherchent un environnement plus modulaire et plus facile à étendre. Il est donc très probable que les nouvelles fonctionnalités ne seront prises en charge que par le nouveau driver de canal.

Du point de vue de l'utilisateur, pendant cette période de transition, les deux pilotes sont essentiellement équivalents : « chan_sip » est recommandé pour ceux qui souhaitent privilégier la stabilité du driver qui est plus mature et qui a été le plus testé, tandis que « chan_pjsip » est recommandé pour ceux qui souhaitent disposer des dernières nouveautés.

Lors de la réalisation de notre mini central téléphonique, les deux drivers ont été affectés à des ports différents : « chan_sip » au port standard 5060 et « chan_pjsip » au port 5061.

Lors de la première exécution après l'installation, le programme nécessite des informations d'accès à un service de communication (voir la figure 13).

Dans la zone SIP, entrez les identifiants définis pour le mini central :

- comme nom d'utilisateur, entrez le numéro de poste suivi du symbole @ et de l'adresse IP du mini central;
- comme mot de passe, entrez le mot de passe défini dans le menu « Applications → Interne ».

Immédiatement après la configuration initiale, le softphone est prêt à effectuer le premier appel en interne (voir la figure 14).

En appelant le poste « * 43 », vous accédez au service « echo test » à l'intérieur du central qui lit un message d'instruction, puis répond en sortie au signal qu'il reçoit en entrée.

Ce service est très utile pour vérifier le fonctionnement de la connexion entre le téléphone et le standard. Nous vous recommandons donc de toujours l'utiliser comme premier appel « test », chaque fois que vous installez un nouvel appareil téléphonique.

En installant plusieurs téléphones, vous pouvez passer des appels entre des postes et vous exercer à transférer des appels, à mettre en attente et découvrir d'autres services téléphoniques.

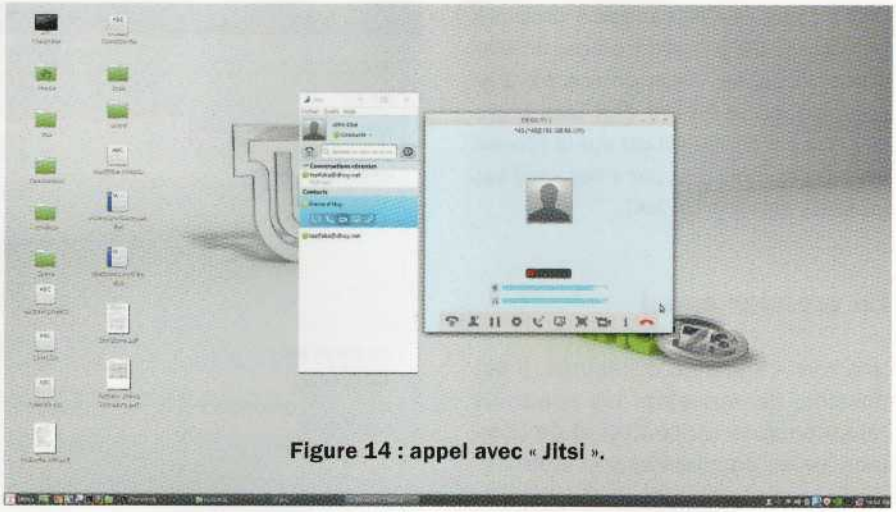


Figure 14 : appel avec « Jitsi ».

Appels externes

Après avoir passé des appels internes, l'étape suivante consiste à connecter notre standard au réseau téléphonique public afin de pouvoir émettre et recevoir des appels depuis/vers l'extérieur.

Le moyen le plus simple et le moins gourmand en matériel est d'utiliser les services offerts par l'un des nombreux opérateurs téléphoniques VoIP.

Chaque opérateur a ses propres particularités, mais tous, après la conclusion du contrat, fournissent les informations d'identification de la connexion via le protocole SIP et attribuent au moins un numéro de téléphone à la connexion.

La **connexion entre le standard et l'opérateur** s'appelle un « **Trunk** ». Elle permet le passage d'un certain nombre d'appels simultanés (voir la figure 15).

La configuration de la jonction ou « Trunk » s'effectue à partir du menu « **Connectivity → Trunk** » dans lequel la partie la plus importante est la configuration des paramètres du protocole SIP (onglet « Setting SIP »).

Les opérateurs fournissent normalement des instructions détaillées pour la configuration d'un « Trunk » sur le logiciel freePBX, y compris les fonctionnalités spéciales.

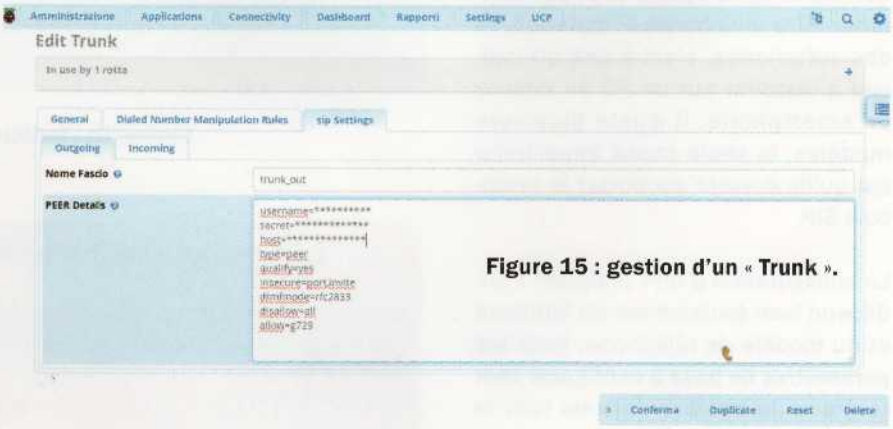


Figure 15 : gestion d'un « Trunk ».

Figure 16 : configuration d'un « Trunk ».

Figure 17 : configuration de la « voie de sortie ».

Dans cet article, nous rapportons une configuration générale valable pour la plupart des opérateurs.

La section « Outgoing » permet de définir les différents paramètres, parmi lesquels les plus importants sont :

- « **username** » et « **secret** » qui sont les informations d'identification et d'authentification du « Trunk » ;
- « **host** » qui est l'adresse du serveur SIP du fournisseur VoIP ;
- « **dtmfmode** » qui est la configuration des tonalités du clavier dans la bande audio ;
- « **disallow** » et « **allow** » qui, respectivement, permettent de désactiver/activer des codecs spécifiques.

Dans la section « **Incoming** » (voir la figure 16), en plus des paramètres similaires à ceux de la section précédente, il est nécessaire de spécifier la chaîne d'enregistrement au format suivant :

<username>:<secret>@<host>:<port>/<extension>

où <username> et <secret> sont les informations d'authentification usuelles, <host> et <port> sont l'adresse et le port (standard 5060) du serveur SIP et <extension> le poste vers lequel sont envoyés les appels entrants, qui est normalement le même que le « username » (nom d'utilisateur).

Une fois qu'au moins un « Trunk » a été configuré, nous pouvons définir une « **voie de sortie** », c'est-à-dire une règle qui indique au standard téléphonique d'envoyer au « Trunk » les appels répondant à certaines exigences.

À l'onglet correspondant « **Route Settings** » (voir la figure 17), allez dans la section générale et donnez un nom à l'itinéraire.

Pour cela, vous devez renseigner la liste des « Trunk » à utiliser en séquence si un appel correspond aux conditions de la « voie de sortie ».

Les « Trunk » sont utilisés les uns après les autres jusqu'à ce que l'un d'entre eux accepte l'appel.

Figure 18 : conditions pour entrer dans le standard.

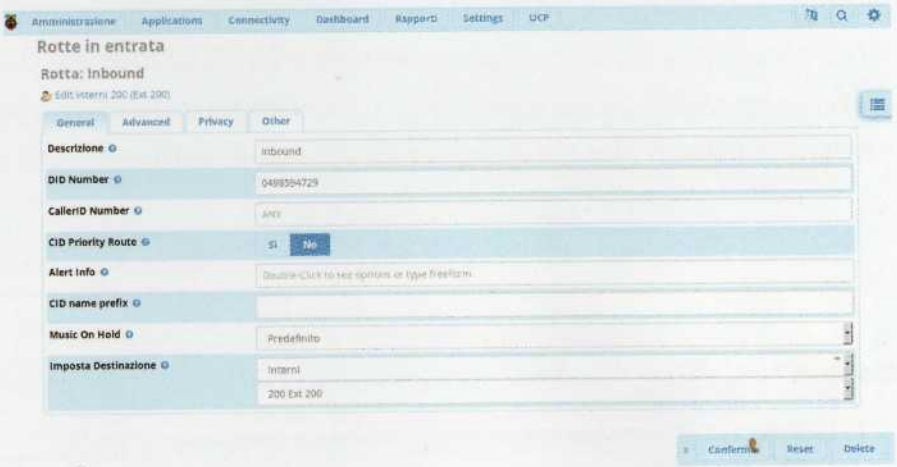


Figure 19 : configuration de l'itinéraire d'entrée.

Ce mécanisme permet de spécifier le « Trunk » à utiliser de préférence et ceux à réserver.

Dans la section « Dial Patterns » (voir la figure 18), nous devons spécifier les conditions auxquelles le numéro appelé doit correspondre pour entrer dans le standard. La condition « [038]. » normalement utilisée pour les appels sortants, indique tous les numéros commençant par 0, 3 ou 8 et suivi de chiffres (le point représente le caractère générique).

Dans les conditions, nous pouvons également spécifier les séquences à ajouter aux chiffres saisis, les préfixes à supprimer avant de transférer l'appel et les règles également des appels internes.

À ce stade, nous devrions pouvoir émettre des appels sortants, mais pour gérer les appels entrants, nous devons ajouter au moins un itinéraire d'entrée. Dans l'exemple présenté en figure 19 (dans la section « In Route »), nous voyons un itinéraire (reporté ci-dessous) très simple qui envoie tous les appels directs au poste 200.

Description : Inbound
DID Number : 0498594723
CallerID Number : ANY
CID Priority Route : NO
Définition de la destination : Interne
200 Ext 200

Conclusion

Dans cet article, nous avons appris les bases de la téléphonie IP (VoIP) et expliqué comment, à l'aide d'un RaspberryPi dans lequel le logiciel approprié doit être installé, il est possible de créer un central VoIP simple mais utile et polyvalent.

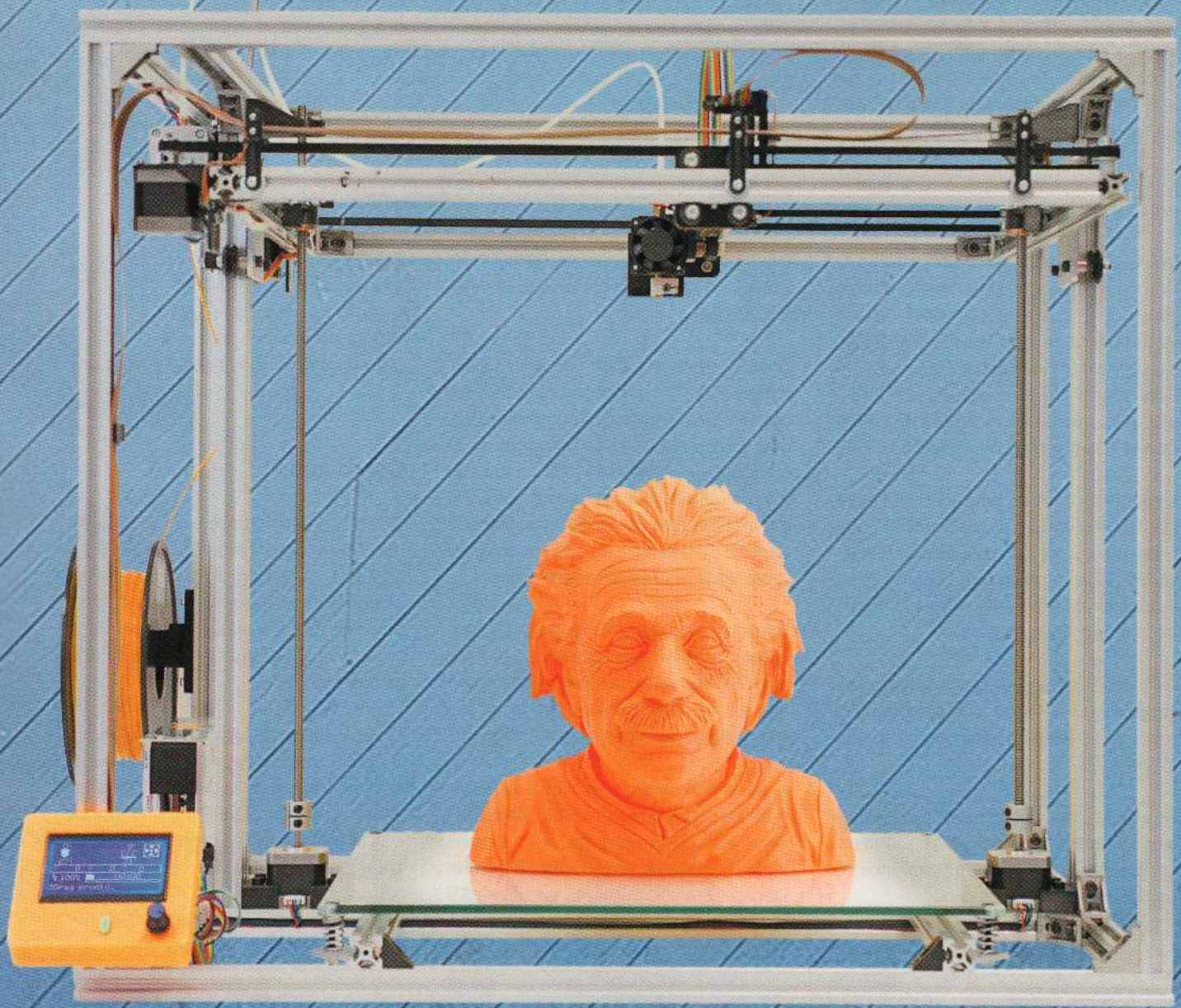
La description du logiciel nous a permis d'établir une configuration de base du mini central afin de commencer à utiliser notre central VoIP dans la plupart des situations qui peuvent être rencontrées dans la pratique.

Dans la prochaine et dernière partie, nous étudierons comment étendre la configuration pour obtenir des fonctionnalités avancées.

La 3DRAG+
Imprimez en grand !
Première partie

de Gabriele Daghetta

Après le succès de l'imprimante 3DRAG décrite dans les numéros 123 à 134, il y a déjà 5 ans, nous vous proposons une version améliorée capable d'imprimer en 3D des pièces de dimensions 400 mm x 400 mm x 400 mm ! Elle intègre toutes les fonctionnalités de la 3DRAG, y compris la caractéristique d'upgrade de la carte de contrôle basée sur un modèle Arduino. Première partie.



Lorsque nous avons créé notre première imprimante 3D, appelée la 3DRAG, nous étions au début de notre expérience en tant que concepteur de machines d'impression en technologie FDM. Cependant, le produit qui en est sorti a été très apprécié par le public et l'est encore aujourd'hui.

Cette imprimante, dérivée de l'originale Sanguinololu du projet RepRap, est basée sur une carte de contrôle considérablement améliorée et utilisant du matériel Arduino.

Grâce à l'évolution constante du firmware Marlin (le cœur du projet RepRap), la 3DRAG est encore aujourd'hui une référence pour les amateurs qui veulent imprimer des objets en PLA ou en ABS, mais aussi pour ceux qui veulent graver directement des circuits imprimés sur une plaque de cuivre nue.

Notre 3DRAG proposait à l'époque une surface d'impression d'objets de 200 mm x 200 mm x 200 mm au maximum, ce qui était déjà considérable il y 5 ans mais qui, pour certaines applications, n'était pas suffisante.

Mais comme nous visions toujours plus haut, nous ne nous sommes pas arrêtés là et nous avons réfléchi à la manière d'augmenter considérablement le volume d'impression. Nous vous proposons donc dans cet article l'imprimante **3DRAG+** qui offre un **volume d'impression 8 fois plus grand** que celui de la 3DRAG originale (64000 cm^3 contre 8000 cm^3). Le résultat de notre travail est la nouvelle imprimante 3DRAG+, que nous présentons dans ces pages et qui a nécessité un long développement et de nombreux tests avant de voir le jour.

De la 3DRAG à la 3DRAG+

Doubler la taille d'impression, soit un volume d'impression multiplié par 8, n'est pas aussi simple que vous pourriez l'imager. Car, même si la solution la plus simple qui viendrait à l'esprit serait de doubler les dimensions des 3 axes de la 3DRAG originale, cela n'est pas vraiment possible.

Si, dans le projet de la 3DRAG+, nous avions simplement agrandi la base du cadre pour permettre un allongement du côté en largeur mais sans toucher à la profondeur et la hauteur, cela n'aurait pas été possible d'un point de vu mécanique.

En voulant aussi augmenter la profondeur, nous avons également rencontré divers problèmes et limitations mécaniques de la 3DRAG originale car sa base ne convenait pas aux contraintes causées par l'impression de grands objets. Il en va de même pour la structure qui supporte l'extrudeuse et l'entraînement de l'axe Z.

En résumé, si nous avions utilisé la structure de base de la 3DRAG originale pour l'adapter à la nouvelle imprimante, nous nous serions retrouvés avec une structure de dimensions doubles, de sorte que pour imprimer un objet de 40 cm de côté, l'imprimante devrait avoir au moins une structure de 80 cm de côté. La raison est que, dans la 3DRAG, l'extrudeuse effectue uniquement un mouvement de montée et de descente.

De plus, la tête qui est fixée sur le plan horizontal impliquerait que la partie la plus extrême de l'objet à imprimer soit déplacée par rapport à la tête (qui se trouve au centre de la machine) d'une distance égale à la largeur maximale de l'axe X et aussi de l'axe Y (en profondeur). Cela n'est pas acceptable, tant pour la taille que pour le coût de fabrication ainsi que le poids qui en découlerait.

De plus, les mouvements d'un plateau d'impression de telles dimensions, qui est en verre trempé pour des raisons de rigidité, n'aurait guère de sens pour des raisons de masses et d'inertie très élevées.

Pour toutes ces raisons, nous avons décidé de passer à une mécanique avec un plateau d'impression fixe, dont la taille ne dépasse pas celle de l'imprimante. C'est donc la tête d'impression qui se déplace le long des axes X et Y, comme la tête a une masse réduite, l'inertie ne pose pas de problèmes particuliers, même aux vitesses d'impression les plus élevées. En fin de compte, nous avons adopté une approche similaire à celle de la 3D Vertex.

En ce qui concerne la 3DRAG originale, le plateau d'impression se déplace le long de barres en acier. Cette solution est valable pour des excursions limitées à 20 cm, mais inacceptable pour des dimensions plus grandes, car si nous avions utilisé des barres de même diamètre mais 2 fois plus grandes (40 cm), elles auraient fléchi vers le centre sous l'effet du poids du plateau plus celui de l'objet en cours d'impression.

L'adoption de guides linéaires sur des rails profilés du fabricant HIWIN (dont les caractéristiques peuvent être consultées à l'adresse https://www.hiwin.de/fr/Produits/Guidages_sur_rail_profile/4263) a été rejetée en raison du coût des guides.

La solution la plus simple consiste à exploiter les emplacements des profilés utilisés pour construire un cadre servant de guide pour chacun des axes.

Pour cette raison, la structure de la 3DRAG+ a été entièrement réalisée avec des profilés en aluminium de type « V-slots » qui ajoutent une nouvelle fonctionnalité par rapport aux profilés traditionnels de type « T-slot ». Le « **V-Slot** » est un **profilé en aluminium de haute qualité** avec une rainure en forme de « **V** » extrêmement lisse sur ses 4 faces.

Grâce à la rainure en « V », ils permettent de réaliser des systèmes mécaniques ayant des mouvements linéaires.

Ces profilés utilisent principalement des poulies coulissantes spéciales (V-Wheel) généralement réalisées avec du POM (polyoxyméthylène), elles se caractérisent par une grande précision et une longue durée de vie.

Ces profilés, que nous avons assemblés avec des angles en aluminium moulé sous pression, ont permis de créer une structure portante à la fois robuste, légère et fonctionnelle (et économique) et qui ne nécessite ni de fixations ni de plaques de renfort malgré les dimensions du cadre très respectables (690 mm x 630 mm x 610 mm).

Le support de l'axe Z se déplace verticalement grâce à deux moteurs pas à pas connectés en parallèle au même driver (3DDRIVER) qui actionnent à leur tour deux tiges filetées en acier inoxydable avec un pas de 2 mm. Cet ensemble comporte un cadre qui permet le mouvement de la tête d'impression selon les directions X et Y.

La tête d'impression, au moyen de l'axe Z, est éloignée ou rapprochée du plateau d'impression, ce dernier étant intégré à la structure. Cette solution permet de simplifier considérablement le châssis, mais surtout de maintenir le centre de gravité de toute la structure vers le bas, ce qui est très utile car le plateau d'impression qui est en verre trempé de 6 mm d'épaisseur pèse environ 3 kg !

Le châssis de l'axe Z est guidé verticalement grâce à deux chariots latéraux équipés de poulies qui coulissent dans les encoches des montants, chacun étant constitué d'un profilé en aluminium de 20 mm x 40 mm.

Le système utilisé pour déplacer l'extrudeuse s'appelle le « Core XY ». L'avantage du « Core XY » est que les moteurs ne se déplacent pas et donc la partie mobile a une masse beaucoup plus faible et donc une inertie plus faible.

Ainsi la vitesse et l'accélération peuvent être augmentées. L'autre intérêt est que l'objet qui est imprimé ne se déplace

Caractéristiques techniques de la 3DRAG+

- Dimensions d'impression : **400 mm x 400 mm x 400 mm** ;
- Matériels utilisables : **ABS** et **PLA** ;
- Mouvements : technologie de type Core XY ;
- **Diamètre du filament** : **1,75 mm** ;
- Diamètre de la buse fournie : 0,4 mm ;
- Diamètre des buses en option : de 0,3 mm à 0,8 mm ;
- Vitesse maximale d'impression : 350 mm/s ;
- **Plateau d'impression fixe en verre trempé de 6 mm** ;
- **5 moteurs** pas à pas (200 pas) ;
- Structure : profilés en aluminium ;
- Résolution mécanique de l'axe Z : 0,625 microns ;
- Résolution mécanique des axes X et Y : 0,0125 mm (12 microns) ;
- Dimensions hors tout : 690 mm x 630 mm x 610 mm ;
- Firmware : Marlin personnalisé pour 3DRAG+ ;
- Logiciel : compatible avec le logiciel Open Source RepetierHost ;
- Systèmes d'exploitation pris en charge par RepetierHost : Windows, Mac OS et Linux.

pas non plus, c'est le deuxième point important pour pouvoir imprimer des objets volumineux.

Le mécanisme comporte deux moteurs qui fonctionnent simultanément pour le positionnement « X » et « Y » de la tête d'impression (à l'aide de 2 courroies de type GT2 de 6 mm avec un pas de 2 mm).

L'utilisation de deux courroies au lieu d'une courroie très longue réduit les erreurs de positionnement, car une seule courroie longue a tendance à se déformer.

Dans le cas de deux courroies plus courtes, le phénomène est moins important dans le temps. La seule exigence est que les deux courroies doivent avoir la même tension.

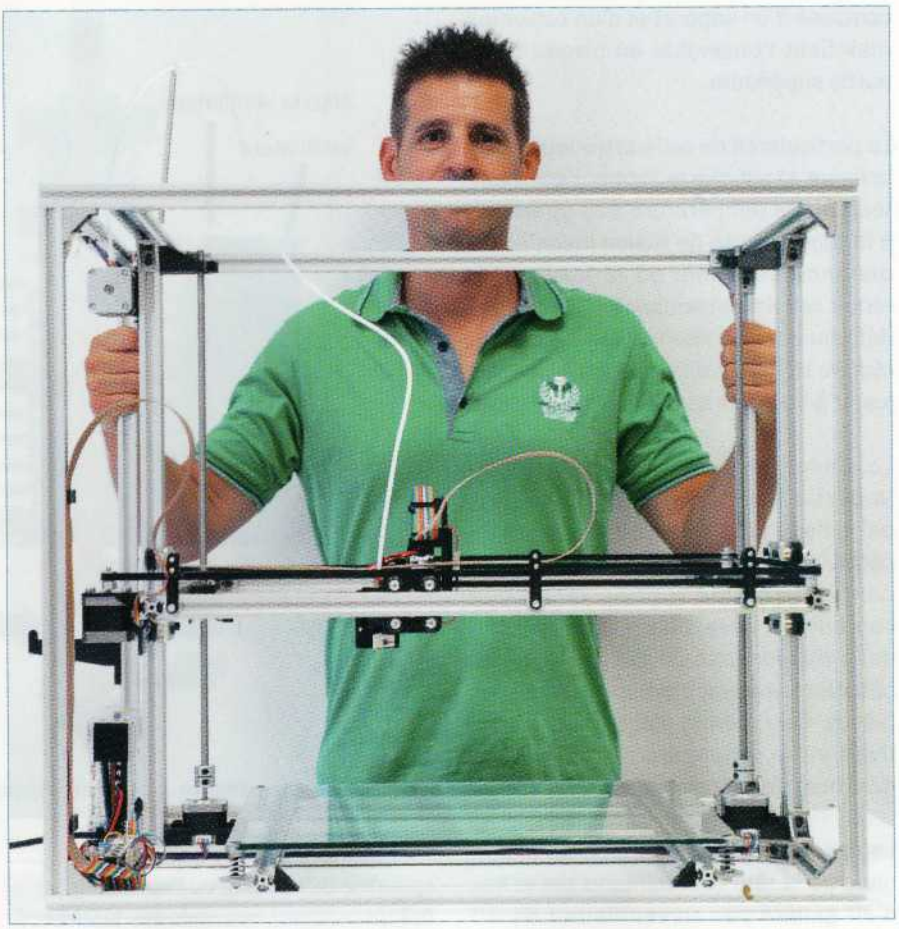
Grâce à cette technologie et au système d'alimentation de la matière (ABS ou PLA) où le moteur qui pousse le fil est fixé au châssis, **la matière atteint la tête d'impression à travers un câble Bowden** (c'est un système de transmission souple du mouvement). Il y a donc moins de masses en mouvement, ce qui permet une plus grande précision.

Le fait que la tête d'impression se déplace selon les directions « X » et « Y » permet, par rapport au système cartésien « Z-head » (celui de la 3DRAG originale), d'imprimer plus rapidement et d'obtenir une plus grande précision (l'inertie est plus faible).

Une autre différence entre la 3DRAG+ et la 3DRAG réside dans le fait que le moteur de l'axe « Y » est monté sur le chariot de l'axe des « X », alors que dans la 3DRAG+ les moteurs des axes « X » et « Y » sont fixés sur le châssis.

De plus, dans la 3DRAG le plateau d'impression se déplace, ce qui, en raison du mouvement des masses non négligeables (plateau + objet imprimé), peut entraîner une perte de vitesse et surtout des vibrations lors de l'impression de très grands objets, ce qui affecte la qualité de l'objet imprimé.

En ce qui concerne la 3DRAG+, le déplacement minimum de l'axe Z (c'est-à-dire la résolution) pour chaque micro-pas, est égale à 0,000625 mm.



Cette mesure est déterminée par : Z = pas de vis / 3200 où le pas de vis est celui de la tige filetée qui agit comme un câble et qui est entraînée par le moteur pas à pas, dans notre cas le pas de vis est égal à 2 mm.

Le micro-pas, comme pour la 3DRAG, est le même. Il est possible de contrôler des fractions de pas, jusqu'à 1/16 de micro-pas. Les moteurs pas à pas étant des NEMA 17 de 200 pas/tour, nous obtenons un nombre maximum de micro-pas égal à 200 x 16 = 3200.

L'extrudeuse

La 3DRAG+ présente également des innovations au niveau de l'extrudeuse (tête d'impression), qui est une tête d'impression de type « **E3D V6** » très légère et prévue pour des **filaments de 1,75 mm**. Cela permet de réduire considérablement les masses en mouvement et donc d'imprimer plus rapidement.

En outre, **l'extrudeuse prend en charge des buses de différents diamètres**, à partir de 0,3 mm pour ceux qui ont besoin d'impressions très détaillées, jusqu'à 0,8 mm pour ceux qui souhaitent réduire les temps d'impression.

L'allègement de la tête est obtenu, comme nous l'avons déjà mentionné, par le fait que l'extrudeuse de la 3DRAG comportait également le moteur d'entraînement du filament de PLA (ou ABS). Ici, la tête d'impression est séparée du mécanisme qui est fixé au cadre comme pour la 3DVERTEX.

Dans le premier cas, le fil est poussé directement dans la tête d'impression, qui le fait fondre et l'extrude sur le plateau d'impression, tandis que dans le second le filament atteint la tête d'impression (en utilisant une technique qui est très légère) à travers un tube en téflon d'un diamètre interne de 2 mm.

La tête d'impression utilisée dans la 3DRAG+ est prête à l'emploi et est produite par la société E3D (http://e3d-online.com). Le modèle que nous avons choisi est le « E3D V6 ». Cette tête d'impression a des dimensions particulièrement réduites, mais nécessite un système de fixation un peu complexe,

composé d'un support et d'un collier qui maintient l'ensemble au niveau de la partie supérieure.

La particularité de cette extrudeuse (voir la figure 1) est que le filament est maintenu à une température très inférieure à la température de fusion jusqu'à une distance de 2 mm de la buse. Cela réduit considérablement les inévitables déformations du filament qui pourraient réduire la vitesse d'impression ou même jusqu'à obstruer complètement la buse.

Le goulot d'étranglement situé dans la partie inférieure de l'extrudeuse, qui relie le bloc chauffant au reste du corps, représente la solution technique pour obtenir une rupture thermique convenable, évitant ainsi qu'une partie de la chaleur générée par la résistance chauffante ne soit dissipée par d'autres pièces. Cela permet d'obtenir une réactivité et un rendement élevé avec un minimum de pertes d'énergie.

Le refroidissement du corps en aluminium de la tête est maintenu par le flux d'air généré par un ventilateur de 40 mm x 40 mm (12 VDC) directement fixé à l'aide d'un clip spécial (fabriqué en PLA avec une imprimante 3D). Il est ainsi possible d'atteindre des températures optimales pour l'extrusion de matière de type PLA et ABS (250 °C).

L'extrudeuse peut supporter différentes buses en laiton avec un diamètre variant de 0,3 à 0,8 mm. Le contrôle de la température s'effectue à l'aide d'une thermistance CTN.

Pour définir le PID, nous utilisons une fonction présente dans Marlin (la version que nous utilisons est la 1.1.0-RC8) qui permet d'effectuer l'autotuning (http://reprap.org/wiki/ PID_Tuning).

Il est ainsi possible de déterminer les trois valeurs à affecter aux paramètres **Kp**, **Ki** et **Kd**.

Le terme PID veut dire « **Proportional Integral Derivative** », ce sont 3 paramètres de contrôle. Dans une imprimante 3D, ces paramètres servent à gérer la montée en température des éléments chauffants, comme par exemple le plateau ou les buses. Ils sont utilisés dans un PID Controller.

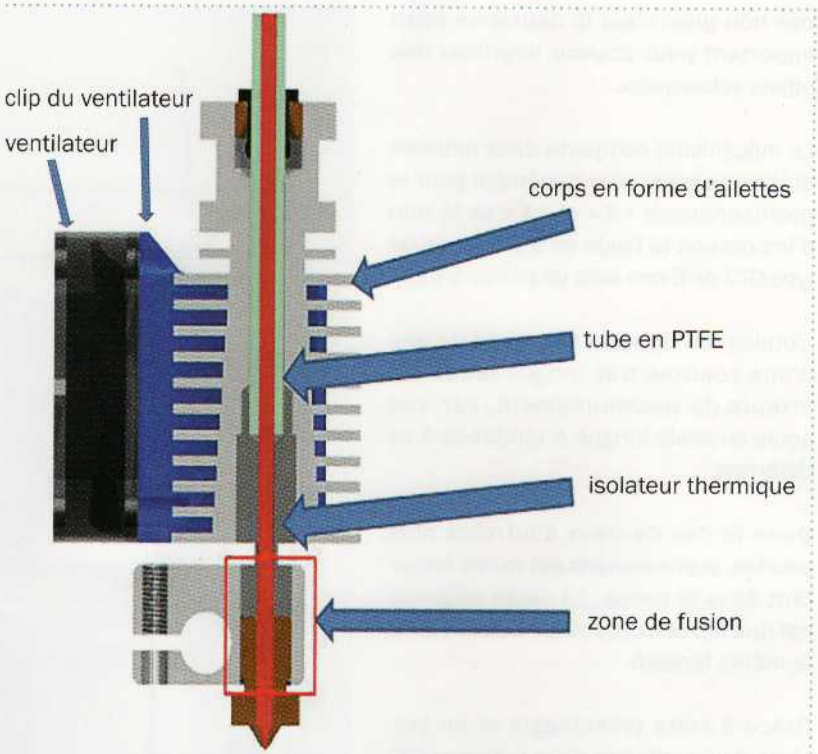


Figure 1 : vue en coupe de la tête d'impression.

Le terme autotuning ou PID Tuning est le fait de modifier ces réglages pour accélérer cette montée en température.

Pour faire cela, il suffit de lancer le logiciel Marlin et de connecter l'imprimante à l'ordinateur.

Ensuite, une fois la communication établie avec l'imprimante, lancez la commande suivante dans Marlin pour calibrer la buse en changeant le terme « Sxxx » par la température à laquelle vous avez l'habitude d'imprimer et le terme « Cy » par le nombre de tests que vous voulez effectuer (nos valeurs sont S250 et C5) :

```
M303 Sxxx Cy
```

Dans la fenêtre de terminal de Marlin, après avoir lancé la commande, vous devriez voir les lignes suivantes qui peuvent varier :

```
bias: 92 d: 92 min: 246.56 max: 253.75
Ku: 32.59 Tu: 54.92
Clasic PID
Kp: 21.58
Ki: 0.74
Kd: 137.31
PID Autotune finished ! Place the Kp, Ki and Kd constants in the configuration.h
```

Notez les valeurs Kp, Ki et Kd.

Il se peut que l'imprimante se déconnecte, auquel cas, débranchez-la puis rebranchez la, puis reconnectez-vous.

Dans la fenêtre de terminal, remplacez les termes par vos valeurs :

```
M301 P23.11 I1.36 D98.44
```

Puis enregistrez avec la commande :

```
M500
```

Le PID de la buse est réglé, la même procédure est utilisée pour le plateau chauffant. Ce dernier dispose d'un verre spécifiquement sélectionné en raison des grandes dimensions, cela afin d'éviter des flexions lors de l'impression.

Il est nécessaire d'imprimer un objet de grandes dimensions avec une certaine rigidité. Le verre trempé de 6 mm d'épaisseur s'est avéré être un bon compromis et a donné des résultats satisfaisants même lorsqu'il est soumis à des contraintes thermiques.

L'utilisation de métal aurait impliqué une masse plus importante de l'extrudeuse pour obtenir la même rigidité.

L'électronique

La **carte de contrôle de l'imprimante 3DRAG+ est identique** à celle de la **3DRAG**. Il s'agit de la carte **3DCONTR-DRIVER** commercialisée par Comelec (www.comelec.fr).

Elle nécessite l'utilisation d'un relais statique SSR pour la gestion de l'élément chauffant du plateau, afin d'éviter la surchauffe des contacts d'un relais classique et du connecteur d'alimentation situé sur la carte.

En fait, compte tenu de la surface de base quadruplée, un élément chauffant beaucoup plus puissant est nécessaire. Il consomme donc un courant beaucoup plus important qui pourrait endommager le MOSFET (HEATER2) qui n'a pas de dissipateur thermique ainsi que le circuit imprimé. La puissance dépasse 200 W !

Quant au câblage, il est sensiblement identique à celui de la 3DRAG avec le contrôleur de dernière génération. Cependant, il y a en supplément un deuxième moteur pas à pas de type Nema 17 sur l'axe Z, car le plateau d'impression se déplace en hauteur à l'aide de deux tiges filetées entraînées chacune par son propre moteur (les deux sont synchronisés, car connectés en parallèle).

La carte contrôleur, qui est **décrite de manière complète dans le numéro 130 d'Electronique et Loisirs Magazine**, est basée sur un microcontrôleur ATmega 2560 et partage en grande partie le matériel avec la carte RAMPS (RepRap Arduino Mega Pololu Shield).

Le processeur ATmega 2560 dispose d'une mémoire programme de 256 Ko et fonctionne à une fréquence de 16 MHz. Il est donc assez puissant et dispose de mémoire suffisante pour contenir le firmware Marlin que nous avons modifié pour contrôler l'imprimante 3DRAG+.

Notre carte peut être programmée directement à partir de l'environnement IDE d'Arduino. Elle dispose d'une connexion USB pour la connecter à un ordinateur via un câble classique « mini USB ». Le même port permet le contrôle de l'impression à partir de l'ordinateur.

La carte permet le montage de quatre modules drivers pour moteurs pas à pas (3DDRIVER), car nous avons un moteur pour chacun des axes X et Y, deux en parallèle pour l'axe Z. Un cinquième moteur contrôle l'engrenage qui pousse le fil de matière plastique à travers le Bowden, à l'intérieur de l'extrudeuse.

La carte dispose également de trois MOSFETS qui contrôlent deux éléments chauffants (un pour le plateau et l'autre pour l'extrudeuse) et un ventilateur basse tension.

De plus, elle dispose d'un convertisseur USB/série qui assure la communication entre l'ATmega avec l'ordinateur et d'une alimentation qui fournit l'énergie nécessaire à l'ensemble.

Les modules drivers pour les moteurs pas à pas sont aussi disponibles auprès de Comelec (3DDRIVER). Ces modules sont constitués d'un circuit intégré Allegro A4988, très polyvalent et qui peut être configuré pour définir à la fois le sens de rotation de l'axe du moteur et le nombre de degrés que l'axe doit effectuer lors de la réception d'une commande.

En d'autres termes, nous pouvons décider si, lorsque nous envoyons une commande, le module doit faire tourner l'axe d'un tour ou 1/2, 1/4, 1/8 et 1/16, en fonction du la précision que nous voulons obtenir. Par conséquent, ils prennent en charge le mode « micropas ».

Le moteur se déplace à chaque impulsion que l'ATmega envoie sur l'entrée « / STEP » du module, selon la configuration de MS1, MS2, MS3. Dans notre cas, nous avons pris le facteur de division maximal, c'est-à-dire que pour un pas, 16 impulsions sont nécessaires.

Avec notre moteur NEMA 17, un tour complet nécessite 3200 impulsions du microcontrôleur (200 pas).

Notez que sur chaque ligne « /STEP » des modules, un transistor NPN est connecté en « parallèle » afin d'amplifier le courant pour piloter une LED. Elle clignote de manière analogue aux impulsions de commandes provenant du microcontrôleur, ceci afin de surveiller le fonctionnement.

Si lors d'une impression, un moteur ne tourne pas malgré les clignotements de la LED correspondante, cela signifie que le problème se situe au niveau du module ou du moteur ou encore au niveau du câblage.

Les LED clignotent à la même fréquence que celles des impulsions de commande. Il est possible d'apercevoir le clignotement uniquement lorsque les moteurs fonctionnent à une vitesse très réduite, car à 25 Hz l'œil humain voit une LED constamment allumée.

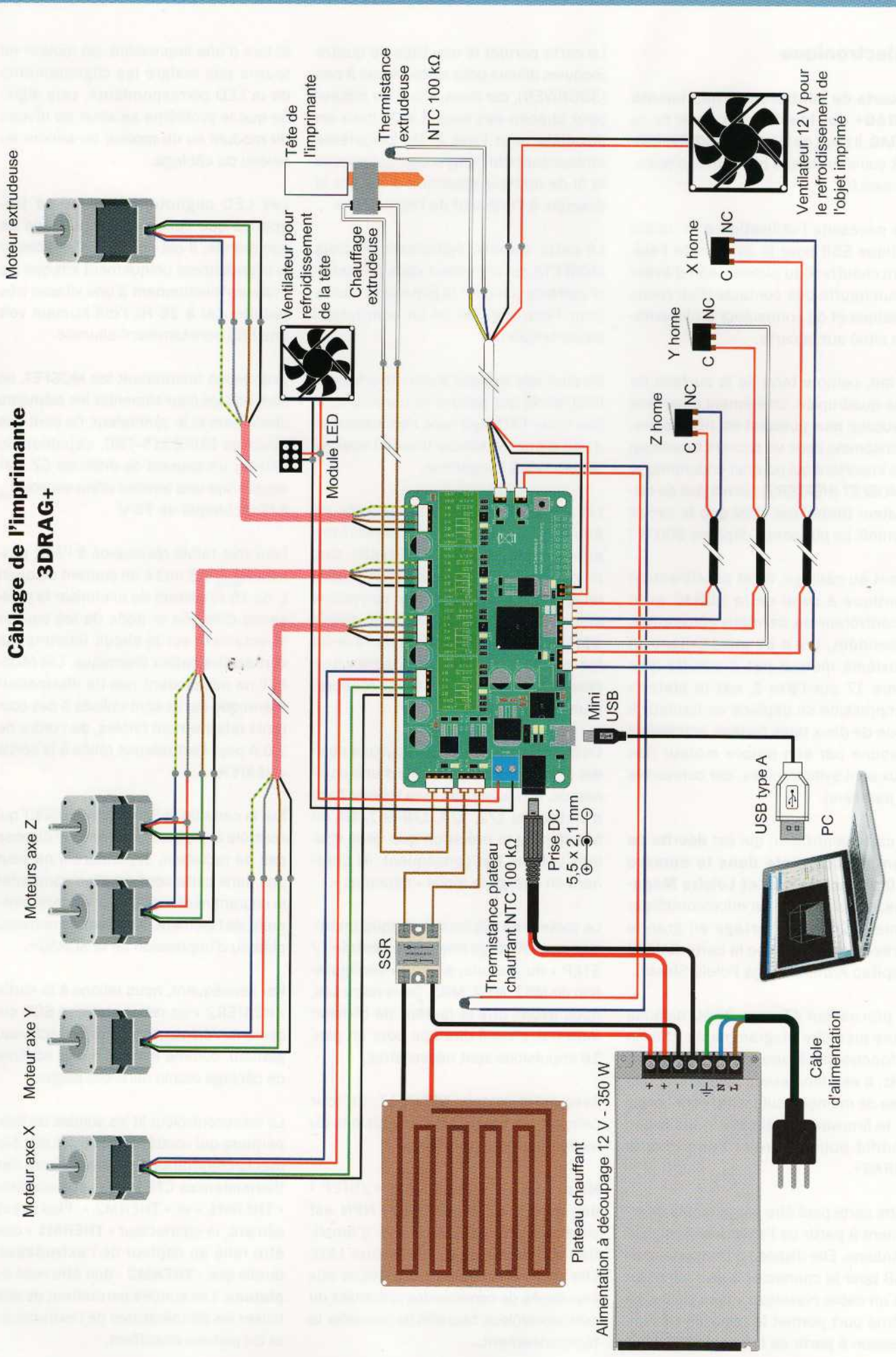
Examinons maintenant les MOSFET, ils sont utilisés pour alimenter les éléments chauffants et le ventilateur. Ce sont des modèles BUK6215-75C, capables de délivrer un courant de drain de 57 A et supportant une tension drain-source V_{DS} à l'état bloqué de 75 V.

Leur très faible résistance à l'état passant R_{DSON} (15 mΩ à un courant de drain I_D de 15 A) permet de minimiser la puissance dissipée et donc de les souder directement sur le circuit imprimé qui sert de dissipateur thermique. Les MOSFET ne nécessitent pas de dissipateur thermique car ils sont utilisés à des courants relativement faibles, de l'ordre de 2,5 A pour l'extrudeuse reliée à la sortie « HEATER1 ».

Sur la carte de la 3DRAG, le MOSFET qui contrôle le plateau chauffant ne dispose pas de radiateur, cependant il ne peut pas dans cette configuration commuter le courant nécessaire pour le fonctionnement de l'élément chauffant du nouveau plateau d'impression de la 3DRAG+.

Par conséquent, nous relions à la sortie « HEATER2 » un relais statique SSR qui alimente l'élément chauffant du nouveau plateau, comme indiqué sur le schéma de câblage visible dans ces pages.

Le microcontrôleur lit les sondes de température qui contrôlent l'activité des éléments chauffants. Ces sondes sont des thermistances CTN reliées aux contacts « THERM1 » et « THERM2 ». Plus précisément, le connecteur « **THERM1** » doit être relié au capteur de l'**extrudeuse**, tandis que « **THERM2** » doit être relié au **plateau**. Les sondes permettent de stabiliser les températures de l'extrudeuse et du plateau chauffant.


Câblage de l'imprimante 3DRAG+
Moteur extrudeuse
Tête de l'imprimante
Thermistance extrudeuse NTC 100 kΩ
Ventilateur 12 V pour le refroidissement de l'objet imprimé
Ventilateur pour refroidissement de la tête
Chauffage extrudeuse
X home
Y home
Z home
NC
C
Module LED
Mini USB
USB type A
PC
Moteurs axe Z
Moteur axe Y
Moteur axe X
Prise DC 5,5 x 2,1 mm
Thermistance plateau chauffant NTC 100 kΩ
SSR
Plateau chauffant
Alimentation à découpage 12 V - 350 W
Câble d'alimentation

Sans ce retour (feedback), même en fonctionnant à courant constant qui correspondrait à une certaine température, les variations de l'environnement pourraient altérer la température à laquelle est chauffée la matière plastique à extruder, avec des problèmes évidents de dépôt et d'impression de l'objet.

En mesurant la température de l'extrudeuse à travers la tension présente sur l'entrée « THERM1 », le microcontrôleur vérifie la valeur de consigne.

Si la température dépasse la valeur de consigne, le microcontrôleur réduit le courant au niveau de l'élément chauffant de l'extrudeuse, dans le cas contraire il l'augmente.

Il en va de même pour l'élément chauffant du plateau d'impression, qui est en option et qui n'affecte pas le fonctionnement de l'imprimante s'il n'est pas présent.

En effet, le firmware de base désactive le contrôle de température du plateau d'impression, ce qui permet de lancer une impression même si le plateau n'a pas atteint la température de consigne. Les thermistances CTN utilisées pour détecter la température de l'extrudeuse et du plateau chauffant sont des modèles 100 kΩ à 25 °C.

Les transistors sont tous pilotés par un signal PWM afin de faire varier la puissance fournie aux moteurs, de contrôler la chaleur générée par les éléments chauffants ainsi que les ventilateurs avec le moins possible de pertes par effet joule. Chaque MOSFET travaille en commutation (ON/OFF).

De même, la sortie qui pilote le relais statique qui commute l'élément chauffant du plateau d'impression, génère un signal de type PWM à très basse fréquence.

Chacune des sorties pilotant les éléments chauffants est munie d'une **LED qui clignote à la même fréquence** que celle du **signal PWM**, soit environ 4 Hz. Cela permet de vérifier visuellement l'état de fonctionnement de ces éléments. Les sorties PWM des ventilateurs ont une fréquence plus élevée et la LED correspondante apparaîtra toujours allumée (persistance rétinienne).

La carte de contrôle lit l'état des commutateurs de fin de course sur les trois axes. Toutes les lignes correspondantes du microcontrôleur sont configurées avec une résistance interne de pull-up activée (par le firmware).

Les interrupteurs de fin de course sont des interrupteurs ou des commutateurs qui, lorsqu'ils sont convenablement positionnés, sont actionnés par le plateau d'impression ou par le support de la tête d'impression. Dans notre cas, ils sont situés sur le cadre.

Lorsqu'un interrupteur de fin de course est activé, le microcontrôleur reçoit des informations sous la forme d'un état logique et arrête le mouvement correspondant en envoyant le chariot du plateau ou en soulevant la tête d'impression.

Notre carte possède trois entrées pour les fins de course : « XSTOP » qui correspond à « YSTOP » et qui coïncide avec « ZSTOP ». Chaque fin de course dispose de trois contacts : « + », « S » et « - ».

Cette solution a été choisie pour connecter à la fois des interrupteurs de fin de course de type électromécanique (micro-interrupteurs) et des détecteurs optiques, composés d'une photodiode et d'une LED infrarouge se faisant face dans une cavité.

Lorsque le chariot arrive, une languette fixée sur celui-ci passe dans la cavité et bloque la lumière qui devrait atteindre la photodiode, cela détermine l'activation de l'interrupteur (optique) de fin de course. Dans notre cas, comme nous utilisons des interrupteurs de fin de course de type électromécanique, nous les connectons entre le points « S » et « - » (masse ou GND).

Par conséquent, la condition d'arrivée de l'interrupteur de fin de course implique un niveau logique bas sur l'entrée correspondante du microcontrôleur.

Pour communiquer avec l'ordinateur, la carte contrôleur utilise un **convertisseur USB/série** de type **FT232RL** (FTDI,) contenant toute la logique nécessaire pour transformer les données au format série TTL en données USB, en triant les données provenant des broches TXD et RXD (broches DP et DM).

Réalisation pratique

Passons maintenant au montage de l'imprimante, elle se compose d'une partie mécanique et d'une partie électronique. Il faut d'abord assembler toute la partie mécanique qui comporte le châssis et toutes les pièces mobiles. Ensuite, il faut assembler la structure portante pour les chariots des axes Z, X et Y, pour le plateau d'impression et pour la partie constituant la tête d'impression.

Examinons succinctement les différentes étapes d'assemblage. Tout d'abord, vous devez monter le squelette du cadre, en assemblant deux profilés en aluminium 20 x 20 de 690 mm et deux de 588 mm.

L'assemblage de ces profilés ainsi que celui de tous les autres profilés utilisés dans l'imprimante s'effectue au moyen d'angles en aluminium moulé sous pression (voir la figure 2), de boulons M5 x 10 TCEI et d'écrous TM5. Cette solution permet une grande rigidité et stabilité du châssis.

Figure 2 : les angles du cadre.

Au milieu, placez un autre profil de 690 mm pour obtenir un ensemble semblable à celui de la figure 3. Ensuite, montez verticalement, dans les angles de la base du châssis, quatre profilés de 570 mm (20 x 20) qui formeront les montants du cadre.

Ensuite, fixez verticalement un profilé supplémentaire de 570 mm (20 x 40), puis prenez deux profilés en aluminium de 690 mm (20 x 20) et deux de 588 mm (identiques à ceux utilisés pour la

base) et fixez-les à la partie supérieure du cadre en utilisant trois angles pour chaque sommet.

Prenez le dernier profilé de 570 mm (20 x 40) et fixez-le au centre du profilé de 588mm (20 x 20) afin d'obtenir un « T » dans le cadre. Vous devez obtenir un résultat semblable à celui de la figure 4.

Maintenant, vous devez réaliser l'ensemble comprenant le cadre de l'axe Z et de ses chariots latéraux utilisés pour permettre les mouvements. Chacun d'eux comporte une plaque en aluminium comme base et doit être fixé au montant correspondant à l'aide des poulies en « V » bloquées par des vis et des écrous de type M5 x 30.

Ensuite, préparez la mécanique qui permet le mouvement horizontal de la tête d'impression, c'est-à-dire le mécanisme X/Y. Il se compose d'un profilé en aluminium de 564 mm (20 x 20) avec les chariots aux extrémités, comme visible en figure 5.

Ce système permet le déplacement le long de l'axe X. En figure 5, il apparaît monté sur le cadre de l'axe Z.

Il ne reste plus qu'à monter le cadre X/Y (c'est-à-dire celui qui se déplace sur l'axe Z) sur les chariots Z précédemment fixés sur les montants latéraux qui servent de guides.

Vous obtenez un résultat semblable à celui de la figure 6, vous apercevez un chariot Z sur le côté extérieur.

Maintenant, vous devez réaliser la partie mécanique permettant d'effectuer le mouvement des axes. Placez tout d'abord deux supports de moteur (voir la figure 7) sur le côté droit de la base et placez entre eux un moteur NEMA17 de 1,5 A (corps court). Faites la même chose pour le côté gauche.

Vissez, de haut en bas, une tige filetée de 500 mm de Ø 8 mm avec un pas de 2 mm jusqu'à ce qu'elle dépasse de 4 à 5 cm de l'emplacement correspondant (pièces jaunes de la figure 7).

Appliquez une rondelle en aluminium sur l'extrémité inférieure de la tige filetée et bloquez-la.

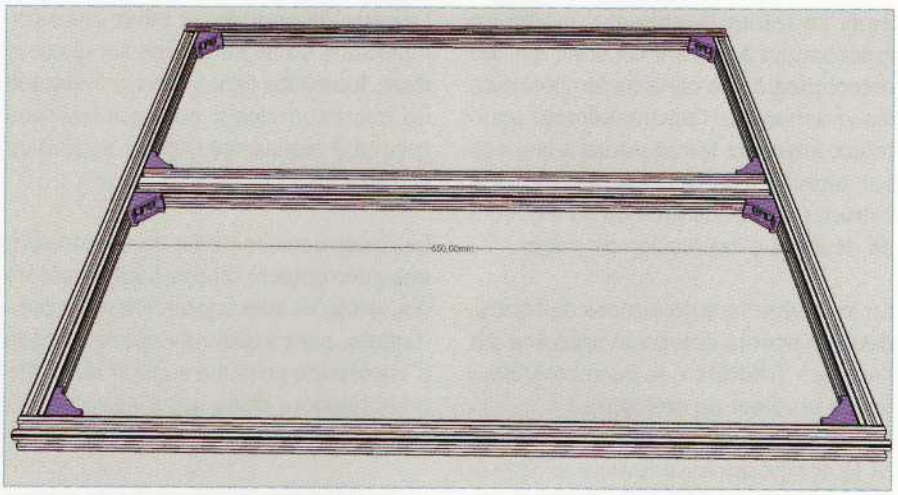


Figure 3 : ici, la base du cadre montée avec le profilé central de renforcement.

