

n°141 HIVER 2017

Radio FM & Lecteur MP3

Enregistreur vocal à PIC

CAO KiCad EDA

Cours MPLAB X

- Effet trémolo
- Etoile de Noël
- Programmateur hebdomadaire
- Reconnaissance faciale avec RaspberryPi

N° 141 Décembre 2017

L 13270 - 141 - F: 8,30 € - RD



Sommaire

ARTICLES

Numéro 141

Hiver 2017

100 pages



04 INFORMATIQUE

RECONNAISSANCE FACIALE AVEC RASPBERRYPI

Nous allons utiliser les fonctionnalités de la librairie SimpleCV afin de reconnaître les visages et les autres détails du corps humain dans des images. Nous concluons l'article avec un programme qui peut reconnaître un visage et activer un relais, ou commander une alarme en présence d'inconnus. Souriez, vous êtes filmés.



16 MAISON

PROGRAMMATEUR HEBDOMADAIRE

Dans cet article, nous vous proposons un temporisateur hebdomadaire disposant d'une sortie relais, d'un clavier et d'un afficheur LCD pour les réglages des différentes fonctions. Ce système est idéal pour commander des dispositifs fonctionnant sur batteries ou panneaux solaires de 12 V. Il comprend une entrée qui permet de forcer l'état du relais lorsqu'un événement externe se produit, cela est utile, comme par exemple, pour désactiver un arrosage automatique lorsqu'il pleut.



27 AUDIO

RADIO FM & LECTEUR MP3 USB

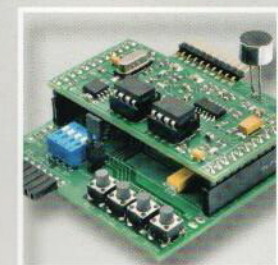
Nous vous décrivons dans cet article un montage atypique, il s'agit d'un récepteur FM avec un port USB sur lequel vous pouvez relier un périphérique de stockage afin de lire vos chansons préférées au format MP3. Sa particularité réside dans le fait qu'il peut être intégré dans une voiture. Le projet source complet en langage C (s'ouvre avec MBLABX) et le programme « FM_Receiver.X.production.hex » du microcontrôleur sont disponibles en téléchargement sur notre site web.



41 MICROCONTRÔLEUR

ENREGISTREUR VOCAL AVEC PIC & EEPROM - PREMIÈRE PARTIE

Deux modules simples, gérés par un microcontrôleur PIC, peuvent résoudre tous les problèmes liés à l'enregistrement et à la reproduction de messages vocaux. Particulièrement adaptés aux systèmes de sécurité et d'alarme, ces modules fonctionnent de manière autonome, mais peuvent également être programmés par un PC.



Les typons des circuits imprimés et les programmes **lorsqu'ils sont libres de droits** sont téléchargeables

L'EDITO HIVER 2017

Chères lectrices, chers lecteurs,

la Rédaction d'Électronique et Loisirs Magazine vous souhaite de Joyeuses Fêtes de Noël ainsi qu'une Bonne et Heureuse Année 2018.

En ces fêtes de fin d'année, la Rédaction vous propose deux montages originaux pour décorer et animer votre maison, notamment l'étoile de Noël avec ses 56 LED Neopixels qui ornera le sommet du sapin en créant des effets lumineux spectaculaires.

Cependant nous n'oublions pas d'être sérieux avec le début de deux cours de haut niveau, un concernant la programmation en C des microcontrôleurs PIC32 et l'autre concernant la réalisation de circuits imprimés sur ordinateur à l'aide du logiciel Open Source KiCad.

Côté réalisations, nous décrivons un projet complet qui est un récepteur FM et lecteur MP3 intégré avec le code source de tous les programmes, un enregistreur vocal à PIC qui remplace les montages à base de circuits ISD devenant de plus en plus rares sur le marché, un programmeur hebdomadaire avec son programme complet ainsi qu'une pédale à effet trémolo pour les adeptes de guitares électriques.

Enfin pour les fans du RaspberryPi, nous proposons un projet complet de reconnaissance faciale.

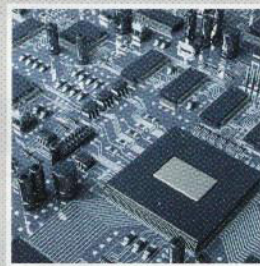
La Rédaction



51 CAO

APPRENEZ À MAÎTRISER KICAD EDA PREMIÈRE PARTIE

Nous commençons ce nouveau cours consacré à la découverte de KiCad EDA, un logiciel de CAO Open Source qui permet de réaliser le développement et la fabrication de cartes électroniques, c'est-à-dire la création de schémas électroniques et de circuits imprimés ainsi que les fichiers Gerber pour la fabrication.



63 AUDIO

EFFET TRÉMOLO POUR GUITARE

Nous vous présentons dans cet article un effet guitare qui fait vibrer le son des cordes à travers un système optique original de modulation d'amplitude. Le montage permet la modulation du signal audio en mode triangulaire, ou en mode rectangulaire (ON/OFF), avec la possibilité d'ajuster, pour chaque onde modulante, la fréquence et la profondeur de la modulation. En plus des commandes pour l'effet, le circuit dispose d'un réglage du volume sonore qui est mis en œuvre sur l'entrée.



73 COURS

COURS MPLAB X IDE PREMIÈRE PARTIE

Nous commençons notre voyage pour découvrir MPLAB X, le nouvel environnement de développement intégré développé et distribué par Microchip afin de remplacer l'environnement MPLAB IDE. C'est l'occasion d'apprendre à connaître les microcontrôleurs de la famille PIC32 de Microchip, et de passer dans la catégorie supérieure des microcontrôleurs fonctionnant en 32 bits.



86 GADGET

ÉTOILE DE NOËL

Nous vous proposons dans cet article de réaliser une étoile de Noël innovante, en utilisant la technologie « LED Neopixel ». Cette étoile est contrôlée par une carte Arduino Micro afin de réduire au minimum l'encombrement. Avec 56 LED Neopixel connectées en cascade et gérées par une carte Arduino, nous avons créé une étoile de Noël à 5 branches qui, placée au sommet du sapin, produira une animation du plus bel effet.



93 GADGET

UNE BOULE DE NOËL À CHANGEMENT DE COULEUR

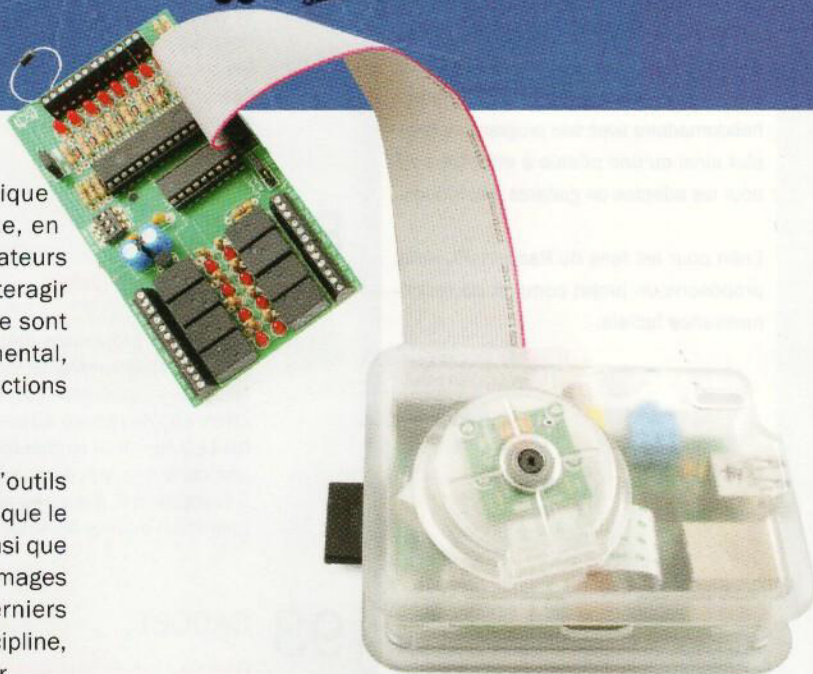
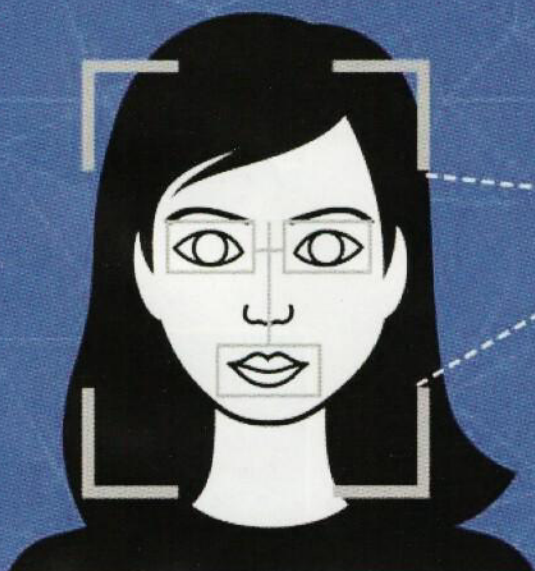
Voici une boule de Noël qui ornera votre sapin d'une manière originale. Elle a la particularité de changer de couleur en produisant des effets lumineux lors du passage d'une couleur à l'autre. Il s'agit d'une boule de Noël à LED autonome, elle utilise une LED RVB spéciale qui intègre un microcontrôleur et qui peut générer des effets lumineux tels que la dissolution des couleurs.



Reconnaissance faciale avec RaspberryPi

Nous allons utiliser les fonctionnalités de la librairie SimpleCV afin de reconnaître les visages et les autres détails du corps humain dans des images. Nous concluons l'article avec un programme qui peut reconnaître un visage et activer un relais, ou commander une alarme en présence d'inconnus. Souriez, vous êtes filmés.

de Marco Magagnin



Les champs d'applications de la vision numérique ont toujours fasciné l'imagination de l'homme, en essayant de fabriquer des robots et des ordinateurs ayant la capacité d'identifier, de reconnaître et d'interagir avec les humains. Aujourd'hui, ces fonctionnalités ne sont plus des rêves mais font partie du domaine expérimental, même si elles ne reflètent pas correctement les prédictions de la science-fiction.

Dans cet article, nous profitons de la disponibilité d'outils informatiques économiques de capture d'images tels que le RaspberryPi et sa caméra vidéo dédiée **CameraPi**, ainsi que des logiciels de capture d'images et de traitement d'images Open Source tels que **OpenCv** et **SimpleCV**. Ces derniers permettent une approche de haut niveau de cette discipline, mais qui reste suffisamment simplifiée pour l'amateur.

Tous les outils et produits ont été présentés dans les magazines **137** (CameraPi) et **140** (Vision numérique avec RaspberryPi et Caméra Pi) d'Electronique et Loisirs Magazine. Dans le numéro 137, nous avons présenté CameraPi et détaillé ses fonctionnalités, en particulier la possibilité d'appliquer des effets aux images ou aux vidéos capturées. Dans le numéro 140, nous avons présenté les différentes possibilités de la vision numérique : acquisition, localisation (ou détection) et identification (ou reconnaissance).

Dans ce même numéro, nous avons décrit et présenté des exemples de détection d'objets simples et d'extrapolation de formes, de couleurs et de contours.

Dans cet article, qui est basé sur le même principe que les précédents cités ci-dessus, bien qu'avec un saut technologique significatif, nous allons étudier la localisation (détection) d'êtres humains à l'intérieur d'une image, plus précisément leurs particularités telles que le visage, les yeux, le nez, etc.

Cette fonctionnalité avancée est présente dans les célèbres réseaux sociaux. Une fois les photos téléchargées sur le réseau, le système les scanne et recherche des visages, les identifie et permet d'associer un nom (voir la figure 1). Si par hasard la même personne est présente dans plusieurs images, elle est reconnue et enregistrée automatiquement, de manière confidentielle (en théorie). Cette dernière caractéristique est ce que nous avons mentionné comme étant l'identification, c'est-à-dire la reconnaissance.

La méthode de reconnaissance

Pour reconnaître des objets réels, ou comme dans notre cas des personnes et leurs détails, dans SimpleCV ainsi que dans OpenCV est présente une méthode connue sous le nom de « **Haar features cascade** » ou aussi comme la méthode « **Viola et Jones** ».

La méthode a été proposée en 2001 par **Paul Viola** et **Michael Jones** (ne pas confondre avec le célèbre chanteur) dans leur article intitulé « **Rapid Object Detection using a Boosted Cascade of Simple Feature** » ce qui signifie : détection rapide d'objets à l'aide d'une cascade de comparaisons successives de caractéristiques simples.

Elle fait partie des toutes premières méthodes **capables de détecter efficacement et en temps réel des objets dans une image**. Inventée à l'origine pour détecter des visages, elle peut également être utilisée pour détecter d'autres types d'objets comme les voitures ou les avions.

La méthode de Viola et Jones est l'une des méthodes les plus connues et les plus utilisées, en particulier pour la détection de visages et la détection de personnes. Elle nécessite plusieurs milliers d'exemples de l'objet que l'on souhaite détecter, pour habituer ou entraîner un classifieur (catégorisation algorithmique d'objets).

Une fois l'apprentissage terminé, le classifieur est utilisé pour détecter la présence de l'objet dans une image en parcourant celle-ci de manière exhaustive,

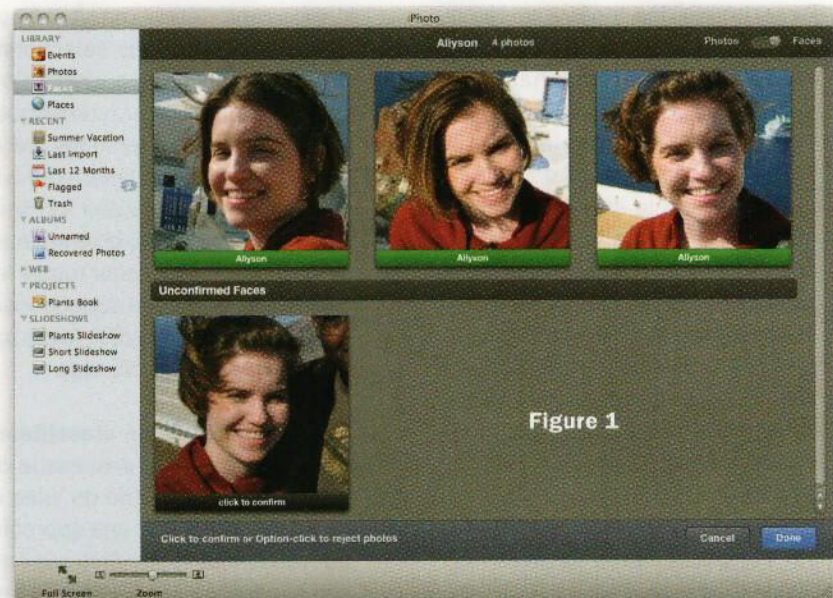


Figure 1

dans toutes les positions et dans toutes les tailles possibles.

La méthode est constituée de 4 composants clés :

- par des **éléments comparables** matérialisés par des réseaux rectangulaires de pixels, appelés caractéristiques (fonctions) de Haar. Plutôt que de travailler directement sur les valeurs des pixels, et pour être à la fois plus efficace et plus rapide, Viola et Jones proposent d'utiliser des caractéristiques, c'est-à-dire une représentation synthétique et informative, calculée à partir des valeurs des pixels. Viola et Jones définissent des caractéristiques très simples, appelées caractéristiques « pseudo-Haar », et qui sont calculées par la différence des sommes des pixels de deux ou plusieurs zones rectangulaires adjacentes.
- par le **calcul rapide et efficace des caractéristiques d'une image**, Viola et Jones proposent également une nouvelle méthode, appelée « image intégrale ». C'est une représentation sous la forme d'une image, de même taille que l'image originale, qui en chacun de ses points contient la somme des pixels situés au-dessus de lui et à sa gauche. Les caractéristiques sont calculées pour toutes les positions et pour toutes les échelles dans une fenêtre de détection de petite taille, typiquement de 24×24 pixels. Ainsi, un très grand nombre de caractéristiques par fenêtre est généré.

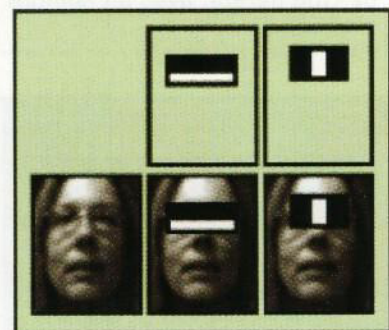
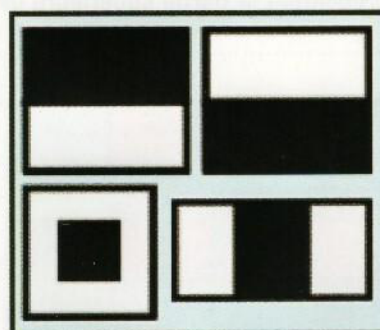


Figure 2

Par exemple une fenêtre de taille 24×24 génère environ 160 000 caractéristiques. En phase de détection, l'ensemble de l'image est parcouru en déplaçant la fenêtre de détection d'un certain pas dans le sens horizontal et vertical (ce pas valant 1 pixel dans l'algorithme original). Les changements d'échelles se font en modifiant successivement la taille de la fenêtre de détection. Viola et Jones utilisent un facteur multiplicatif de 1,25, jusqu'à ce que la fenêtre couvre la totalité de l'image ;

- par l'**application de la méthode d'apprentissage** pour les systèmes de vision numérique. Il s'agit de l'utilisation de la méthode de « boosting » qui permet de sélectionner les meilleures caractéristiques. Le « boosting » est un principe qui consiste à construire un classifieur « fort » à partir d'une combinaison pondérée de classifieurs « faibles », c'est-à-dire donnant en moyenne une réponse meilleure qu'un tirage aléatoire. Viola et Jones adaptent ce principe en assimilant une caractéristique à un classifieur faible, en construisant un classifieur faible qui n'utilise qu'une seule caractéristique. L'apprentissage du classifieur faible consiste alors à trouver la valeur de seuil de la caractéristique qui permet de mieux séparer les exemples positifs des exemples négatifs. Le classifieur se réduit alors à un couple (caractéristique, seuil). L'algorithme de « boosting » utilisé est en pratique une version modifiée d'AdaBoost, qui est utilisée à la fois pour la sélection et pour l'apprentissage d'un classifieur « fort ». L'algorithme est de type itératif, avec un nombre

d'itérations déterminé. À chaque itération, l'algorithme sélectionne une caractéristique, qui sera ajoutée à la liste des caractéristiques sélectionnées aux itérations précédentes, et le tout va contribuer à la construction du classifieur fort final. Cette sélection se fait en entraînant un classifieur faible pour toutes les caractéristiques et en sélectionnant celui avec l'erreur la plus faible sur l'ensemble d'apprentissage ;

- par une **cascade de classifieurs** afin d'accélérer le processus de détection. La méthode de Viola et Jones est basée sur une approche par recherche exhaustive sur l'ensemble de l'image, qui teste la présence de l'objet dans une fenêtre pour toutes les positions et pour plusieurs échelles. Cette approche est extrêmement lourde en calculs d'où l'idée d'organiser l'algorithme de détection en une cascade de classifieurs. Si la fenêtre contient l'objet, l'exemple est passé au classifieur suivant. Si la fenêtre ne contient pas l'objet, l'exemple est définitivement écarté. L'idée est que l'immense majorité des fenêtres testées étant négatives (elles ne contiennent pas l'objet), il est avantageux de pouvoir les rejeter avec le moins de calculs possible. Ainsi, les classifieurs les plus simples, donc les plus rapides, sont situés au début de la cascade, et rejettent très rapidement la grande majorité des exemples négatifs.

L'algorithme d'extraction des caractéristiques est dérivé de l'**ondelette de Haar**. Il s'agit d'une fonction constante par morceaux, l'ondelette est représentée par deux tableaux carrés, l'un représentant la partie supérieure de

l'ondelette et l'autre représentant la partie inférieure de l'ondelette.

Dans le modèle utilisé pour extraire les caractéristiques des images, les matrices de référence ont des formes différentes, telles que celles représentées en figure 2. Elles sont plus adaptées pour identifier les particularités du corps humain, comme les yeux ou le nez. D'où leur nom de « Haar Features » afin de les distinguer de la signification originale (ondelette).

La même figure montre la forme des fonctions utilisées par OpenCV et Simple CV. Ces derniers vérifient la présence ou non d'une « feature de Haar » dans une partie de l'image, à laquelle a été soustraite la valeur moyenne des pixels de la partie noire du « masque », à partir de la valeur moyenne des pixels de la partie claire du « masque ».

Si la différence est supérieure à une certaine valeur de seuil, la caractéristique est considérée comme présente. La valeur de seuil est déterminée pour chaque fonction pendant le processus d'élaboration de la fonction pour localiser des objets spécifiques, comme les détails du corps humain.

Le processus d'apprentissage se concrétise en « présentant » au système de vision numérique autant d'images que possible de l'objet à identifier et autant d'images que possible qui n'ont rien à voir avec l'objet.

À partir de la masse des données, les valeurs de seuil sont calculées pour chacune des caractéristiques, qui dans le cas de OpenCV et SimpleCV, sont stockées dans un fichier au format « .xml ».

Les parties de l'image analysées se composent généralement de matrices

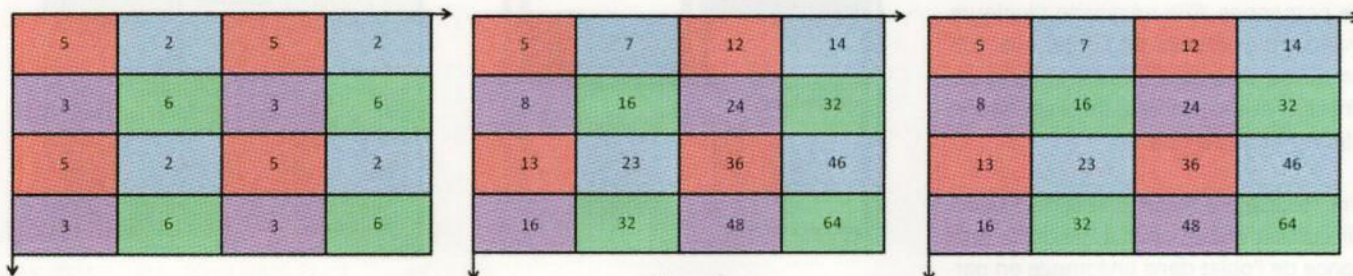


Figure 3

de 24 x 24 pixels, qui doivent être « comparées » avec toutes les caractéristiques attendues.

En traitant une matrice de 24 x 24 pixels, nous obtenons environ 160 000 caractéristiques, cela nécessiterait des opérations de comparaison et des durées de traitement inacceptables.

Comme nous l'avons mentionné, Viola et Jones ont introduit une simplification du calcul basée sur l'image intégrale. Cette méthode permet d'effectuer les calculs pour l'ensemble de la matrice avec seulement 4 pixels à la fois (voir la figure 3).

Par ailleurs, la méthode a été développée d'un point de vue mathématique en 1984, vous pouvez la consulter à cette adresse : <https://computerscience-source.wordpress.com/2010/09/03/computer-vision-the-integral-image/>.

Le processus consiste à assigner à un pixel donné dans une position donnée dans la matrice (image) la somme de tous les pixels dans la zone au-dessus et à gauche de sa position, comme le montre la figure 3.

En partant du pixel en haut à gauche et en poursuivant vers la droite et vers le bas, le processus de calcul incrémental de la valeur de chaque pixel est assez efficace.

Après le processus d'intégration, le calcul de la valeur moyenne des pixels contenus dans une zone donnée de l'image, telle que la zone « D » de la figure 3, se fait en appliquant la formule ci-dessous aux pixels juste en bas à droite de chaque zone :

$$D = A+B+C+D - (A+B) - (A+C) + A$$

Malgré la simplification, le volume de traitement est encore excessif pour être efficace.

Cependant, les auteurs de la méthode ont remarqué que certaines caractéristiques étaient plus pertinentes que d'autres afin de reconnaître certaines parties, telles que les yeux, alors que d'autres caractéristiques étaient moins pertinentes, comme les joues. Certaines ne sont carrément pas pertinentes.

Comment sélectionner les caractéristiques les plus pertinentes pour reconnaître un élément particulier et en éliminer d'autres ? Tout simplement en utilisant l'algorithme mathématique connu sous le nom de « **Adaboost** » (Adaptive Boosting). Il s'agit d'un algorithme « d'apprentissage automatique » (machine learning) conçu pour optimiser la performance d'autres algorithmes d'apprentissage.

D'une manière simple, il a pour but de rechercher le plus petit ensemble possible qui garantit, pour un pourcentage donné (par exemple 75 %) de précision dans le résultat, la reconnaissance ou le rejet d'un objet.

Le processus mentionné ci-dessus montre qu'avec 200 caractéristiques, il est possible d'obtenir un degré de localisation avec un pourcentage de précision de 95 %.

Dans un souci de précision, les implémentations de la méthode utilisent environ 6 000 caractéristiques sélectionnées parmi les 160 000 initiales.

Si nous nous étions arrêtés ici, pour chaque portion de matrice de 24 x 24 pixels, nous devrions comparer 6 000 caractéristiques, ce qui est considérable pour une image entière.

Heureusement, les auteurs ont trouvé une solution à ce problème.

Dans la plupart des cas, dans une image, la plupart des zones ne contiennent pas d'objets recherchés.

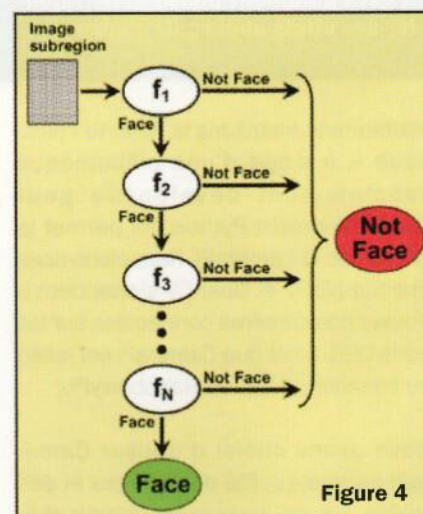


Figure 4

Pour cette raison, il convient de définir une méthode simple pour comprendre si une partie de l'image appartient à l'objet recherché ou si elle ne fait pas partie de celui-ci. Dans ce deuxième cas, la partie de l'image est écartée et n'est plus traitée.

La méthode doit en quelque sorte se concentrer sur les zones qui semblent faire partie des objets recherchés et doit les analyser en profondeur. Ce besoin a conduit à l'introduction du concept de « **cascade** » ou de **cascade de classifieurs**.

Au lieu d'appliquer les 6 000 caractéristiques à chaque zone de l'image, celles-ci sont regroupées en différents niveaux de classification et appliquées un niveau à la fois à l'image.

Si une zone de l'image est rejetée par les classifieurs du premier niveau, elle ne sera plus traitée pour les niveaux supérieurs.

Si la zone de l'image passe le premier niveau, elle sera comparée aux caractéristiques qui appartiennent au deuxième niveau, et ainsi de suite.

Si une zone de l'image passe tous les niveaux, alors le détail appartient à l'objet recherché, tel que représenté en figure 4.

La méthode de classification proposée subdivise les 6 000 caractéristiques en 38 niveaux de classification. Les 5 premiers niveaux sont composés respectivement de 1, 10, 25, 25 et 50 caractéristiques. De cette façon, il est nécessaire de comparer chaque zone de l'image avec environ 10 caractéristiques en moyenne.

Avant de passer à un exemple pratique, rappelons-nous que pour fonctionner, la méthode doit être constituée par une grande quantité d'images (600 et plus) contenant différentes versions d'objets à rechercher et autant d'images qui ne les contiennent pas. Heureusement, pour les formes principales associées au corps humain, nous trouvons déjà des travaux réalisés et encodés dans les packages de OpenCV et de SimpleCV. Cela nous permet d'expérimenter immédiatement.

Examinons le visage

Après notre longue introduction à la méthode de reconnaissance d'objet basée sur l'algorithme de « **Haar-FeaturesCascade** », passons à des exemples pratiques. Nous profitons également de l'amélioration du système d'exploitation du RaspberryPi et de l'installation d'une nouvelle librairie qui facilite la gestion de la CameraPi.

Pour nos expériences, nous avons besoin que le **RaspberryPi soit connecté à un réseau local avec un câble Ethernet et à un moniteur HDMI** avec le câble approprié.

Reportez-vous aux anciens numéros. Nous préférons nous connecter au RaspberryPi à distance via le protocole **SSH** en utilisant les logiciels **PuTTY** et **WinSCP**. Cependant, rien ne vous interdit de connecter un clavier et une souris aux sorties USB et d'exécuter des exemples à partir du bureau.

Nous avons décrit dans les numéros 137 et 140 l'installation et la configuration de CameraPi ainsi que l'installation et la configuration des logiciels OpenCV et SimpleCV. Nous vous conseillons de vous reporter à ces articles pour une compréhension de la suite.

Nous supposons aussi que vous avez les connaissances de bases pour la gestion du RaspberryPi et du système d'exploitation Raspbian.

Alimentons le RaspberryPi, vérifions que le moniteur fonctionne et connectez-vous avec PuTTY (ou comme vous le souhaitez) en tant qu'utilisateur « root ». Comme d'habitude, mettons à jour le système d'exploitation avec les commandes :

```
apt-get update
apt-get upgrade
apt-get dist-upgrade
```

Si votre distribution n'est pas récente, il faudra du temps pour effectuer la mise à jour. Ensuite, avant de continuer il faudra redémarrer avec la commande :

```
reboot
```

Intel Pocket Avatars une messagerie vidéo instantanée

Cette fois, ce n'est pas un processeur mais une application particulière dédiée aux smartphones, qui décrète l'entrée d'Intel dans le monde des applications. Pocket Avatar est en fait une nouvelle application de messagerie instantanée pour Android et iOS, qui permet d'envoyer à ses contacts des messages en format vidéo.

Contrairement à Snapchat ou Slingshot, l'application d'Intel utilise des avatars en 3D à la place des visages des utilisateurs. Un algorithme développé par Intel permet de donner vie à cet avatar. Cette application permet aux utilisateurs de discuter en utilisant des avatars 3D créés grâce aux expressions faciales capturées par la caméra du smartphone.

Grâce à la technologie de suivi du visage, les contacts prendront les visages les plus divers, comme ceux d'un dessin animé, mais les expressions et les mouvements de la bouche reproduiront ceux de l'utilisateur de l'avatar.

L'utilisateur enregistre son message vidéo en filmant son visage. Ensuite, il sélectionne l'avatar qu'il souhaite et là, la magie d'Intel opère et permet à l'avatar de prendre les expressions faciales de l'utilisateur et de bouger la bouche en même temps que les mots prononcés par l'utilisateur.

Il est également possible de rajouter des effets audio puis d'envoyer la

vidéo directement à un contact qui dispose aussi de l'application ou alors sous forme de vidéo par mail, MMS ou encore directement sur les réseaux sociaux.

L'application est un test pour la technologie de suivi du visage, confirmant ainsi l'engagement des laboratoires Intel dans la recherche 3D et le traitement vidéo. Elle est gratuite pour les smartphones Android et iPhone et dispose d'une cinquantaine d'avatars différents, dont certains sont payants.

Maintenant, installons la librairie « picamera », il s'agit d'une bibliothèque spécialement développée pour l'environnement Python qui permet la gestion de la CameraPi. Rappelons-nous que SimpleCV et OpenCV s'attendent à trouver des caméras connectées sur les ports USB, alors que CameraPi est reliée au connecteur CSI du RaspberryPi.

Nous avons choisi d'utiliser CameraPi pour la qualité des images et des vidéos que nous pouvons obtenir et la

gamme des fonctions disponibles pour le traitement de l'image.

Dans l'article du numéro 140, nous avons utilisé l'artifice pour appeler, depuis le système, les fonctionnalités d'interfaçage de CameraPi et enregistrer l'image que nous traiterions par la suite avec les instructions (reportez-vous au Listing1 du numéro 140 à la page 51) : `subprocess.call("raspistill -n -w %s -h %s -o Listing1_1.bmp" % (640, 480), shell=True)`

Avec la nouvelle librairie, cela n'est plus nécessaire et nous interagissons avec la CameraPi à l'aide des fonctions Python disponibles. La documentation de la librairie « **picamera** », ainsi que d'autres informations utiles sur les fonctionnalités disponibles et l'utilisation de CameraPi, se trouvent à l'adresse suivante (en anglais) :

<http://picamera.readthedocs.org/en/release-1.5>.

Installons la bibliothèque avec la commande (voir la figure 5) :

apt-get install python-picamera

Dans les exemples suivants, nous nous concentrons sur la reconnaissance du visage et des détails du corps humain, pour lesquels nous avons les caractéristiques « Haar Cascade ». Quelles sont-elles ? Celles pour lesquelles les développeurs de OpenCV et de SimpleCV ont effectué le processus d'apprentissage et qui l'ont intégré dans les deux librairies. Nous les trouvons dans le dossier (voir la figure 6) :

/usr/local/lib/python2.7/dist-packages/SimpleCV/Features/HaarCascades/

D'accord, le chemin est un peu long, mais c'est leur emplacement. Nous vous avertirons lorsque les problèmes surgiront. Notez que la librairie SimpleCV est installée dans le dossier **Python2.7**.

Vous devez utiliser cette version de Python pour exécuter les exemples. Si vous exécutez les exemples sous Python3, vous obtiendrez probablement des erreurs.

De plus, pour atteindre le fichier « .xml » dans le dossier cité, normalement le nom suffit. Par exemple, « face », comme indiqué dans la plupart des exemples téléchargeables sur le web.

Cependant, si ce paramètre produit des erreurs, vous devez utiliser le chemin complet. Dans notre cas, nous avons constaté que cela fonctionnait pour appeler la « Feature » avec l'extension, par exemple « face.xml », mais cela n'est pas toujours le cas.

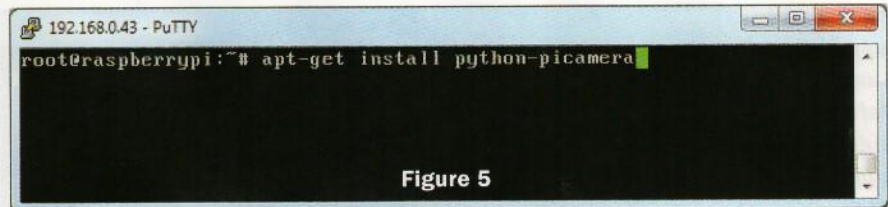


Figure 5

Dans le dossier mentionné ci-dessus,

nous trouvons un certain nombre de fichiers qui peuvent être différents selon la version de SimpleCV installée. Dans notre cas, nous avons installé la version 1.3 (<http://simplecv.org/download>) qui comporte les fichiers suivants :

eye.xml
fullbody.xml
mouth.xml
two_eyes_big.xml
face2.xml
glasses.xml
nose.xml
two_eyes_small.xml
face3.xml
left_ear.xml

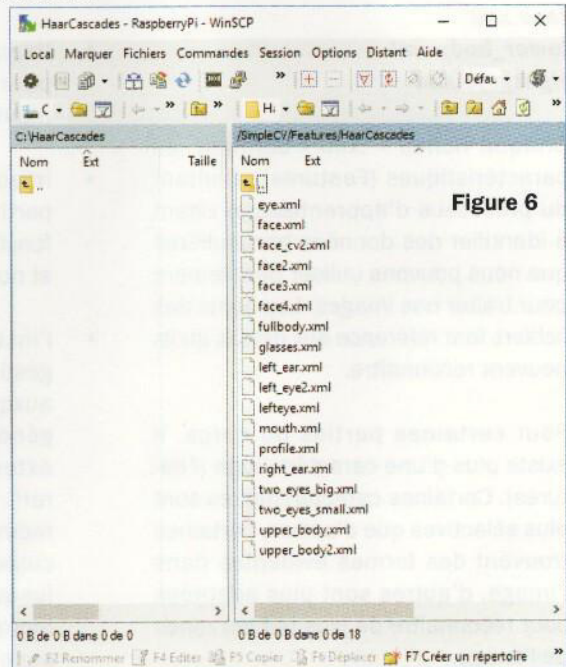


Figure 6

Listing 1

```
#!/usr/bin/python2.7

# Programme test Haar Features

import picamera
from SimpleCV import Image
import time

with picamera.PiCamera() as camera:
    camera.resolution = (640, 480)
    camera.start_preview()
    time.sleep(10)
    camera.capture('img.jpg')
    img=Image("img.jpg")

    print(img.listHaarFeatures())
    trovato=img.findHaarFeatures('face.xml')
    if trovato:
        for trovato in trovato:
            print " Trouvé aux coordonnées: " + str(trovato.coordinates())
            trovato.draw()
    else:
        print " Non trouvé "
        camera.stop_preview()
        img.save('img1.jpg')
        img.show()
        time.sleep(10)
```


profile.xml
upper_body2.xml
face4.xml
left_eye2.xml
right_ear.xml
upper_body.xml
face_cv2.xml
lefteye.xml
right_eye2.xml
face.xml
lower_body.xml
right_eye.xml

Chaque fichier « .xml » contient les caractéristiques (Features) résultant du processus d'apprentissage visant à identifier des données particulières que nous pouvons utiliser directement pour traiter nos images. Les noms des fichiers font référence aux détails qu'ils peuvent reconnaître.

Pour certaines parties du corps, il existe plus d'une caractéristique (Features). Certaines caractéristiques sont plus sélectives que d'autres, certaines trouvent des formes évidentes dans l'image, d'autres sont plus adaptées pour reconnaître de plus petites zones de l'image.

Dans les applications réelles, plusieurs caractéristiques s'appliquent à une même image, alors testez-les toutes.

Dans des cas plus particuliers, ou pour reconnaître des objets autres que des visages et/ou des détails du corps humain, vous devez réaliser vos propres fichiers de caractéristiques avec des processus d'apprentissages appropriés. Mais cela est une autre histoire.

Encore une petite curiosité, dans la version de SimpleCV que nous avons installée, le nom de fichier « lefteye.xml » a été écrit sans le tiret bas (tiret de la touche 8 du clavier). Les erreurs, en informatique, sont toujours en embuscade. Souvenons-nous de cette anomalie si nous devons faire appel à des programmes pour cette caractéristique.