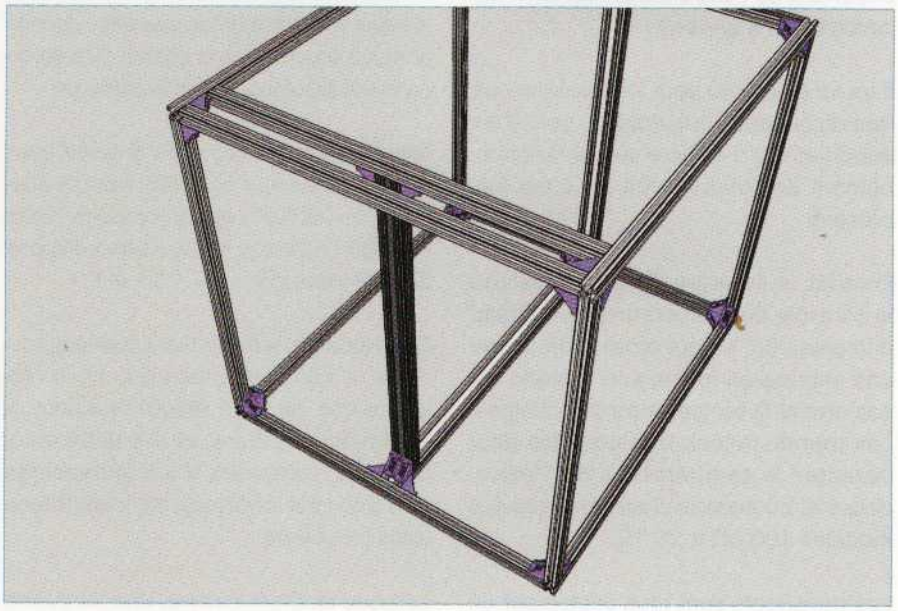


Figure 4 : le cadre terminé.

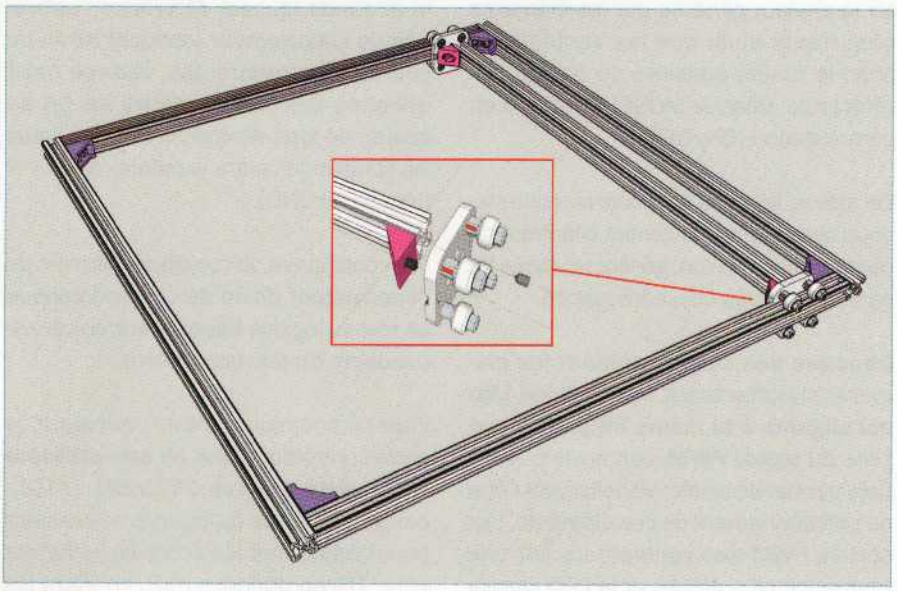


Figure 5 : assemblage de l'armature mobile avec le profilé équipé à chaque extrémité d'un chariot permettant le mouvement sur l'axe Y.

Ensuite, serrez les tiges filetées jusqu'à ce que l'arbre de chaque moteur soit complètement inséré dans son support.

Prenez deux supports de roulements Z et insérez dans chaque support un roulement, fixez-les sur le cadre au même niveau que l'angle supérieur.

Préparez maintenant le chariot de tête, il permettra le mouvement sur l'axe Y.

Prenez la plaque carrée noire appropriée en aluminium anodisé et placez-la sur le guide du chariot X avec les poulies correspondantes, puis vissez la plaque en aluminium de forme spéciale pour permettre le maintien des courroies.

Complétez le chariot Y en installant la deuxième plaque pour la fixation des courroies.

Positionnez les deux moteurs NEMA17 avec les poulies dentées sur le cadre X/Y, afin de permettre les mouvements le long des axes X et Y.

Prenez la courroie crantée et divisez-la en deux longueurs égales (environ 2,5 m chacune), puis positionnez-la dans l'ordre suivant : sur la poulie du moteur droit, puis sur la poulie centrale et ensuite sur la plaque avant du chariot Y à l'aide d'une pince.

Placez le reste de la courroie sur les autres poulies jusqu'à atteindre l'autre plaque où elle sera maintenue à l'aide d'un collier en plastique.

Effectuez la même opération avec l'autre courroie, l'ensemble doit correspondre au résultat de la figure 8.

Passons maintenant à la tête d'impression, qui sera fixée au chariot de tête (voir la figure 9) en fixant sur la partie en forme d'ailette le clip du ventilateur (voir la figure 10).

Les fils doivent passer comme indiqué en figure 14 et doivent être fixés à l'aide d'un collier.

Une fois cela effectué, sur le même clip, vous devez monter un ventilateur de 40 x 40 x 10 mm. Ce dernier doit être orienté de sorte que le flux d'air soit dirigé vers la tête d'impression.

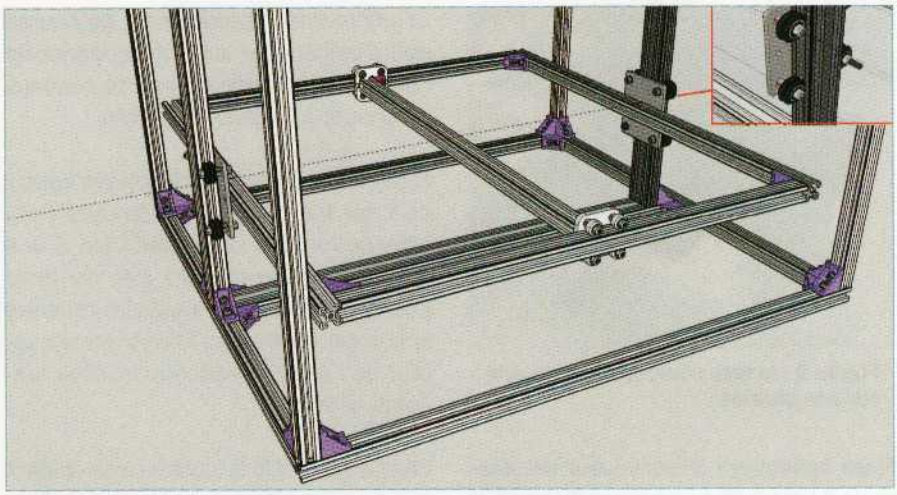


Figure 6 : châssis X/Y inséré dans celui de l'imprimante avec ses chariots de l'axe Z.

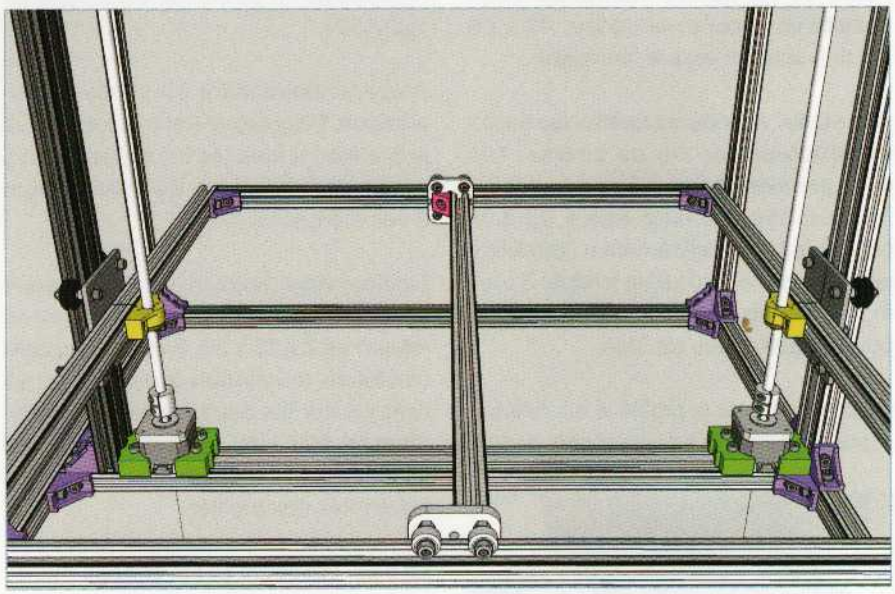


Figure 7 : assemblage du système d'entraînement de l'axe Z.

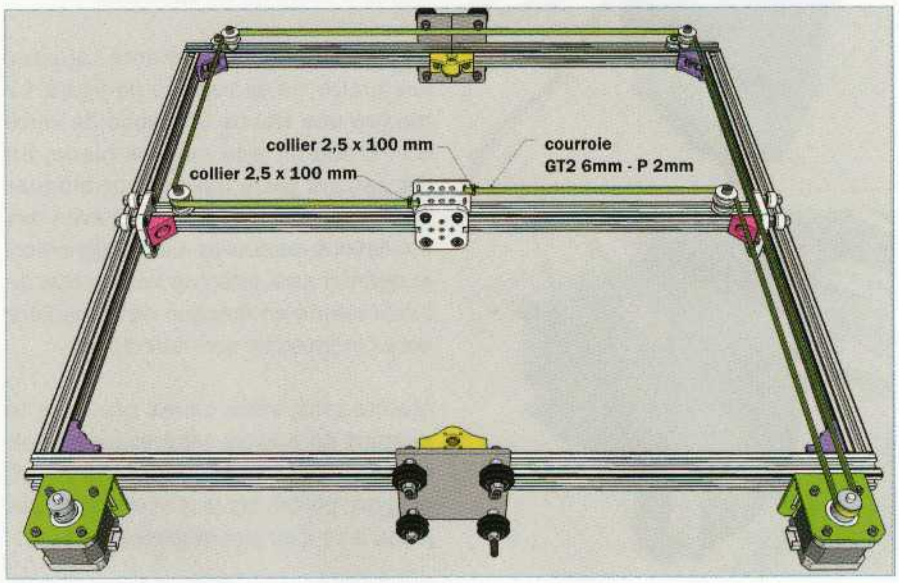


Figure 8 : cadre support de tête avec les moteurs pas à pas montés et les courroies des mouvements X et Y.

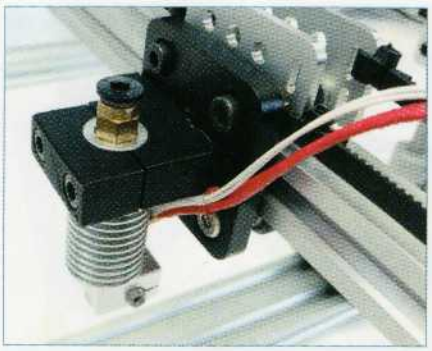
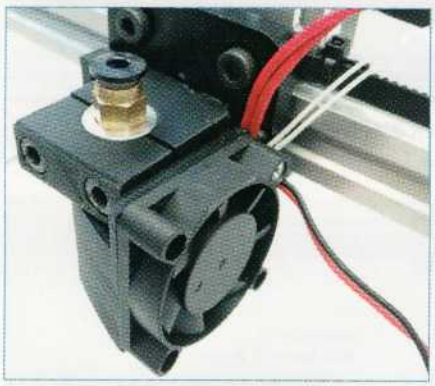


Figure 9 : la tête d'impression montée sur son chariot.

Fixez également le convoyeur en plastique à l'extrémité inférieure du conduit d'air (comme indiqué en figure 10) à l'aide de vis autotaraudeuses. Fixez ensuite un second ventilateur 40 x 40 qui doit souffler vers le convoyeur.

À ce stade, vous devez monter les micro-interrupteurs de fin de course. Les deux premiers doivent être adaptés en raccourcissant la languette à 33 mm, puis en la pliant légèrement à l'extrémité vers le corps. À l'aide d'un foret de 3 mm, élargissez les trous de montage pour pouvoir utiliser des vis 3MA.

Fixez ensuite sur le profilé le micro-interrupteur de l'axe Y à l'aide d'une vis M3.



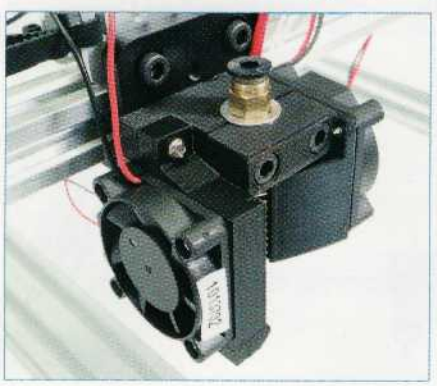


Figure 10 : les ventilateurs et le convoyeur montés sur la tête d'impression.

Le micro-interrupteur de fin de course doit s'enclencher lorsque le chariot de l'axe Y est à environ 42 mm de distance du chariot à l'extrémité gauche.

Fixez sur le profilé le micro-interrupteur de l'axe X de manière à ce qu'il s'enclenche lorsque le chariot X est à une distance d'environ 85 mm du bord interne du profilé sur lequel les moteurs sont montés. Prenez le micro-interrupteur de l'axe Z et raccourcissez la languette à 23 mm.

Fixez-le au profilé de sorte que le contact soit à environ 2 mm de l'angle situé en dessous. Le positionnement des interrupteurs de fin de course est illustré en figure 11.

Passons maintenant au plateau d'impression. Disposez-le (manipulez-le avec précaution !) avec les trous évasés vers le bas et insérez une vis dans chacun d'eux (de bas en haut).

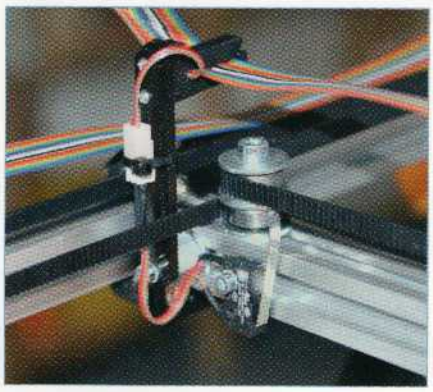
Ensuite, vous devez insérer sur chaque vis une rondelle plate de 5 x 20 et un ressort de 2 x 15 x 26. Prenez les quatre carrés en aluminium à angle droit et fixez-les sur les deux profilés en aluminium de 627 mm, de sorte qu'un côté de la plaque soit à 80 mm de l'une des extrémités des profilés.

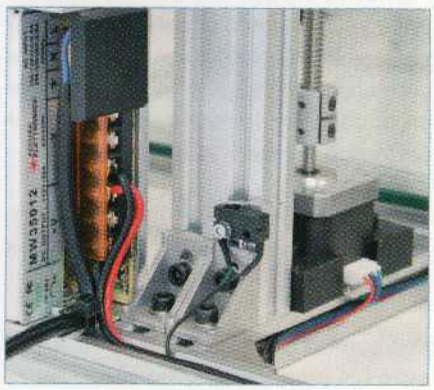
Le bloc, constitué par le plateau monté sur les deux profilés en aluminium, doit être fixé à la base du cadre de l'imprimante à l'aide de vis spéciales.

Une fois l'ensemble monté, ajustez les quatre vis de réglage (la figure 12 montre une vis) de la plaque de verre afin d'obtenir une surface plane, en les serrant juste assez pour bloquer légèrement les ressorts. Cela évite des oscillations excessives. Le réglage précis et définitif sera effectué lors du test de l'imprimante en fonction de la manière dont l'impression sera effectuée.

Maintenant, vous devez préparer le support de bobine. Insérez le bras de support dans la fente appropriée du support, puis faites-le glisser vers le bas jusqu'à ce qu'il soit maintenu en buté.

Vous devez ensuite fixer le support de bobine sur le châssis à environ 150 mm de distance de la base.





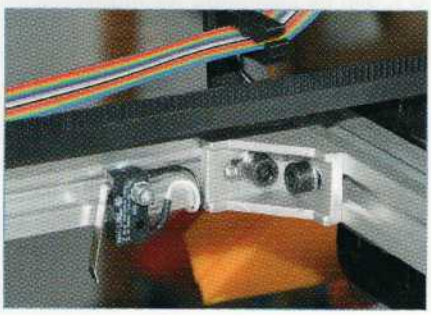


Figure 11 : position des micro-interrupteurs de l'axe Z (milieu), de l'axe X (en bas) et de l'axe Y (en haut).

Passons maintenant à l'assemblage du mécanisme d'alimentation (également appelé Feeder). Fixez au support du moteur NEMA17 de 2,5 A (type long) le support du « bowden » et insérez sur l'axe du moteur le système d'entrainement en prenant soin d'aligner le point de fixation de ce dernier avec la partie plate de la goupille.

Fixez le système d'entrainement sur l'axe du moteur après avoir aligné sa rainure dentée sur les trous de guidage du filament présents sur le support « bowden ».

Insérez le roulement sur la demi-coquille appropriée et montez la demi-coquille correspondante, puis vissez, sur le moteur pas à pas, la vis dépassant du support du roulement.

Maintenant, disposez le guide du filament, c'est-à-dire le « bowden », qui va du moteur à la tête d'impression, et fixez-le au cadre incurvé à l'aide d'un ressort de traction. Ensuite, passez le tube en PTFE, qui dépasse de la tête, dans l'œillet supérieur du ressort, puis fixez-le au profilé supérieur du cadre en l'alignant avec la fente gauche du montant.

Fixez le « bowden » sur le guide du chariot Z (dans la fente gauche) à l'aide des vis et insérez le tube en PTFE provenant de la tête d'impression dans le raccord (enfoncez-le complètement).

Le système d'alimentation de la matière (PLA/ABS) doit correspondre à celui illustré en figure 13.

À ce stade, la mécanique est terminée et vous pouvez passer à l'installation de l'électronique, qui comprend la carte contrôleur et le circuit imprimé pour le câblage de la tête d'impression.

Pour cette imprimante, nous avons prévu de connecter les câbles de la tête et ceux des ventilateurs à un circuit imprimé, afin d'obtenir un câblage plus ordonné. Sur le circuit imprimé, des borniers sont prévus à cet effet ainsi qu'un connecteur de type HE14 avec un détrompeur et un câble plat.

Le câblage de la carte est indiqué dans l'encadré intitulé « Carte de câblage », le tableau indique les connexions entre le connecteur HE14 et les borniers.

Il est donc nécessaire de fixer cette carte, à l'aide d'un support approprié, au profilé central du cadre où la tête est fixée, comme indiqué en figure 14 (où vous apercevez le câblage de la tête d'impression).

La carte contrôleur est fournie déjà assemblée et prête à l'emploi avec le bootloader et le firmware chargés. Pour plus de renseignements, reportez-vous au numéro 130 d'Electronique et Loisirs Magazine.

Notez que le firmware de base est écrit pour fonctionner sans chauffage du plateau d'impression Z qui est toujours actif, grâce à l'instruction « #define DISABLE_Z » réglée sur « false. Cela permet de verrouiller les deux moteurs pendant le fonctionnement afin d'éviter des vibrations.

En ce qui concerne le câblage, après l'avoir fixé au châssis de l'imprimante, n'oubliez pas de connecter les fils des moteurs pas à pas aux connecteurs correspondants à l'aide des câbles plats.

La correspondance entre les moteurs et les connecteurs est la suivante :

- le **moteur de l'axe X** se connecte à **X-MOTOR** ;
- le **moteur de l'axe Y** se connecte à **Y-MOTOR** ;

CÂBLAGE de la carte

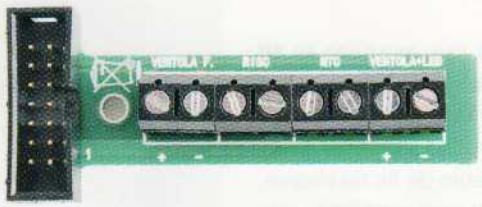


Photo du circuit imprimé avec les borniers et le connecteur HE14 permettant de simplifier le câblage.

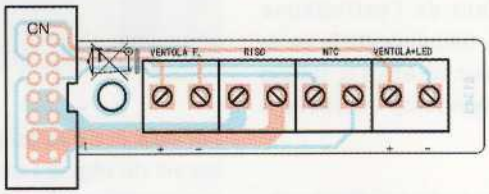


Schéma de câblage, ci-dessous le tableau de correspondance des broches du connecteur HE14 et des borniers.

Broches du connecteur HE 14	Borniers
1, 2, 3, 4	RISC
5, 6, 7, 8	RISC
9	NTC
10	NTC
11	+ ventilateur
12	- ventilateur
13	- LED
14	+ LED

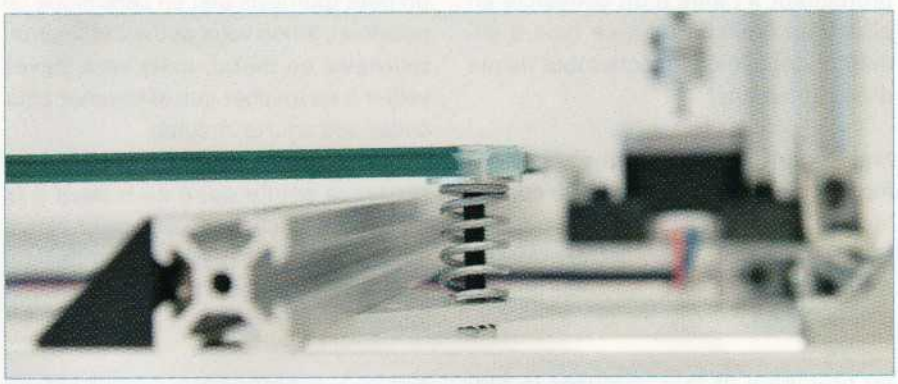


Figure 12 : suspension et vis de réglage de la plaque de verre dans un des coins.

- le moteur qui monte et abaisse la **tête d'impression** se connecte à **Z-MOTOR** ;
- le moteur qui fait **avancer le fil dans l'extrudeuse** se connecte à **E-MOTOR**.

Pour les commutateurs de fin de course, ceux des axes X et Y sont connectés respectivement à XSTOP et YSTOP, tandis que celui du chariot de la tête d'impression doit être connecté à ZSTOP.

L'élément chauffant de l'extrudeuse doit être connecté, peu importe la polarité, aux connecteur HEATER1, tandis que la CTN correspondante est reliée à THERM1.

Fixez l'alimentation en bas à gauche du cadre et connectez les câbles au bornier de la manière suivante :

- fil **jaune/vert** vers la **masse** du bornier ;
- fil **bleu** vers le point **N** du bornier ;
- fil **marron** vers le point **L** du bornier.

Ensuite, connectez la sortie d'alimentation à un câble bicolore rouge/noir muni d'une fiche à insérer dans la prise DC de la carte contrôleur. Le fil rouge doit être connecté à la borne « +V » et le noir à la borne « -V » de l'alimentation.

Le câble secteur doit être fixé à la partie inférieure du module d'alimentation au moyen d'une attache de 2,5 x 200 mm, ce qui le protégera d'éventuelles coupures accidentelles.

Il est préférable d'isoler les bornes soumises à la tension du secteur en les protégeant à l'aide d'un couvercle en plastique (spécifique à ce type d'alimentation) afin d'empêcher tout risque d'électrocution.

Maintenant, l'imprimante est prête à fonctionner. Connectez la carte contrôleur à l'ordinateur à l'aide d'un câble USB A/mini USB, vous pouvez démarrer l'imprimante en effectuant un test d'impression.

Au préalable, il faut disposer le plateau d'impression parfaitement à plat par rapport à la buse, en agissant sur les vis de réglage des ressorts. Utilisez éventuellement, un comparateur pour obtenir une planéité parfaite du plateau. (environ ± 2/10 de mm).

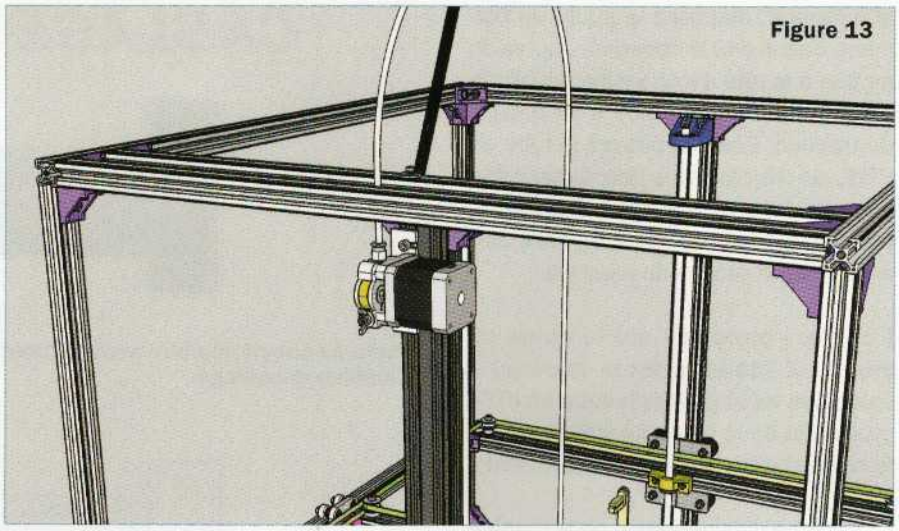


Figure 13

Vérifiez également si les moteurs bougent correctement et dans le bon sens, synchronisez-les les uns avec les autres.

Si la tête ne bouge pas, c'est qu'un micro-interrupteur (X ou Y ou les deux) n'est pas connecté, de sorte que la carte ne détecte pas l'interrupteur de fin de course.

Les réglages

Pour obtenir un fonctionnement optimal de l'imprimante, vous devez régler les drivers de la carte.

La procédure est la suivante : mettez l'imprimante sous tension, munissez-vous d'un multimètre réglé sur le calibre 2 VDC (pleine échelle) ; prenez un petit tournevis plat en céramique (si possible), sinon vous pouvez utiliser un tournevis en métal, mais vous devez veiller à ne toucher que le trimmer pour éviter des courts-circuits.

Mettez la pointe noire du testeur à la masse de la carte et la pointe rouge sur le point test (pad ou pastille) qui se trouve au-dessus du premier circuit intégré driver.

Ajustez le potentiomètre jusqu'à obtenir au multimètre une tension de 0,41 V.

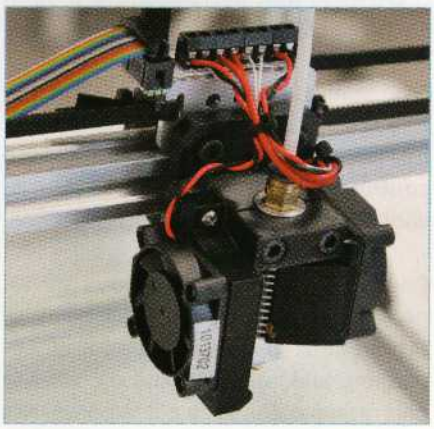


Figure 14 : câblage de la tête d'impression à la carte contenant les borniers.

Faites de même pour les autres drivers, en vous rappelant que l'**axe des Z doit être réglé à 0,65 V** car il doit alimenter deux moteurs en parallèle.

Conclusion

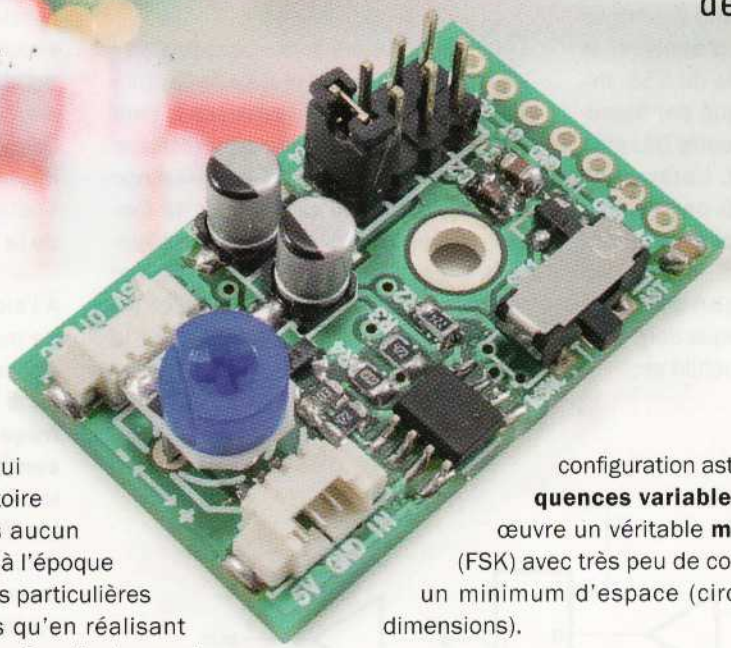
Nous concluons pour l'instant avec la partie mécanique et le réglage de l'imprimante. Dans le prochain numéro, nous aborderons l'installation et la personnalisation des réglages de l'élément chauffant du plateau d'impression.

Nous décrirons les différentes commandes du panneau de contrôle (avec un afficheur graphique) pour l'utilisation de l'imprimante de manière autonome.

Nous modifierons le mode micro-pas par défaut et nous verrons comment intervenir sur le firmware de la carte contrôleur.

CARTES DE PROTOTYPAGE NE555

Nous vous présentons dans cet article une carte de prototypage ou breakout board qui vient compléter la gamme décrite dans les numéros 135 à 138 d'Electronique et Loisirs Magazine. Cette carte permet de mettre en œuvre des circuits astables ou monostables en exploitant le circuit intégré le plus populaire parmi les circuits de type « timer ».



de Boris Landoni

L'un des circuits intégrés qui ont contribué à forger l'histoire de l'électronique est sans aucun doute le **NE555**, car il permettait à l'époque d'implémenter plusieurs fonctions particulières qui ne pouvaient être obtenues qu'en réalisant un circuit spécifique (pour chaque fonction) avec des composants discrets.

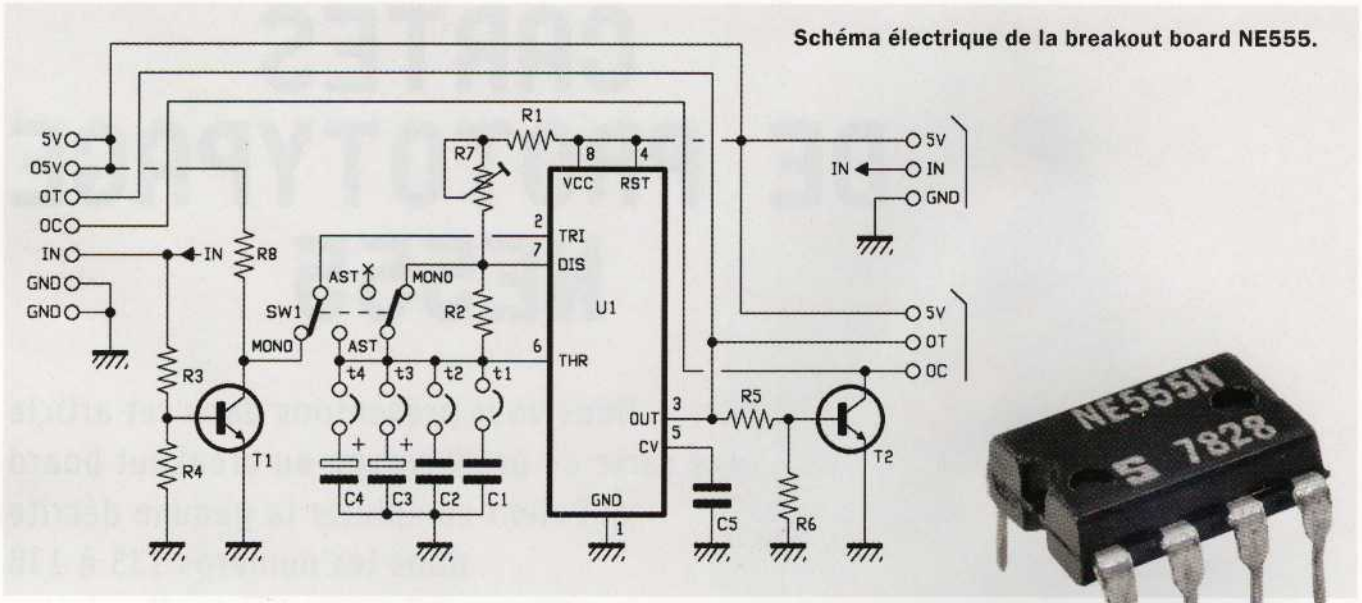
Le **NE555** est un **temporisateur** (timer) qui peut être configuré de diverses manières. Il comporte une paire de comparateurs dont les sorties sont reliées aux entrées « **SET** » et « **RESET** » d'une **bascule RS**, de façon à implémenter des **multivibrateurs astables**, **monostables** ou **bistables**.

Mais ce n'est pas tout, il permet de fournir une « image » de la tension de référence de la paire des comparateurs, et donc la **tension à laquelle la commutation se produit**. Cette caractéristique permet de faire générer au NE555, dans une configuration astable, des **signaux de fréquences variables**, et donc de mettre en œuvre un véritable **modulateur de fréquence** (FSK) avec très peu de composants externes dans un minimum d'espace (circuit imprimé de faibles dimensions).

Au fil des années, le NE555 a également été amélioré (il y a 20 ans, il fonctionnait en mode astable à une fréquence maximale de 200 kHz alors que les modèles actuels atteignent le mégahertz). Etant donné qu'il est toujours d'actualité, nous avons décidé de réaliser une breakout board spécifique au NE555, c'est-à-dire une carte de prototypage au format miniature qui permet de tester ses fonctionnalités et de les intégrer dans d'autres circuits lors de la phase de développement.

La version du 555 utilisée dans ce montage est de type CMS afin d'obtenir un circuit de petite taille, comme nous l'avons

Schéma électrique de la breakout board NE555.



Le circuit intégré NE555 original dans un boîtier DIP de 2 x 4 broches.

fait pour les autres breakout board (reportez-vous aux numéros 135 à 138).

Le circuit intégré NE555

Avant de passer au schéma électrique de la carte, il convient d'analyser le schéma structurel interne du 555. Initialement, il a été fabriqué par Signetics, un fabricant de circuits intégrés des années 1960-1970. L'entreprise a été rachetée en 1975 par Philips, Signetics se retrouve actuellement dans l'entreprise NXP Semiconductors. De nombreux fabricants de semi-conducteurs le fabrique dont NXP, STM (SGS-Thomson), Fairchild etc.

Comme le montre la figure 1, le **NE555 comporte deux comparateurs** ayant en commun un **diviseur de tension multiple** qui polarise l'entrée **inverseuse** (en haut) et l'entrée **non inverseuse** (en bas) d'un amplificateur opérationnel.

Les comparateurs sont « accessibles » depuis l'extérieur au niveau de l'entrée non inverseuse de la partie supérieure (les termes supérieure et inférieure se réfèrent au potentiel de référence qu'elles reçoivent du diviseur de tension multiple, lorsque la partie supérieure reçoit la tension la plus élevée) via la broche 6 (**TRESHOLD**) et au niveau de l'entrée inverseuse via la broche 2 (**TRIGGER**).

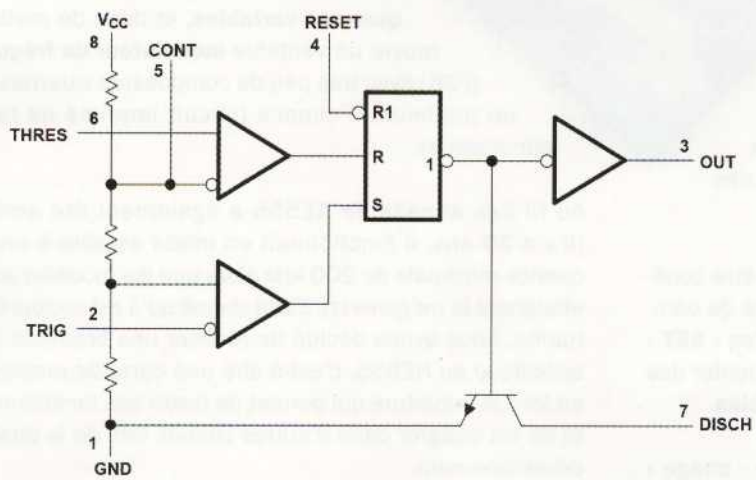


Figure 1 : schéma synoptique interne d'un NE555 valable pour tous les types de boîtiers.

Le nœud entre la première et la seconde résistance (en partant de l'alimentation positive) du diviseur de tension multiple des deux comparateurs (alimentés comme le reste des étages internes du NE555 via la broche 8) est « accessible » de l'extérieur au moyen de la broche 5 (**Control Voltage**).

À l'aide de cette broche, il est possible de modifier les tensions de référence des comparateurs de manière **à contrôler à l'aide d'une tension externe la fréquence de fonctionnement dans la configuration astable** (en effectuant le décalage de fréquence comme dans les VCO c'est-à-dire comme dans les oscillateurs commandés en tension) ou la durée de l'impulsion du circuit en configuration monostable.

Plus précisément, dans la configuration astable, lorsque la tension sur la broche 5 diminue, la période de commutation de la sortie diminue et donc la fréquence générée augmente.

Inversement, si la tension augmente jusqu'à un niveau caractéristique (c'est-à-dire sans aucun circuit connecté à la broche 5), la période de commutation est allongée et donc la fréquence générée est réduite.

En **configuration monostable**, cependant, **la tension modifie la durée des impulsions**, ce qui permet, en pilotant le trigger avec des impulsions de fréquence constante, d'obtenir un **modulateur PWM**.

La sortie du comparateur supérieur est reliée à la broche RESET, et celle du comparateur inférieur à la broche SET d'une bascule de type RS. Il s'agit d'un circuit logique dont la sortie directe (Q) passe au niveau logique haut (correspondant au potentiel de la broche 8) lorsque l'entrée SET est à niveau haut, et inversement la sortie prend un niveau logique bas si l'entrée SET est à un niveau bas.

La sortie adopte un comportement exactement opposé lorsque l'entrée RESET est sollicitée. Si celle-ci est à un niveau logique haut, la bascule est réinitialisée ce qui signifie qu'elle réinitialise l'état de la sortie Q. La bascule a également une sortie complémentaire « /Q » (il faut prononcer Q barre) dont la condition logique est toujours l'inverse de la sortie Q.

Dans le NE555, cette sortie est reliée, à travers une résistance, à la base d'un transistor NPN configuré en collecteur ouvert. L'émetteur étant relié à la masse et le collecteur à la broche **DISCHARGE** (7). Le transistor peut commuter un courant de collecteur de 200 mA lorsqu'il est utilisé en mode « sink » (absorption de courant).

La sortie directe Q est connectée à la broche OUT (3) du circuit intégré par l'intermédiaire d'un buffer interne (étage tampon) de type « push-pull » capable de délivrer un courant maximal de 200 mA.

Le négatif de l'alimentation, c'est-à-dire la masse, sert de référence pour le circuit NE555 et est relié à la broche 1. La broche 4 correspond à l'entrée RESET (réinitialisation de la bascule). Cette broche permet d'imposer de l'extérieur une réinitialisation du circuit en lui appliquant un niveau logique bas.

La logique de fonctionnement du NE555, représenté par le synoptique de la figure 1, peut prendre quatre états différents :

1. La broche « **RESET** » est à un **niveau bas : la bascule est remise à zéro**, le transistor de décharge conduit et la sortie reste à un niveau bas. Aucune autre opération n'est possible ;
2. La **tension** sur la broche « **TRIG** » **est inférieure à 1/3** de celle d'alimentation **VCC** : la bascule est activée (SET) et la **sortie passe à un niveau haut**, le transistor de décharge est **bloqué** ;
3. La **tension** sur la broche « **THRES** » **est supérieure à 2/3** de celle d'alimentation **VCC** : la bascule est remise à zéro (RESET) et **la sortie passe à un niveau logique bas**, le transistor de décharge conduit ;
4. Les **tensions** sur les broches « **THRES** » et « **TRIG** » sont **respectivement inférieure à 2/3 de VCC et supérieure à 1/3 de VCC** : la **bascule conserve son état précédent** ainsi que la sortie et le transistor de décharge.

Schéma électrique de la breakout board NE555

Maintenant que nous avons abordé la description du NE555, nous allons examiner le schéma électrique de la carte de prototypage. Bien évidemment, cette carte permet le fonctionnement du circuit dans les deux configurations, c'est-à-dire comme multivibrateur astable ou monostable.

Le circuit intégré NE555, dénommé U1, est monté en configuration classique. Sa broche 1 est reliée à la masse et sa broche 8 à l'alimentation positive. Sa broche 4 (RESET) est reliée au positif de l'alimentation, car dans notre projet nous ne voulions pas utiliser la fonction de réinitialisation externe.

De même, la fonction « Control Voltage » (broche 5) n'est pas utilisée et, conformément aux instructions du fabricant, nous l'avons connectée à un condensateur de filtrage (C5) relié à la masse.

La sortie « **OUT** » (broche 3) **pilote** le transistor **T2** qui permet d'amplifier le courant de sortie, jusqu'à 500 mA. Le transistor T2 est monté en configuration « **collecteur ouvert** » et peut fonctionner dans des circuits **où la tension ne doit pas dépasser 45 VDC** (sur son collecteur).

Sa **base est polarisée par le diviseur de tension** formé par les résistances **R5** et **R6**, de sorte que lorsque l'état logique de la sortie « **OUT** » est à un **niveau bas**, le transistor **T2** est au repos (non conducteur ou **bloqué**). Inversement, lorsque la broche 3 est à un **niveau haut, T2 devient conducteur** et entre en saturation.

Nous arrivons à la partie la plus intéressante du montage, à savoir les broches 2, 6 et 7, qui si elles sont configurées de manière appropriée, permettent **d'obtenir les deux modes de fonctionnement**, à savoir le **mode monostable** et le **mode bistable**. Sur la carte, ces 2 modes sont gérés par le double inverseur **SW1**.

Pour obtenir le **mode monostable**, l'inverseur **SW1** doit être positionné sur « **MONO** », ainsi la broche « DIS » est reliée à plusieurs condensateurs sélectionnables au moyen de cavaliers nommés t1, t2, t3, t4.

Le double inverseur SW1 court-circuite la résistance R2, utilisée dans la configuration astable, et relie entre elles les broches 6 et 7.

L'autre moitié du double inverseur SW1 relie la broche 2 au collecteur du transistor T1, qui fonctionne comme un interrupteur statique afin de commuter l'entrée de déclenchement « **TRIGGER** » du NE555.

La base du transistor T1, à travers le diviseur de tension constitué par les résistances R3 et R4, est polarisée par la tension qui est appliquée sur l'entrée « IN ». Cette dernière est donc l'**entrée de déclenchement** (trigger), **qui est active uniquement en mode monostable** (puisque dans la position « AST » SW1 isole T1 du reste du montage) et qui est conçue pour recevoir des impulsions dont la tension a une valeur comprise entre 3 V et 12 V (typiquement au niveau TTL 0 V / 5 V).

Le fonctionnement en mode monostable s'explique de la manière suivante : le condensateur de temporisation se trouve initialement déchargé

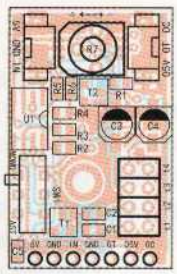
(celui connecté par l'un des cavaliers entre la broche 6 et la masse ...).

En alimentant le circuit, la broche « **THR** » est **initialement à 0 V** (puisque le condensateur est déchargé) et donc, sous l'effet de la polarisation interne due au diviseur de tension multiple, la sortie du comparateur supérieur est ramenée à un niveau logique bas (0 V) et **n'a pas d'effet sur l'état de la bascule**.

La broche « TRI », dans les conditions de repos (pas de tension sur l'entrée « IN »), est au niveau logique imposé par la polarisation interne du circuit intégré.

Ainsi, le deuxième comparateur a sa sortie à un niveau bas, car la polarisation interne met l'entrée non inverseuse à un niveau bas, et donc la sortie du NE555 prend ce niveau logique.

Plan de montage de la breakout board NE555



Plan de câblage des composants de la breakout board NE555.

Photo de l'un de nos prototypes de la breakout board NE555.

En **appliquant une tension** sur l'entrée « **IN** » de la carte, le transistor **T1 passe en saturation** et son collecteur amène la broche 2 du 555 à un potentiel inférieur à celui imposé par le diviseur de tension interne, ce qui déclenche la sortie du comparateur correspondant à un niveau logique 1, et donc détermine un **niveau logique haut** sur la broche « **OUT** » de U1.

De ce fait, la sortie complémentaire de la bascule est portée à un niveau logique bas et permet ainsi de bloquer le transistor interne dont le collecteur est relié à la broche 7, ce qui permet de charger le condensateur de temporisation.

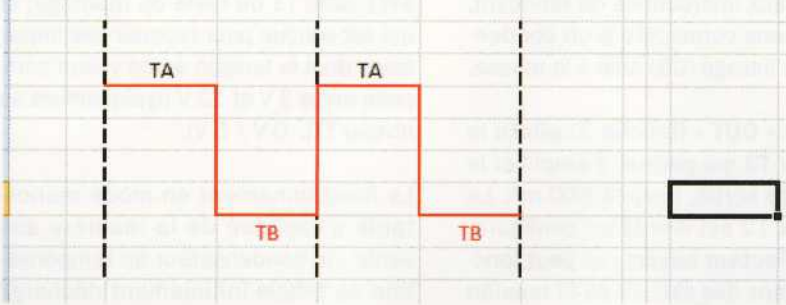
Lorsque la différence de potentiel sur ce dernier dépasse celle que le diviseur de tension interne du NE555 applique sur l'entrée non inverseuse du comparateur supérieur, ce dernier bascule sa sortie d'un niveau logique bas vers un niveau logique haut et réinitialise la bascule, portant ainsi à un niveau logique bas (0 V) la sortie « OUT ».

La **forme de l'onde** ainsi obtenue aux bornes du condensateur correspond à un **circuit RC de premier ordre**, c'est-à-dire une exponentielle croissante.

Lorsque cette exponentielle atteint une valeur égale à deux tiers de la tension d'alimentation Vcc, la bascule interne est désactivée, ramenant la sortie et le condensateur à zéro.

Tableau 1

		ASTABLE					MONOSTABLE	
		Tmin = 1 kohm		Tmax = 1 Mohm			1 kohm	1 Mohm
C (uF)	C (F)	TAmin (ms)	TBmin (ms)	TAmax (ms)	TBmax (ms)	t	Tmin (ms)	Tmax (ms)
1 uF	0,000001	0,153153	0,15246	0,846153	0,15246		0,0011	1,1011
2,2 uF	0,0000022	0,336937	0,335412	1,861537	0,335412		0,00242	2,42242
3,3 uF	0,0000033	0,505405	0,503118	2,792305	0,503118	t1	0,00363	3,63363
4,7 uF	0,0000047	0,719819	0,716562	3,976919	0,716562		0,00517	5,17517
5,8 uF	0,0000058	0,888287	0,884268	4,907687	0,884268		0,00638	6,38638
10 uF	0,00001	1,53153	1,5246	8,46153	1,5246	t2	0,011	11,011
12 uF	0,000012	1,837836	1,82952	10,15384	1,82952		0,0132	13,2132
15 uF	0,000015	2,297295	2,2869	12,6923	2,2869		0,0165	16,5165
18 uF	0,000018	2,756754	2,74428	15,23075	2,74428		0,0198	19,8198
22 uF	0,000022	3,369366	3,35412	18,61537	3,35412		0,0242	24,2242
33 uF	0,000033	5,054049	5,03118	27,92305	5,03118	t3	0,0363	36,3363
47 uF	0,000047	7,198191	7,16562	39,76919	7,16562		0,0517	51,7517
58 uF	0,000058	8,882874	8,84268	49,07687	8,84268		0,0638	63,8638
68 uF	0,000068	10,4144	10,36728	57,5384	10,36728		0,0748	74,8748
100 uF	0,0001	15,3153	15,246	84,6153	15,246	t4	0,11	110,11



Liste des composants

R1 .....1 kΩ boîtier 0805
R2 .....220 kΩ boîtier 0805
R3 .....4,7 kΩ boîtier 0805
R4 .....10 kΩ boîtier 0805
R5 .....4,7 kΩ boîtier 0805
R6 .....10 kΩ boîtier 0805
R7 .....Trimmer 1 MΩ

C1 .....3,3 µF céramique boîtier 0805
C2 .....10 µF céramique boîtier 0805
C3 .....33 µF 6,3 V électrolytique série WCAP-ASLI, ∅ 4 mm x 5,35 mm
C4 .....100 µF 6,3 V électrolytique série WCAP-ASLI, ∅ 4 mm x 5,35 mm
C5 .....10 nF céramique boîtier 0805

T1 .....BC817
T2 .....BC817
U1 .....NE555DR
SW1 ...interrupteur 2 voies

Divers

Barrette mâle 4 pôles (x2)
Cavaliers
Barrette mâle 7 pôles coudée à 90°
Connecteur JST 3 pôles, pas de 1,25mm pour circuit imprimé (x2)
Connecteur JST 3 pôles, pas de 1,25mm fils volants (x2)

NB : les typons des circuits imprimés à l'échelle 1 sont disponibles en téléchargement dans le sommaire détaillé de la revue.

La durée de l'impulsion **t** est donnée par la formule suivante :

$t = 1,1 * R * C_t$

où R est la résistance de charge du condensateur de temporisation (C_t) insérée entre les broches 6,7 et la masse. Comme R est composée par la résistance R1 et le trimmer R7, nous pouvons faire varier la durée de l'impulsion.

Le tableau 1 montre toutes les combinaisons possibles entre les condensateurs sélectionnés à l'aide des cavaliers et les valeurs minimale et maximale de R, correspondant respectivement à la position du curseur vers les broches 6, 7 et vers R1.

La **durée de l'impulsion** obtenue à partir de la formule **est en secondes si R est exprimé en kiloohm et C_t en millifarad** (1 mF = 1000 µF) ou R en mégohm et C_t en microfarad.

À ce stade, l'état logique bas de la sortie de la bascule détermine un niveau logique 1 (haut) sur la sortie « /Q » et la saturation du transistor interne relié à la broche 7.

En effet, ce dernier court-circuite le condensateur de temporisation qui se décharge jusqu'à ce qu'une nouvelle impulsion arrive sur l'entrée de déclenchement (trigger).

Notez que si l'entrée « IN » est alimentée en continu, lorsque la sortie OUT revient à un niveau logique 0, le monostable est déclenché de nouveau et un nouveau cycle de temporisation commence.

Examinons maintenant le fonctionnement en **mode astable**, qui est obtenu en positionnant le double inverseur sur la position **AST**. Dans ce cas, la broche 2 est déconnectée du collecteur de T1 (donc l'état de l'entrée « IN » n'a pas d'influence) et R2 n'est plus court-circuitée.

Ainsi, les broches 2 et 6 sont reliées entre elles et la broche 7 est reliée à la résistance R7 qui permet de faire varier la fréquence de travail du multivibrateur (toujours selon le tableau 1) et à la résistance R2.

La broche 6 (THR) est reliée de l'autre côté à R2 et à un ou plusieurs condensateur(s) sélectionné(s) à l'aide du ou des cavalier(s). La capacité résultante est C_t. La fréquence de commutation de la sortie du NE555, dans la configuration astable, est donnée par la formule :

$f = 1/2(C_t * 2R)$

où C_t est le condenseur sélectionné et R la résistance donnée par la somme de R1 et R7 (les considérations déjà faites pour le mode monostable sont valables).

Analysons le **fonctionnement du multivibrateur**, supposons que le double inverseur soit positionné sur « AST », le circuit est alimenté et le **condensateur est déchargé dans la condition initiale**.

En négligeant l'effet de C5, qui se charge presque immédiatement et ne perturbe donc pas la tension de référence appliquée aux diviseurs de tension internes des comparateurs du NE555, l'état initial du circuit implique que les **broches 2 et 6 sont à environ 0 V** car la sortie du comparateur inférieur est à un niveau haut.

Ceci détermine un **niveau logique haut** sur l'entrée « **SET** » de la bascule et sur la sortie Q. La sortie « /Q » prend un niveau logique 0 laissant dans un état bloqué le transistor connecté à la broche « **DISCHARGE** ». Ainsi, le condensateur se charge.

À ce moment, C_t commence à se charger, grâce au courant circulant dans les résistances R1 et R7 (montées en série) ainsi qu'également grâce au courant circulant dans R2, jusqu'à dépasser la tension de seuil du comparateur inférieur qui est égale à 1/3 de celle appliquée sur la broche 8.

La sortie du comparateur est donc ramenée à un niveau logique bas (0) et libère l'entrée « SET » de la bascule, dont la sortie reste à un niveau haut, jusqu'à ce que la tension aux extrémités de C_t, continuant à croître, n'atteigne pas une valeur supérieure à 2/3 de la tension d'alimentation du NE555, c'est-à-dire au seuil de commutation du comparateur supérieur.

Lorsque cela se produit, la sortie du comparateur supérieur passe à un niveau logique haut et réinitialise la bascule. La sortie directe Q passe à un niveau logique 0 tandis que la sortie « /Q » prend un niveau haut. Donc le transistor interne connecté à la broche 7 entre en saturation, lequel à travers R2 décharge le condensateur C_t.

Cette situation se maintient tant que la tension sur les broches du condensateur C_t ne tombe pas en dessous du seuil de commutation du comparateur inférieur, puis la sortie de ce dernier revient à un niveau logique haut et active la bascule, la sortie Q passe alors à un niveau logique 1 et « /Q » à 0.

Dans cette condition, le condensateur reprend sa charge car la broche 7 se trouve dans un état « ouvert » (non connectée). La charge se produit de nouveau jusqu'aux 2/3 de la tension appliquée entre les broches 1 et 8 du NE555, ainsi le cycle recommence.

En conséquence, **la sortie OUT est maintenue à un niveau haut pendant la période où le condensateur C_t a une tension à ses bornes passant de 1/3 à 2/3 de la tension d'alimentation** (soit 5 VDC, donc la transition se situe entre 1,67 V et 3,33 V).

La sortie **OUT passe à un niveau bas** (environ 0 V) **pendant la phase de décharge de C_t**, c'est-à-dire lorsque la tension à ses bornes chute de 2/3 à 1/3 de la tension d'alimentation.

Les durées dépendent des valeurs des résistances R1, R2 et R7 et sont d'autant plus semblables que la valeur de R2 est plus petite par rapport à la somme des valeurs de R1 et R7.

En effet, la charge s'effectue à travers ces 3 résistances, tandis que la décharge ne s'effectue qu'à travers la résistance R2.

Il faut mentionner que la **décharge a une valeur de seuil de 1,67 V** et, à résistance égale, **la décharge est plus rapide que la charge à 3,33 V**.

Donc, pour une certaine valeur de R2, la durée de l'état logique 1 sur la sortie du NE555 est égale à celui de l'état logique 0.

Le cycle décrit continue tant que la tension d'alimentation est maintenue, c'est-à-dire tant que le double inverseur n'est pas manœuvré (et bien sûr tant que le montage est alimenté).

Nous concluons la description du montage en remarquant que, comme il s'agit d'une carte de prototypage, nous avons prévu 2 types de connecteurs pour l'alimentation et les connexions d'entrée et de sortie du trigger.

Toutes les connexions sont accessibles soit via des connecteurs JST soit via une barrette au pas de 2,54 mm. Cela vous permettra d'intégrer la carte dans du matériel existant.

Le connecteur d'alimentation est celui qui comporte les points 5V, GND et IN (l'entrée TTL du trigger en mode monostable), tandis que l'autre, en plus du report du 5 V, comporte la sortie OUT (OT) du NE555 et celle du collecteur du transistor T2 (sortie de puissance « OC »). Sur la barrette, vous retrouvez toutes les connexions des deux connecteurs JST.

Réalisation pratique

Après avoir expliqué le fonctionnement des deux modes du NE555 pouvant être obtenus à partir de cette carte de prototypage, nous allons maintenant passer à la fabrication du circuit imprimé double face.

Vous devez tout d'abord télécharger sur notre site les typons dans le sommaire détaillé de la revue (à l'onglet « Télécharger », descendre vers le bas de la page).

Notez aussi que la carte existe en version montée, avec tous les composants soudés et prête à l'emploi.

Une fois le circuit imprimé réalisé et percé, commencez par disposer à l'aide de colle les composants CMS non polarisés (les résistances et les condensateurs C1, C2 et C5).

Vous devez utiliser un fer à souder pour composants CMS ayant une panne de 0,2 mm de Ø et de la soudure de 0,5 mm de Ø (spéciale CMS).

Soudez délicatement sans excès d'étain, car il pourrait se propager sous les composants (risques de court-circuits).

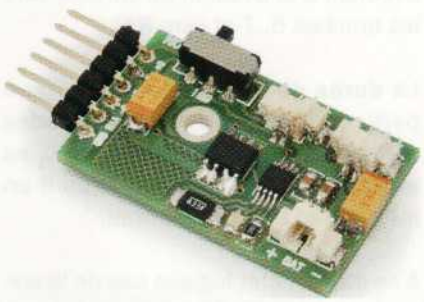
Ensuite, toujours à l'aide d'une pointe de colle, positionnez le NE555 U1 avec son repère en forme de « U » orienté vers l'intérieur du circuit (vers R4).

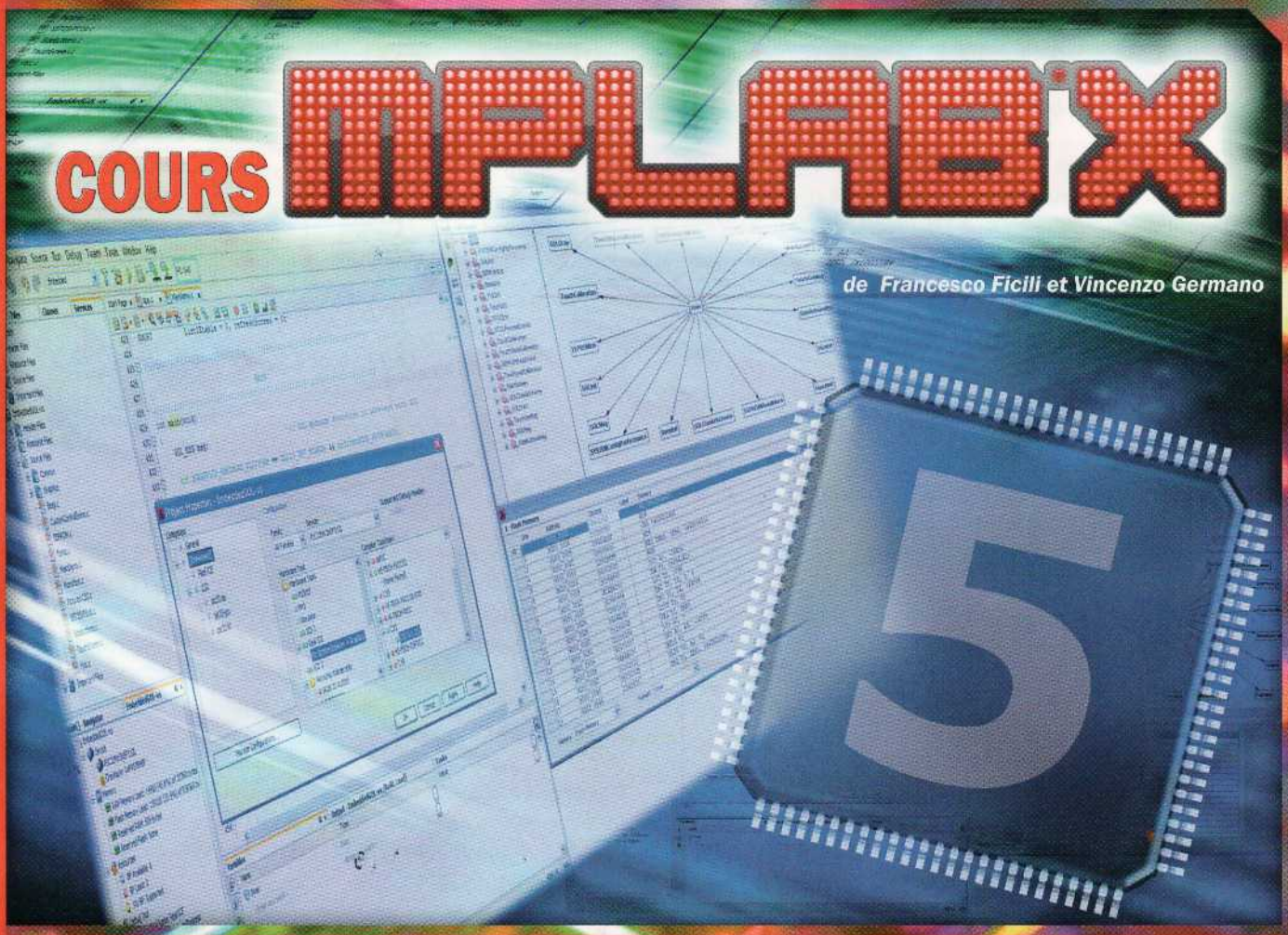
Puis commencez par souder une patte, laissez refroidir et soudez la patte diamétralement opposée puis laissez refroidir entre chaque soudure. Procédez ainsi de suite pour les autres pattes.

Continuez en soudant délicatement les transistors et les condensateurs polarisés en respectant l'orientation du marquage, reportez-vous à l'encadré intitulé « Plan de montage de la breakout board NE555 ».

Terminez en soudant le double inverseur, le trimmer et les barrettes des cavaliers. Eventuellement, soudez les connecteurs JST ou la barrette mâle au pas de 2,54 mm, selon votre convenance.

À l'aide d'une loupe, vérifiez que tous les composants sont correctement soudés et qu'il n'y a pas de cheveux d'ange entre les pattes de U1.





de Francesco Ficili et Vincenzo Germano

Cours MPLAB X

Cinquième partie

Nous continuons notre route à la découverte de MPLAB-X, le nouvel environnement de développement réalisé par Microchip Technology et qui remplace l'ancien IDE MPLAB. Dans cette leçon, nous allons aborder la manière de créer des applications en utilisant la pile (stack) « USB host » de Microchip.

Dans les dernières leçons, nous avons créé une application de type périphérique « USB Device » sur un PIC32, en utilisant la « stack » USB Open Source de Microchip et l'ordonnanceur (scheduler) introduit dans la leçon 3. Maintenant, nous allons faire à peu près la même chose, mais en utilisant la stack USB Host (hôte USB) au lieu de la pile périphérique. L'exemple d'application que nous allons implémenter nous permettra de sauvegarder sur une clé USB ou un disque dur la tension lue sur un canal analogique du microcontrôleur connecté à un potentiomètre.

La stack USB host

Nous savons que la structure définie par la norme USB comprend une unité hôte et une ou plusieurs unités périphériques.

Les périphériques USB ne peuvent pas communiquer entre eux, mais uniquement avec l'hôte (host), qui joue le rôle de Maître au niveau du bus USB.

Dans la plupart des applications, le rôle de l'hôte est assumé par un ordinateur personnel, doté d'un système d'exploitation complexe tel que Microsoft Windows ou Mac OS et de ressources matérielles virtuelles infinies pour l'application. Le système peut être facilement configuré dynamiquement (s'il ne l'a pas déjà fait) à l'aide du pilote (driver) relatif au périphérique USB avec lequel il doit s'interfacer.

Dans le cas d'un système hôte embarqué, la situation est toutefois différente. Dans la plupart des applications, vous n'avez pas de système d'exploitation de type Windows/Linux, mais un RTOS simple ou, comme dans notre cas, un seul scheduler. Il n'est alors pas possible de mettre à jour le système à l'aide d'un nouveau driver.

Nous devons donc traiter le problème différemment, en considérant une application spécifique qui utilisera le bus à l'aide d'un firmware (programme) que nous devons développer.

Architecture de la stack USB Host

L'architecture de la stack USB Host de Microchip est représentée de manière schématique en figure 1. Vous pouvez remarquer qu'elle se compose de trois couches, à savoir « **Application** », « **USB Client Driver Layer** » et « **USB Host Layer** » décrites ci-après :

- **Application Layer** : il s'agit de la couche de l'application d'implémentation, elle regroupe toutes les fonctions de haut niveau de l'application spécifique. Elle est généralement développée à partir de zéro par le développeur de l'application ;
- **USB Client Driver Layer** : comme nous le savons, chaque périphérique USB implémente une fonction spécifique (comme par exemple une souris, un clavier, une mémoire, etc.). Chacun de ces dispositifs a une implémentation spécifique différente et nécessite par conséquent un pilote (driver) spécifique pour être interfacé. Cette couche implémente les différents pilotes clients de tous les périphériques USB possibles pouvant être connectés à un système hôte embarqué ;
- **USB Host Layer** : cette couche fournit une abstraction du protocole USB et prend notamment en charge la mise en œuvre des services suivants :
  - identification du périphérique ;
  - énumération du périphérique ;
  - gestion du pilote client spécifique ;
  - abstraction de la communication avec le périphérique.

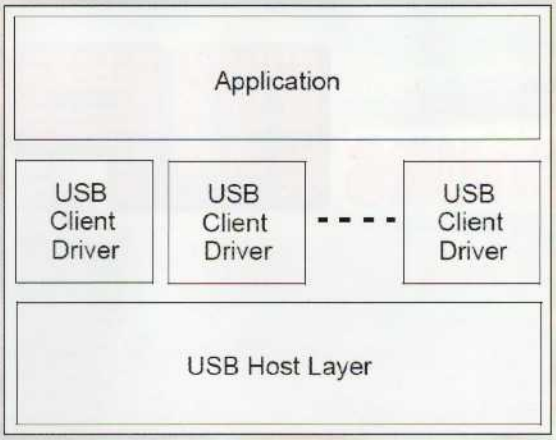


Figure 1 : architecture de la stack USB Host.

Lorsqu'un périphérique est connecté au bus, la couche hôte (Host) l'identifie en lisant les descripteurs (structures d'identification spécifiques définies par la norme USB 2.0), puis vérifie si un pilote du périphérique est présent dans une liste spécifique de pilotes clients. Si un fichier approprié est trouvé, il l'initialise pour lui permettre d'être utilisé à des niveaux supérieurs.

L'opération d'identification et d'énumération est gérée en arrière-plan par une tâche spécifique réalisée sous la forme d'une machine d'état dont l'implémentation est visible en figure 2.

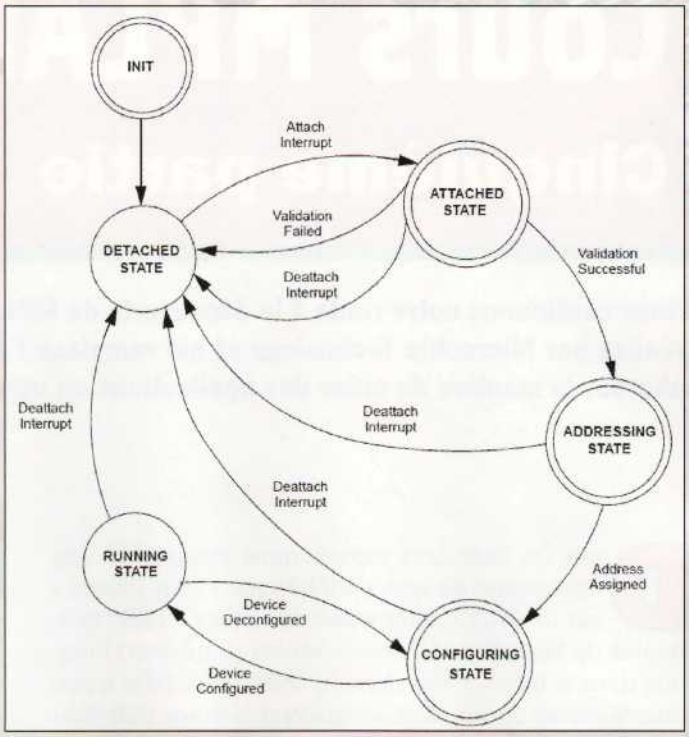


Figure 2 : implémentation de la machine d'état qui gère l'énumération d'un périphérique USB dans la stack USB Host de Microchip.

Comme vous pouvez le voir, une fois connecté, un périphérique passe par différents états, avant d'être configuré puis mis en fonction. Il pourra alors être réellement utilisé par l'hôte.

Table des pilotes (drivers) clients

Comme nous l'avons vu précédemment, chaque périphérique USB (ou famille de périphériques) possède un pilote spécifique (pilote client) qui lui permet d'être géré. Comme un hôte (Host) doit généralement prendre en charge plusieurs périphériques (cela peut se produire si une classe de périphériques doit être prise en charge, ou même si un périphérique unique mais composite est pris en charge, c'est-à-dire si le périphérique implémente plusieurs fonctions à l'aide d'une interface), il doit généralement implémenter plusieurs pilotes clients.

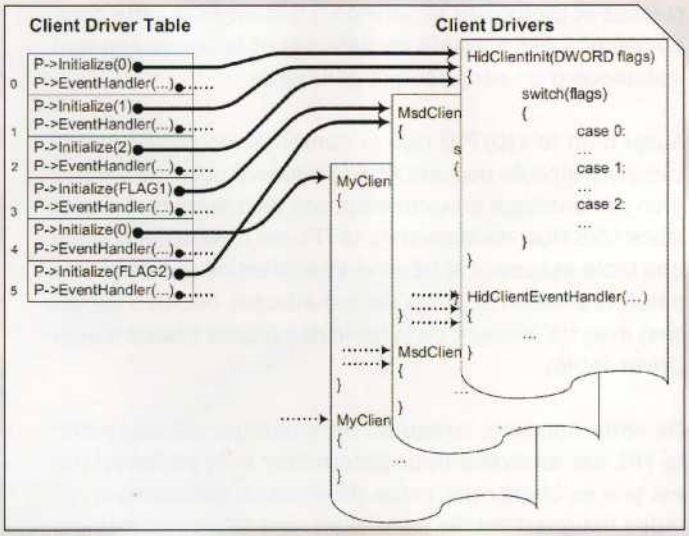


Figure 3 : « Client Driver Table » ou table des pilotes clients.

Lors de la mise en œuvre de la stack USB Host de Microchip, cette exigence est satisfaite par l'implémentation d'une table appelée « **Client Driver Table** » ou **table des pilotes clients** (CDT). Chaque entrée de cette table correspond à un seul pilote client et contient un pointeur sur la fonction d'initialisation et sur la fonction de rappel qui gère la gestion des événements (« event handling callback » ou rappel de gestion des événements).

La figure 3 représente schématiquement l'implémentation de la « Client Driver Table ». **Lorsqu'un périphérique est connecté au bus**, la **couche hôte** (layer host) **lit ses descripteurs et détermine si le périphérique peut être pris en charge ou non**. Si tel est le cas, l'entrée correcte de la table des pilotes clients est indexée et la routine d'initialisation est appelée.

Une fois le périphérique initialisé, lorsqu'un événement particulier est généré lors d'une activité, le gestionnaire d'événement (event handler) spécifique est appelé pour gérer l'événement spécifique.

Liste des périphériques ciblés (Targeted Peripheral List)

Comme mentionné précédemment, une implémentation d'un hôte embarqué, contrairement à un ordinateur personnel, ne peut prendre en charge qu'un nombre limité de périphériques USB.

Cet ensemble, dans l'implémentation de Microchip, est défini par une liste de **périphériques ciblés** (**Targeted Peripheral List** ou **TPL**) c'est-à-dire une liste des périphériques pris en charge. Les périphériques pris en charge peuvent être identifiés de deux manières :

- combinaison VID/PID ;
- combinaison classe/sous-classe/protocole.

Un exemple d'un élément d'une TPL qui utilise la combinaison classe/sous-classe/protocole est illustré en figure 4. La spécification USB exige que chaque périphérique soit identifié au moyen d'un VID (acronyme de Vendor ID) et d'un PID (Product ID).

Ainsi, les périphériques sont généralement identifiés par une paire de nombres hexadécimaux, comme par exemple : 1014:003E. Les 4 premiers chiffres représentent l'ID du constructeur (04b3 = IBM), c'est le VID (Vendor = Constructeur) et les 4 derniers chiffres représentent l'ID du périphérique ou PID (3108 = ThinkPad Optical Travel Mouse).

La combinaison de ces deux nombres identifie une famille spécifique de périphériques d'un constructeur afin que vous puissiez déterminer une implémentation adaptée et un support du périphérique.

Une autre méthode, prévue par la spécification, consiste à déterminer le comportement d'un périphérique USB générique. Par conséquent pour concevoir le pilote client approprié, il suffit d'identifier la classe à laquelle il appartient.

Toujours selon la spécification, les périphériques USB sont divisés en classes, et à l'intérieur de celles-ci, en sous-

Description	Class Name	Class Code	Subclass Code	Protocol
Flash Drive	Mass Storage	0x08	0x06	0x50

Figure 4 : exemple d'un élément d'une TPL identifiée par la combinaison classe/sous-classe/protocole.

classes et protocoles supportés. En identifiant cette combinaison, il est possible de déterminer le comportement spécifique d'un périphérique générique.

Aussi bien le VID/PID que la combinaison classe/sous-classe/protocole peuvent être facilement obtenus à partir d'un périphérique en examinant ses descripteurs. Dans la stack USB Host de Microchip, la TPL est modélisée comme une table associant le périphérique identifié (qui peut être déterminé avec l'une des deux méthodes décrites ci-dessus) avec un élément de la table des pilotes clients (Client Driver Table).

De cette manière, lorsqu'un périphérique est connecté, la TPL est analysée pour déterminer si le périphérique est pris en charge par l'hôte (Host) et, le cas échéant, un index indique l'entrée de la table des pilotes clients qui doit être utilisée pour gérer le périphérique. En plus de fournir une référence pour la « Client Driver Table », la TPL contient d'autres informations telles que des flags et des paramètres de configuration pouvant être transmis à un pilote client spécifique en fonction des besoins de l'application. La figure 5 montre une représentation schématique de ce qui précède.

Notez que d'après la figure 5, une seule entrée de la TPL peut désigner le même pilote (driver) dans la CDT (premier et troisième Device identifier).

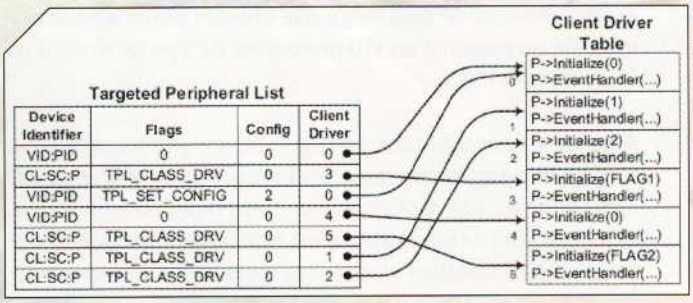


Figure 5 : relation entre la TPL et la Client Driver Table.

Cela permet, lorsque des périphériques sont identifiés au moyen de la combinaison VID/PID, d'utiliser le même pilote dans le cas où l'implémentation de leur interface serait effectivement la même.

Utilisées en combinaison, **la TPL et la CDT constituent un mécanisme puissant pour gérer un groupe hétérogène de périphériques USB devant être pris en charge par un seul hôte embarqué**. Si vous souhaitez plus d'informations sur la structure de la stack USB host de Microchip, vous pouvez vous référer aux notes d'application AN1140 et AN1141.

La classe USB MSD

Cependant, comme nous l'avons vu précédemment, selon le périphérique que nous allons interfacer, l'implémentation spécifique de l'application du côté de l'hôte (Host) changera, car elle devra certainement implémenter un driver spécifique ainsi qu'une couche supplémentaire en fonction de l'application envisagée.

Heureusement, Microchip fournit une série de pilotes clients pour les classes les plus courantes de périphériques USB (telles que les lecteurs flash, les périphériques HID, les imprimantes, etc.) et d'autres couches supplémentaires nécessaires pour certaines applications (par exemple dans le cas de la classe MSD, des composants pour la gestion du système de fichiers et de l'interface SCSI sont également fournis).

Ces paquets font partie de la stack USB (ou plus généralement de la MLA). Il existe également une implémentation générale (appelée « generic host » ou hôte générique) qui est en fait un modèle pour l'implémentation des pilotes clients pour les classes non officiellement prises en charge par la pile (stack).

L'application spécifique que nous analyserons en détail dans cette leçon est l'implémentation d'un hôte USB (USB Host) capable de gérer un périphérique de stockage de masse (MSD) avec un système de fichiers.

Nous allons donc implémenter un hôte capable d'interfacer des clés USB courantes ou des disques durs. La structure que nous allons construire est basée sur l'architecture décrite précédemment et ajoute des couches juste au-dessus du driver du périphérique de stockage de masse, comme illustré en figure 6.

Ces couches supplémentaires sont :

- **SCSI Command Set** : ce niveau gère l'interface SCSI vers la mémoire ;
- **File System** : implémente la gestion du système de fichiers (le type FAT32 est utilisé) et constitue l'interface de haut niveau utilisée par l'application pour interagir avec la clé ou le disque ;

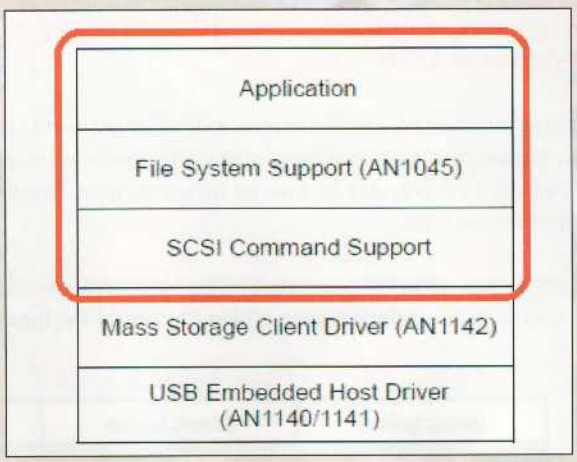


Figure 6 : architecture d'une application hôte prenant en charge la classe USB MSD.

- **Application** : cette couche implémente l'application spécifique, en l'occurrence une application d'enregistrement de données.

La couche de gestion du système de fichiers fournit plusieurs API pour la gestion des disques, telles que les fonctions d'écriture, de lecture, de gestion des répertoires, etc. Pour obtenir une description détaillée des différentes API, il est possible de consulter la note d'application AN1045 : « Implementing File I/O Functions Using Microchip's Memory Disk Drive File System Library ».

Configuration d'un projet hôte USB (USB Host)

Comme nous l'avons abordé dans les paragraphes précédents, **l'implémentation d'une application « USB Host »** qui **interagit avec une mémoire de masse de type USB peut être une tâche difficile**, car la mise en œuvre du « **framework host** » est encore plus complexe que celle du périphérique (device). En particulier, la configuration du framework pour une application spécifique peut être une procédure très compliquée à réaliser.

Pour rappel, en programmation, un framework (ou infrastructure logicielle) désigne un ensemble cohérent de composants logiciels structurels, qui sert à créer les fondations ainsi que les grandes lignes de tout ou d'une partie d'un logiciel (architecture).

Un framework se distingue d'une simple librairie principalement par :

- son caractère générique, faiblement spécialisé, contrairement à certaines librairies. Un framework peut à ce titre être constitué de plusieurs librairies chacune spécialisée dans un domaine ;

Figure 7 : écran principal de l'outil UsbConfig.

- la structure qu'il impose de par sa construction même implique que le développeur doit respecter certains patrons de conception. Les librairies le constituant sont alors organisées selon le même paradigme.

Les frameworks sont donc conçus et utilisés pour modeler l'architecture des logiciels applicatifs, des applications web, des composants logiciels, etc. Les frameworks sont acquis par les informaticiens, puis incorporés dans des logiciels applicatifs mis sur le marché, ils sont par conséquent rarement achetés et installés séparément par l'utilisateur final.

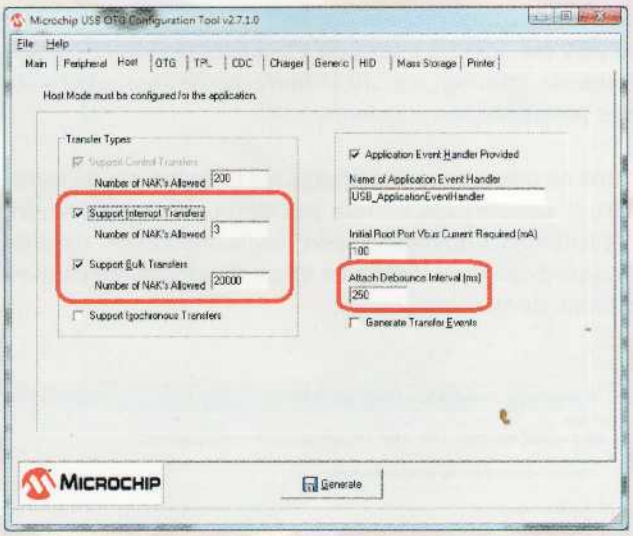


Figure 8 : configuration des paramètres de l'hôte.

Ainsi, dans un projet « USB host » construit avec la stack USB de Microchip, toutes les configurations relatives au framework sont contenues dans les fichiers « **usb_config.c** » et « **usb_config.h** ». En particulier, outre la TPL et la CDT, tous les paramètres génériques et spécifiques des classes particulières prises en charge sont inclus.

Comme ces configurations varient d'une application à l'autre (ainsi que la composition de la TPL et de la CDT), Microchip a créé une application, appelée précisément « UsbConfig », qui vous permet de générer le code relatif à la configuration à l'aide d'une interface graphique intuitive.

NB : l'outil de configuration « **USBConfig.exe** » est disponible en téléchargement avec les exercices de cette leçon dans le sommaire détaillé de la revue (allez vers le bas de la page web dans la section « Télécharger »).

Examinons le fonctionnement de cette application en générant les fichiers liés au projet pratique que nous allons illustrer dans cette leçon. L'écran initial de l'outil est présenté en figure 7. Il permet de sélectionner la plateforme et le type de périphérique (outre l'hôte embarqué et le périphérique, il existe également des implémentations

hybrides, telles que l'OTG, qui ne seront pas traitées dans ce cours).

Sélectionnons **PIC32** en tant que **plate-forme** et « **USB Embedded Host** » comme **type de périphérique** (Device Type) et passons à l'onglet « Host ». À partir de là, il est possible de choisir le type de transfert à utiliser et d'autres options de base telles que le nom du rappel de gestion des événements (callback), le courant maximum à délivrer, etc.

Sélectionnons « **Support Interrupt Transfers** » (3) et « **Support Bulk Transfers** » (20000) comme mode de transfert de données. Le mode isochrone (synchronisation) n'est pas requis pour l'application. Choisissons **100 mA** comme courant maximal (Initial Root Port Vbus Current Required) et **250 ms** comme intervalle de rebondissement (Attach Debouce Interval) afin de prendre également en charge des périphériques plus lents.

Nous ne prenons pas en charge les événements de transfert et conservons le nom par défaut du gestionnaire d'événements de l'application (Application Event Handler Provided doit être coché). La figure 8 montre une capture d'écran de la configuration.

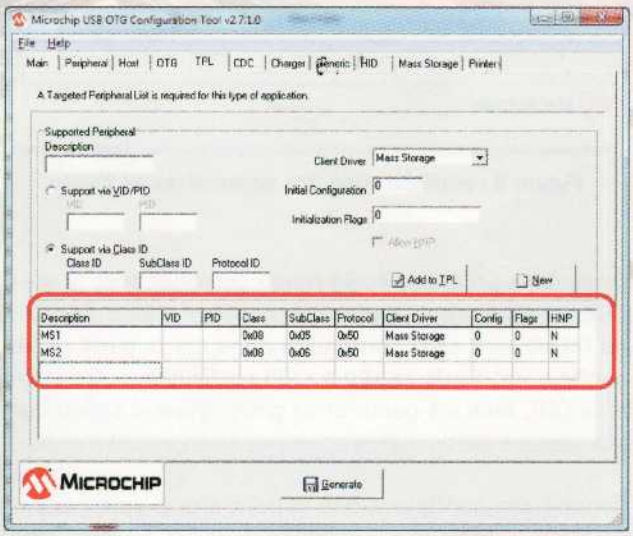


Figure 9 : création de la TPL.

Passons maintenant à l'onglet « **TPL** » pour créer la liste des périphériques ciblés, comme illustré en figure 9.

Dans cet onglet, nous devons insérer les différentes entrées de la TPL, constituées des paramètres identifiant le périphérique (avec l'une des deux méthodes décrites ci-dessus, la combinaison VID/PID ou la combinaison classe/sous-classe/protocole), du pilote client (pour cela, il existe une liste déroulante contenant les pilotes clients les plus courants, ou il est possible d'utiliser la classe générique pour en implémenter un nouveau driver), ainsi que tous les flags et les configurations.

Dans notre cas, nous allons insérer deux entrées, toutes les deux connectées au pilote « **Mass Storage Client Driver** ».

La première identifiée par la classe/sousclasse/protocole « 0x08 – 0x05 – 0x50 » et la seconde par « 0x08 - 0x06 - 0x50 ». Cela est dû au fait que certains périphériques de stockage de masse utilisent la sous-classe « 0x05 » tandis que d'autres la sous-classe « 0x06 ».

De cette manière, nous sommes sûrs que notre hôte embarqué reconnaîtra les deux, en connectant toujours le même pilote client. Dans les deux cas, nous n'insérons pas de flags, ni d'options particulières.

Dans le champ « Supported Peripheral Description » insérez « MS1 », ensuite sélectionnez dans la liste déroulante « Client Driver » « Mass Storage. Puis cochez la case « Support via Class ID ».

Ensuite insérez dans les champs suivants les valeurs :

Class ID : 0x08 ;
Sub Class ID : 0x05 ;
Protocol ID : 0x50.

Enfin cliquez sur le bouton « **Add to TPL** », la ligne en dessous doit s'afficher avec les valeurs entrées précédemment (voir la figure 9). Faites de même pour MS2.

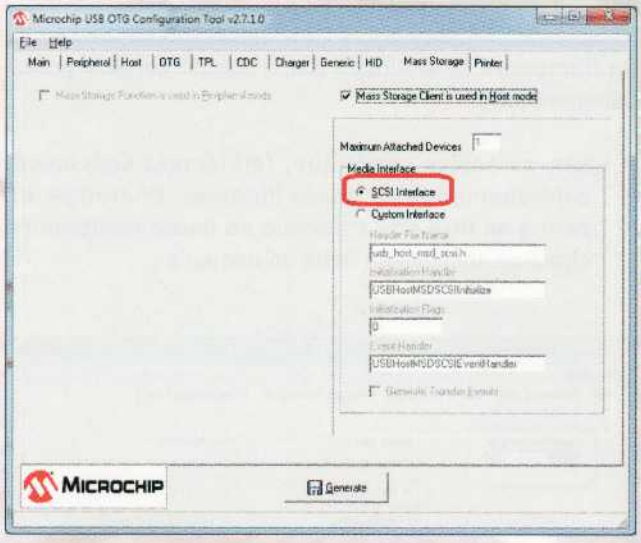


Figure 10 : définition de l'interface du disque dur.

Maintenant, cliquez sur l'onglet « **Mass Storage** » et cochez la case « **Mass Storage Client is used in Host mode** » pour insérer les détails de l'interface spécifique, comme illustré en figure 10. Dans notre cas, nous utiliserons l'interface SCSI pour la gestion des disques, il faut donc cocher « SCSI Interface ».

À ce stade, vous pouvez cliquer sur le bouton « **Generate** » pour créer les fichiers.

Si nous ouvrons le fichier « **usb_config.c** », nous trouvons l'implémentation en C de la TPL et de la CDT, reportées respectivement dans le listing 1 et sur le listing 2.

Exemple pratique : enregistreur de données USB Host

À ce stade, nous pouvons passer à la description de l'exemple d'application que nous présentons dans cette leçon.

Comme nous l'avons déjà dit dans les paragraphes précédents, notre idée est de réaliser une application de type datalogger (enregistrement de données) qui enregistre sur une clé USB les valeurs lues par un canal analogique relié à un potentiomètre. Une application interne de contrôle démarre et termine les phases de « log ».

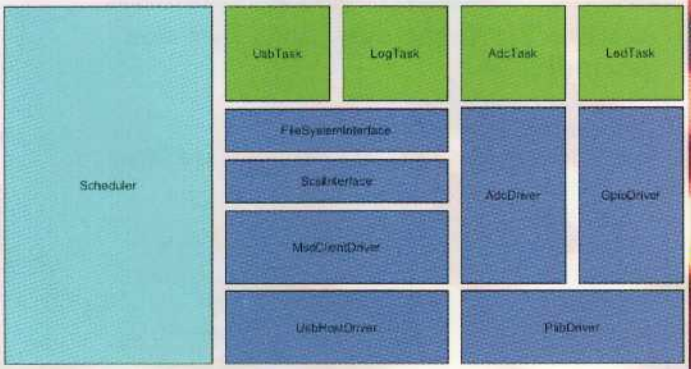


Figure 11 : architecture du programme.

Listing 1 : implémentation en C de la TPL générée par l'outil.

```
// ******************************************************************************************
// USB Embedded Host Targeted Peripheral List (TPL)
// ******************************************************************************************

USB_TPL usbTPL[] =
{
   { INIT_CL_SC_P( 0x08ul, 0x05ul, 0x50ul ), 0, 0, {TPL_CLASS_DRV} } // MS1
   ,
   { INIT_CL_SC_P( 0x08ul, 0x06ul, 0x50ul ), 0, 1, {TPL_CLASS_DRV} } // MS2
};
```

Listing 2 : implémentation en C de la CDT générée par l'outil.

```
// ******************************************************************************************
// Client Driver Function Pointer Table for the USB Embedded Host foundation
// ******************************************************************************************

CLIENT_DRIVER_TABLE usbClientDrvTable[] =
{
   {
      USBHostMSDInitialize,
      USBHostMSDEventHandler,
      0
   }
   ,
   {
      USBHostMSDInitialize,
      USBHostMSDEventHandler,
      0
   }
};
```

L'architecture du programme est illustrée en figure 11 et, comme nous pouvons le voir, elle repose, comme d'habitude, sur le scheduler présenté dans les leçons précédentes.

Listing 3 : Table des tâches et des périodicités.

```
/* Table des tâches à appeler */
void (*TaskArray[ACTIVE_TASK_NUMBER]) (void) =
{
   LedTask,
   UsbTask,
   LogTask,
   AdcTask
};

/* Table des tâches des périodes d'expiration ou de timeout (un timeout correspond au dépassement d'un délai accordé pour l'exécution d'une certaine action. */
UINT16 TaskPeriodTimeoutMs[ACTIVE_TASK_NUMBER] =
{
   LED_TASK_PERIOD_MS,
   USB_TASK_PERIOD_MS,
   LOG_TASK_PERIOD_MS,
   ADC_TASK_PERIOD_MS,
};
```

Comme d'habitude, pour que le scheduler fonctionne correctement, puisqu'il s'agit d'une implémentation non préemptive, nous devons développer nos tâches en utilisant le paradigme multitâche coopératif. Comme nous pouvons le voir dans l'architecture, les quatre tâches suivantes ont été prévues. **UsbTask** : cette tâche est chargée d'appeler la tâche qui maintient les machines d'état du pilote USB Host (USBTasks()) ;

LogTask : cette tâche implémente les machines d'état de l'application de « logging ». En particulier, une machine d'état est implémentée pour le contrôle (LogControl()) et une autre machine d'état gère efficacement la mémoire LoggingStateMachine()) ;

AdcTask : cette tâche implémente la machine d'état qui lit les données du convertisseur analogique/digital et génère des événements de « logging » interceptés par la tâche de « log » ;

LedTask : cette tâche concerne la gestion des LED du système.

Outre l'ordonnanceur (scheduler), notre architecture repose sur la stack USB Host précédemment présentée (et en particulier sur l'implémentation du pilote client MSD) pour la gestion de l'USB et sur les librairies du système ainsi que les pilotes des périphériques pour la gestion de l'ADC et du port GPIO.

La figure 12 illustre le « Project Manager » du projet et met en évidence les quatre tâches qui composent la couche d'application.

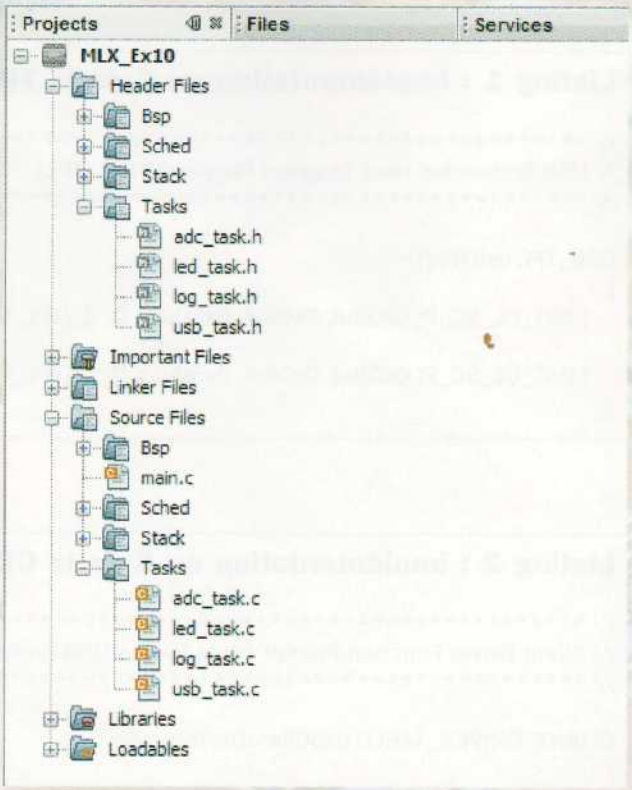


Figure 12 : structure du « Project Manager » dans MPLAB-X.

Listing 4 : Implémentation de la machine d'état de la tâche « Adc ».

```
/*****************************************************
* Fonction :   void AdcStateMachine (void)
* Input :    Non
* Output : Non
* Auteur : F.Ficili
* Description :    Gestion de la machine d'état d'acquisition des données Adc
* Date :   11/04/18
*****************************************************/
void AdcStateMachine (void)
{
   static AdcSmStateType AdcSmState = Idle;
```

```
static UINT16 AcqDelayCounter = 0;
UINT16 AdcDataRaw = 0;

switch (AdcSmState)
{
   case Idle:
      /* Vérifie le statut du logger (enregistreur) */
      if (ReceiveEvt(&LoggerReady))
      {
         /* Va à WaitLogDelay */
         AdcSmState = WaitLogDelay;
      }
      break;

   case WaitLogDelay:
      /* Incrémente le compteur*/
      AcqDelayCounter++;

      /* Si le compteur expire */
      if (AcqDelayCounter >= ACQ_DELAY_MS)
      {
         /* Réinitialise le compteur */
         AcqDelayCounter = 0;
         /* Va à AcquireData */
         AdcSmState = AcquireData;
      }
      break;

   case AcquireData:
      /* Obtient les données */
      AdcDataRaw = ReadAdcData();
      /* Convertis les données en texte*/
      itoa(AdcDataBuffer,AdcDataRaw,10);

      /* Va à LogData */
      AdcSmState = LogData;
      break;

   case LogData:
      /* Si un log d'arrêt n'est pas reçu */
      if (!ReceiveEvt(&StopLog))
      {
         /* Log des évenements */
         GenerateEvt(&LogDataEvt);
         /* Va à WaitLogDelay */
         AdcSmState = WaitLogDelay;
      }
      else
      {
         /* Arrêt des logs des évenements */
         GenerateEvt(&CloseLogFile);
         /* Va à Idle */
         AdcSmState = Idle;
      }
      break;

   default:
      break;
   }
}
```

UsbTask	10 ms
LogTask	50 ms
AdcTask	100 ms
LedTask	100 ms

Tableau 1 : périodicités des tâches.

La table des tâches du scheduler est reportée dans le listing 3. Du point de vue de la périodicité, les valeurs indiquées dans le tableau 1 sont utilisées.

Passons maintenant à l'analyse plus détaillée de certaines tâches, c'est-à-dire les tâches « AdcTask » et « LogTaskcar » afin de comprendre comment elles ont été développées, en commençant par « AdcTask ». Les deux autres tâches « UsbTask » et « LedTask » effectuent principalement des opérations de service. Cette tâche a pour but d'acquérir les valeurs du canal analogique et d'envoyer les données associées (déjà converties au format texte) à la tâche qui prend en charge l'enregistrement.

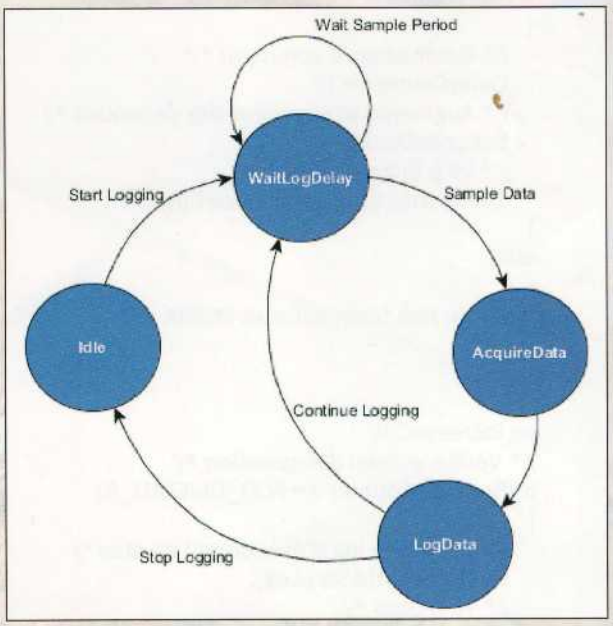


Figure 13 : Machine d'état « AdcStateMachine ».

Comme d'habitude, nous optons pour une structure de type machine d'état finie, dont l'implémentation est présentée dans le listing 4. Comme vous pouvez le voir, il existe **quatre états** : « **Idle** », « **WaitLogDelay** », « **AcquireData** », « **LogData** ».

Initialement, la machine vérifie si l'événement « LoggerReady » a été reçu, ce qui est réalisé par la tâche « LogTask », dès que le périphérique MSD a été énuméré et correctement initialisé. Cela empêche la création de logs d'événements avant que la mémoire ne soit prête à les mémoriser.

Listing 5 : implémentation de la machine d'état « LogControl ».

```
/*************************************************
* Fonction :      void LogControl (void)
* Input :         Non
* Output :        UINT8
* Auteur :        F.Ficili
* Description :   Génère un événement d'arrêt du log (journal)
* Date :          19/04/18
*************************************************/
UINT8 LogControl (void)
{
  /* État de la machine d'état */
  static SecCounterStateType SecCounterState = WaitSecDelay;
  /* Compteur de retard */
  static UINT16 DelayCounter = 0;
  /* Compteur des secondes */
  static UINT16 SecondsCounter = 0;

  switch (SecCounterState)
  {
    case WaitSecDelay:
      /* Augmente le compteur */
      DelayCounter++;
      /* Vérifie le compteur */
      if (DelayCounter >= SECOND_DELAY_MS)
      {
        /* Réinitialise le compteur */
        DelayCounter = 0;
        /* Augmente le compteur des secondes */
        SecondsCounter++;
        /* Va à IncreaseCnt*/
        SecCounterState = IncreaseCnt;
      }
      else
      {
        /* Ne rien faire, satisfait MISRA */
      }
      break;

    case IncreaseCnt:
      /* Vérifie le délai d'acquisition */
      if (SecondsCounter >= ACQ_TIMEOUT_S)
      {
        /* Génère le log d'évènement de stop*/
        GenerateEvt(&StopLog);
        /* Va à Stop */
        SecCounterState = Stop;
      }
      else
      {
        /* Reviens à WaitSecDelay */
        SecCounterState = WaitSecDelay;
      }
      break;

    case Stop:
      break;

    default:
      break;
  }
}
```

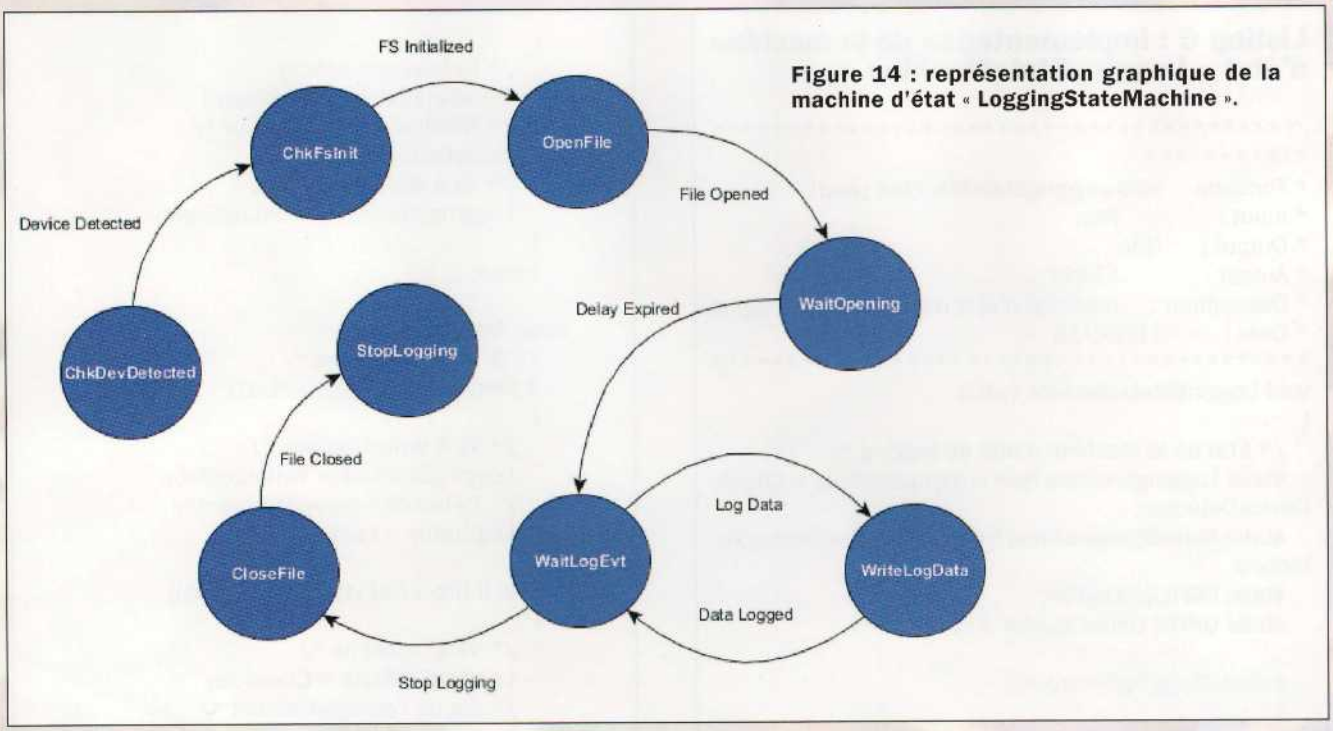


Figure 14 : représentation graphique de la machine d'état « LoggingStateMachine ».

À ce stade, la machine d'état entre dans un cycle dans lequel, à la fin de la période d'échantillonnage, les données sont acquises et converties en texte (« WaitLogDelay », « AcquireData »).

Ensuite, dans l'état « LogData », une vérification du log d'un événement d'arrêt est effectuée pour savoir s'il a été reçu ou non.

Si l'événement n'a pas été reçu, l'événement « LogDataEvt » est envoyé et l'acquisition continue (retour à l'état « WaitLogData »), sinon l'événement « CloseLogFile » est envoyé pour indiquer à la tâche de log de terminer et de fermer le fichier puis de retourner à l'état « Idle ».

La représentation graphique de la machine d'état relative à la fonction « AdcStateMachine » est illustrée en figure 13.

Examinons maintenant l'implémentation de la tâche « LogTask », qui implémente toutes les opérations liées au log des données sur la mémoire USB. Cette tâche implémente deux machines d'état. La première (LogControl) est chargée de contrôler l'activité du log (journal), en générant un événement d'arrêt après un certain temps.

La seconde (LoggingStateMachine) se charge d'enregistrer (logger) les données reçues de la tâche « AdcTask » sur la mémoire USB. Le code source de la machine d'état « LogControl » est reporté dans le listing 5.

L'implémentation est plutôt simple, il suffit de gérer un compteur afin de suivre le temps écoulé et générer l'événement d'arrêt lorsque le compteur dépasse une certaine « valeur » de temps.

Nous ne nous attardons pas et passons directement à l'analyse de la machine d'état « LoggingStateMachine » dont le code est reporté dans le listing 6.

Cette machine d'état tire parti de l'interface de haut niveau fournie par l'interface du système de fichier (File System Interface) afin de gérer directement la mémoire. Les API utilisées par cette tâche sont les suivantes :

- **BYTE USBHostMSDSCSIMediaDetect(void)** : vérifie si un périphérique MSD-SCSI est connecté au bus USB (et a été correctement énuméré) ;
- **int FSInit(void)** : initialise le système de fichier ;
- **FSFILE * FSfopen(const char * fileName, const char *mode)** : ouvre un fichier (comme indiqué) et renvoie un pointeur vers le flux de données ;
- **size_t FSfwrite(const void *data_to_write, size_t size, size_t n, FSFILE *stream)** : écrit le tableau des données transmis au fichier pointé par le flux ;
- **int FSfclose(FSFILE *fo)** : ferme le fichier transmis avec la référence « fo ».

En utilisant ces API de haut niveau, il est relativement facile d'écrire une application d'enregistrement de données. Comme cela apparaît dans le listing, 8 états ont été identifiés, définis par le type énumératif indiqué dans le listing 7.

Les deux premiers états de la machine vérifient si le périphérique est connecté et a été correctement énuméré et si le système de fichiers est initialisé.