Pour commencer, copiez le programme du Listing 1 dans un fichier contenu dans le dossier « /home » ou un autre dossier à votre préférence, nommez-le « Test_Haar.py ». Ce programme effectue les tâches suivantes :

```

192.168.0.43 - PuTTY
root@raspberrypi:/home# python Prova_Haar.py
[ 'eye.xml', 'two_eyes_big.xml', 'face.xml', 'profile.xml', 'face4.xml', 'right_e
ye.xml', 'lefteye.xml', 'upper_body.xml', 'lower_body.xml', 'left_ear.xml', 'upp
er_body2.xml', 'fullbody.xml', 'face2.xml', 'glasses.xml', 'right_eye2.xml', 'tw
o_eyes_small.xml', 'mouth.xml', 'face_cv2.xml', 'nose.xml', 'left_eye2.xml', 'ri
ght_ear.xml', 'face3.xml' ]
None
Trovato alle coordinate : 1340 2691
root@raspberrypi:/home#
    
```

Figure 7

- importe la librairie « picamera » pour gérer la CameraPi directement à partir du langage Python ;
- importe la fonction « Image » à partir de la librairie SimpleCV et la fonction « time » pour gérer le délai et nous laisser « voir » les images ;
- l'instruction « with » permet une gestion « propre » des ressources auxquelles elle est associée, en général ce sont des ressources externes comme notre CameraPi. La ressource est utilisée au moment où l'instruction est exécutée et est délivrée correctement jusqu'à la fin du programme ou lorsqu'une erreur se produit. La seconde partie de l'instruction crée l'objet « camera » en tant qu'instance de la classe Picamera, fournie par la librairie que nous avons précédemment installée ;
- définit la résolution de la CaméraPi ;
- initialise l'aperçu (preview) de l'image provenant de la caméra (rappelez-vous « raspistill? ») ;



Figure 9



Figure 10



Figure 11



Figure 12

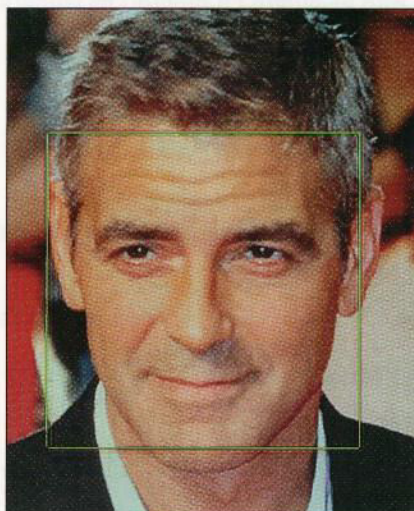


Figure 8

- le « Time sleep » conserve l'aperçu actif pendant 10 secondes, donc nous avons le temps de nous positionner devant la caméra. Nous pouvons utiliser des photos ou images issues de magazines. Testez différentes alternatives. Vous comprendrez que ce système n'est pas la meilleure façon d'ouvrir une porte ;
- capture l'image et l'enregistre dans un fichier ;
- appelle le fichier en tant qu'objet « Image » pour le rendre utilisable par SimpleCV ;
- avec la fonction « listHaarFeatures() », ajoute à la console la liste des caractéristiques (Features) disponibles, comme nous l'avons vu plus haut en figure 6 ;
- enfin, le programme effectue la recherche des détails anatomiques indiqués dans les caractéristiques correspondantes qui sont rappelées dans la fonction ;

- pour chaque détail identifié, le programme ajoute à la console les coordonnées du centre du détail et met en surbrillance à l'aide d'un rectangle la zone de l'image où le détail a été localisé ;
- à la fin du processus, l'aperçu s'arrête et l'image est sauvegardée avec les détails en surbrillance, puis elle est affichée sur l'écran pendant 10 secondes.

En modifiant les instructions dans le programme, vous pouvez expérimenter toute la gamme des détails reconnaissables et vous donner une idée pratique de l'incidence de la taille de l'image, des conditions de luminosité et de l'orientation du visage dans l'image.

Par exemple, reconnaître un nez dont le visage est en position frontale dans l'image est très différent de reconnaître le nez de profil. Pour tous ces tests, vous pouvez utiliser le programme du listing 1 en modifiant simplement les paramètres de la fonction « findHaarFeatures() ».

Voici quelques exemples. En figure 7, vous pouvez observer la console du processus de détection du visage dans l'image. Dans la figure 8, le visage identifié. Reconnu ?

En figure 9, nous utilisons le programme du Listing 1 avec la caractéristique « lefteye.xml » afin de reconnaître et de mettre en surbrillance l'œil gauche dans un carré.

```
trovati=img.findHaarFeatures('lefteye.xml')
```

En figure 10, le résultat du programme avec la caractéristique « nose.xml ».

```
trovati=img.findHaarFeatures('nose.xml')
```

En figure 11, la caractéristique « two_eyes_big.xml » permet de localiser la zone de l'image où les deux yeux sont présents.

```
trovati=img.findHaarFeatures('two_eyes_big.xml')
```

En figure 12, le résultat du programme avec la caractéristique « mouth.xml ».

```
trovati=img.findHaarFeatures('mouth.xml')
```

Dans le cas où le programme ne détecte pas de détail désiré, essayez avec les caractéristiques alternatives, comme par exemple, dans le cas des deux yeux : « two_eyes_small.xml ».

Maintenant, c'est à vous d'expérimenter tout ce que vous pouvez. Modifiez le programme pour capturer une image, même à partir d'un fichier, et recherchez les détails que vous désirez. Par exemple, des visages entiers, des bustes, des yeux, des oreilles, des lunettes, etc.

Enfin, nous concluons avec un autre programme un peu plus complexe. Il permet de mémoriser un visage et de comparer celui-ci à d'autres visages analysés dans les images suivantes, pour savoir s'il est identique ou pas à un autre visage.

C'est une sorte de « mot de passe humain » qui pourrait être utilisé, par exemple, pour permettre l'accès à un local ou à un ordinateur. Nous vous conseillons de le faire qu'à titre de démonstration car le programme est très simple et il ne doit pas être utilisé pour une véritable application.

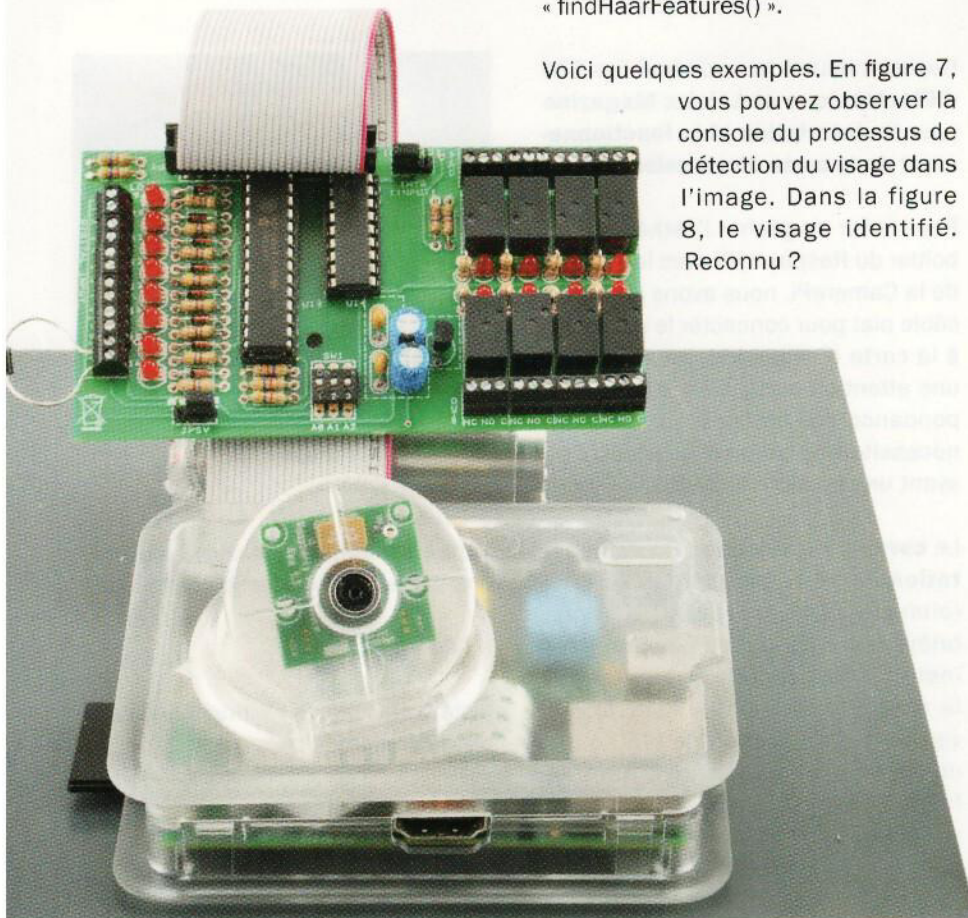


Figure 13 : le RaspberryPi avec la CameraPi.


```
192.168.0.43 - PuTTY
root@raspberrypi:/home# python pifacepass.py
Salvataggio volto di riferimento eseguito
Termino il programma
root@raspberrypi:/home#
```

Figure 14

```
192.168.0.43 - PuTTY
root@raspberrypi:/home# python pifacepass.py
Bentornato - mi sembri il viso giusto
Desideri utilizzare l'ultima foto come password? Y/N
OK...
root@raspberrypi:/home#
```

Figure 16

```
192.168.0.43 - PuTTY
root@raspberrypi:/home# python pifacepass.py
Non ti riconosco
Attivo l'allarme
root@raspberrypi:/home#
```

Figure 18

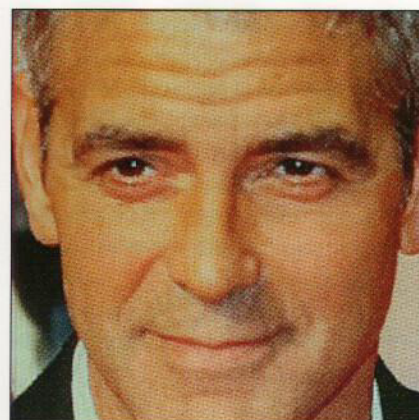


Figure 15

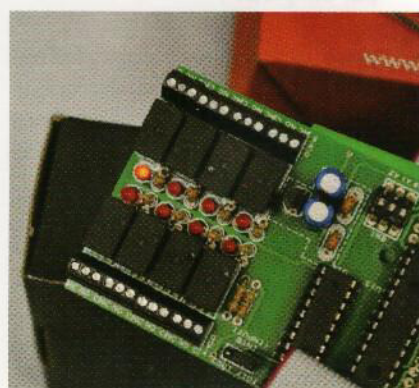


Figure 17

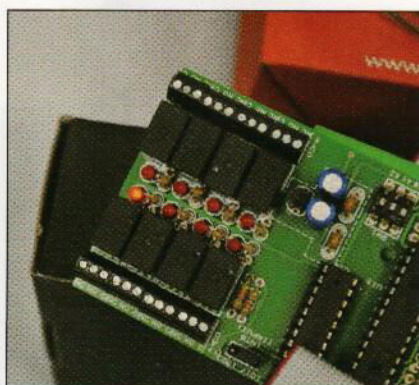


Figure 19

Vous devez effectuer l'apprentissage avec une seule image au format « jpeg ». Pour que le visage de référence puisse être reconnu, vous devez analyser les visages successifs avec un angle et une position très similaire à celle utilisée lors de la sauvegarde du visage de référence. Après avoir effectué l'apprentissage du programme, essayez de « reconnaître » votre visage à partir de l'une de vos photos. Vous serez surpris par le comportement du système.

Étudions maintenant le programme « **pyfacepass.py** » du Listing 2, dont la version originale est disponible à l'adresse : <https://gist.github.com/rishimukherjee/2220293>.

Nous l'avons modifié pour qu'il fonctionne avec le **RaspberryPi**, la **CameraPi** et la **carte d'extension I2C** (décrite dans le numéro 134) qui comporte 8 sorties relais, de sorte qu'un relais puisse être déclenché pour une détection positive ou pour une sortie d'alarme en cas d'échec de la reconnaissance.

Reportez-vous à la revue **numéro 134 d'Electronique et Loisirs Magazine** pour la description et le fonctionnement de la carte d'extension I2C.

Pour éviter de gâcher l'esthétique du boîtier du RaspberryPi avec le support de la CameraPi, nous avons utilisé un câble plat pour connecter le port GPIO à la carte d'extension, en accordant une attention particulière à la correspondance des broches. La carte I2C nécessite une alimentation séparée ayant une tension de 9 VDC.

Le **cavaller de sélection de l'alimentation** doit être en position « **5VR** » (alimentation séparée). Résumons brièvement les étapes à suivre pour installer le logiciel de gestion de la carte d'extension. Activez le driver (pilote) de gestion de l'I2C en le mettant **en commentaire** dans le fichier « **raspi-blacklist.conf** » qui se trouve dans le répertoire « **/etc/modprobe.d** ».

Effectuez un redémarrage pour prendre en compte la nouvelle configuration.

Tapez la commande suivante pour charger le pilote de gestion du bus I2C :

modprobe i2c-dev

Installez la librairie « **wiringpi2** » à l'aide de la commande :

pip install wiringpi2

Si l'une de ces opérations vous semble complexe ou si vous voulez approfondir le fonctionnement de la carte d'extension, reportez-vous à l'article paru dans

le numéro 134 intitulé « Carte d'extension I2C 64 I/O pour RaspberryPi ».

Examinons maintenant la logique du programme. À l'initialisation du programme, les bibliothèques nécessaires à la gestion des images et à la gestion de la carte d'extension sont importées.

Les paramètres sont réglés pour configurer la fonction qui identifie des points significatifs pour comparer l'image de référence avec celles capturées par la caméra. Les broches de la carte d'extension sont configurées en sorties (sortie OK et sortie d'alarme) en fonction du résultat de la reconnaissance. Il s'agit respectivement des broches **GPA0** et **GPA7**.

Listing 3

```
#!/usr/bin/python

import time
from SimpleCV import Image, Color

immagine = Image("password.jpg")
keys = immagine.findKeypoints()
keys.draw(color=Color.BLUE)
immagine.show()
immagine.save('foto_2.jpg')
time.sleep(10)
```

Après les déclarations, le programme recherche l'existence d'un fichier contenant l'image de référence à utiliser pour reconnaître les visages capturés. Si l'objet « **password** » n'existe pas, il est défini comme « **None** » (aucun).

Dans cette condition, le visage qui apparaît sur la caméra pour la première fois est mémorisé en tant que visage de référence et le programme finit son exécution (voir la figure 14). Comme image de référence, est mémorisé l'encadrement qui représente le seul visage, comme visible en figure 15.

Par contre, si une image de référence précédemment mémorisée est trouvée, le traitement suivant est effectué :

- vérification, à l'aide de la fonction « **findHaarFeatures** » que nous avons précédemment décrite, si un visage est détecté dans l'image. Si aucun visage n'est détecté, un

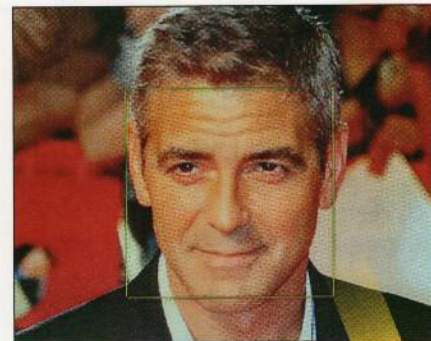
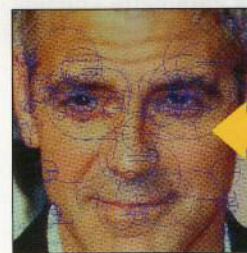


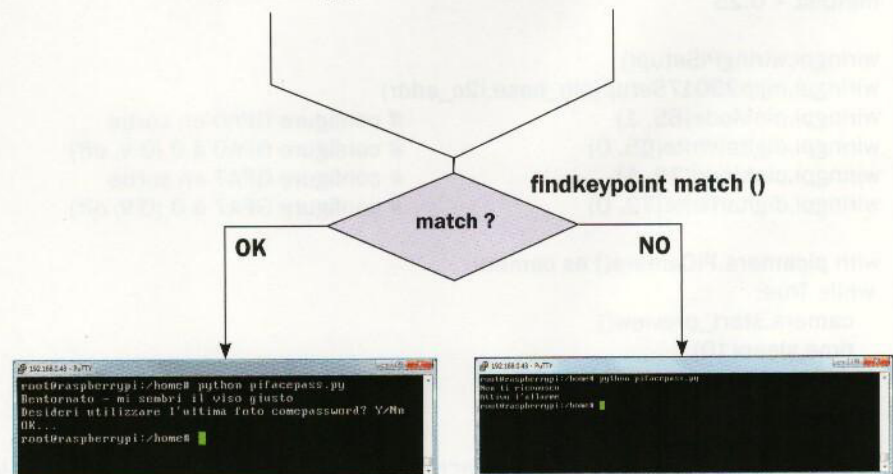
Figure 20



pifacepw.jpg

password.jpg

passwordmatch.jpg



message d'erreur est généré et le programme se termine ;

des deux images et vérifie le niveau de compatibilité ;

- sélection du visage appartenant au premier plan de l'image (ce qui dans le cadre apparaît en plus grand, c'est-à-dire au premier plan), et création de l'objet « **template** » (modèle) ;
- le visage ainsi détecté est recadré et sauvegardé avec le nom « **passwordmatch.jpg** » ;
- les objets « **password** » (mot de passe) qui contiennent le visage de référence et les objets « **template** » (modèles) qui contiennent le visage nouvellement identifié, sont comparés à l'aide de la fonction « **findKeypointMatch()** ». Cette dernière compare les points importants extraits
- dans le cas d'une **détection positive du visage**, un message de reconnaissance est affiché et le **relais** correspondant à la **sortie « 0 »** est **activé** (voir les figures 16 et 17). C'est à cet endroit du programme que vous pouvez ajouter des instructions supplémentaires liées à la reconnaissance, telles que l'envoi d'un message par mail ou d'un tweet. Le programme demande ensuite si vous souhaitez utiliser le dernier visage comme nouveau visage de référence ;
- si oui, le nouveau visage est enregistré dans le fichier « **password.jpg** ». Dans tous les cas, le programme est alors terminé ;

Listing 2

```
#!/usr/bin/python

import picamera
import time
from SimpleCV import Color, Image, np
import wiringpi2 as wiringpi

pin_base = 65
i2c_addr = 0x20

quality = 400                                     # Paramètre « quality » pour la fonction findKeypointMatch
minMatch = 0.3                                     # Paramètre « minDist » pour la fonction findKeypointMatch

try:
    password = Image("password.jpg")
except:
    password = None

mode = "unsaved"
saved = False
minDist = 0.25

wiringpi.wiringPiSetup()
wiringpi.mcp23017Setup(pin_base,i2c_addr)
wiringpi.pinMode(65, 1)                           # configure GPA0 en sortie
wiringpi.digitalWrite(65, 0)                       # configure GPA0 à 0 (0 V, off)
wiringpi.pinMode(72, 1)                           # configure GPA7 en sortie
wiringpi.digitalWrite(72, 0)                       # configure GPA7 à 0 (0 V, off)

with picamera.PiCamera() as camera:
    while True:
        camera.start_preview()
        time.sleep(10)
        camera.capture('pifacepw.jpg')
        image=Image("pifacepw.jpg")
        camera.stop_preview()
        faces = image.findHaarFeatures("face.xml") # Reconnaît le visage à travers le fichier HaarCFeatures « face »
        if faces:
            if not password:
                faces.draw()
                face = faces[-1]
                password = face.crop().save("password.jpg")
                print " Enregistrement du visage de référence "
                print " Fin du programme "
                break
            else:
                faces.draw()
                face = faces[-1]
                template = face.crop()
                template.save("passwordmatch.jpg")
                keypoints = password.findKeypointMatch(template,quality,minDist,minMatch)
                if keypoints:
                    print " Bienvenue - vous avez le visage droit "
                    wiringpi.digitalWrite(65, 1)
                    wiringpi.digitalWrite(72, 0)
                    domanda = raw_input("Souhaitez-vous utiliser la dernière photo comme mot de passe ? Y/N").strip()
                    if domanda == "Y":
                        image = cam.getImage().scale(320, 240)
                        faces = image.findHaarFeatures("face.xml")
                        tryit = 1
                        while not tryit == 10 or not faces:
                            image = cam.getImage().scale(320, 240)
```



```

    faces = image.findHaarFeatures("face.xml")
    tryit += 1
    if not faces:
        "Pas de visage trouvé"
        break
    else:
        faces.draw()
        face = faces[-1]
        password = face.crop().save("password.jpg")
        face.crop().show()
        print " Enregistrement effectué "
        print " Fin du programme "
        time.sleep(1)
        break
    else:
        print "OK..."
        break

else:
    print " Je ne vous reconnais pas "
    print " Activation de l'alarme "
    wiringpi.digitalWrite(65, 0)
    wiringpi.digitalWrite(72, 1)
    break
else:
    break

```

- si le **résultat de la comparaison entre le visage de référence et le visage nouvellement acquis est négatif**, le **relais** correspondant à la **sortie « 7 »** est activé (voir les figures 18 et 19). C'est à cet endroit du programme que vous pouvez ajouter des instructions à exécuter pour un défaut de reconnaissance, comme par exemple envoyer un mail ou un tweet. Le programme ensuite se termine.

Cette méthode correspond à celle utilisée par les programmes d'archivage de photos, que nous avons cités dans l'article, pour identifier, nommer, puis reconnaître des personnages dans nos images.

Avant de conclure, examinons les points clés ou points importants (keypoints) cités dans la méthode de correspondance utilisée dans le programme du Listing 2.

Les « **keypoints** » sont les points importants qui vous permettent de comparer les points d'une image calculés et extraits à l'aide de la fonction « **findKeypoints()** ».

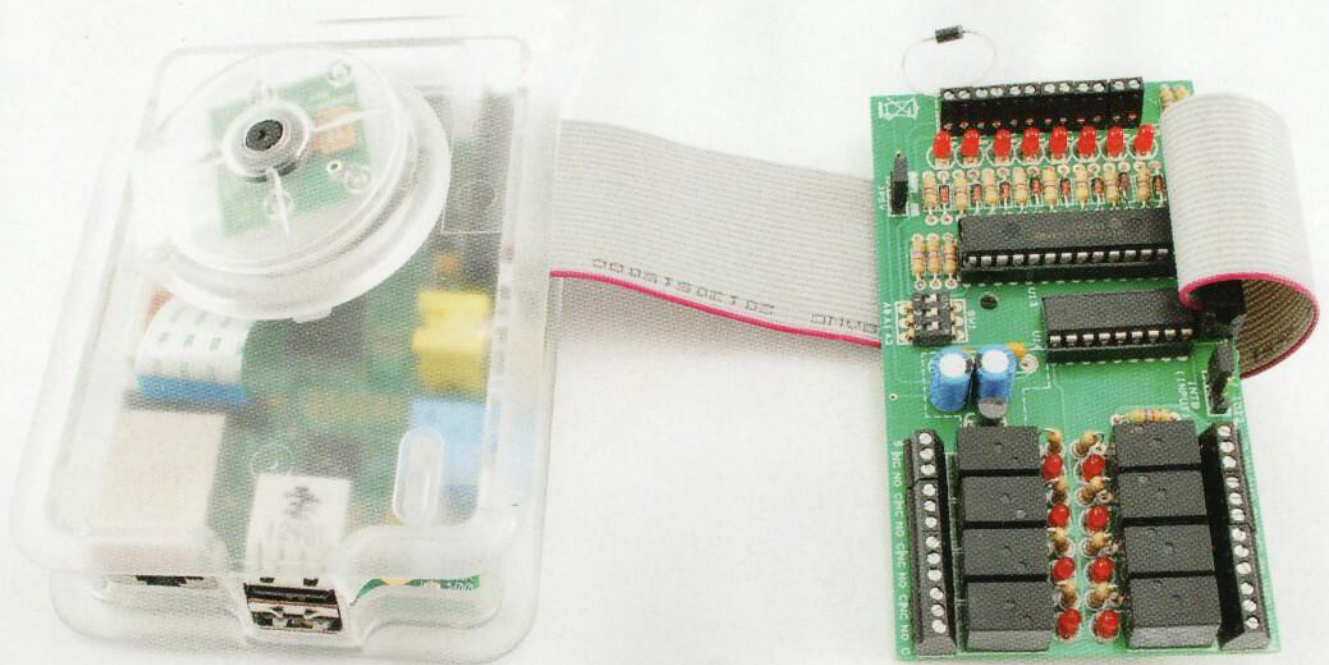
En figure 20, vous pouvez voir les zones identifiées par la fonction « **findKeypoints()** » et comparées avec celles de l'image qui contient le visage de référence.

Vous pouvez expérimenter les résultats de cette fonction en exécutant le programme du Listing 3 et en modifiant le nom du fichier. Testez-le avec les images « **password.jpg** » et « **passwordmatch.jpg** ».

Y a-t-il une correspondance ?

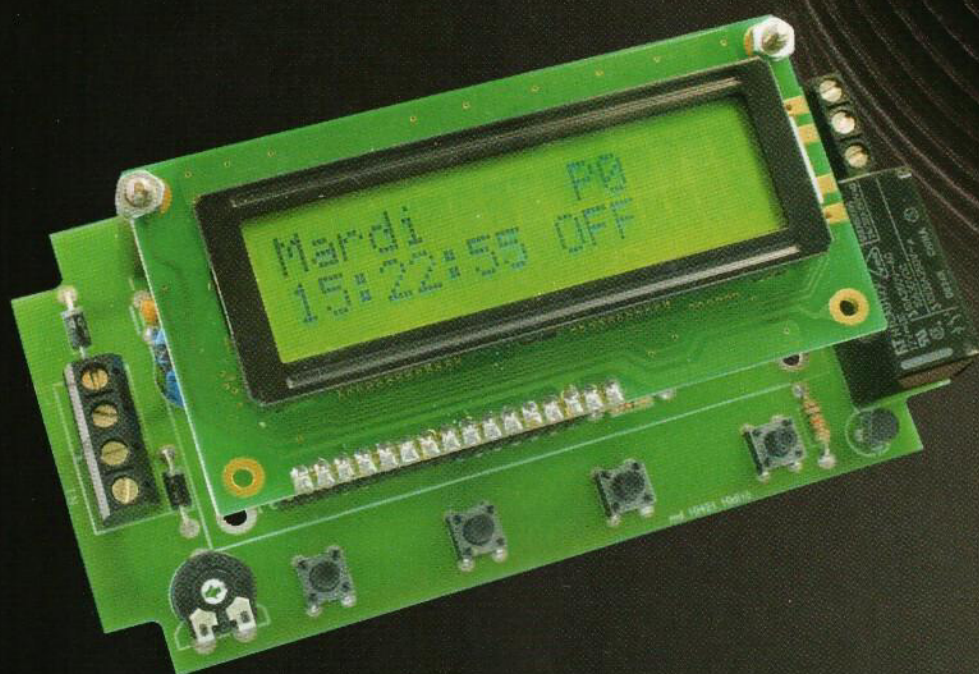
Nous espérons que cet article vous aura fait prendre conscience de la complexité du sujet lié à l'identification d'objets ou de zones du corps humain et combien leurs applications ont été simplifiées dans les programmes d'identification et de reconnaissance grâce à la recherche d'algorithmes et de méthodes de traitement de l'image et de leurs transpositions dans un logiciel.

En particulier, nous avons vu comment la librairie de SimpleCV permet d'implémenter un certain nombre de concepts avec très peu d'instructions, ce qui jusqu'à présent était réservé aux spécialistes. ■



Programmateur hebdomadaire

Dans cet article, nous vous proposons un temporisateur hebdomadaire disposant d'une sortie relais, d'un clavier et d'un afficheur LCD pour les réglages des différentes fonctions. Ce système est idéal pour commander des dispositifs fonctionnant sur batteries ou panneaux solaires de 12 V. Il comprend une entrée qui permet de forcer l'état du relais lorsqu'un événement externe se produit, cela est utile, comme par exemple, pour désactiver un arrosage automatique lorsqu'il pleut.



Il n'est pas rare de devoir contrôler des charges électriques comme des ampoules, des pompes, etc., en les faisant fonctionner pour des durées programmées qui peuvent varier selon le jour de la semaine.

Dans ce cas, il n'y a rien de plus simple que d'acheter une minuterie hebdomadaire disponible dans le commerce. Tout est simple avec ce type d'appareil, du moins tant que la tension du réseau est présente. Mais que se passe-t-il en cas de coupure de courant ?

Est-ce que les dispositifs sont alimentés en basse tension (par exemple 12 V) provenant de batteries ou de panneaux photovoltaïques ? Dans ce cas, vous ne trouverez pas facilement dans le commerce une minuterie fonctionnant en basse tension, à moins que vous y apportiez d'importantes modifications.

Afin de vous épargner cet effort, nous avons conçu un appareil fonctionnant en 12 VDC. Il s'agit d'une minuterie (ou programmeur) hebdomadaire capable d'exécuter un programme différent pour chacun des 7 jours de la semaine.

L'appareil est doté d'une sortie relais pouvant commuter des charges de 5 A, de 4 boutons pour paramétrer les différents programmes et gérer le fonctionnement (marche/arrêt manuel) à l'aide d'un afficheur LCD sur lequel sont visualisées les indications correspondantes aussi bien pendant la phase de programmation que de fonctionnement.

La gestion de l'ensemble est confiée à un microcontrôleur, il est donc possible d'implémenter des fonctions spéciales. Pour chaque jour de la semaine, il est possible de définir jusqu'à 3 programmes, c'est-à-dire 3 cycles d'activation/désactivation du relais et donc de la charge à gérer.

Qu'est-ce que cela signifie ? Eh bien c'est très simple, par exemple pour chaque jour nous pouvons activer la charge de 10 h 00 à 10 h 30, puis de nouveau de 14 h 00 à 14 h 30, puis encore de 20 h 00 à 20 h 20. Nous pouvons ainsi configurer l'activation de manière différente pour chaque jour de la semaine.

En plus du contrôle automatique propre au programmeur, nous avons ajouté une commande externe et une commande manuelle.

Dans le premier cas, au moyen de l'entrée (IN) active à un niveau logique 0 (mise à la masse), nous imposons un événement qui, en fonction de la configuration souhaitée, peut activer ou désactiver le relais. En d'autres termes, dans le menu de configuration, nous avons prévu un paramètre qui affecte la façon dont le relais se comporte lors de l'activation de l'entrée de commande externe (IN).

Dans le cas de la commande manuelle, cette dernière constitue une exception au programme paramétré et vous permet de définir l'activation/désactivation forcée du relais dans un certain intervalle du jour en cours. L'exception ne se répète pas, dans le sens où si vous l'avez définie pour le lundi, le lundi suivant elle ne sera pas exécutée.

de Claudio Visconti

Schéma électrique

Analysons maintenant la structure du schéma électrique, elle est relativement simple car le composant principal est un microcontrôleur **PIC16F876A**. Celui-ci gère les différentes fonctions, les programmes de la minuterie, l'entrée de commande externe, les boutons avec lesquels l'utilisateur configure les différents programmes et commandes et, bien entendu, le microcontrôleur visualise les différentes configurations sur un afficheur LCD. En fonctionnement normal, l'écran affiche les jours de la semaine et l'heure.

Pour épauler le microcontrôleur, qui seul ne peut malheureusement pas tout faire, nous avons ajouté un module **RTC ET674 (U2)** que nous avons décrit dans le numéro **102 d'Electronique et Loisirs Magazine** (référez-vous à cet article pour les explications détaillées du module RTC). Ce module fournit et mémorise l'horodatage en temps réel (calendrier) en le communiquant au microcontrôleur lorsque celui-ci en a besoin.

Le module ET674 communique avec le PIC par l'intermédiaire de deux lignes (full-duplex), correspondant aux broches 5 et 6 de U2, qui sont reliées respectivement aux broches 17 et 18 du PIC16F876A.

D'une surface de quelques centimètres carrés avec une consommation négligeable, ce module RTC génère l'heure et la date du système et les maintient également en l'absence d'alimentation, grâce à un système de sauvegarde interne doté d'une pile bouton au lithium qui mémorise les données d'horodatage lorsque le circuit est hors tension.

La communication avec le microcontrôleur s'effectue en mode esclave par l'intermédiaire d'un port série configuré de la manière suivante : 9600 bps, pas de parité, 8 bits de données, 1 bit d'arrêt.

L'afficheur LCD adopté dans ce projet est un modèle classique à 2 lignes de 16 caractères. Son contrôleur est de type Hitachi HD44780 ou compatible. Pour le gérer, le microcontrôleur utilise les broches **RC0, RC1, RC2, RC3 configurées en sorties** pour l'envoi des données et les broches **RC4 et RC5**, également

Caractéristiques techniques :

- Tension d'alimentation : **12 VDC**
- Courant absorbé : **80 mA**
- Tension maximale commutable en sortie : **250 VCA**
- Courant commutable en sortie : **5 A**
- Nombre maximal de programmes par jour : **3**
- Périodicité : **hebdomadaire**

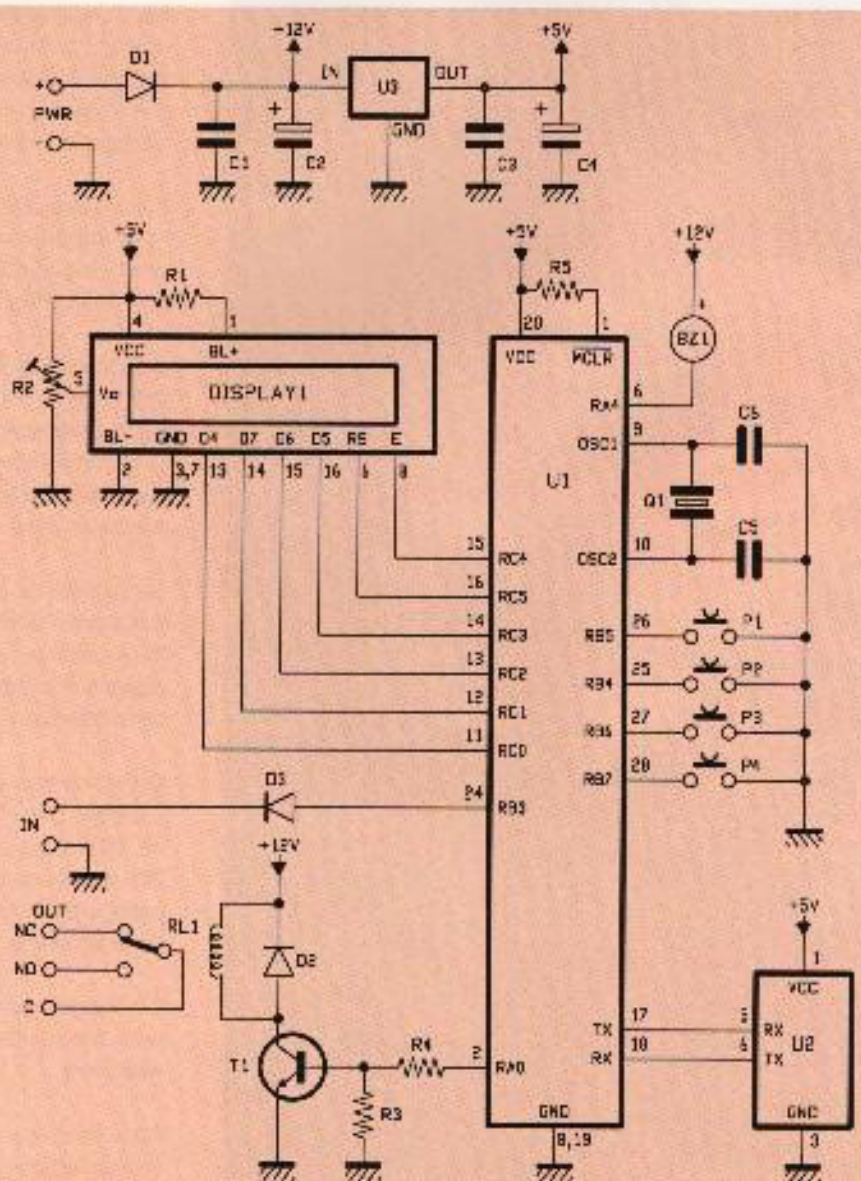


Schéma électrique du programmeur hebdomadaire.

configurées en sorties, réservées au mode de fonctionnement de l'afficheur (broches « E » et « RS » du LCD).

Avant de procéder à l'analyse du montage, attardons-nous quelques instants sur le fonctionnement de l'afficheur. La logique de communication utilise un bus de données de 4 bits ou de 8 bits (dans

notre cas la communication avec l'afficheur s'effectue sur 4 bits) et 3 lignes de commandes « R/W », « R » et « E ».

La première « R/W » détermine le sens de transfert des données, c'est-à-dire si nous souhaitons recevoir des données du microcontrôleur ou si nous voulons lui en envoyer.

Du grain de sable à l'atome

Le temps et la mesure du temps sont deux problèmes qui accompagnent la vie. Le plus ancien instrument permettant la mesure du temps, que nous utilisons encore aujourd'hui, est le sablier. C'est un instrument qui permet de mesurer un intervalle de temps correspondant à la durée d'écoulement d'une quantité calibrée de « sable », à l'intérieur d'un récipient transparent. À l'origine, il était constitué de deux bulbes ou ampoules de verre placés l'un sur l'autre et reliés par un tuyau fin ou par un diaphragme, l'ensemble étant scellé avec de la cire. Les techniques de fabrication du verre ont permis par la suite de les réaliser d'une seule pièce.

Un sablier, contrairement à ce que pourrait laisser penser son nom, ne contient pas un sable ordinaire mais un mélange de poudre de marbre calciné, de coquilles d'œufs et de plomb ou de zinc.

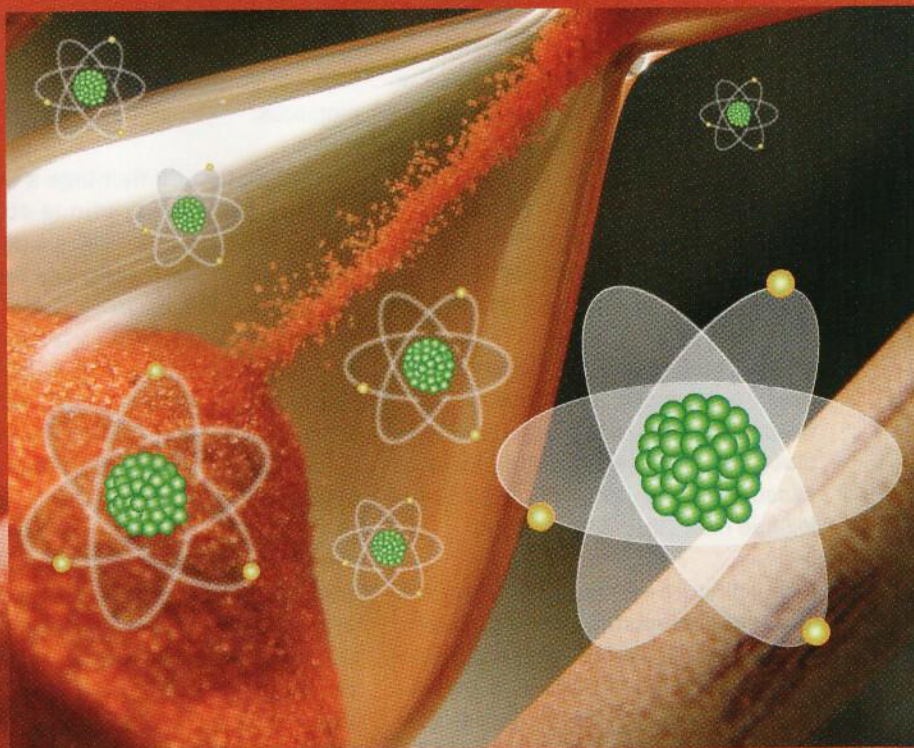
Pour commencer à décompter le temps, le sablier doit être renversé, de sorte que l'écoulement du sable s'effectue de la partie supérieure vers la partie inférieure. Le diamètre de la section de communication entre les deux parties ainsi que la quantité de sable sont calibrés de telle sorte que la partie supérieure se vide dans la partie inférieure selon une plage de temps bien définie.

Étant donné que le sablier était lourd et qu'il ne pouvait pas être porté au poignet (car pour compter le temps avec exactitude il faut qu'il soit positionné verticalement), l'homme a inventé l'horloge mécanique qui permettait de compter le temps à l'aide d'un ressort spiral et d'une série d'engrenages. Dans les horloges traditionnelles, le ressort fait osciller un balancier disposé sur un axe. La tension du ressort spiral détermine la vitesse du battement (tic-tac).

Cependant, les systèmes mécaniques ne garantissent pas une grande précision, l'horloge électronique ou à quartz a été inventée. Elle est beaucoup plus précise que celles mécaniques, car son fonctionnement est basé sur un oscillateur stabilisé par un quartz.