Listing 6 : implémentation de la machine d'état « LoggingStateMachine ».

```
/************************************
**********
* Fonction :   void LoggingStateMachine (void)
* Input :          Non
* Output :     Non
* Auteur :         F.Ficili
* Description :    machine d'état de la tâche de logging
* Date :       11/04/18
************************************/
void LoggingStateMachine (void)
{
  /* État de la machine d'état de logging */
  static LoggingSmStateType LoggingSmState = Check-
DeviceDetected;
  static MassStorageStatusType MsStatus = DeviceDe-
tached;
  static FSFILE *LogFile;
  static UINT8 DelayCounter = 0;

  switch (LoggingSmState)
  {
    case CheckDeviceDetected:
      /* Vérifie l'état du périphérique */
      if(USBHostMSDSCSIMediaDetect())
      {
        /* Périphérique attaché */
        MsStatus = DeviceAttached;
        /* Va à CheckFsInitialized */
        LoggingSmState = CheckFsInitialized;
      }
      else
      {
        /* Périphérique détaché */
        MsStatus = DeviceDetached;
      }
      break;

    case CheckFsInitialized:
      /* Si le système de fichier est initialisé */
      if(FSInit())
      {
        /* Va à OpenFile */
        LoggingSmState = OpenFile;
      }
      break;

    case OpenFile:
      /* Ouvre le fichier */
      LogFile = FSfopen("LogFile.txt",FS_WRITE);
      /* Va à WaitOpening */
      LoggingSmState = WaitOpening;
      break;

    case WaitOpening:
      /* Incrémente le compteur */
      DelayCounter++;
      /* Si le compteur expire */
      if (DelayCounter >= OPENING_FILE_DELAY_MS)
      {
        /* Événement prêt */
        GenerateEvt(&LoggerReady);
        /* Réinitialise le compteur */
        DelayCounter = 0;
        /* Va à WaitLogEvent */
        LoggingSmState = WaitLogEvent;
      }
      break;

    case WaitLogEvent:
      /* Si log de données */
      if (ReceiveEvt(&LogDataEvt))
      {
        /* Va à WriteLogData */
        LoggingSmState = WriteLogData;
        /* Début de l'enregistrement*/
        LogStatus = Logging;
      }
      else if (ReceiveEvt(&CloseLogFile))
      {
        /* Va à CloseFile */
        LoggingSmState = CloseFile;
        /* Fin de l'enregistrement */
        LogStatus = NotLogging;
      }
      else
      {
        /* Ne rien faire */
      }
      break;

    case WriteLogData:
      /* Écris les données sur la clé USB */
       FSfwrite(AdcDataBuffer,1,ADC_BUFFER_LEN-
GHT,LogFile);
      /* Insère une nouvelle ligne */
      FSfwrite("\r\n",1,2,LogFile);

      /* Reviens à WaitLogEvent */
      LoggingSmState = WaitLogEvent;
      break;

    case CloseFile:
      /* Ferme le fichier */
      FSfclose(LogFile);
      /* Va à Stop */
      LoggingSmState = StopLogging;
      break;

    case StopLogging:
      /* Ne rien faire */
      break;

    default:
      break;

  }
}
```

Raw Data	Voltage [V]
1023	3,30
1023	3,30
1023	3,30
1011	3,26
894	2,88
815	2,63
676	2,18
679	2,19
737	2,38
826	2,66
895	2,89
975	3,15
895	2,89
792	2,55
679	2,19
596	1,92
534	1,72
384	1,24
272	0,88
146	0,47
54	0,17
125	0,40
187	0,60
254	0,82
307	0,99

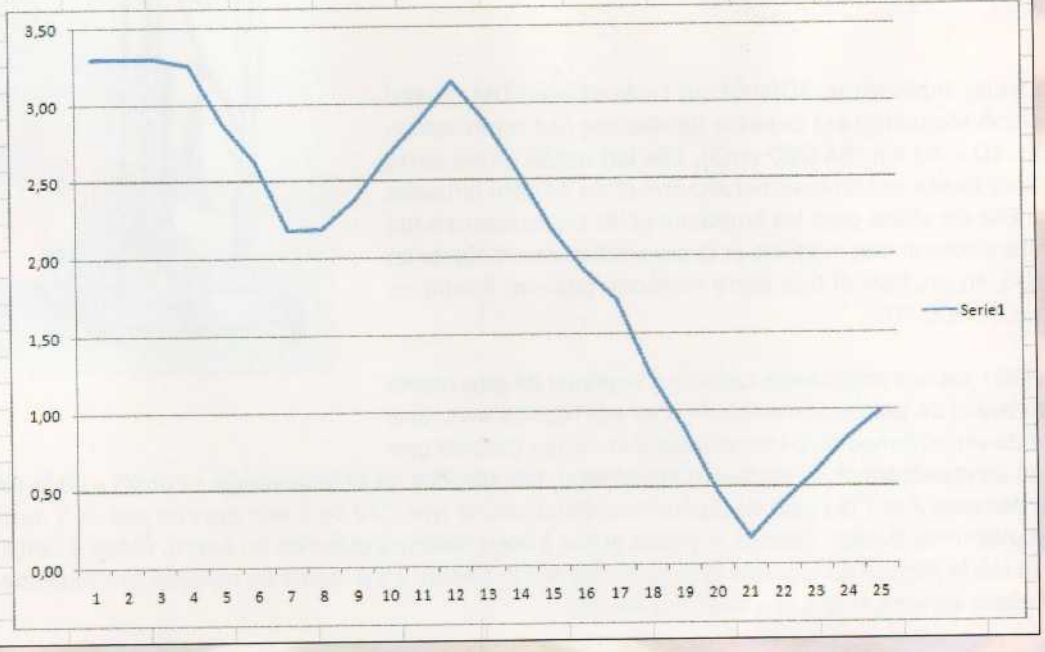


Figure 15 : graphique de l'acquisition généré avec Excel.

Listing 7 : typedef (type de structure) pour définir le type énumératif LoggingSmStateType.

```
/* État de la machine d'état de logging typedef */

typedef enum LoggingSmStateEnum
{
    CheckDeviceDetected,
    CheckFsInitialized,
    OpenFile,
    WaitOpening,
    WaitLogEvent,
    WriteLogData,
    CloseFile,
    StopLogging,
} LoggingSmStateType;
```

Le fichier est ensuite ouvert à l'aide de l'API « FSfopen » et un retard est généré pour donner le temps au pilote client d'effectuer l'opération.

À ce stade, la machine attend les événements de log provenant de la tâche Adc. Si un événement de log est reçu, les données sont enregistrées et la machine retourne à l'état de détection (WaitLogData), sinon elle se ferme et passe à l'état final (StopLogging).

Comme vous pouvez le noter, l'implémentation (représentée sous forme graphique en figure 14) est assez simple, car nous exploitons pleinement le potentiel des infrastructures sur lesquelles l'application est prise en charge, c'est-à-dire le scheduler et la stack USB host. Ces derniers font la majeure partie du travail, permettant ainsi à l'application de fonctionner à un niveau élevé.

Pour tester notre application, téléchargez simplement l'application sur la demoboard du cours, reliez une clé USB au port USB host et mettez-la sous tension.

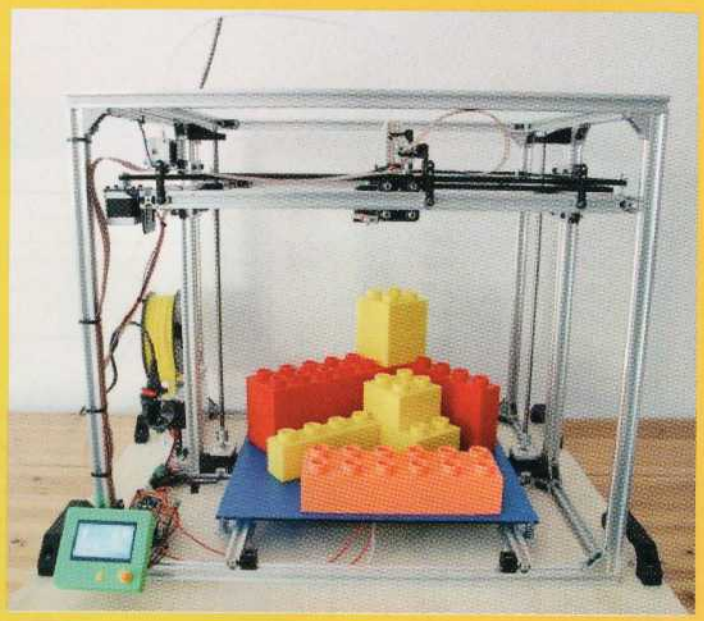
L'application, une fois le périphérique USB énuméré, démarrera l'activité de logging (enregistrement) qui se terminera après un certain temps (environ 30 secondes). La figure 15 montre le graphique de l'acquisition généré avec Excel.

Conclusion

Dans cette leçon, nous avons analysé en détail la partie host de la stack USB de Microchip et nous avons créé une application d'enregistrement de données très complexe. Dans la prochaine leçon, nous analyserons une autre pile complexe et puissante de Microchip, la pile TCP/IP. ■

3DRAG+

imprimez en grand
40 cm x 40 cm x 40 cm



La nouvelle imprimante 3DRAG+ en technologie FDM (Fused Deposition Modeling) est capable de réaliser des impressions de 40 x 40 x 40 cm (64 000 cm3). Elle est dotée d'une carte contrôleur basée sur du matériel Arduino et du célèbre firmware Marlin. Elle est idéale pour les amateurs et les professionnels qui souhaitent obtenir des modèles et des reproductions d'objets en PLA, ABS, Nylon, bois et tout autre matériau pouvant fondre en dessous de 300 °C.

La 3DRAG+ est une imprimante capable d'imprimer de gros objets tout en ayant de petites dimensions. Elle est fournie avec une plaque de verre trempé de 6 mm d'épaisseur, ce qui garantit une planéité sans précédent du plateau d'impression. Elle dispose de la technologie « CoreXY » où la tête d'impression se déplace le long des axes X et Y (à l'aide de 2 courroies distinctes de type GT2 de 6 mm avec un pas de 2 mm) permettant ainsi d'obtenir des mouvements fluides, rapides et précis grâce à deux moteurs externes au cadre. Grâce à cette technologie et au système Bowden (où le moteur qui pousse le fil se trouve sur le cadre), il y a moins de masses en mouvement, cela permet d'atteindre des vitesses élevées et une précision supérieure.

La structure de l'imprimante est entièrement réalisée avec des profilés en aluminium de type « V-slot », faciles à assembler. De plus, grâce à des poulies spéciales en POM (Polyoxyméthylène), permettant de réaliser des systèmes de déplacement linéaire d'une grande précision et d'une longue durée de vie, la structure de l'imprimante est à la fois robuste et légère. L'axe Z se déplace verticalement grâce à deux montants en aluminium de section 20 x 40 mm. Deux moteurs pas à pas Nema17 gèrent le mouvement de l'axe Z. La 3DRAG+ introduit également des innovations au niveau de l'extrudeuse en utilisant une tête d'impression très légère de type E3D V6 pour des filaments de 1,75 mm. Cela permet d'obtenir de très faibles masses en mouvement et donc des accélérations plus élevées et des vibrations moindres. De plus, l'utilisation de cette extrudeuse vous permet de choisir parmi une large gamme de buses : de 0,3 mm pour ceux qui ont besoin d'impressions très détaillées, jusqu'à 0,8 mm pour ceux qui souhaitent réduire les temps d'impression. a 3DRAG+ peut être personnalisée grâce à un écran graphique (en option) vous permettant de gérer directement les impressions sans connecter un ordinateur.

N.B. L'imprimante est fournie sans écran pour l'impression autonome (en option) et sans la plaque chauffante (en option).

Caractéristiques techniques de la 3DRAG+

- Dimensions d'impression : 400 mm x 400 mm x 400 mm ;
- Matériels utilisables : ABS et PLA ;
- Mouvements : technologie de type Core XY ;
- Diamètre du filament : 1,75 mm ;
- Diamètre de la buse fournie : 0,4 mm ;
- Diamètre des buses en option : de 0,3 mm à 0,8 mm ;
- Vitesse maximale d'impression : 350 mm/s ;
- Plateau d'impression fixe en verre trempé de 6 mm ;
- 5 moteurs pas à pas (200 pas) ;
- Structure : profilés en aluminium V-slot ;
- Résolution mécanique de l'axe Z : 0,625 microns ;
- Résolution mécanique des axes X et Y : 0,0125 mm (12 microns) ;
- Plateau chauffant : 40 x 40 cm - 12V/240W - avec adhésif 3M (en option) ;
- Alimentation à découpage : 220VAC /12VDC 350W ;
- Dimensions hors tout : 690 mm x 630 mm x 610 mm ;
- Firmware : Marlin personnalisé pour 3DRAG+ ;
- Logiciel : compatible avec le logiciel Open Source RepetierHost ;
- Systèmes d'exploitation pris en charge par RepetierHost : Windows, Mac OS et Linux.

3DRAG+ en kit non montée
Référence : 3D4040
Prix : 1022,00€

3DRAG+ Version montée
Référence : 3D4040/KM
Prix : 1328,00€

Comelec 12/2018

COMELEC CD 908 - 13720 BELCODÈNE FRANCE
Tél : 04 42 70 63 90 Fax : 04 42 70 63 95 contact@comelec.fr
SAS au capital de 7622,45 € RCS de Marseille TVA N° FR48399518083

Caractéristiques techniques et vente en ligne sur :
www.comelec.fr

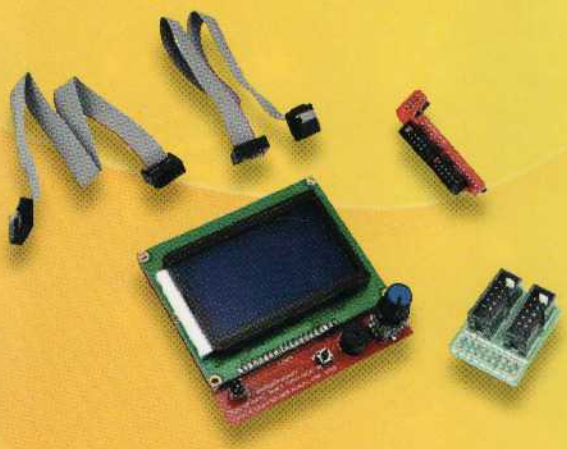
Tête d'impression (extrudeuse) pour la 3DRAG+

Tête d'impression complète (E3D-V6) pour la 3DRAG+, utilisable avec des filaments de 1,75 mm. Elle est équipée d'un corps en aluminium, d'une buse en laiton de diamètre 0,4 mm, d'un chauffage de 40 W, d'une CTN de 100 kΩ et d'un tube en PTFE (téflon) d'environ 1 mètre de long ainsi que les câbles de raccordement d'environ 90 cm de long. Masse : environ 88 g.

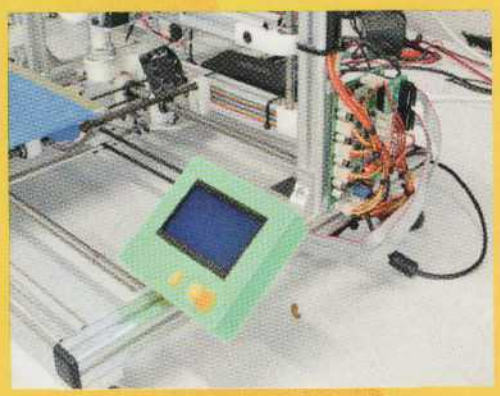
Tête d'impression complète

Référence : E3DV6SET *Prix : 14,00€*

Contrôleur pour impression autonome avec écran graphique



Ce contrôleur est équipé d'un encodeur rotatif, d'un écran LCD graphique rétroéclairé de 128 x 64 pixels et d'un logement pour une carte SD. Cette dernière permet de stocker des fichiers et de les imprimer directement sans ordinateur. L'afficheur permet de visualiser en temps réel, la température, l'accélération, la vitesse, le débit, le préchauffage, le fonctionnement des moteurs et de l'extrudeuse. Le contrôleur est alimenté directement à partir de l'imprimante. La carte est fournie déjà assemblée et testée, avec deux câbles plats et deux adaptateurs.



Contrôleur pour impression autonome

Référence : ET1147K

Prix : 32,80€

Ruban chauffant en Kapton 40 x 40 cm pour la 3DRAG+

Ruban chauffant à appliquer, à l'aide d'un adhésif 3M fourni, sous la vitre du plateau d'impression. Grâce à ce ruban chauffant, il sera possible d'atteindre, sur la plaque de verre, une température maximale d'environ 80 °C (en fonction de la température ambiante) permettant ainsi une meilleure adhérence de l'objet à imprimer. Il est alimenté en 12 V et consomme environ 22 A. Matériau : kapton, taille de la zone d'impression (en mm) : 400 x 400. Pour un fonctionnement correct, il est nécessaire de connecter un CTN de 100 kΩ.

Ruban chauffant

Référence : 3D4040HPLATE

Prix : 64,90€

Relais statique pour la 3DRAG+

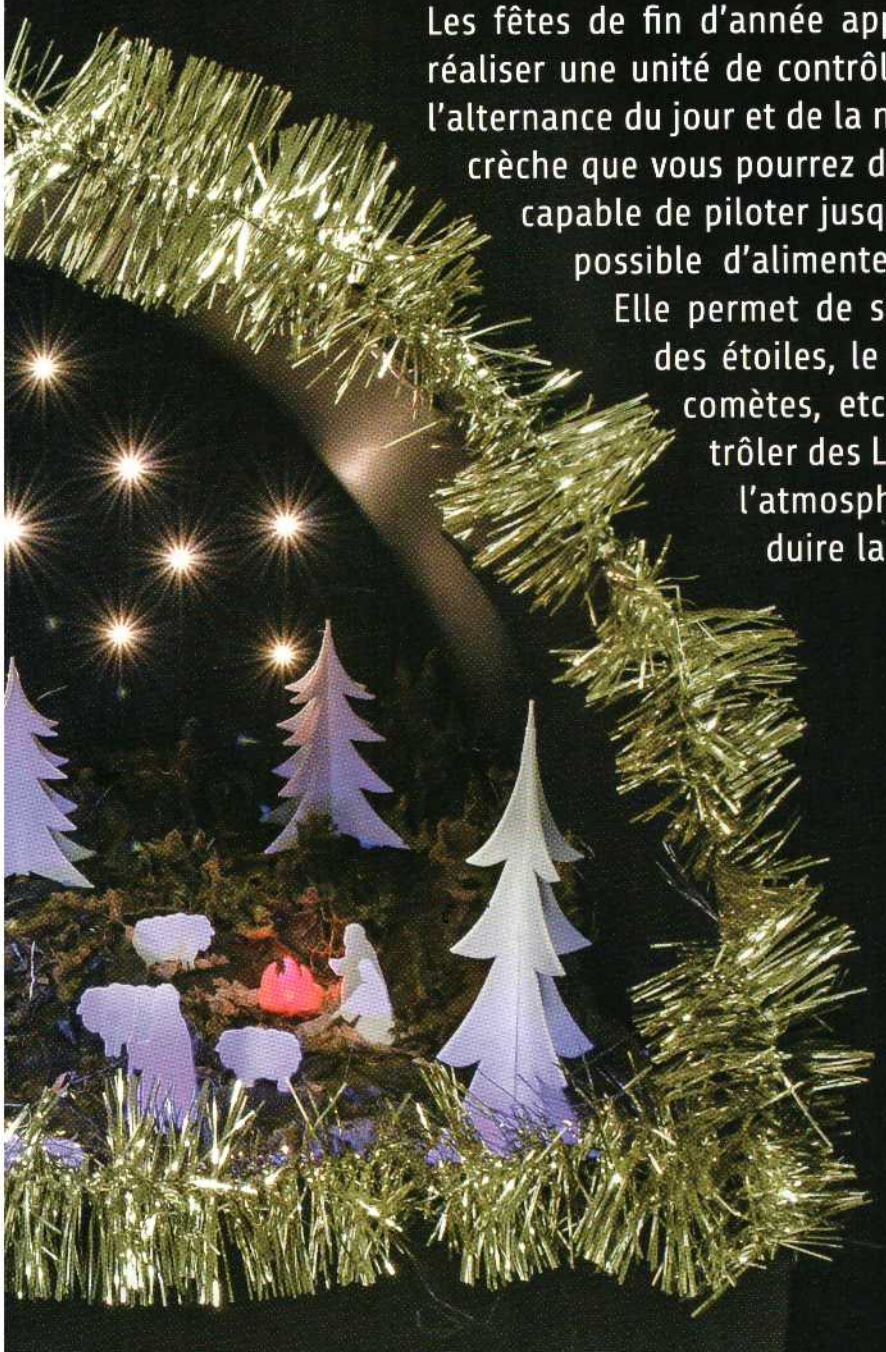
Ce relais permet de piloter le plateau chauffant avec un courant maximal d'environ 30 A sous une tension de 12 V. Il comporte un mode de sélection du fonctionnement de la logique du signal de contrôle. L'emballage comprend également 4 cosses Faston à sertir sur les câbles pour effectuer les connexions, ainsi que deux écrous de fixation au châssis de l'imprimante.

Relais statique

Référence : ET1357K *Prix : 16,90€*

UNE CRÈCHE AVEC ARDUINO

de Boris Landoni



Les fêtes de fin d'année approchant, nous vous proposons de réaliser une unité de contrôle capable de simuler cycliquement l'alternance du jour et de la nuit afin de recréer l'ambiance d'une crèche que vous pourrez décorer à votre goût. La centrale est capable de piloter jusqu'à six sorties avec lesquelles il est possible d'alimenter des LED ou des petits moteurs. Elle permet de simuler la lumière du jour, la lueur des étoiles, le feu des cheminées, la présence de comètes, etc. Cette unité peut également contrôler des LED de type NeoPixel et, pour recréer l'atmosphère de Noël, elle permet de reproduire la musique de votre choix.

Comme chaque année, à l'approche des fêtes de Noël, nous pensons à un projet thématique qui peut rendre cette période originale et plus chaleureuse. Comme nous ne proposons plus depuis quelques années ce qui est l'un des classiques de l'électronique, le contrôle des lumières d'une crèche, nous avons décidé d'opter pour une unité similaire mais dans une version plus moderne. Pour plus de simplicité, nous avons opté pour un matériel connu et apprécié de milliers d'expérimentateurs, à savoir Arduino.

C'est ainsi que ce projet est né, il s'agit d'un contrôleur de lumière et de son pour crèche. Le circuit est basé sur un microcontrôleur ATmega 328 (cela correspond à une carte Arduino UNO) qui contrôle directement des LED. De plus, la centrale fonctionne à basse tension et peut donc être mise à la portée des enfants, sans aucun danger pour leur santé. Aujourd'hui, avec la grande disponibilité des LED sous forme de bandes, il est très simple de fabriquer une crèche avec un berceau.

En plus du contrôle des LED, notre centrale permet de gérer un signal audio, grâce à un module lecteur MP3 innovant monté sur la carte. Il est capable de reproduire dans un haut-parleur (en monophonie) des morceaux de musique de votre choix à partir d'une carte SD, afin de recréer l'ambiance de Noël.

Le projet

Examinons d'abord les fonctions disponibles qu'offre notre centrale, elle se différencie des systèmes que l'on peut trouver dans le commerce qui ne permettent , dans la grande majorité, qu'un simple contrôle de lumières qui simulent le lever et le coucher de soleil.

Avec notre centrale, il est possible d'effectuer un contrôle d'une petite ou d'une grande crèche que se trouve dans les églises, dans des communautés locales, etc.

Nous avons donc créé un circuit très performant, capable de piloter quatre charges lumineuses à LED fonctionnant sous 12 V et pouvant fournir individuellement un courant de 6 A. C'est également le courant maximum que le circuit imprimé permet, sinon il faudra étamer abondamment les pistes de cuivre qui vont du bornier d'alimentation aux transistors MOSFET ainsi que celles des borniers de sortie.

Par courant individuel, nous entendons que chaque canal peut commuter jusqu'à 6 A, mais la façon dont le circuit

imprimé est construit ne permet pas de faire fonctionner de manière continue toutes les sorties à 6 A.

Ces valeurs, étant donné que le circuit est conçu pour piloter des bandes LED ou des compositions de LED, sont plus que suffisantes pour une crèche de maison, et même pour une structure extérieure, grâce au rendement lumineux des LED.

Les sorties de l'unité de contrôle sont au nombre de six, le programme que nous fournissons permet la gestion des quatre premières sorties pour la mise en œuvre de la simulation de la lumière du jour, des étoiles, des feux de cheminées et de la comète. Les lumières s'allument et s'éteignent progressivement suivant un cycle qui simule une journée entière.

À ces sorties de base, vous pouvez ajouter, en modifiant le sketch de l'ATmega 328, les deux autres sorties auxquelles vous pouvez assigner le contrôle de divers accessoires de la crèche comme les mouvements des personnages ou des animaux, etc.

En ce qui concerne les sorties de base, nous avons divisé la séquence complète en quatre phases appelées : jour, coucher de soleil, nuit et lever de soleil. La durée des phases peut être ajustée à l'aide d'un trimmer et l'excursion correspondante peut être réinitialisée en intervenant sur le sketch.

Ceci est donc valable pour le jour et la nuit, mais aussi pour les deux phases de transition (coucher de soleil et lever de soleil). Ces deux dernières sont les phases les plus suggestives.

Pendant le coucher de soleil, la luminosité du jour (réalisée avec des LED connectées sur la sortie « SOLE ») diminue peu à peu, tandis que dans le ciel les étoiles commencent à apparaître (la sortie correspondante est activée).

À un certain moment, avant la fin du cycle, les feux de la cheminée s'allume (sortie « FUOCO »).

Notre circuit est capable de simuler le scintillement du feu de cheminée en pilotant la sortie « FUOCO » avec une séquence d'impulsions appropriée.

Lorsque toutes les étoiles sont complètement illuminées, la comète apparaît grâce à l'activation de la sortie « COMETA ».

Évidemment, à l'exception de petits détails, durant la période de l'aube, toutes les lumières brillantes s'éteignent progressivement et la lumière du jour augmente lentement jusqu'à ce que la luminosité maximale soit atteinte.

Le schéma électrique

Passons maintenant aux aspects plus techniques, en analysant le schéma électrique de l'unité de contrôle qui est basée sur un microcontrôleur Atmega 328 qui gère de manière appropriée les sorties ainsi que le module MP3 qui reproduit le son de la crèche.

Le microcontrôleur U1, lors de la phase d'initialisation, configure la broche PB0 en tant que sortie pour envoyer les commandes séries à une bande de LED Neopixel. Les broches PB1, PB2, PB3, PD3, PD5, PD6 sont aussi configurées en tant que sorties afin de commander les transistors MOSFET de type canal N (les transistors sont des modèles STP36N06 avec un I_D de 36 A et une V_{DS} de 60V) qui commutent physiquement l'alimentation de la ou des LED correspondante(s).

Les broches du microcontrôleur utilisées pour gérer les sorties pilotent chacune la grille d'un MOSFET, à travers une résistance de 1 kΩ, et une LED avec sa résistance de limitation de courant. Cela permet de surveiller l'état des transistors.

Chaque sortie génère des impulsions rectangulaires compatibles TTL de fréquence constante et de largeur variable, selon la technique PWM. Cela permet de faire varier la valeur moyenne de la tension appliquée par les MOSFETS à leurs charges respectives et par conséquent cela permet de faire varier la puissance et la luminosité avec un très bon rendement puisque la résistance à l'état passant (ON) du MOSFET est très faible (seulement 40 mΩ).

Ainsi, la puissance dissipée est très faible par rapport à une commande de type linéaire (transistor ballast), où le transistor devrait dissiper la différence entre la tension d'alimentation et celle appliquée à la charge.

Les signaux PWM sont générés par les modules PWM internes du microcontrôleur, configurés de manière appropriée par le programme pour chaque sortie. La configuration du circuit prévoit d'alimenter les bandes de LED en reliant respectivement les bornes « positives » et « négatives » aux points « + » et « - » des sorties relatives.

Cependant, comme les MOSFET peuvent supporter une tension drain-source de 60 V, il est possible d'alimenter les bandes de LED avec une tension de 24 VDC à 36 VDC afin de réduire le courant circulant, tout en obtenant la même puissance lumineuse. Par exemple, pour la sortie « OUTEX1 », le « + » est relié au + 12V, il est possible d'appliquer une tension 24 VDC à 36 VDC en isolant au préalable la piste de cuivre (en fonction de votre utilisation), car il ne faut pas appliquer une tension de 36 VDC sur l'anode de D1.

Si vous utilisez une alimentation externe, le « + » de la bande LED doit être relié au positif de l'alimentation choisie et le « - » à la sortie correspondante de la centrale (drain du MOSFET correspondant), à condition que les masses de l'alimentation et de la carte soient reliées ensemble.

Nous avons déjà dit que la séquence complète générée par notre centrale se compose de quatre phases (jour, coucher de soleil, nuit et lever de soleil) dont la durée peut être réglée indépendamment au moyen de trimmers reliés au 5 V et dont la tension présente sur leur point milieu est lue par l'entrée analogique correspondante du microcontrôleur via le convertisseur analogique/numérique interne de l'ATmega 328.

Dans notre projet, l'ADC est affecté aux broches PC0 à PC5 grâce auxquelles les tensions des 6 trimmers sont lues de manière séquentielle. Les trimmers R1, R2, R3, R4 gèrent les fonctions précitées et les trimmers R5 et R6 sont disponibles pour des fonctions assignables à volonté en fonction du programme du microcontrôleur.

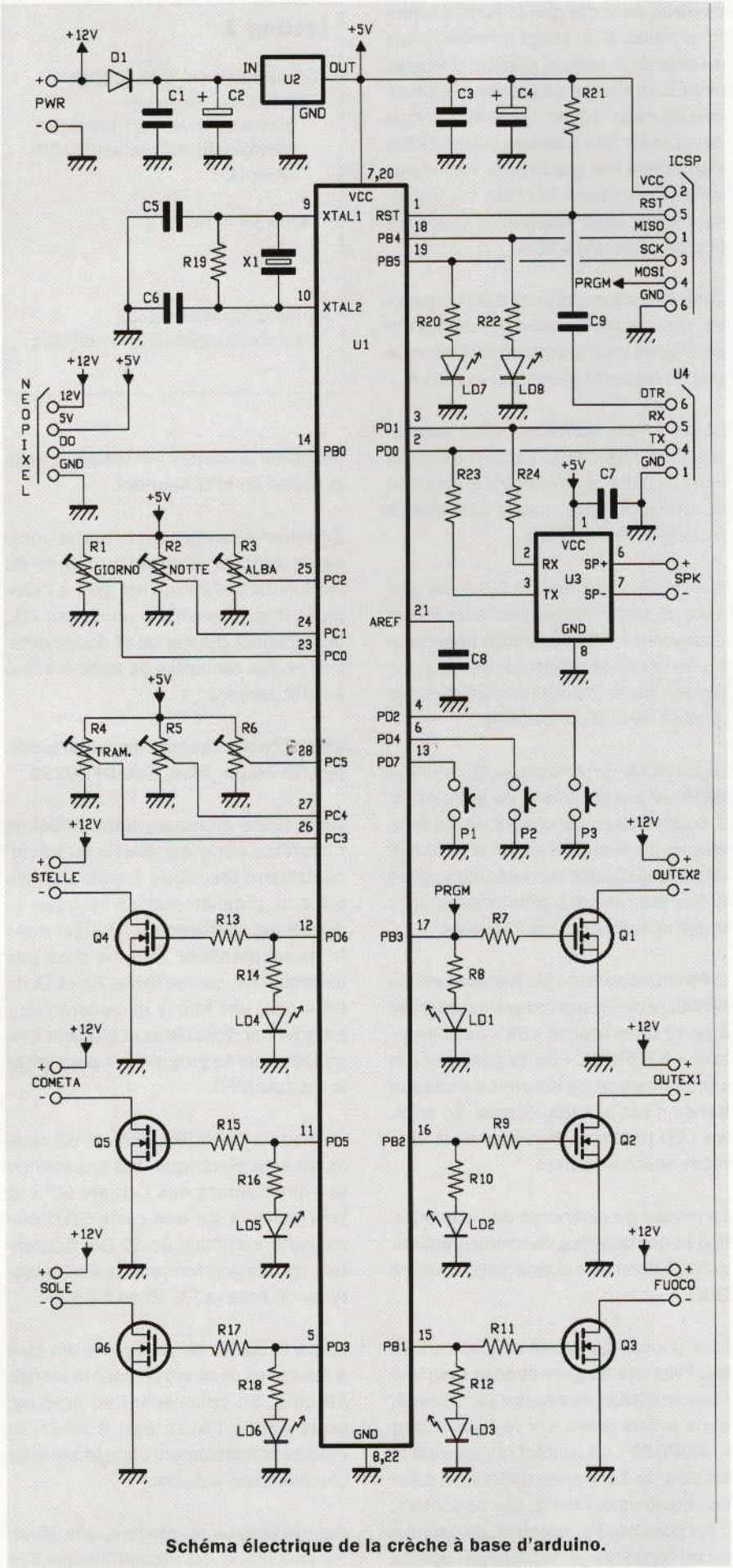


Schéma électrique de la crèche à base d'arduino.

La durée des cycles est directement proportionnelle à la tension appliquée par le trimmer relatif, ainsi en amenant le curseur de R1 vers la masse la durée du jour est raccourcie. À l'inverse, vers le 5 V, la durée du jour est augmentée.

Attention cependant à un détail particulier, les trimmers R1, R2, R3, R4 ne contrôlent pas directement les sorties PWM des différentes phases, mais déterminent les temps d'exécution. Les rapports cycliques des signaux PWM des sorties « SOLE », « STELLE », « COMETA » et « FUOCO » sont gérés en fonction des périodicités définies par les routines correspondantes du programme.

Ce fonctionnement est logique car les phases décrites impliquent plusieurs sorties. Dans le cas, par exemple, de la transition « jour/nuit », les sorties « SOLE » et « STELLE » sont impliquées. Elles s'affaiblissent dans ce cas, mais au contraire elles augmentent lors du passage « nuit/jour ».

Les trimmers R1 à R4 sont donc utilisés pour gérer la durée du jour (R1), de la nuit (R2), du lever de soleil (R3) et du coucher de soleil (R4). La LED LD8, pilotée par la broche PB4, s'allume brièvement lorsque l'unité de contrôle passe d'une phase à l'autre.

La durée de chaque phase est définie à l'aide du trimmer relatif et la durée maximale de chaque phase est d'environ 100 secondes. Pour être précis, la durée est obtenue en lisant l'ADC sur 10 bits (entre 0 et 1023) en considérant les valeurs en millisecondes et en les multipliant par 100 au niveau du firmware.

Par exemple :
1023 ms x 100 = 102300 ms = 102 s

Si vous désirez modifier la durée, vous pouvez éditer le sketch et modifier la partie qui détermine la temporisation en vous référant au Listing 1. Il suffit de modifier la ligne :

```
trimvalue[i]=trimvalue[i]*100;
```

Ici, vous pouvez modifier la valeur 100 en la remplaçant par ce que vous voulez. Par exemple, en entrant 200, vous obtenez un doublement de la durée entre deux phases, tandis qu'avec une

valeur de 50 vous obtenez une division par 2 de la durée.

Les deux autres trimmers peuvent être configurés par firmware, ainsi que les sorties « OUTEX1 » et « OUTEX2 ». Avec notre programme (sketch), le trimmer R5 est libre, tandis que R6 est utilisé pour gérer le volume audio.

Les sorties MOSFET peuvent être connectées aussi bien à des bandes de LED qu'à de petites ampoules de 12 V, mais aussi à des LED de puissance ou encore à des moteurs électriques afin d'entraîner des roues, à d'autres mécanismes, ainsi qu'à des petites pompes pour générer l'écoulement d'une chute d'eau ou tout ce que votre imagination vous suggère pour créer une crèche plus « vivante ».

Restons sur le sujet des sorties, il convient d'aborder celle qui est dédiée au contrôle des LED NeoPixel, elle permet d'ajouter des animations plus lumineuses, telles que par exemple la queue de la comète.

Les bandes de LED NeoPixel sont gérées en parallèle avec une seule broche du microcontrôleur d'Arduino. Dans notre cas, cette broche est facilement configurable à partir du sketch. La communication est unidirectionnelle et gère un groupe de LED.

NeoPixel est une solution particulière de LED RVB « intelligente » pouvant être contrôlée par exemple via Arduino. Chaque LED NeoPixel dispose d'un contrôleur embarqué, qui peut être interfacé avec une seule ligne de données.

La librairie propriétaire mise à disposition par Adafruit (www.adafruit.com) facilite l'intégration avec l'environnement Arduino.

Chaque LED RVB peut être gérée individuellement via une commande spéciale insérée dans les données séries de contrôle et peut produire jusqu'à 256 nuances de couleurs, pour un total de 16 777216 combinaisons de couleurs.

Une particularité des LED NeoPixel est qu'elles peuvent être connectées en cascade de sorte que la ligne de données passe d'un étage à l'autre, mais au-delà d'un certain nombre d'étages de LED, la vitesse de gestion est considérablement réduite. De ce fait, si vous devez créer des matrices pour afficher rapidement des graphiques, vous devez utiliser plusieurs broches du microcontrôleur avec seulement quelques LED sur chacune d'elle.

Cette limitation n'affecte pas notre projet, car aucune fonction par défaut n'a été prévue pour la sortie NeoPixel, vous pouvez l'ajouter en éditant le sketch.

Le canal des données utilisé pour la communication avec les LED NeoPixel est similaire à une communication de type « oneWire », avec une vitesse maximale de 800 kbps.

Pour chaque bande de LED, il est possible de régler la fréquence de rafraîchissement, afin de rendre imperceptibles certains effets de lumière, en restant dans la limite imposée par le nombre de LED connectées.

Le protocole de commande du système NeoPixel prévoit l'envoi de groupes de 3 octets dans une chaîne de 24 bits, chacun contenant la valeur d'éclairage de chaque couleur de base (d'abord les 8 bits pour le vert, puis ceux pour le rouge et enfin ceux pour le bleu).

L'alimentation des LED NeoPixel est de 5 VDC, cette tension peut être prélevée à partir de la broche « 5V » du connecteur « NEOPIXEL » de la carte, car la consommation de courant de chaque bande n'est pas trop élevée. En effet, les LED tricolores NeoPixel sont allumées alternativement.

La masse de référence de l'alimentation et des données du microcontrôleur doivent donc être connectées au point GND.

Pour pouvoir gérer des bandes de LED NeoPixel très longues et donc avec une consommation de courant plus élevée, nous avons prévu sur le connecteur « NEOPIXEL » un contact qui apporte la tension de 12 V prise directement sur l'anode de la diode D1. Sur ce contact, il est possible, par exemple, de connecter un régulateur à découpage capable de fournir le courant nécessaire à toute la bande de LED NeoPixel.

Listing 1

```
if(runtime>time_trim+200){
  for (int i=0;i<6;i++){
    trimvalue[i]=(ReadTrimmer(i));
    trimvalue[i]=trimvalue[i]*100;
    delay (1);
  }
  time_trim=millis();
 }


int ReadTrimmer (int id){
 return(analogRead(trimmer[id]));
}
```

Examinons maintenant la partie audio de la crèche. L'audio provient du module DFR0299 qui est géré à l'aide d'une interface série à un niveau TTL. Vous pouvez trouver de la documentation et des exemples de code Arduino à cette adresse :

https://www.dfrobot.com/wiki/index.php/DFPlayer_Mini_SKU:DFR0299

Dans notre montage, nous utilisons l'interface série matérielle du microcontrôleur, identique à celle utilisée pour la programmation lorsque le sketch est « téléversé » (chargé) dans le microcontrôleur. Cela ne pose pas de problème, car les lignes RX et TX de l'ATmega, une fois le microcontrôleur programmé, sont libres et peuvent être utilisées par le programme pour gérer le module MP3.

Le module DFR0299, nommé U3 dans le schéma électrique, est capable de lire directement des fichiers MP3 et WAV stockés sur une carte SD d'une capacité maximale de 32 Go, à condition qu'elle soit formatée avec un système de fichiers FAT16 ou FAT32.

La particularité de ce module est qu'il a été conçu et construit pour le monde Arduino. En choisissant ce module, nous avons l'avantage d'avoir un module complètement compatible avec l'architecture Arduino.

Pour la gestion du module, une librairie spécifique est disponible dans le

Listing 2

```
void gestAudio(int track){

 digitalWrite (LEDPLAY,LOW);
 if (playing==true){
    digitalWrite (LEDPLAY,HIGH);
    track++;
    if (track==5) track=1;
     if (stopVol==false){
       if(runtime>time_volume_running+200){
         if(startVol==false){
          if (volume>3) {
            volume=volume-3;
            Serial.print("dec ");
            Serial.println(volume);
          }
          else
          {
            playtrack(track);
            startVol=true;
          }
         }
         else
         {
          if (volume<(trimvalue[TRIMMER_VOL]/3500)) {
            volume=volume+3;
           Serial.print("inc ");
           Serial.println(volume);
          }
          else
          {
            startVol=false;
            stopVol=true;
            Serial.println("fin de cycle");
          }
         }
        time_volume_running=millis();
        setvolume(volume);
        }
      }
    }
}
```

sommaire détaillé de la revue ainsi que les autres fichiers du projet (en téléchargement). Elle vous permet de choisir les fichiers à reproduire, ou encore d'effectuer un réglage du volume.

Le module contient un décodeur capable de décompresser l'audio au format MP3 et un contrôleur capable d'accéder via le bus SPI aux données contenues dans la carte SD.

Au fur et à mesure de la lecture du flux de données, le décodeur transforme les données décodées en un signal audio non compressé, qui est ensuite amplifié par un petit amplificateur BF intégré monophonique d'une puissance de 3 W, ce qui est largement suffisant pour sonoriser la pièce où se situe la crèche.

Si vous avez besoin de plus de puissance, utilisez les sorties audio « DAC_R » et « DAC_L » pour piloter un amplificateur de puissance. Ces sorties sont accessibles au niveau des broches 4 et 5 du module. Dans ce cas, vous obtenez un son stéréophonique.

Toutes les fonctions du module U3 sont gérées par des instructions spécifiques de type série provenant du microcontrôleur. Les commandes de volume sont générées par la lecture de la tension présente sur le curseur du trimmer R6 grâce au convertisseur ADC de l'ATmega.

Nous avons également prévu d'utiliser, parmi les commandes de gestion de la reproduction (Play, Stop, Pause, FWD, etc.) uniquement celle de lecture et d'arrêt (Play et Stop). Ces commandes sont implémentées à l'aide de 3 boutons reliés au microcontrôleur via les lignes PD2, PD4 et PD7 (toutes configurées en entrée avec leur résistance interne de pull-up activée) qui ont toutes la même fonction, à savoir activer/désactiver l'audio (voir le Listing 2).

Plus précisément, l'appui sur l'une des touches P1, P2, P3 lorsqu'une piste est en cours de lecture provoque son arrêt. Une pression successive fait reprendre la lecture depuis le début de la piste précédemment arrêtée.

Lorsque l'audio est en cours de lecture, la LED LD7, qui est reliée à la ligne PB5 du microcontrôleur configuré en sortie, est allumée. Les lignes PD0 et PD1, qui correspondent à l'UART, sont reliées au module U3 à travers des résistances pour éviter que lorsque le convertisseur série/USB est connecté à la carte, les tensions des deux modules interfèrent.

Les fichiers audio que vous voulez lire doivent être nommés sur la carte SD dans l'ordre suivant : 001.mp3, 002.mp3, 003.mp3 et 004.mp3.

En effet, le premier fichier est lu pendant la phase de « jour », le second pendant la phase de coucher de soleil, le troisième lorsque le feu est allumé et le quatrième pendant la phase de lever de soleil (aube).

Pendant le passage d'une phase à l'autre, le volume de lecture est progressivement réduit à la fin du morceau et le morceau suivant est joué avec le volume qui augmente progressivement.

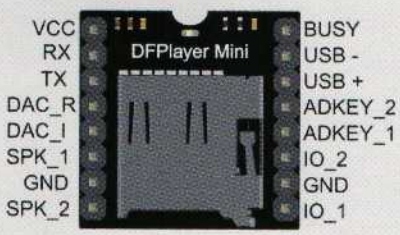
Nous complétons la description du circuit avec l'alimentation, l'oscillateur de l'ATmega 328P et la programmation en circuit. En ce qui concerne l'alimentation, le montage doit être alimenté aux points « +/- PWR » avec une tension comprise entre 12 VDC et 14 VDC.

Le module MP3 DFR0299

Le module que nous utilisons pour la reproduction audio nous permet de lire des morceaux de musique MP3 sans devoir utiliser du matériel coûteux et sans exigence technique concernant le traitement du signal. En fait, le module DFR0299 est un décodeur complet qui utilise un support de type carte SD. Il décrypte les fichiers audio MP3 et dispose d'un amplificateur BF intégré qui est relié directement à un haut-parleur. Sa puissance de sortie est de 3 W sous une impédance de 8 Ω, ce qui est plus que suffisant pour notre crèche maison.

Il est donc idéal pour lire des fichiers audio standard tels que des fichiers MP3, sans nécessiter de ressources informatiques importantes (y compris les CODEC) pour leur traitement. Ce module a été conçu pour être géré par Arduino, ce dernier se limite à l'envoi de commandes séries pour gérer la reproduction. Tout le reste est réalisé par le module DFR0299.

Ce module peut être utilisé de manière autonome, il s'alimente avec une tension de 5 VDC et consomme un courant de 0,45 A pour fournir 3 W sous 8 Ω. Il suffit de lui raccorder un haut-parleur et des boutons. Il peut aussi être utilisé en combinaison avec une carte Arduino UNO ou toute autre carte équipée d'un port série compatible TTL. Il se contrôle via des broches d'entrées/sorties de type série ou analogique.

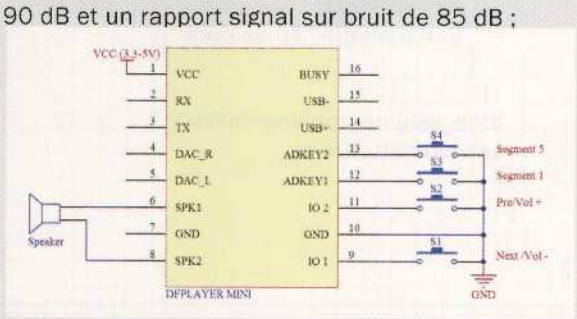


Brochage du module DFR0299.

Il prend en charge jusqu'à 100 dossiers, chaque dossier peut contenir jusqu'à 255 pistes MP3. Le réglage du volume, qu'il soit obtenu par une communication série ou à l'aide de boutons, prévoit 30 niveaux (en + et en - à partir de la position centrale). L'alimentation est comprise entre 3,2 VDC et 5 VDC, le courant consommé en veille est de 20 mA.

Les principales caractéristiques du module sont les suivantes :

- décodage MP3 11172-3 et ISO13813-3 layer3 audio decoding ;
- possibilité d'égalisation : Pop, Rock, Jazz, Classique ;
- fréquences d'échantillonnage en reproduction (kHz): 8/11,025/12/16/22,05/24/32/44,1/48 ;
- convertisseur DAC 24 bits, il supporte une plage dynamique de 90 dB et un rapport signal sur bruit de 85 dB ;
- prend en charge les systèmes de fichiers FAT16 et FAT32 ;
- capacité maximale de mémoire adressable : 32 Go (SD) ;
- sortie audio stéréo haute impédance pour amplificateur BF : DAC_R et DAC_L ;
- dimensions : 20 mm x 20 mm x 13 mm.

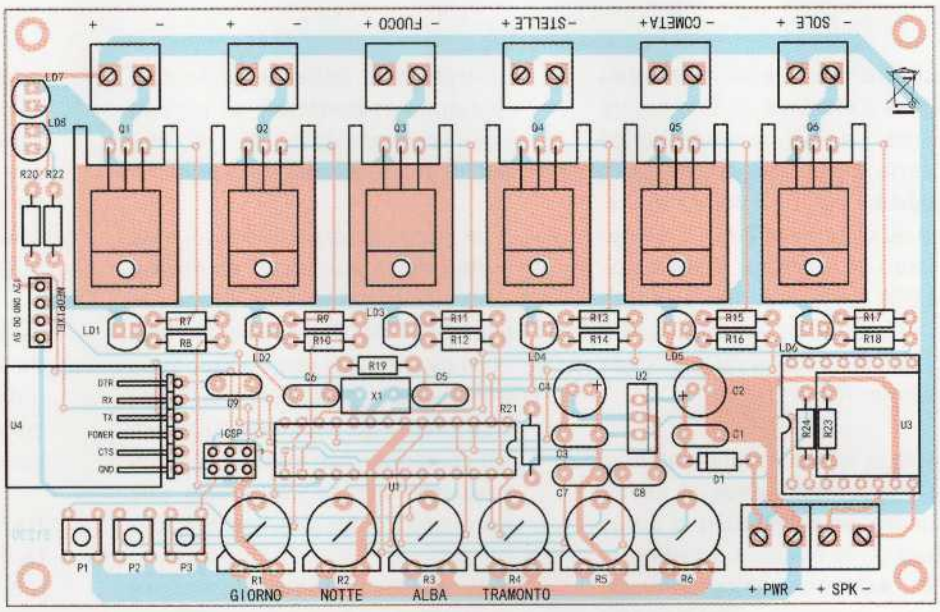


Le module peut être utilisé de manière autonome avec seulement 4 boutons. S1 et S2 ont la double fonction de contrôle du volume (pression longue) ou de saut entre les pistes (pression courte).

Bien que dans notre projet nous utilisons le module en mono, il dispose de sorties stéréo à haute impédance pouvant attaquer un amplificateur BF, ou encore une table de mixage, ou même des écouteurs de 300 Ω. L'amplificateur BF intégré est de type monophonique, il amplifie la somme des deux canaux gauche et droit. Le tableau ci-dessous décrit le brochage du module.

N°	Broche	Description	Note
1	VCC	Alimentation	de 3,2 VDC à 5 VDC Typiquement : 4,2 VDC
2	RX	Entrée UART série	
3	TX	Sortie UART série	
4	DAC_R	Sortie audio canal droit	Sortie droite haute impédance écouteur et ampli BF
5	DAC_L	Sortie audio canal gauche	Sortie gauche haute impédance écouteur et ampli BF
6	SPK2	- HP	Sortie HP - 3 W
7	GND	Masse	Masse de l'alimentation
8	SPK1	+ HP	Sortie HP + 3 W
9	IO1	Trigger port 1	Pression courte = piste précédente Pression longue = diminution du volume
10	GND	Masse	Masse de l'alimentation
11	IO2	Trigger port 2	Pression courte = piste suivante Pression longue = augmentation du volume
12	ADKEY1	AD Port 1	Trigger joue la 1ère piste
13	ADKEY2	AD Port 2	Trigger joue la 5ème piste
14	USB+	USB+ DP	Port USB
15	USB-	USB- DM	Port USB

Plan de montage de la crèche



Plan de câblage des composants de la carte.

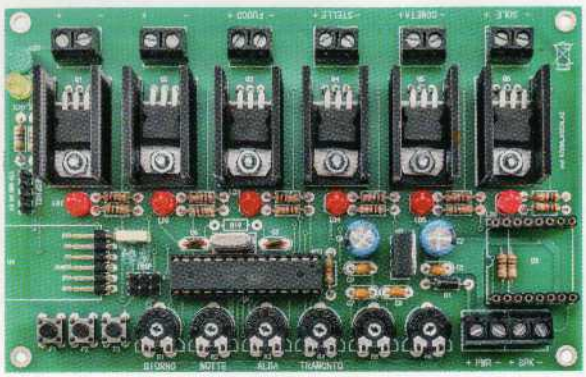


Photo de l'un de nos prototypes. En bas à droite, vous pouvez voir le connecteur sur lequel est enfiché le module DFR0299. À gauche, le connecteur sur lequel est connecté le convertisseur série/USB.

Liste des composants

R1..... trimmer 10 kΩ mono-tour
R2..... trimmer 10 kΩ mono-tour
R3..... trimmer 10 kΩ mono-tour
R4..... trimmer 10 kΩ mono-tour
R5..... trimmer 10 kΩ mono-tour
R6..... trimmer 10 kΩ mono-tour
R7..... 1 kΩ
R8..... 1 kΩ
R9..... 1 kΩ
R10.... 1 kΩ
R11.... 1 kΩ
R12.... 1 kΩ
R13.... 1 kΩ
R14.... 1 kΩ
R15.... 1 kΩ
R16.... 1 kΩ
R17.... 1 kΩ
R18.... 1 kΩ
R19.... NC
R20.... 470 Ω
R21.... 4,7 kΩ
R22.... 470 Ω
R23.... 1 kΩ
R24.... 1 kΩ
C1..... 100 nF céramique
C2..... 220 µF/25 V électrolytique
C3..... 100 nF céramique
C4..... 220 µF/25 V électrolytique
C5..... 15 pF céramique
C6..... 15 pF céramique
C7..... 100 nF céramique
C8..... 100 nF céramique
C9..... 100 nF céramique

D1..... 1N4007
U1..... ATMEGA328P
U2..... 7805
U3..... module DFPlayer Mini (DFR0299)
U4..... convertisseur USB/TTL (FTDI5V)
X1..... quartz 16 MHz
Q1..... STP36NF06
Q2..... STP36NF06
Q3..... STP36NF06
Q4..... STP36NF06
Q5..... STP36NF06
Q6..... STP36NF06
P1 à P3 microswitch
LD1 à LD6 LED rouge 5 mm
LD7.... LED verte 5 mm
LD8.... LED jaune 5 mm

Divers

Bornier 2 pôles au pas de 5,08 mm (x8)
Barrette femelle 8 pôles (x2)
Barrette mâle 3 pôles (x2)
Barrette mâle 4 pôles
Barrette mâle 6 pôles coudée à 90°
Support circuit intégré 2 x 14 broches
Dissipateurs (x6)
Vis 10 mm 3 MA (x6)
Ecrou 3 MA (x6)

NB : les typons des circuits imprimés à l'échelle 1 sont disponibles en téléchargement dans le sommaire détaillé de la revue.

Fonctionnement des sorties

La séquence d'éclairage des lumières s'effectue de la manière suivante. Le jour, la seule lampe (ou série de lampes ou de LED) allumée est celle qui simule la présence du soleil c'est-à-dire la phase de « jour » de la crèche. Cette lampe (ou LED) reste allumée pendant une durée comprise entre 0 et 102 secondes en fonction de la position du trimmer R1.

Une fois le temps écoulé, une nouvelle phase commence, à savoir le coucher de soleil, dont la durée dépend de la position du trimmer R4. En agissant sur ce trimmer, une valeur comprise entre 0 et 102 secondes peut être déterminée.

Peu à peu, la lumière du jour est atténuée au fur et à mesure que la luminosité des lampes simulant les étoiles augmente, jusqu'à ce que le soleil soit complètement couché, ce qui coïncide avec l'illumination maximale des étoiles.

À la moitié de cette phase de transition, les feux des cheminées commencent à s'allumer pour atteindre la luminosité maximale à la fin du cycle.

Notez que cette sortie génère une lumière clignotante qui simule la lueur des flammes. Enfin, à la fin du cycle, la lumière qui simule la comète s'allume progressivement.

La transition entre la luminosité minimale et maximale se produit dans un temps égal à 1/4 de celui défini pour le coucher de soleil.

À ce stade, nous nous trouvons au milieu de la nuit avec les étoiles dans le ciel qui brillent, la comète complètement illuminée et les feux allumés dans les maisons avec leurs tremblements typiques. La durée de cette phase, qui correspond à la nuit, est comprise, comme pour celle du jour, entre 0 et 102 secondes. Elle est réglée à l'aide du trimmer R2.