Par la suite, les systèmes complexes comme le GPS ont nécessité des horloges d'une plus grande précision. C'est pourquoi les horloges atomiques ont été fabriquées. Leur fonctionnement est basé sur l'activité atomique, c'est-à-dire le rayonnement provenant d'un atome, qui par définition est stable pour un atome donné depuis la création de l'univers.

Une horloge atomique est donc une horloge qui utilise la fréquence du rayonnement électromagnétique émis par un électron lors du passage d'un niveau d'énergie à un autre pour assurer l'exactitude et la stabilité du signal oscillant qu'elle produit. Un de ses principaux usages est le maintien du « Temps Atomique International » (TAI) et la distribution du « Temps Universel Coordonné » (UTC) qui sont les échelles de temps de référence. La précision est de l'ordre d'une seconde pour 160 millions d'années.



Pour notre application, nous devons simplement visualiser les données provenant du microcontrôleur.

Nous disposons donc la broche 7 de l'afficheur LCD à la masse, ce qui correspond à un état logique bas (0), c'est le mode de fonctionnement « Write » (écriture sur l'afficheur).

L'afficheur reçoit simplement les informations et son tampon de mémoire est utilisé pour visualiser les données à afficher. Le mode de commande de l'afficheur LCD permet de définir la position de chaque caractère et la structure de ce dernier. Autrement dit, pour afficher un symbole, nous devons toujours envoyer une paire de données indiquant

en premier la position où le caractère doit être affiché et en second le contenu du caractère à afficher.

Pour cela, la ligne **RS** indique à l'afficheur si les données arrivant sur le bus doivent être interprétées comme des **commandes** ou des **données** (caractères) à afficher.

Une commande correspond à un octet qui déplace le curseur ou réinitialise l'écran, tandis que les données sont, bien entendu, les caractères à afficher.

Le microcontrôleur met la ligne RS à un niveau haut lorsqu'il envoie sur le bus (RC0 à RC3) des impulsions relatives aux instructions que l'afficheur doit effectuer (par exemple l'avancement du curseur ou la position à laquelle il doit écrire) et à un niveau logique bas (0) lorsqu'il envoie des caractères (données) à afficher.

La broche **E** pour « **Entrée de validation** » correspond à « **Enable** » et est utilisée pour **activer l'écran** si elle est placée à un niveau logique bas ou le désactiver si elle est placée à un niveau logique haut (1).

Avant d'écrire un ou plusieurs caractères, le microcontrôleur remet à zéro le contrôleur de l'afficheur avec l'instruction (commande) « **0000** ». Elle envoie 4 bits sur les lignes D4 à D7 (bus de données) de l'afficheur après avoir mis à 0 la broche « **E** » pour activer l'écran et à 1 la broche « **RS** » de manière à indiquer à l'afficheur que les données qui arrivent sur le bus doivent être interprétées comme une commande de réinitialisation et par conséquent ne doivent pas être affichées. Ensuite, le microcontrôleur désactive la ligne « **E** » en la mettant à un niveau logique 1, tout en laissant les 4 bits inchangés.

Une fois la commande de reset effectuée, le microcontrôleur envoie, l'une après l'autre, les paires d'instructions qui déplacent le curseur (une impulsion sur la broche « **RS** » indique au driver HD44780 que les instructions qu'il reçoit doivent être interprétées comme des commandes) et qui écrivent les caractères (dans ce cas « **RS** » est mise à 0, et donc le contrôleur interprète les données qu'il reçoit comme des caractères à afficher). Comme cela est requis, chaque instruction est scrutée par l'impulsion d'activation sur la ligne « **E** ».

En plus des broches décrites précédemment, l'afficheur en comporte 6 autres dont 3 sont utilisées pour l'alimentation (+5V et 2 masses GND), une autre « **Vo** » est polarisée par un potentiomètre alimenté en 5 V et qui permet d'ajuster le **contraste de l'afficheur**.

Enfin, les 2 dernières broches sont réservées au **rétroéclairage**. La broche positive correspond à « **BL+** » (5V) et la borne négative à « **BL-** ». Le rétroéclairage permet de voir les caractères de l'écran LCD dans des conditions lumineuses sombres.

L'alimentation du montage s'effectue avec une tension continue comprise entre 11 VDC et 15 VDC. La diode **D1** **protège le circuit contre une inversion de polarité**, la tension « +12 V » permet d'alimenter la bobine du relais. Le microcontrôleur, le module RTC et l'afficheur LCD fonctionnent en 5 VDC, cette tension stabilisée est fournie par un régulateur 7805 à partir de la tension +12V présente sur la cathode de D1.

Les broches **RB4**, **RB5**, **RB6** et **RB7** du microcontrôleur sont **configurées en entrées** avec chacune leur résistance de **pull-up activée**, afin de pouvoir lire l'état des 4 boutons poussoirs. La broche **RA0** est **configurée en sortie** et pilote le transistor **T1**. Ce dernier, alimenté par son collecteur à travers la bobine du relais, provoque la fermeture du contact du relais lorsque RAO est à un niveau haut.

La broche **RB3** est **configurée en entrée** avec sa résistance interne de **pull-up activée**. Elle détecte la commande externe qui, selon le réglage, force l'activation ou la désactivation du relais RL1 selon que RAO est respectivement à un niveau logique haut ou bas.

La commande externe peut provenir d'un contact à sec (interrupteur, bouton, relais), mais aussi d'un transistor NPN connecté en collecteur ouvert.

Enfin, la broche **RA4** est **configurée en sortie** pour piloter le buzzer « **BZ1** » en mode « **sink** ». Le buzzer assiste les procédures de programmation et répète acoustiquement le début et la fin des intervalles de temps.

Le fonctionnement du montage

Le programme qui s'exécute dans le microcontrôleur fait en sorte qu'après l'initialisation des entrées/sorties, le PIC interroge le module RTC puis commande l'afficheur pour visualiser l'heure et le jour de la semaine actuelle (cette information provient du module U2). Puis ensuite, le programme lit cycliquement l'état de chacun des 4 boutons pour tester si l'un d'entre eux est pressé.

Pendant l'exécution d'un programme de temporisation, une routine gère en boucle l'état de l'entrée « **IN** » (ligne RB3), afin de vérifier si elle est forcée par une commande externe. Si une telle éventualité se produit, le PIC force l'état du relais de sortie comme défini dans la configuration (par défaut le relais est activé, dans le sens où si une commande externe est détectée, le relais est activé de force).

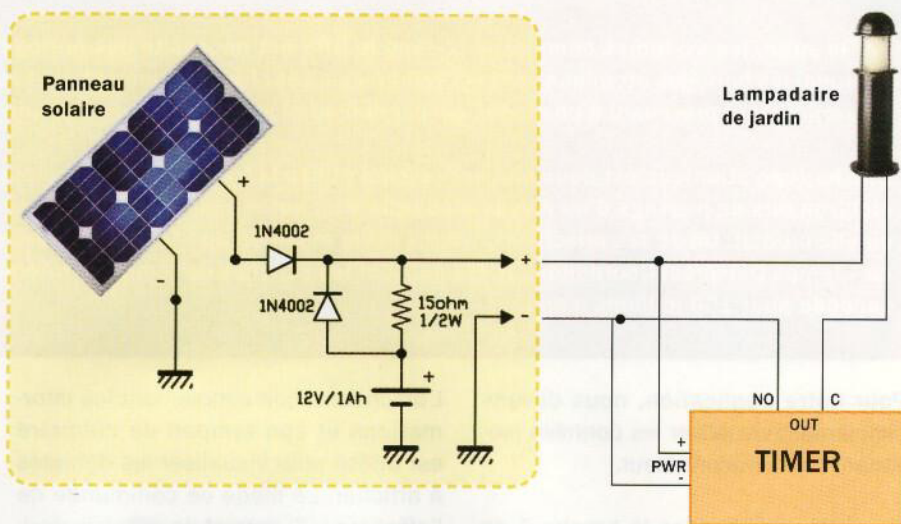
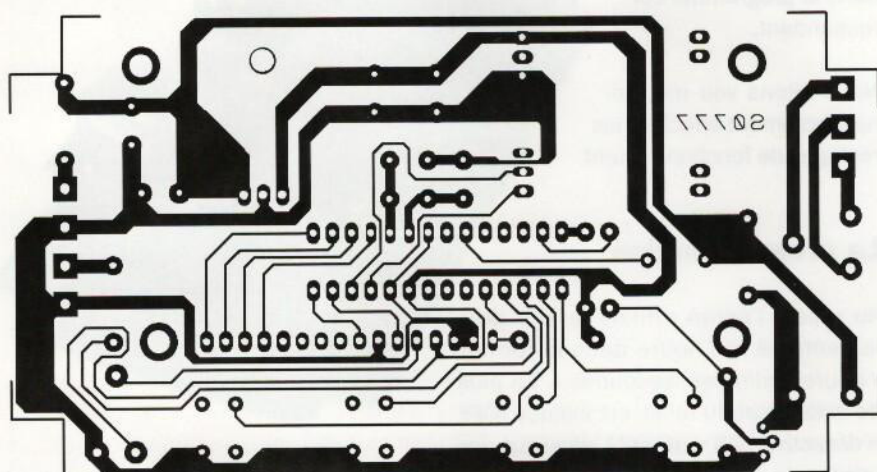
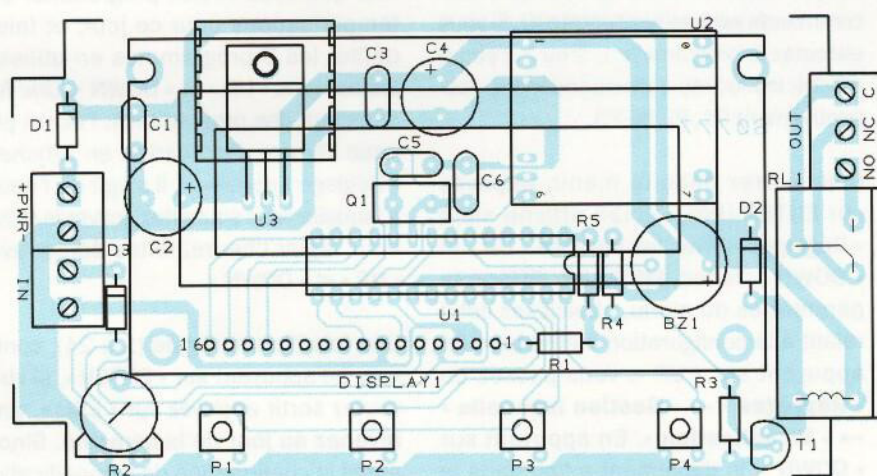


Figure 1 : utilisation du programmeur hebdomadaire pour le contrôle d'un lampadaire de jardin fonctionnant en basse tension. L'alimentation est prélevée à partir d'une batterie chargée au moyen d'un panneau solaire qui alimente le programmeur. Ce dernier permet d'allumer le lampadaire en fonction du réglage du programme hebdomadaire.

Plan de montage du programmeur hebdomadaire



Circuit imprimé à l'échelle 1 : 1 côté soudures du programmeur hebdomadaire.



Plan de câblage des composants du programmeur hebdomadaire.

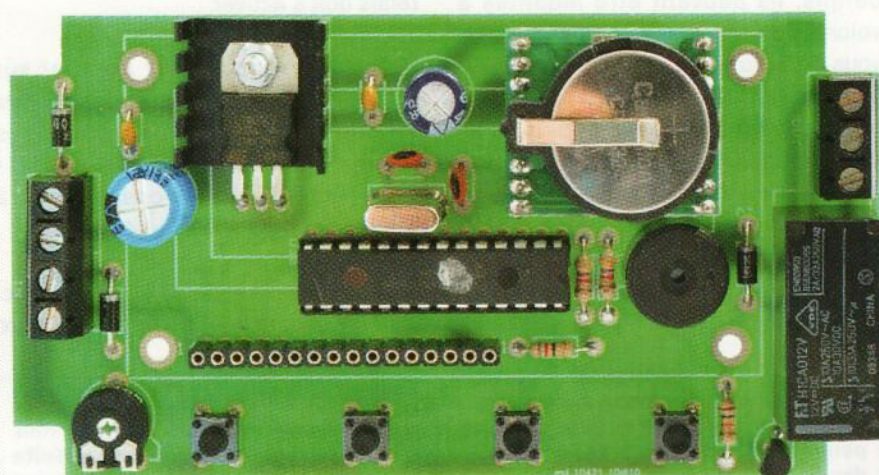


Photo de l'un de nos prototypes du programmeur hebdomadaire.

Liste des composants

R1..... 82 Ω
R2..... trimmer monotour 4,7 k Ω
R3..... 10 k Ω
R4..... 4,7 k Ω
R5..... 4,7 k Ω

C1..... 100 nF multicouche
C2..... 470 μ F/25 V électrolytique
C3..... 100 nF multicouche
C4..... 470 μ F/16 V électrolytique
C5..... 22 pF céramique
C6..... 22 pF céramique

U1..... PIC16F876A
U2..... module RTC (ET674)
U3..... 7805

D1..... 1N4007
D2..... 1N4007
D3..... 1N4007

Q1..... quartz 8 MHz

T1 BC547

RL1.... relais 12 V FTR-H1CA012V
BZ1.... buzzer 12V électronique
intégrée

P1 à P4 microswitch

DISPLAY1 afficheur LCD 16
caractères 2 lignes

Divers :

Bornier 2 pôles au pas de
2,54mm (x2)
Bornier 3 pôles au pas de
2,54mm
Support circuit intégré 2 x 14
broches
Barrette femelle 2 pôles (x2)
Barrette femelle 3 pôles (x2)
Barrette femelle 16 pôles
Barrette mâle 16 pôles
Dissipateur ML26 ou ML7
Vis 10 mm 3 MA
Ecroû 3 MA

Notez que **le relais reste dans l'état déterminé** par la configuration **aussi longtemps** que l'entrée « IN » **reste fermée**. Dès que le programme du microcontrôleur détecte un retour de l'entrée RB3 à un niveau logique 1, le relais est géré de manière normale, c'est-à-dire par les temporisations paramétrées par l'utilisateur.

Durant toutes les phases de fonctionnement, une autre routine interroge cycliquement le module RTC pour mettre à jour continuellement l'heure et la date du système.

Comme mentionné précédemment, il est possible d'effectuer **3 programmes par jour**, différents **pour chacun des 7 jours** de la semaine.

Chaque programme consiste à activer le relais de sortie pendant un certain laps de temps, puis de le désactiver.

Pendant la phase de programmation, les boutons suivants ont les fonctions suivantes :

- **P1 = DOWN** (« en arrière ») : décrémente d'un pas ou d'une minute le réglage de l'heure ou l'heure d'activation/désactivation en cours de programmation ;
- **P2 = UP** (« en avant ») : incrémente d'un pas ou d'une minute le réglage de l'heure ou l'heure d'activation/désactivation en cours de programmation ;
- **P3 = ESC** (sortie) : sort de la procédure ou de la commande du menu, ou retourne au menu précédent ;
- **P4 = ENTER** (validation) : confirmation des données entrées, et sortie de la programmation.

Chaque fois qu'une touche est pressée, le buzzer émet un bip court pour confirmer la pression.

Par conséquent, si vous appuyez sur un bouton et que vous n'entendez pas de bip, cela signifie que la fonction correspondante ne sera pas exécutée, vous devez appuyer de nouveau.

Lors de l'exécution d'un programme, il est possible d'exécuter une exception précédemment définie au cours de la journée et avant le début du programme.

Dans la pratique, vous pouvez changer l'état du relais de sortie par rapport à l'état défini dans le programme correspondant.

Nous allons voir maintenant comment effectuer les réglages de fonctionnement.

La programmation

Au repos, l'écran affiche le jour de la semaine et l'heure dans le format « heures, minutes, secondes ». En plus de cela, l'état du relais est indiqué (OFF = désactivé ; ON = activé), ainsi que les raisons de cet état. P0 indique qu'il n'y a pas de programme en cours d'exécution (veille). PM indique la présence d'une commande manuelle tandis que PI signifie que l'état du relais RL1 est dû à une commande externe sur l'entrée IN. Si vous exécutez le programme 1, 2 ou 3, l'écran affiche les lettres correspondantes, respectivement P1, P2 ou P3.

Pour entrer dans le menu, appuyez sur ENTER (P4), l'écran affiche alors « Réglages ». Avec les boutons « UP » et « DOWN », faites défiler les différents paramètres du menu, choisissez celui relatif à la configuration à effectuer. En appuyant sur « UP », vous passez de « Réglages » → « Gestion manuelle » → « Visualisation ». En appuyant sur « DOWN », le défilement a lieu dans le sens inverse (en arrière).

Dans le menu « Réglages », vous pouvez voir pour chaque jour les programmes définis, ils peuvent être modifiés à volonté. En appuyant une fois sur P4, vous accédez au réglage de la semaine (l'afficheur visualise « Réglage Semaine ») et en appuyant une fois de plus, vous accédez à la programmation du premier jour (l'écran affiche « Lundi »). Si vous voulez changer le jour à afficher ou à modifier, appuyez sur « UP » ou « DOWN » pour faire défiler « Lundi » → « Mardi » → « Mercredi », etc.



Pour modifier un jour particulier, lorsque l'afficheur indique son nom, appuyez sur P4. Le numéro de programme apparaît. Appuyez sur « ENTER » pour confirmer que vous voulez programmer des temporisations pour ce jour, et faites défiler les 3 programmes en utilisant les boutons « UP » et « DOWN ». Une fois dans l'un des programmes, l'écran propose l'heure d'activation en affichant « Réglage heure → », il s'agit de l'heure à laquelle vous souhaitez activer le relais. Pour régler l'heure, faites défiler avec « UP » et « DOWN ».

Une fois l'heure réglée (1 à 24), confirmez en appuyant sur « ENTER », si vous voulez sortir appuyez sur « ESC », vous revenez au jour de la semaine. Sinon, après la confirmation de la modification du réglage de l'heure par l'appui sur « ENTRER », l'écran affiche « Réglage minute → », cela signifie que vous êtes invité à définir la minute à laquelle le relais doit s'activer.

Notez que vous pouvez appuyer sur « ENTER », même lorsque vous vérifiez un programme, sans l'affecter. Vous pouvez quitter en appuyant sur « ESC » deux fois (une fois, vous revenez à « Réglage heure → »).

Après la confirmation de la minute d'activation, vous devez programmer l'heure

Listing 1 : ci-après, nous publions un extrait du programme du microcontrôleur concernant la routine de gestion du réglage de l'heure et de la date. Comme vous pouvez le remarquer, il est demandé en premier le réglage de l'heure, ensuite des minutes, ensuite du jour de la semaine en cours et enfin lorsque les données sont correctement insérées, les réglages sont enregistrés dans le module RTC de façon permanente.

Listing 1

ORARIO:

```

SELECT CASE REGOLA_ORA_MENU
CASE 0
  LCDout $FE,$01," Heures"
  LCDout $FE,$C0," ",DEC2 HOR1,":XX"
  SELECT CASE PULSANTE
  CASE 2
    IF HOR1<23 THEN
      HOR1=HOR1+1
    ELSE
      HOR1=0
    ENDIF
  CASE 1
    IF HOR1>0 THEN
      HOR1=HOR1-1
    ELSE
      HOR1=23
    ENDIF
  CASE 3
    REGOLA_ORA_MENU=1
  CASE 4
    MENU=32
  END SELECT
CASE 1
  LCDout $FE,$01," Minutes"
  LCDout $FE,$C0," XX:",DEC2 MI1
  SELECT CASE PULSANTE
  CASE 2
    IF MI1<59 THEN
      MI1=MI1+1
    ELSE
      MI1=0
    ENDIF
  CASE 1
    IF MI1>0 THEN
      MI1=MI1-1
    ELSE
      MI1=59
    ENDIF
  CASE 3
    REGOLA_ORA_MENU=2
  CASE 4
    REGOLA_ORA_MENU=0
  END SELECT
CASE 2
  LCDout $FE,$01," Jour"
  SELECT CASE SETT
  CASE 1
    LCDout $FE,$C0," Lundi"
  CASE 2
    LCDout $FE,$C0," Mardi"
  CASE 3
    LCDout $FE,$C0," Mercredi"
  CASE 4

```

```

    LCDout $FE,$C0," Jeudi"
  CASE 5
    LCDout $FE,$C0," Vendredi"
  CASE 6
    LCDout $FE,$C0," Samedi"
  CASE 7
    LCDout $FE,$C0," Dimanche"
  END SELECT

```

```

SELECT CASE PULSANTE
CASE 2
  IF sett<7 THEN
    sett=sett+1
  ELSE
    sett=1
  ENDIF
CASE 1
  IF sett>1 THEN
    sett=sett-1
  ELSE
    sett=7
  ENDIF
CASE 3
  REGOLA_ORA_MENU=3
CASE 4
  REGOLA_ORA_MENU=0
END SELECT

```

```

CASE 3
  LCDout $FE,$01," Sauvegarde"
  SEC=0
  SEROUT TXCLOCK,2,["*W3"]
  PAUSE 500
  SEROUT TXCLOCK,2,["0",#sett]
  pause 500
  SEROUT TXCLOCK,2,["*W2"]
  PAUSE 500
  IF HOR1<=9 THEN
    SEROUT TXCLOCK,2,["0"]
  ENDIF
  SEROUT TXCLOCK,2, [#HOR1]
  PAUSE 500
  SEROUT TXCLOCK,2, ["*W1"]
  PAUSE 500
  IF MI1<=9 THEN
    SEROUT TXCLOCK,2, ["0"]
  ENDIF
  SEROUT TXCLOCK,2, [#MI1]
  PAUSE 500
  SEROUT TXCLOCK,2, ["*W0"]
  PAUSE 500
  SEROUT TXCLOCK,2, ["00"]
  MENU=0
END SELECT

```

RETURN



et la minute d'arrêt (désactivation du relais). Cela s'effectue d'une manière similaire à celle qui vient d'être décrite. L'écran affiche en premier « Réglage heure ← » et « Réglage minute ← ».

Chaque fois que vous appuyez sur « ENTER », lorsque l'écran affiche « Réglage minute → » ou « Réglage minute ← », la mémorisation des heures et des minutes est effective, ce qui est confirmé par l'apparition de « Sauvegarde » pendant quelques secondes.

Une fois les réglages de l'heure et de la minute de désactivation (OFF) effectués, le microcontrôleur sort de la procédure de réglage et affiche l'heure, l'état du relais, le jour de la semaine et l'état de fonctionnement (PO, PI, PM ou P1, P2, P3).

Pour plus de clarté, notez que le symbole « → » signifie le début de la temporisation (**activation du relais**), et que le symbole « ← » correspond à la fin de la temporisation (**désactivation du relais**), c'est-à-dire la fin du programme en cours.

Pour effectuer un nouveau réglage sur un autre programme ou un autre jour, vous devez refaire les mêmes étapes que précédemment. Appuyez sur « ENTER » pour entrer dans les réglages, puis appuyez de nouveau jusqu'à ce que les paramètres de la semaine apparaissent (« Réglage Semaine »), puis appuyez une troisième fois pour afficher « Lundi ».

À ce stade, à l'aide des boutons « UP » et « DOWN », choisissez un jour à votre convenance.

Lorsque l'écran affiche le jour désiré, appuyez sur « ENTER », vous devez voir la liste des programmes et l'écran affiche « Programme N1 ». Si vous voulez afficher ou modifier le « Programme N1 », appuyez sur « ENTER », à l'aide des boutons « UP » et « DOWN » faites défiler jusqu'au programme désiré, puis appuyez sur « ENTER » pour accéder aux réglages du programme (heure et minute de « ON » et de « OFF » du relais).

Le menu « Réglages » comprend deux autres sections qui sont des « Réglage heure/jour » et « Réglage d'entrée ». Elles sont accessibles à l'aide de « UP » ou « DOWN », lorsque l'écran visualise le menu « Réglages ». La première permet (en appuyant sur « ENTER » et « UP »/« DOWN ») de régler l'heure du système, tandis que la seconde concerne la gestion de l'entrée de commande externe.

En accédant à cette dernière (appuyez sur « ENTER » lorsque l'écran affiche « Réglage d'entrée »), vous pouvez choisir, à l'aide des boutons « UP » et « DOWN », le mode de fonctionnement de la commande externe, c'est-à-dire si le relais doit être activé ou au repos lorsque qu'un évènement externe se produit.

La première éventualité se produit si l'écran affiche « Actif » et que vous confirmez, tandis que la seconde éventualité est obtenue en confirmant le message « Repos ». La confirmation est effectuée avec une pression sur la touche « ENTER », la procédure de réglage est alors automatiquement fermée et l'affichage revient à l'indication du jour, de l'heure, etc.

Examinons maintenant les fonctions accessibles depuis le menu « Visualisation ». Il s'agit d'une option qui résume, jour par jour, les paramètres actuellement mémorisés pour chaque jour. En appuyant sur « ENTER », vous accédez aux jours et l'afficheur indique « Lundi ».

À ce stade, pour visualiser les réglages du jour « Lundi », appuyez sur « ENTER ». Si vous voulez afficher les réglages d'un autre jour, appuyez sur les boutons « UP » ou « DOWN ». Une fois le jour sélectionné, appuyez sur « ENTER » pour accéder aux réglages. L'écran affiche « Prog 1 », ce qui indique qu'une nouvelle pression sur « ENTER » permet d'accéder à l'affichage du premier programme du jour en question. Pour sélectionner un autre programme, comme d'habitude utilisez les boutons « UP » ou « DOWN ».

Pour visualiser les réglages d'un programme, appuyez sur « ENTER » de sorte que l'écran affiche, dans l'ordre, l'heure de démarrage (ON) et l'heure d'arrêt (OFF). En appuyant une nouvelle fois sur « ENTER », vous revenez à l'écran d'accueil (jour, heure, état du relais, etc.).

Abordons, enfin, la fonction de « Gestion manuelle », qui permet de définir un programme forcé, constituant de fait une exception à la programmation en vigueur. Cette fonction est exécutable une seule fois pour le jour en cours. En d'autres termes, la fonction de « Gestion manuelle » permet de forcer à l'état actif la sortie du programmeur hebdomadaire indépendamment de l'ensemble des programmes mémorisés dans la phase de réglage.

Sélectionnez cette fonction en faisant apparaître sur l'afficheur « Gestion manuelle ». Appuyez sur « ENTER » de nouveau pour visualiser le message « Actif ». Ensuite, vous pouvez décider de l'état du relais enclenché ou au repos. La sélection s'effectue à l'aide des boutons « UP » ou « DOWN », qui inversent la sélection à chaque pression.

Choisissez ce que le programmeur doit faire, confirmez le message « Actif », une pression supplémentaire sur « ENTER » permet de passer au réglage de l'heure « Réglage heure → ». Avec les boutons « UP » ou « DOWN » réglez l'heure d'activation du relais. Si au contraire, vous avez sélectionné le message « Inactif », le message « Réglage heure ← » apparaît.

Après avoir défini l'heure avec la touche « ENTER », passez au réglage des minutes (« Réglage minute → » ou « Réglage minute ← ») selon que vous avez choisi « Actif » ou « Inactif ».

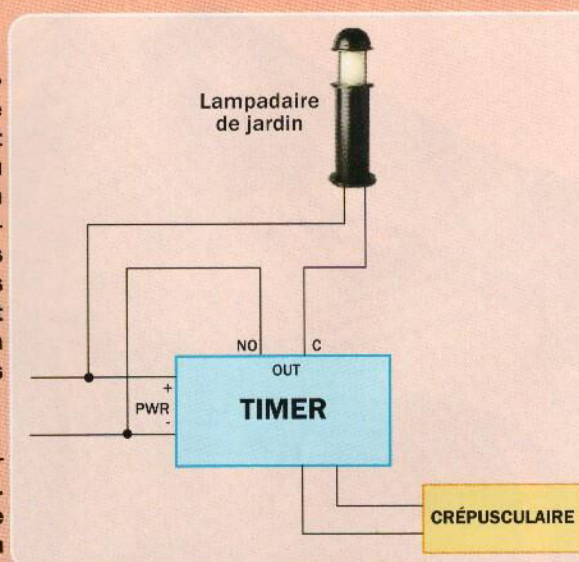
La commande externe

Notre programmeur hebdomadaire est doté d'une entrée « IN » pour la commande externe du relais. En mettant cette entrée à la masse à l'aide d'un commutateur ou d'un transistor, l'état du relais RL1 est forcé selon la condition définie dans le menu de configuration, indépendamment du fait que le circuit est en train d'exécuter un programme ou est au repos et du type de programme qui peut être en cours d'exécution. Ainsi, nous pouvons déterminer en fonction d'une condition sur l'entrée si le relais doit être activé ou désactivé. Étant entendu que, si au moment de la commande, RL1 est actif, l'ordre d'activation ne changera rien. Il en est de même pour un ordre de désactivation si le relais est déjà au repos.

L'entrée de commande externe est utilisée pour certaines applications, telles que, par exemple, les systèmes d'irrigation des jardins. Dans ce cas, l'entrée « IN » peut être commandée par un capteur de pluie ou un interrupteur crépusculaire, et en combinant l'activation de l'entrée (IN à la masse) avec une désactivation forcée du relais.

De cette façon, il est possible d'arrêter l'irrigation lorsqu'il pleut (car cela est inutile) ou lorsqu'il fait jour (il est préférable d'arroser le soir car le sol absorbe mieux l'eau et il y a moins d'évaporation puisque le soleil est pratiquement couché).

L'entrée peut servir aussi dans le cas où le programmeur commande des éclairages basse tension aux abords de l'entrée d'une maison. En combinant l'entrée « IN » avec un interrupteur crépusculaire et en réglant l'activation du relais en fonction de l'entrée, il est possible d'allumer l'éclairage en cas de ciel sombre en forçant l'état du relais indépendamment des plages horaires programmées.



En appuyant encore sur « ENTER », l'affichage indique « Sauvegarde » et passe ensuite à la définition de l'heure d'allumage (si vous travaillez sur « Inactif ») et d'extinction (si vous travaillez sur « Actif »). Rappelons que l'option « Actif » détermine, indépendamment de la programmation effectuée, l'heure et la minute d'activation du relais, tandis que « Inactif » détermine l'heure et la minute de désactivation du relais.

Réalisation pratique

Il est temps de passer à la construction de notre programmeur hebdomadaire. Tous les composants tiennent sur un seul circuit imprimé simple face, facilement réalisable par photogravure classique.

Vous pouvez télécharger le typon du circuit ainsi que les fichiers Gerber pour une fabrication professionnelle sur notre site www.electroniquemagazine.com dans le sommaire détaillé de la revue.

Une fois le circuit imprimé gravé et percé, commencez par souder comme d'habitude les composants ayant un profil bas : les résistances, les diodes en respectant l'orientation de la bague, le support du PIC, les barrettes femelles pour le module

RTC et l'afficheur, les condensateurs non polarisés. Ensuite, continuez en soudant le trimmer, les boutons poussoirs, le transistor (son méplat est face à R3), le buzzer, les condensateurs polarisés C2 et C4 en respectant la polarité (la patte la plus longue correspond au positif). Soudez le quartz.

Pliez à 90°, à environ 5 mm, les broches du régulateur. Positionnez-le à l'aide d'une vis sur un radiateur de résistance thermique 18 à 20 °C/W. Insérez les pattes du régulateur dans les trous correspondant du circuit imprimé et plaquez le radiateur contre le circuit. Vissez l'ensemble « radiateur + régulateur » sur le circuit imprimé, puis soudez les pattes du régulateur. Enfin, finissez en soudant les borniers et le relais. Disposez le module RTC sur son emplacement, en respectant l'orientation (aidez-vous du plan de câblage et des photos).

Programmez le PIC et insérez-le dans son support. Notez que le code source du programme et le fichier « .hex » sont disponibles en téléchargement avec le typon et les fichiers Gerber.

L'afficheur doit être fixé au circuit imprimé à l'aide de 4 entretoises de 25 mm de hauteur.

Engagez d'abord les contacts de l'afficheur dans ceux de la barrette femelle à 16 contacts disposée sur le circuit imprimé, puis vissez les entretoises sans forcer.

Une fois l'assemblage terminé et après vérification que tous les composants sont bien insérés correctement à leur place, vous pouvez choisir un boîtier en plastique de dimensions appropriées. Vous devez découper la zone d'affichage de l'écran LCD ainsi que percer des trous pour avoir accès aux boutons, sinon vous pouvez utiliser des boutons poussoirs déportés sur la face avant du boîtier, que vous vissez à l'aide d'écrous et de rondelles. Sur les côtés du boîtier, percez des trous pour le passage des câbles de l'alimentation, de l'entrée externe et des contacts du relais.

Pour l'alimentation, servez-vous d'une alimentation stabilisée dont la tension est comprise entre 11 VDC et 15 VDC avec un courant minimum de 100 mA.

Vous pouvez utiliser aussi une batterie de 12 V/10 Ah qui vous fournira au moins 5 jours d'autonomie. Pour une plus grande autonomie, supprimez le rétroéclairage de l'afficheur en enlevant la résistance R1.

RADIO FM & LECTEUR MP3 USB



Nous vous décrivons dans cet article un montage atypique, il s'agit d'un récepteur FM avec un port USB sur lequel vous pouvez relier un périphérique de stockage afin de lire vos chansons préférées au format MP3. Sa particularité réside dans le fait qu'il peut être intégré dans une voiture.

..... de Luca Pascarella

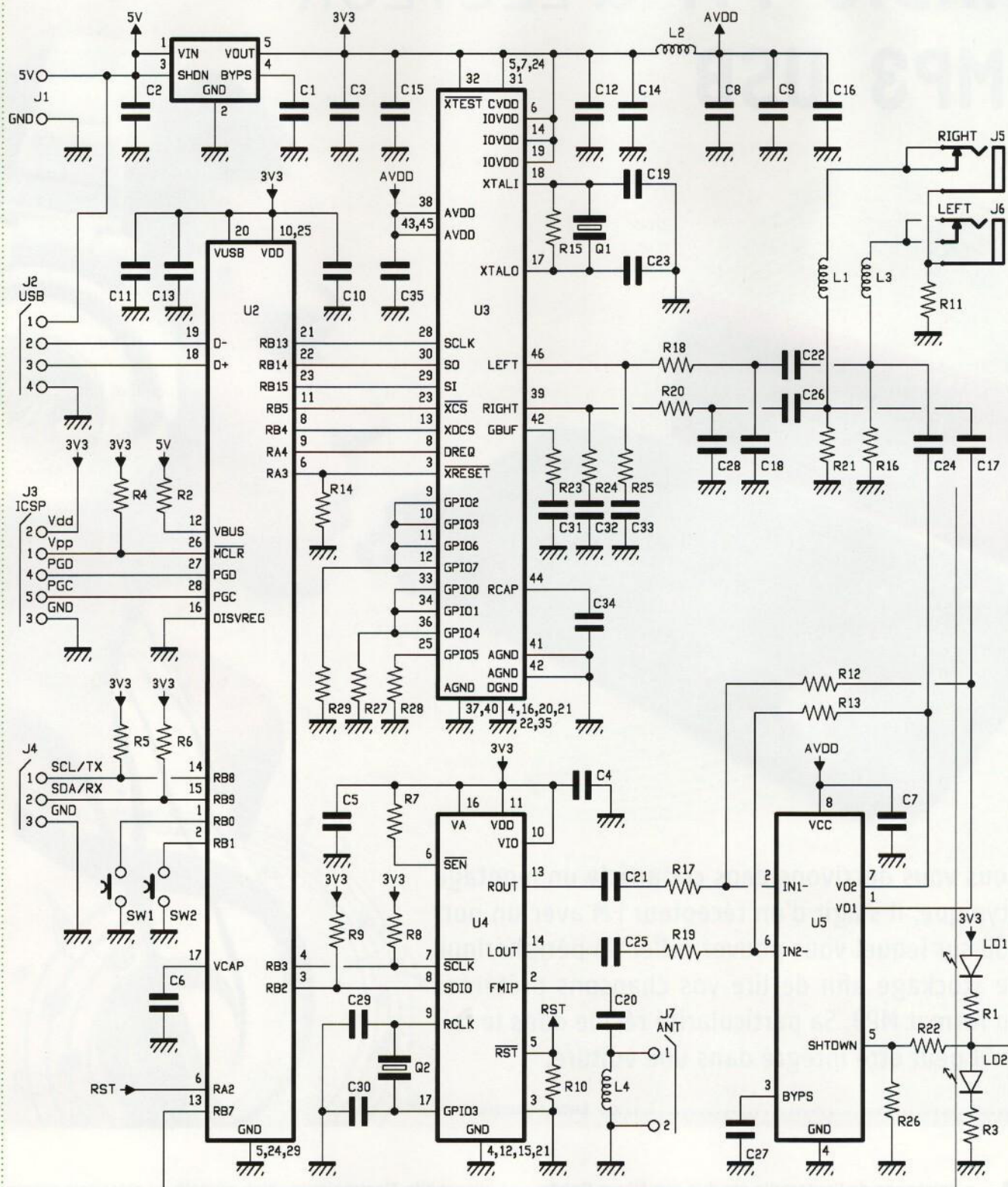
La naissance de l'autoradio est due aux frères **Galvin**, qui en 1928, fondent la « Manufacturing Corporation Galvin ». Au début, ils équipèrent les voitures de la police américaine, et plus tard l'autoradio (à lampes) trouve la faveur du public.

L'évolution de l'électronique et le concept d'autoradio sont toujours d'actualité, les ingénieurs dans le secteur de l'automobile intègrent dans l'habitacle un nombre croissant de

dispositifs électroniques, bien au-delà du classique autoradio. Il suffit de se pencher sur les systèmes de divertissement dont sont équipées les voitures modernes, elles intègrent des systèmes audio numériques ainsi que de véritables systèmes « Home Cinéma » avec plusieurs écrans.

Le dispositif que nous vous proposons dans ces pages va dans ce sens, car il combine une radio FM avec un lecteur de fichiers MP3 directement à partir d'une clé USB.

Figure A : schéma électrique du récepteur FM/lecteur MP3.



L'ensemble est extrêmement compact et peut être installé dans un véhicule, ou vous pouvez l'utiliser à la maison avec l'émetteur FM décrit dans le numéro 138 ou encore l'intégrer dans des systèmes compacts pour une reproduction sonore.

Le circuit peut fonctionner de manière autonome ou en tant que maître/esclave, dans le sens où il peut être utilisé seul ou gérer via le bus I²C. Cette dernière solution est optimale dans le cas d'une intégration dans un système audio d'un véhicule équipé d'une unité

maître qui gère les différentes fonctions du système, y compris le téléphone, la radio, la navigation, etc.

Dans les pages qui suivent, nous allons décrire le projet proposé en tant que point de départ pour la réalisation de

systèmes plus complexes, vous apprendrez les bases pour utiliser ce récepteur FM numérique.

Schéma électrique

Il est visible en figure A, le cœur du montage est le microcontrôleur U2, un **PIC24FJ64GB002** de chez **Microchip**. Celui-ci appartient à la famille des PIC24F, il est équipé d'un port USB hôte et peut exécuter le firmware chargé dans la mémoire Flash à une vitesse de 16 MIPS, grâce à une fréquence d'horloge de 32 MHz générée en interne.

En fait, le microcontrôleur dispose d'une horloge interne de 8 MHz et d'un multiplicateur de fréquence PLL (x4) interne qui permet d'obtenir une fréquence d'horloge de 32 MHz.

Ce microcontrôleur a été choisi car il dispose de nombreux périphériques matériels tels que le bus I²C, le bus SPI, l'UART, etc. De plus, il peut gérer des périphériques de stockage tels que des clés USB grâce à son hôte USB incorporé.

Ce PIC est disponible dans le commerce sous différents formats de boîtiers afin de permettre aux concepteurs de l'intégrer dans des systèmes de petites dimensions.