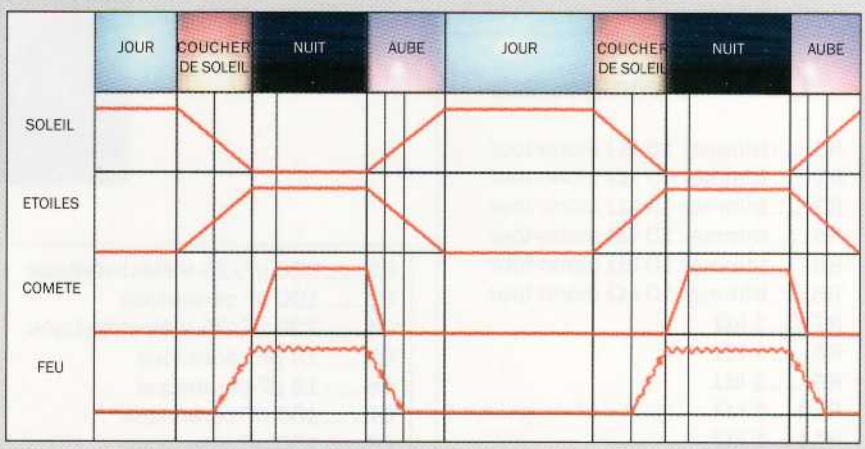
À la fin du temps imparti, la quatrième et dernière phase qui est l'aube commence. Peu à peu, l'intensité lumineuse des étoiles diminue tandis que la luminosité du jour augmente jusqu'à l'extinction complète des étoiles et l'éclairage complet des lampes qui simulent la lumière du jour.

Au début de cette phase, la comète s'éteint progressivement mais beaucoup plus rapidement, de sorte que, après une période égale à 1/4 du temps imparti pour le lever de soleil, la comète est complètement éteinte.

De même, au début de cette phase de transition, la luminosité des feux des cheminées des maisons commence à diminuer jusqu'à leur extinction.

Dans ce cas, le passage de la luminosité maximale à la luminosité minimale (éteint) se produit dans un temps égal à 1/2 de celui défini pour l'aube, à l'aide du potentiomètre R3.

À ce stade, nous avons simulé un cycle de 24 heures et le système se prépare à répéter la même séquence indéfiniment.



La consommation de courant dépend du nombre de LED et/ou des moteurs connectés aux sorties. La tension d'alimentation est directement reliée aux sorties « MOSFET » et au connecteur « NEOPIXEL ».

La tension d'alimentation est également reliée à l'anode de la diode D1, qui protège le circuit d'une inversion de polarité. Ensuite, la tension est appliquée à l'entrée du régulateur de tension U2 qui permet d'obtenir une tension stabilisée de 5 VDC pour alimenter le microcontrôleur, le module audio U3 et les composants périphériques.

Le convertisseur série/USB est alimenté par la tension USB provenant de l'ordinateur sur lequel il sera connecté.

Les condensateurs C1 et C2 placés sur la ligne d'alimentation filtrent les fluctuations de la tension due à la consommation des LED et en général des charges connectées aux sorties des MOSFET.

Cela est nécessaire, car ces fluctuations pourraient faire varier la tension d'alimentation et ainsi perturber le bon fonctionnement du microcontrôleur.

La fonction des condensateurs C3 et C4 est similaire.

Maintenant, abordons l'interface de programmation en circuit, qui doit être utilisée uniquement si vous souhaitez charger le « bootloader » dans un microcontrôleur vierge. Vous devez obligatoirement charger en premier le « bootloader » pour ensuite transférer le sketch à partir de l'IDE d'Arduino.

Le connecteur de programmation en circuit se nomme « ICSP » sur le schéma. Il est relié aux lignes « RST » (RESET ou réinitialisation), « MISO » (entrée des données du programmateur et sortie du micro U1), « MOSI » (sortie des données du programmateur et entrée des données du micro), SCK (horloge de communication série du bus SPI constitué par les broches MISO et MOSI), ainsi qu'à la masse et l'alimentation Vcc qui est fournie par le programmateur pour alimenter le circuit.

Rappelez-vous que lors de la phase de programmation du « bootloader », le circuit doit être alimenté à partir du bornier « +/- PWR ».

Concluons avec l'oscillateur du microcontrôleur, qui est cadencé par le quartz X1, les condensateurs C5 et C6 servent au démarrage de l'oscillateur qui fonctionne à 16 MHz.

Notez que la résistance R19 est présente sur le schéma, mais n'est pas nécessaire pour le fonctionnement du circuit.

Réalisation pratique

Passons maintenant à la construction de la carte qui nécessite la préparation d'un circuit imprimé double face dont les typons peuvent être téléchargés sur notre site, dans le sommaire détaillé de la revue, avec les autres fichiers du projet. Notez que la carte est disponible sous forme de kit, la carte Arduino UNO, la carte SD, le module DFR0299 et le convertisseur sont à prévoir en supplément.

Une fois le circuit imprimé gravé et percé, commencez par souder les composants ayant un profil bas c'est-à-dire les résistances, les condensateurs non polarisés, la diode D1, le connecteur coudé pour le convertisseur série/USB (en option), les trimmers, les boutons poussoir ainsi que le support de l'Atmega dont le repère en forme de « U » doit se situer vers le régulateur U2.

Continuez en soudant les LED, elles doivent être orientées correctement comme indiqué dans le plan de montage. La cathode est située du côté biseauté, l'anode ayant la patte la plus longue.

Maintenant, soudez les 2 condensateurs électrolytiques C2 et C4 en prêtant attention à leur orientation. Le « moins » est indiqué sur le boîtier, la patte la plus longue correspond au positif. Continuez avec le régulateur 7805, sa semelle métallique se situe vers les condensateurs C1 et C2.

Insérez et soudez les deux barrettes femelles à 8 broches pour le module audio, la barrette pour l'ICSP (4 pôles) et les 8 borniers.

Maintenant, vous devez monter les 6 transistors MOSFETS, chacun d'eux doit être fixé à un dissipateur thermique de type ML26. Commencez par plier délicatement les pattes à 90°. Présentez le radiateur vers la face métallique du transistor puis vissez l'ensemble « radiateur + MOSFET » sur le circuit imprimé. Répétez l'opération pour les autres MOSFET.

Une fois que les pattes sont insérées correctement dans les pastilles du circuit imprimé et que les radiateurs sont maintenus par les vis, soudez les transistors.

Vérifiez que chaque composant est correctement monté et qu'éventuellement son orientation est correcte. Insérez le module DFR0299 dans son emplacement en l'orientant de sorte que le connecteur de la carte SD se situe vers l'extérieur du circuit imprimé. Ensuite, insérez le microcontrôleur dans son support, l'encoche doit faire face au régulateur U2.

N'oubliez-pas que dans le module DFR0299, vous devez insérer une carte SD contenant des fichiers audio que vous aurez au préalable copiés à l'aide d'un ordinateur.

Il convient de rappeler pour des raisons évidentes qu'il est préférable que la durée de chaque morceau ne dépasse pas celle de la phase qui lui correspond. Par exemple, la durée du fichier audio de la phase du jour ne doit pas dépasser le temps d'exécution de cette phase. Ceci est valable pour les autres fichiers audio.

Si par contre le fichier dure moins longtemps que la phase correspondante, cela ne pose pas de problème, car le morceau est joué de manière cyclique jusqu'à la fin de l'exécution de la phase concernée.

Notez aussi que les fichiers audio doivent être de préférence placés à la racine de la carte SD. Le module est capable de rechercher des fichiers, cependant s'il y a d'autres fichiers audio sur la carte, il se peut que l'ordre dans lequel les fichiers doivent être reproduits ne soit pas respecté.

À ce stade, l'unité de contrôle peut avoir ou peut ne pas avoir besoin d'être programmée. Si vous avez acheté un microcontrôleur vierge, vous devez charger le « bootloader » en utilisant le connecteur de programmation ICSP.

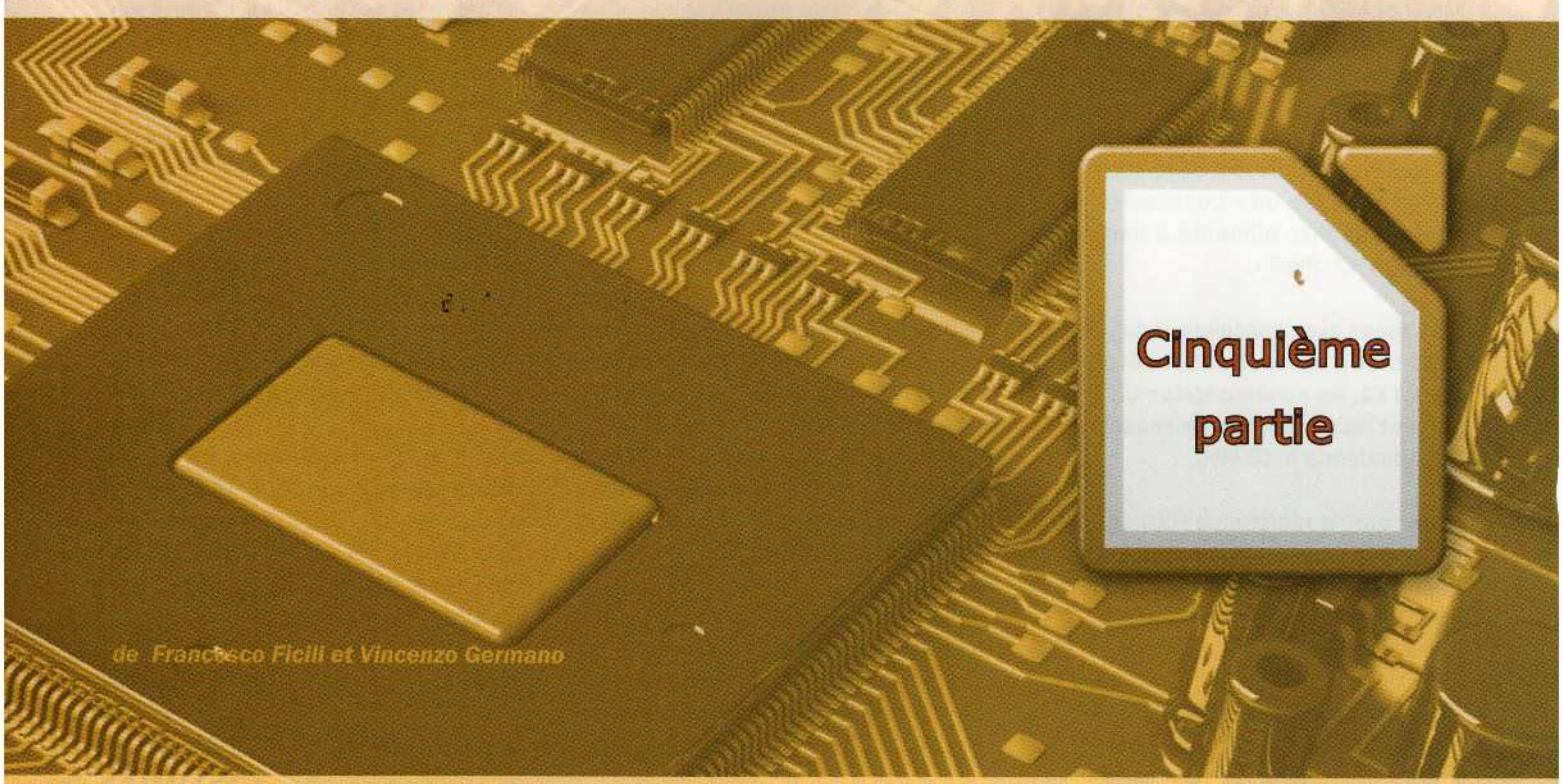
Une fois le « bootloader » chargé, montez le module série/USB sur le connecteur U4. Il permet de relier la carte à un PC via le port USB. Sur le PC, vous devez installer l'environnement de développement d'Arduino qui vous permettra de « téléverser » le sketch dans la carte.

Si vous achetez la carte en kit, le microcontrôleur est déjà programmé avec les fonctions de base (c'est-à-dire les 4 sorties gérées par les 4 trimmers). Cependant, pour rendre le projet facilement adaptable à vos besoins, nous fournissons le sketch en téléchargement avec les autres fichiers du projet.

Si vous avez acheté le kit et que vous ne vous contentez pas de la configuration de base en souhaitant apporter certaines modifications, vous devez installer le convertisseur série/USB.

Nous vous souhaitons de Joyeuses Fêtes ! ■

Apprenez à maîtriser kicad EDA



Cinquième partie

de Francesco Ficili et Vincenzo Germano

Nous poursuivons l'étude de Pcbnew, l'éditeur utilisé pour la mise en œuvre des circuits imprimés dans KiCad, en illustrant l'utilisation des librairies et de l'éditeur d'empreintes. Nous continuons ainsi le développement de notre projet pratique. Cinquième partie.

Dans la leçon précédente de ce cours, nous avons présenté et décrit Pcbnew, l'éditeur utilisé par KiCad pour la réalisation des circuits imprimés. Nous avons également décrit les règles de base pour le placement et le routage des composants. De plus, nous avons poursuivi le développement de notre projet pratique, en terminant la phase d'association et d'importation des composants au sein de Pcbnew.

Dans cette cinquième partie, nous poursuivons l'analyse de Pcbnew, en abordant des sujets tels que la gestion des librairies, la mise en œuvre des plans de masse et l'utilisation de l'éditeur d'empreintes (footprint) intégré à Pcbnew.

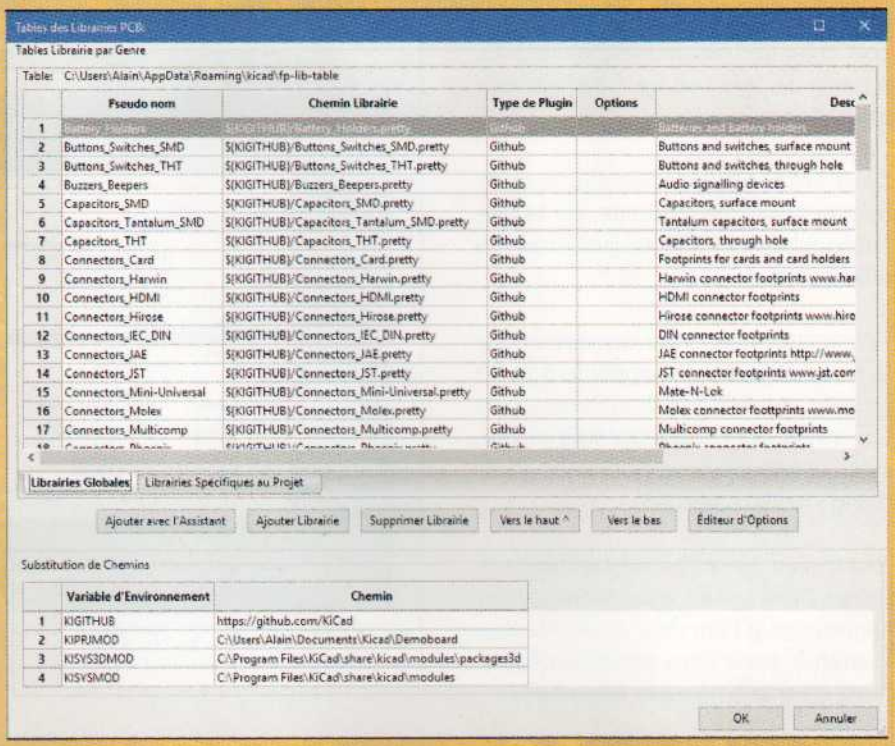
Nous poursuivrons également le développement du projet pratique en entrant dans la phase de réalisation du circuit imprimé (layout) de la carte.

Gestion des librairies dans Pcbnew

Pcbnew présente, tout comme Eeschema, un gestionnaire de librairies intégré, c'est-à-dire un outil permettant d'importer des librairies existantes dans un projet. Le gestionnaire de librairies peut être ouvert directement depuis Pcbnew, en sélectionnant le menu « **Préférences → Gestionnaire des Librairies d'Empreintes** ». Une fenêtre semblable à celle de la figure 1 s'ouvre.

À partir de cette fenêtre, vous pouvez facilement ajouter une seule librairie d'un module (bibliothèques de modules) ou un chemin spécifique contenant un groupe de librairies.

Comme nous l'avons expliqué dans les leçons précédentes, il est possible de trouver facilement sur le web des librairies KiCad développées par la communauté. Il existe également des outils permettant de convertir des librairies développées pour d'autres applications de CAO, telles que Autodesk Eagle. Il est également possible, grâce à l'éditeur, de développer ses propres bibliothèques de composants.



	Pseudo nom	Chemin Librairie	Type de Plugin	Options	Desc
2	Buttons_Switches_SMD	$(KIGITHUB)/Buttons_Switches_SMD.pretty	Github		Buttons and switches, surface mount
3	Buttons_Switches_THT	$(KIGITHUB)/Buttons_Switches_THT.pretty	Github		Buttons and switches, through hole
4	Buzzers_Beepers	$(KIGITHUB)/Buzzers_Beepers.pretty	Github		Audio signalling devices
5	Capacitors_SMD	$(KIGITHUB)/Capacitors_SMD.pretty	Github		Capacitors, surface mount
6	Capacitors_Tantalum_SMD	$(KIGITHUB)/Capacitors_Tantalum_SMD.pretty	Github		Tantalum capacitors, surface mount
7	Capacitors_THT	$(KIGITHUB)/Capacitors_THT.pretty	Github		Capacitors, through hole
8	Connectors_Card	$(KIGITHUB)/Connectors_Card.pretty	Github		Footprints for cards and card holders
9	Connectors_Harwin	$(KIGITHUB)/Connectors_Harwin.pretty	Github		Harwin connector footprints www.har
10	Connectors_HDMI	$(KIGITHUB)/Connectors_HDMI.pretty	Github		HDMI connector footprints
11	Connectors_Hirose	$(KIGITHUB)/Connectors_Hirose.pretty	Github		Hirose connector footprints www.hiro
12	Connectors_IEC_DIN	$(KIGITHUB)/Connectors_IEC_DIN.pretty	Github		DIN connector footprints
13	Connectors_JAE	$(KIGITHUB)/Connectors_JAE.pretty	Github		JAE connector footprints http://www.
14	Connectors_JST	$(KIGITHUB)/Connectors_JST.pretty	Github		JST connector footprints www.jst.com
15	Connectors_Mini-Universal	$(KIGITHUB)/Connectors_Mini-Universal.pretty	Github		Mate-N-Lok
16	Connectors_Molex	$(KIGITHUB)/Connectors_Molex.pretty	Github		Molex connector footprints www.mo
17	Connectors_Multicomp	$(KIGITHUB)/Connectors_Multicomp.pretty	Github		Multicomp connector footprints

	Variable d'Environnement	Chemin
1	KIGITHUB	https://github.com/KiCad
2	KIPRJMOD	C:\Users\Alain\Documents\Kicad\Demoboard
3	KISYS3DMOD	C:\Program Files\KiCad\share\kicad\modules\packages3d
4	KISYSMOD	C:\Program Files\KiCad\share\kicad\modules

Figure 1 : gestionnaire des librairies sous Pcbnew.

Éditeur de composants

Passons maintenant à l'analyse de l'éditeur d'empreintes (footprint) intégré dans Pcbnew. Pour ouvrir l'éditeur d'empreintes (des composants), sélectionnez l'icône correspondante dans la barre de projet Pcbnew, comme illustré en figure 2. En cliquant sur l'icône, la fenêtre de l'éditeur d'empreintes visible en figure 3 s'ouvre.

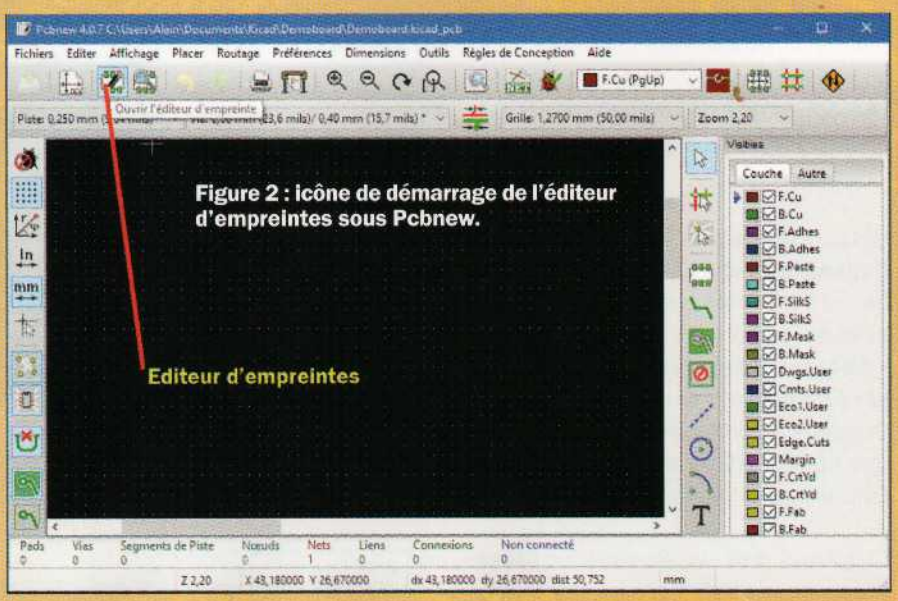


Figure 2 : icône de démarrage de l'éditeur d'empreintes sous Pcbnew.

Pour illustrer l'utilisation de l'éditeur d'empreintes, nous allons prendre un exemple simple. Imaginons que nous devons créer une empreinte, c'est-à-dire que nous devons dessiner le boîtier d'un composant avec les trous (pastilles) correspondant à ses broches, bien évidement avec les bonnes dimensions afin d'éviter des surprises lorsque nous devrons souder le composant.

Nous prenons le cas du boîtier SOIC18 qui est de type CMS (donc sans pastilles traversantes) et qui n'est pas présent dans les librairies de base de KiCad.

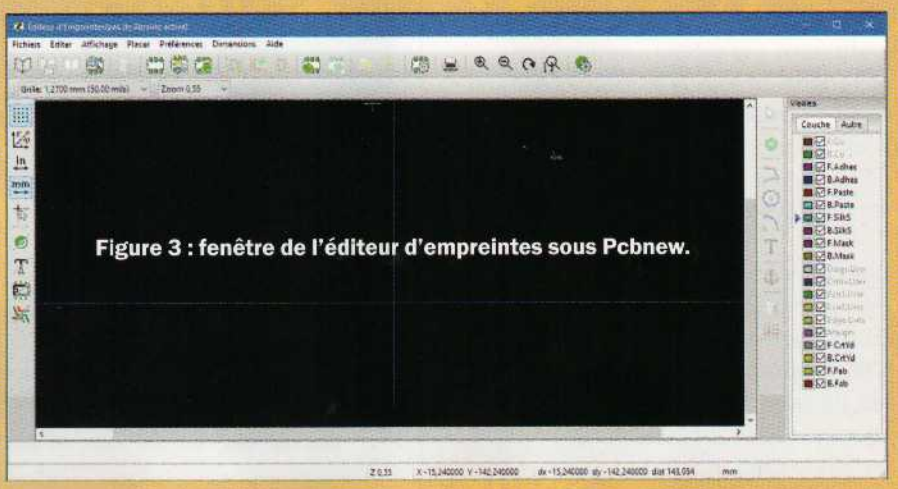


Figure 3 : fenêtre de l'éditeur d'empreintes sous Pcbnew.

Il existe essentiellement deux méthodes pour créer une nouvelle empreinte : soit à partir de zéro soit en modifiant une empreinte existante.

Dans cet exemple, nous allons opter pour la deuxième méthode, car la première est utilisée très rarement par les utilisateurs de KiCad.

Cependant, vous devez considérer que la réalisation d'une empreinte à partir de zéro ne change pratiquement rien en ce qui concerne la procédure décrite, sinon le fait que nous évitons de partir d'une « feuille blanche ».

Tout d'abord, sélectionnons la libraire de travail. Dans ce cas, nous allons ajouter l'empreinte à l'intérieur d'une librairie existante, mais nous pourrions également créer une nouvelle bibliothèque.

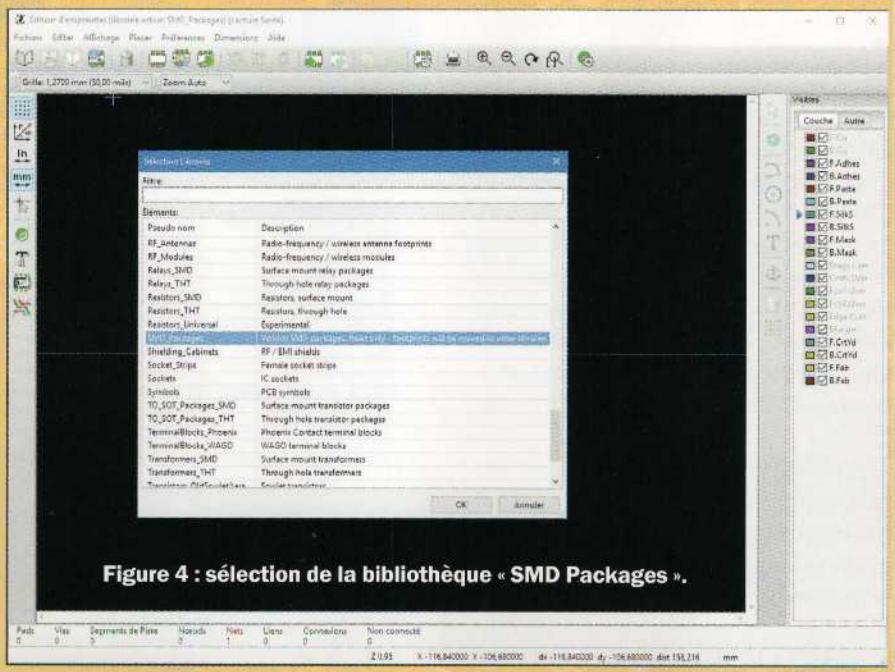


Figure 4 : sélection de la bibliothèque « SMD Packages ».

Les boîtiers de type SOIC se trouvent dans la librairie « SMD Packages », nous allons donc l'utiliser comme une bibliothèque de travail. Pour cela, sélectionnons dans le menu « **Fichiers** » l'option « **Sélection de la Librairie Active** ». Une fenêtre semblable à celle de la figure 4 s'ouvre.

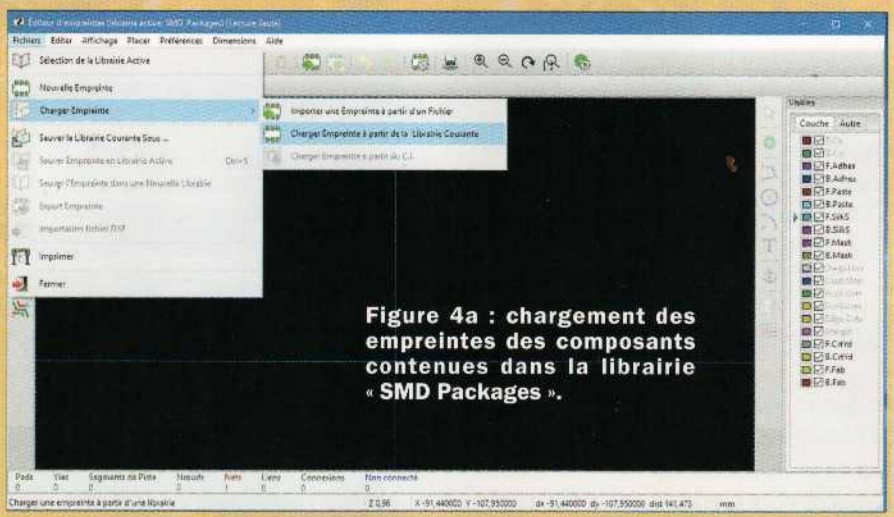


Figure 4a : chargement des empreintes des composants contenues dans la librairie « SMD Packages ».

Sélectionnons la librairie « **SMD Packages** » et cliquons sur le bouton « OK ». Nous devons maintenant accéder au menu « **Fichiers** » → « **Charger Empreinte** » → et « **Charger Empreinte à partir de la Librairie Courante** », comme visible en figure 4a. Une petite fenêtre, nommée « Charger Empreinte », s'ouvre.

Cliquons sur le bouton « **Liste tous** » comme indiqué en figure 4b. Une nouvelle fenêtre dénommée « Empreintes » s'ouvre, sélectionnons alors dans cette liste l'empreinte qui se rapproche le plus physiquement du composant que nous voulons créer. Pour cela, choisissons l'empreinte « SO-16-N » et cliquons sur « OK », comme illustré en figure 5.

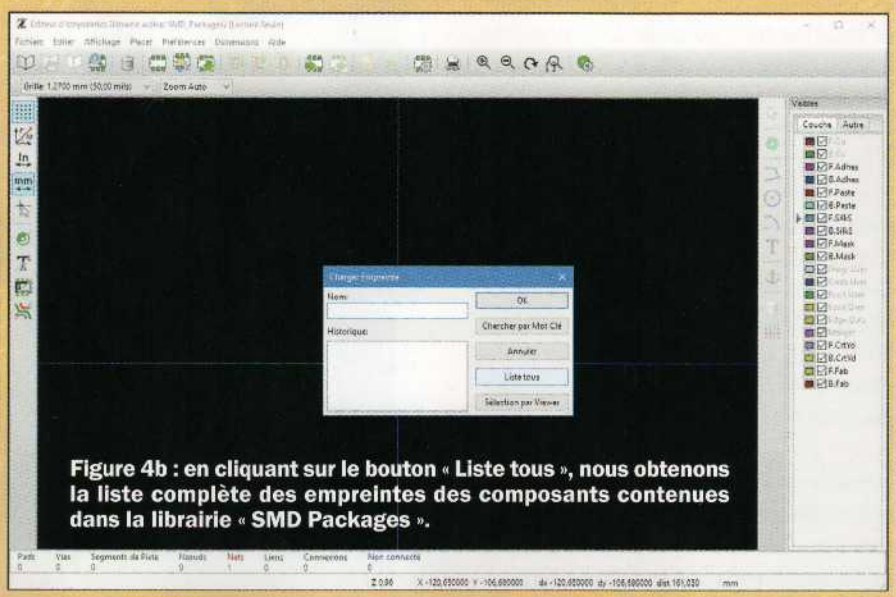


Figure 4b : en cliquant sur le bouton « Liste tous », nous obtenons la liste complète des empreintes des composants contenues dans la librairie « SMD Packages ».

Normalement, vous devez obtenir un résultat semblable à celui de la figure 6 où vous voyez apparaître une empreinte d'un composant correspondant à un boîtier de type « SOIC16 », c'est-à-dire un circuit intégré comportant 16 broches en boîtier CMS.

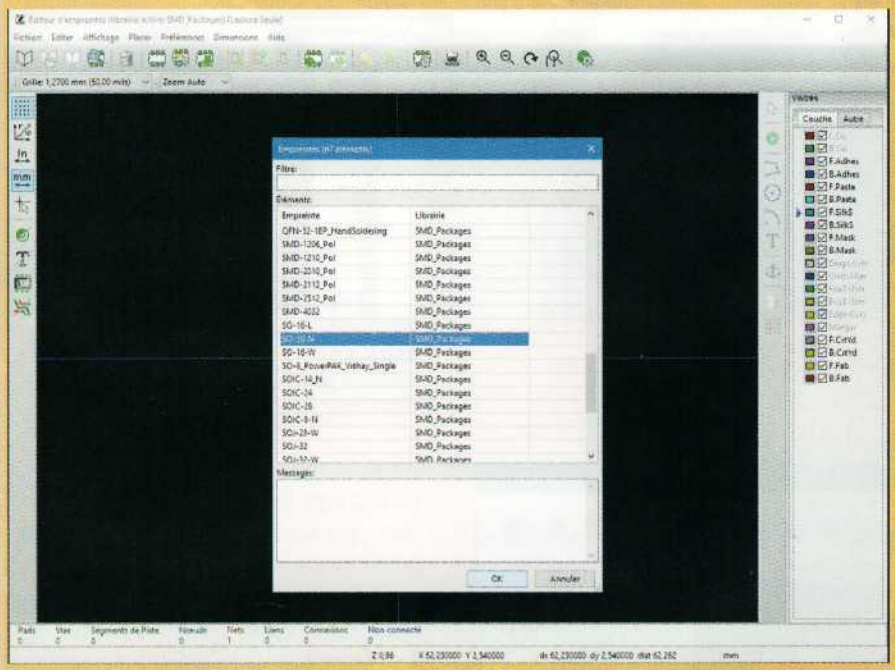


Figure 5 : sélection de l'empreinte de départ (SOIC-16) à partir de laquelle nous allons réaliser l'empreinte SOIC-18.

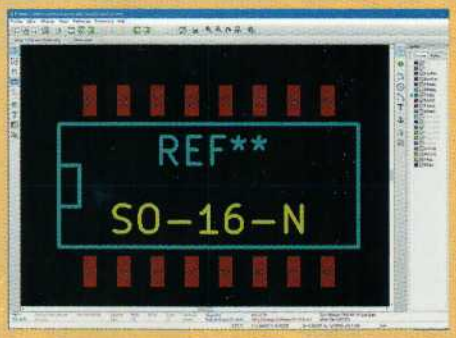


Figure 6 : ici l'empreinte de départ SOIC-16 ouverte dans l'Editeur d'empreintes de Pcbnew.

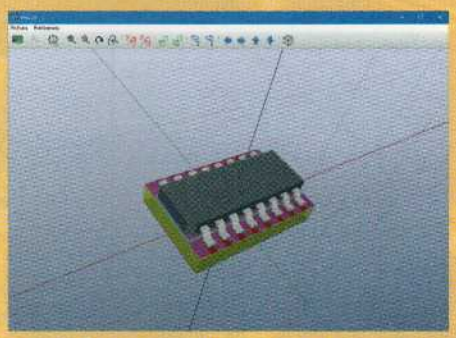


Figure 6a : l'empreinte SOIC-16 affichée en 3D dans le visualisateur 3D. Cela vous permet de vous donner une idée de la structure physique du composant.

Pour vous faire une idée plus précise du composant, allez dans le menu « Affichage » → « 3D Visualisateur », vous devez obtenir un écran semblable à celui de la figure 6b (sous réserve que votre carte graphique gère au minimum Open GL version 4.0).

Revenons à notre empreinte (fermez la fenêtre 3D), nous devons donc ajouter 2 pads (ou pastilles) et modifier la taille du composant (pour cela référez-vous à la documentation technique du composant pour obtenir les bonnes dimensions physiques : pas entre les pads, dimensions du boîtier, etc.).

Pour modifier le module (composant) correctement, nous devons insérer deux pads supplémentaires. Nous copions donc les pads 8 et 9. Par exemple à partie de la pad 8, faisons un clic droit de la souris, puis sélectionnons dans la boîte de dialogue « **Dupliquer Pad** ».

Nous déplaçons vers la droite la nouvelle pad à une distance qui correspond au pas (écartement entre 2 pads) des autres pads. Normalement le pas ou pitch dans les documentations techniques est de 1.27 mm pour ce type de boîtier.

Nous faisons de même pour la pad (pastille) 9. Nous nous retrouvons donc avec 2 pads numérotées 8 et 2 autres numérotées 9. Pour cela, nous devons modifier les propriétés pour avoir une numération correcte des pads du composant.

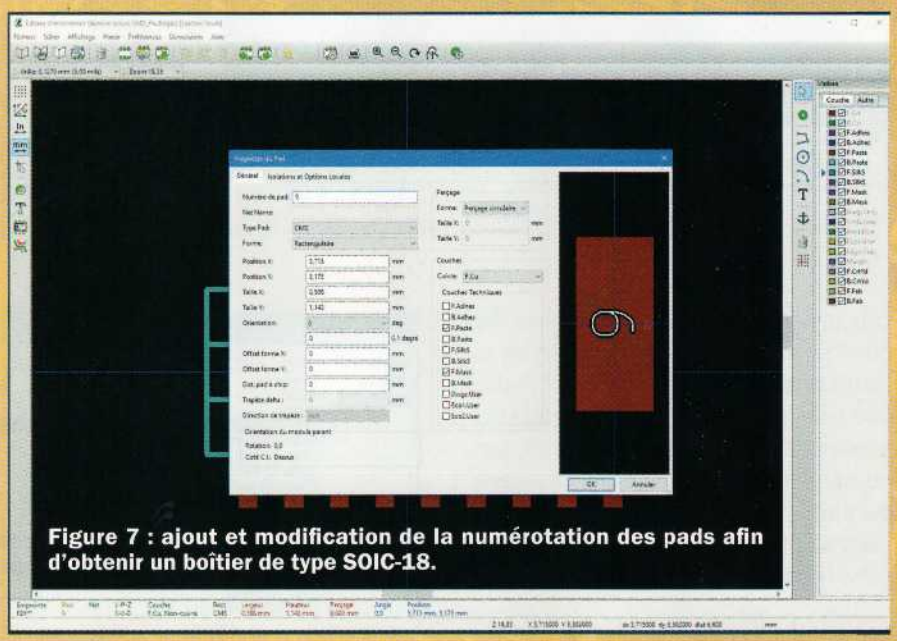


Figure 7 : ajout et modification de la numérotation des pads afin d'obtenir un boîtier de type SOIC-18.

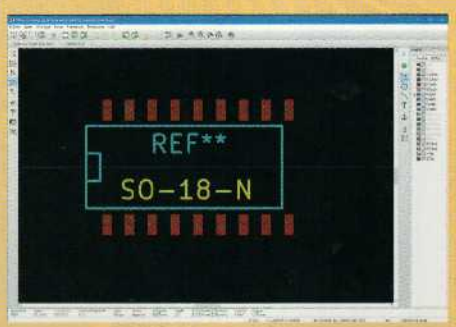


Figure 8 : ici, les pads correctement numérotées mais le contour du boîtier doit être ajusté.

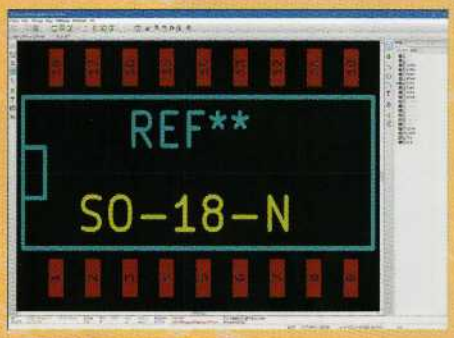


Figure 9 : ici, le contour de l'empreinte ajusté aux bonnes dimensions physiques du boîtier du composant réel.

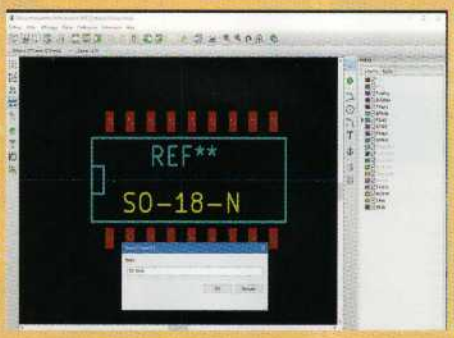


Figure 9a

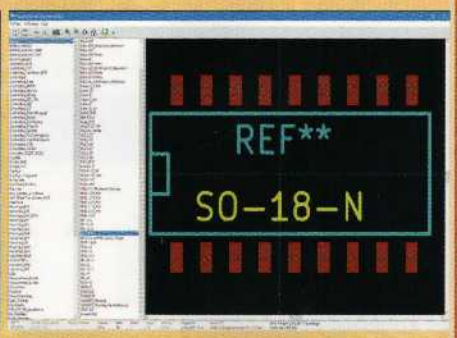


Figure 10 : ici l'empreinte « SO-18-N » visible à partir de Pcbnew (« Placer → Empreinte »).

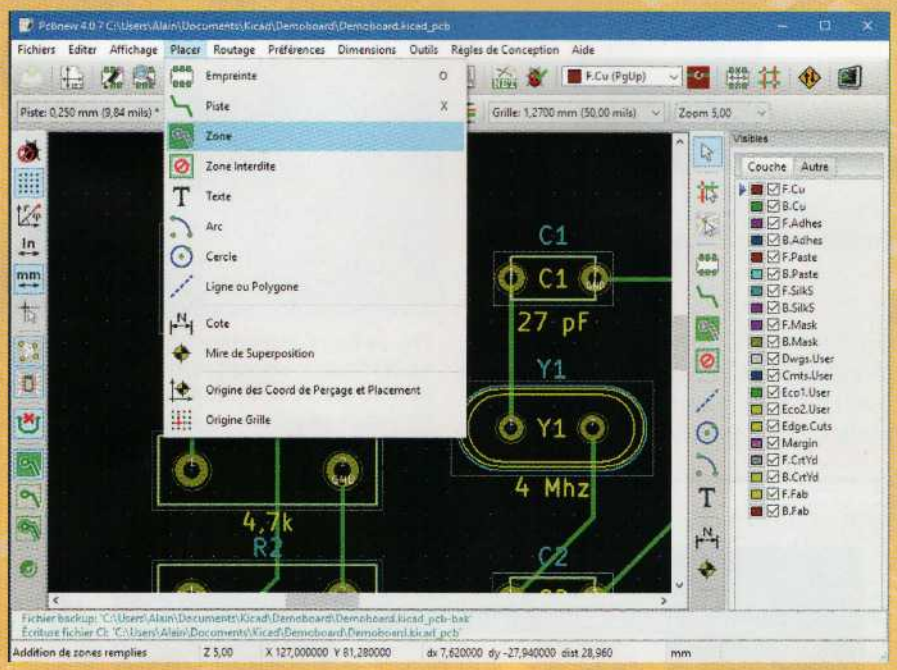


Figure 11 : sélection de l'outil « Zone » à partir du menu « Placer » de Pcbnew.

Le résultat final est visible en figure 9. Nous pouvons maintenant enregistrer la nouvelle empreinte en sélectionnant le menu « **Fichiers → Sauver Empreinte dans Librairie Active** » ou en utilisant les touches de raccourci « **Ctrl + S** ».

L'éditeur vous demandera de saisir le nom avec lequel vous souhaitez enregistrer la nouvelle empreinte.

Dans la boîte de dialogue « Sauver Empreinte », tapez « SO-18-N » (voir la figure 9a) et cliquez sur le bouton « OK ».

À ce stade, le module est présent dans les librairies d'empreintes et peut être utilisé dans Pcbnew. Si nous utilisons le visualisateur d'empreintes à partir de Pcbnew, nous trouverons le module

Cliquons sur la nouvelle pads 8 (qui a été dupliquée) en faisant un clic droit de la souris puis sélectionnons « Editer Pad ». Une boîte de dialogue s'ouvre semblable à celle de la figure 7. Changeons son numéro en 9 dans le champ « **Numéro de pad** ». Faisons de même pour les pads du haut pour arriver jusqu'à la dernière dont le numéro est 18.

Vous devez obtenir un résultat comparable à celui de la figure 8, où toutes les pads sont correctement numérotées mais où le contour du boîtier (sérigraphie en bleu pâle) est trop petit car nous avons ajouté des pads. Il faut donc l'ajuster aux dimensions correctes.

À ce stade, il ne reste plus qu'à modifier le contour du composant (c'est-à-dire la sérigraphie) et modifier le nom du composant, il faut le nommer **SO-18-N**.

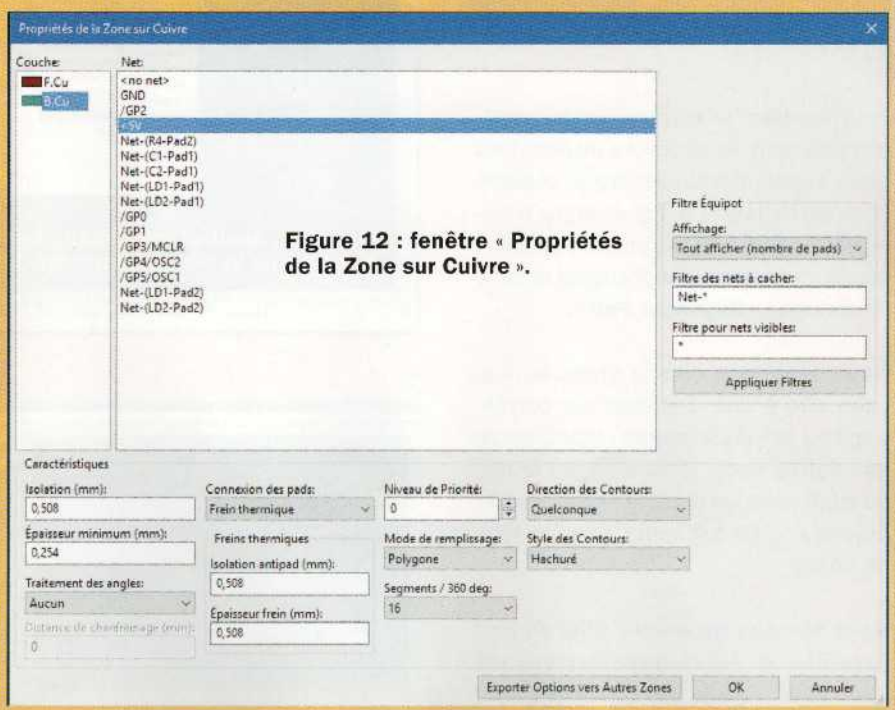


Figure 12 : fenêtre « Propriétés de la Zone sur Cuivre ».

dans la librairie « SMD packages », comme illustré en figure 10.

Plans de masse et remplissage

L'une des caractéristiques typiques de la CAO électronique est la gestion des remplissages, tels que les plans de masse, les masques et les plans d'application de la pâte à braser. KiCad propose un outil de remplissage d'une zone, accessible à partir de la barre d'outils, avec les autres outils de dessin, comme illustré en figure 11.

À partir de Pcbnew, allons dans le menu « **Placer → Zone** » pour sélectionner l'outil de remplissage. Ce dernier modifie son comportement en fonction de l'utilisation. La différence de comportement varie en fonction de la couche sur laquelle il travaille (en particulier s'il s'agit d'une couche de cuivre ou non).

Examinons l'utilisation de l'outil pour la réalisation d'un plan de masse, en nous reportant à notre exemple abordé dans les leçons précédentes. Tout d'abord, assurez-vous que vous travaillez sur l'une des couches de cuivre ((F.Cu(PgUp) ou (B.Cu(PgDn)).

Ensuite, une fois que vous avez sélectionné l'outil « **Zone** » (voir la figure 11), un pointeur en forme de crayon apparaît, cliquez alors avec le bouton gauche de la souris dans la zone de travail de Pcbnew, vous devez voir apparaître la fenêtre « **Propriétés de la Zone sur Cuivre** » comme illustré en figure 12. Il s'agit de la fenêtre des propriétés de remplissage d'une couche de cuivre.

Dans cette fenêtre, vous pouvez sélectionner la couche (top ou bottom), la « net » que vous souhaitez « remplir » (par exemple la net GND) et d'autres options (coins arrondis ou non, mode de remplissage, frein thermique, etc).

Dans notre exemple, nous avons sélectionné la couche top (F.Cu(PgUp) et la net « GND » afin de créer un plan de masse sur la couche supérieure. Une fois les différentes options définies, nous devons sélectionner la zone de remplissage.

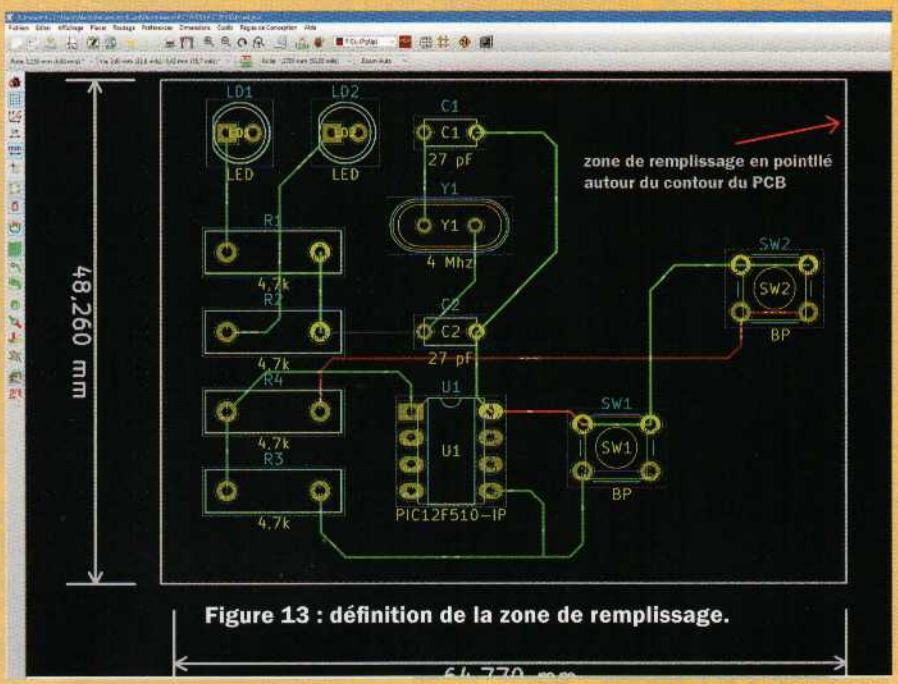


Figure 13 : définition de la zone de remplissage.

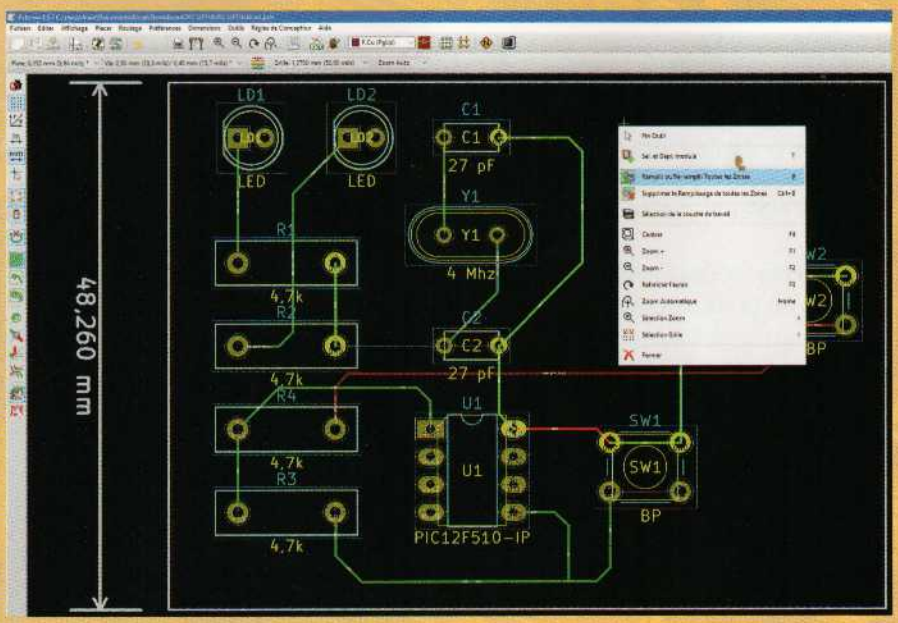


Figure 14 : sélection de l'option « Remplir ou Re-remplir Toutes les Zones » en cliquant sur le bouton droit de la souris.

Dans notre cas, nous sélectionnons une zone qui couvre tout le circuit imprimé, le contour comporte des hachures de couleur rouge comme vous pouvez le voir en figure 13.

À ce stade, pour remplir la zone sélectionnée, il suffit de cliquer sur le bouton droit de la souris dans la zone à remplir et de sélectionner l'option « **Remplir ou Re-remplir Toutes les Zones** », comme illustré en figure 14. Le résultat final est visible en figure 15, la net « GND » de la couche de cuivre supérieure (top ou (F.Cu(PgUp)) a été remplie (zone de couleur rouge).

Dans cet exemple, nous avons rempli la zone de cuivre supérieure (couche top en rouge).

Nous pouvons compléter le plan de masse également sur la partie inférieure du cuivre (couche bottom en vert). Nous devons donc répéter le processus en sélectionnant la couche (B.Cu(PgDn) dans les propriétés de remplissage des zones de cuivre.

Il est alors nécessaire de tracer (définir) une nouvelle zone de remplissage.

Projet pratique : la Demoboard PIC18F4550

Reprenons nos travaux liés au projet pratique du cours c'est-à-dire la Demoboard à base de PIC18F4550.

Lors du cours précédent, nous avions terminé le processus d'association des composants et importé la netlist.

Nous allons maintenant procéder à la réalisation du circuit imprimé (layout) de la carte, en commençant par le placement des composants.

Après l'importation de la netlist, nous distribuons les modules ou empreintes des composants comme nous l'avons vu dans l'exemple abordé dans la leçon précédente.

NB : le terme « nous distribuons » veut dire répartir de manière cohérente les composants en réduisant au minimum les longueurs des pistes.

Par exemple, si vous vous reportez au schéma de la demoboard, en examinant la partie alimentation, vous vous apercevez que les condensateurs C14 et C15 sont proches, il est donc préférable de les placer physiquement un à côté de l'autre sur le circuit imprimé de façon à avoir des pistes les plus courtes possible. Cela simplifie le routage et évite des perturbations électriques qui pourraient altérer le fonctionnement du circuit.

Un autre exemple est celui des condensateurs C1 et C2 qui doivent être le plus proche possible du quartz et l'ensemble constitué par C1, C2 et Y1 doit être lui-même à côté du microcontrôleur.

Utilisez cette règle pour le reste du montage et en général pour toutes vos réalisations.

Ainsi, nous obtenons une distribution semblable à celle illustrée en figure 16, bien évidement vous pouvez avoir une distribution différente selon vos contraintes (boîtier, emplacement, etc.).

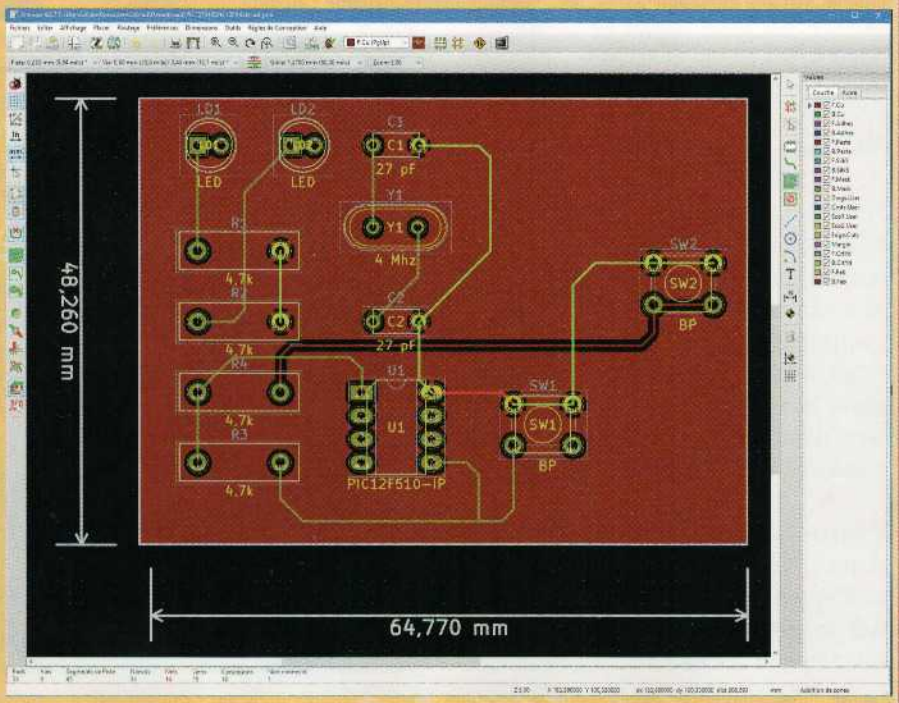


Figure 15 : réalisation du plan de masse.

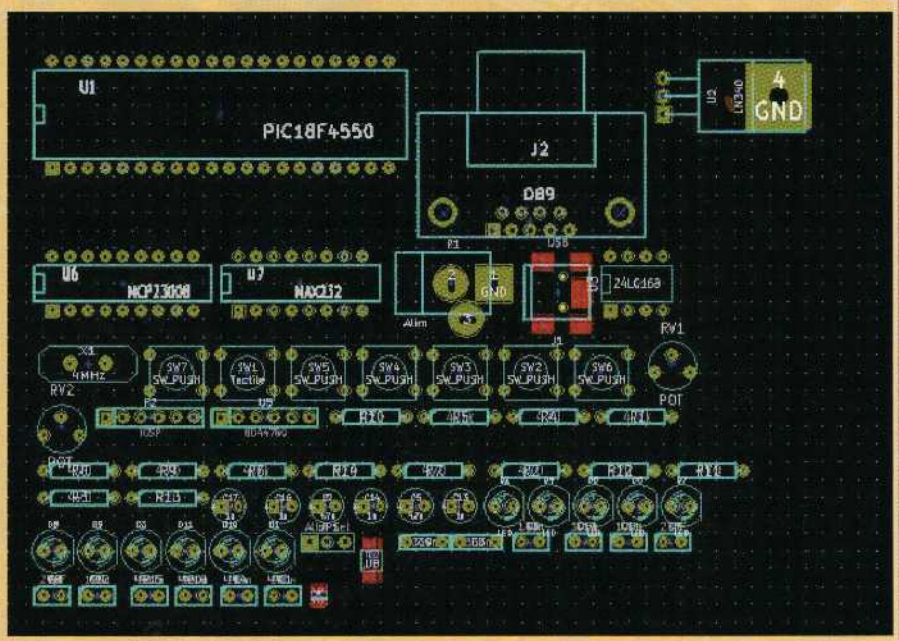


Figure 16 : ici, une distribution possible des composants sur le circuit imprimé.

Nous commençons donc par le dessin du contour du circuit imprimé afin de travailler avec une zone ayant des dimensions fixes.

Nous optons pour une surface de dimensions 120 mm x 150 mm (mais vous pouvez choisir d'autres dimensions qui correspondent à vos besoins).

Positionnons-nous sur la couche « bord » et traçons les contours.

Pour dessiner le contour aux bonnes dimensions, nous utilisons l'outil de cotation.

À ce stade, nous insérons quatre trous à chaque angle du circuit imprimé afin de pouvoir utiliser des entretoises permettant la fixation de la carte dans un boîtier. Pour insérer les trous, allons dans le menu « **Placer → Empreinte** » et dans la liste sélectionnons l'empreinte « **1Pin** », comme illustré en figure 17.

Le résultat final doit être semblable à celui représenté en figure 18.

Avant de passer au routage des pistes du circuit imprimé, nous allons examiner la manière de modifier une empreinte d'un composant ou ses propriétés, s'ils ne conviennent pas.

Prenons l'exemple de la mémoire EEPROM dénommée U3. Positionnons le pointeur de la souris sur l'empreinte U3, faisons un clic droit et sélectionnons

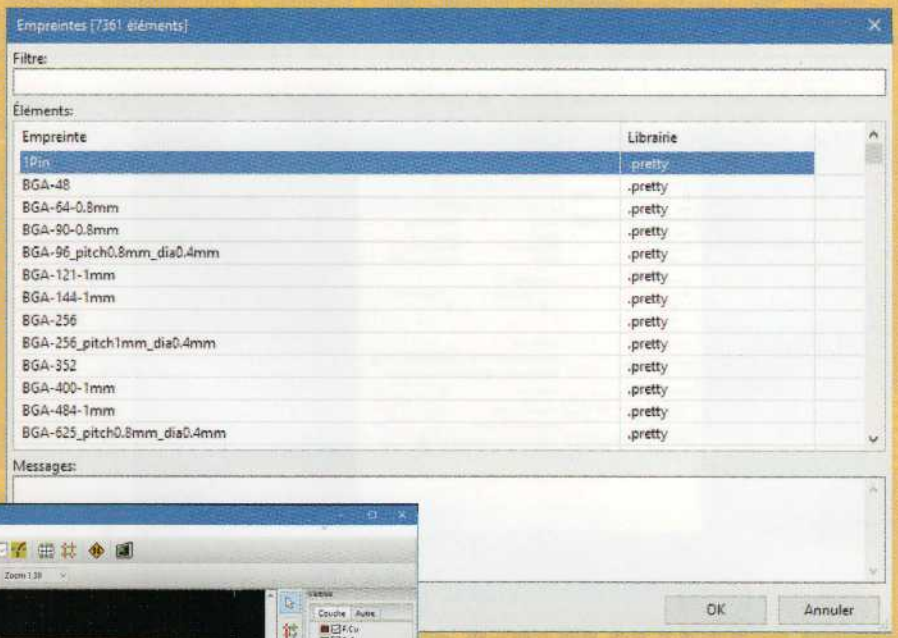


Figure 17 : sélection de l'empreinte « 1Pin » afin de placer un trou dans chaque angle du circuit imprimé.

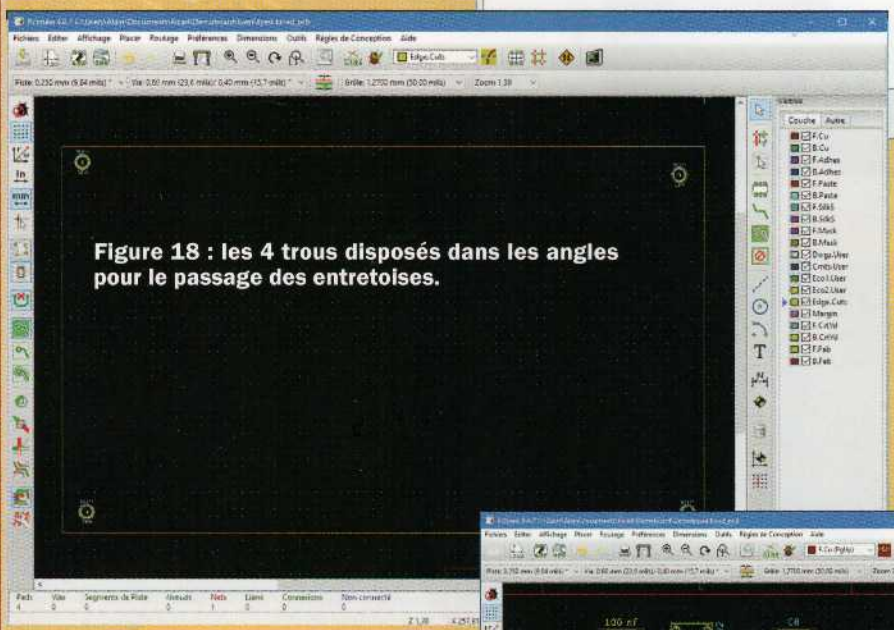
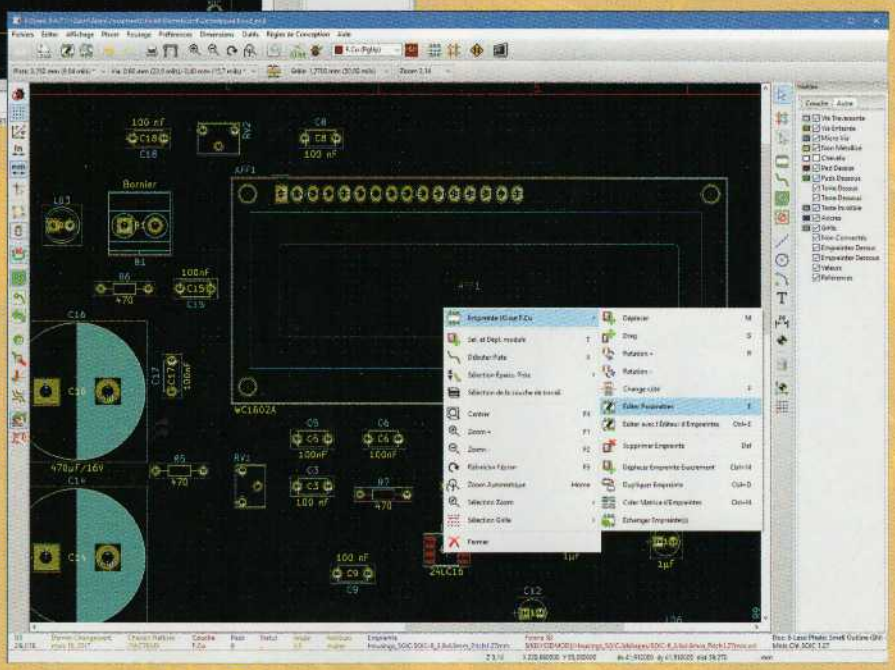


Figure 18 : les 4 trous disposés dans les angles pour le passage des entretoises.

Figure 19 : édition des paramètres d'une empreinte d'un composant, ici c'est la mémoire U3 qui est sélectionnée.



« **Empreinte U3 sur F.Cu** » puis « **Editer paramètres** » (voir la figure 19).

La fenêtre « **Propriétés de l'Empreinte** » s'ouvre, comme illustré en figure 20.

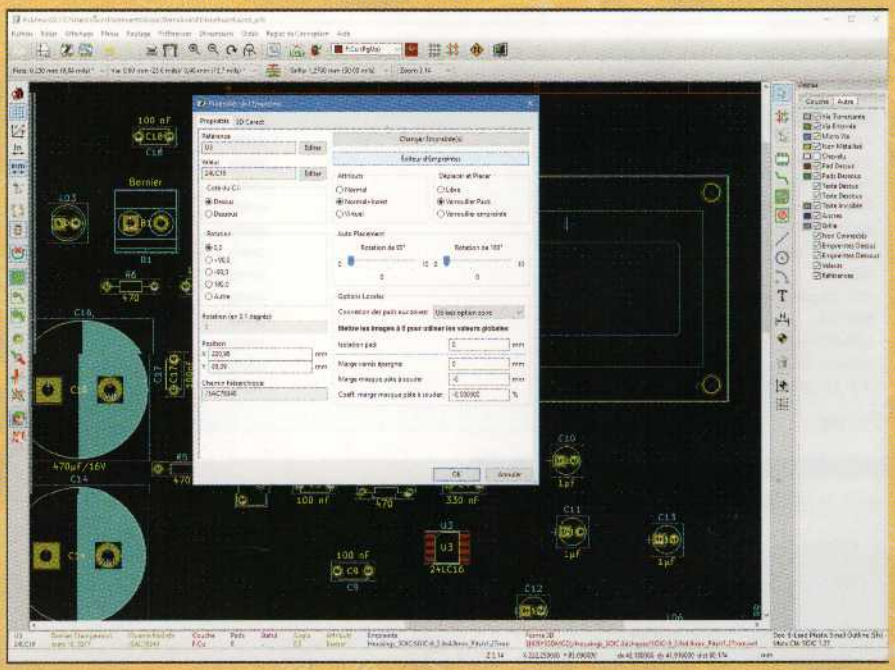
À partir de cette fenêtre, nous pouvons modifier la référence du composant, sa valeur, sa disposition. Il est possible aussi de lui associer une autre empreinte différente en cliquant sur le bouton « Changer Empreinte(s) ».

Et enfin, il est possible de modifier son empreinte actuelle en cliquant sur le bouton « **Editeur d'Empreintes** ». Ce dernier s'ouvre comme illustré en figure 20a, vous pouvez alors modifier les pads, les dimensions, etc. comme nous l'avons étudié un peu plus haut dans cette leçon.

Selon vos besoins vous pouvez effectuer des modifications, mais nous vous conseillons de sauvegarder le dossier où se situent vos librairies avant d'effectuer des changements au niveau des empreintes.

À ce stade, nous pouvons procéder au placement définitif des divers composants sur le circuit imprimé afin de les placer de manière ordonnée et en les alignant. Nous commençons par placer le réseau de composants du bus

Figure 20 : fenêtre des propriétés de l'empreinte U3.



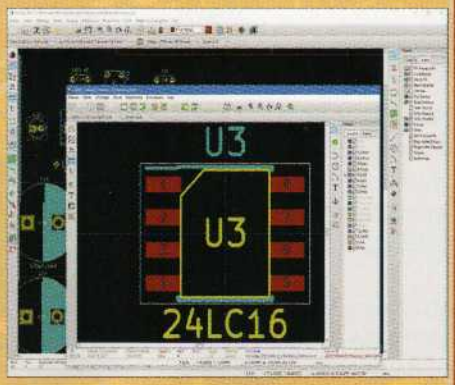


Figure 20a : fenêtre de l'éditeur d'empreinte de U3 à partir de la fenêtre des propriétés de l'empreinte.

Figure 21 : placement des composants du bus I²C.

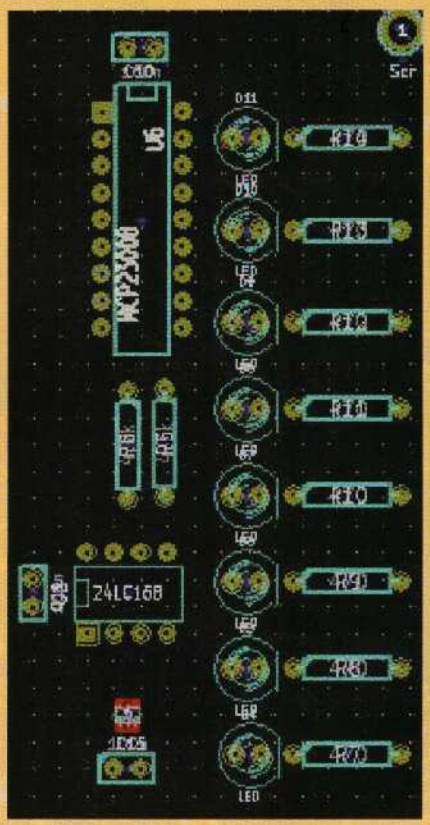


Figure 22 : placement de l'afficheur LCD et des boutons poussoirs.

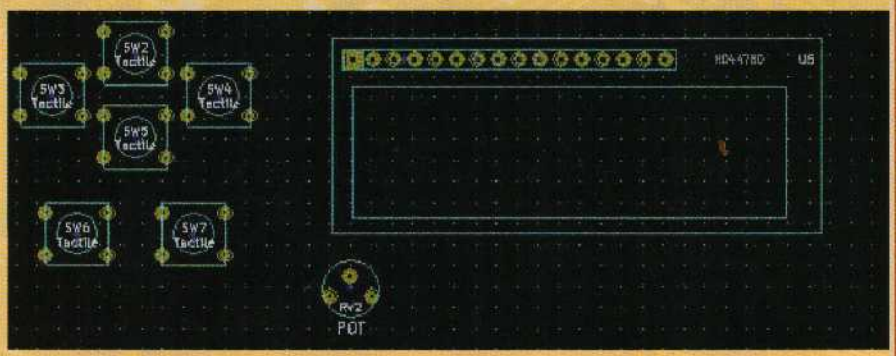


Figure 23 : placement du PIC18F4550, du circuit oscillateur, du bouton de reset, du thermomètre TC72 et du potentiomètre analogique dans la zone centrale du circuit imprimé.

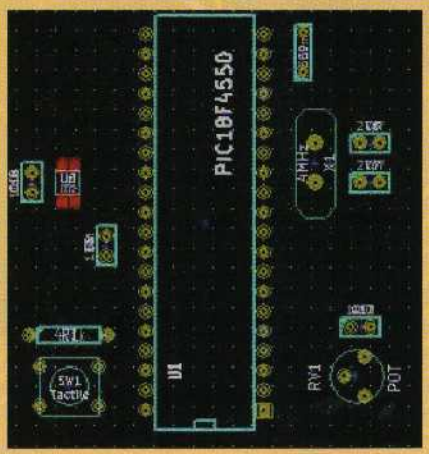
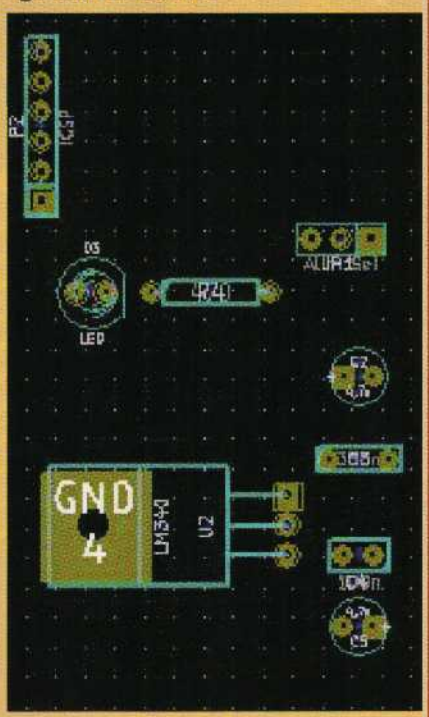


Figure 24 : placement du régulateur de tension et du connecteur de programmation ICSP.



I²C (y compris les LED connectées au circuit d'extension des ports) sur le côté droit de la carte, comme illustré en figure 21. Maintenant, positionnons l'afficheur et les boutons.

Pour l'afficheur LCD, nous décidons de le placer en haut et au centre du circuit imprimé et les boutons à gauche. Vous pouvez cependant choisir un autre emplacement, mais en gardant

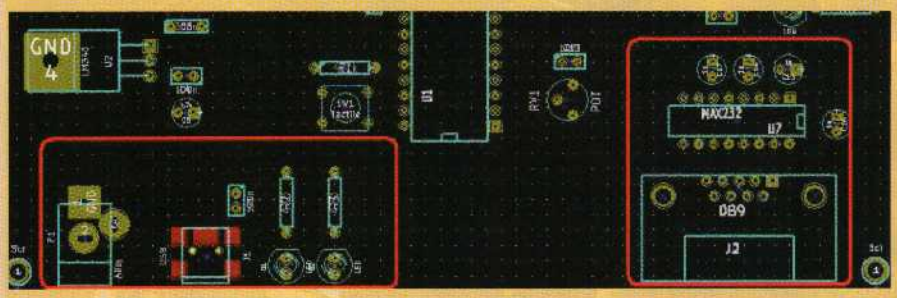
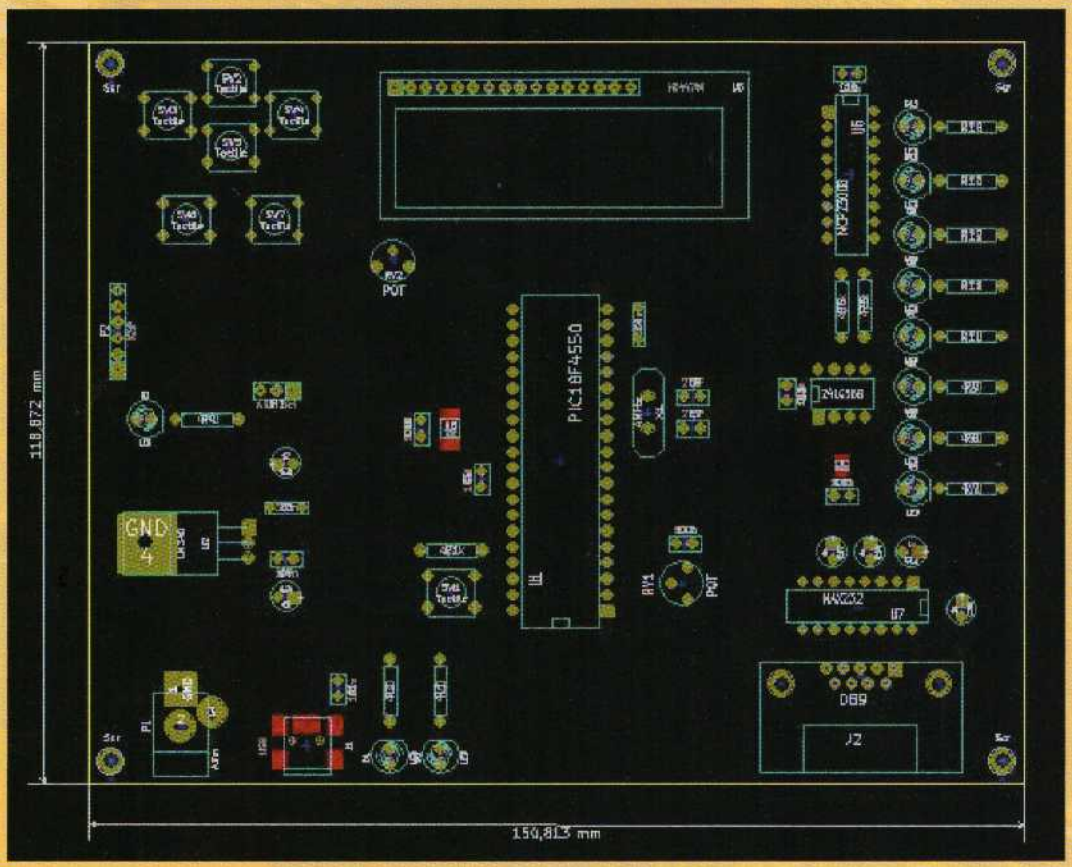


Figure 25 : placement de l'interface RS232, du connecteur USB, des deux LED et du jack d'alimentation.

Figure 26 : placement terminé de tous les composants sur le circuit imprimé.



à l'esprit de minimiser la longueur des pistes.

Nous disposons les boutons poussoirs SW2 et SW5 en croix de manière à réaliser un petit clavier qui permet une navigation.

En dessous, nous plaçons les poussoirs SW6 et SW7 qui serviront d'accès aux différentes fonctions du circuit.

Enfin, nous positionnons le potentiomètre de réglage du contraste de l'afficheur juste en dessous de ce dernier, l'ensemble est illustré en figure 22.

Nous continuons en plaçant le microcontrôleur PIC18F4550, le circuit oscillateur, le bouton de réinitialisation, le thermomètre TC72 et le potentiomètre analogique.

Nous disposons ces composants dans la zone centrale du circuit imprimé, de manière à faciliter les différentes connexions entre les broches du microcontrôleur et celles des différents périphériques.

La figure 23 illustre un exemple de positionnement.

Sur le côté gauche de la carte, nous plaçons le régulateur de tension et le connecteur de programmation ICSP, comme illustré en figure 24.

Nous pouvons maintenant positionner les composants restants, à savoir l'interface RS232, le connecteur USB, les deux LED connectées directement au PIC18F4550 et la prise d'alimentation.

Le placement de ces derniers composants est illustré en figure 25.

Le placement complet de tous les composants est illustré en figure 26.

À ce stade, la phase de positionnement de tous les composants est terminée et la phase de routage peut commencer.

Vous pouvez avoir une disposition différente de celle illustrée, l'essentiel étant d'avoir des connexions physiques les plus courtes possible entre les composants. Cela est d'autant plus vrai pour les circuits haute fréquence.

Conclusion

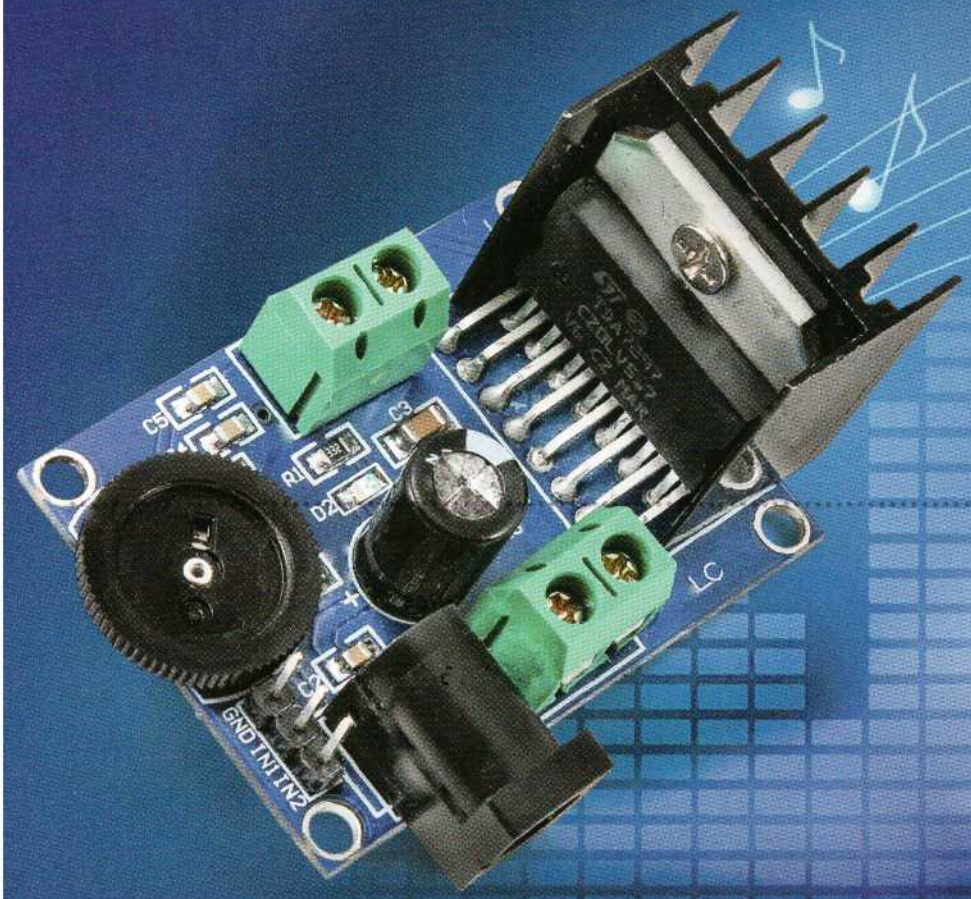
Dans cette cinquième leçon, nous avons étudié d'autres fonctionnalités de Pcbnew, telles que la gestion des librairies, l'éditeur intégré d'empreintes et l'outil de remplissage de zone, que nous avons utilisé pour créer un plan de masse.

Nous avons également poursuivi le développement du projet pratique, en achevant la phase de placement des composants.

Dans la prochaine et dernière partie de ce cours, nous terminerons l'analyse de Pcbnew en illustrant la génération des fichiers gerber et le visualisateur intégré à la suite. Enfin, nous terminerons le développement du projet pratique.

AMPLIFICATEUR BF 2 x 15 W à TDA7297

de Davide Scullino



Cet amplificateur BF stéréo compact est utilisable à la maison comme dans la voiture. Il peut remplacer des étages de puissance défectueux ou servir d'amplificateur de banc de test pour l'analyse d'équipements audio (préamplificateurs, baladeur MP3, etc.). Il peut aussi être utilisé dans une barre de son de fabrication maison pour remplacer la partie audio des téléviseurs à écran plat qui est inexistante dans les modèles d'entrée de gamme.

Lorsque vous réparez un équipement audio, avoir un amplificateur BF sur votre table de travail est toujours utile. Si c'est le cas, cela est beaucoup mieux, car vous pouvez tester des équipements audio stéréo tels que des lecteurs de disques compacts, des platines à cassettes, des tuners, des lecteurs MP3, etc.

Pour le laboratoire ? Pas seulement. Nous avons conçu et nous vous proposons de réaliser un amplificateur BF très compact à base d'un **circuit intégré monolithique** de chez **STMicroelectronics**. L'amplificateur, qui comporte très peu de composants externes, est capable de fournir une puissance de 2 x 15 W efficaces sous 8 Ω avec un taux de distorsion de 1 % (VCC = 18 VDC).

La puissance de sortie relativement élevée et la bonne qualité sonore en termes de distorsion harmonique et de bande passante rendent le circuit adapté également aux applications qui n'ont rien à voir avec le laboratoire.

Par exemple, il peut être utilisé comme étage de puissance pour une paire d'enceintes amplifiées pour PC (l'amplificateur est placé dans l'une des enceintes et relié à l'autre par un câble).

Comme nous l'avons évoqué, il peut faire partie d'une barre de son fabriquée en bois avec des haut-parleurs de bonne qualité sur laquelle viendra se poser l'écran plat (ces derniers manquant cruellement de basses car il n'y a plus de

caisse de résonnance par rapport aux TV cathodiques).

L'amplificateur peut également remplacer un étage final défaillant d'une chaîne Hi-Fi compacte dont les composants ne peuvent plus être trouvés en raison de son âge avancé, ou de sa bonne qualité sonore par rapport à sa petite taille et à laquelle vous êtes attaché. En fait, il s'agit d'un amplificateur qui peut sans aucun doute être placé parmi ceux de haute-fidélité.

Nous aborderons les applications après avoir examiné le schéma électrique et analysé son fonctionnement ainsi que ses caractéristiques.

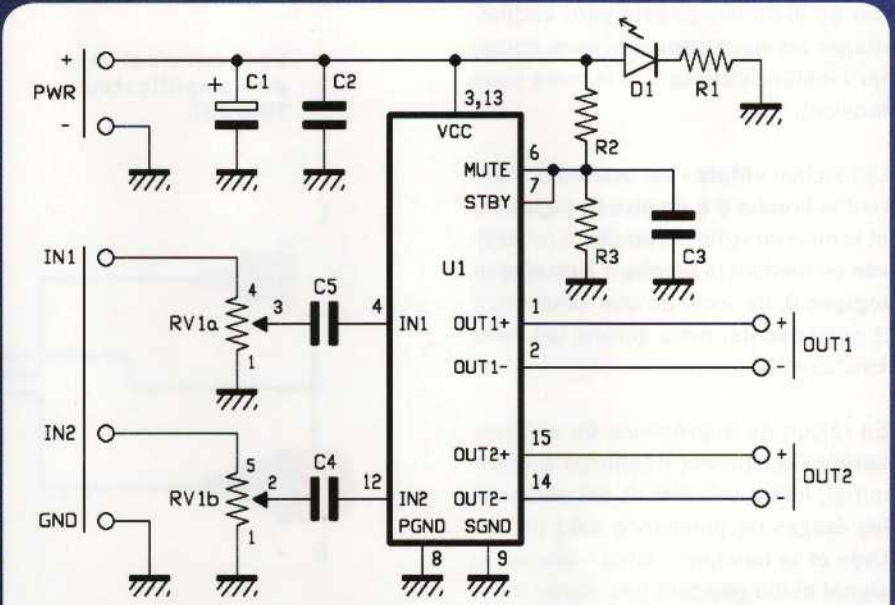


Schéma électrique de l'amplificateur BF à base de TDA7297.

Le schéma électrique

L'amplificateur est composé d'un circuit monolithique à usage universel, comportant deux étages d'amplification BF avec des sorties en pont. Ils peuvent être assimilés à des amplificateurs opérationnels connectés en pont dont l'étage de sortie a une faible impédance.

Chaque étage a un gain en tension fixe qui est paramétré en interne par des résistances de contre-réaction (elles sont assez précises et garantissent un écart maximal du gain entre les canaux gauche et droit de seulement 0,5 dB). Ce **gain** est égal à **32 dB** ce qui correspond à une **amplification de 39 fois** du signal d'entrée. Cela signifie qu'en injectant une tension (signal) efficace de 0,1 V à l'entrée, nous obtenons une tension de 3,9 V en sortie.

La puissance de sortie efficace maximale sur un haut-parleur de 8 Ω est de 15 W, la tension efficace en sortie est alors de 10,9 V. Comme le gain en tension est de 39, pour obtenir cette puissance, il faut injecter sur l'entrée une tension (signal audio) d'une valeur de 0,28 V_{eff} soit 280 mV_{eff}.

Attention cependant au fait que, pour chaque canal, le **TDA7297 ne peut délivrer qu'un courant maximal de 2 A**, ce qui pour un régime sinusoïdal signifie un peu plus de 1,4 A. Cette valeur, plus que suffisante pour développer 15 W_{eff} sous 8 Ω, ne permet d'atteindre qu'un maximum de 8 W_{eff} sous 4 Ω.

L'amplificateur peut donc être connecté à des haut-parleurs de 4 Ω, mais la puissance maximale disponible sera inférieure à celle sous 8 Ω. Si possible, utilisez donc des haut-parleurs ou enceintes de 8 Ω pour obtenir une bonne puissance de sortie avec une faible distorsion. Pour obtenir une puissance de 8 W_{eff} sous 4 Ω, il faut appliquer sur l'entrée un signal audio de 198 mV_{eff}.

Vous remarquerez que, pour fonctionner, l'amplificateur nécessite un nombre restreint de composants externes, car en interne il dispose des étages suivants : un réseau de polarisation des entrées non inverseuses des amplificateurs opérationnels nécessaire lorsque l'alimentation est de type mono-tension, deux réseaux R/C (cellules Boucherot) en sortie pour compenser les variations d'impédance des haut-parleurs aux fréquences élevées et éviter des auto-oscillations, et d'un circuit anti-cloc à la mise sous tension.

Cela rend le schéma très simple, comme vous pouvez le voir. En fait, le circuit intégré n'est entouré que de deux condensateurs pour le découplage en continu des entrées L et R (C4 et C5) et d'un diviseur de tension formé par les résistances R2 et R3 avec un condensateur de filtrage pour les fonctions « **Mute** » et « **Standby** ».

Plus précisément, le condensateur **C3** permet, lors de la mise sous tension du circuit, de **désactiver les sorties** des amplificateurs le temps que ce dernier se stabilise afin **d'éviter le classique « cloc »** dans les haut-parleurs, en particulier ceux des basses.

Caractéristiques techniques

- **Tension d'alimentation : de 7 VDC à 18 VDC ;**
- **Puissance de sortie (RMS) : 15 W + 15 W sous 8 Ω ;**
- **Puissance de sortie (RMS) : 8 W + 8 W sous 4 Ω ;**
- **Courant maximal consommé : 2,5 A ;**
- **Courant de repos : 50 mA ;**
- **Réponse en fréquence : de 22 Hz à 100 kHz ;**
- **Distorsion harmonique (@1W) : 0,1 % ;**
- **Distorsion harmonique à la puissance maximale : 1 % ;**
- **Séparation des canaux (diaphonie) : 60 dB ;**
- **Impédance d'entrée : 30 k Ω**
- **Protection thermique ;**
- **Protection contre les courts-circuits en sortie.**

Car au fil du temps cela peut endommager les haut-parleurs (sans compter l'instant stressant à la mise sous tension).

La fonction « **Mute** » est **activée** en mettant la **broche 6 à un niveau logique 0** et la mise en veille (« **Standby** ») est **activée** en mettant la **broche 7 à un niveau logique 0**, de sorte qu'avec seulement 3 composants, nous gérons les deux fonctions.

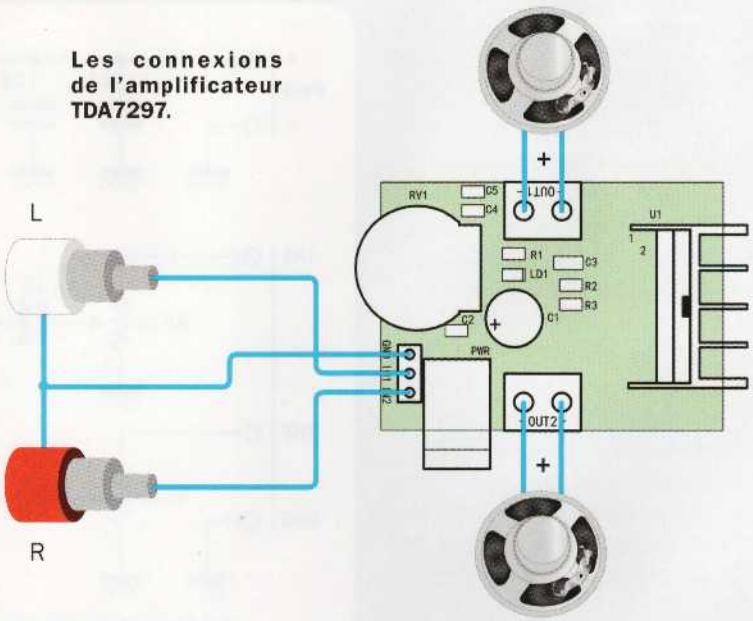
En raison de la présence du condensateur C3, qui est déchargé à l'état initial, lorsque le circuit est alimenté les étages de puissance sont désactivés et la fonction « Mute » bloque le signal audio pendant une durée d'environ 100 millisecondes (le temps que C3 se charge), après quoi le TD7297 fonctionne normalement.

Le condensateur C3 se charge à travers la résistance R2. Lorsque l'amplificateur est éteint (coupure d'alimentation), C3 se décharge à travers R3. S'il n'y avait pas cette résistance, **C3 resterait chargé pendant une longue période** et lors d'un allumage ultérieur de l'amplificateur, les fonctions « **Mute** » et « **Standby** » **n'auraient aucun effet** car les broches 6 et 7 se trouveraient à un niveau haut (C3 ne pouvant se décharger), il y aurait alors un cloc à l'allumage.

Le fabricant recommande d'allumer le TDA7297 en désactivant d'abord la fonction de veille (Standby) et ensuite la fonction de « Mute » (sourdine). Comme le seuil de désactivation de la fonction « Standby » est de 0,9 V, lorsque la tension aux bornes du condensateur C3 atteint environ 1 V, le circuit intégré s'allume.

Cependant le seuil de désactivation de la fonction Mute se situe à 2,9 V environ, il faut donc attendre que la tension aux bornes de C3 atteigne cette valeur pour désactiver le Mute. Ainsi, nous pouvons gérer avec un seul réseau RC les 2 broches de contrôle des deux fonctions, comme cela nous sommes certains de respecter la séquence d'allumage correcte. Il en est de même pour l'arrêt de l'amplificateur, l'utilisation d'un seul réseau est correcte.

Les connexions de l'amplificateur TDA7297.



En effet, le fabricant ST préconise d'abord d'activer le mode Mute, puis le mode veille. Cela est respecté, car lors de l'arrêt du circuit TDA7297, le condensateur C3 se décharge à travers R3 si bien que la tension à ses bornes passe en dessous du seuil du mode Mute (2,9 V) pour continuer à décroitre jusqu'à descendre en dessous de 0,9 V ce qui active le mode « Standby ». L'ordre de la séquence d'extinction est ainsi bien respecté.

Les sorties du circuit intégré sont reliées directement aux haut-parleurs. Bien que le circuit fonctionne sous une seule tension d'alimentation, les condensateurs de découplage ne sont pas nécessaires car les haut-parleurs sont connectés entre les sorties en pont des deux étages de puissance internes de chaque canal (en fait, chaque sortie est constituée en interne par 2 amplificateurs fonctionnant en pont).

Chaque entrée dispose d'un potentiomètre de réglage de volume (en fait, il s'agit d'un double potentiomètre) afin de doser le niveau du signal sur chaque entrée. Le niveau est maximal lorsque les points milieux du potentiomètre se situent vers les points IN1 et IN2 et est minimal lorsqu'ils se situent vers la masse (GND).

Les condensateurs C1 et C2 filtrent la ligne d'alimentation contre les perturbations et l'ondulation résiduelle provenant du secteur.

Ils permettent aussi une régulation de la tension en minimisant les fluctuations de la tension provoquées par les variations de la puissance de sortie qui dépend du niveau et de la nature du signal audio. Par exemple, ces fluctuations sont moins importantes avec de la musique classique qu'avec de la musique moderne.

Pour **obtenir la puissance maximale sous 8 Ω**, la tension d'alimentation doit être d'environ **16,5 V** ou **17 V**.

Sa présence est signalée par l'allumage de la LED D1 polarisée par la résistance R1.

Réalisation pratique

Après vous avoir expliqué le schéma électrique, passons à la construction de l'amplificateur. Ce dernier nécessite un circuit imprimé double face dont les typons et les fichiers Gerber sont disponibles en téléchargement sur notre site dans le sommaire détaillé de la revue (au bas de la page web).

Notez que pour obtenir un encombrement réduit, le montage comporte quelques composants CMS à souder. Le reste des composants sont de type traditionnel, c'est-à-dire le circuit TDA7297, les borniers, le condensateur C1, la prise d'alimentation et le potentiomètre. La construction nécessite donc un équipement approprié,

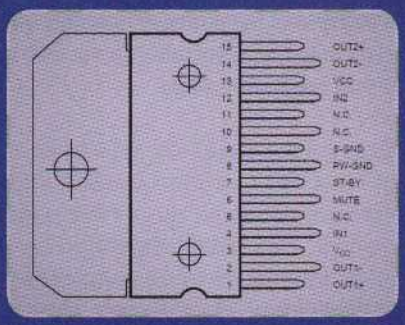
Le circuit intégré **TDA7297**

Le composant sur lequel est basé notre amplificateur à deux canaux est un circuit intégré encapsulé dans un boîtier Multiwatt de 15 broches. Chaque canal comporte deux amplificateurs de puissance configurés en pont (les signaux amplifiés sont égaux en amplitude mais en opposition de phase l'un par rapport l'autre). Il accepte une tension d'alimentation comprise entre 6 VDC et 18 VDC, et peut délivrer un maximum de 2 A en crête sur chaque sortie soit 1,41 A efficaces (régime sinusoïdal).

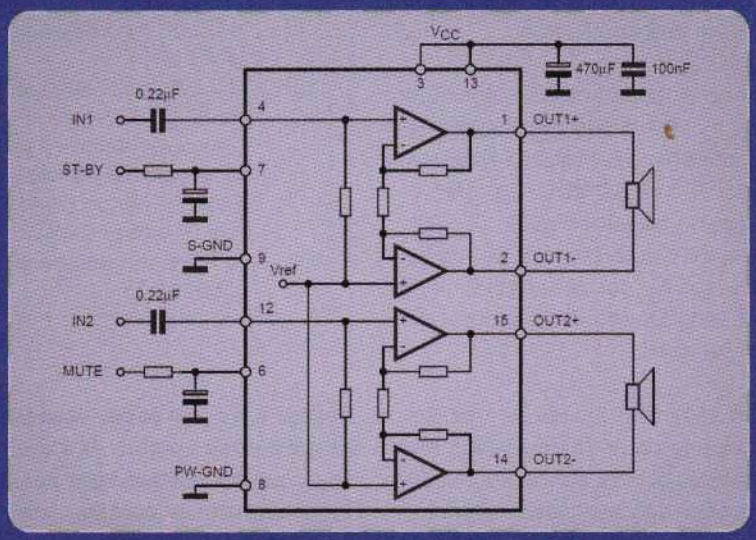
La puissance de sortie maximale par canal sous une charge de 8 Ω et une tension d'alimentation de 16,5 VDC est de 15 W_{eff} (par canal). La **distorsion harmonique** (THD) est de **0,1 % à 1 W sous 8 Ω** pour une fréquence de **1 kHz**. Sous 4 Ω, la puissance est inférieure car le courant en sortie est limité.

Le circuit intégré réalise la plupart des fonctions et dispose en interne de tous les composants nécessaires pour les fonctions de polarisation, de compensation d'impédance de charge (empêche les auto-oscillations à haute fréquence), de stabilisation thermique, d'inversion du signal pour piloter l'étage en pont de chaque canal.

Le **TDA7297 dispose en interne** d'un dispositif de **protection contre les courts-circuits en sortie**, d'une **limitation du courant de sortie**, ainsi que d'une **protection thermique** empêchant la surchauffe du circuit intégré, lorsqu'il fournit une puissance excessive ou lorsqu'il est mal refroidi.



En plus de cela, nous trouvons dans le TDA7297 un circuit de « Mute » et un circuit de « Standby ». Ce dernier permet de désactiver les étages de puissance pour minimiser la consommation qui est de seulement 100 µA dans ce mode, tout en laissant le circuit alimenté. Quant à la fonction « Mute », elle permet de réduire le signal audio appliqué aux entrées en l'atténuant de 80 dB, ce qui correspond à un niveau inaudible en sortie. Ces deux fonctions sont actives en appliquant un niveau bas sur leurs broches respectives, qui est généralement inférieur à 2,9 V pour le mode « Mute » (broche 6) et inférieur à 1 V pour le mode « Standby » (broche 7). L'impédance d'entrée pour chaque canal est de 30 kΩ et l'amplitude maximale du signal d'entrée pour une puissance maximale en sortie sous 8 Ω est de 280 mV_{eff}.



à savoir un fer à souder d'une puissance de 20 W avec une pointe très fine de 0,1 mm ou 0,2 mm de ∅ et de la soudure de 0,5 mm de ∅ pour les composants CMS. Nous vous conseillons d'utiliser du flux de soudure, une bonne loupe et une pince à épiler pour positionner les CMS. Soyez patients et attentifs au marquage des résistances. Avant de souder C3 et D1 qui sont des CMS, vérifiez leur orientation. Si vous vous trompez, il est délicat de dessouder un CMS, donc vérifiez plusieurs fois avant de souder.

Le montage doit être effectué en soudant d'abord les plus petits composants (c'est-à-dire en premier les CMS) comme les résistances et les condensateurs ainsi que la LED.

Ensuite soudez les composants traversant (en dernier le TDA7297 avec son radiateur). Le potentiomètre doit être soudé horizontalement (vous pouvez aussi effectuer un montage sur châssis d'un potentiomètre double avec des câbles blindés reliés au circuit imprimé, la tresse de chaque câble doit être soudée d'un seul côté à la masse afin d'éviter des bouclages de masse qui pourraient produire des bruits parasites dans les HP). Pour l'orientation des composants, reportez-vous à l'encadré intitulé « Plan de montage ».

NB : concernant le TDA7297, il est préférable de le fixer sur le radiateur puis ensuite de le présenter dans son emplacement sur le circuit imprimé. Vérifiez si son positionnement est correct, puis soudez ses broches en le laissant refroidir entre chaque soudure.

Le **radiateur du TDA7297** doit avoir une **résistance thermique d'au moins 8 °C/W**. Utilisez de la graisse au silicone entre la semelle du circuit intégré et le dissipateur.

Il n'est pas nécessaire d'insérer une feuille de mica isolante, mais pensez à ne pas toucher le circuit en fonctionnement et **vérifiez que le radiateur**

n'entre pas en contact avec un objet ou une surface métallique dans le cas d'une mise en boîtier.

Sinon, dans le cas où la mise en boitier est critique (boîtier métallique, manque de place, etc.), vous pouvez insérer une feuille de téflon ou de mica isolante entre le composant et le dissipateur. Dans ce cas, n'oubliez pas d'utiliser également des canons isolants en plastique pour les vis 3MA. Vérifiez à l'aide d'un ohmmètre l'isolation électrique entre le radiateur et la partie métallique du TDA7297.

Une fois l'assemblage terminé, l'amplificateur est prêt à l'emploi. Il ne nécessite pas d'étalonnage ou d'ajustement. Vous pouvez l'alimenter avec une alimentation non régulée capable de fournir une tension de 17 VDC et un courant d'au moins 4 A. Utilisez des haut-parleurs de 8 Ω ou 4 Ω (mais pas en dessous de cette valeur).

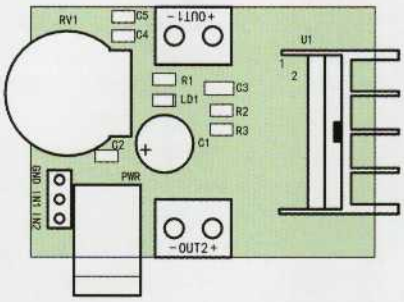
Si vous optez pour des haut-parleurs de 4 Ω, n'oubliez pas d'alimenter le circuit avec une tension comprise entre 11 VDC et 12 VDC. En effet, il n'est plus nécessaire d'avoir 17 VDC, car dans ce cas le courant de sortie dépasse 1,4 A_{eff} par canal et donc la protection de limitation du courant de sortie intervient.

Pour un test rapide, connectez les entrées à une paire de prises RCA (ou mieux si vous pouvez les monter sur un support pour des applications stéréo) en utilisant un câble coaxial blindé.

Rappelez-vous, que pour chaque câble, la tresse (qui forme un bouclier électromagnétique) doit être reliée à la masse du circuit imprimé (que d'un seul côté), les conducteurs internes de chaque câble aux points IN1 et la masse (respectivement à IN2 et la masse).

Une fois cela effectué, connectez aux sorties OUT1 et OUT2, à l'aide de deux câbles plats rouge et noir de 2 x 0,75 mm², deux haut-parleurs de 8 Ω /15 W ou deux enceintes acoustiques ayant une impédance de 8 Ω, peu importe la puissance tant qu'elle n'est pas inférieure à 20 W. Reliez un lecteur MP3 ou mieux un lecteur CD aux entrées de l'amplificateur.

Plan de montage de l'amplificateur TDA7297



Plan de câblage des composants de l'amplificateur TDA7297.

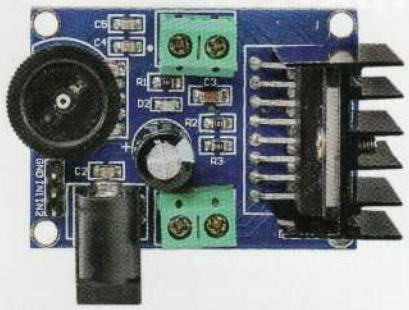


Photo de l'un de nos prototypes de l'amplificateur TDA7297.

Liste des composants de l'amplificateur TDA7297

R1..... 3,3 kΩ boîtier CMS 0805
R2..... 10 kΩ boîtier CMS 0805
R3..... 10 kΩ boîtier CMS 0805
C1..... 470 µF/16 V électrolytique
C2..... 100 nF céramique boîtier CMS 0805
C3..... 10 µF céramique boîtier CMS 1206 (selon le type de condensateur il peut être polarisé, le positif est indiqué par un trait blanc sur le boîtier CMS).
C4..... 220 nF céramique boîtier CMS 0805
C5..... 220 nF céramique boîtier CMS 0805

D1..... LED verte boîtier CMS 0805
U1..... TDA7297

RV1.... potentiomètre stéréo 10 kΩ

Divers

Bornier 2 pôles (x2)
Barrette mâle 3 pôles
Fiche d'alimentation
Dissipateur 8 °C/W
Vis 10 mm / écrous 3MA

Veillez à ne pas relier ensemble les négatifs des sorties de l'amplificateur, car les sorties sont pontées et leurs négatifs ne sont pas référencés à la masse (GND). **Si vous reliez les négatifs des sorties, vous risquez d'endommager le TDA7297.**

Mettez le circuit sous tension, rappelez-vous qu'avant de commencer la reproduction sonore, vous devez tourner le potentiomètre de volume dans le sens inverse des aiguilles d'une montre de façon à ce que le volume soit au minimum, puis tournez progressivement le potentiomètre jusqu'à entendre le son.

Utilisation de l'amplificateur

En ce qui concerne l'application, le montage peut être utilisé comme amplificateur de salon (amplificateur Hi-Fi), ou comme amplificateur de voiture pour autoradio. Il peut aussi amplifier le signal fourni par un lecteur MP3 en utilisation nomade.

Quelle que soit l'application envisagée, rappelez-vous que le boîtier de l'amplificateur doit avoir des orifices pour évacuer la dissipation de chaleur produite par le TDA7297. Evitez également le contact du dissipateur avec d'éventuelles parties métalliques du boîtier, car bien que le radiateur soit relié à la masse, il est toujours préférable d'éviter des bouclages de masses qui pourraient créer des interférences dans les haut-parleurs. N'oubliez pas de connecter la masse de l'alimentation en un seul point du boîtier et d'isoler éventuellement de la partie métallique du boîtier les connexions des entrées RCA et celles de sorties afin d'éviter des courts-circuits. Les négatifs (-) des haut-parleurs ne doivent pas être reliés à la masse (GND). ■

ET 1278

DETECTEUR DE PLUIE

Ce montage a été conçu pour être associé à une commande d'irrigation, à une station météo ou encore à des stores motorisés, il ferme un relais lorsqu'il détecte la pluie.

de Alessandro Sottocornola

L'un des principaux problèmes liés à l'utilisation de systèmes d'irrigation est l'arrivée de la pluie lors de l'exécution d'une séquence d'arrosage car, dans ce cas, il est inutile d'irriguer la pelouse ou les plantes alors que la nature leur fournit déjà de l'eau.

Sans oublier que l'ajout d'eau sur un sol déjà humide à cause des précipitations peut provoquer le débordement des cours d'eau qui à leur tour peuvent inonder des jardins, des terrains voire même des maisons. Un autre problème est la stagnation de l'eau après la pluie, car même dans ce cas, il est inutile d'irriguer le jardin, car il y a suffisamment d'eau.