Le modèle que nous avons utilisé est le QFN (Quad Flat No-leads) de 28 broches (7 broches sur chacun des 4 côtés), de plus la broche (pad) centrale appelée « Thermal pad » doit être soudée sur le circuit imprimé afin de permettre la dissipation de chaleur du composant.

Le PIC24FJ64GB002 appartient à la catégorie des « XLP » (eXtreme Low Power) de Microchip, qui est adaptée aux applications fonctionnant sur batterie, car il consomme très peu d'énergie, surtout en veille (mode sleep).

Le microcontrôleur régit l'ensemble du montage à partir des commandes arrivant sur le bus I²C ou par les boutons SW1 et SW2 reliés directement à 2 entrées dotées de résistances de tirage (pull-up). Le PIC lit aussi les données à partir du périphérique de stockage et

les envoie au décodeur MP3, il contrôle également le fonctionnement du tuner FM.

Le circuit intégré U3, un **VS1053** du fabricant VLSI Solution, reçoit les données au format MP3 et génère un signal audio analogique capable de piloter un casque.

Pour fonctionner correctement, U3 nécessite une tension d'alimentation stabilisée de 3,3 V. Celle-ci est fournie par le régulateur de tension U1 (TC1015), à très faible chute de tension.

Continuons l'analyse du schéma avec le circuit U4, un **SI4703**. Il s'agit d'un composant majeur qui est un **récepteur complet avec syntoniseur FM** à commande numérique via un protocole série à 2 ou 3 lignes.

Dans notre cas, nous avons utilisé le protocole série sur 2 lignes, les broches 7 (SCLK) et 8 (SDIO) sont reliées au bus I²C du microcontrôleur U2. Pour fonctionner correctement, le bus I²C nécessite la présence de 2 résistances de tirage (pull-up) R8 et R9, chacune sur une ligne. Ces résistances sont essentielles, car elles permettent une communication bidirectionnelle sur la même connexion électrique.

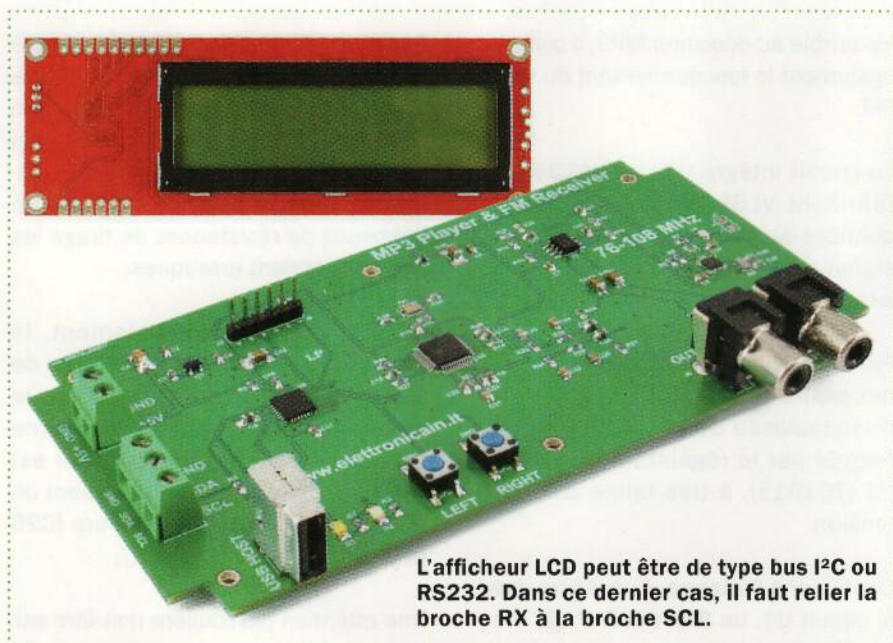
Par exemple, le signal SDA est généré alternativement par le périphérique Maître (MASTER) et par le périphérique esclave (SLAVE). Il est essentiel que les sorties des périphériques soient à collecteur ouvert (ou drain ouvert), car en l'absence de résistances de tirage les signaux seraient erratiques.

Pour fonctionner correctement, le tuner FM nécessite une horloge de 32,768 kHz qui peut être fournie directement par une source externe ou, comme dans notre cas, qui est générée par le circuit U4 en reliant un quartz (Q2) et 2 condensateurs (C29 et C30) aux broches 9 et 17.

Une attention particulière doit être exigée lors de la mise sous tension. En effet lorsque le circuit U4 est alimenté, il effectue un « scan » des niveaux logiques présents sur certaines broches et en fonction de la combinaison présente sur ces entrées, il démarre selon le mode de fonctionnement souhaité. Il s'agit donc d'avoir une configuration correcte de ces broches au démarrage.

Par exemple, pour sélectionner un mode de communication sur 2 lignes entre le PIC et U4, la broche 6 (SEN) doit se trouver à un niveau logique haut lorsque le récepteur FM est mis sous tension.





L'afficheur LCD peut être de type bus I²C ou RS232. Dans ce dernier cas, il faut relier la broche RX à la broche SCL.

La **sortie audio du Si4703 (U4)**, contrairement à celle du VS1053, **nécessite une amplification de puissance** pour piloter correctement un casque.

Pour cette raison, nous avons utilisé U5, un amplificateur audio de 150 mW configuré de manière à amplifier les signaux provenant du tuner FM et de les rendre disponibles en sortie sur les connecteurs J5 et J6.

Pour n'avoir qu'une seule sortie stéréo pour l'ensemble du montage (radio + MP3), nous avons fusionné les deux signaux dans U5.

Le problème est que l'audio fournie par le décodeur MP3 a un niveau plus élevé, ce qui nécessite la présence des composants R18, C18, C22, R16 pour le canal gauche et des composants R20, C28, C26 et R21 pour le canal droit.

Afin d'éviter un mélange des signaux audio, le microcontrôleur active le circuit intégré correspondant uniquement à la fonction souhaitée (radio FM ou lecteur MP3).

Les sorties audio, présentes sur les connecteurs RCA (J5 et J6), sont découplées en continu par les condensateurs C17, C24, C22 et C26, car U5 présente sur les sorties une tension continue égale à AVDD/2.

Passons maintenant au connecteur J3 qui correspond au connecteur ICSP pour la programmation et le débogage en circuit, le connecteur J2 permet d'insérer la clé USB.

Quant au connecteur J4, il véhicule les signaux du bus I²C et la masse. Le connecteur J4 permet aussi de gérer un dispositif externe.

Le bus I²C du microcontrôleur peut fonctionner alternativement en tant que maître ou en tant qu'esclave, cependant comme le PIC peut reconfigurer par logiciel ses périphériques d'entrées/sorties, il est possible d'utiliser deux signaux RX et TX d'un des ports UART pour piloter un afficheur LCD de type série.

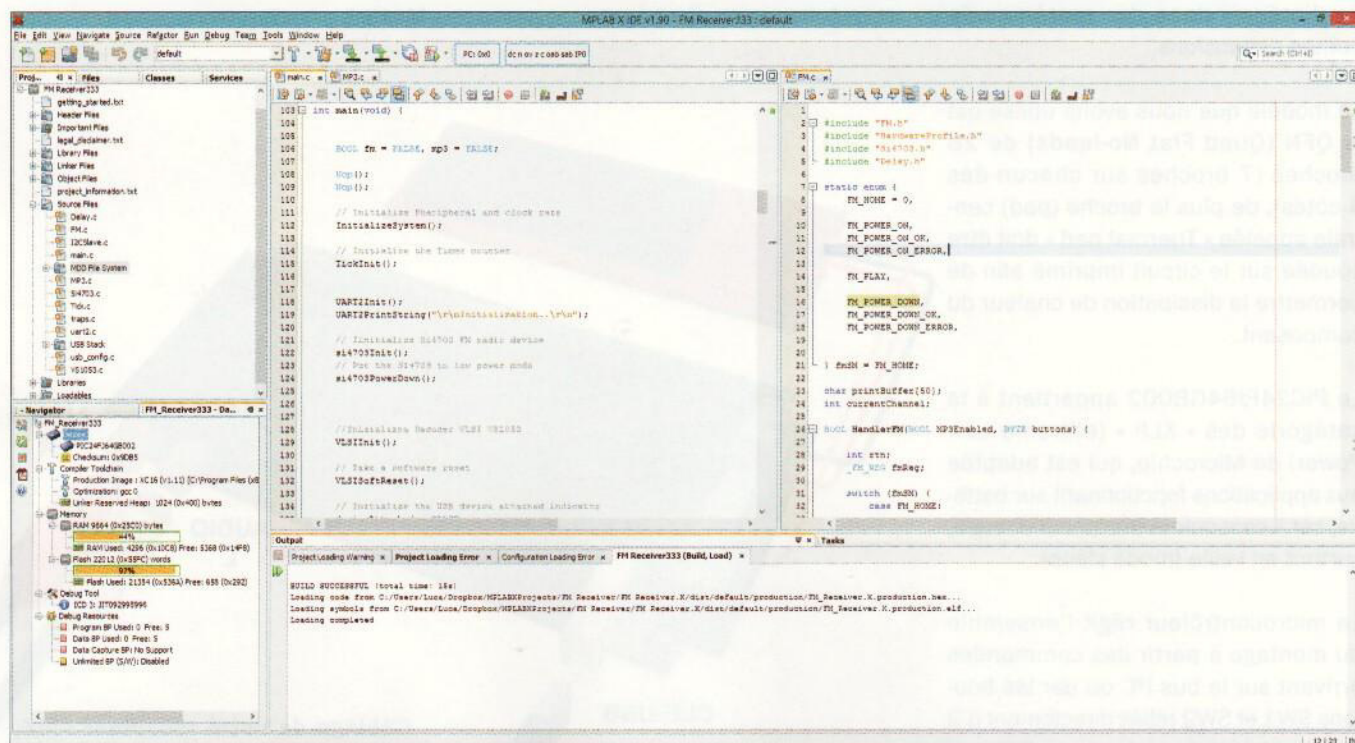


Figure 1 : fenêtre principale de l'environnement de développement MPLAB X IDE visualisant le logiciel développé pour le récepteur FM/lecteur MP3.

Fonctionnement

Les fonctions du circuit sont gérées par deux boutons SW1 (LEFT) et SW2 (RIGHT). Ils permettent de sélectionner soit la radio FM soit le lecteur MP3, de régler le volume (UP et DOWN) et de passer à la station radio suivante ou au morceau MP3 suivant.

Les différentes commandes sont définies en fonction de la **durée de la pression** sur les boutons :

- **pression brève** (durée inférieure à 250 ms) ;
- **pression normale** (durée inférieure à 500 ms) ;
- **pression longue** (moins de 1 s) ;
- **pression très longue** (inférieure à 4 s).

À la mise sous tension, le circuit initialise tous les périphériques, y compris le décodeur MP3 et le récepteur FM.

Au cours de cette phase, la LED LD1 s'allume de manière fixe et à la fin de la configuration, les deux LED clignotent alternativement et rapidement. Le programme attend alors la commande de démarrage du mode désiré (FM ou MP3) et le signale par un clignotement alterné mais lent des LED.

Pour choisir la source sonore, vous devez effectuer une **très longue pression** sur l'un des boutons : le bouton « **LEFT** » sélectionne le décodeur **MP3** et commence à jouer le 1^{er} morceau de la clé USB (l'activation est indiquée par le clignotement de LD1), tandis que le bouton « **RIGHT** » sélectionne la radio **FM** avec la fréquence de 88,4 MHz (LD2 clignote).

La sélection d'un mode fait pointer le programme dans une routine à l'intérieur de laquelle une brève pression respectivement sur les boutons « **LEFT** » et « **RIGHT** » permet de diminuer ou augmenter le volume.

Par contre, une **pression normale** sur les boutons « **LEFT** » et « **RIGHT** » permet, dans le cas du mode de fonctionnement en radio FM, le **passage à la station de radio précédente ou suivante**, alors que dans le mode MP3 le bouton « **LEFT** » permet de **répéter le morceau en cours**

Le tuner FM SI4703

Le circuit intégré SI4703 de la firme Silicon Laboratories est un récepteur et un syntoniseur radio FM intégré sur une seule puce. En outre, le SI4703 est capable de recevoir et de décoder à la fois les informations de type Radio Data Service (RDS) et de type Radio Broadcast Data Service (RBDS). Les caractéristiques de ce composant permettent d'obtenir des informations supplémentaires sur certaines stations radio, telles que le nom de l'artiste en cours de diffusion ou le nom de la station sur laquelle vous êtes syntonisé.

La couverture en réception de la bande radio FM est très large, et à travers le programme il est possible d'adapter la réception des stations américaines, européennes ou asiatiques.

Pour faire fonctionner le SI4703, il est nécessaire d'utiliser un quartz de 32,768 kHz sous une tension d'alimentation de 3,3 V et un dispositif maître, tel qu'un microcontrôleur, pour gérer les commandes via le bus I²C ou via un protocole sur 3 lignes.

Dans le tableau 1 sont résumées les principales caractéristiques du circuit FM SI4703.

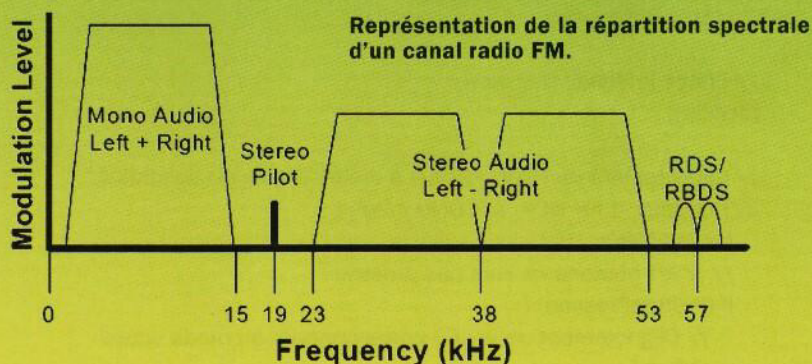


Tableau 1 : résumé des principales caractéristiques du circuit intégré SI4703.

Nom du paramètre	Valeur
Bande de fréquence FM reçue	De 76 MHz à 108 MHz
Largeur de la bande	50 kHz, 100 kHz ou 200kHz
Filtre numérique de sélection des canaux	Oui
Contrôle automatique de la fréquence (AFC)	Permet la recherche et la syntonisation automatique
Contrôle automatique du volume (AGC)	Permet d'adapter le volume sonore automatiquement
Mesure de l'intensité du signal reçu (RSSI)	Par pas de 1 dBμV
Suppression du bruit	Oui
Tension d'alimentation	De 2,7 V à 5,5 V
Dimensions du boîtier	QFN 3 x 3 mm
Réception infos RDS / RBDS	Canaux + 57kHz

et le bouton « **RIGHT** » de **passer au morceau suivant**.

Enfin, dans le mode radio FM, une pression longue sur le bouton « **RIGHT** » permet d'afficher sur l'écran LCD les informations RDS de la station de radio en cours d'écoute.

Revenons un instant sur le bornier J4, qui comprend les lignes SCL, SDA et GND du microcontrôleur. À l'aide de bibliothèques appropriées, il est possible d'obtenir du microcontrôleur un mode de fonctionnement Maître ou Esclave du bus I²C, selon l'application que vous désirez mettre en place.

Le circuit intégré TPA6111A2

Le TPA6111A2 est un amplificateur stéréo fabriqué par Texas Instruments, capable de fournir une puissance de 150 mW sous une charge de 16 Ω . Le composant peut être utilisé pour piloter directement des écouteurs ou un casque stéréo. Sa tension d'alimentation s'étend de 2,5 VDC à 5,5 VDC et ses petites dimensions le rendent approprié pour une utilisation dans des applications portables et/ou alimentées par batterie.

Le TPA6111A2 intègre un circuit de protection qui évite la destruction du composant dans le cas d'un court-circuit accidentel en sortie ou contre un échauffement excessif du composant. Son taux de distorsion harmonique totale (THD + N) est inférieur à 1 % de 20 Hz à 20 000 Hz.

L'intégration du composant à l'intérieur d'un dispositif audio portable est simple, car le nombre de composants passifs nécessaires à son fonctionnement est très faible. En effet, il suffit de 2 condensateurs de découplage en entrée et en sortie, et de 2 résistances de réglage du gain pour chaque canal.

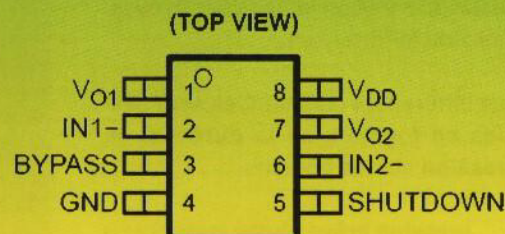


Figure 5 : brochage de l'amplificateur TAP6111A2 vu de dessus.

Listing 1

```
// Entre dans le gestionnaire de tâches coopératif
while (1) {

    // Efface le Watch Dog Timer
    ClrWdt();

    // Clignotement de LED0 (la plus à droite) toutes les secondes.
    if (TickGet() - t >= TICK_SECOND / 2ul) {
        t = TickGet();
        // Si les boutons ne sont pas pressés
        if (!buttonPressed) {
            // Clignotement de la LED correspondant au mode actuel
            if (fm) {
                flashLight(LED_TOGGLE, LED_RED, NULL, NULL);
            } else if (mp3) {
                flashLight(LED_TOGGLE, LED_WHITE, NULL, NULL);
            } else {
                flashLight(LED_TOGGLE, LED_BOTH, NULL, NULL);
            }
        }
    }
}

// Gestion des boutons
buttons = buttonHandler();

// Gestion du processus FM
// Entre en mode FM lorsqu'on appuie sur le bouton de droite pendant 4 s
fm = HandlerFM(mp3, buttons);

// Gestion du processus MP3
// Entre en mode MP3 lorsqu'on appuie sur le bouton de gauche pendant 4 s
mp3 = HandlerMP3(fm, buttons);

// Processus de gestion de la pile USB
USBTasks();
}
```

Supposons deux scénarios dans un véhicule :

- en mode Esclave, le montage pourrait faire partie d'un ensemble complexe contrôlé à partir d'une console unique au moyen du bus I2C ;
- le mode Maître peut devenir intéressant si vous disposez en aval d'un amplificateur audio à commande numérique, de façon à piloter l'affichage ou le gain de l'amplificateur, mais aussi à mettre ce dernier en veille. La gestion numérique permettrait de laisser l'amplificateur relié à la batterie en l'allumant avec une commande logique au lieu de couper physiquement l'alimentation à l'aide d'un interrupteur conventionnel.

Pour rendre le projet intéressant d'un point de vu didactique, nous avons laissé de côté le mode I2C en faveur de l'utilisation d'un port configurable par le firmware du PIC.

Cette caractéristique permet d'attribuer dynamiquement, pendant le fonctionnement, un périphérique matériel interne au microcontrôleur à un ensemble de broches. Dans ce cas, nous avons décidé de configurer le périphérique UART2 en l'affectant aux broches RB8 et RB9 du microcontrôleur.

De cette façon, nous pouvons utiliser le connecteur J4 pour communiquer avec un affichage LCD de type série.

Sinon, vous pouvez utiliser un convertisseur série USB (compatible TTL) pour visualiser sur une fenêtre d'un ordinateur les informations générées en réponse à des événements tels que la réception d'une nouvelle station de radio.

Il est clair que le choix d'un afficheur LCD présente un avantage pratique, même si le nombre d'informations visualisables est limité par le nombre de lignes et de caractères de l'écran LCD.

Le software

Maintenant, nous allons décrire le logiciel et l'environnement de développement utilisé pour mettre en œuvre le programme exécutable (.hex) du projet.

En ce qui concerne l'environnement de développement, nous utilisons la dernière version de l'IDE de Microchip à savoir « **MPLAB X** ».

Vous pouvez le télécharger librement à l'adresse suivante :

<http://www.microchip.com/mplab/mplab-x-ide>.

Une fois l'installation de « MPLAB-X » terminée, nous vous conseillons de supprimer les versions antérieures, importez notre projet pour afficher le code source (voir la figure 1).

Pour compiler notre projet, il est nécessaire d'avoir le compilateur **MPLAB XC16**.

Vous pouvez télécharger une version complète limitée à 60 jours à cette adresse suivante :

<http://www.microchip.com/mplab/compilers>.

Pour compiler notre projet, la version gratuite « Free » convient parfaitement, cependant l'optimisation de la compilation est minimale, cela ne présente aucun problème pour notre projet.

Commençons par l'analyse d'une partie du code source dans le Listing 1. Dans cet extrait de code, nous trouvons une boucle infinie nommée « **while (1)** »,

Listing 2 : fonctionnement du programme pour la partie FM

```
case FM_PLAY:

    if (update && TickGet() - timeout > 4 * TICK_SECOND) {
        clearAll();
        currentChannel = getChannel();
        sprintf(printBuffer, "FM: %02d.%01dMHz",
            currentChannel / 10, currentChannel % 10);
        UART2PrintString(printBuffer);
        update = FALSE;
    }
    // Gestion du bouton de décrémentation
    if (buttons == SW_LEFT_SHORT_PRESS) {
        // Décrémenter le volume
        rtn = volumeDown();
        clearRow(LINE_2);
        sprintf(printBuffer, "Volume (Down): %d", rtn);
        UART2PrintString(printBuffer);
        timeout = TickGet();
        update = TRUE;
    } else if (buttons == SW_LEFT_NORMAL_PRESS) {
        // Décrémenter la station radio
        rtn = seekDown();
        clearRow(LINE_2);
        sprintf(printBuffer, "FM (Down): %02d.%01dMHz",
            rtn / 10, rtn % 10);
        UART2PrintString(printBuffer);
        timeout = TickGet();
        update = TRUE;
    }

    // Gestion du bouton d'incrémenter
    if (buttons == SW_RIGHT_SHORT_PRESS) {
        // Incrémenter le volume
        rtn = volumeUp();
        clearRow(LINE_2);
        sprintf(printBuffer, "Volume (Up): %d", rtn);
        UART2PrintString(printBuffer);
        timeout = TickGet();
        update = TRUE;
    } else if (buttons == SW_RIGHT_NORMAL_PRESS) {
        // Incrémenter la station de radio
        rtn = seekUp();
        clearRow(LINE_2);
        sprintf(printBuffer, "FM (Up): %02d.%01dMHz", rtn / 10, rtn %
10);

        UART2PrintString(printBuffer);
        timeout = TickGet();
        update = TRUE;
    }

    // Gestion de la procédure de mise hors tension
    if (buttons == SW_RIGHT_LONG_PRESS) {
        int k;
        for (k = 0; k < 1; k++) {
            readRegisters(&fmReg);
            clearAll();
        }
    }
}
```



```

UART2PrintString("Radio Status:");
clearRow(LINE_2);
if (fmReg.items.statusRSSI.bits.rdsr) {
    UART2PrintString("RDS Available");
    DelayMs(700);
    clearRow(LINE_2);
    BYTE blockerrors = fmReg.items.statusRSSI.bits.BLERA;

```

// La commande et l'état des périphériques sont obtenus par des structures de données organisées en 16 registres de 16 bits chacun. La fonctionnalité des bits peut être séparée en deux grandes catégories : le contrôle et l'état. Les bits du registre BLERA indiquent combien d'erreurs ont été corrigées dans le bloc A. Si BLERA indique 6 erreurs ou plus, les données dans RDSA doivent être ignorées. Voir AN230 du SI4703.

```

        if (blockerrors == 0)
            UART2PrintString("No RDS errors");
        else if (blockerrors == 1)
            UART2PrintString("1-2 RDS errors");
        else if (blockerrors == 2)
            UART2PrintString("3-5 RDS errors");
        else if (blockerrors == 3)
            UART2PrintString("6+ RDS errors");
    } else
        UART2PrintString("Pas de RDS");
    DelayMs(1000);

    if (fmReg.items.statusRSSI.bits.stc) {
        clearRow(LINE_2);
        UART2PrintString("Tune Complete");
        DelayMs(1000);
    }

    clearRow(LINE_2);
    if (fmReg.items.statusRSSI.bits.sfb1)
        UART2PrintString("Recherche échouée ");
    else
        UART2PrintString("Recherche réussie ");
    DelayMs(1000);

    if (fmReg.items.statusRSSI.bits.afcrl) {
        clearRow(LINE_2);
        UART2PrintString("AFC/Canal non valide");
        DelayMs(1000);
    }

    if (fmReg.items.statusRSSI.bits.rdss) {
        clearRow(LINE_2);
        UART2PrintString("RDS Synch");
        DelayMs(1000);
    }
    clearRow(LINE_2);
    if (fmReg.items.statusRSSI.bits.ST)
        UART2PrintString("Stereo");
    else
        UART2PrintString("Mono");
    DelayMs(1000);

```

dans laquelle sont invoquées les routines de gestion qui permettent de faire clignoter les LED, de gérer les boutons, d'activer les modes MP3 ou FM, et enfin de gérer la librairie USB.

La mise en œuvre des tâches successives nécessite que chaque routine s'exécute le plus vite possible afin d'invoquer la suivante. Cette invocation périodique à intervalles réguliers de chaque routine crée un mécanisme multitâche simple.

Ceci correspond à un type de multitâche non-préemptif, par conséquent, si un processus ne répond pas volontairement à temps pour la prochaine tâche, le système peut cesser de fonctionner définitivement.

C'est pour cette raison que nous avons activé le « watch dog » et avons inséré la fonction de réinitialisation du compteur (Listing 1 ClrWdt()) dans la boucle principale. Nous pouvons ainsi réinitialiser automatiquement le circuit si pour une raison quelconque un processus (ou routine) rencontrerait un dysfonctionnement.

Dans le Listing 2, nous avons reporté la routine de gestion du récepteur radio FM et, dans le Listing 3, une partie de la routine de gestion du lecteur MP3. Les fonctions « **HandlerFM** », respectivement « **HandlerMP3** », sont structurées en plusieurs branches qui prennent des états différents.

De cette façon, il est possible de démarrer une sous-tâche, l'exécuter et réassigner la CPU à la tâche en cours.

Lorsque la CPU revient à la tâche en cours, les informations des états sont conservées dans une variable statique « **fmSM** » (respectivement **mp3SM**) qui permet de reprendre l'exécution là où elle a été volontairement arrêtée.

De cette manière, il est possible de recréer un environnement multitâche, même sans la présence d'un véritable système d'exploitation.

Poursuivons la description de la fonction « **HandlerFM** » (respectivement « **HandlerMP3** »), qui comme nous pouvons le voir dans le code source,


```

clearAll();
BYTE rssi = fmReg.items.statusRSSI.bits.rssi;
sprintf(printBuffer, "RSSI = %d of 75", rssi);
UART2PrintString(printBuffer);
for (i = 0; i < 10; i++) {
    readRegisters(&fmReg);
    if (fmReg.items.statusRSSI.bits.rdsr) {
        // clearAll();
        // UART2PrintString("Nous avons du RDS !");
        BYTE str[10], Ah, Al, Bh, Bl, Ch, Cl, Dh, Dl;
        Ah = (fmReg.items.rdsa & 0xFF00) >> 8;
        Al = (fmReg.items.rdsa & 0x00FF);

        Bh = (fmReg.items.rdsb & 0xFF00) >> 8;
        Bl = (fmReg.items.rdsb & 0x00FF);

        Ch = (fmReg.items.rdsc & 0xFF00) >> 8;
        Cl = (fmReg.items.rdsc & 0x00FF);

        Dh = (fmReg.items.rdsd & 0xFF00) >> 8;
        Dl = (fmReg.items.rdsd & 0x00FF);

        str[0] = Ah;
        str[1] = Al;
        str[2] = Bh;
        str[3] = Bl;
        str[4] = Ch;
        str[5] = Cl;
        str[6] = Dh;
        str[7] = Dl;
        str[8] = '\0';

        clearRow(LINE_2);
        //UART2PrintString("RDS: ");
        UART2PrintString(str);
        DelayMs(40); // Attend que le bit RDS soit effacé
        break;
    } else {
        clearRow(LINE_2);
        DelayMs(5);
        UART2PrintString("Pas de RDS");
        DelayMs(30);
    }
}
// Efface le Watch Dog Timer
ClrWdt();
//delay(500);
}
}
timeout = TickGet();
update = TRUE;
}
// Gestion de la procédure de mise hors tension
if (buttons == SW_RIGHT_4000MS_PRESS) {
    fmSM = FM_POWER_DOWN;
}
break;

```

// À partir de AN230, en utilisant la méthode polling, 40ms devrait être suffisant entre 2 contrôles

prévoit pour chaque branche une fonction spécifique qui est exécutée étape par étape.

Le fonctionnement complet de la radio (ou du lecteur MP3), depuis la phase d'allumage jusqu'à la phase d'arrêt, fait appel à toutes les branches du programme.

L'augmentation ou la diminution du volume, le passage d'une station radio à la suivante (ou une piste MP3), sont gérés à l'intérieur de la branche « FM_PLAY » (respectivement « MP3_PLAY »).

Ainsi dans chaque cycle, la présence du flag « buttons » indiquant la fin de la pression sur un bouton par l'utilisateur est vérifiée. Si le test est positif, le programme procède à l'action requise.

De même, l'extinction de la radio FM (ou du lecteur MP3) se produit en vérifiant l'état du flag qui signale une pression longue sur le bouton correspondant.

Si le test est positif, le programme passe à l'étape suivante qui consiste à éteindre la radio (ou le MP3) en retournant une valeur « FALSE », l'utilisateur peut alors démarrer de nouveau la radio ou le lecteur MP3.

Réalisation pratique

Nous abordons maintenant le dernier aspect, probablement le plus intéressant, en faisant d'abord une brève description des étapes et conseils à suivre pour assembler correctement les composants de ce projet.

De toute évidence, la première étape consiste à obtenir l'ensemble des composants, ainsi que tout le matériel nécessaire pour assembler le circuit imprimé.

Nous mettons en téléchargement libre, comme d'habitude, sur notre site www.electroniquemagazine.com dans le sommaire détaillé de la revue (à l'onglet « Télécharger ») les fichiers GERBER (pour une fabrication professionnelle) et les typons des circuits imprimés ainsi que le programme du microcontrôleur.

Plan de montage

Figure B : circuit imprimé à l'échelle 1 : 1 côté soudures.

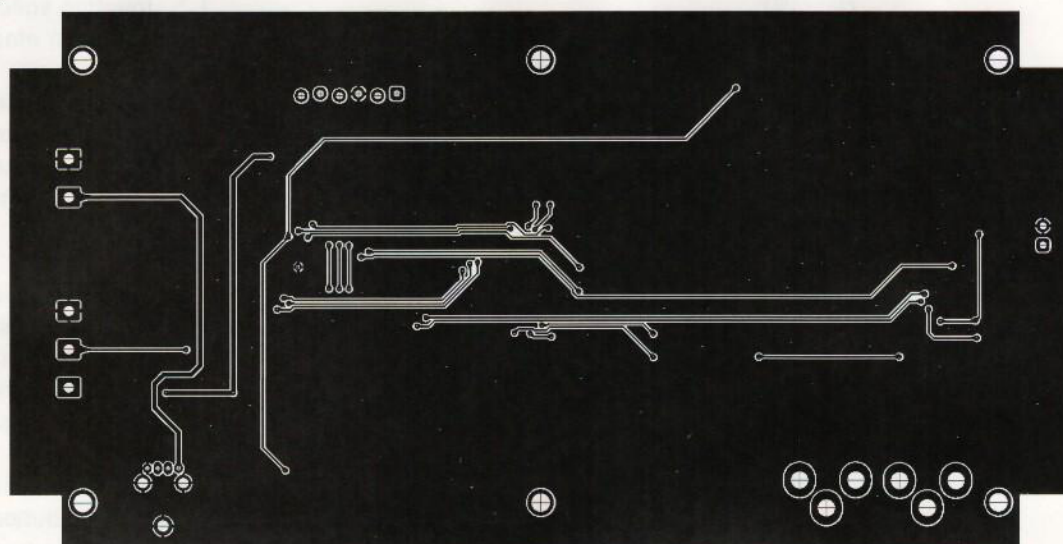
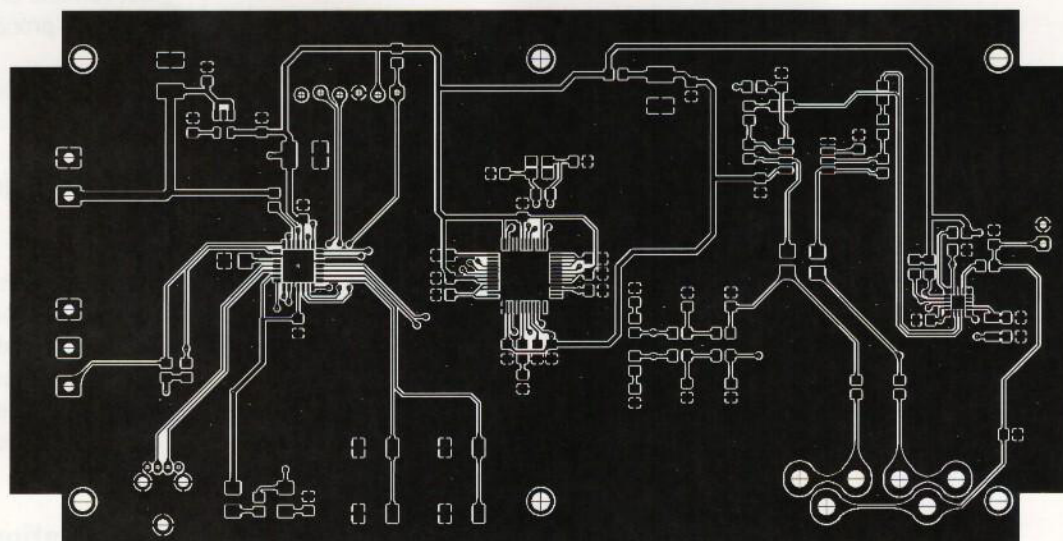


Figure C : circuit imprimé à l'échelle 1 : 1 côté composants.



Liste des composants

R1.....1 k Ω boîtier 0805
 R2.....4,7 k Ω boîtier 0805
 R3.....1 k Ω boîtier 0805
 R4.....100 k Ω boîtier 0805
 R5.....10 k Ω boîtier 0805
 R6.....10 k Ω boîtier 0805
 R7.....10 k Ω boîtier 0805
 R8.....4,7 k Ω boîtier 0805
 R9.....4,7 k Ω boîtier 0805
 R10.....10 k Ω boîtier 0805
 R11.....non connectée
 R12.....20 k Ω boîtier 0805
 R13.....20 k Ω boîtier 0805
 R14.....100 k Ω boîtier 0805
 R15.....1 M Ω boîtier 0805
 R16.....10 k Ω boîtier 0805
 R17.....20 k Ω boîtier 0805
 R18.....470 Ω boîtier 0805

R19.....20 k Ω boîtier 0805
 R20.....470 Ω boîtier 0805
 R21.....10 k Ω boîtier 0805
 R22.....0 Ω boîtier 0805
 R23.....10 Ω boîtier 0805
 R24.....20 Ω boîtier 0805
 R25.....20 Ω boîtier 0805
 R26.....non connectée
 C1.....470 pF céramique boîtier 0805
 C2.....100 nF céramique boîtier 0805
 C3.....100 nF céramique boîtier 0805
 C4.....100 μ F céramique boîtier 1812
 C5.....100 nF céramique boîtier 0805
 C6.....10 μ F céramique boîtier 1206

C7.....100 nF céramique boîtier 0805
 C8.....100 nF céramique boîtier 0805
 C9.....100 nF céramique boîtier 0805
 C10.....100 nF céramique boîtier 0805
 C11.....100 nF céramique boîtier 0805
 C12.....100 nF céramique boîtier 0805
 C13.....100 nF céramique boîtier 0805
 C14.....100 nF céramique boîtier 0805
 C15.....100 μ F céramique boîtier 1812
 C16.....100 μ F céramique boîtier 1812

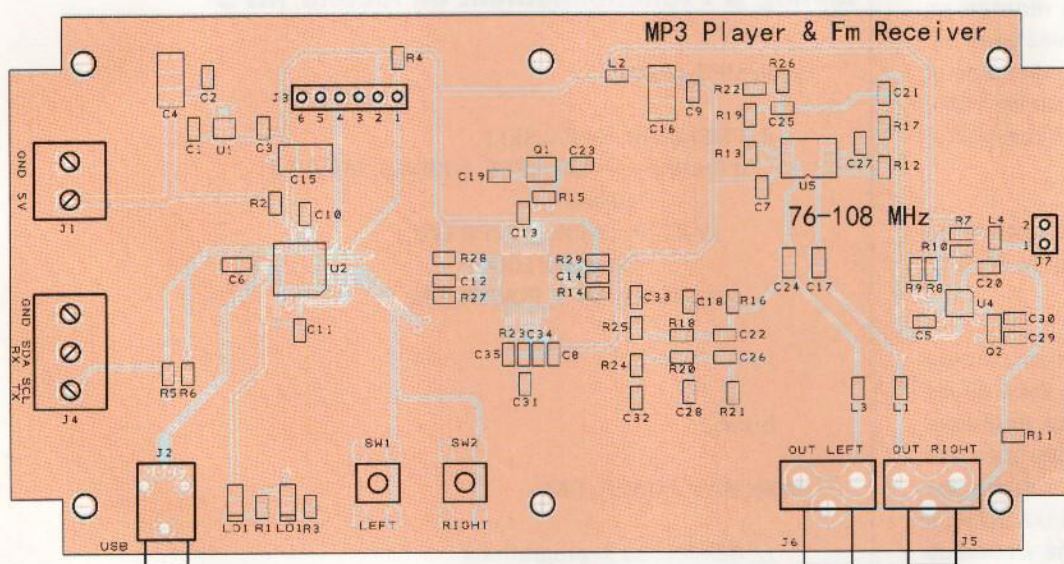


Figure D : plan de câblage des composants.

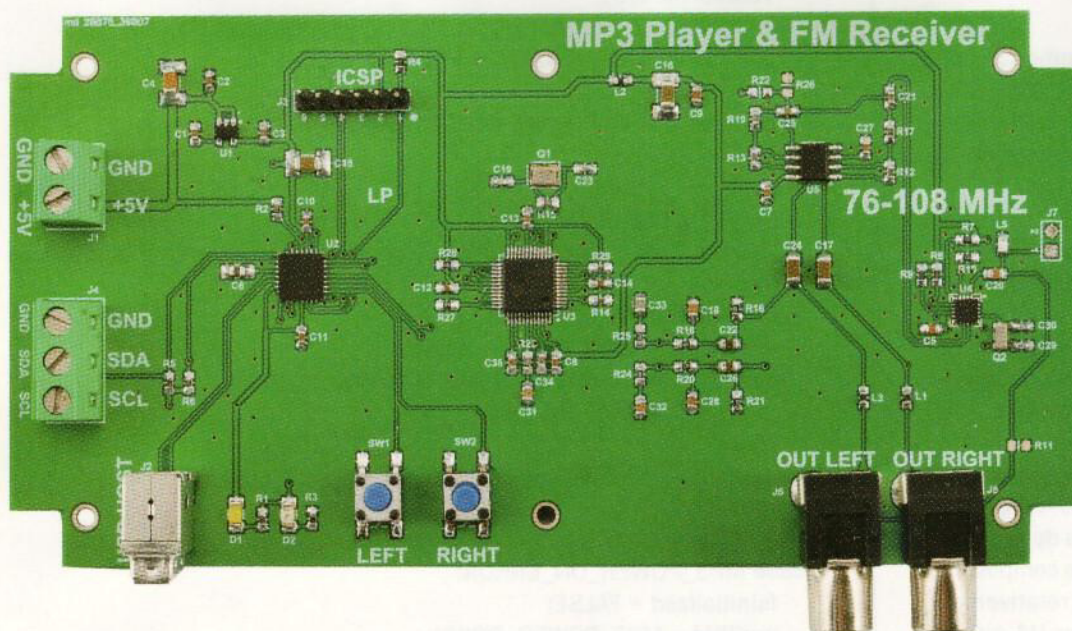


Figure E : photo de l'un de nos prototypes.