Pour contourner ces obstacles, une unité de contrôle d'irrigation qui se respecte doit disposer d'un capteur capable de détecter les précipitations et la présence d'eau dans le terrain. Elle doit en outre fournir un signal qui inhibe ou retarde l'exécution des séquences d'irrigation.

Ce dernier aspect est géré par le programme des systèmes d'irrigation récents qui disposent d'une entrée, généralement qui est de type à contact à sec ou par résistance de tirage (pull-up). Dès que cette entrée est activée, l'unité de contrôle saute le cycle d'irrigation ou le retarde pendant une période suffisante (par exemple, quelques heures) afin d'évacuer l'eau en excès. Ce décompte du temps doit commencer à partir du moment où le capteur ne détecte plus la présence d'eau.

Il existe **deux types de capteurs** disponibles dans le commerce pour détecter la pluie : les **capteurs interdigités** et les **capteurs composés de matériau poreux**.

Ceux du premier type sont placés sur le sol et, lorsqu'il pleut, l'eau provoque une **conduction électrique entre les contacts** disposés sur toute la surface du capteur, déterminant ainsi entre les électrodes une **forte baisse de la résistance** par rapport à une condition sèche (dans ce cas la résistance est théoriquement infinie).

En lisant l'état du capteur avec une entrée logique équipée d'une résistance de pull-up, lors d'une situation sèche, le capteur est au repos et son état logique est à un niveau haut (1 logique). Lorsqu'il détecte de l'humidité, la présence d'eau

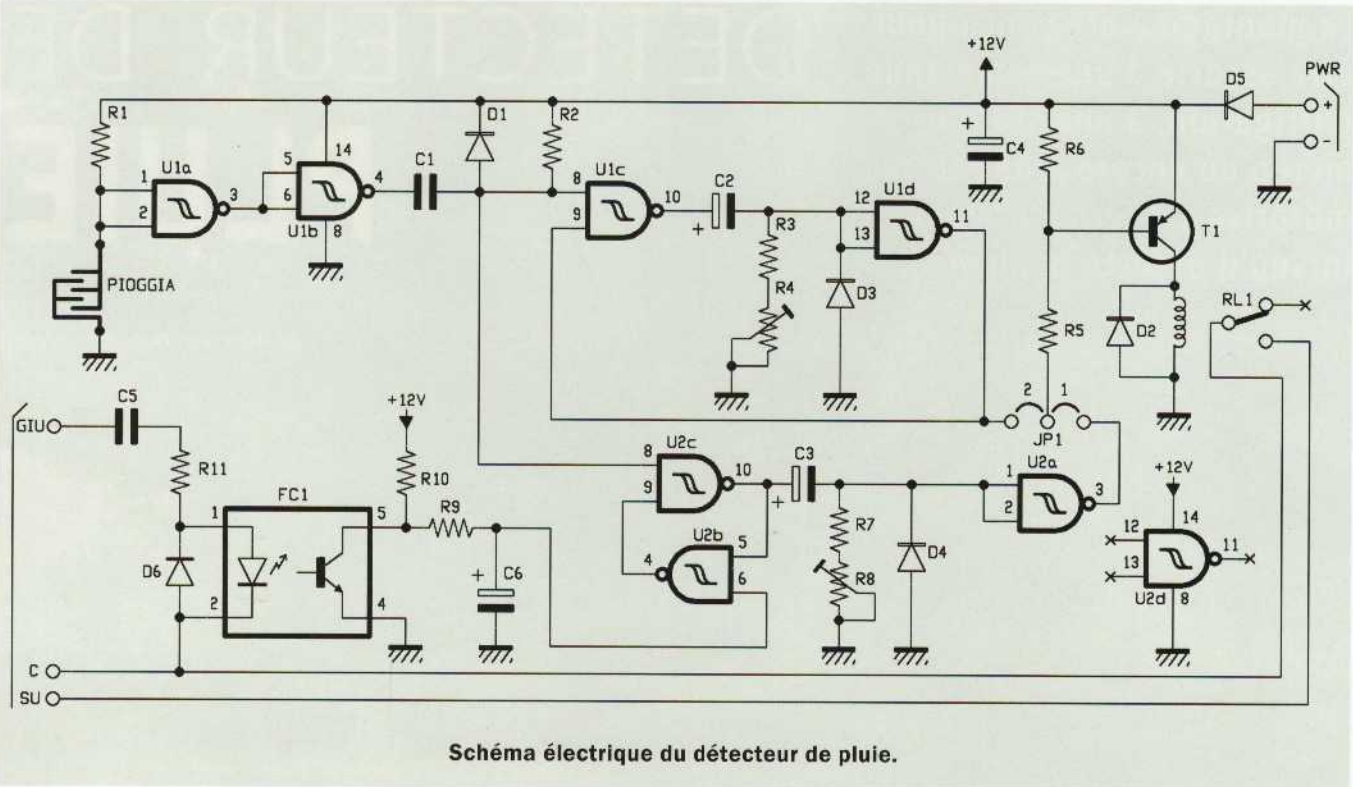


Schéma électrique du détecteur de pluie.

fait varier sa résistance et le capteur se trouve à un niveau bas (0 logique).

La lecture de l'état logique du capteur peut être confiée à un comparateur ou à un simple MOSFET, en connectant une résistance entre l'alimentation positive et la grille.

Quant aux capteurs à base de matériaux poreux, ils sont composés d'un boîtier dans lequel **se dilate un parallélépipède formé de matériau poreux**. Lorsqu'il est mouillé, il pousse un micro-contacteur qui se ferme. Lorsque le matériau sèche, il se contracte et relâche le micro-contacteur, ouvrant ainsi le contact électrique.

Ces capteurs sont très précis et réglables à l'aide d'une vis, le micro-contacteur peut être écarté de sorte qu'il ne soit actionné qu'avec une certaine dilatation du matériau poreux et donc avec une certaine quantité d'eau.

Notre projet

Nous vous proposons de réaliser un capteur du premier type (interdigité), à l'aide d'un minimum d'électronique (sans microcontrôleur et donc sans programmation) qui peut être utilisé non seulement pour arrêter le cycle d'un système d'irrigation, mais aussi pour donner un ordre de fermeture d'un volet motorisé ou encore la mise en marche d'une pompe. En fait, la sortie du capteur est dotée d'un relais disposant d'un contact à sec.

Il ne s'agit donc pas d'un simple capteur, mais d'une véritable commande adaptable à des systèmes d'irrigation, à des mécanismes motorisés, mais aussi à diverses autres applications où il est nécessaire de détecter la présence de pluie ou d'eau.

Par exemple, il est possible de l'utiliser en tant que capteur d'inondation d'un sous-sol ou d'un local.

Le **circuit comprend** essentiellement un **timer** (**monostable**) et un **circuit bistable** permettant d'exécuter deux tâches indépendantes lorsqu'elles détectent le même événement.

La détection d'eau provoque la diminution de la résistance entre les broches 1, 2 du circuit « U1a » (NAND) et la masse, ce qui correspond alors au contact de l'électrode du capteur avec l'eau.

Lorsque l'événement se produit, un timer est déclenché. Ce dernier maintient une condition logique pendant un certain intervalle de temps ajustable dans certaines limites et le circuit bistable reste activé. Cette configuration rend le circuit très polyvalent.

En fait, le timer est utilisé, par exemple, pour bloquer le système d'irrigation pendant un temps déterminé, ou pour donner une commande impulsionnelle à un circuit qui doit actionner un mécanisme motorisé.

Le circuit bistable a été, par contre, conçu pour activer un système d'irrigation ou un mécanisme lorsqu'un état stable est nécessaire, c'est-à-dire une commande qui, une fois l'événement détecté, n'est plus désactivée jusqu'à une intervention manuelle ou lorsqu'un signal de réinitialisation est reçu.

Reportez-vous au schéma électrique qui montre la composition du circuit détecteur de pluie. Il comporte un **circuit monostable** constitué par les portes logiques « **U1c** » et « **U1d** » (deux portes NAND appartenant à un circuit intégré 4093, contenant 4 entrées de type triggers de Schmitt, une solution permettant des commutations plus précises

autour du niveau du seuil logique) et un **circuit bistable** composé par les portes NAND « **U2b** » et « **U2c** ».

Les **deux circuits sont déclenchés** par le buffer (tampon) constitué des NAND « **U1a** » et « **U1b** », ces dernières étant connectées en cascade et configurées en inverseur logique, elles permettent d'acquérir la tension délivrée par le capteur. Lorsque ce dernier est dans un environnement sec, les broches 1 et 2 sont maintenues à un niveau logique haut (1) à travers la résistance de tirage (pull-up) R1.

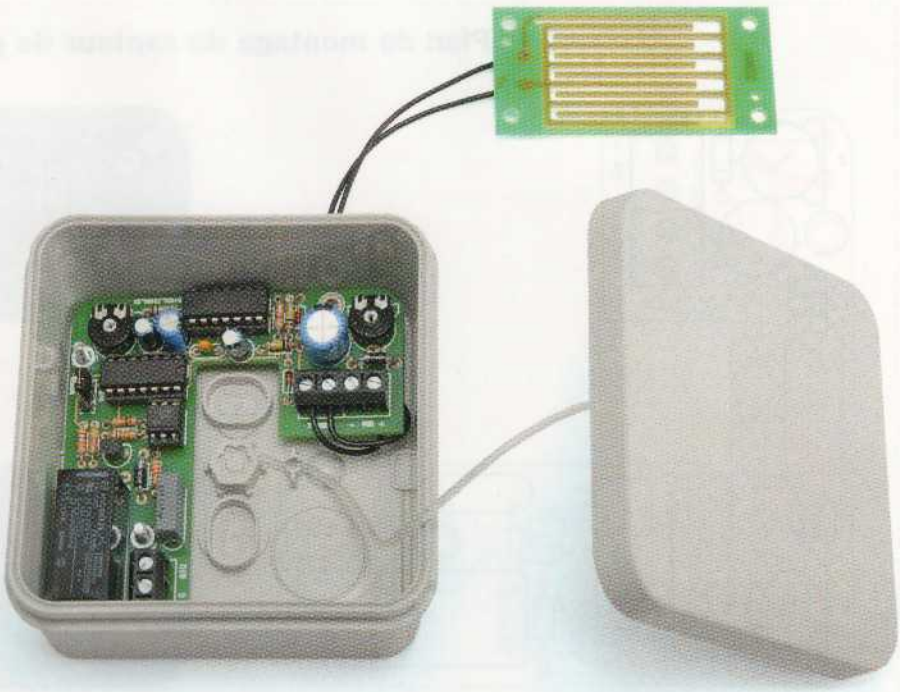
Lorsque le capteur se trouve dans un milieu suffisamment humide, sa résistance interne chute à une valeur telle qu'elle ne peut plus maintenir une tension suffisante entre les broches 1 et 2 et la masse, et donc le capteur ne peut plus maintenir un niveau logique haut entre ses broches.

Dans ce cas, la broche 3 de « U1a » bascule d'un niveau logique bas (0) à un niveau logique haut (1) et la sortie de « U1b » passe à un niveau logique bas (0), entraînant la broche de C1 qui y est connectée à environ 0 V. Cela implique, sur la broche 8 de « U1c », la présence d'une impulsion à un niveau logique 0 d'une durée d'environ R2 * C1, qui excite le circuit monostable et le circuit bistable.

Supposons que C2 soit déchargé et que la sortie de « U1d » soit à un niveau logique haut. Dans ces conditions, « U1c » se retrouve avec une entrée à 0 et l'autre à 1 (alors que précédemment les deux entrées étaient à 1 et la sortie à 0), sa sortie (broche 10) est portée à un niveau logique haut.

Comme C2 est déchargé, ce niveau se retrouve sur les entrées de la porte « U1d » qui est configurée comme une porte NOT. Cela maintient sa sortie à un niveau logique 0, maintenant ainsi la condition actuelle du monostable même si la broche 8 revient à un niveau logique 1 car C1 se charge ou alors le capteur se retrouve dans un milieu sec.

Pendant ce temps, le condensateur C2 commence à se charger à travers les résistances R3 et R4 car la sortie de « U1c » est à un niveau logique 1.



Une chute de tension croissante apparait entre les armatures de C2, si bien que sa broche négative fait chuter la tension présente sur les entrées de « U1d » (les entrées se rapprochent de plus en plus d'un niveau 0) dans un temps environ égal au produit de la somme des résistances et de sa valeur capacitive.

Étant donné que R4 est un trimmer monté en tant que résistance variable, le temps de charge peut être ajusté dans certaines limites afin de l'adapter à vos besoins. Ainsi, il est possible de régler la durée du circuit monostable. Puisque R4 a une valeur de 10 MΩ, le temps peut être réglé entre quelques fractions de seconde et un peu plus d'une douzaine de secondes.

Une fois que C2 a été suffisamment chargé, la sortie de « U1d » passe d'un niveau logique 0 à un niveau logique 1 et donc la broche 9 de « U1c » se retrouve à un niveau logique 1.

À ce stade, puisque C1 est chargé et que la broche 8 est également à un niveau logique 1, la sortie de « U1c » revient à un niveau logique 0 et provoque la décharge rapide (presque instantanée) du condensateur C2 à travers la diode D3, qui est polarisée en direct et est conductrice du moins tant que la chute de tension à ses bornes dépasse 0,6 V. Cela réinitialise le monostable, qui est prêt pour un nouveau cycle.

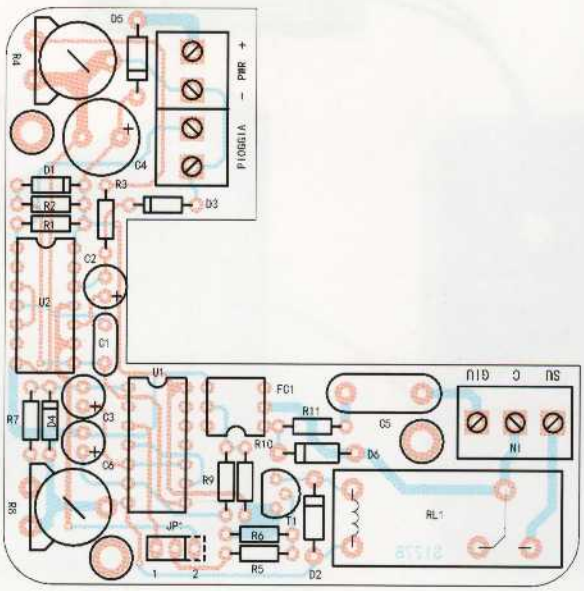
Si le cavalier JP1 est positionné entre le point central et le contact 2, lorsque le monostable est excité et pendant toute la durée de sa temporisation, le transistor T1 est polarisé en raison de l'état bas de la sortie « U1d » et entre donc en saturation, alimentant ainsi la bobine du relais RL1, ce qui a pour effet de fermer les contacts « C » et « SU » du bornier. En reliant les contacts « C » et « SU » à un mécanisme motorisé, il est alors possible de le commander.

Examinons maintenant le circuit bistable, qui peut être utilisé comme une alternative au circuit monostable, toujours en pilotant le relais. Pour être en mode bistable, il faut placer le cavalier JP1 entre le point central et le contact 1.

Lors de la mise sous tension du circuit, le condensateur C6 est initialement déchargé, la broche 6 de la porte NAND « U2b » se trouve à un niveau logique 0. Ainsi, la sortie de la NAND est forcée à un niveau logique 1.

Supposons que le capteur soit dans un milieu sec, la broche 8 de la porte NAND « U2c » est également à un niveau logique 1, la sortie de cette dernière est donc à 0 et renvoie cette condition sur la broche 5 de « U2b ».

Plan de montage du capteur de pluie



Plan de câblage des composants du capteur de pluie.

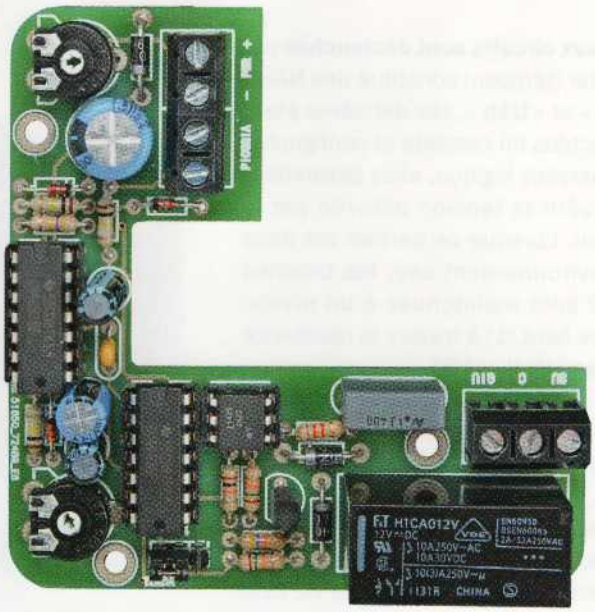


Photo de l'un de nos prototypes du capteur de pluie.

Liste des composants

R1..... 470 kΩ
R2..... 100 kΩ
R3..... 100 kΩ
R4..... trimmer 10 MΩ monotour
R5..... 15 kΩ
R6..... 47 kΩ
R7..... 100 kΩ
R8..... trimmer 10 MΩ monotour
R9..... 10 kΩ
R10.... 15 kΩ
R11.... 8,2 kΩ

C1..... 100 nF céramique
C2..... 4,7 µF/16 V électrolytique
C3..... 22 µF/16 V électrolytique
C4..... 1000 µF/16 V électrolytique
C5..... 100 nF/400 V polyester pas de 10 mm
C6..... 4,7 µF/16 V électrolytique

D1..... 1N4148
D2..... 1N4007
D3..... 1N4148
D4..... 1N4148
D5..... 1N4007
D6..... 1N4007

T1..... BC557
U1..... 4093
U2..... 4093
FC1.... 4N25
RL1.... FTR-H1CA012V 1RT 12V

(https://fr.rs-online.com code commande : 790-3329)

Divers

Bornier 2 pôles pas de 5 mm (x2)
Bornier 3 pôles pas de 5 mm
Barrette mâles 3 pôles
Support CI 2 x 3 broches
Support CI 2 x 7 broches (x2)
Cavalier

NB : les typons des circuits imprimés à l'échelle 1 sont disponibles en téléchargement dans le sommaire détaillé de la revue.

Dans ces conditions, nous sommes sûrs que la sortie de « U2b » est maintenue à un niveau logique 1.

Supposons maintenant que le capteur soit dans un milieu suffisamment humide, nous trouvons donc un niveau logique 0 sur la sortie de « U1b », ce qui détermine une condition de niveau bas entre le nœud « D1-C1 » et donc sur la broche 8 de « U2c ». De ce fait, la broche 10 passe à un niveau logique haut et, à travers le condensateur C3, passe une impulsion de niveau haut qui maintient les entrées de « U2a » à un niveau haut jusqu'à ce que, lors de la charge de C3, la tension aux bornes de la résistance série formée par R7 et R8 ne chute pas en dessous de 1/4 de la tension d'alimentation.

Cela se produit généralement pour une période égale à τ, qui est la constante de temps et qui est égale au produit :

$\tau = (R7+R8) * C3$

La constante de temps **τ** est exprimée en **secondes** si la somme des **résistances** est exprimée en **mégaohm** (MΩ) et la **capacité** de C3 en **microfarad** (µF).

Tant que le niveau haut est présent sur les broches 1 et 2 de « U2a », la broche 3 de « U2a » reste à un niveau logique 0, déterminant également dans ce cas la saturation du transistor et l'activation du relais, ce qui ferme les contacts « C » et « SU ».

Comme R8 est un trimmer, le temps pendant lequel la sortie de « U2a » reste à un niveau logique 0, et donc la durée d'activation du relais, est réglable entre

quelques fractions de seconde et plus de 10 secondes.

Une fois que C3 est suffisamment chargé, une tension proche d'un niveau logique 0 est présente sur les broches 1 et 2 de « U2a », le relais revient alors au repos car la broche 3 de « U2a » passe à niveau haut et bloque le transistor T1.

Contrairement à ce qui se passe avec le circuit monostable qui se trouve dans la partie haute du schéma, le circuit bistable n'est pas redéclenchable, en ce sens que le cycle que nous venons de décrire ne se produit qu'une fois.

Les portes « U2c » et « U2d » forment un circuit bistable qui, une fois qu'il reçoit une impulsion logique à 0, reste dans la condition jusqu'à une réinitialisation qui a lieu lorsque la broche 6 de U2 est amenée à un niveau logique 1.

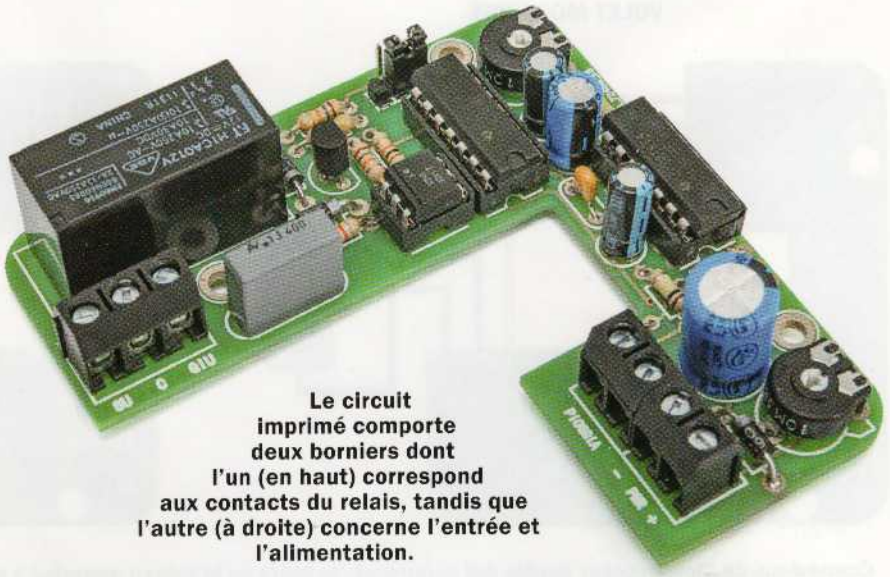
Cela se produit si l'entrée GIU est alimentée par une tension alternative. Cette entrée est conçue pour réinitialiser le circuit monostable lorsqu'une commande manuelle de l'interrupteur qui alimente le mécanisme motorisé a lieu. Cela permet de préparer les circuits bistable et le monostable à recommencer un nouveau cycle.

En résumé, le circuit monostable convient pour effectuer une commande temporisée d'un volet motorisé ou d'un système d'irrigation, tandis que le circuit bistable (en bas du schéma) effectue une commande stable qui est désactivée si nous décidons, par exemple, de commander à nouveau le volet motorisé.

La façon dont le circuit est conçu fait que si la pluie continue de mouiller le capteur, nous pouvons par exemple remonter un volet motorisé mais il redescendra ensuite automatiquement dès lors que l'interrupteur « DOWN » sera relâché.

Le circuit utilisé pour détecter la commande et réinitialiser la bascule est constitué d'un optocoupleur contenant une LED servant de dispositif d'entrée et d'un transistor NPN en sortie.

En reliant le point GIU à la broche « UP » (montée) de l'interrupteur et le point C au commun, et en alimentant les deux contacts du moteur, lorsque l'interrupteur est actionné, les contacts sont soumis au 220 VAC. La tension alternative traverse le condensateur C5 (avec une perte) pour atteindre la résistance R11.



Le circuit imprimé comporte deux borniers dont l'un (en haut) correspond aux contacts du relais, tandis que l'autre (à droite) concerne l'entrée et l'alimentation.

Ces deux composants introduisent une impédance, dont la **partie imaginaire est la réactance capacitive**, qui limite le courant dans la LED de l'optocoupleur.

Le condensateur vaut 100 nF, la fréquence du secteur étant de 50 Hz, donc la **réactance capacitive** vaut :

Xc = 1 / (2 π f C)
Xc = 1 / (6,28 x 50 Hz x 0,1 µF)
= 31,85 kΩ

En série avec la résistance R11, dont la valeur est de 8,2 k Ω, le condensateur forme une impédance d'une valeur suffisante pour permettre la polarisation de la LED à l'intérieur de l'optocoupleur, avec un courant d'environ 5 mA.

Ce dernier est suffisant pour saturer le transistor interne de l'optocoupleur, permettant ainsi de décharger le condensateur C6, et donc de stabiliser l'état logique de la broche 6 de U2. Le condensateur C6 filtre aussi toutes les perturbations susceptibles de provoquer une fausse commutation de la bascule.

La diode en parallèle avec la LED de l'optocoupleur sert à protéger celle-ci des alternances négatives, pendant lesquelles elle doit être bloquée. La diode D6 doit supporter des tensions inverses supérieures à 300 V.

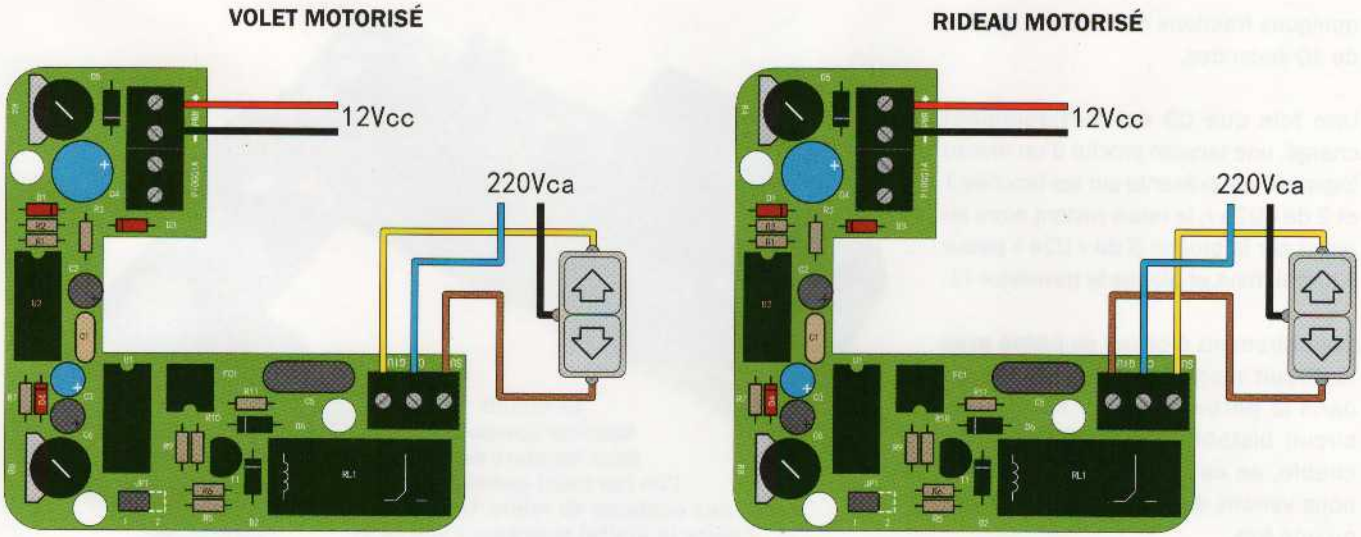
L'optocoupleur permet d'isoler galvaniquement le circuit soumis à la tension secteur de celui de la basse tension qui alimente les portes logiques, l'étage de sortie de FC1 et le transistor T1, ainsi que le capteur. Le circuit basse tension doit être alimenté avec une tension continue allant de 12 VDC à 14 VDC entre les points + et - PWR.

La diode D5 protège la ligne d'alimentation continue contre toute inversion de polarité qui pourrait se produire accidentellement lors de la connexion aux points + et - PWR.

Réalisation pratique

Passons maintenant à la construction du circuit, qui nécessite la préparation (ou l'achat, puisque tout le matériel est disponible sous forme de kit) d'un circuit imprimé double face dont les typons peuvent être téléchargés sur notre site web dans le sommaire détaillé de la revue (en bas de la page web).

Une fois le circuit imprimé gravé et percé, vous pouvez commencer par souder les composants ayant un profil bas, c'est-à-dire les résistances et les diodes en respectant la polarité de ces dernières. Continuez en soudant les condensateurs non polarisés, les supports des circuits



Connexions de l'interrupteur double qui commande le store ou le rideau motorisé à notre circuit. Notez que dans le premier cas (schéma de gauche), les connexions sont inversées par rapport au second cas, car s'il pleut, le rideau doit être levé et le volet baissé.

intégrés (deux de 2 fois 7 broches pour les 4093 et un de 2 fois 3 broches pour l'optocoupleur), les trimmers, la barrette à 3 pôles pour le cavalier, les condensateurs polarisés en prêtant attention à leur orientation (le + correspond à la patte la plus longue), les borniers et le relais.

Reportez-vous à l'encadré intitulé « Plan de montage du capteur de pluie » afin d'orienter correctement les semi-conducteurs et les condensateurs électrolytiques. Une fois les soudures terminées, insérez les circuits intégrés dans leurs supports respectifs, en orientant correctement leur repère en forme de « U ».

À ce stade, vous devez vous procurer un boîtier en plastique (de préférence étanche) pour protéger le circuit et le capteur de pluie, dont les deux fils doivent être connectés au bornier dénommé « PIOGGIA ».

Le circuit imprimé doit être maintenu à l'aide de vis insérées dans le fond du boîtier, c'est-à-dire avec des entretoises et des écrous 3MA. Les dimensions du boîtier que nous avons utilisé sont de 80 mm x 80mm x 45 mm. Il est doté de passe- câbles pour les fils.

Utilisation

Le raccordement au système à contrôler doit être effectué comme indiqué dans les schémas de câblage figurant à la page précédente, qui montrent les systèmes pour volets motorisés et pour rideaux (stores) motorisés.

Tous les fils du boîtier doivent sortir des passe- câbles prévus à cet effet, ces derniers seront ensuite remplis de silicone afin de protéger le circuit de l'humidité, de l'eau ou de la poussière. Le silicone doit sécher au moins deux heures avant de bouger les fils.

Une fois l'assemblage terminé, le capteur de pluie doit être placé au sol ou sur la pelouse, puis relié au boîtier avec des fils ayant une longueur désirée, les fils doivent avoir une très faible résistance dans le cas où la longueur excède 10 mètres. La longueur n'est pas critique, **mais ne dépassez pas 20 mètres afin d'éviter les interférences** provenant du secteur 230 VAC pouvant introduire des perturbations telles qu'une commutation non désirable des entrées des portes logiques (U1).

Si vous devez avoir de très longues connexions et que vous voulez éviter le bruit, utilisez un câble coaxial blindé. Dans ce cas, les deux fils doivent être connectés au bornier nommé « PIOGGIA » sur le circuit imprimé et la tresse de blindage du câble doit être connectée à la masse de l'alimentation du circuit.

Si le montage doit servir de détecteur d'inondation d'un sous-sol ou d'un local, posez le capteur sur le sol (peut-être en le collant avec un mastic transparent en silicone).

Câblez les fils au boîtier en les isolant. Vous pouvez alors relier les contacts du relais à l'entrée d'un système d'alarme ou à une entrée d'une télécommande GSM (série TDG) afin d'envoyer un SMS en cas d'inondation.

Notez que le capteur peut également être utilisé pour détecter d'autres liquides autre que l'eau (du moment qu'ils sont électriquement conducteurs) tels que des huiles, des détergents, etc.

Par contre le capteur ne fonctionne pas avec l'eau distillée, car théoriquement, elle est exempte d'ions dissous et donc elle n'est pas électriquement conductrice.

Dans tous les cas, évitez d'immerger le capteur dans des liquides très inflammables (solvants, essences), car même s'il fonctionne à basse tension, il est préférable d'éviter des risques d'incendie.

Si vous souhaitez utiliser le capteur pour communiquer les conditions de pluie à une station météo, utilisez le circuit monostable (le cavalier JP1 sur le contact 2) avec une temporisation courte, environ une demi-seconde. Il en est de même si vous souhaitez connecter le circuit à une alarme. ■

Nous utilisons un PIC10F322 dans une carte de démonstration qui vous permettra de tester ses fonctionnalités en réalisant un contrôle de la température.

de Vincenzo MENDOLA

THERMOSTAT À MICROCONTRÔLEUR

Lorsque nous parlons de microcontrôleur, nous pensons aujourd'hui à des composants très complexes et performants, en oubliant qu'il existe toute une série de familles de microcontrôleurs 8 bits très simples, destinées à des applications sans exigences particulières, proposées à des prix très abordables destinées aux expérimentateurs ou amateurs. Ces composants économiques, mais pourtant essentiels, peuvent résoudre des applications relativement complexes. Ce n'est pas un hasard si le secteur automobile utilise aujourd'hui des dizaines, voire des centaines de microcontrôleurs, parmi lesquels des microcontrôleurs 8 bits de petite taille pour la gestion des sous-systèmes embarqués.

Dans cet article, nous souhaitons vous montrer à quel point il n'est pas toujours nécessaire de créer des applications électroniques nécessitant un contrôle plus ou moins avancé, mais au contraire d'utiliser des circuits de prototypage tels que ceux de la famille Arduino ou RaspberryPi, ainsi que des montages simples dotés de petits microcontrôleurs. Les principaux fabricants de microcontrôleurs ont généralement au moins une famille de microcontrôleurs 8 bits dans leur catalogue.

Nous vous présentons ici un produit de chez Microchip, le PIC10F322, dont les caractéristiques nous ont permis de créer un thermostat simple, polyvalent, mais aussi d'autres applications. Le projet que nous vous présentons est en fait une carte de démonstration (demoboard) qui permet de tester certaines des fonctionnalités intéressantes du microcontrôleur.

Parmi les nombreuses qualités du PIC10F322, nous pouvons noter qu'il peut être programmé en langage C sans trop de difficulté, en utilisant le compilateur XC8 ainsi que l'environnement de développement MPLAB-X dont nous consacrons un cours dans la revue (reportez-vous aux numéros 141 à 144 ainsi qu'au cours décrit dans ce numéro).

Notre microcontrôleur PIC10F322

Examinons le **PIC10F322**, en donnant un aperçu de ses caractéristiques. Ce composant est disponible en version 6 broches dans un boîtier (package) **SOT-23** et en deux versions 8 broches, dans les packages PDIP et DFN. Nous avons choisi la très petite version SOT-23, facilement soudable à la main, pour montrer le potentiel de ce microcontrôleur de seulement 6 broches, dont deux sont utilisées pour l'alimenter (VDD et VSS). Le microcontrôleur dispose d'une mémoire Flash d'une capacité de 0,896 ko et d'une mémoire RAM de 64 octets. Cela semble ridicule, mais en réalité cela est plus que suffisant pour réaliser de nombreuses applications.

La version que nous avons choisie est dotée de 4 entrées/sorties GPIO, 3 convertisseurs A/D de 8 bits, 2 timers de 8 bits, 2 modules PWM de 10 bits capables de travailler jusqu'à 16 kHz, une référence de tension, un module de circuits logiques configurable et qui inclut les fonctions logiques suivantes : AND, OR, XOR, D Flop, D Latch, SR, JK, un oscillateur à contrôle numérique et un générateur d'onde complémentaire.

L'ensemble fonctionne grâce à l'oscillateur interne RC qui peut être réglé entre 31 kHz et 16 MHz. Si vous souhaitez réduire le nombre de composants externes nécessaires au fonctionnement, la tension d'alimentation est comprise entre 1,8 V et 3,6 V pour la version PIC10LF (fonctionnement direct sur pile ou batterie) et entre 2,3 V et 5,5 V pour la version PIC10F.

Ce microcontrôleur fait partie de la série XLP (extra low power) de Microchip. Il se caractérise par une consommation typique de 25 µA à 1 MHz, alimenté sous une tension de 1,8 V. En veille, il consomme un courant de seulement 20 nA pour la version « LF » alimentée sous 1,8 V. Pour plus de détails sur les caractéristiques et l'utilisation des périphériques intégrés au microcontrôleur, reportez-vous à la documentation technique disponible à l'adresse suivante :

http://ww1.microchip.com/downloads/en/DeviceDoc/40001585D.pdf

En lisant en détail la documentation, vous découvrirez une autre caractéristique intéressante de ce microcontrôleur. Il intègre un capteur de température que nous avons décidé d'utiliser dans notre projet afin de montrer le potentiel du PIC10F322.

Schéma électrique

Une fois la description du microcontrôleur terminée, nous allons aborder celle de la carte d'expérimentation que nous avons réalisée. Reportez-vous au schéma électrique du thermostat, vous pouvez voir que le PIC est entouré de divers composants et cavaliers pour sélectionner l'utilisation de ses entrées/sorties. Les sorties du PIC sont capables de piloter directement des LED, c'est pour cette raison que nous avons inséré dans notre carte deux LED (LD1 et LD2) qui peuvent être utilisées pour visualiser des états. Les broches relatives (RA1 et RA0) pilotent les LED à travers deux résistances de limitation de courant R3 et R4. Les cavaliers **JP1** et **JP2** permettent de **désolidariser les LED des signaux de programmation** afin d'obtenir un flashage correct de la mémoire du PIC. En effet, **RA1** et **RA0** ont respectivement la fonction de **PGC** et **PGD** pour l'interface de programmation en circuit ICSP.

La broche **RA2** du microcontrôleur est reliée à la **base** du transistor **T1** (NPN). La résistance **R2** et le condensateur **C2** forment une **cellule RC introduisant une constante de temps** permettant de créer un faible retard afin d'assurer un **actionnement stable du relais**. Le collecteur de T1 pilote la bobine du relais,

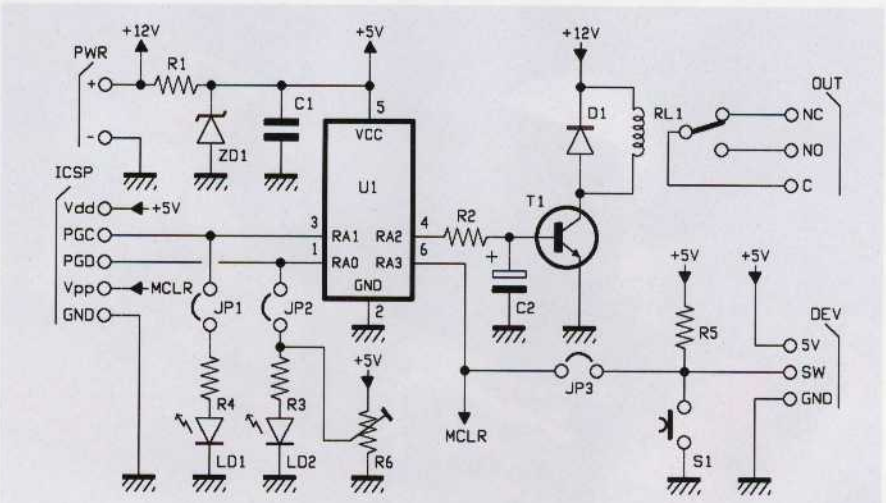


Schéma électrique de notre thermostat à PIC10F322.

dont les 3 contacts C, NC et NO sont accessibles de l'extérieur afin de contrôler différents types de charge.

NB : les lettres suivantes signifient : **C** (Common = Commun), **NO** (Normaly Opened = Normalement Ouvert), et **NC** ou **NF** (Normaly Closed = Normalement Fermé).

Dans notre cas, si nous voulons fabriquer un thermostat, nous pouvons alimenter un ventilateur au-dessus d'une certaine température ou un chauffage en dessous d'un seuil de température défini au préalable. En parallèle sur la bobine du relais, nous trouvons une diode de recirculation ou de roue libre qui protège la jonction base-collecteur du transistor lorsque celui-ci passe d'un état de conduction à un état bloqué. En effet, **le caractère inductif de la bobine fait que, à chaque fois que le transistor passe d'un état de conduction à un état bloqué, elle tente de maintenir un courant qui s'oppose à la cause qu'il l'a créé** (coupure d'alimentation), il est donc de **polarité inverse** à celui d'alimentation.

Toujours dans l'optique de réaliser la fonction de thermostat, nous avons prévu le **trimmer R6**, qui peut être inséré via le cavalier **JP2**, et dont le **point milieu est relié au convertisseur A/N** du microcontrôleur. Le trimmer est alimenté par la même tension de 5 V qui fait fonctionner le microcontrôleur. Si nous voulons **effectuer une lecture de la tension** (utiliser le convertisseur A/D) par l'intermédiaire de R6, **nous devons déconnecter la LED LD2 ainsi que la résistance R3** qui se trouvent en parallèle sur le circuit du trimmer. Le trimmer R6 peut être utilisé pour définir un seuil à partir duquel vous devez activer, par exemple un ventilateur, via le relais.

Dans le montage, nous avons inséré le bouton poussoir S1 avec sa résistance R5 de pull-up. Ce bouton est fonctionnel lorsque le cavalier J3 est fermé. Cependant, lors de **la phase de programmation, J3 doit être ouvert** car la ligne à laquelle le bouton est connecté est également reliée à la broche MCLR (ICSP). En appuyant sur S1, la ligne RA3/MCLR du microcontrôleur prend un niveau logique 0 alors que lorsqu'il est relâché la ligne passe à un niveau haut. En plus du bouton S1, le **connecteur DEV permet une commande externe** de type TOR provenant par exemple d'un contact d'un relais ou d'une sortie de type collecteur ouvert. Les broches du connecteur DEV ne doivent pas être utilisées en même temps que le bouton S1 (l'un ou l'autre mais pas les deux à la fois).

Bien évidemment, afin d'assurer la programmation correcte en circuit du microcontrôleur, vous devez utiliser un programmateur approprié. Rappelez-vous que les **trois cavaliers JP1, JP2** et **JP3** sont utilisés pour **déconnecter**, lors de la phase de **programmation**, les broches **PGC**, **PGD** et **MCLR** du PIC du reste du circuit et qui sont utilisées normalement pendant le fonctionnement du circuit. **Ces cavaliers seront donc retirés lors de la phase de programmation du PIC** (via le connecteur ICSP) et devront être insérés une

fois la programmation terminée (fonctionnement normal du circuit).

Le montage peut être alimenté en appliquant une tension continue de 12 VDC sur la fiche d'alimentation « PWR ». Cette tension est ensuite limitée et stabilisée par la diode zener D2 de 5,1 V qui, grâce à la résistance R1 qui limite le courant, alimente le microcontrôleur U1 et le reste du circuit. La tension de 5,1 V ainsi obtenue est filtrée par le condensateur C1. Ce dernier supprime toute perturbation qui pourrait apparaître sur la ligne d'alimentation. Notez que la tension d'alimentation de 12 V alimente directement la bobine du relais et le circuit de commande formé par le transistor T1.

Notez également que le microcontrôleur, qui est alimenté en 5,1 V, pourrait également fonctionner à des tensions plus faibles, en utilisant les microcontrôleurs de la série LF, sous réserve que les niveaux logiques de l'interface à laquelle est reliée le thermostat sont compatibles avec la tension d'alimentation.

Réalisation pratique

Après avoir décrit le schéma, passons à la partie pratique. Ceux qui assembleront le circuit devront être équipés d'un fer à souder de bonne qualité pour composants CMS, de préférence une station de soudage serait l'idéal. Cependant, un fer à souder CMS de type stylet avec une pointe de 0,2 mm et d'une puissance de 20 à 25 watts convient parfaitement.

Les composants utilisés sont tous de type traversant, à l'exception du microcontrôleur PIC10F322, qui se présente sous la forme d'un boîtier CMS SOT-23 à 6 broches. Il peut constituer un problème critique pour ceux qui n'ont aucune compétence manuelle avec les CMS. Il devra être soudé en premier (pour ne pas être gêné par les autres composants beaucoup plus imposants), de manière délicate et en laissant refroidir quelques secondes entre chaque soudure de chacune des 6 broches. Pour le souder, nous vous rappelons les recommandations habituelles. Utilisez des flux de qualité (par exemple le Kester 951) que vous appliquez en petite quantité sur le circuit avant de le souder.

Plan de montage du thermostat

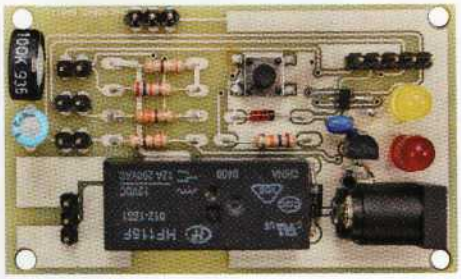
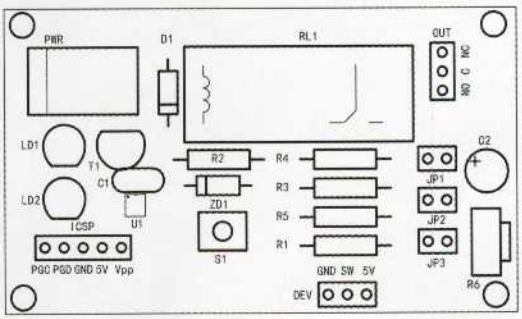


Photo de l'un de nos prototypes du thermostat à microcontrôleur PIC10F322. Ce dernier se trouve à droite de la LED jaune.



Plan de câblage des composants du thermostat à microcontrôleur PIC10F322. Ce dernier doit être soudé en premier en prêtant attention à ce qu'il n'ait pas de pont de soudure entre les broches.

Liste des composants

R1..... 390 Ω
R2..... 27 k Ω
R3..... 330 Ω
R4..... 330 Ω
R5..... 27 k Ω
R6..... trimmer 10 kΩ
C1..... 100 nF céramique
C2..... 10 µF/63 V électrolytique
LD1.... LED rouge 5 mm
LD2.... LED jaune 5 mm
ZD1.... Zener 5,1 V/400mW
D1..... 1N4007

T1 ..... BC547
RL1.... Relais miniature Hongfa HF115F (conrad.com)
DEV.... déviateur (sur fils volants)
S1..... microswitch
U1..... PIC10F322T-I/OT

Divers :

Fiche d'alimentation
Barrette mâle 2 pôles (x3)
Barrette mâle 3 pôles (x2)
Barrette mâle 5 pôles
Barrette femelle 3 pôles

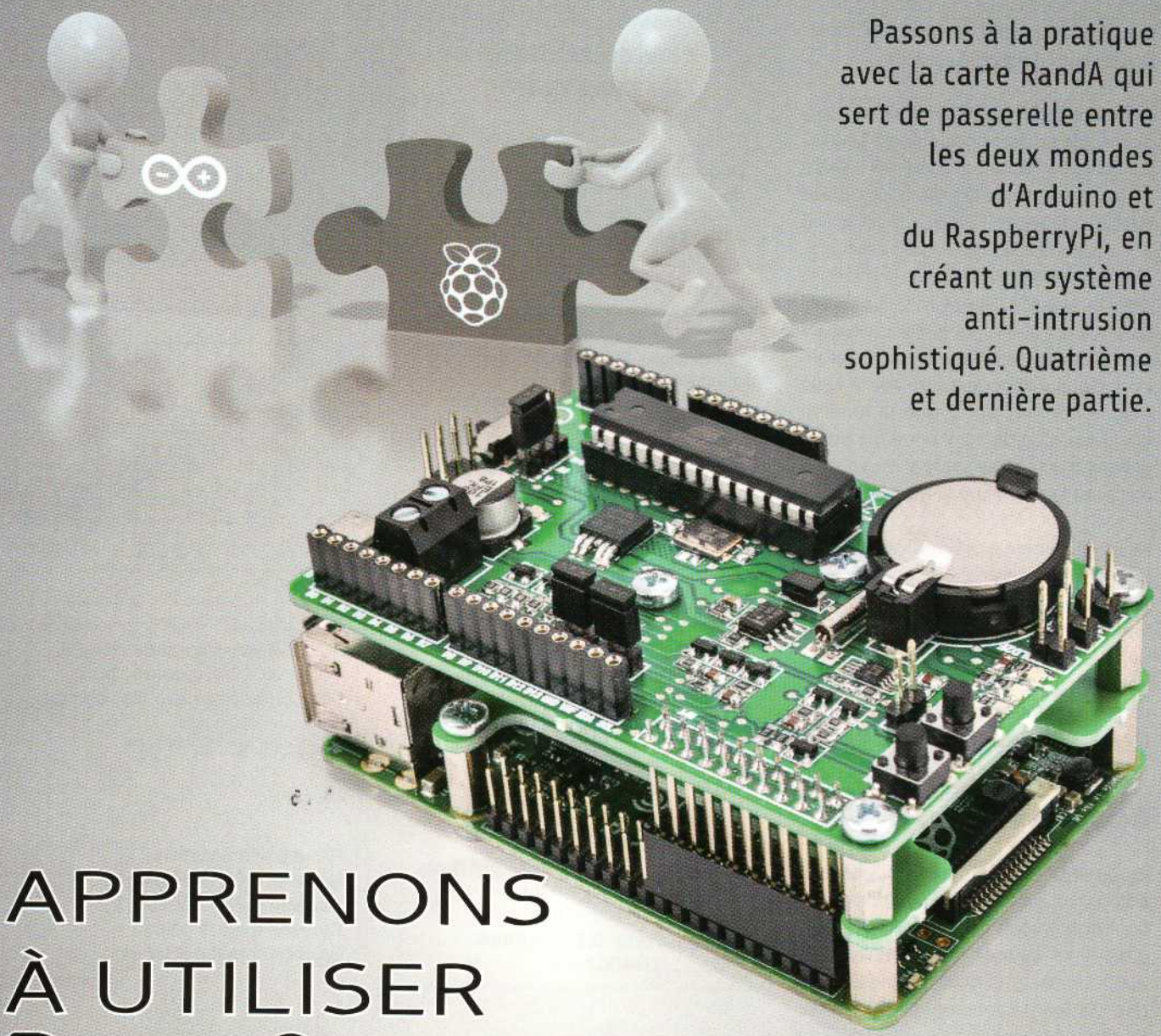
Utilisez un fer CMS à pointe conique très fine et enfin ne vous attardez pas sur les soudures afin d'éviter une surchauffe du microcontrôleur. Une fois le PIC soudé, vérifiez attentivement à l'aide d'une loupe s'il n'y a pas de pont d'étain entre les broches.

Pour l'orientation correcte des composants polarisés, les LED, les diodes, le transistor et le condensateur électrolytique (ainsi que le microcontrôleur), référez-vous à l'encadré intitulé « Plan de montage du thermostat ». Le repère du PIC en forme de point (broche 1) sur le boîtier doit faire face à la LED LD2.

N'oubliez pas de souder les barrettes au pas de 2,54 mm pour réaliser les cavaliers. Pour les contacts du relais, vous pouvez laisser les emplacements libres ou utiliser un bornier en fonction de vos besoins, il en va de même pour le connecteur DEV.

Le firmware

Pour l'application du thermostat, vous trouverez sur notre site dans le sommaire détaillé de la revue les fichiers du projet, un exemple de programme utilisant une seule LED (LED1), un inverseur (le commun au bouton SW, le contact NO au 5 V et le NF à GND) et le trimmer pour régler le seuil de la température à partir duquel le relais sera activé et commutera donc la charge. Notre carte peut être utilisée pour implémenter un thermostat, une minuterie (timer), une sortie PWM, etc. adapter pour chacune des fonctions. En ce qui concerne le timer, celui-ci permet de définir la durée et le bouton lance le décompte du temps. ■



Passons à la pratique avec la carte RandA qui sert de passerelle entre les deux mondes d'Arduino et du RaspberryPi, en créant un système anti-intrusion sophistiqué. Quatrième et dernière partie.

APPRENONS À UTILISER RANDA

de Daniele DENARO

RandA est la carte que nous avons réalisée (reportez-vous aux numéros 142 à 144) afin de faire coopérer Arduino et RaspberryPi. Elle se monte sur ce dernier et ouvre un monde de possibilités qui n'existerait pas avec les deux unités prises séparément.

Pour énumérer les utilisations possibles, il suffit d'un peu d'imagination, car elle contient des éléments qui permettent d'utiliser le RaspberryPi dans une application typiquement réservée à Arduino (et vice versa).

Avant d'aborder la mise en œuvre d'une application complexe et complète, commençons par quelques idées concrètes, c'est-à-dire une première application avec un périphérique très répandu et très bon marché, une webcam.

Surveillance vidéo

En utilisant une webcam disposant d'une interface USB standard et un servomoteur pour modélisme, il est possible de construire une plate-forme dotée d'une webcam pouvant surveiller une zone avec un champ de vision de 180 degrés (voir les figures 1a et 1b).

Vous devez ensuite connecter la webcam au RaspberryPi, via une prise USB, et les trois fils du servomoteur à la carte RandA, en vous assurant que le fil rouge (central) soit relié à la broche « 5 V », le fil brun à la masse GND et enfin le fil jaune à une broche de pilotage configurée en sortie, par exemple la broche D10. Pour visualiser l'image de la caméra via le web, nous pouvons utiliser le logiciel « **Motion** » que nous

Figure 1a : éléments de contrôle de l'environnement.