C1747 μ F céramique boîtier 1206
 C183,3 nF céramique boîtier 0805
 C1933 pF céramique boîtier 0805
 C201 nF céramique boîtier 0805
 C21.....1 μ F céramique boîtier 0805
 C2210 μ F céramique boîtier 0805
 C2333 pF céramique boîtier 0805
 C24.....47 μ F céramique boîtier 1206
 C251 μ F céramique boîtier 0805
 C2610 μ F céramique boîtier 0805
 C27.....100 nF céramique boîtier 0805
 C283,3 nF céramique boîtier 0805
 C2922 pF céramique boîtier 0805
 C3022 pF céramique boîtier 0805
 C31.....47 nF céramique boîtier 0805
 C3210 nF céramique boîtier 0805
 C3310 nF céramique boîtier 0805

C341 μ F céramique boîtier 0805
 C3510 nF céramique boîtier 0805
 U1.....TC1015-3.3VCT713
 U2PIC24FJ64GB002-I/ML
 U3.....VS1053B
 U4SI4703-C19-GM
 U5TPA6111A2D

 SW1...microswitch CMS
 SW2...microswitch CMS

 LD1....LED verte boîtier 1206
 LD2....LED rouge boîtier 1206

 L1ferrite 1k Ω /100 MHz boîtier 0805
 L2ferrite 1k Ω /100 MHz boîtier 0805
 L3ferrite 1k Ω /100 MHz boîtier 0805

L4inductance 270 nH boîtier 0805
 LCD.....série LCDSER 16 x 2
 Q1.....quartz 12,288 MHz
 Q2.....quartz 32,768 kHz

Divers

Borniers 2 pôles
 Borniers 3 pôles
 Connecteur USB-A femelle
 Barrette mâle 6 pôles
 Prise RCA pour circuit imprimé (x 2)

Le circuit est composé presque entièrement de composants montés en surface (CMS). Vous devez avoir un peu d'expérience dans ce domaine et vous devez être équipé d'un fer à souder pour CMS (20 W) avec une pointe de 0,2 mm.

D'autre part vous devez utiliser de la soudure pour composants CMS de 0,5 mm de diamètre.

Pour la mise en place des composants avant le soudage, vous devez appliquer sans excès aux emplacements prévus de la colle spéciale pour composants CMS. Pour monter correctement les circuits U3 et U4, il est souhaitable d'avoir une station de soudage à air chaud.

Comme d'habitude, commencez par souder les résistances CMS, puis les condensateurs CMS, les circuits intégrés et ensuite les poussoirs, les connecteurs et les borniers.

En cas d'excès de soudure, notamment au niveau des circuits intégrés, utilisez de la tresse à dessouder pour CMS afin d'enlever la soudure qui pourrait se trouver sous le composant et provoquer des courts-circuits.

Sans aucun doute, les composants les plus critiques à souder sont U2, U3 et U4. Nous vous conseillons de commencer, après avoir souder les composants passifs, par souder U1 (relativement facile) et de terminer par U4 qui est probablement le composant du circuit imprimé le plus critique à souder.

Pour obtenir un bon résultat, il est conseillé d'utiliser des aérosols pour le soudage des composants CMS.

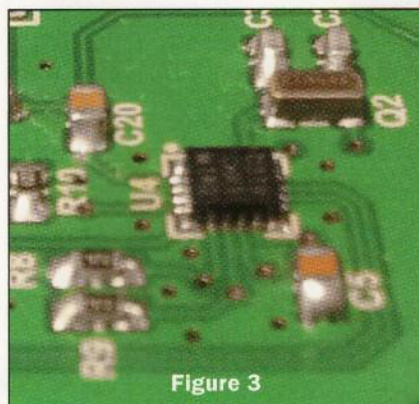


Figure 3

Listing 3 : fonctionnement du lecteur MP3

```
case MP3_HOME:

    if (FMEnabled == FALSE) {
        if (buttons == SW_LEFT_4000MS_PRESS) {
            clearAll();
            UART2PrintString("MP3 ON");
            DelayMs(1000);
            mp3SM = MP3_POWER_ON;
        }
    }

    break;

case MP3_POWER_ON:

    // Réinitialise le programme
    VLSISoftReset();
    clearAll();
    UART2PrintString("Waiting for a USB device...");
    DelayMs(700);
    mp3SM = MP3_POWER_ON_OK;
    break;

case MP3_POWER_ON_OK:
    // Si la clé est connectée, va à l'étape suivante sinon attend la présence
    // d'une clé USB
    if (fsInitalized == TRUE)
        mp3SM = MP3_SEARCH_FIRST;
    else if (USBHostMSDSCSIMediaDetect())
        mp3SM = MP3_FS_INIT;
    else
        mp3SM = MP3_POWER_ON_ERROR;
    break;

case MP3_POWER_ON_ERROR:
    fsInitalized = FALSE;
    mp3SM = MP3_POWER_DOWN;
    break;

case MP3_FS_INIT:
    if (FSInit())
        mp3SM = MP3_FS_INIT_OK;
    else
        mp3SM = MP3_FS_INIT_ERROR;
    break;

case MP3_FS_INIT_OK:
    fsInitalized = TRUE;
    clearAll();
    UART2PrintString("File system OK");
    DelayMs(700);
    mp3SM = MP3_SEARCH_FIRST;
    break;

case MP3_FS_INIT_ERROR:
    fsInitalized = FALSE;
```



```

        clearAll();
        UART2PrintString("File system Error");
        DelayMs(700);
        mp3SM = MP3_POWER_DOWN;
        break;

case MP3_SEARCH_FIRST:
    // Démarre la recherche de fichiers MP3
    clearAll();
    UART2PrintString("Ouverture du fichier MP3 ");
    DelayMs(700);
    clearRow(LINE_2);
    UART2PrintString("Recherche...");
    DelayMs(700);

    if (FindFirst("*.mp3", attributes, &rec) == 0) {
        p = getPointer(&rec);
        clearAll();
        sprintf(printBuffer, "Found: %s", p);
        UART2PrintString(printBuffer);
        mp3SM = MP3_OPEN_FILE;
    } else {
        mp3SM = MP3_SEARCH_ERROR;
    }
    break;

case MP3_SEARCH_NEXT:

    if (FindNext(&rec) == 0) {
        p = getPointer(&rec);
        clearAll();
        sprintf(printBuffer, "Found: %s", p);
        UART2PrintString(printBuffer);
        mp3SM = MP3_OPEN_FILE;
    } else {
        mp3SM = MP3_SEARCH_ERROR;
    }
    break;

case MP3_SEARCH_ERROR:
    clearAll();
    UART2PrintString("Le dossier est vide ");
    DelayMs(700);
    mp3SM = MP3_POWER_DOWN;
    break;

case MP3_OPEN_FILE:
    // Essaie d'ouvrir le fichier trouvé
    if (myFile != NULL)
        FSfclose(myFile);

    myFile = FSfopen(p, FS_READ);
    if (myFile == NULL)
        mp3SM = MP3_OPEN_FILE_ERROR;
    else
        mp3SM = MP3_OPEN_FILE_OK;

    break;

```

Nous vous recommandons de procéder de la façon suivante. Vaporisez une fine couche d'aérosol sur le circuit imprimé et ensuite, sur chaque « pad » de l'emplacement du composant CMS étalez à l'aide du fer à souder très peu de soudure.

Le même procédé doit être effectué avec le « pad » central (« pad » thermique) tout en veillant à ce que la soudure ne vienne pas en contact avec les « pads » latéraux.

Pour souder U4, vous devez d'abord chauffer légèrement l'endroit du circuit imprimé sur lequel il doit être monté, puis à l'aide d'une pince disposez délicatement U4 (en respectant bien évidemment l'orientation) sur les pads. Commencez par chauffer le centre, puis ensuite soudez les broches.

Évitez de trop rapprocher la station d'air chaud afin de ne pas endommager le circuit intégré. Le résultat doit correspondre à celui de la figure 3.

Dans le cas où le soudage à l'air chaud aurait laissé des endroits sans étain, à l'aide du fer à souder, ajoutez délicatement un peu d'étain en vérifiant que de la soudure ne s'introduise pas en-dessous du circuit.

Utiliser du flux à souder lors de cette opération. Le rôle du flux est de nettoyer les « pads » à souder et de s'oxyder à leur place quand la température augmente.

Notez aussi que lorsque vous soudez des composants CMS, la température du fer doit être aux alentours de 350 °C environ, il est déconseillé de dépasser une température supérieure à 400 °C.

Des températures élevées peuvent apparemment rendre plus facile le soudage, cependant l'effet contraire se produit avec un vieillissement prématuré de l'étain (oxydation plus rapide).

En fixant une température convenable, comprise entre 350 °C et 370 °C, le flux s'évapore moins vite et favorise la soudure tout en empêchant les « pads » du circuit imprimé de brûler.


```

case MP3_OPEN_FILE_OK:
    //UART2PrintString("Fichier ouvert.\r\n");
    mp3SM = MP3_PLAY;
    break;

case MP3_OPEN_FILE_ERROR:
    //UART2PrintString("Fichier non ouvert.\r\n");
    mp3SM = MP3_SEARCH_NEXT;
    break;

case MP3_PLAY:

    if ((numBytes = FSread(myData, 1, sizeof (myData), myFile)) != 0)
        VLSIPutArray(myData, numBytes);
    else
        mp3SM = MP3_CLOSE_FILE;

    // Gestion du bouton de décrémentation
    if (buttons == SW_RIGHT_SHORT_PRESS) {
        // décrémente le volume
        rtn = VLSI_GetLeft();
        if (rtn > 0) {
            rtn--;
            VLSI_SetVol(rtn, rtn);
        }
        clearAll();
        sprintf(printBuffer, "Volume: -%ddB", rtn);
        UART2PrintString(printBuffer);
    } else if (buttons == SW_RIGHT_NORMAL_PRESS) {
        clearAll();
        UART2PrintString("Skip track");
        mp3SM = MP3_CLOSE_FILE;
    }

    // Gestion du bouton d'incrémentation
    if (buttons == SW_LEFT_SHORT_PRESS) {
        // Incrémente le volume
        rtn = VLSI_GetLeft();
        if (rtn < 255) {
            rtn++;
            VLSI_SetVol(rtn, rtn);
        }
        clearAll();
        sprintf(printBuffer, "Volume: -%ddB", rtn);
        UART2PrintString(printBuffer);
    } else if (buttons == SW_LEFT_NORMAL_PRESS) {
        clearAll();
        UART2PrintString("Recommence");
        mp3SM = MP3_SEARCH_FIRST;
    }

    // Gestion de la mise hors tension
    if (buttons == SW_LEFT_4000MS_PRESS) {
        mp3SM = MP3_POWER_DOWN;
    }
    break;

```

Le reste des composants à souder sont classiques, il s'agit des borniers, des prises RCA, du connecteur USB et des boutons poussoirs. Terminez enfin par la barrette mâle à 6 pôles.

Nous vous recommandons l'utilisation d'un produit sous forme d'aérosol pour le nettoyage des résidus de soudure, de manière à rendre le circuit propre et brillant.

Programmation du firmware

Nous allons maintenant programmer le dsPIC, c'est-à-dire transférer le programme (firmware) dans la mémoire FLASH. Pour cela il est préférable d'utiliser un programmeur de type ICD3 ou ICD4 de Microchip et de relier celui-ci au connecteur ICSP du circuit.

Reliez l'ICD3 au PC via le câble USB, alimentez le montage et lancez l'environnement de développement MPLAB IDE.

Sélectionnez le programme exécutable (.hex) correspondant et démarrez le processus de programmation.

Une fois la programmation terminée, débranchez le connecteur ICSP puis redémarrez le circuit afin que le programme du microcontrôleur dsPIC s'exécute correctement.

Pour une utilisation dans une voiture ou un camion, rappelez-vous que vous disposez d'une tension de 12 ou 24 V, vous devez utiliser un régulateur classique de type 7805, dont la tension d'entrée peut atteindre 35 V, correctement filtré sur son entrée et sa sortie.

Comme nous l'avons mentionné, l'ensemble du projet est téléchargeable gratuitement sur notre site **www.electroniquemagazine.com** dans le sommaire détaillé de la revue **141** à l'onglet « Télécharger » (descendre vers le bas de la page). Il comprend les fichiers GERBER (pour une fabrication professionnelle), les typons des circuits imprimés ainsi que **le projet source complet en langage C** (s'ouvre avec MBLABX) et le programme « **FM_Receiver.X.production.hex** » du microcontrôleur qui se trouve dans le dossier : **FM Receiver.X\dist\default\production.**

Enregistreur vocal avec PIC & EEPROM

Première partie

de Marco LANDONI

Deux modules simples, gérés par un microcontrôleur PIC, peuvent résoudre tous les problèmes liés à l'enregistrement et à la reproduction de messages vocaux. Particulièrement adaptés aux systèmes de sécurité et d'alarme, ces modules fonctionnent de manière autonome, mais peuvent également être programmés par un PC.

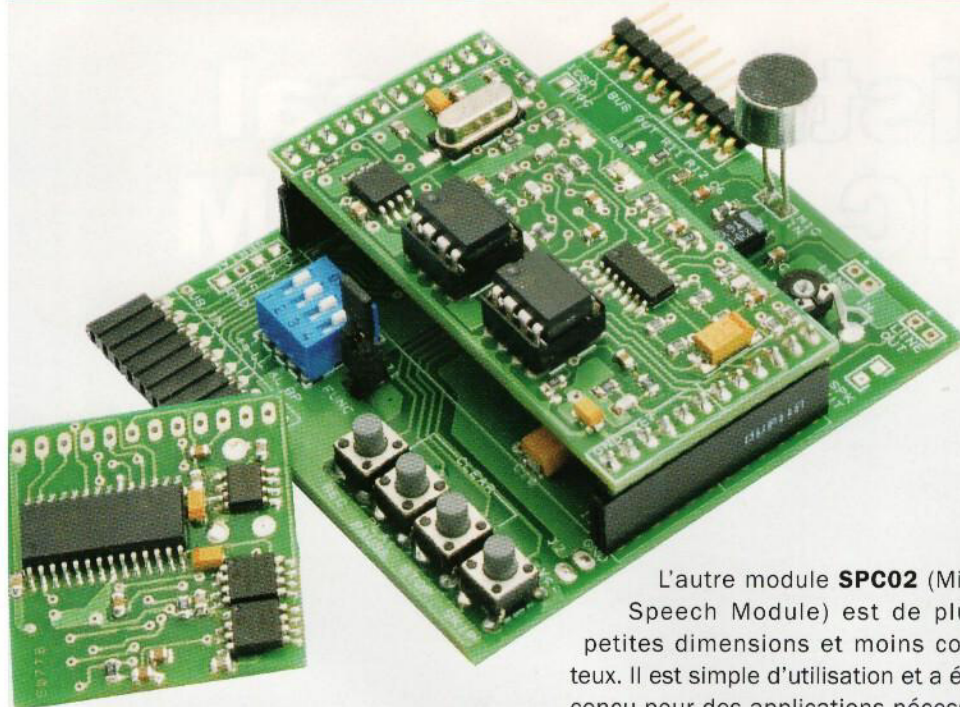


La famille des produits des enregistreurs numériques de messages vocaux, basés sur les circuits intégrés **ISD** (Informatin Storage Device), connaît un grand succès chez les amateurs et les utilisateurs professionnels. Les applications sont multiples, et ces circuits sont simples d'utilisation et faciles à mettre en œuvre.

Ces produits reposent essentiellement sur une famille de circuits intégrés, l'**ISDXXX** commercialisée par **Winbond Semiconductors**. Ces circuits sont disponibles en boîtier DIP ou CMS. Ils contiennent des **codecs audio**, des **filtres analogiques**, une **mémoire FLASH** et un petit microcontrôleur pour gérer le tout.

D'un point de vue fonctionnel, ces circuits peuvent **capter** un **signal vocal**, le **numériser** et **l'enregistrer** dans la mémoire **FLASH** en **temps réel** et de manière très simple. Ils sont aussi capables d'effectuer l'opération inverse, c'est-à-dire récupérer les données enregistrées dans la mémoire et les convertir en signaux analogiques pour reproduire un son.

L'utilisateur n'a rien à faire, il suffit de les connecter correctement et d'ajouter quelques composants passifs externes. La logique de fonctionnement, la durée maximale d'enregistrement, le nombre maximal de séquences enregistrables et leur gestion varient selon le circuit intégré de la famille. Il est donc possible d'implémenter de nombreuses applications.



Vous pouvez diffuser un message de bienvenue, ou réaliser une alarme vocale pour les systèmes de sécurité, ou tout simplement fabriquer un gadget.

Compte tenu de l'intérêt particulier de nos lecteurs pour ce type de produit et de la difficulté toujours croissante de trouver ce type de composant (la société qui a développé à l'origine ces circuits intégrés de stockage d'informations a connu des problèmes, maintenant les puces sont produites en Asie par le groupe Nuvoton Technology qui n'est pas représenté en Europe), nous avons pensé à **vous proposer un module d'enregistrement vocal avec des fonctions similaires** à celles d'un circuit **ISD**, mais utilisant la **combinaison** d'un **microcontrôleur** et d'une mémoire **EEPROM**.

Ce choix nous permet de fabriquer des modules très compacts qui se substituent aux circuits ISD mais qui, grâce au programme du microcontrôleur, intègrent toutes les fonctions et les caractéristiques requises par nos lecteurs.

Ce que nous vous proposons est un couple de modules. Le plus grand, dénommé **SPC01** est un module de parole (Speech Module). Il offre une grande souplesse d'utilisation et une grande capacité d'enregistrement, il s'adresse à ceux qui veulent un système complètement fonctionnel pouvant être utilisé dans divers contextes et de diverses manières.

L'autre module **SPC02** (Mini Speech Module) est de plus petites dimensions et moins coûteux. Il est simple d'utilisation et a été conçu pour des applications nécessitant de répéter seulement un ou deux messages courts avec une activation manuelle ou par un déclenchement externe.

Le module SPC01

En figure 1 vous pouvez observer le synoptique du module SPC01. La philosophie adoptée lors de la conception du module était de créer un module compact permettant l'enregistrement et la reproduction de messages, tout en intégrant les composants nécessaires afin d'implémenter les nombreuses fonctions requises.

Nous avons délibérément éliminé du projet les sections de préamplification

du microphone, d'amplification et de logique de commande, dans le but de créer un module universel afin d'être intégré dans des applications personnalisées.

Ainsi, ceux qui veulent enregistrer leur propre voix directement à partir d'un microphone, devront utiliser un préamplificateur pour microphone et connecter sa sortie à l'une des entrées du module d'enregistrement. À l'inverse, ceux qui disposent d'un microphone déjà amplifié (sortie d'un préampli audio), peuvent simplement le connecter à l'entrée de notre circuit en évitant la construction d'un préamplificateur pour microphone.

Il en va de même pour les sorties au niveau ligne. Pour alimenter un haut-parleur, il est nécessaire d'avoir un amplificateur de puissance et de relier l'entrée de ce dernier à la sortie audio du module.

Les entrées analogiques ainsi que les sorties sont au nombre de deux, chacune correspondant à une fréquence d'échantillonnage déterminée. Ainsi, le microcontrôleur utilisera automatiquement l'entrée relative à la fréquence d'échantillonnage sélectionnée.

Les **fréquences d'échantillonnage** d'enregistrement sont paramétrables sur **8 ksp/s** (8 000 échantillons par seconde) et **16 ksp/s**, cela permet d'obtenir dans le premier cas une bande passante allant jusqu'à 3,1 kHz et dans le second cas jusqu'à 5,7 kHz.

Les circuits intégrés de la famille ISD (Information Storage Device) peuvent enregistrer et reproduire des messages vocaux avec une durée de 10 secondes à plusieurs minutes.

Leur facilité d'utilisation, la présence d'une mémoire non volatile et le petit nombre de composants externes nécessaires pour fonctionner rendent ces circuits très populaires auprès des amateurs et des utilisateurs professionnels.

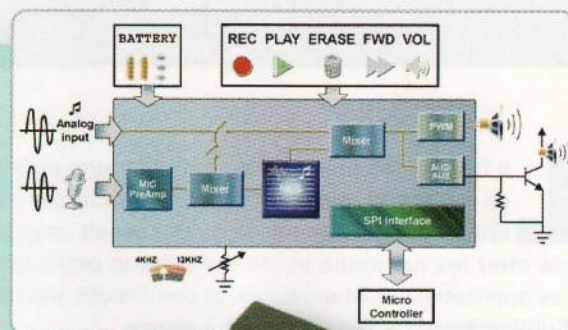


Schéma synoptique du module SPC01

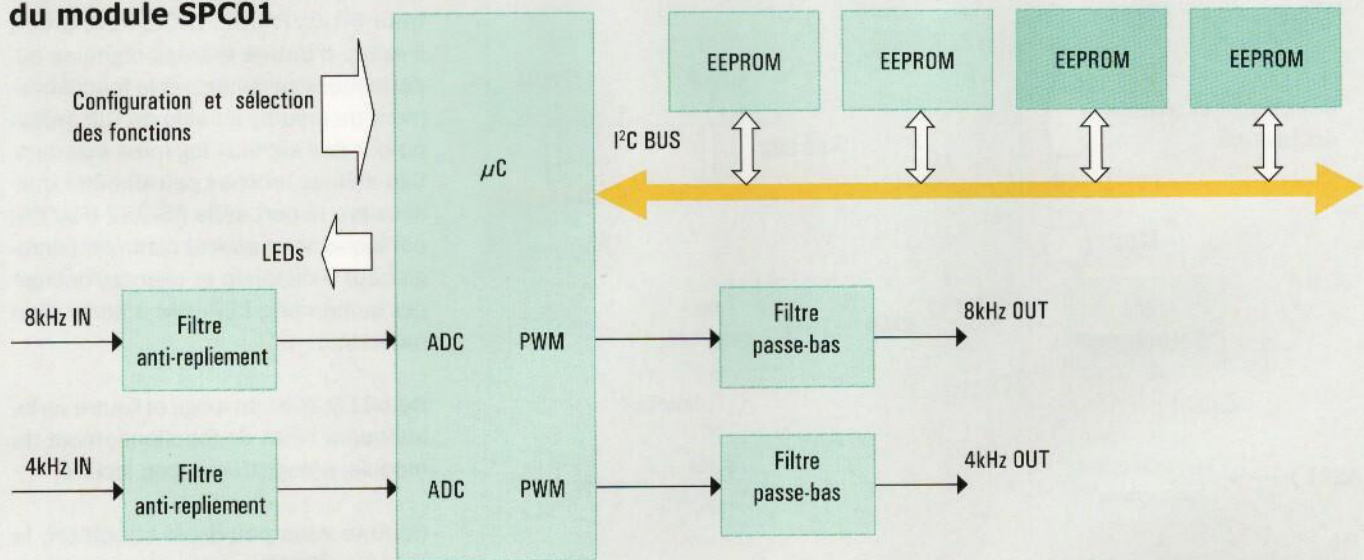
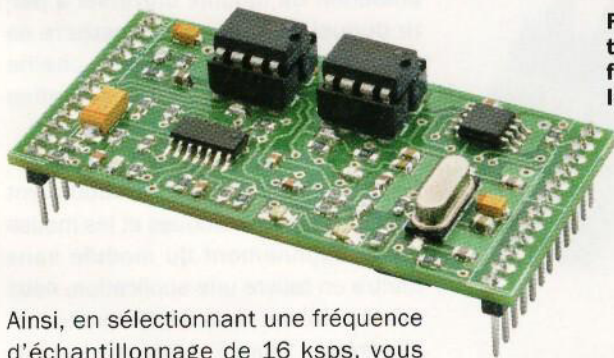


Figure 1 : schéma synoptique complet du module SPC01.



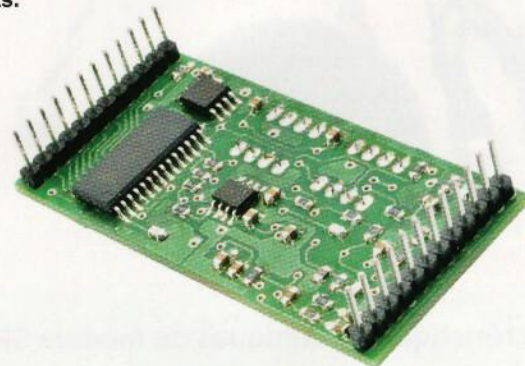
Ainsi, en sélectionnant une fréquence d'échantillonnage de 16 ksp/s, vous obtenez une meilleure qualité sonore que pour la fréquence de 8 ksp/s, cependant le temps d'enregistrement sera plus petit.

Nous avons décidé de ne pas utiliser de codecs audio externes afin de miniaturiser le système au maximum. À la place, les **entrées ADC** du **PIC** sont utilisées pour capturer le signal d'entrée, et les périphériques PWM pour convertir le signal digital en analogique (sortie).

Dans certaines limites, il est possible de choisir la quantité de mémoire à insérer dans le module. Jusqu'à **4 mémoires peuvent être montées sur le circuit**, cela permet d'obtenir des temps d'enregistrement différents.

Vous n'avez pas à configurer des paramètres car le microcontrôleur reconnaîtra automatiquement le nombre de circuits mémoire présents et les gèrera correctement.

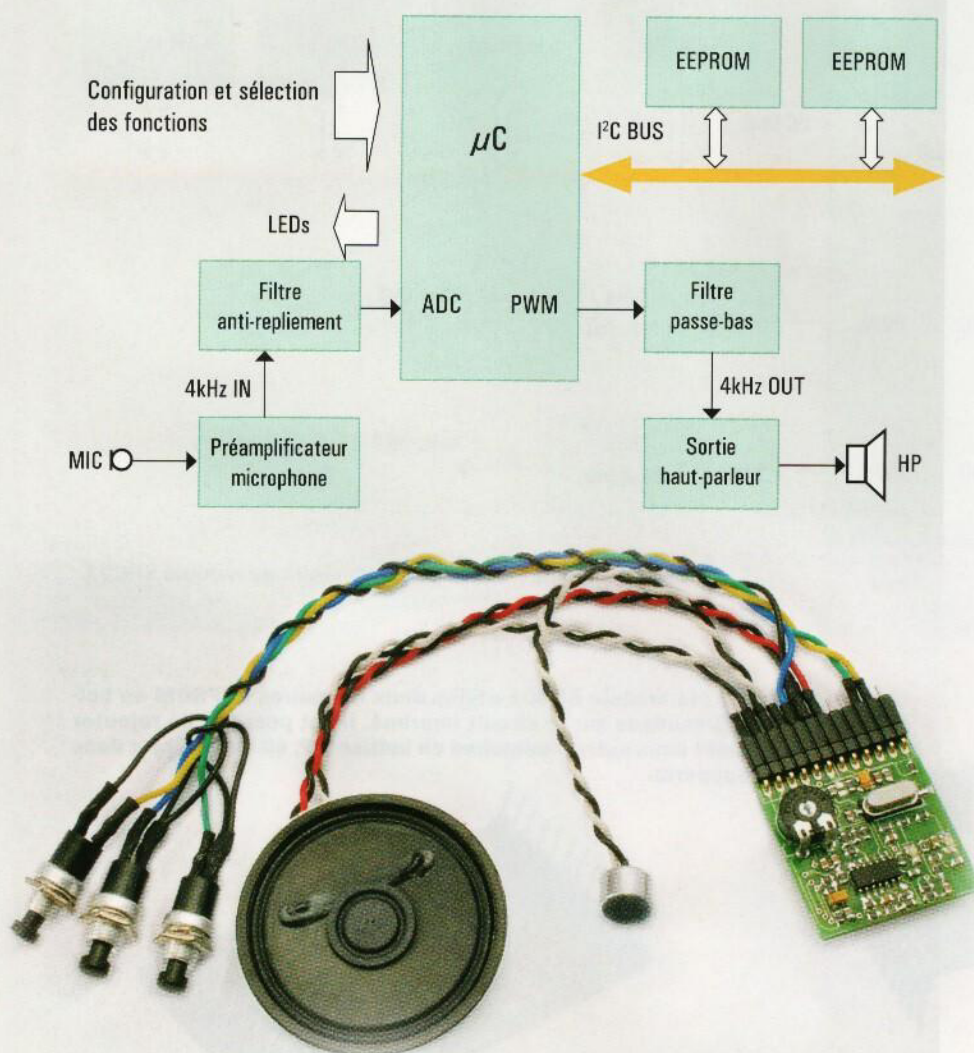
Figure 2 : le module SPC01 utilise deux mémoires EEPROM en boîtier CMS soudées sur le circuit imprimé. Il est possible de rajouter facilement deux autres mémoires en boîtier DIP, en les insérant dans leurs supports.



Caractéristiques techniques du module SPC01 :

- **Format d'enregistrement : ADPCM 4 bits ;**
- **Durée d'enregistrement :**
65 s à 16 ksp/s / 130 s à 8 ksp/s avec 4 mémoires ;
48 s à 16 ksp/s / 96 s à 8 ksp/s avec 3 mémoires ;
32 s à 16 ksp/s / 64 s à 8 ksp/s avec 2 mémoires ;
- **Fréquence d'échantillonnage : 16 ksp/s ou 8 ksp/s sélectionnable ;**
- **Résolution : 8 bits ;**
- **Tension d'entrée : 1,5 Vpp max ;**
- **Tension de sortie : 4,5 Vpp max à la tension d'entrée max ;**
- **Commandes : lecture (PLAY), pause, stop, enregistrement (REC), saut de piste (TRACK SKIP) ;**
- **Fonctions : bip fin de piste ;**
lecture d'une piste en boucle ;
lecture en boucle de toutes les pistes ;
accès à une piste de manière séquentielle ou indexée ;
suppression de la dernière piste ou suppression totale ;
commandes et fonctions via le port série (RS-232) ;
LED rouge et verte de signalisation.

Synoptique et caractéristiques du module SPC02



Caractéristiques techniques du module SPC02 :

- Format d'enregistrement : ADPCM 4 bits ;
- Durée d'enregistrement : 65 s à 8 ksps ;
- Fréquence d'échantillonnage : 8 ksps ;
- Résolution : 8 bits ;
- Entrée analogique pour microphone : (40 mVpp max) ;
- Sortie analogique amplifiée pour HP : (500 mW à 16 Ω) ;
- Commandes : PLAY piste 1, PLAY piste 2, REC ;
- Fonctions : commandes et fonctions via le port série (RS-232), LED rouge et verte de signalisation.

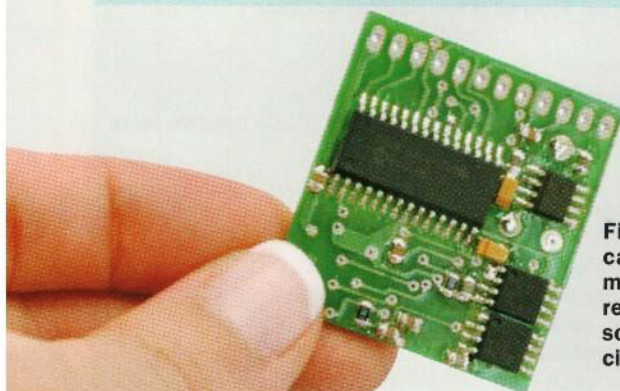


Figure 3 : schéma synoptique et caractéristiques techniques du module SPC02 d'enregistrement/reproduction, les caractéristiques sont très similaires à celles d'un circuit ISD.

En plus des entrées digitales pour les boutons de contrôle de l'enregistreur (PLAY/PAUSE, STOP, REC, SKIP), il existe d'autres entrées digitales qui permettent de configurer le fonctionnement du module à l'aide de DIP switch ou par des signaux logiques externes. Ces mêmes broches peuvent être utilisées par le port série RS-232 (RX/TX), par lequel vous pouvez contrôler l'enregistreur à distance et charger/décharger la mémoire EEPROM à partir d'un ordinateur.

Deux LED, l'une en rouge et l'autre verte, indiquent l'état de fonctionnement du module, enregistrement ou lecture.

Comme vous pouvez le constater, le module SPC01 ne dispose pas d'entrée pour un microphone, ni de sortie pour un haut-parleur. Ce choix résulte du fait que nous souhaitions présenter un module universel à partir duquel chacun peut construire sa propre application avec une chaîne de préamplification et d'amplification spécifique.

Pour ceux d'entre vous qui voudraient tester les caractéristiques et les modes de fonctionnement du module sans mettre en œuvre une application, nous avons créé une carte de démonstration (illustrée en figure 9) à l'aide de laquelle vous pouvez monter le module SPC01 et commencer à l'utiliser. Cette carte est dotée d'une entrée microphone, d'une entrée ligne, d'une sortie haut-parleur et d'une sortie ligne. Les DIP switch permettent un réglage simple des différentes fonctions disponibles, y compris celle de l'interface série.

Le module SPC02

Le schéma synoptique et les photographies sont illustrées en figure 3. Vous remarquerez que le module SPC02 est principalement réservé à une utilisation pour des messages vocaux de courte durée, par rapport au SPC01. En particulier, afin de réduire les coûts et les dimensions, il a été décidé d'utiliser que 2 mémoires EEPROM et de fixer une seule fréquence d'échantillonnage à 8 ksps. Cela permet d'enregistrer jusqu'à 65 secondes de messages vocaux.

Par contre, le **module SPC02 dispose d'un préamplificateur pour microphone** et d'une **sortie amplifiée** pour haut-parleur avec un contrôle du volume. La taille du module et le nombre de composants externes nécessaires au fonctionnement ont été minimisés. Le module n'a besoin que d'un microphone, un haut-parleur, et 3 boutons ainsi que d'une alimentation pour fonctionner.

L'interface utilisateur est simplifiée car certaines fonctionnalités implémentées dans le module SPC01 ont été supprimées. Seules les commandes basiques sont présentes pour une utilisation simple et immédiate.

Seulement deux messages peuvent être enregistrés, chacun peut être reproduit à l'aide d'un bouton poussoir dédié. Les deux LED de signalisation rouge et verte sont présentes, et il est possible de charger/décharger les mémoires à l'aide de l'interface série et d'un PC.

Fonctionnement

Bien que l'enregistrement et la lecture audio au format numérique semblent être quelque chose de banale de nos jours (il existe des millions de lecteurs MP3 dans les smartphones), nous vous confirmons que ce n'est pas du tout le cas.

En effet, suite à une simple pression sur le bouton PLAY, un processus complexe impliquant la mise en œuvre du matériel (hardware) et du logiciel (software) a été pensé par de nombreux techniciens et ingénieurs.

Soulignons que le traitement numérique audio est très peu abordé en tant que sujet d'étude, et que pour un néophyte il est difficilement compréhensible. Nombre de nos lecteurs nous ont posé des questions, essentiellement sur le fonctionnement d'un tel circuit ou encore sur des besoins de clarifications au niveau logiciel.

Nous allons donc aborder la description du firmware (programme) du microcontrôleur qui est un PIC18F2420. Celui-ci gère les modules, voici la liste des fonctions qu'il exécute :

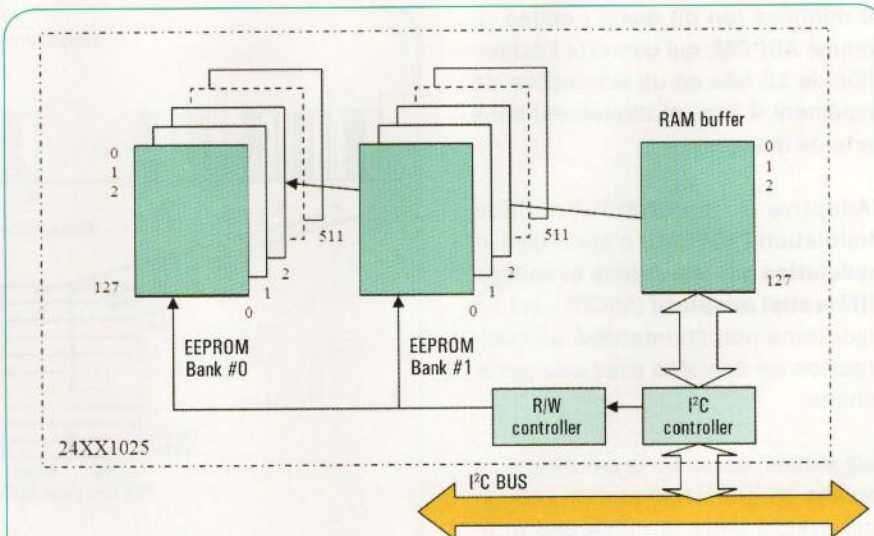


Figure 4 : structure interne d'un circuit mémoire série EEPROM de type 24XX1025. En plus des circuits de contrôles, il contient 2 bancs de mémoire constitués chacun de 512 pages (1024 pages). Chaque page a une taille de 128 octets. Notez aussi la présence d'une mémoire RAM tampon (RAM buffer) de 128 octets.

- gestion de la logique de fonctionnement de l'enregistrement (boutons, DIP switch et port série) ;
- gestion des périphériques ADC et PWM ;
- compression/décompression dans l'ADPCM des échantillons audio acquis ;
- gestion de la mémoire tampon RAM interne de 128 octets (bytes) servant d'interface avec la mémoire EEPROM ;
- gestion de la logique des pages de mémoire de l'EEPROM. Chaque mémoire dispose de 1024 pages de 128 octets chacune soit un total de 128 kB (kilobytes) ou 1 Mb (méga-bit).

Durant l'**enregistrement**, le **signal analogique** est acquis et **numérisé** par le **convertisseur A/D** du microcontrôleur et, après une série d'opérations de **mise en forme** du signal, celui-ci est envoyé vers les **mémoires EEPROM** pour être sauvegardé (enregistré) de façon permanente (voir la figure 5). Pour cela, un « **timer** » interne au microcontrôleur PIC analyse les intervalles de temps d'échantillonnage du signal analogique toutes les 62,5 µs pour une fréquence d'échantillonnage de 16 ksps (ou toutes les 125 µs pour une fréquence de 8 ksps).

Les données ainsi acquises sont transmises à un algorithme de compression

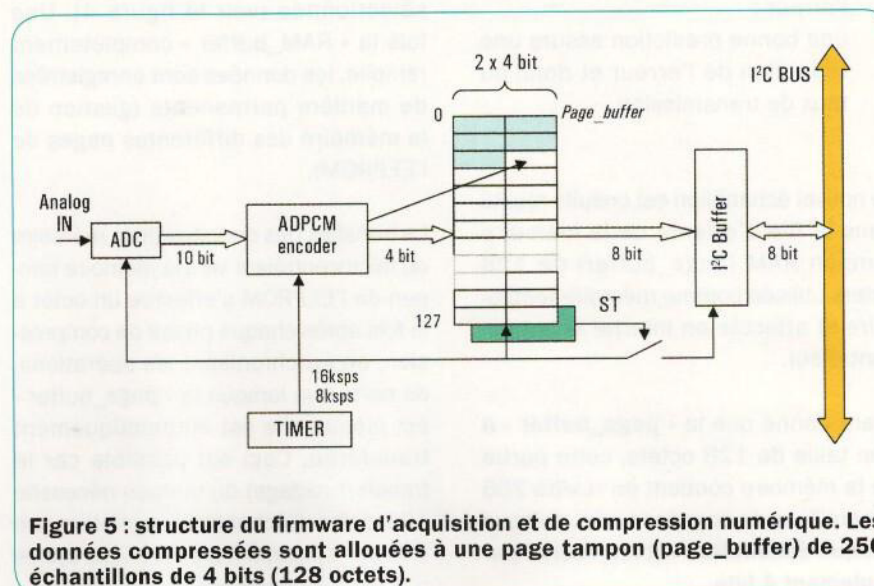


Figure 5 : structure du firmware d'acquisition et de compression numérique. Les données compressées sont allouées à une page tampon (page_buffer) de 256 échantillons de 4 bits (128 octets).

de données (on dit aussi « codec »), nommé **ADPCM**, qui convertit l'échantillon de **10 bits** en un échantillon de seulement **4 bits**, pratiquement sans perte de qualité.

L'Adaptive Differential Pulse Code Modulation (ADPCM), c'est-à-dire la **modulation par impulsions et codage différentiel adaptatif** (MICDA), est un algorithme non standardisé de compression de données avec une perte minime.

L'algorithme repose sur la présence d'un schéma de prédiction et d'un codage des erreurs entre la prédiction et le signal original. Les erreurs étant souvent infimes, une compression intéressante est possible grâce à la diminution des bits nécessaires à l'opération.

Dans le cas de l'ADPCM, le codage est dynamique et tient compte de la grandeur des erreurs en modifiant le pas de quantification. Lors du décodage, les erreurs sont ajoutées au signal issu de la prédiction pour obtenir un signal plus ou moins fidèle à l'original. Ce système est surtout utilisé pour les fichiers audio, en particulier les échantillons vocaux.

En résumé, l'ADPCM tient compte des propriétés suivantes :

- le signal est quasi-stationnaire ;
- la prédiction est adaptative ;
- la prédiction utilise les données précédentes et fournit de nouvelles données ;
- la prédiction tente de minimiser l'erreur ;
- une bonne prédiction assure une réduction de l'erreur et donc du taux de transmission.

Le nouvel échantillon est ensuite ajouté dans la file d'attente de la mémoire tampon RAM (page_buffer) de 128 octets utilisée comme mémoire temporaire et affectée en interne au microcontrôleur.

Étant donné que la « **page_buffer** » a une taille de 128 octets, cette partie de la mémoire contient en réalité 256 échantillons de données audio, puisque **chaque échantillon** est représenté par **seulement 4 bits**.

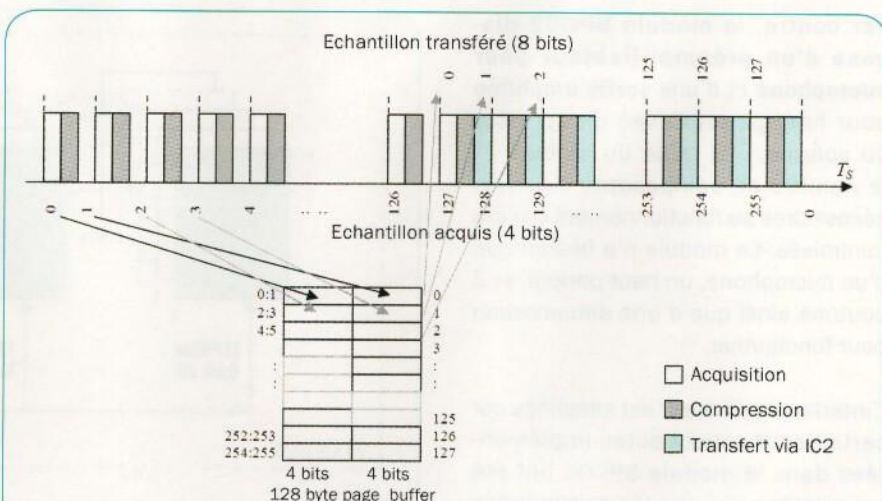


Figure 6 : Transfert de la « page_buffer » au bus I2C. Le transfert du premier octet vers la mémoire EEPROM initialise le cycle d'acquisition de l'échantillon n° 127 et se termine au cours du cycle d'acquisition de l'échantillon n° 255.

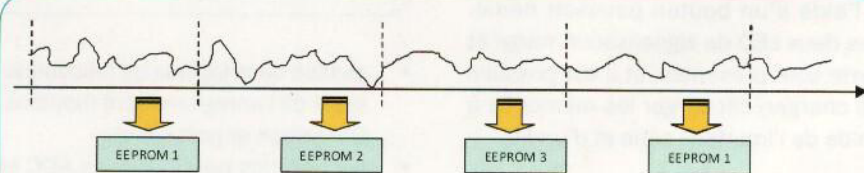


Figure 7 : séquence de mémorisation du signal d'entrée : les différents bancs de 256 échantillons sont stockés cycliquement dans la mémoire disponible (dans ce cas, il y a 3 mémoires EEPROM).

Par conséquent, grâce à la compression de l'algorithme ADPCM, il est possible de doubler la capacité de stockage d'une mémoire.

Chaque « page_buffer », via le bus I2C, est ensuite transférée à la mémoire RAM (RAM_buffer) interne de l'EEPROM sélectionnée (voir la figure 4). Une fois la « RAM_buffer » complètement remplie, les données sont enregistrées de manière permanente (gestion de la mémoire des différentes pages de l'EEPROM).

Le transfert des données de la mémoire du microcontrôleur vers la mémoire tampon de l'EEPROM s'effectue un octet à la fois après chaque phase de compression, en synchronisant les opérations, de sorte que lorsque la « page_buffer » est pleine, elle est automatiquement transférée. Ceci est possible car le transfert (vidage) du tampon nécessite 128 cycles d'échantillonnage alors que pour le remplir, il nécessite 256 cycles d'échantillonnage (voir la figure 6).

Une fois la « **RAM_buffer** » de la mémoire EEPROM remplie, les données sont réellement écrites. Cette opération nécessite environ **5 ms** pendant lesquelles la mémoire ne peut répondre à aucune commande et, par conséquent, il est impossible, par exemple, de reprendre un transfert de données du bus vers cette mémoire.

Pour cette raison, chaque fois qu'une mémoire est occupée à écrire une page, la page suivante (la 2^{ème}) est destinée à une autre mémoire, et lors de la phase d'écriture de la deuxième mémoire, la page suivante (la 3^{ème}) est dirigée vers la troisième mémoire et ainsi de suite, puis le cycle recommence.

Ainsi, les pages successives ne sont pas écrites dans une même mémoire physique, mais sont écrites de manière séquentielle sur les différentes mémoires présentes. Il en découle qu'il est nécessaire d'avoir au moins deux mémoires EEPROM présentes dans le montage.

Le lave-vaisselle qui parle

Fabriquer des appareils à synthèse vocale, capables de prononcer un ensemble de phrases préétablies selon des événements spécifiques, est relativement simple, surtout s'il s'agit de produire des milliers d'exemplaires. Dans ce cas, les composants sont programmés directement en usine lors du processus de fabrication à partir du silicium, faisant économiser au fabricant la phase de programmation.

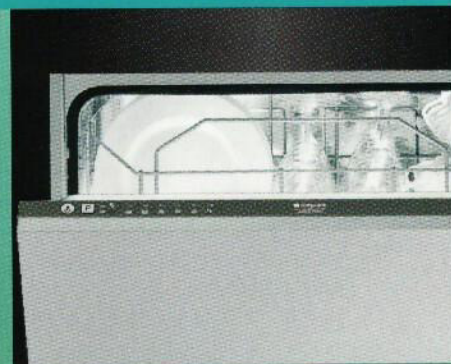
Théoriquement, le champ d'application de ces équipements est illimité et d'une grande utilité. Les applications les plus intéressantes, il y a quelques années, étaient les horloges parlantes qui n'ont pas connu un grand succès.

Mais aujourd'hui, grâce à l'introduction de l'intelligence artificielle qui bénéficie d'investissements considérables, les lave-vaisselles et plus généralement tous les appareils électroménagers (ou autres) permettent d'expliquer l'intérêt d'un programme et les différentes options les plus appropriées pour une situation donnée et cela en plusieurs langues.

Par exemple, Whirlpool annonce une nouvelle collaboration avec Amazon afin de lancer de nouveaux appareils électroménagers répondant à une commande vocale grâce à l'assistant personnel « Alexa » d'Amazon. Grâce à ce système, les utilisateurs peuvent par exemple demander à Alexa de démarrer ou d'arrêter le cycle de leur lave-linge ou de leur lave-vaisselle, de démarrer, régler et arrêter la minuterie du four, ou encore demander les températures actuelles du réfrigérateur.

Ces nouvelles fonctionnalités seront intégrées sur une large gamme d'appareils intelligents présentés au CES 2017 à Las Vegas. La technologie Alexa d'Amazon pour Whirlpool sera disponible début 2018.

D'autre part, le concurrent « Google Home » d'Alexa en collaboration avec LG permettra de contrôler un lave-linge, un lave-vaisselle ou encore un frigo à la voix. Concrètement, cette technologie permettra de contrôler les appareils à la voix via l'assistant vocal Google Assistant, et de consulter leurs statuts, par exemple le cycle de lavage en cours, directement sur un smartphone.



En fait, lorsque la première mémoire est occupée pendant la phase d'écriture, l'autre est disponible pour recevoir des données. À l'inverse, lorsque la seconde mémoire écrit son bloc de données, la première est disponible pour recevoir de nouvelles données.

Étant donné **qu'une page à la fois est écrite** dans l'EEPROM, un **nombre entier de pages** sera toujours enregistré, **même si la commande de STOP** (arrêt de l'enregistrement) est **activée** avant l'acquisition complète d'un ensemble de 256 échantillons. Cela signifie que nous enregistrons toujours des pistes d'une **durée multiple de 16 ms** (ou 32 ms dans le cas d'une fréquence de 8 ksps).

Comme nous l'avons mentionné, selon le module utilisé et le nombre de mémoires présentes, il est possible d'effectuer plusieurs enregistrements, jusqu'à un maximum de 64. Pour implémenter cette fonction, une page de la mémoire a été réservée (la 1024^{ème}

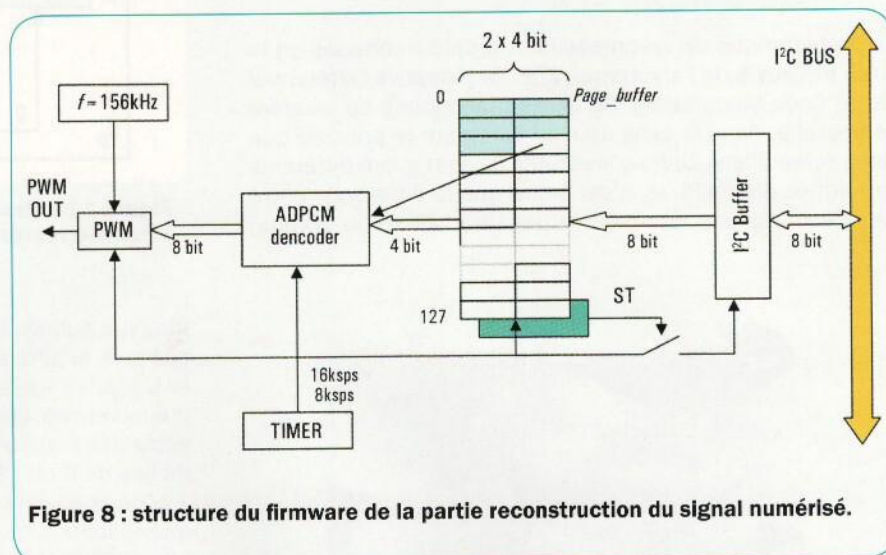


Figure 8 : structure du firmware de la partie reconstruction du signal numérisé.

page de l'EEPROM, la plus élevée) afin de créer une **table d'indexation** (on dit aussi table d'allocation des fichiers pour un disque dur).

Le tableau contient **64 cellules de 16 bits** et chaque **cellule contient le numéro de la page et l'adresse**

mémoire à laquelle l'enregistrement se termine.

Lorsque vous supprimez une piste, seule la référence dans la table d'indexation sera supprimée de la mémoire, l'enregistrement reste physiquement en mémoire.

La compression numérique

Vous entendez souvent parler de compression numérique de l'information, tels que les formats « .mp3 », « .ogg », etc., mais peu de personnes savent ce qui se cache derrière ces acronymes cryptiques. Tout d'abord, il est important de noter que la compression d'un signal numérique signifie la réduction du nombre d'octets qui constituent ce signal.

En d'autres termes, si vous avez besoin de 900 kilo-octets pour représenter numériquement 10 secondes de musique ou de paroles, la version compressée de cette même section peut être inférieure de moitié (400 kilo-octets) voire atteindre un dixième de la taille du fichier original (90 kilo-octets). Cela représente un avantage certain si vous voulez, par exemple, stocker des fichiers audio, car l'économie de la quantité de mémoire est importante.

Le premier inconvénient est que la compression numérique d'un signal audio implique principalement une perte de qualité sonore importante, mais tolérable. Le deuxième implique la nécessité d'avoir une configuration matérielle (hardware) suffisamment puissante pour pouvoir compresser/décompresser les données en temps réel. En fait pour simplifier, la compression ou la décompression d'un signal audio implique d'effectuer des opérations mathématiques complexes et lourdes sur des échantillons numérisés du signal audio analogique d'origine.

Nous ne prétendons pas vous enseigner tous ces processus, car il faudrait plusieurs centaines de pages, mais dans cet encadré, nous présentons une technique de compression « simple » utilisée dans le domaine audio et qui peut être implémentée sur un matériel peu performant en termes d'opérations mathématiques complexes, comme par exemple un microcontrôleur 8 bits de la famille des PIC 18FXXX.

Cette technique de compression « simple » consiste en la mise en œuvre de l'algorithme ADPCM (Adaptive Differential Pulse Code Modulation). Ce dernier fonctionne de manière temporelle, dans le sens où il se fonde sur le principe que deux échantillons audio consécutifs ne sont guère différents en termes d'amplitude, c'est-à-dire que la différence entre deux échantillons représentés par « n » bits (8 ou 16 bits)

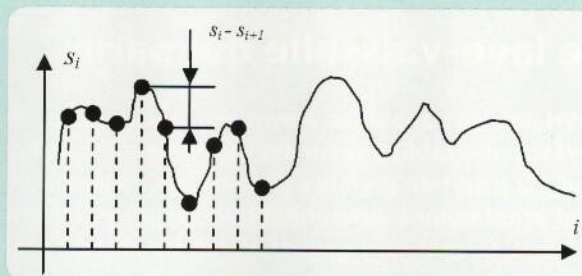


Figure 10 : les deux échantillons consécutifs « s_i » et « s_{i+1} », représentés par « n » bits, sont similaires en termes d'amplitude. La différence peut donc être représentée sur moins de « n » bits.

peut être représentée par « m » bits où « m » est inférieur à « n » (voir la figure 10).

Dans l'algorithme de la figure 11, la représentation des « m » bits n'est cependant pas la différence réelle entre deux échantillons, mais la différence « d » entre un échantillon et celui calculé au moyen d'une relation mathématique appelée prédicteur « s_p ». La différence « d » est quantifiée uniquement sur les niveaux « m » (typiquement 16 ou 32 correspondant à 4 ou 5 bits), nous obtenons ainsi le signal « t ».

Le pas de quantification « q » est variable, plus « d » est grande, plus le pas de quantification est grand. Cependant, il est nécessaire de s'assurer que les niveaux de « m » soient

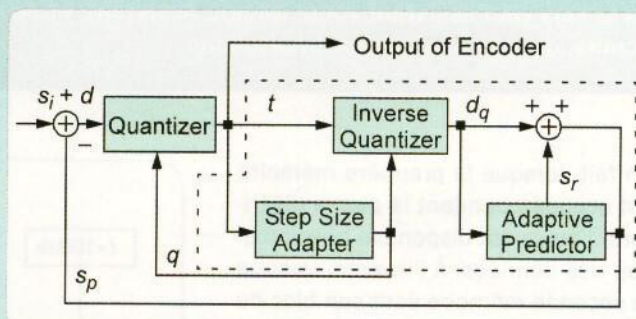


Figure 11 : structure d'un encodeur ADPCM. La zone en pointillée représente les différentes parties du décodeur.

toujours suffisants pour représenter toutes les valeurs possibles de la différence « d ». De cette manière, en mémorisant le signal « t » à la place du signal d'origine, il est possible d'économiser une grande quantité d'espace mémoire, car seuls des mots d'une taille de 4 ou 5 bits sont sauvegardés au lieu de 8 ou 16 bits. En suivant le synoptique de la figure 11, nous voyons que le signal « t » est envoyé au bloc de quantification inverse pour reporter le signal « t » de « m » bits aux « n » bits originaux. Ainsi, le signal « dq » sortant du bloc de quantification inverse arrive sur l'entrée du bloc de prédiction de l'échantillon. Ce bloc est très complexe, dans le standard ADPCM l'échantillon prédéterminé est obtenu par une moyenne pondérée entre les 6 derniers échantillons et les 2 derniers échantillons prédéterminés « s_p ».

Cela est difficile à mettre en œuvre avec un simple microcontrôleur tel qu'un PIC18FXXX. Pour cette raison, dans le programme que nous avons mis en œuvre, nous avons choisi



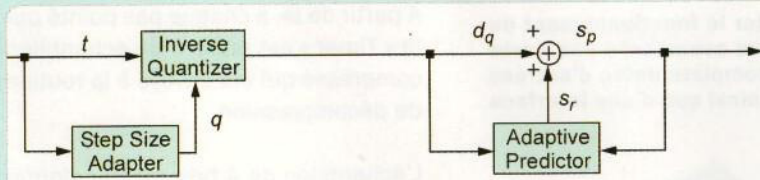


Figure 12 : le décodeur ADPCM.

d'utiliser une logique plus simple pour calculer l'échantillon « s_p », nous effectuons la somme de l'échantillon d'entrée « s_i » et de la différence du signal déquantifié « d_q ». Le bloc en pointillé sur la figure 11 constitue le décodeur pour le signal ADPCM. En fait, l'étage de décompression du signal, représenté en figure 12, est constitué par les mêmes blocs de la figure 11 dans la zone en pointillée.

Comme vous pouvez le voir, la partie décompression du signal est plus simple que la partie compression, car elle ne comprend pas le bloc de quantification à pas variable. Un tel décodeur reçoit sur son entrée des échantillons compressés représentés sur « m » bits et les convertit en échantillons de « n » bits.

L'implémentation de cette fonction en langage « C » est bien expliquée dans la note d'application AN643 de Microchip intitulée « ADPCM using PIC microcontroller ». Nous nous sommes basés sur cette note d'application, mais dans notre cas le code a été transposé en assembleur et les formats des données pris en charge ont été modifiés afin d'optimiser la vitesse d'exécution. Notre fonction d'encodage (« `enc_f()` ») dans le fichier « `asm_func.asm` » accepte en entrée une variable de 10 bits et restitue un codage compressé sur 4 bits. La fonction de décodage (« `dec_f()` ») dans le fichier « `asm_func.asm` » accepte sur son entrée un codage compressé sur 4 bits et restitue une variable décompressée sur 10 bits.

Nous ne listons pas le code source de la note d'application AN643, mais nous allons consacrer quelques explications à la logique utilisée pour implémenter les blocs de

quantification, cette application est très efficace et peut être utilisée dans d'autres applications.

Supposons que nous voulions quantifier un signal « a » sur n bits et que chaque niveau de quantification a une amplitude « b ». Il est évident qu'il ne peut pas être plus grand que la quantité maximale « u » représentée avec « n » bits.

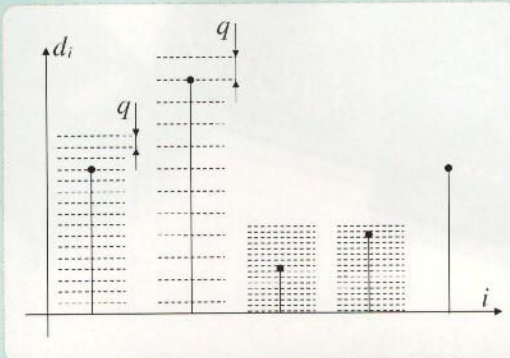


Figure 13 : pas de quantification « q » variable appliqué à la différence « d_i » de l'échantillon. Les niveaux plus grands du signal utilisent un plus grand pas de quantification « q » pour faire en sorte que les « m » bits (dans la figure ci-contre $m = 4$ pour 16 niveaux) représentant le signal soient toujours suffisants.

Si nous comparons « a » avec la moitié de u , dans le cas où $a > u$, il faut prendre le MSB du résultat et calculer « $a = a - u$ ».

Si nous comparons de nouveau « a » avec « $u/2$ », et si $a > u$ alors il faut prendre le MSB-1 du résultat et calculer « $a = a - u$ ». Nous continuons ainsi de suite les comparaisons jusqu'à atteindre les « n » bits.

En effectuant les « n » pas, nous pouvons quantifier la valeur de « a ». Cette méthode, également utilisée dans la conversion de type « ADC SAR », s'appelle la recherche binaire ou recherche dichotomique. Elle est largement utilisée en raison de son efficacité.

Listing 1 : Exemple de recherche binaire ; une donnée « a » de 16 bits est quantifiée sur 4 bits (codage).

```
;
unsigned int  a, u;      // a, u sont sur 16 bits
;
;
u = 0x7FFF;             // initialise u
code = 0x00;            // initialise le codage

if(a > u)
{
    // MSB: bit #3
    a = a - u;
    code = code | 0x08;
}
```

```
u = u >> 1;             // u = u / 2

if(a > u)
{
    // bit #2
    a = a - u;
    code = code | 0x04;
}
u = u >> 1;             // u = u / 2

if(a > u)
{
    // bit #1
    a = a - u;
    code = code | 0x02;
}
u = u >> 1;             // u = u / 2

if(a > u)
{
    // LSB: bit #0
    code = code | 0x01;
}
```

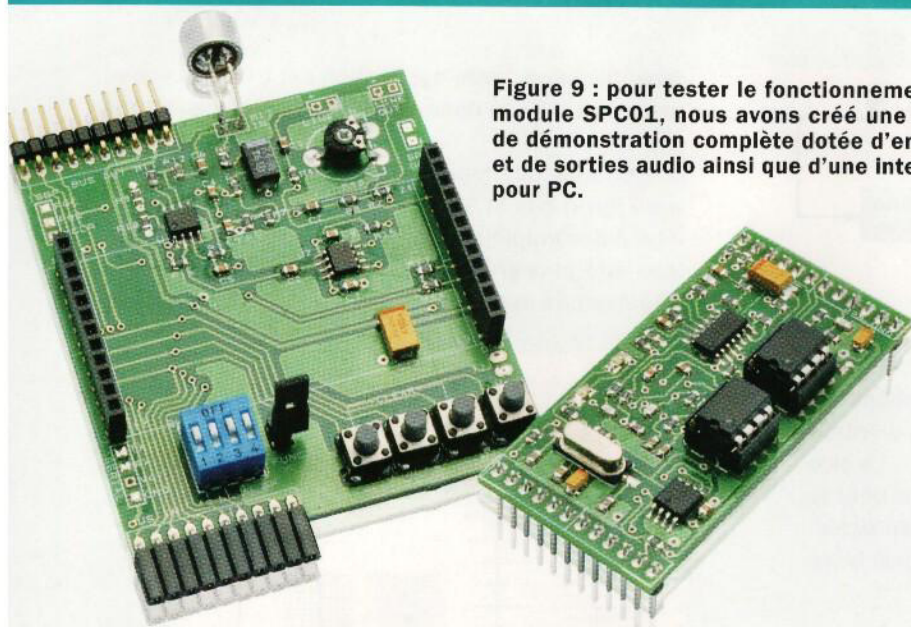



Figure 9 : pour tester le fonctionnement du module SPC01, nous avons créé une carte de démonstration complète dotée d'entrées et de sorties audio ainsi que d'une interface pour PC.

Les pistes suivantes sont enregistrées à partir de la dernière page mémoire utilisée.

Lors de la lecture d'une piste, une routine spéciale extrait de la table d'indexation le début et la fin de la piste pour la reproduire.

Il en découle que seules les données nécessaires sont lues dans la mémoire.

Le processus de lecture s'effectue dans l'ordre inverse par rapport à l'enregistrement.

Le flux de données se déplace des mémoires EEPROM vers le tampon (page_buffer) et lorsqu'elles passent par l'algorithme de décompression, elles sont envoyées au périphérique PWM pour être ensuite filtrées à l'extérieur (voir la figure 8).

Bien évidemment, pendant la reproduction, vous devez lire les pages de données cycliquement d'une mémoire à l'autre, selon la séquence dans laquelle elles ont été écrites. Chaque page est temporairement transférée vers la « page_buffer » interne du microcontrôleur.

À partir de là, à chaque pas pointé par le « Timer » est prélevé un échantillon compressé qui est envoyé à la routine de décompression.

L'échantillon de 4 bits est transformé en 8 bits pour moduler le rapport cyclique (duty-cycle) du périphérique PWM du microcontrôleur qui est utilisé comme convertisseur D/A (digital/analogique).

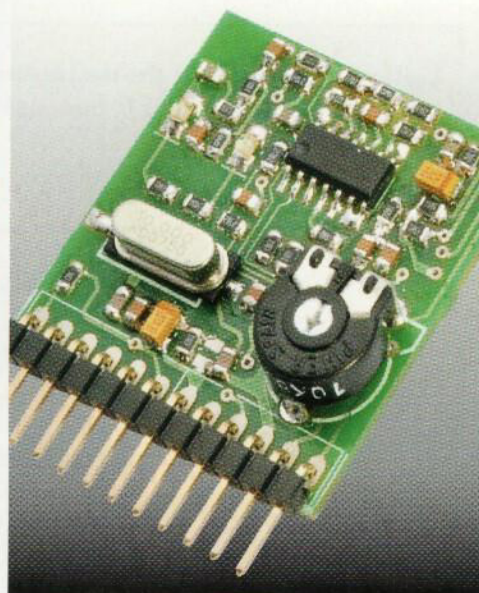
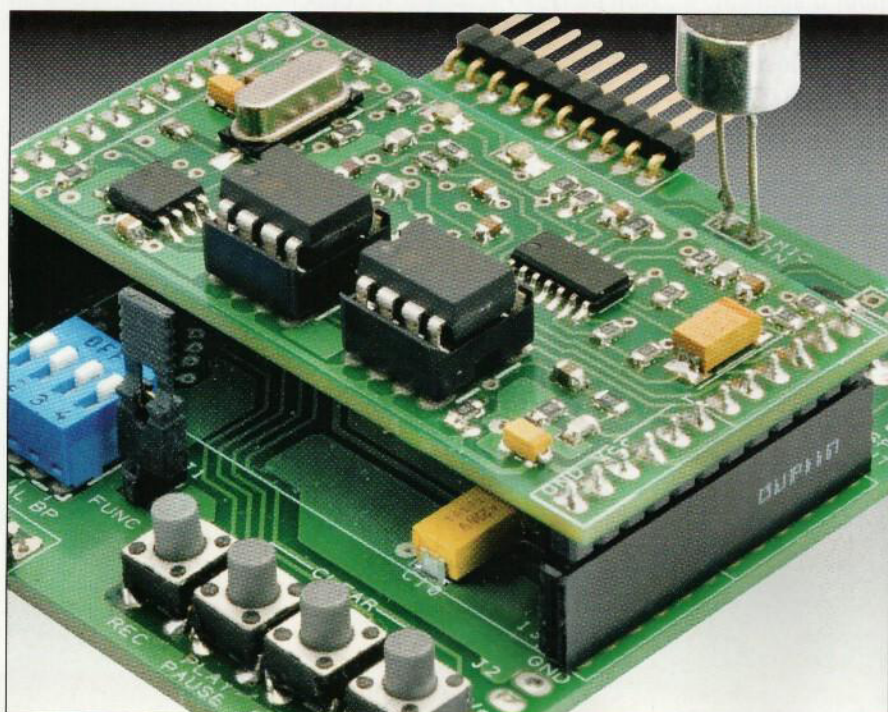
À ce point de cette première partie de l'article, nous pouvons conclure la présentation générale de notre système et aussi la partie théorie de fonctionnement.

Dans les différents encadrés, nous avons examiné succinctement la notion de compression numérique, en fournissant le minimum d'informations théoriques nécessaires pour tous ceux qui veulent en savoir plus.

Dans la seconde partie, nous étudierons les schémas électriques des deux modules ainsi que celui de la carte de démonstration, et nous aborderons la réalisation pratique du montage.

Enfin, dans la troisième partie, nous nous consacrons à la mise en œuvre du firmware du microcontrôleur.

Connaître et comprendre les routines que vous utiliserez vous permettra de réaliser d'autres projets similaires pour d'autres applications variées. ■





Apprenez à maîtriser

KiCad EDA

**Première
partie**

de **Francesco Ficili et Vincenzo Germano**

Nous commençons ce nouveau cours consacré à la découverte de KiCad EDA, un logiciel de CAO Open Source qui permet de réaliser le développement et la fabrication de cartes électroniques, c'est-à-dire la création de schémas électroniques et de circuits imprimés ainsi que les fichiers Gerber pour la fabrication.

Étant des experts en électronique, sans aucune prétention, nous savons que la première étape primordiale de tout projet électronique est la réalisation du circuit imprimé.

De nos jours, le développement d'une carte électronique est de plus en plus complexe en raison de la miniaturisation des composants, ainsi sa conception ne peut aboutir que par des méthodes de fabrications professionnelles. Cela nécessite l'utilisation d'outils de conception assistée par ordinateur (CAO) plus ou moins complexes et donc souvent inabordable financièrement pour l'amateur électronicien.

La conception assistée par ordinateur existe depuis des décennies dans divers secteurs, allant du bâtiment à la mécanique et à l'aéronautique, et bien sûr à l'électronique. Les programmes de CAO électroniques (dans notre cas KiCad EDA) existent depuis au moins trente ans, mais ce sont pour la plupart des produits professionnels très coûteux et donc inaccessibles pour la majeure partie du public.

Pendant plusieurs années, des développeurs et amateurs passionnés ont créé des logiciels de conception simples et gratuits ou très peu coûteux, qui ont démocratisé la CAO dans les ordinateurs des amateurs électroniciens.



kicad

Figure 1 : le logo de KiCad.

Le secteur professionnel de la CAO électronique est dominé par trois grands acteurs qui sont OrCAD de Cadence, Altium Designer, anciennement Protel, développé à l'origine par une société Australienne et Eagle PCB récemment racheté par Autodesk.

Ces outils, qui sont incontestablement très performants, représentent des coûts que seules les entreprises ayant un certain chiffre d'affaire peuvent supporter. Les licences de ces logiciels peuvent coûter de plusieurs milliers d'euros (Eagle étant le moins « cher ») à quelques dizaines de milliers d'euros pour Altium Designer.

Heureusement, la diffusion de logiciel Open Source a également touché le monde de l'électronique, aidant l'amateur qui veut concevoir et fabriquer des

cartes électroniques, mais aussi les petites sociétés qui ont besoin de concevoir des cartes électroniques sous la forme de prototypes ou de petites séries et qui n'amortissons pas l'investissement de l'achat d'un logiciel de CAO professionnel.

Parmi les logiciels Open Source de CAO, KiCad EDA se démarque. Il s'agit d'une suite de conception et fabrication de cartes électroniques entièrement ouverte et multiplateforme, développée et distribuée sous licence GNU par Jean-Pierre Charras.

De toute évidence, le niveau de ce logiciel ne peut pas être comparé à des outils tels que Altium Designer ou Orcad, mais il n'a rien à envier à des logiciels payants comme Autodesk Eagle.

Bien qu'entièrement Open Source (et donc gratuit), l'outil est extrêmement puissant et bénéficie d'une réputation car il est utilisé par l'Université de Grenoble où il a été développé en 1992 par Jean-Pierre Charras qui été enseignant, par le CERN qui a largement contribué au développement de KiCad et aussi par la Fondation RaspberryPi.

En figure 2, vous pouvez observer la croissance de l'utilisation de KiCad à partir de 2007 (courbe en bleu) par rapport aux autres logiciels de CAO payants.

KiCad est utilisé par de nombreux amateurs et petites entreprises du monde entier, grâce à sa facilité d'utilisation qui le caractérise, de plus il est totalement gratuit et prend au fil des ans de plus en plus de parts du marché par rapport aux solutions de CAO professionnelles.

Introduction à KiCad EDA

KiCad est un outil de CAO spécialement conçu pour dessiner les schémas électroniques et les circuits imprimés. À l'heure actuelle, KiCad est distribué sous la licence GNU GPLv2 totalement gratuite et il fonctionne sous Windows, Linux et Mac OS X.

KiCad n'a aucune limitation de taille des circuits imprimés et peut facilement gérer jusqu'à 32 couches de cuivre, jusqu'à 14 couches techniques, et 4 couches auxiliaires. KiCad peut créer tous les fichiers nécessaires à la génération de cartes électroniques, tels que la création de composants et

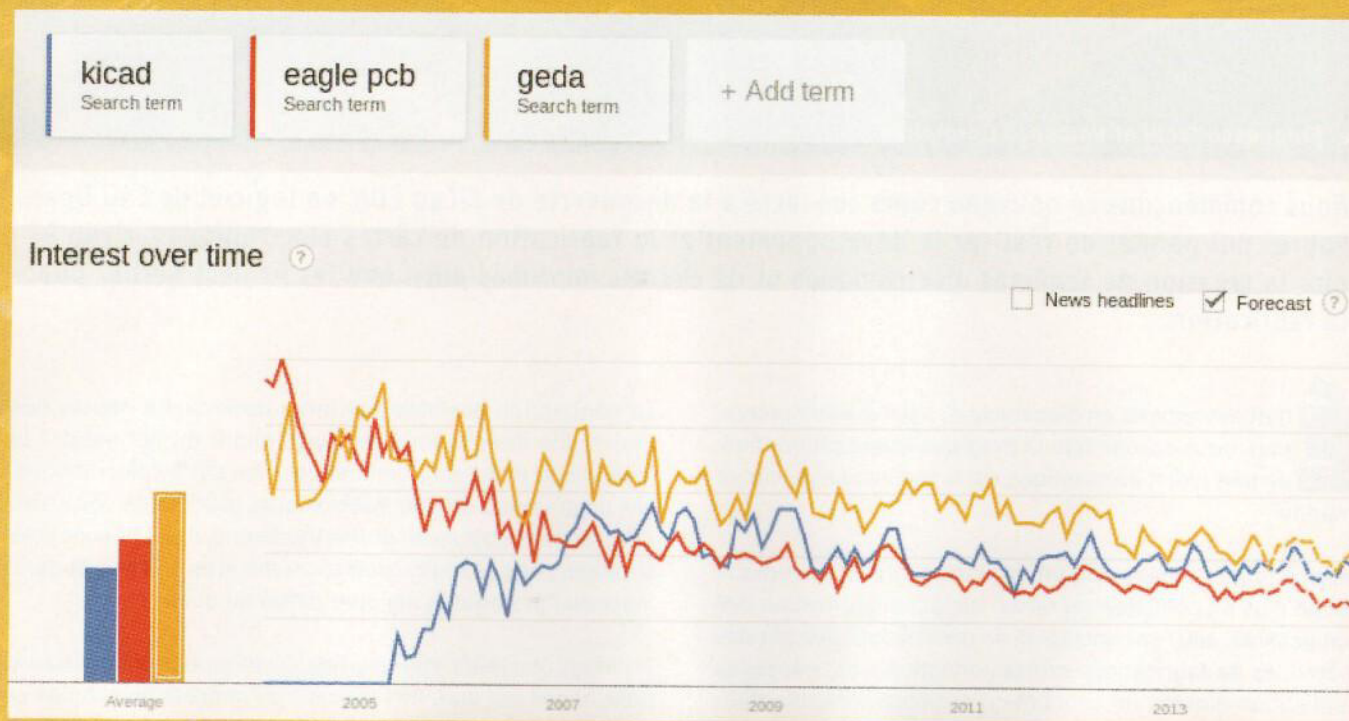


Figure 2 : répartition de KiCad par rapport aux autres logiciels de CAO électronique ayant des performances similaires.

les fichiers Gerber pour la fabrication du circuit imprimé, le perçage et l'implantation des composants.

KiCad dispose d'un gestionnaire intégré de bibliothèque de composants, doté d'un éditeur pour le schéma et l'empreinte physique des composants. De plus, il est possible de visualiser les fichiers Gerber (GerberView) et d'autres outils comme « PCB Calculator » permet le calcul de la largeur des pistes, l'isolation électrique, etc.

KiCad est un environnement de développement intégré (IDE) et se compose d'un ensemble de logiciels interdépendants inclus dans un gestionnaire de projet (Project Manager), qui gère les fichiers du projet et lance les programmes individuels à partir d'une seule interface utilisateur.

À l'intérieur du gestionnaire de projet (Project Manager), il existe 4 programmes principaux décrits ci-après :

- **Eeschema** : c'est l'éditeur de schémas électroniques, il intègre aussi l'éditeur pour les bibliothèques de symboles des composants. Les fichiers de Eeschema sont compatibles d'un système d'exploitation à un autre. Il s'agit d'une application intégrée à partir de laquelle toutes les fonctions de dessin, de contrôle, de mise en page, de gestion des bibliothèques et d'accès au logiciel de conception de circuit imprimé sont accessibles. Eeschema est conçu pour fonctionner avec PcbNew, le logiciel d'édition de circuit imprimés de KiCad. Il permet aussi d'exporter des fichiers « Netlist » en listant toutes les connexions électriques, pour d'autres logiciels. Eeschema inclut un éditeur de symboles schématiques de composants, qui permet de créer, modifier des composants et de gérer les bibliothèques. Il intègre aussi plusieurs autres fonctions essentielles pour la création de schémas électroniques modernes, telles que la vérification des règles électriques ou ERC (Electrical Rules Check) pour le contrôle des connexions manquantes ou incorrectes, l'exportation de fichiers de tracé en

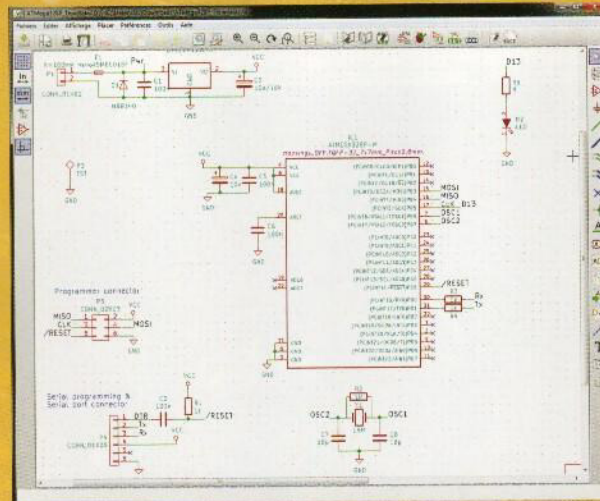
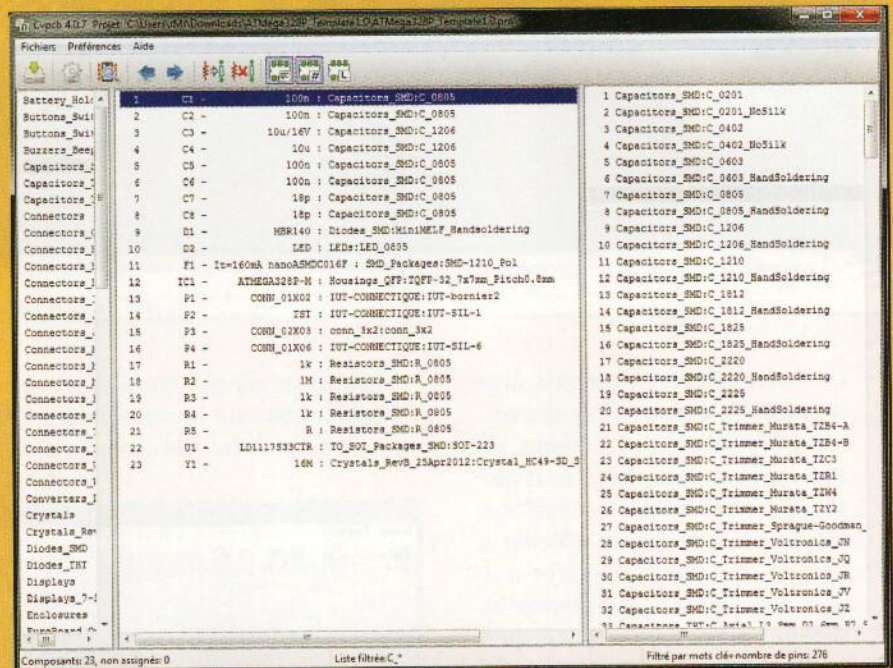


Figure 3 : exemple de schéma électronique réalisé avec Eeschema.