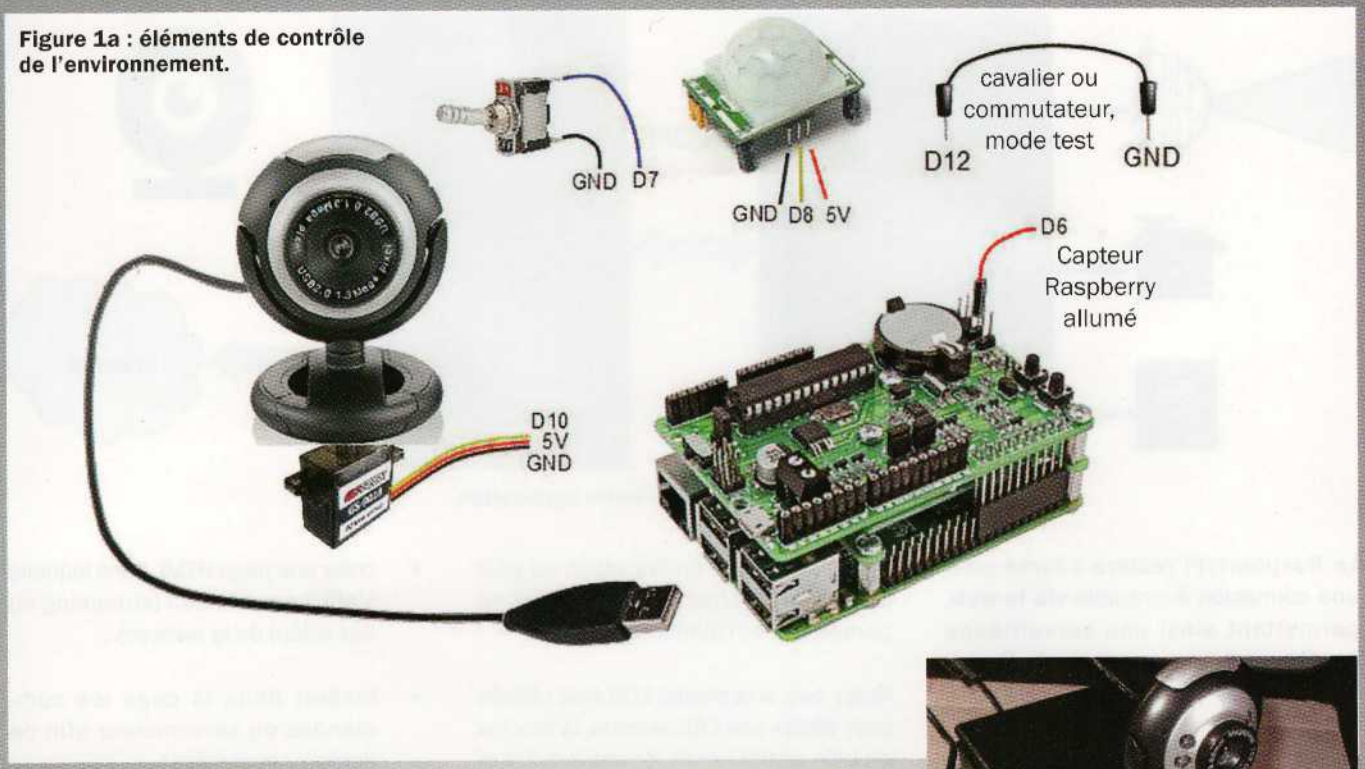


Figure 1b : webcam équipée d'un servomoteur pour la rotation et d'un PIR pour la détection de mouvement (intrusion).

installons sur le RaspberryPi à l'aide de la commande suivante :

```
sudo apt-get install Motion
```

Mais d'abord, il est préférable de mettre à jour le système en utilisant la commande suivante : `sudo apt-get update`

Ce logiciel possède également une gestion intégrée de la détection de mouvement.

Cependant, pour notre application, nous utiliserons un capteur de mouvement PIR. La raison de ce choix réside dans le fait que nous allons utiliser la possibilité offerte par RandA, qui est d'allumer le RaspberryPi, afin de ne conserver qu'Arduino, car ce dernier consomme très peu de courant.

Pour vous donner une idée de la configuration, le tableau 1 récapitule quelques données de consommation.

Vous remarquerez qu'avec Arduino à l'arrêt, la consommation est réduite à seulement 7 mA ou 12 mA en fonctionnement (contrôle de l'environnement).

Nous pouvons ensuite connecter le capteur PIR à la broche D3, qui correspond à l'interruption externe numéro 1, afin de réactiver Arduino, qui à son tour active le RaspberryPi. Mais, pour ne pas trop compliquer le sketch, dans cette application nous nous contenterons d'attendre une « intrusion », la consommation sera d'environ 28 mA.

En résumé, **en présence d'un signal du capteur PIR** (détection d'un mouvement), **Arduino activera le RaspberryPi**, qui ordonnera de **prendre une photo**, puis de l'acquérir et de **l'envoyer par e-mail**.

Tableau 1 : consommation

Arduino Uno (alimenté par USB avec une fréquence de fonctionnement de 16 MHz)	Fonctionnement normal (consommation due à l'ATmega328P et aux circuits d'interface USB).	environ **48mA** (sans connexions sur les ports).
	Fonctionnement en mode Sleep (Power Down Mode) activé avec une interruption externe (broche 2 ou 3). Les circuits de l'interface USB restent actifs. L'ATmega consomme en réalité quelques microampères.	environ **34mA** (presque complètement à cause de la carte).
RandA (avec RaspberryPi et l'ATmega328P cadencé à 16 MHz)	Fonctionnement normal Arduino + RaspberryPi (Arduino ne comprend que l'ATmega328P, car les circuits d'interface USB ne sont pas présents).	environ **330mA**.
	Fonctionnement avec le RaspberryPi éteint. (seulement l'Atmega328P mais avec le PIR et le servomoteur actifs).	environ **28mA**. (23 mA sans servomoteur)
	Fonctionnement avec le RaspberryPi éteint et Arduino en mode Sleep (Power Down Mode) activé avec interruption externe. Cette fois, l'ATmega consomme quelques milliampères, même en mode veille, en raison des ports connectés au RaspberryPi.	< **7mA** 3,2mA (ATmega) 3,7mA (carte seule) 12 mA (avec le PIR et le servomoteur connectés)

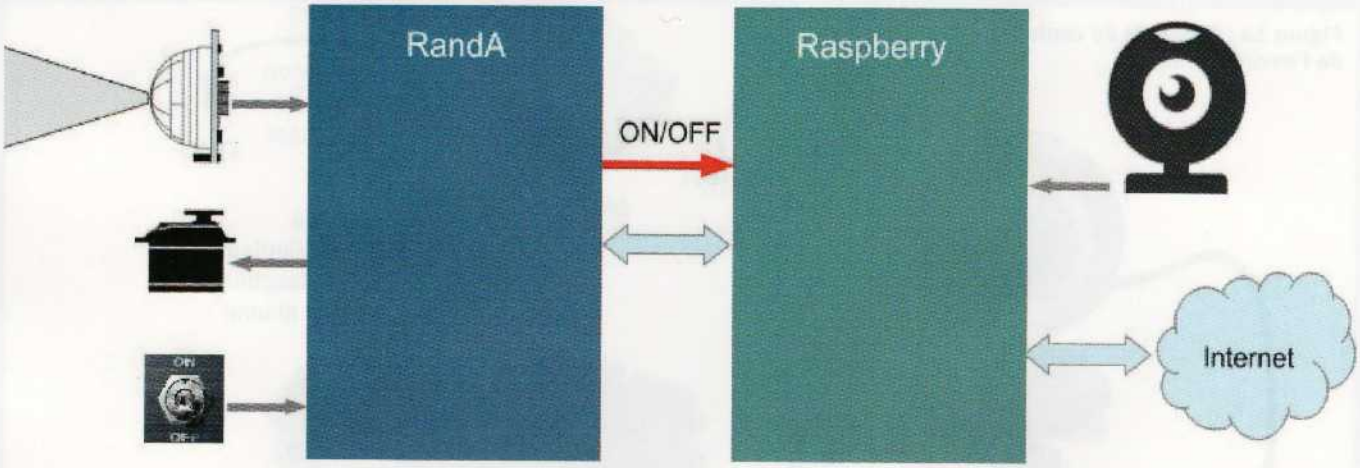


Figure 2 : schéma général de notre application.

Le RaspberryPi restera allumé pour une connexion éventuelle **via le web**, permettant ainsi une **surveillance continue et panoramique de l'environnement**.

Le contrôle de l'environnement sera activé ou désactivé à l'aide d'un commutateur (ou relais) connecté à Arduino (par exemple sur la broche D7).

Cela permet de mettre en route ou d'arrêter la surveillance d'une pièce ou d'un local.

De plus, la broche D6 sera utilisée pour vérifier si le RaspberryPi est activé ou non.

Cette broche est en fait reliée au pôle positif de la broche LD3, qui est en parallèle avec l'alimentation.

Notez que, si la broche LD3 était utilisée pour piloter une LED externe, la tension au pôle positif serait réduite à 1,2 V et son état devrait donc être lu avec une entrée analogique (par exemple A0).

Une fois que la partie mécanique a été définie et que les connexions ont été établies, il est maintenant temps de passer au logiciel.

Nous devrons en particulier :

- configurer correctement le programme « Motion » ;
- créer une page HTML dans laquelle s'affichera la vidéo (streaming ou flux vidéo) de la webcam ;
- insérer dans la page les commandes du servomoteur afin de déplacer la caméra ;
- créer des scripts CGI pour contrôler Arduino ;
- créer un sketch Arduino pour détecter la signalisation du PIR, pour activer le RaspberryPi et piloter le servomoteur en fonction des commandes reçues du RaspberryPi sur le port série. Le sketch doit également détecter la commutation de l'interrupteur (ou relais) marche/arrêt.

Tableau 2

Variable	Valeur d'origine	Configuration pour la détection de mouvement	Configuration pour l'affichage du navigateur	Description
daemon	off	on	on	Partie en arrière plan.
process_id_file	/var/run/motion/motion.pid	/tmp/motion.pid	/tmp/motion.pid	Fichier contenant le PID du processus. Utile pour le fermer.
framerate	2	24	24	Images par seconde pour une séquence de photos et de films.
output_normal	on	first	off	Prend et enregistre des photos lorsqu'il détecte un mouvement : OFF = désactivé, first = enregistre la première photo prise.
gap	60	10	60	Nombre de secondes pour déclarer l'événement fermé.
ffmpeg_cap_new	on	off	off	Films en mpeg (désactivé).
Webcam_maxrate	1	24	24	Images par seconde pour la webcam.
Webcam_localhost	on	off	off	Seulement la vision locale (non).
control_port	8080	0	0	Contrôle du programme à distance (désactivé).
on_picture_save	*non actif (commenté)*	/home/pi/sendPicture.sh %f	*non actif*	Script activé lorsqu'une photo est prise. La variable "% f" contient le nom du fichier.

Le programme « Motion »

Le logiciel « Motion » est un outil puissant (et gratuit) de gestion de flux vidéos visant à détecter les mouvements provenant d'une webcam ou d'une caméra. Une documentation complète (en anglais) est disponible sur le site : https://motion-project.github.io/motion_guide.html.

En réalité, « Motion » intègre de nombreuses autres fonctionnalités. Tout d'abord, il inclut un serveur « http » dédié au streaming de la webcam. Il peut aussi piloter les fonctionnalités vidéo à distance, via un port « http » supplémentaire (fonctionnalité que nous n'utiliserons pas).

L'activité principale de notre projet consiste à détecter un mouvement en fonction de la modification d'un certain nombre de pixels pour une durée donnée.

Tous les paramètres, c'est-à-dire le nombre de pixels nécessaires pour déclencher l'événement, la marge d'erreur, les éventuelles zones de la trame à contrôler, etc., peuvent être modifiés en intervenant sur le fichier de configuration appelé « motion.conf ».

Si l'événement se produit, il peut prendre une (ou plusieurs) photos ou enregistrer une vidéo d'une durée prédéterminée. De plus, dans le cas où un événement se produit, il peut lancer un script.

Dans l'application décrite ici, nous utiliserons cette fonction pour envoyer un e-mail avec une photo en pièce jointe.

Fondamentalement, lorsque notre capteur PIR détecte un mouvement, il demande à RandA d'allumer le RaspberryPi, ce dernier démarre alors le programme « Motion ». Après un certain délai inévitable dû au démarrage du RaspberryPi, « Motion » sera prêt à détecter de manière autonome l'activité dans la pièce sous surveillance et pourra ensuite prendre une photo et l'envoyer par e-mail.

Il est clair que, compte tenu de l'objectif de l'application, les dizaines de secondes nécessaires au démarrage du RaspberryPi n'ont aucune influence sur la capture de l'événement, qui prend certainement plus que quelques secondes.

À ce stade, le RaspberryPi reste allumé et est prêt à recevoir une éventuelle connexion « http » sur la page HTML préparée à cette fin. La **page permet l'affichage du flux vidéo** depuis le port du serveur « Motion » (port 8081 modifiable). La page permet également d'envoyer une commande de réinitialisation qui rétablit la situation de départ en désactivant le RaspberryPi.

Pour permettre le contrôle de la webcam, un mode « **test webcam** » a été configuré. Le RaspberryPi est immédiatement activé lorsque RandA détecte la commutation de l'interrupteur et reste activé sans détection de mouvement ni prise de vue. Ainsi, le serveur web est toujours disponible pour afficher et déplacer la webcam.

Le mode « test » est activé, au démarrage de RandA, lors de la connexion de la broche D12 à la masse (GND). Le fichier de configuration standard de « Motion » est copié au moment de l'installation dans le répertoire « /etc/motion ».

Ce fichier doit être modifié à la fois pour la détection du mouvement et pour la vision de la webcam. Nous allons donc en faire deux copies ; une que nous placerons dans le dossier « **/home/pi** » (dossier de l'utilisateur pi) et l'autre que nous placerons dans le **dossier contenant les scripts CGI**.

La première copie sera utilisée en cas de mise sous tension du RaspberryPi par une alarme et aura pour but de configurer « Motion » afin qu'il prenne une photo et qu'il active le script pour envoyer cette dernière. La seconde copie du fichier sera utilisée pour l'activation de la webcam sans prendre de photos.

Le fichier de configuration sera modifié selon les éléments décrits dans le tableau 2.

Comme visible dans le fichier de configuration de base, dans le cas de la détection d'un mouvement, une seule photo est prise et est placée dans le dossier « /tmp/motion ». Le nom du fichier contiendra une référence d'horodatage et est contenu dans la variable « %f ».

Parmi les différents types d'événements pouvant être utilisés, nous sélectionnerons celui qui s'active en prenant une photo et en passant le nom du fichier au script.

Notez qu'avant d'envoyer des emails avec le programme « SendMail », nous devons configurer le fichier « **Mail.properties** ». Le script, qui est déclenché par la prise d'une photo, s'appelle « /home/pi/sendPicture.sh » et est reporté dans le listing 1.

Listing 1 : sendPicture.sh

```
#!/bin/bash
# Nom de la photo à envoyer à $1 (paramètre passé par « Motion »)
/home/pi/bin/SendMail mailto=toto.lulu@aol.com subject="Alarm!" attach=$1
# Fait le travail et ferme « Motion »
sudo pkill motion
```

Listing 2 : à ajouter à la fin de « /etc/rc.local »

```
# si le fichier de démarrage personnalisé existe, il le lance en tant qu'utilisateur
pi
FSTARTUP="/home/pi/pistartup.sh"
if [ -x $FSTARTUP ];
then
  sudo -u pi /home/pi/pistartup.sh
fi
exit 0
```

Le programme « Motion » est lancé avec la commande : **motion -c nomdufichier deconfiguration**

Comme mentionné précédemment, le programme « Motion » sera lancé à la mise sous tension du RaspberryPi, qui, normalement, sera alimenté pour prendre la photo du mouvement détecté et l'envoyer par email.

Le système d'exploitation Linux lance au démarrage le script « /etc/rc.local », mais au lieu d'insérer le lancement de « Motion » à la fin de ce fichier, nous suggérons de modifier « rc.local » en ajoutant un lien à un script placé dans le dossier de l'utilisateur.

De cette manière, nous pouvons modifier le lancement de programmes ou de scripts sans devoir modifier des fichiers système et/ou sans avoir besoin d'autorisations « root ». Voir un exemple reporté au listing 2.

Dans le Listing 3 est reporté le script lancé au début.

Page HTML du serveur Web

Dans cette application, le serveur web n'est activé qu'à la suite d'une alarme ou d'une détection de mouvement (sauf en mode test). L'accès à cette page doit être autorisé uniquement à l'aide d'un mot de passe, afin d'éviter qu'une personne étrangère ait accès à la webcam et visualise ce qui se passe dans votre maison ou local.

Dans « Tomcat », pour activer la protection, vous devez intervenir sur le fichier contenu dans le dossier se trouvant dans l'application, c'est-à-dire dans : « /WEB-INF/web.xml ».

Isoler l'application ne consiste pas à utiliser le dossier de base « root » du serveur web, mais à créer un nouveau dossier dans « webapps », qui est le dossier général des applications web.

En fait, en plus de l'application de base présente dans « root », il existe d'autres répertoires contenant des applications web qui correspondent à une application spécifique (statiques, avec « servlet » ou un script CGI).

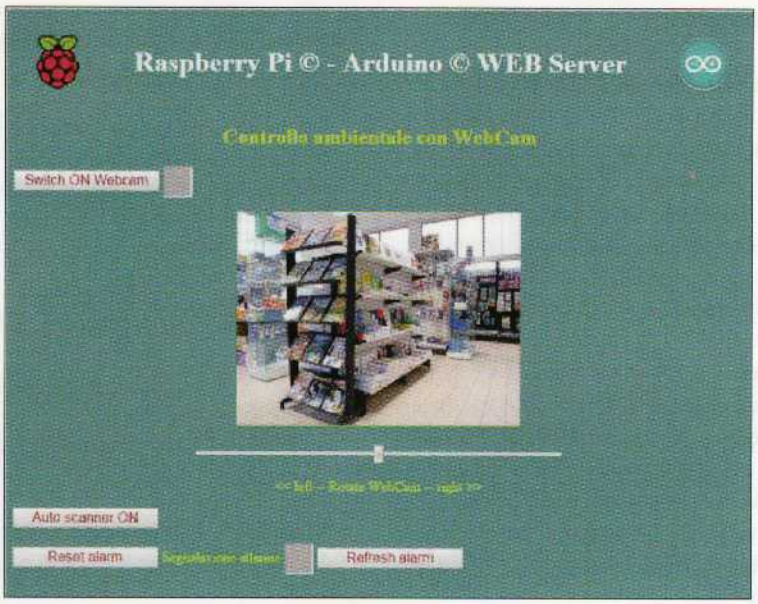


Figure 3 : la page HTML de la webcam.

Listing 3 : pistartup.sh

```
#!/bin/bash
#exit 0          # pour désactiver le script, supprimez le commentaire
ser=/dev/ttyS0
stty -F $ser 9600
sleep 5
echo "M" > $ser
echo "Commande M" > /home/pi/start.log
  # délai pour permettre à Arduino de répondre
sleep 1
  # lit la réponse d'Arduino
read -r -t 2 replay < $ser
if [ ${#replay} -lt 4 ]; then
 echo "Pas de réponse !" >> /home/pi/start.log
 exit 1
fi
replay=${replay:0:4}
echo $replay >> /home/pi/start.log
if [ $replay = "TEST" ]; then
 echo "No motion" >> /home/pi/start.log
 exit 0                  # si en mode test alors il s'éteint
fi
if [ $replay = "CTRL" ]; then
 echo "Start motion !" >> /home/pi/start.log
  # sinon il ferme tout processus Motion actif
sudo pkill motion
  # et active Motion avec le détecteur de mouvement et l'envoi de la photo
 motion -c /home/pi/motion-detect.conf
fi
```

Mais chaque dossier doit contenir les sous-dossiers « META-INF » et surtout « WEB-INF » avec un fichier « web.xml ».

Nous allons créer un dossier que nous nommons « Control_Environnement » (mais vous pouvez le nommer différemment) dans lequel nous créons les deux sous-dossiers « META-INF » et « WEB-INF ».

À la racine du répertoire « Control_Environnement » (évitez les accents pour les noms des fichiers), nous mettrons notre page HTML.

Listing 4 : Web.xml

```
<web-app xmlns="http://java.sun.com/xml/ns/javaee"
 xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
 xsi:schemaLocation="http://java.sun.com/xml/ns/javaee
            http://java.sun.com/xml/ns/javaee/web-app_3_0.xsd"
 version="3.0"
 metadata-complete="true">

<description> Contrôle de l'environnement avec mot de passe </description>
<display-name> Control_Environnement</display-name>

<!-- Définit le nom d'une règle de sécurité -->
<security-role>
 <description> Control_Environnement</description>
 <role-name>Control</role-name>
</security-role>

<!– La règle de sécurité est appliquée à l'URL suivante –>
<!– La méthode de sécurité appliquée est le nom d'utilisateur/mot de passe de
base -->
 <security-constraint>
  <web-resource-collection>
        <web-resource-name> Control_Environnement</web-resource-name>
          <url-pattern>/*</url-pattern>
  </web-resource-collection>
  <auth-constraint>
          <role-name>Control</role-name>
  </auth-constraint>
 </security-constraint>

 <login-config>
        <auth-method>BASIC</auth-method>
  <realm-name> Control_Environnement </realm-name>
 </login-config>
<!– -->
</web-app>
```

Raspberry

Au démarrage, le script « pistartup.sh» est activé. Le script demande à Arduino s'il est en mode test ou normal. S'il est en mode test il se termine. S'il est en mode normal (alarme), il lance « action », qu'il fermera après l'opération.

Figure 4 : activité du RaspberryPi.

Raspberry

Le serveur Web est actif. Si vous appelez la page WebCam.html, vous pouvez lancer « action » en mode webcam et déplacer ou contrôler/réinitialiser le flag d'alarme (via les scripts présents dans le dossier Control_Environnement)

Dans « WEB-INF », nous créerons un fichier « web.xml », semblable à celui reporté au listing 4.

À ce stade, nous devons créer une autorisation pour l'application, en intervenant sur le fichier : « /home/apache-tomcat-7.0.47/conf/tomcat-users.xml », c'est-à-dire en lui ajoutant la ligne suivante : **<user username="toto" password="lulu" roles="Control"/>**

Evidemment, le nom d'utilisateur et le mot de passe ont été choisis au hasard et vous devez les remplacer par ceux que vous souhaitez.

Enfin, nous créons le dossier « cgi » dans le répertoire « WEB-INF » qui contiendra les scripts « bash » de l'application. La page HTML (dénommée par exemple « WebCam.html ») doit être placée dans le dossier de l'application « Control_Environnement », puis référencée comme suit : « http:/...../ Control_Environnement /WebCam.html ».

Pour l'accès, une autorisation sera nécessaire via le nom d'utilisateur et le mot de passe que nous avons spécifié. La page doit être utilisée avec un navigateur compatible HTML5 en raison du « slide » (diapositive) et du mode simplifié permettant d'afficher le flux vidéo de la webcam.

Nous suggérons d'utiliser les navigateurs Firefox ou Chrome, qui présentent des caractéristiques graphiques compatibles avec les nouveaux éléments du langage HTML5, par rapport à Internet Explorer.

La page contient essentiellement :

- la fenêtre où le flux vidéo de la webcam est affiché ;
- un curseur pour déplacer la webcam ;
- trois boutons, un pour activer la réception de la webcam, un pour activer le mode de balayage automatique (balayage en continu, comme le mouvement d'un essuie-glace), plus un bouton pour réinitialiser l'alarme ;
- un bouton d'actualisation du signal d'alarme utile en mode test.

La trame du flux vidéo est un simple champ image dont la source fait référence à un autre serveur web doté du port 8081.

Listing 5 : StartWCam.sh

```
#!/bin/bash
# Détection des paramètres passés par l'appel http
# Ils sont présents dans la variable d'environnement QUERY_STRING
# au format nom1=val1&nom2=val2 ...

qstring=$QUERY_STRING
# Remplace & par un espace pour isoler les paires par=val
qstring=${qstring//&/ }
# Détecte les paires par=val et les place dans un tableau appelé par
read -r -a par <<< "$qstring"
ang=-1

# Cycle pour chaque élément p du tableau par
for p in ${par[@]}; do
pn=${p%=*}                # pn contient le nom
pv=${p#*=}                # pv contient la valeur

# analyse (parsing) de la commande
if [ $pn = "wcam" ]; then
fstart=$pv
fi

done    # fin de cycle

# action

mconfig="/home/apache-tomcat-7.0.47/webapps/ Control_Environnement/WEB-INF/cgi/motion.conf"
if [ $fstart = "OK" ]; then
motion -c $mconfig        # lance Motion et répond à l'appel http
echo "Content-type: text/html"
echo ""
echo "OK"
else
pid=$(< /tmp/motion.pid)  # sinon il ferme Motion et répond
sudo kill $pid
echo "Content-type: text/html"
echo ""
echo "NOK"
fi
```

En fait, le logiciel « Motion » peut également créer son propre serveur web pour le routage du flux :

```
<div align="center">
<img id="vid" src="http://:8081/"
width="320" height="240" style="-
background:#999; border:#CF3 groo-
ve"/> 
</div>
```

Les boutons sont contrôlés par des fonctions Javascript. La page HTML, avec tous les logiciels de cette application, est contenue dans l'archive « Control_Environnement.zip » en téléchargement dans le sommaire détaillé de la revue.

Listing 6 : RandAcmd.sh

```
#!/bin/bash
ser=/dev/ttyS0

# Détection des paramètres passés par l'appel http
# Ils sont présents dans la variable d'environnement QUERY_STRING
# au format nom1=val1&nom2=val2 ...

qstring=$QUERY_STRING
# Remplace & par un espace pour isoler les paires par=val
qstring=${qstring//&/ }
# Détecte les paires par=val et les place dans un tableau appelé par
read -r -a par <<< "$qstring"

# Cycle pour chaque élément p du tableau par
for p in ${par[@]}; do
pn=${p%=*}                # pn contient le nom
pv=${p#*=}                # pv contient la valeur
```

```
 # analyse (parsing) de la commande et l'envoie à Arduino
case $pn in
    QM)
      echo "M" > $ser
      ;;
    QA)
      echo "Q" > $ser
      ;;
    AN)
      echo "A"$pv > $ser
      ;;
    SC)
      echo "S"$pv > $ser
      ;;
    RA)
      echo "R" > $ser
      ;;
esac
done # fin du cycle

sleep 0.2   # Retard pour donner le temps à Arduino de répondre
 # Lit la réponse d'Arduino
read -r -t 2 replay < $ser

 # Il la répète en réponse à l'appel http
 # mais d'abord l'en-tête de la réponse http doit être fermée
 # avec une ligne vide.
 # La sortie est automatiquement lue par le serveur Tomcat.
echo "Content-type: text/html"
echo ""
echo $replay   # réponse http (à la fonction Javascript)
```

Fonctions Javascript pour contrôler la Webcam

Les fonctions s'exécutent sous AJAX, puis dialoguent avec le serveur web en arrière-plan. Elles lancent essentiellement des scripts « bash » en transmettant des paramètres (mode CGI).

Pour l'activation de la webcam, un script « bash » s'exécutant sur le RaspberryPi suffit, mais pour le déplacement de la caméra, le script « bash » s'exécutant sur le RaspberryPi devra envoyer en plus une commande à RandA via le port série.

Le curseur du mouvement de la webcam (en dessous de la fenêtre vidéo sur la page web) appelle la fonction correspondante que lorsque le bouton de la souris est relâché. La position de la webcam n'est pas mise à jour continuellement.

Scripts CGI activés par les fonctions Javascript

Pour cette application, il a été décidé d'utiliser le mode CGI pour ne pas

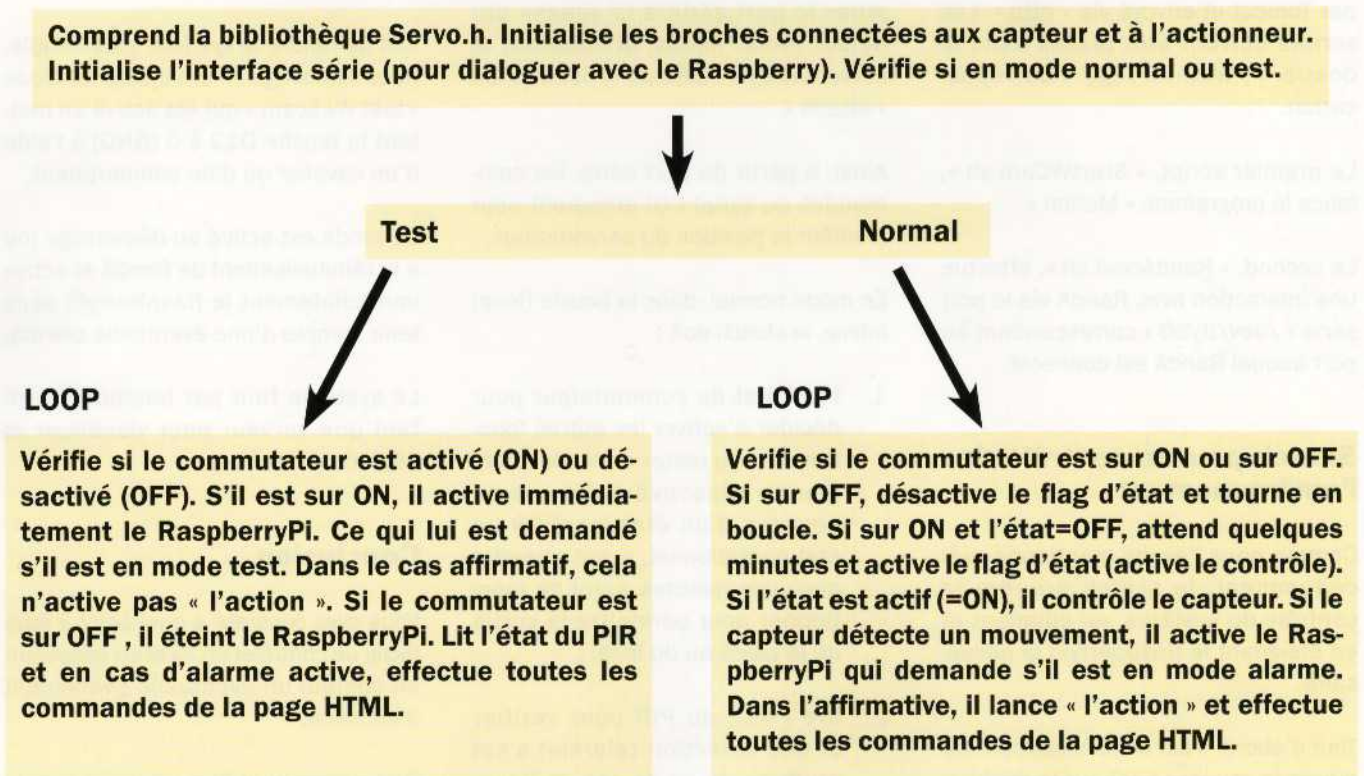


Figure 5 : organigramme simplifié du sketch pour le contrôle de l'environnement (local ou autre).

être obligé de construire une application avec des éléments Java et Servlet appartenant au domaine des applications web.

Ceux qui pratiquent le « Java Web Application » peuvent facilement basculer vers la mise en œuvre du dialogue AJAX via Servlet ou JSP.

Les scripts sont conceptuellement divisés en deux parties : dans la première, s'effectue la détection des paramètres passés par la fonction Javascript, tandis que dans la seconde, l'action réelle est exécutée.

Les paramètres sont inclus dans la variable d'environnement « QUERY_STRING » que le système d'exploitation transmet au script (tâche exécutée par Tomcat).

Dans la pratique, il est nécessaire d'analyser la chaîne pour détecter les paires « nom = valeur » séparées par le caractère « & ».

Une fois la fonction terminée, il est également possible d'y répondre, par exemple, via un "echo ....".

En fait, le flux de sortie est intercepté par Tomcat et envoyé via « http ». Les scripts doivent être placés dans le dossier « /WEB-INF/cgi/ » de l'application.

Le premier script, « StartWCam.sh », lance le programme « Motion ».

Le second, « RandAcmd.sh », effectue une interaction avec RandA via le port série « /dev/ttyS0 » correspondant au port auquel RandA est connecté.

Sketch pour le contrôle de l'environnement

Comme nous l'avons mentionné précédemment, le sketch prendra le contrôle du système, en allumant et en éteignant le RaspberryPi si nécessaire.

Tout d'abord, il est donc nécessaire de placer le cavalier SW2 sur la position « Arduino toujours alimenté » (faites-le avec le système hors tension).

Les deux autres cavaliers JP1 et JP2 sont fermés, conformément à la configuration standard.

Au démarrage, le sketch doit configurer la broche D7 en entrée avec la résistance de pull-up activée (mode « INPUT_PULLUP »), afin qu'elle puisse vérifier la position du commutateur.

La broche D8 doit être configurée en entrée (mode « INPUT ») pour recevoir le signal du PIR.

Enfin, la broche D10 doit être configurée en sortie (mode « OUTPUT ») pour contrôler le servomoteur à l'aide de la librairie « Servo.h » qui doit être incluse dans le sketch.

Le sketch doit également configurer en entrée la broche D6 (mode « INPUT ») afin de vérifier que le RaspberryPi est activé ou non.

De même, la broche D12 doit être configurée en entrée avec la résistance de pull-up activée (mode « INPUT_PULLUP ») afin de vérifier si le mode actuel est le mode « TEST » ou le mode « NORMAL ».

Pour terminer, le sketch doit configurer le port série à la vitesse par défaut (9600 bauds) et initialiser le servomoteur à l'aide de la commande « attach ».

Ainsi, à partir du port série, les commandes du script CGI arriveront pour modifier la position du servomoteur.

En mode normal, dans la boucle (loop) infinie, le sketch doit :

1. lire l'état du commutateur pour décider d'activer les autres fonctions ou de rester en veille (commande désactivée). En cas du passage d'un état inactif à un état opérationnel, il doit attendre quelques minutes avant de fonctionner pour permettre la sortie de la pièce ou du local ;
2. lire l'état du PIR pour vérifier si une détection (alarme) s'est produite. Dans ce cas, le RaspberryPi doit être allumé et, dès qu'il est prêt, prendre une photo et la joindre au courrier qui sera envoyé, puis le laisser allumer jusqu'à ce qu'une commande de veille (stand-by) soit envoyée à partir de la page HTML.

Comme il s'agit d'un exemple de démonstration, nous avons utilisé un simple commutateur pour activer ou désactiver le contrôle de l'environnement.

Dans une application réelle, nous devons camoufler le commutateur ou utiliser un clavier sécurisé doté d'une sortie relais, dans lequel nous tapons un code d'activation et de désactivation.

Nous pouvons également gérer l'activation/désactivation à distance à l'aide d'un système de télécommande, de manière à désactiver le système avant d'entrer dans le local et l'activer après l'avoir quitté.

La figure 5 montre le schéma de principe du sketch, nous ne publions pas l'intégralité du code source pour des raisons d'espace, mais vous pouvez le trouver dans l'archive « Control_Environnement.zip », le fichier se nomme « CtrPirWCam.ino ».

Afin de rendre le système plus flexible, nous avons également ajouté un mode « test Webcam » qui est activé en mettant la broche D12 à 0 (GND) à l'aide d'un cavalier ou d'un commutateur.

Ce mode est activé au démarrage (ou à la réinitialisation) de RandA et active immédiatement le RaspberryPi sans tenir compte d'une éventuelle alarme.

Le système finit par fonctionner en tant que serveur pour visualiser et déplacer la webcam.

Conclusion

Vous avez constaté à quel point il était facile de contrôler un local en observant en continu un éventuelle évènement d'intrusion.

Ce système peut être complété par un modem GSM/HSDPA pour l'envoi de SMS. ■

UN SAPIN DE NOËL ÉLECTRONIQUE

Pour ces fêtes de fin d'année, nous vous proposons une animation lumineuse ludique, sans microcontrôleur, mais avec une électronique traditionnelle. Ce montage permettra de décorer votre sapin ou votre crèche en recréant un jeu de lumière semblable aux boules lumineuses d'un sapin de Noël. Vous pourrez le disposer ou le suspendre où vous voulez dans votre maison, car il fonctionne sur pile.

de Davide Scullino

Depuis de nombreuses années, nous avons décrit toutes sortes de montages pour les décorations de Noël et en particulier pour le sapin. Des statiques, des animés, des lumineux, en forme de sphère, de cloche, d'étoile, de paquet cadeau, et bien plus encore.

Le montage que nous vous proposons dans cet article est encore plus original, car même s'il possède des lumières qui s'illuminent, sa forme ressemble à un petit sapin de Noël à suspendre. De plus, il permet aux plus jeunes de s'initier à l'électronique, car le montage est très simple et du plus bel effet si vous en réalisez plusieurs en les disposant de manière judicieuse.

Il s'agit donc d'un sapin de Noël miniature avec des lumières intégrées, qui peut être suspendu aux branches d'un vrai sapin de Noël, mais qui peut également servir d'élément décoratif pouvant être placé n'importe où car il est petit, léger et fonctionne avec une seule pile.

Notre petit montage dispose d'un circuit imprimé en forme de sapin constitué d'une base avec, à ses extrémités, 8 LED, dont la moitié sont de couleur rouge et l'autre moitié de couleur verte. L'ensemble est alimenté par une pile bouton de 3 V de type CR2032. Une fois l'alimentation établie, c'est-à-dire en fermant le cavalier approprié à l'arrière, les LED se mettent à

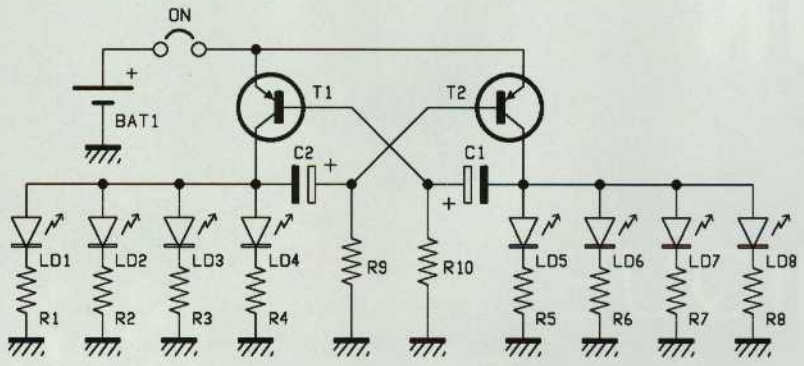


Schéma électrique du sapin de Noël.

clignoter en alternance, s'allumant de manière à composer une animation en « zigzag ».

Mais sans perdre plus de temps, examinons le schéma électrique, qui, comme vous pouvez le constater, est vraiment très simple.

Schéma électrique

Comme l'électronique nous l'enseigne, pour allumer deux LED ou des groupes de LED, un générateur de signaux rectangulaires en opposition de phase suffit.

Dans sa forme la plus simple, le circuit est appelé **multivibrateur astable**.

Le multivibrateur est précisément le cœur de notre projet, et à lui seul, il permet d'allumer les 8 LED. Une alimentation basse tension continue lui suffit.

Le montage est alimenté par une pile bouton via le cavalier « ON », ce dernier permet d'activer et de désactiver l'alimentation afin de mettre en fonction la décoration lumineuse.

Le **multivibrateur astable** est la partie du circuit constituée par les **deux transistors bipolaires T1** et **T2**, de type **PNP**, ainsi que par les résistances R9 et R10 et les condensateurs C1 et C2.

Les dipôles, constitués de chaque côté par les LED avec leurs résistances en série, constituent les charges des collecteurs des deux transistors.

Ces derniers sont donc simultanément les éléments de commutation du circuit astable et les pilotes (drivers) des LED. Cela permet de rendre le montage vraiment simple.

Notre circuit, qui est de type astable, est un **générateur de signal rectangulaire**. Le circuit ne peut pas maintenir un état stable (d'où son nom « astable ») à cause du fait que les transistors commutent continuellement et alternativement. Cela implique que les transistors T1 et T2 ont toujours des niveaux de tension opposés sur leurs collecteurs respectifs.

Ce phénomène commence dès que le circuit est alimenté. Initialement, les deux condensateurs sont déchargés, ainsi le transistor dont la tension de seuil est la plus basse entre en conduction et bloque l'autre, jusqu'à ce que le condensateur, qui est relié à la base du transistor bloqué, soit complètement chargé.

Le circuit fonctionne parce qu'il existe des différences entre les caractéristiques des composants et en particulier entre les transistors, car même s'ils sont du même type, ce ne sont pas des transistors idéaux. Il existe d'un transistor à l'autre des différences de gain pouvant varier du simple au double, paradoxalement si ces différences n'existaient pas le circuit ne fonctionnerait pas.

Ainsi, la dispersion des caractéristiques des différents paramètres des transistors appartenant à un même lot de production permet de s'assurer qu'il existe des différences entre les tensions de seuil et le gain h_{FE} de chaque transistor. Les tolérances des résistances R9 et R10 et des condensateurs C1 et C2 y contribuent également.

Pour comprendre le fonctionnement du circuit astable, supposons que dès que le montage est alimenté, T1 commence à conduire. Son collecteur passe alors à un niveau haut (alimentation positive) et court-circuite le positif du condensateur C2. Ce dernier, initialement déchargé, porte la base de T2 à un niveau haut qui maintient T2 bloqué.

À ce stade, C2 commence à se charger à travers R9 avec une polarité positive sur le « - », jusqu'à ce que la base de T2 soit à un potentiel négatif et qu'il entre en saturation (conduction).

Le collecteur de T2 prend alors un niveau haut et décharge C1, portant la base de T1 à un niveau haut et donc le transistor T1 se bloque, ce qui implique que son collecteur arrête de faire circuler le courant.

Dès lors, C2 commence à se charger avec une polarité positive sur la broche « + ». À un moment donné, il porte la base de T1 vers un potentiel qui provoque de nouveau son blocage, de sorte que son collecteur revienne à 0 V et décharge immédiatement C1. Cela porte la base de T1 à un niveau bas, ce dernier devenant de nouveau conducteur.

Le cycle est répété tant que le montage est alimenté, le circuit génère deux ondes rectangulaires sur T1 et T2 en opposition de phase.

Ces tensions sont suffisantes pour polariser et donc allumer les deux groupes de LED situés entre les collecteurs et la masse, et qui sont chacun formés par 4 LED de couleur verte et rouge. Cette disposition particulière sur le circuit imprimé permet d'allumer en alternance les LED vertes d'un côté et les LED rouges du côté opposé et inversement.

Les résistances sont calculées pour allumer les LED rouges avec un courant qui représente un compromis entre la durée de vie de la pile et la luminosité.

Le courant est légèrement supérieur à 5 mA pour les LED rouges et un peu moins pour les vertes.

Les résistances ont toutes une valeur de 180 Ω, cependant les chutes de tension des LED sont différentes. Pour les rouges, la chute de tension est de 1,8 V (typique) contre 2,1 V (typique) pour les vertes.

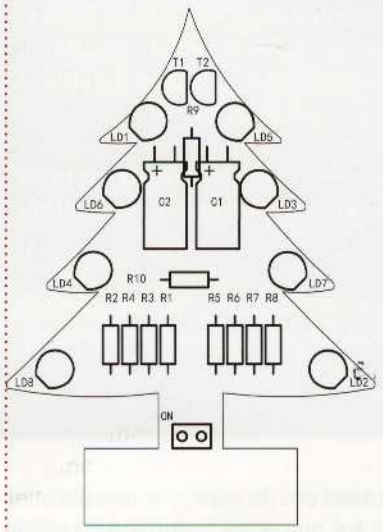
Réalisation pratique

Nous venons d'aborder le fonctionnement du circuit, passons maintenant à sa construction. Le circuit est facilement reproductible, même pour les débutants.

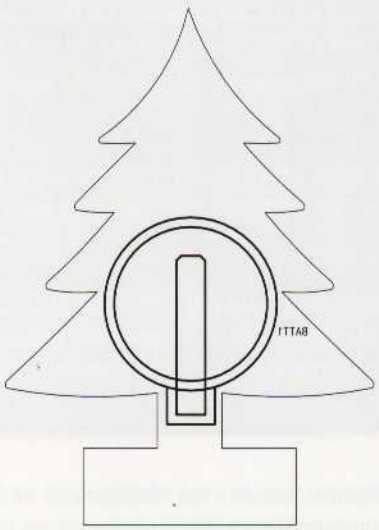
En fait, mis à part le circuit imprimé, qui est un circuit double face, mais qui se fabrique sans grande difficulté par photogravure, tous les composants sont de type traversant, faciles à trouver dans le commerce et à souder.

Pour la préparation du circuit imprimé double face, après avoir pris une plaque de cuivre présensibilisée de dimensions égales à la hauteur et à la largeur maximales du sapin, imprimez

Plan de montage du sapin de Noël



Plan d'implantation des composants du sapin de Noël.



Implantation du support de la pile CR2032 sur la face inférieure du circuit (face bottom).

Vue de la face supérieure (face top) avec les composants soudés.

Vue de la face inférieure, vous apercevez le support

Liste des composants

R1.....180 Ω
R2.....180 Ω
R3.....180 Ω
R4.....180 Ω
R5.....180 Ω
R6.....180 Ω
R7.....180 Ω
R8.....180 Ω
R9.....47 kΩ
R10....47 kΩ

C1.....22 µF/16 V électrolytique
C2.....22 µF/16 V électrolytique

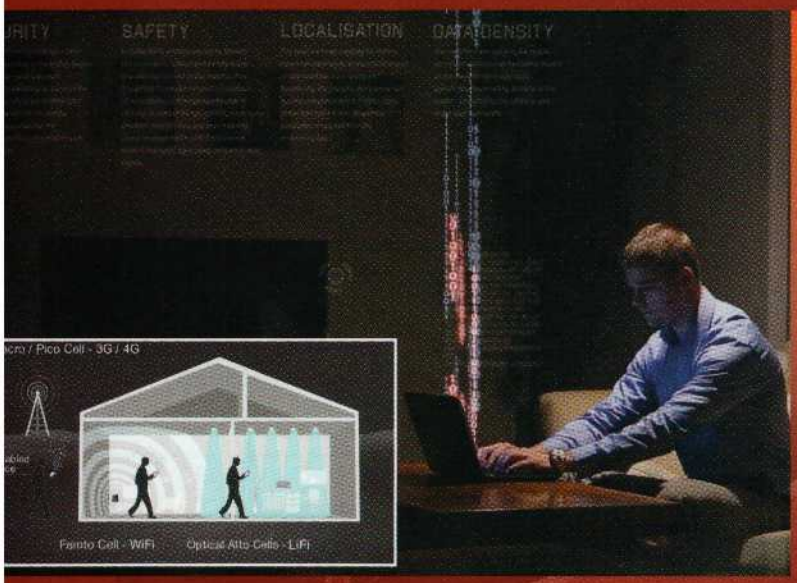
T1 .....BC557
T2 .....BC557

LD1....LED 3 mm verte
LD2....LED 3 mm rouge
LD3....LED 3 mm rouge
LD4....LED 3 mm verte
LD5....LED 3 mm verte
LD6....LED 3 mm rouge
LD7....LED 3 mm verte
LD8....LED 3 mm rouge

Divers

Barrette mâle 2 pôles
Cavalier
Support de pile CR2032
Pile CR2032

NB : les typons des circuits imprimés à l'échelle 1 sont disponibles en téléchargement dans le sommaire détaillé de la revue.



Les LED, au delà de l'éclairage

Non seulement l'éclairage à LED est devenu la principale source de lumière artificielle en intérieur, mais depuis quelques années il joue un rôle primordial dans les systèmes innovants de transmission de données. Il s'agit de réseaux optiques sans fil, c'est-à-dire des réseaux Wi-Fi qui utilisent un éclairage à LED, ils sont appelés Li-Fi (Light Fidelity).

La technique sous-jacente est assez simple à expliquer, mais un peu plus difficile à mettre en œuvre. Les LED fonctionnent à une valeur de courant modulable par des impulsions numériques et donc par les données à transmettre. La fréquence de modulation peut être assez élevée, de manière à obtenir des valeurs intéressantes de vitesses de transfert. Notre œil ne voit qu'une lumière continue, mais en analysant la lumière avec des photodétecteurs (tels que des photodiodes rapides), il est possible d'extraire la composante modulée.

L'utilisation de filtres pour la lumière ambiante et de techniques de séparation de la modulation, permet de recevoir des données à des distances intéressantes et dans des environnements relativement grands, à l'aide de dispositifs tels que le « LiFi-XC » (www.purelifi.com).

Les LED les plus prometteuses sont d'un nouveau type appelé « LED FCD ». Lors de tests réalisés, la technologie Li-Fi a permis d'atteindre un taux de transfert de 50 Go/s. La vitesse théorique maximale étant cependant de 200 Go/s, en exploitant une lumière dont la longueur d'onde est comprise entre 750 nm et 375 nm.

chaque face sur un film transparent (pour imprimante jet d'encre), puis développez la plaque de cuivre présensibilisée à l'aide d'une insoleuse.

Gravez ensuite la plaque à l'aide de perchlorure de fer, puis réalisez les perçages des pattes des composants. Le circuit imprimé doit être découpé en forme de sapin, en enlevant éventuellement les arêtes à l'aide d'une lime fine de façon à obtenir une forme épurée.

Pour l'assemblage des composants, il n'y a pas de suggestion particulière, si ce n'est de commencer par souder les vias, c'est-à-dire les interconnexions entre les deux couches de cuivre, dont une partie sera réalisée par les pattes des composants qui sont communes aux deux faces.

Les autres sont obtenues en soudant de chaque côté des morceaux de pattes de composants dans les trous correspondants, de façon à obtenir une continuité électrique entre les deux faces.

Ensuite, soudez les résistances et les condensateurs électrolytiques en respectant leur polarité (le « - » est indiqué sur le boîtier et la patte la plus longue est le « + »). Vous devez plier leurs pattes (sans les casser) à 90° afin de les coller sur la surface du circuit pour limiter la hauteur.

Soudez ensuite les LED vertes et rouges en les alternant de chaque côté du sapin.

Faites attention à leur polarité, la patte la plus longue correspond à l'anode et la partie biseautée du boîtier correspond à la cathode. Si vous inversez le sens, elles ne s'allumeront pas.

Terminez en soudant les deux transistors, vérifiez leur orientation. Leur méplat doit être orienté vers la droite. N'oubliez-pas le cavalier et sur l'autre face le support de pile.

Suivez le plan de montage visible dans ces pages pour une orientation correcte des composants polarisés.

N'oubliez pas de maintenir une distance de 3 à 4 mm entre la surface du circuit imprimé et le corps des diodes, afin qu'elles dépassent et qu'il soit facile de masquer le reste des composants avec un morceau de carton découpé en forme de sapin. Pour cela, placez les transistors aussi bas que possible et utilisez des condensateurs de 16 V pour un encombrement réduit.

Une fois l'assemblage terminé, insérez une pile CR2032 dans le support et faites passer un morceau de fil à coudre ou de pêche dans le trou situé en haut du circuit imprimé pour former un anneau qui servira à le suspendre.

Une fois en place, notre gadget est prêt à fonctionner car il ne nécessite pas de réglages particuliers. Pour l'allumer, insérez simplement le cavalier, il permet à la tension de la pile d'atteindre l'électronique.

Il ne nous reste plus qu'à vous laisser préparer le sapin et à vous souhaiter de joyeuses fêtes de fin d'année ! ■

Contrôle à distance
Utiliser son téléphone GSM
comme télécommande et récepteur d'alarmes

Installations antivol pour les bâtiments civils et industriels, pour les voitures, contrôle des systèmes de climatisation, chauffage, contrôle de pompes et de systèmes d'irrigation, contrôle d'installations industrielles etc.

Prix TTC - Photos non contractuelles, publicité valable pour les mois de parution. Prix exprimés en euros TTC. sauf erreurs typographiques ou omissions

Réf. : TDG133 **79,00 €**
Télécommande bidirectionnelle GSM
2 entrées/2 sorties relais.
Coffret Réf. : BOXTDG **15,90 €**

Permet de piloter deux relais à distance (en mode monostable ou bistable) grâce à des messages SMS (sécurisés par mot de passe) envoyés depuis votre téléphone portable. Mémorisez jusqu'à huit numéros de téléphone d'appel d'alarme déclenchables via les 2 entrées. Convient également comme récepteur de système d'accès piloté par 200 numéros de téléphone. Boîtier en option. Requiert une carte SIM (non incluse).

Réf. : TDG134 **76,00 €**
Télécommande GSM
1 sortie relais.
Coffret Réf. : BOXTDG134 **15,55 €**

Système à utiliser en association avec le système électrique d'ouverture du portail. La fermeture peut être activée en envoyant un appel avec le téléphone portable via la carte Sim insérée sur la platine GSM. L'appel ne coûte rien. Le dispositif enverra une commande à la centrale de contrôle du portail qui procèdera à l'ouverture ou la fermeture. Gestion à distance des utilisateurs via SMS (demande de mot de passe d'authentification) ou via le PC local (avec l'interface FT782M en option) avec le logiciel approprié. Alimentation de 9 à 32 Vdc. Le système comprend: le dispositif monté et testé avec le module GSM/GPRS SIM900 et l'antenne. Boîtier en option. Requiert une carte SIM (non incluse).

Réf. : TDG140 **84,90 €**
Télécommande GSM bidirectionnelle
2 entrées/2 sorties relais.
Coffret Réf. : BOXTDG **15,90 €**

Avec commandes par DTMF ou SMS. Système de contrôle à distance bidirectionnel qui utilise le réseau GSM pour activer et contrôler. Possibilité de mémoriser 8 numeros pour l'envoi des alarmes déclenchables via les 2 entrées. Alimentation comprise entre 9 et 32 Vdc. Applications typiques en mode SMS ou DTMF :

- Installations antivol pour les batiments civils et industriels.

- Installations antivol pour voitures
- Contrôle des systèmes de climatisation /chauffage.

- Contrôle de pompes et de systèmes d'irrigation.
- Contrôle d'installations industrielles.

Carte SIM (non incluse).

Réf. : TDG135 **114,00 €**
Télécommande bidirectionnelle GSM
2 entrées/2 sorties relais.
Coffret Réf. : BOXTDG135 **15,55 €**

Avec composeur téléphonique. Utilisé dans le secteur de la sécurité et de l'automatisme à distance, il dispose de 2 entrées optoisolées et de 2 sorties relais contrôlables par DTMF, appel téléphonique (menu vocal guidé) et commande SMS. Les deux sorties peuvent être utilisées pour gérer à distance des dispositifs divers tels que des lumières, moteurs,etc. Le dispositif offre la possibilité de mémoriser 8 numeros de téléphone pour l'envoi de message d'alarme par appel téléphonique ou sms. La gestion des paramètres peut être faite à distance par SMS ou en local via un PC avec le logiciel adéquat (nécessite une interface USB). L'utilisateur peut personnaliser les messages vocaux et les écouter à l'aide du petit HP incorporé. Boîtier en option. Carte SIM (non incluse).

Comelec 12 / 2018

COMELEC
CD 908 - 13720 BELCODÈNE
Tél. : 04 42 70 63 90
Fax : 04 42 70 63 95
www.comelec.fr