Figure 4 : exemple d'association entre le composant et un type d'empreinte (footprint).



plusieurs formats (Postscript, PDF, HPGL, SVG), ou encore la génération de la liste des composants du projet ou BOM (Bill of Materials). En figure 3 vous apercevez une capture d'écran d'une partie d'un schéma électronique réalisé avec Eeschema ;

- **CvPcb** : c'est l'outil qui permet d'associer les composants de votre schéma aux empreintes des composants utilisées pour le dessin du circuit imprimé. Cette association est ajoutée au fichier « Netlist » créé par Eeschema. Ce fichier spécifie quelle empreinte

du circuit imprimé est associée à chaque composant du schéma seulement si le champ « Footprint » (empreinte) du composant est initialisé. C'est le cas lorsque les empreintes des composants sont associées pendant la saisie du schéma, en fixant le champ « Footprint » du composant, ou qu'il est déjà fixé dans la bibliothèque schématique lors du chargement du symbole. CvPcb permet le filtrage de la liste d'empreintes, la visualisation de l'empreinte et de son modèle 3D pour aider à s'assurer que la bonne empreinte sera associée à chaque composant.

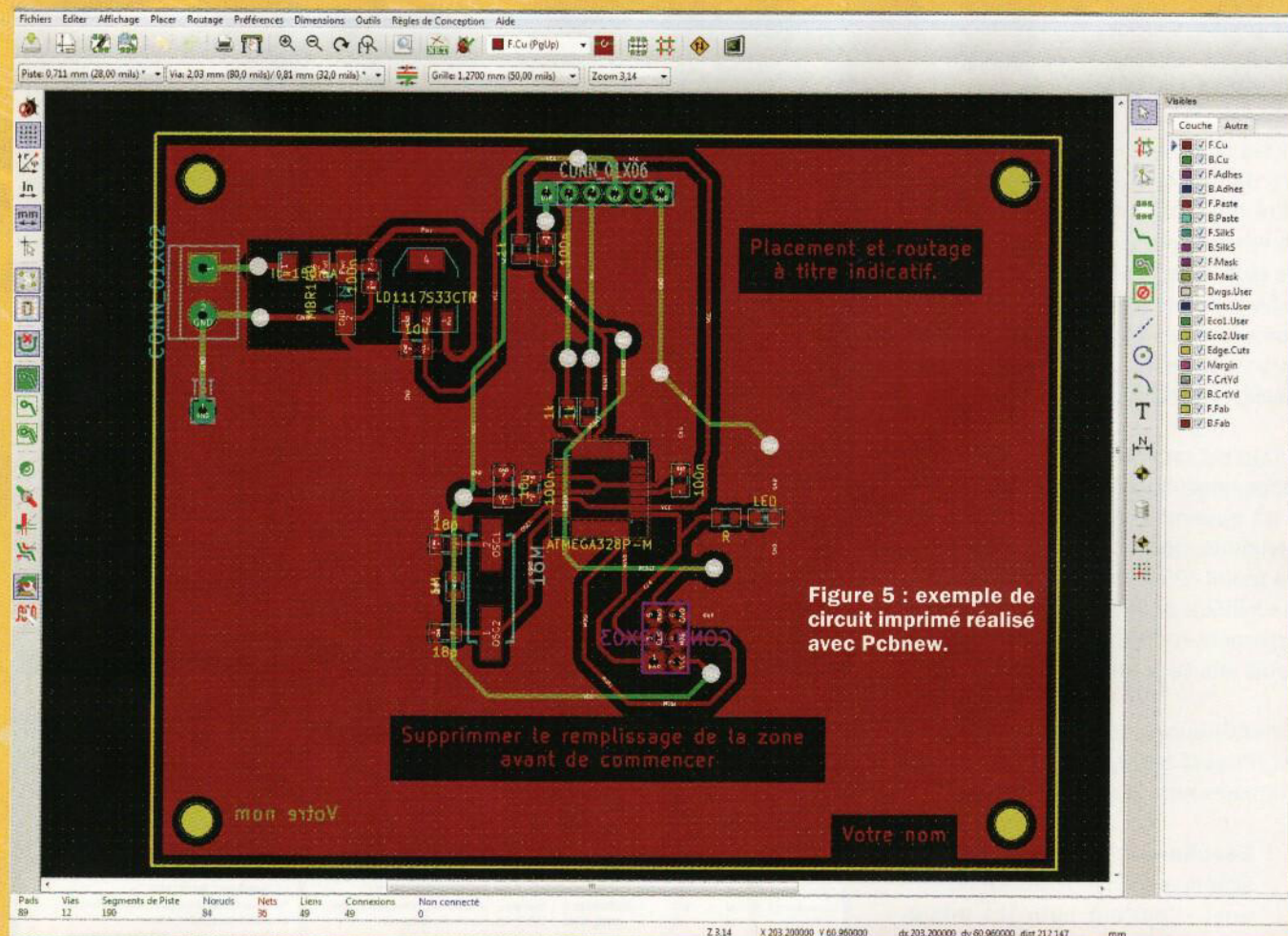


Figure 5 : exemple de circuit imprimé réalisé avec Pcbnew.

Les composants peuvent être associés à leurs empreintes correspondantes manuellement, ou automatiquement. CvPcb vous permet de voir la liste des empreintes disponibles, et de les afficher à l'écran, pour vous assurer que vous associez la bonne empreinte. CvPcb ne peut être lancé qu'à partir de Eeschema, à partir de la barre d'outils supérieure, soit lorsque Eeschema est démarré à partir du gestionnaire de projet KiCad, soit lorsque Eeschema est lancé comme une application indépendante. La figure 4 montre une capture d'écran de la liste des symboles des composants associés aux empreintes, éditée avec Cvpcb ;

- **Pcbnew** : c'est l'outil de création de circuits imprimés. Il est utilisé en combinaison avec la saisie de schéma Eeschema pour créer des circuits imprimés. Pcbnew gère les bibliothèques d'empreintes, chaque empreinte est le dessin d'un composant physique, incluant

les pastilles qui établissent les connexions sur le circuit imprimé. Les empreintes nécessaires sont

automatiquement chargées lors de la lecture de la « Netliste ». Toute modification d'empreintes

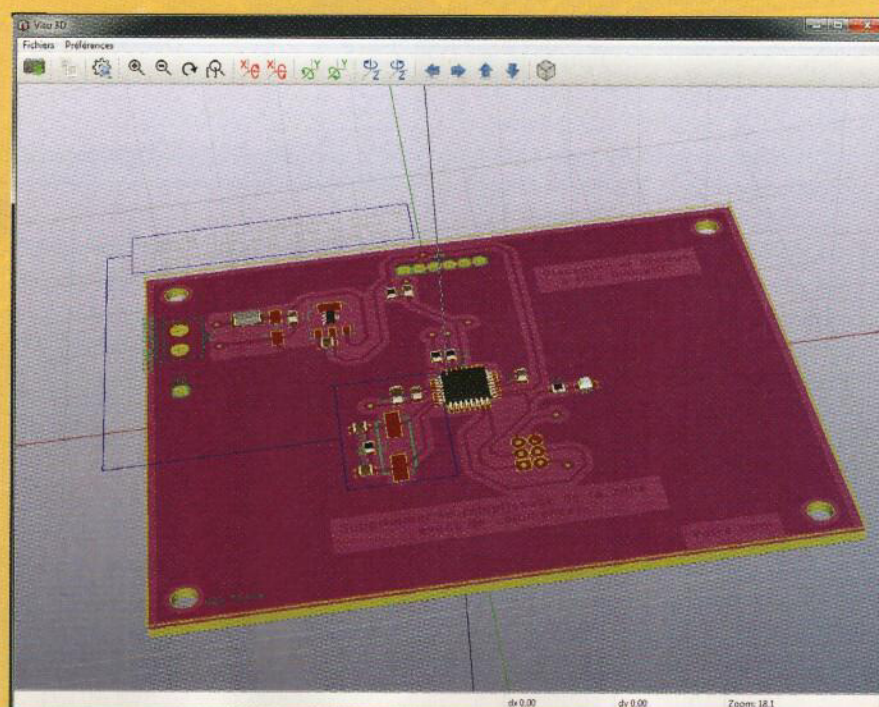


Figure 5A : visualisation en 3D du circuit imprimé réalisé avec Pcbnew.

ou d'annotation qui est effectuée dans le schéma est mise à jour dans Pcbnew grâce à la régénération de la « netliste » et de sa relecture dans Pcbnew. Il dispose d'un outil de vérification des règles de conception (DRC), qui évite les problèmes de distance minimale entre les pistes ou les pastilles, et évite également les mauvaises connexions ou celles qui sont « en l'air ». Lors de l'utilisation du routeur interactif, il vérifie en permanence les règles de conception et vous aide à tracer automatiquement des pistes individuelles. Pcbnew fournit un affichage du chevelu, montrant les liaisons des broches des empreintes des composants qui sont connectées sur le schéma. Ces connexions suivent dynamiquement le déplacement des empreintes et le tracé des pistes. Pcbnew dispose également d'un autorouteur simple mais efficace, pour aider à la réalisation du circuit imprimé. Il permet aussi l'exportation vers d'autres autorouteurs plus avancés. La plus petite unité dans Pcbnew est de 1 nanomètre. Toutes les dimensions sont mémorisées en nanomètres. Pcbnew peut gérer jusqu'à 32 couches de cuivre, 14 couches techniques (sérigraphies, masques de soudure, adhésif de composant, pâte à souder et découpes des contours), plus 4 couches auxiliaires (dessins ou commentaires). Il possède également un outil de visualisation en trois dimensions du circuit imprimé (3D PCB). La figure 5 montre une capture d'écran d'un circuit imprimé réalisé avec Pcbnew ;

- **Gerberview** : il s'agit d'un visualiseur de fichier Gerber au format RS274X et qui peut également afficher des fichiers de perçage à partir de Pcbnew au format « Excellon ». Il accepte jusqu'à 32 fichiers (fichiers Gerber et/ou Drill files). Les fichiers peuvent être affichés en mode transparence ou en mode empilé.

En outre, d'autres outils sont disponibles dans la dernière version :

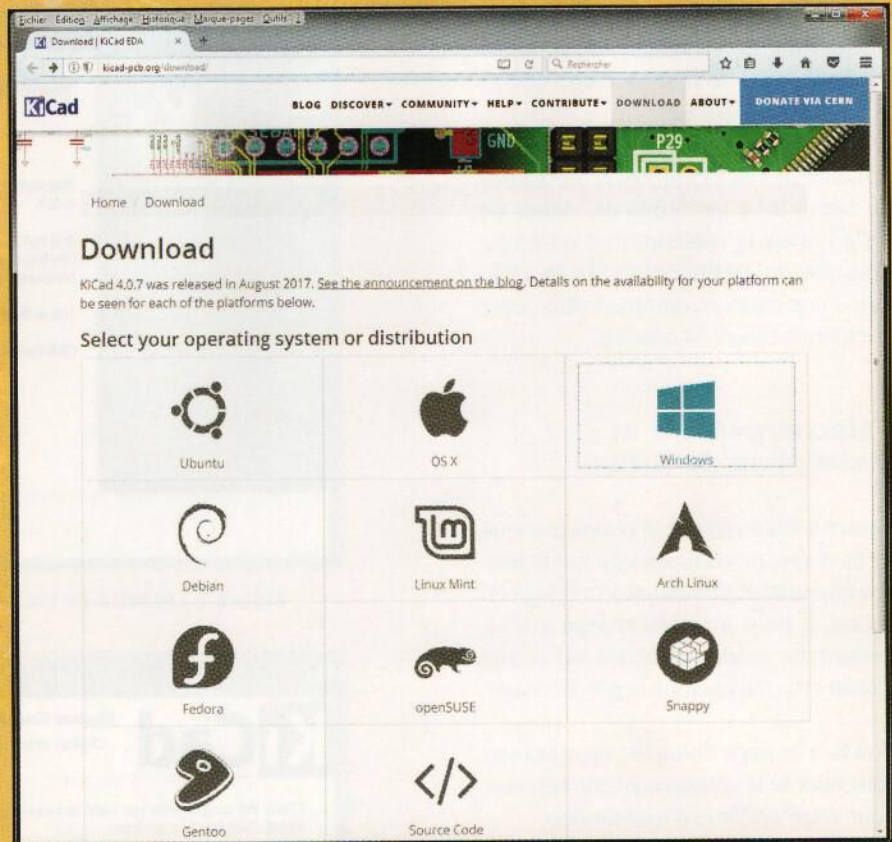


Figure 6 : la page web de téléchargement de KiCad contient différentes versions fonctionnant avec la pour la plupart des systèmes d'exploitation.

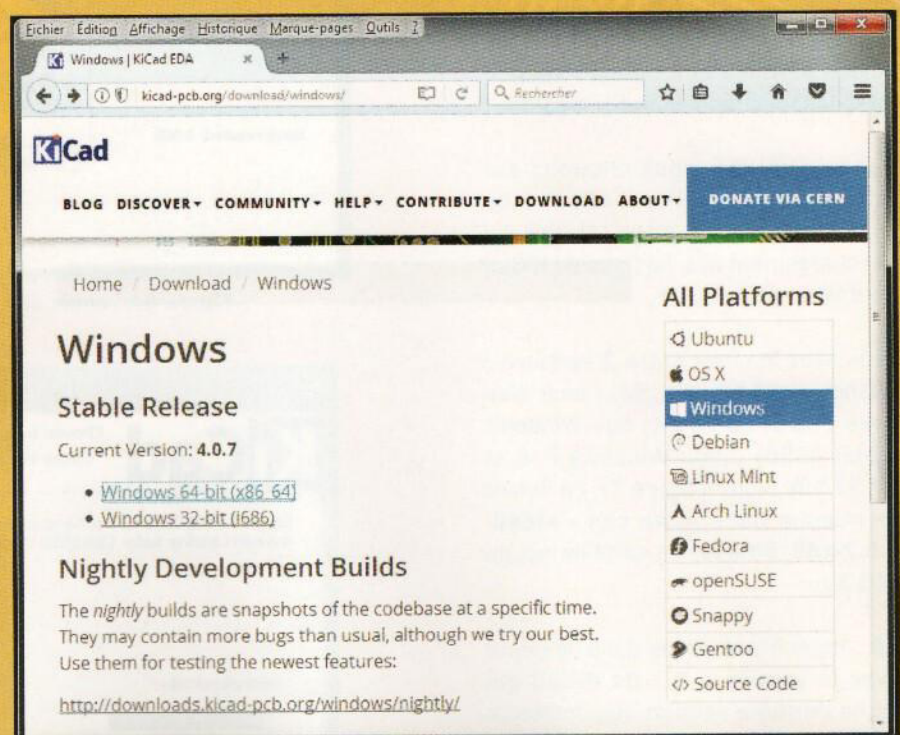


Figure 7 : Ici la page relative de téléchargement pour Windows s'ouvre après avoir cliqué sur l'icône Windows de la figure 6.

- **Bitmap2Component** : c'est un convertisseur d'images bitmap en symboles ou empreintes, utile pour insérer des logos dans les PCB ;
- **PcbCalculator** : il permet de calculer la largeur des pistes, l'isolation électrique, le code couleur et les lignes de transmissions ;

- **Wings 3D** : c'est un logiciel qui permet de visualiser en 3D le circuit imprimé réalisé avec Pcbnew (voir la figure 5A).

Au cours de ces leçons, nous étudierons en détail tous les outils de travail de KiCad, avec la réalisation d'un projet pratique qui débouchera sur la création d'une carte de démonstration pour microcontrôleurs Microchip.

Téléchargement et installation de KiCad

Passons maintenant à la phase pratique de ce cours, en commençant par le téléchargement et l'installation du logiciel KiCad. Il peut être téléchargé gratuitement sur le web en allant sur le site officiel <http://kicad-pcb.org/download/>.

Une fois la page chargée, vous pouvez sélectionner la version qui vous convient pour votre système d'exploitation.

Il existe des versions en téléchargement pour Windows, Linux et Mac OS X. Vous pouvez aussi télécharger le code source de KiCad afin que quiconque puisse développer sa propre version avec des fonctionnalités sur mesure.

Dans notre cas, nous cliquons sur l'icône Windows qui apparaît en bleu sur la figure 6. La page relative au téléchargement des versions de KiCad pour Windows s'ouvre.

Vous avez le choix entre 2 versions : « Windows 64-bit (x86_64) » pour Windows 7, 8 et 10 64 bits ou « Windows 32-bit (i686) » pour Windows 7, 8 et 10 32 bits (voir la figure 7). Le fichier se nomme dans notre cas « **kicad-4.0.7-x86_64.exe** » et sa taille est de 705 Mo.

NB : nous travaillerons dans ce cours avec la version 4.0.7 de KiCad qui est la dernière version disponible fin août 2017. Des versions plus récentes seront disponibles au fur et à mesure de l'avancement des leçons.

Téléchargez toujours la dernière version. Le principe d'installation et d'utilisation de KiCad sera identique à ce qui sera décrit dans ce cours.



Figure 8 : fenêtre de bienvenue d'installation de KiCad.

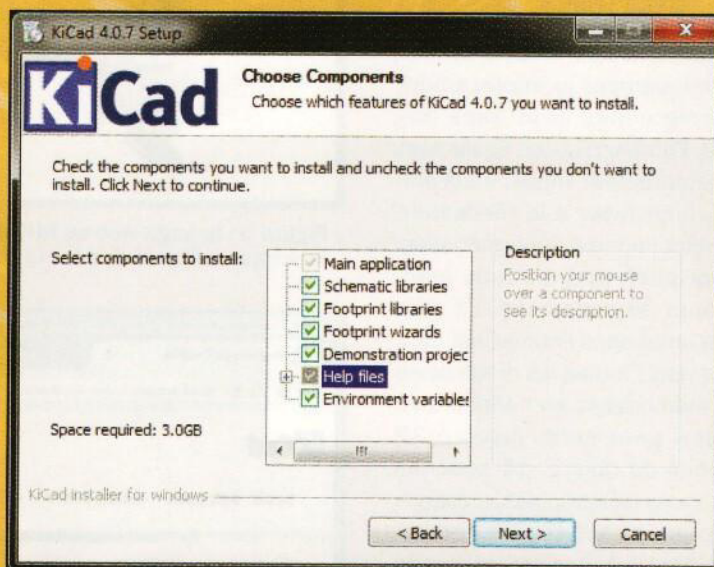


Figure 8A : choix des composants à installer.

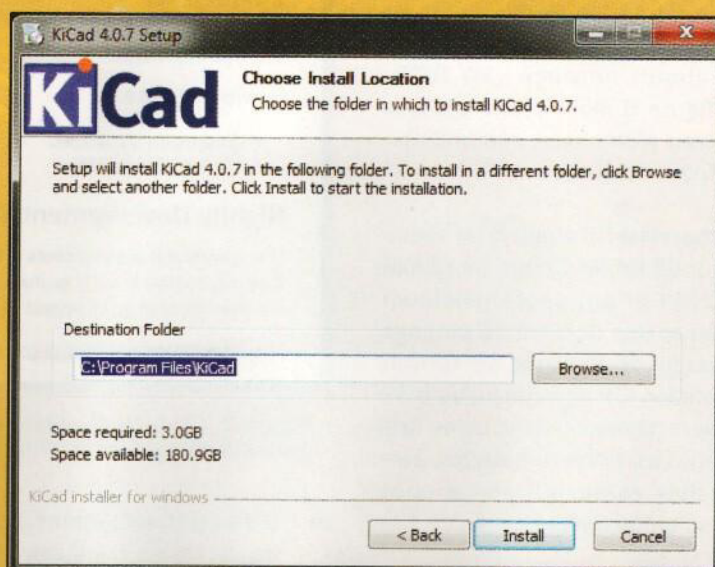


Figure 8B : emplacement d'installation de KiCad.

Après avoir cliqué sur le lien du fichier correspondant à votre système d'exploitation, le téléchargement commence, cela peut prendre plusieurs minutes en fonction de la vitesse de votre connexion internet.

Une fois le téléchargement terminé, exécutez le fichier « kicad-4.0.7-x86_64.exe » en tant qu'administrateur. Vous devez obtenir la fenêtre de bienvenue d'installation que vous pouvez voir en figure 8.

Cliquez sur le bouton « Next » pour continuer. Une nouvelle fenêtre s'ouvre en vous demandant les composants que vous souhaitez installer.

Cliquez sur le signe « + » à côté de « Help File », qui correspond à la documentation, et cochez la case « French », décochez les cases des autres langues, cela occupera moins d'espace disque (voir la figure 8A).

Notez que par défaut Kicad détecte la langue du système et s'installe en langage « Français ». Si cela n'est pas le cas, vous pouvez par la suite modifier la langue dans le menu « Préférences » → « Langue ».

Cliquez ensuite sur le bouton « Next », la figure 8B s'affiche et vous invite à sélectionner l'emplacement de l'installation de Kicad. Laissez la destination par défaut, sinon choisissez un dossier à votre convenance à l'aide du bouton « Browse ».

Cliquez sur le bouton « Install » pour commencer le processus d'installation (voir la figure 8C). Au bout de quelques instants, la fenêtre de la figure 8D s'affiche vous indiquant que l'installation de Kicad est terminée.

Cochez la case d'installation de « Wings 3D », puis cliquez sur le bouton « Finish ». Wings 3D permet de visualiser un circuit imprimé en 3D. Assurez-vous que vous êtes connecté à internet.

En cliquant sur « Finish » vous êtes redirigé vers la page web de téléchargement de Wings (voir la figure 9A).

Dans notre cas, nous choisissons la version « Stable release 2.1.5 » pour

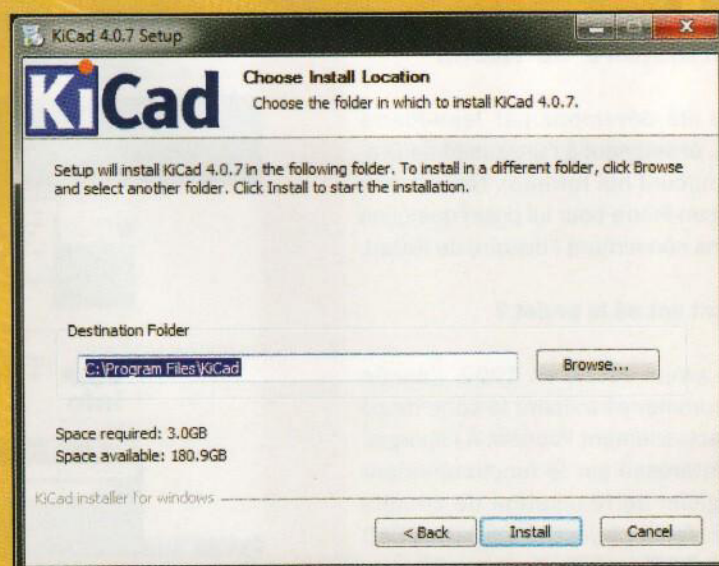


Figure 8C : processus d'installation en cours.

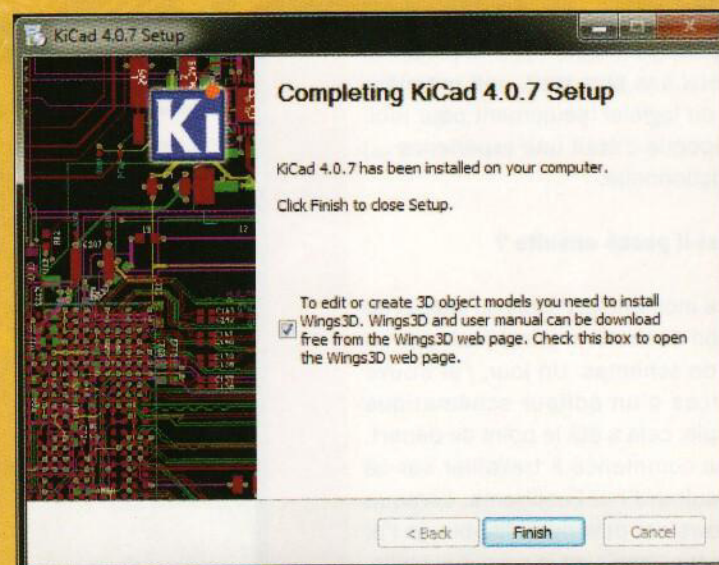


Figure 8D : l'installation de Kicad est terminée.

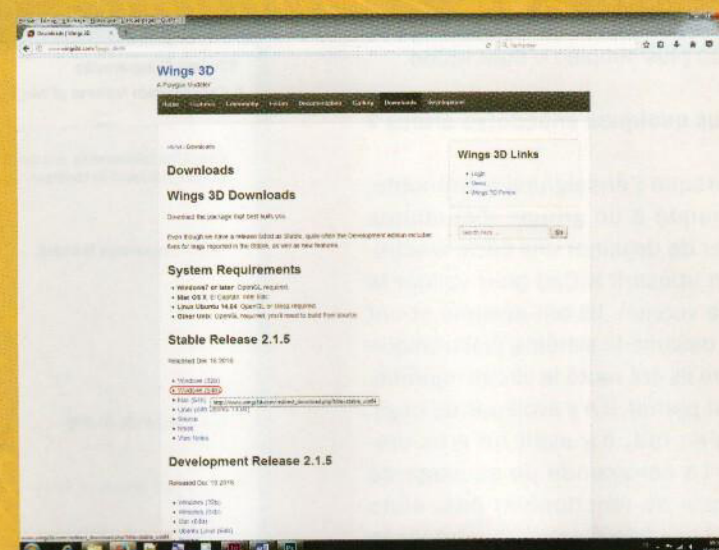


Figure 9A : la page web de téléchargement de Wings.

L'histoire de KiCad

KiCad a été développé par Jean-Pierre Charras, enseignant à l'université de Grenoble (aujourd'hui retraité). Nous avons écrit à Jean-Pierre pour lui poser quelques questions concernant l'histoire de KiCad.

Comment est né le projet ?

JPC : le projet est né en 1992, l'année où j'ai commencé à écrire le code de ce qui est actuellement Pcbnew. À l'époque, j'étais intéressé par le fonctionnement d'un logiciel de réalisation de circuits imprimés (editor layout). J'utilisais OrCAD sous MS-DOS pour mes projets électroniques. J'ai commencé à travailler sur le projet lorsque les premiers compilateurs gratuits 32 bits et les premières bibliothèques graphiques (libgrx) apparurent sur le Web. Deux ans plus tard, une première version du logiciel (seulement pour moi, car à l'époque c'était une expérience ...) était fonctionnelle.

Que s'est-il passé ensuite ?

JPC : À ce moment-là, j'avais un logiciel de réalisation de circuit imprimé mais pas un éditeur de schémas. Un jour, j'ai trouvé les sources d'un éditeur schématique très simple, cela a été le point de départ. J'ai donc commencé à travailler sur ce qui est aujourd'hui Eeschema. Lorsque wxWindows est devenu utilisable, je l'ai intégré pour remplacer la première bibliothèque graphique. Quelques années plus tard, une première version de l'outil complet (schéma + pcb) était entièrement fonctionnelle, bien évidemment elle était beaucoup plus simple : c'était KiCad.

Avez-vous quelques anecdotes drôles ?

JPC : Lorsque j'enseignais à Grenoble, j'ai demandé à un groupe d'étudiants d'essayer de dessiner une carte électronique en utilisant KiCad pour valider la première version. Ils ont accepté et ont d'abord dessiné le schéma électronique et ensuite ils ont routé le circuit imprimé. Tout était parfait, il n'y avait pas de bugs. Sauf qu'en fait, il y avait un gros problème ! La commande de sauvegarde de Pcbnew ne fonctionnait pas, alors la première carte électronique réalisée avec KiCad n'a jamais vu le jour ...

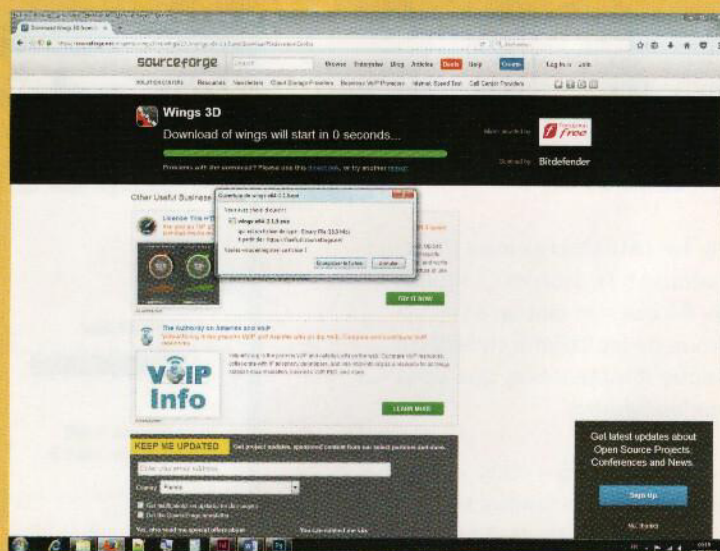


Figure 9B : la page « Source Forge » de téléchargement de « Wing ».

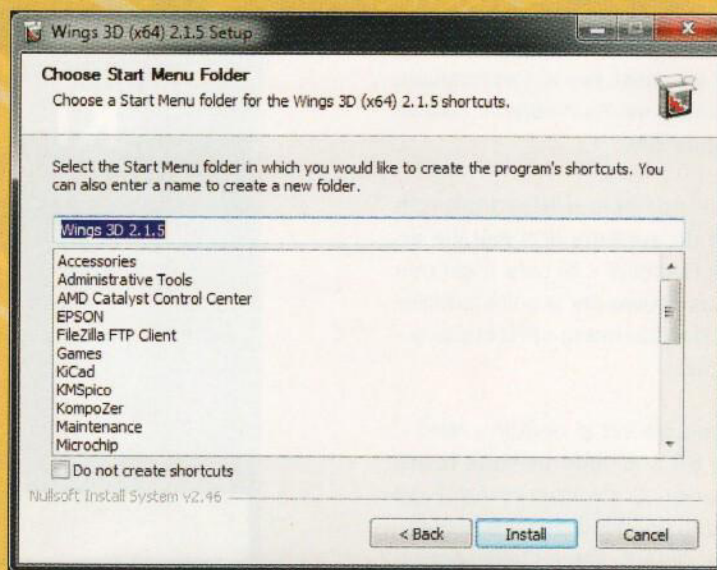


Figure 9C : installation de « Wing ».

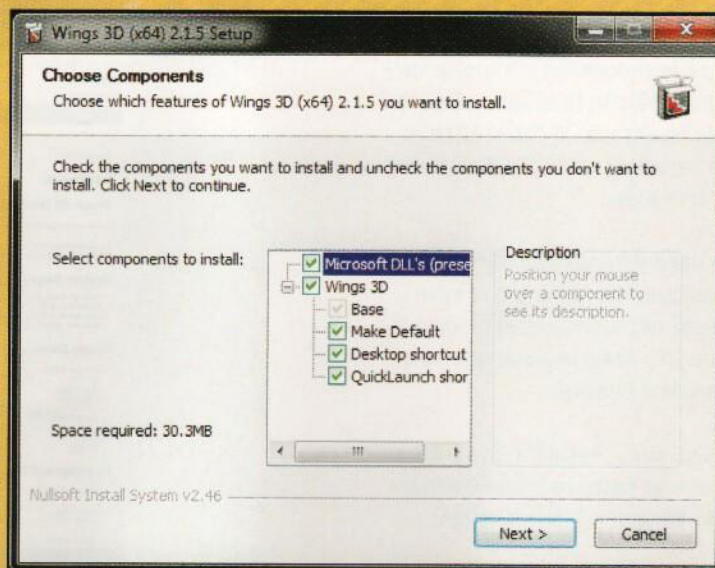


Figure 9D : choix des composants de « Wing » à installer.

Windows 64 bits (encerclée en rouge sur la figure 9A).

Cliquez alors sur le lien « Windows (64b) » ou autre en fonction de votre système d'exploitation. Vous serez alors redirigé vers la page « Source Forge » de téléchargement de « Wing » (voir la figure 9B). Au bout de quelques secondes une petite fenêtre s'ouvre, cliquez sur « Enregistrer le fichier ».

Ouvrez le dossier où vous venez d'enregistrer le fichier, il se nomme « **wings-x64-2.1.5.exe** ». Exécutez-le en tant qu'administrateur, la fenêtre de la figure 9C s'ouvre vous invitant à installer « Wings ».

Cliquez sur le bouton « Install », la fenêtre de la figure 9D apparaît.

Cochez la case « Microsoft DLL » afin d'installer « Visual C++ Redistributable » puis cliquez sur le bouton « Next ».

Si Visual C++ est déjà installé sur votre système, une fenêtre apparaît avec 3 boutons « Repair », « Uninstall » et « Close ». Dans ce cas, cliquez sur « Close » pour que le processus d'installation de « Wings » continue.

Si Visual C++ n'est pas installé, cliquez sur « Install ». À la fin de l'installation de « Visual C++ Redistributable », le processus d'installation de « Wings » continuera.

La fenêtre d'emplacement de l'installation de « Wings » s'ouvre (voir la figure 10), laissez le chemin par défaut sinon sélectionnez à votre convenance un autre dossier à l'aide du bouton « Browse ».

Cliquez sur « Next », le processus d'installation commence (voir la figure 11). Au bout de quelques instants, l'installation se termine, cliquez alors sur le bouton « Close » pour fermer la fenêtre (voir la figure 12).

À ce stade, vous disposez de la suite complète correctement installée avec la visualisation 3D des circuits imprimés.

Notez que si vous créez des composants spécifiques dans KiCad, vous

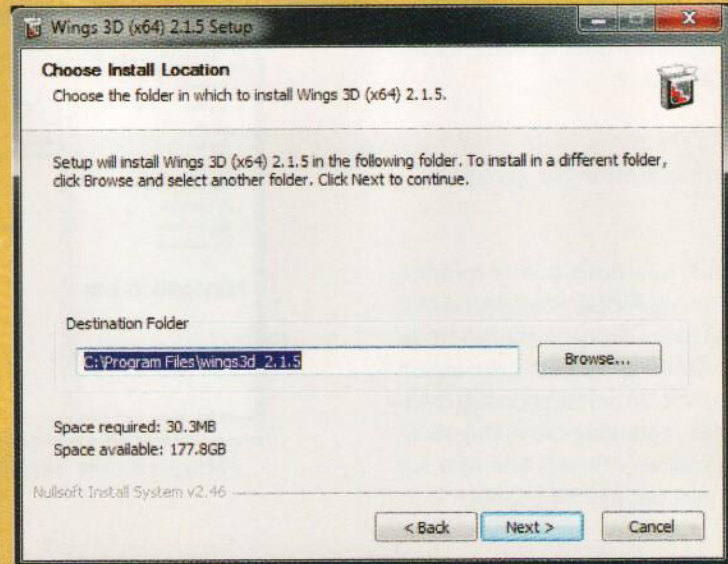


Figure 10 : choix de l'emplacement de l'installation de « Wings ».

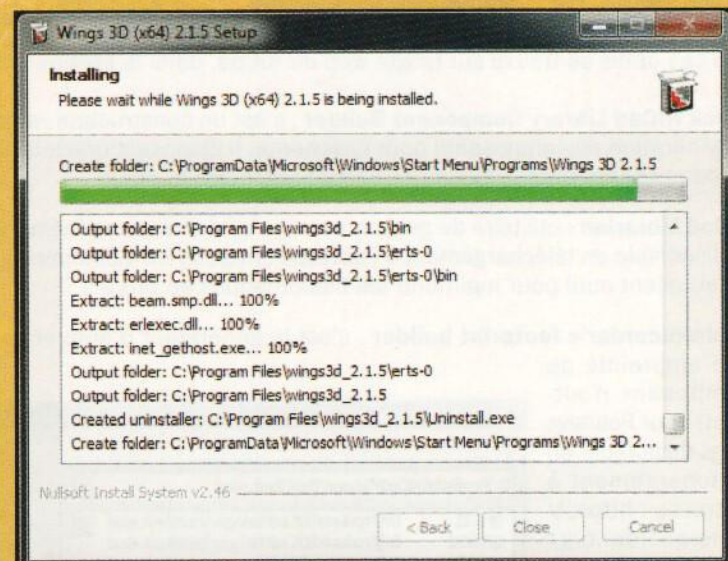


Figure 11 : processus d'installation de « Wings ».

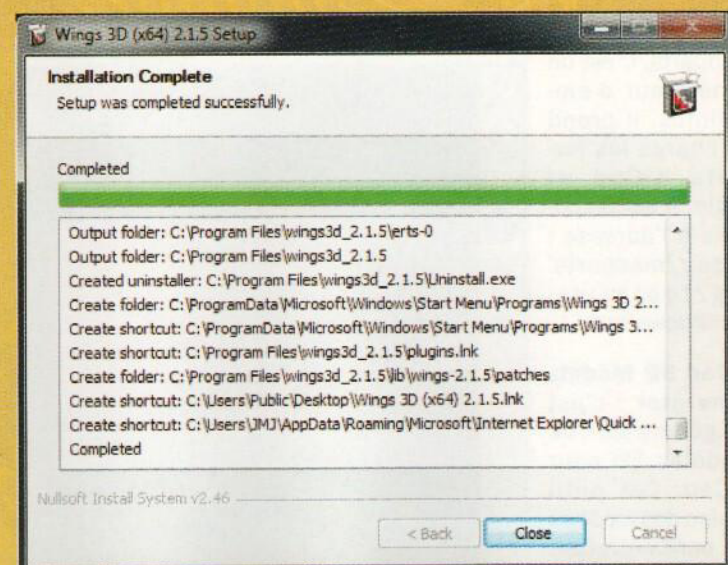


Figure 12 : l'installation est terminée, cliquez sur « Close » pour fermer la fenêtre.

devrez créer aussi le modèle en 3D pour Wings afin que le composant s'affiche en 3D.

Le gestionnaire de projet KiCad

Maintenant que nous avons terminé l'installation de KiCad, nous allons étudier l'interface afin de comprendre la structure. Étant donné que KiCad est un IDE, il possède un certain nombre d'outils intégrés regroupés dans une seule interface appelée « Project Manager », il s'agit du gestionnaire de projet.

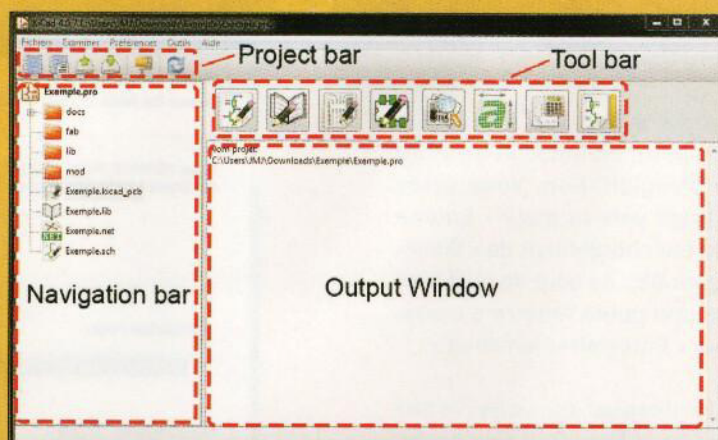


Figure 13 : les différentes zones du gestionnaire de projet.

Les outils externes

KiCad dispose d'un certain nombre d'outils externes afin d'aider les utilisateurs dans certaines tâches. Une liste de certains de ces outils se trouve sur le site web de KiCad, dans la section « Utilities ».

- **Quick KiCad Library Component Builder** : c'est un constructeur rapide de composants pour la bibliothèque KiCad. Il permet la génération de composants pour Eeschema. Il dispose d'une interface web très intuitive qui simplifie la génération de composants (<http://kicad.rohrbacher.net/quicklib.php>) ;
- **KiCad Librarian** : utilitaire de gestion pour les bibliothèques schématiques et d'empreinte pour KiCad. Il est autonome et disponible en téléchargement à l'adresse : https://www.compuphase.com/electronics/kicadlibrarian_en.htm. C'est un excellent outil pour maintenir les bibliothèques en ordre ;
- **CycleRecorder's footprint builder** : c'est le générateur d'empreinte de CycleRecorder. Cet outil permet de générer une empreinte de composant (footprint) pour Pcbnew. Il est disponible en téléchargement à l'adresse : <https://cyclerecorder.org/footprintbuilder/> ;
- **Madparts footprint editor** : c'est l'éditeur d'empreinte de Madparts. C'est un générateur d'empreintes. Il prend en charge les formats KiCad et Eagle. Il est disponible à l'adresse : <https://madparts.org/footprint.html#download> ;
- **KiCad 3D Models generator** : c'est un générateur de modèles 3D pour KiCad. Cet outil génère des modèles 3D paramétrables pour le visualiseur 3D de KiCad.

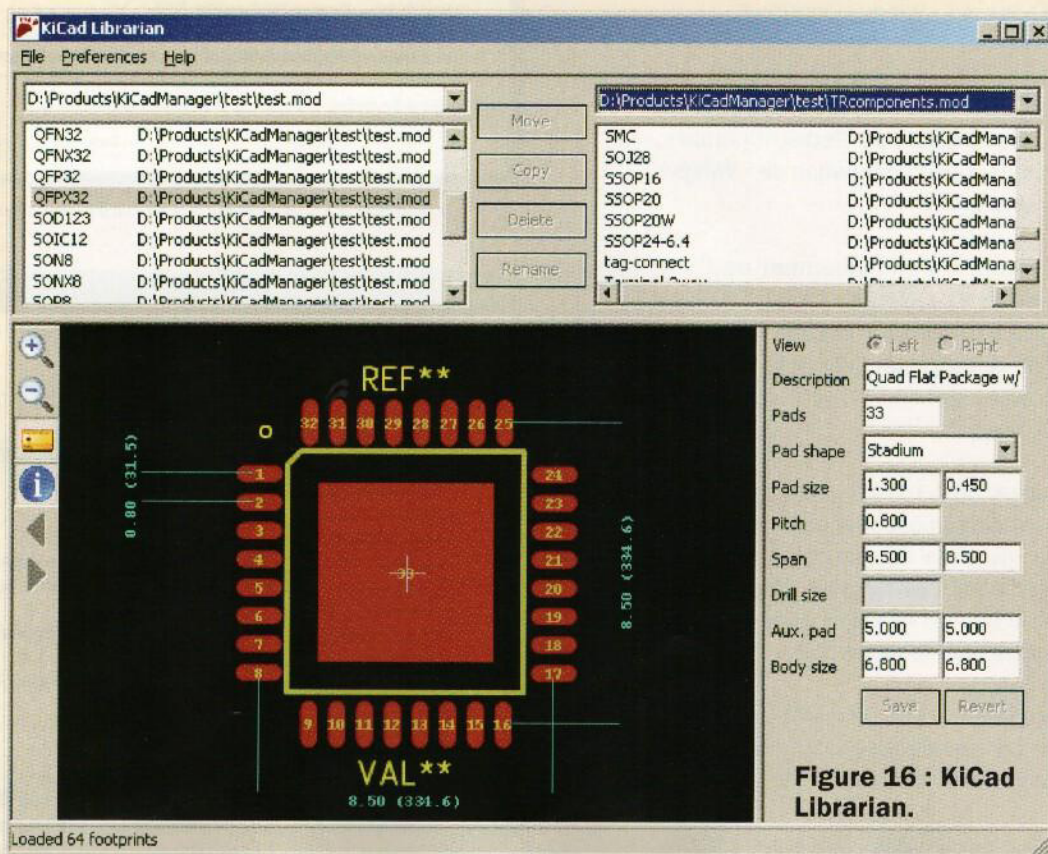


Figure 16 : KiCad Librarian.

Ce dernier gère les fichiers du projet et s'occupe de l'intégration des diverses applications (Pcbnew, Gerberview, etc.), fournissant ainsi à l'utilisateur une interface unique pour la gestion de projet.

Le gestionnaire de projet de Kicad se compose de quatre composants principaux :

- **Project Bar** : c'est la zone de gestion du projet, vous pouvez ouvrir, créer, enregistrer, archiver et mettre à jour l'affichage d'un projet. Au-dessus de la Project Bar, dans « Préférences » vous pouvez sélectionner la langue ainsi que d'autres options ;
- **Navigation Bar** : cette zone permet de gérer les fichiers du projet. Tous les fichiers du projet sont affichés, vous pouvez les éditer, les renommer et les supprimer. Pour les ouvrir, faites un double clic sur l'un des fichiers ;
- **Tool Bar** : c'est la zone correspondant aux outils qui composent l'IDE. Les icônes permettent de lancer les principales applications telles que la saisie de schéma Eeschema, Cvp pcb, Pcbnew ou encore Gerberview ;
- **Output Window** : c'est la fenêtre de sortie de KiCad, elle indique les opérations en cours d'exécution et affiche les éventuels messages d'erreurs ou d'avertissements.

La figure 13 montre le gestionnaire de projet avec ses différentes zones décrites ci-dessus. Le gestionnaire de projet permet également de créer un nouveau projet.

Pour cela, cliquez simplement sur l'icône « Créer un nouveau projet » situé à l'extrême gauche de la « Project Bar » en dessous du menu « Fichier », comme le montre la figure 14.

NB : en amenant le pointeur de la souris sur une icône, vous verrez apparaître la description de l'icône en question.

En cliquant sur l'icône « Créer un nouveau projet », une boîte de dialogue

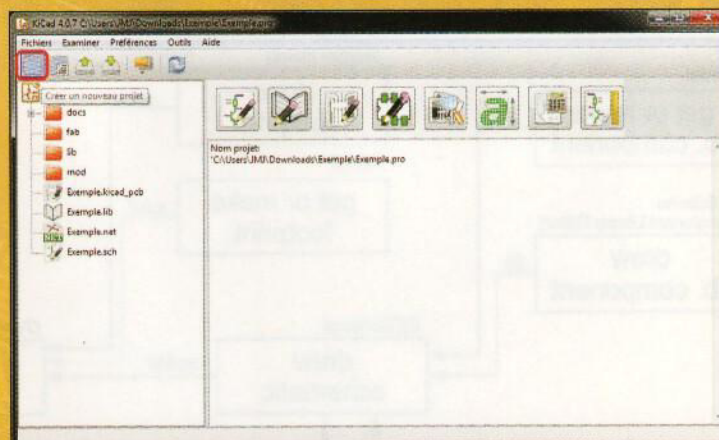


Figure 14 : création d'un nouveau projet.

s'ouvre vous invitant à donner un nom au projet et à sélectionner un emplacement (dossier) sur le disque dur. Vous pouvez aussi créer un nouveau dossier à votre convenance. Ensuite cliquez sur le bouton « Enregistrer » pour enregistrer le projet dont le nom porte l'extension « .pro ». Il s'agit d'un fichier de projet sous KiCad. Vous pouvez maintenant utiliser les différents outils de la CAO.

Dans le paragraphe suivant, nous allons clarifier l'échange des données d'un projet entre les différents programmes constituant KiCad.

Echange des données dans KiCad

KiCad est conçu pour faciliter le développement d'un projet dans un certain ordre. Il est en réalité composé de plusieurs logiciels spécifiques qui coopèrent entre eux.

Tout d'abord, vous devez commencer par dessiner le schéma avec Eeschema (saisie du schéma), puis ensuite router (dessiner) le circuit imprimé à l'aide de Pcbnew et enfin générer les fichiers de fabrication de la carte électronique à l'aide de Gerbview. Ce sont les 3 étapes essentielles d'un projet.

En dépit de ses ressemblances avec d'autres logiciels de conception de circuits imprimés, KiCad se caractérise par une organisation dans laquelle symboles et empreintes de composants sont des entités disjointes.

Le travail avec KiCad s'organise en deux tâches principales : la réalisation du schéma et celle du circuit imprimé. Une librairie de composants et une librairie d'empreintes sont nécessaires pour ces deux tâches. KiCad en fournit de nombreuses. Si celles-ci ne vous suffisaient pas, KiCad dispose également d'outils pour créer de nouveaux composants schématiques ou de nouvelles empreintes.

Le passage à une étape ultérieure ne se produit qu'après l'achèvement de la précédente, ce qui minimise d'autant le nombre d'erreurs dans un projet, car cela serait très difficile de localiser et de récupérer des erreurs dans les étapes finales.

L'organisation d'un projet dans KiCad est schématisée en figure 15. Les blocs grisés représentent les outils qui ne sont pas intégrés dans KiCad, mais génèrent des sorties compatibles avec ce dernier.

Comme vous pouvez le voir, la première étape du développement d'un projet sous KiCad consiste à dessiner le schéma électrique. Le point de départ de ce bloc est une bibliothèque (ou librairie) de composants existante ou la création de composants spécifiques à l'aide de l'éditeur de bibliothèques schématiques et de l'éditeur d'empreintes.

Cependant, il faut considérer que la création de composants sous Eeschema ne requiert pas la définition d'une empreinte de ce dernier, car dans KiCad le schéma d'un composant et son empreinte sont deux entités distinctes.

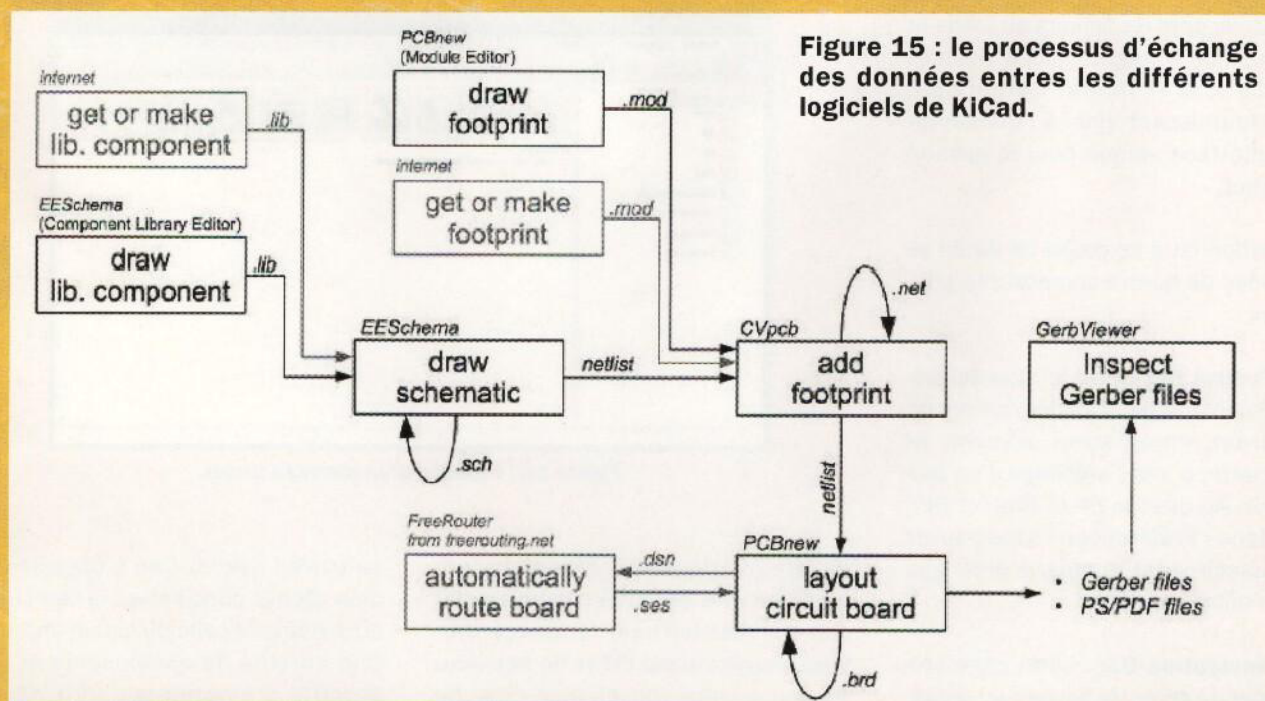


Figure 15 : le processus d'échange des données entre les différents logiciels de KiCad.

Elles sont constituées de deux bibliothèques différentes (et donc 2 éditeurs) ayant des fichiers de type différent (« .lib » pour le schéma du composant et « .mod » pour l'empreinte du même composant).

Cette approche est différente des autres logiciels similaires de CAO électronique et c'est l'une des particularités de KiCad. Une fois la librairie de composants obtenue ou générée, il est possible de passer au dessin réel du schéma sous Eeschema.

Ce processus produit deux fichiers de sorties différents : un fichier « .sch » qui constitue le fichier de travail sous Eeschema et un fichier « .net » (netlist) qui est le point de départ de l'outil suivant, c'est-à-dire Cvpcb.

La fonction de Cvpcb est d'associer le schéma électrique à des empreintes. Pour ce faire, Cvpcb acquiert en plus de la « Netliste », les bibliothèques d'empreintes, qui peuvent être préexistantes ou générées par un autre outil (Pcbnew ou un outil externe comme un générateur d'empreinte pour KiCad).

Cvpcb constitue donc le lien entre le schéma et l'empreinte, en mettant ensemble ces deux entités, qui jusqu'à présent étaient séparées.

Cette approche, d'une part, introduit apparemment une étape supplémentaire (seulement théorique, car la sélection de l'empreinte doit être faite dans la conception du schéma), et d'autre part réduit considérablement la taille des bibliothèques.

La sortie du processus d'association Cvpcb est le fichier « .net » de départ, enrichi des informations relatives aux empreintes des composants.

Ce fichier est pris comme point d'entrée pour Pcbnew, qui l'utilise comme fichier de départ pour générer le fichier « .kicad_pcb » (format de fichier du circuit imprimé), c'est-à-dire le fichier de travail de Pcbnew.

À ce stade, le routage du circuit imprimé peut être effectué, Pcbnew produit en sortie un fichier Gerber ou Postscript permettant la fabrication de la carte.

L'étape finale du travail est la visualisation des fichiers de sortie générés par Pcbnew, en utilisant Gerbview.

Cette étape, bien que facultative, est importante car elle permet de s'assurer qu'il n'y a pas d'erreurs inattendues lors de la conception, ce qui pourrait conduire à la fabrication d'un mauvais circuit.

L'analyse des fichiers Gerber est l'étape ultime du travail avec KiCad.

De toute évidence, certaines étapes peuvent subir des itérations afin de corriger les erreurs en revenant à l'étape précédente et en effectuant les modifications nécessaires afin de régénérer des fichiers corrects pour l'étape suivante.

Ce processus, généralement appelé rétro-annotation, permet de maintenir une cohérence absolue entre toutes les étapes du projet, de sorte qu'il n'y ait pas d'incohérences entre les différentes phases de conception, de la saisie du schéma au routage du circuit imprimé.

Nous arrivons à la fin de cette première leçon, nous avons présenté l'environnement de KiCad, et nous vous avons détaillé les étapes de téléchargement et d'installation.

Ensuite nous avons décrit le gestionnaire de projet et le processus d'échange des données entre les différents logiciels constituant la suite KiCad.

Dans la prochaine leçon, nous allons aborder l'utilisation d'Eeschema afin de saisir le schéma d'un exemple de projet pratique.

Effet trémolo pour guitare

Nous vous présentons dans cet article un effet guitare qui fait vibrer le son des cordes à travers un système optique original de modulation d'amplitude.



de Davide Scullino

Ceux d'entre vous qui jouez de la guitare électrique utilisent normalement des effets, outre la manette à distorsion intégrée à l'instrument qui se comporte comme un « **Pitch Control** » (elle altère la fréquence de la note jouée car elle modifie la tension mécanique des cordes).

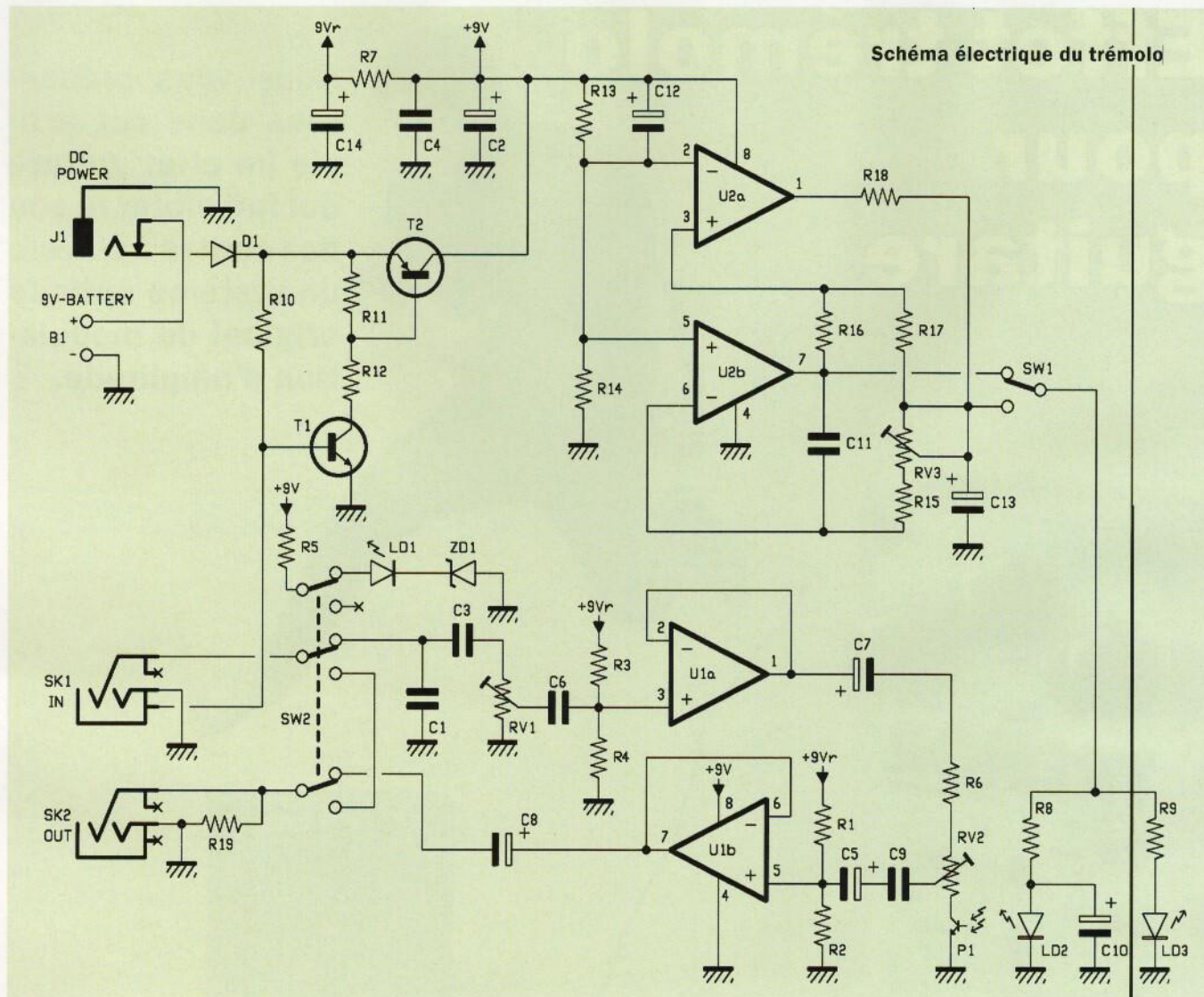
Parmi les effets les plus utilisés, c'est sans aucun doute l'**effet trémolo**, qui, comme son nom l'indique, fait trembler ou vibrer le son généré grâce à un **modulateur d'amplitude** qui augmente ou réduit périodiquement l'amplitude du signal avec une fréquence réglable à volonté dans certaines limites (généralement entre quelques hertz et quelques dizaines de hertz).

Dans ces pages, nous allons réaliser un effet trémolo dont les caractéristiques sont proches du domaine professionnel. Vous pourrez insérer le montage dans un boîtier métallique robuste afin qu'il résiste aux pieds des guitaristes les plus « rudes ».

Pour rappel, l'**effet trémolo** est soit un **effet monophonique de répétition d'une seule note**, soit une **alternance polyphonique de deux ou plusieurs sons sous forme d'accords**. Suivant les instruments de musique, sa notation diffère. Ce peut être un tremblement (voix), un roulement (xylophone) ou une rapide alternance digitale (guitare).

L'effet trémolo permet de rendre un son ou un groupe de sons plus expressif, mais aussi pour les instruments ne pouvant soutenir un son comme le xylophone ou la mandoline, une technique pour obtenir un son prolongé. Le montage permet la **modulation du signal audio** en mode **triangulaire**, ou en mode **rectangulaire** (ON/OFF), avec la possibilité d'**ajuster**, pour chaque onde modulante, la **fréquence** et la **profondeur de la modulation**. En plus des commandes pour l'effet, le circuit dispose d'un **réglage du volume** sonore qui est mis en œuvre sur l'entrée.

Schéma électrique du trémolo



L'ensemble du montage est alimenté par une pile de 9 VDC, ce qui permet de rendre le circuit autonome. Cependant, vous pouvez alimenter le circuit à l'aide d'un câble muni d'une prise jack provenant d'une alimentation stabilisée de 9 VDC.

Le positif de l'alimentation doit être relié au point central du jack. L'alimentation avec une pile vous permet de déplacer et d'installer facilement et rapidement l'effet, ce qui limite le nombre de câbles, qui dans le cas d'une utilisation sur scène peut devenir problématique.

Schéma électrique

Analysons le schéma électrique qui réalise l'effet trémolo, il est principalement constitué par deux amplificateurs

opérationnels à faible bruit de type **NE5532** (U1 et U2).

Le signal d'entrée provenant du câble de la guitare arrive dans le circuit par l'intermédiaire de la prise jack 6,35 mm nommée SK1.

Selon la position de l'interrupteur tripolaire **SW2**, le signal audio de la guitare est relié à l'entrée de l'amplificateur opérationnel **U1a** (broche 3) et le signal de sortie provenant de la broche 7 de **U1b** est dirigé vers la prise de sortie SK2.

De plus, il fournit l'alimentation à la LED LD1 de signalisation afin de montrer que l'effet est actif. SW2 correspond au bouton que le guitariste presse avec son pied pour activer ou désactiver l'effet. Cet interrupteur est du type à enclenchement, c'est-à-dire qu'il a une position stable lors de la commutation.

Comme nous l'avons indiqué, SW2 commute dans une position le signal d'entrée directement vers la prise de sortie SK2, tandis que dans l'autre position il dirige le signal d'entrée vers l'amplificateur U1a et le signal de sortie provenant de U1b vers la prise SK2.

Sur la première position l'effet trémolo est désactivé, tandis que sur la deuxième position l'effet est activé.

Une pression sur le bouton SW2 active l'effet, une deuxième pression le désactive, une troisième pression réactive l'effet et ainsi de suite. Le 3^{ème} contact de SW2 sert à allumer ou éteindre la LED LD1.

Étudions maintenant le fonctionnement de notre effet trémolo. En commutant l'interrupteur SW2 en position haute (c'est-à-dire celle du schéma électrique),

le signal d'entrée (IN) provenant de la guitare est filtré par le condensateur C1 qui supprime les parasites pouvant interférer avec le signal audio, captés par le câble de guitare.

Le signal audio passe à travers le condensateur C3 de découplage qui **bloque la composante continue** pouvant être présente dans le signal et atteint le potentiomètre RV1 dont le rôle est de régler le volume (le niveau sonore augmente au fur et à mesure que le point milieu du curseur s'éloigne de la masse). Ce potentiomètre est très utile pour adapter la profondeur de l'effet trémolo ainsi que le niveau de l'amplificateur.

À partir du curseur du potentiomètre, le signal est donc ajusté en amplitude pour atteindre le condensateur C6 qui permet de découpler le signal audio de la composante continue du réseau de polarisation de l'amplificateur opérationnel U1a.

Le signal audio est ainsi appliqué à l'**entrée non-inverseuse** de U1a (broche 3), ce dernier étant configuré en étage tampon (buffer) car son **gain est unitaire**. Il n'amplifie pas la tension, mais seulement le courant.

Le **diviseur de tension** formé par les résistances R3 et R4 fixe le potentiel de la broche 3 de U1a à environ la **moitié de la tension d'alimentation**, ce qui permet à l'amplificateur opérationnel de **transmettre toute la dynamique du signal audio**, dont les amplitudes sont à la fois positives et négatives (le signal audio se comporte comme un signal sinusoïdal).

S'il n'y avait pas le diviseur de tension, c'est-à-dire une polarisation de U1a, ce dernier bloquerait les signaux négatifs. Les demi-ondes négatives en sortie seraient alors tronquées et donc le signal fortement distordu.

Le signal audio sort par la broche 1 de l'amplificateur opérationnel U1a, et passe à travers le condensateur de découplage C7 qui bloque la composante continue de U1a.

Le signal audio atteint ainsi l'**étage de modulation**, composé par la résistance

R6, le potentiomètre RV2 et le phototransistor P1. Celui-ci est utilisé comme **élément de modulation** lorsqu'il reçoit sur sa base la lumière émise par la LED LD2.

Ainsi, P1 devient **plus ou moins conducteur en fonction de la quantité de lumière reçue**. Par conséquent **son collecteur se retrouve plus ou moins vers la masse**, ce qui a pour effet de faire varier le courant de collecteur.

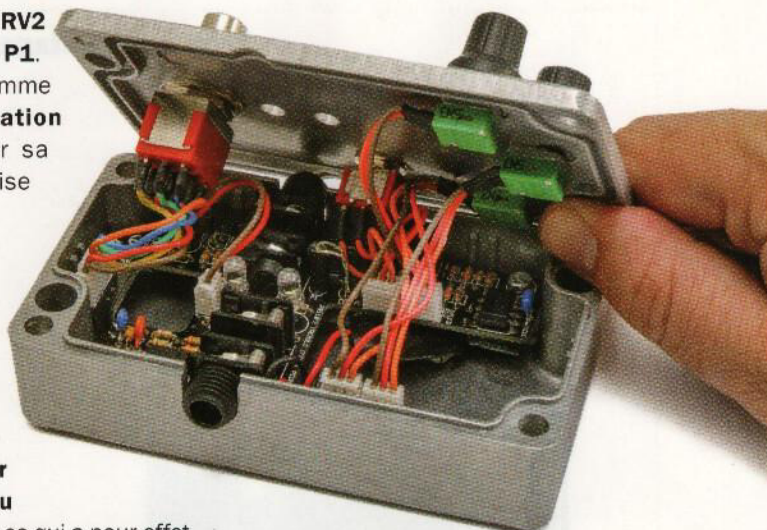
P1 ainsi que le diviseur de tension formé par R6 et RV2 se comportent comme une résistance variable. **Plus le courant de collecteur de P1 est grand, plus le niveau du signal prélevé sur le curseur du potentiomètre RV2 est petit** et vice versa.

Dans la pratique, P1 permet de faire varier le rapport du diviseur de tension dans lequel il est inséré, P1 réalise donc une véritable modulation du signal.

Le courant de collecteur, et par conséquent la **conductance du phototransistor** situé entre le collecteur et l'émetteur, **dépend de l'intensité lumineuse émise par LD2**. Cette dernière est **couplée optiquement au phototransistor P1** à l'aide d'un tube qui empêche la dispersion de la lumière émise par la LED.

Ainsi, toute la lumière arrive sur la base de P1, le tube protège aussi de la lumière ambiante qui créerait des interférences et perturberait le fonctionnement du modulateur.

Notez que le système utilisé pour la modulation est une réédition d'un montage plus traditionnel, utilisé par le passé, mais encore préféré pour certains effets comme le trémolo.



À l'origine, le système était basé sur une photorésistance reliée en série avec un amplificateur opérationnel et une ampoule à incandescence pilotée (alimentée) par un oscillateur.

La photorésistance et l'ampoule étaient couplées dans un tube fermé aux extrémités.

Cette solution garantit la meilleure linéarité possible car la photorésistance est linéaire ainsi que l'ampoule, dans la mesure où cette dernière est alimentée par une tension qui ne fait pas trop monter en température le filament.

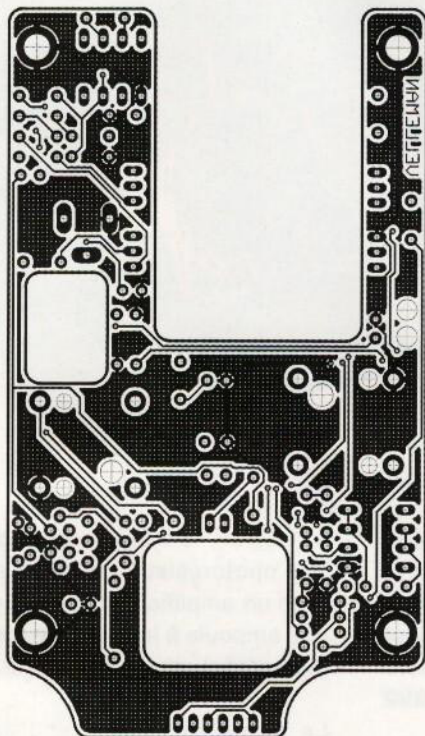
Revenons au schéma électrique, le signal modulé atteint le condensateur de découplage C9 qui bloque la composante continue du diviseur de tension.

Le signal arrive sur l'entrée **non-inverseuse** de U1b, qui est aussi montée en

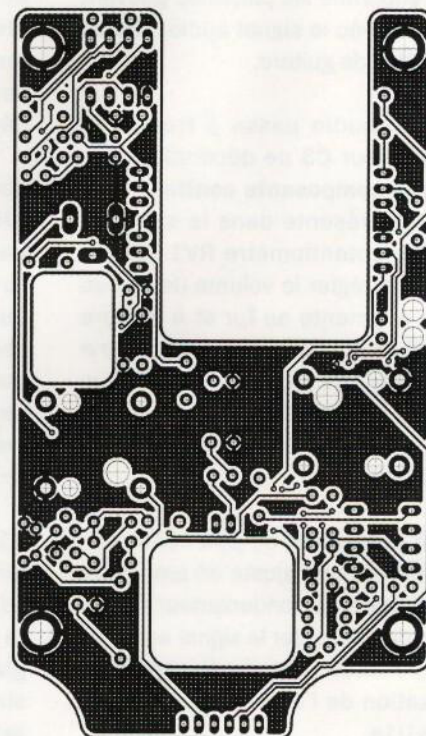
CARACTÉRISTIQUES TECHNIQUES

- Tension d'alimentation : 9 VDC ;
- Courant absorbé : 30 mA ;
- Fréquence du trémolo : de 1 Hz à 50 Hz ;
- Enveloppe de la modulation : carrée et triangulaire ;
- Commutateur de dérivation à pédale ;
- Contrôle du volume ;
- Contrôle de la fréquence et de la profondeur de la modulation ;
- Entrée et sortie sur connecteurs jacks 6,35 mm ;
- Dimensions : 115 mm x 65 mm x 25 mm.

Plan de montage du circuit trémolo



Circuit imprimé à l'échelle 1 : 1 côté soudures du circuit trémolo.



Circuit imprimé à l'échelle 1 : 1 côté composants du circuit trémolo.

Liste des composants

R1..... 330 k Ω
 R2..... 470 k Ω
 R3..... 330 k Ω
 R4..... 470 k Ω
 R5..... 390 Ω
 R6..... 10 k Ω
 R7..... 100 k Ω
 R8..... 470 Ω
 R9..... 470 Ω
 R10.... 1 M Ω
 R11.... 10 k Ω
 R12.... 4,7 k Ω
 R13.... 47 k Ω
 R14.... 33 k Ω
 R15.... 10 k Ω
 R16.... 39 k Ω
 R17.... 100 k Ω
 R18.... 100 Ω
 R19.... 1 M Ω
 RV1.... potentiomètre 100 k Ω
 RV2.... potentiomètre 10 k Ω
 RV3.... potentiomètre 500 k Ω

 C1..... 470 pF céramique
 C2..... 220 μ F/25 V électrolytique
 C3..... 100 nF céramique
 C4..... 100 nF céramique

C5..... 10 μ F/63 V électrolytique
 C6..... 47 nF céramique
 C7..... 10 μ F/63 V électrolytique
 C8..... 10 μ F/63 V électrolytique
 C9..... 100 nF céramique
 C10.... 47 μ F/63 V électrolytique
 C11.... 1 μ F céramique
 C12.... 10 μ F/63 V électrolytique
 C13.... 100 μ F/25 V électrolytique
 C14.... 10 μ F/63 V électrolytique

 D1..... 1N4007

 T1 BC547
 T2 BC557

 U1..... NE5532P
 U2..... NE5532P

 LD1.... LED 5 mm bleue
 LD2.... LED 3 mm blanche
 LD3.... LED 5 mm blanche

 ZD1.... zener 3,3 V/400 mW

 SW1... Interrupteur à levier
 SW2... double interrupteur 3 pôles

 P1 phototransistor NPN
 BPW85B ou équivalent

Divers

Prise alimentation pour ci
 Clip pour pile 9 V 6F22

 Connecteur jack 6,35 mm pour ci
 (x2)

 Boîtier métallique

 Bouton pour potentiomètre (x3)

 Connecteur mâle 3 pôles JST 2 mm
 pour ci (x4)

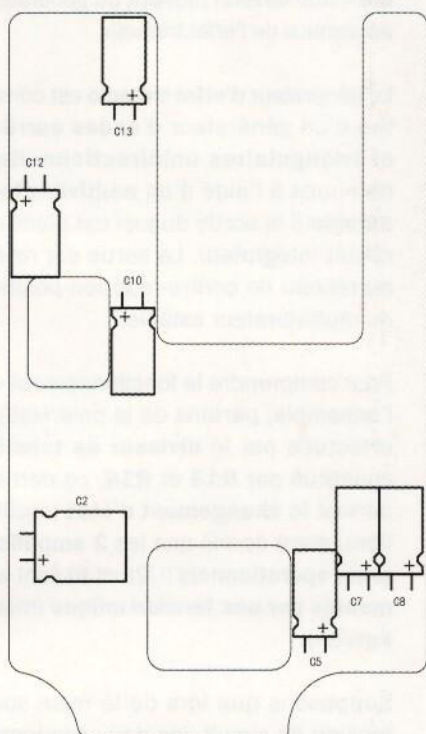
 Connecteur femelle 3 pôles JST 2
 mm volant (x4)

 Connecteur mâle 6 pôles JST 2 mm
 pour ci

 Connecteur femelle 6 pôles JST 2
 mm volant

 Connecteur mâle 2 pôles JST 2 mm
 pour ci

 Connecteur femelle 2 pôles JST 2
 mm volant



Plan de câblage des composants côté soudures (bottom) du circuit trémolo.

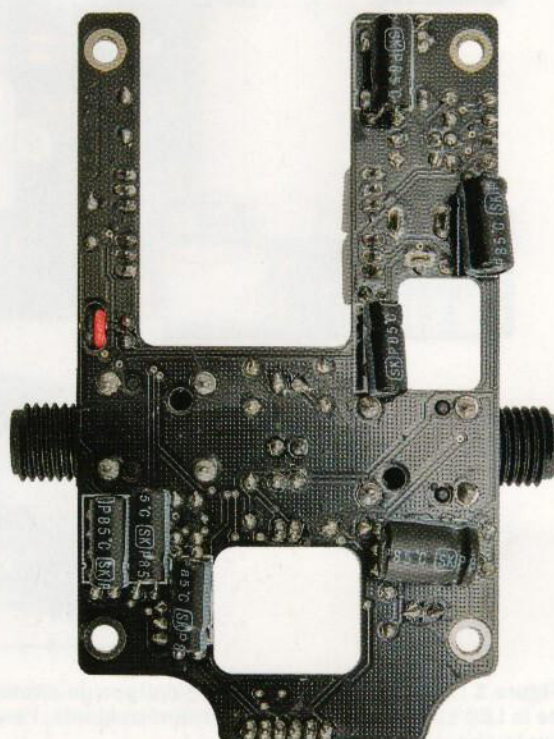


Photo de l'un de nos prototypes côté soudures (bottom) du circuit trémolo.

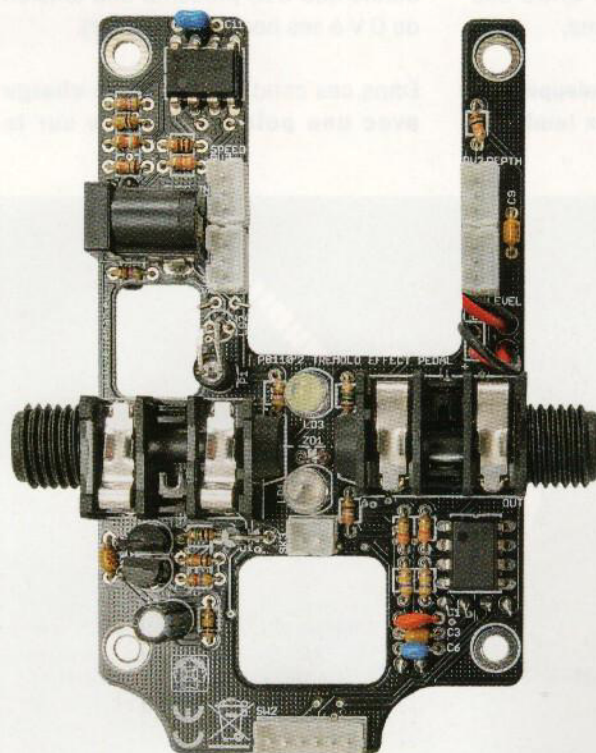
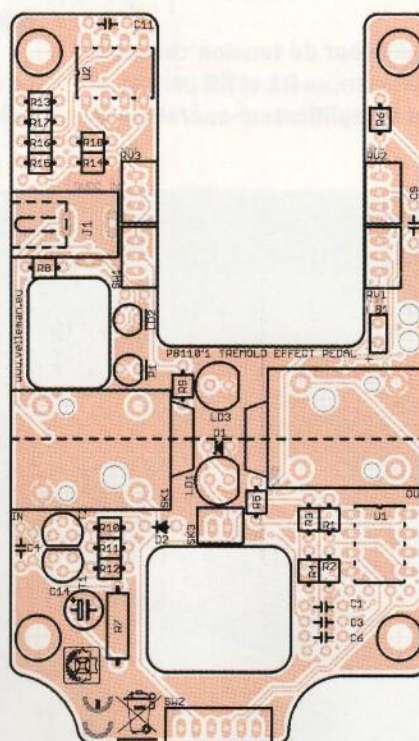


Photo de l'un de nos prototypes côté composants (top) du circuit trémolo.



Plan de câblage des composants côté composants (top) du circuit trémolo.

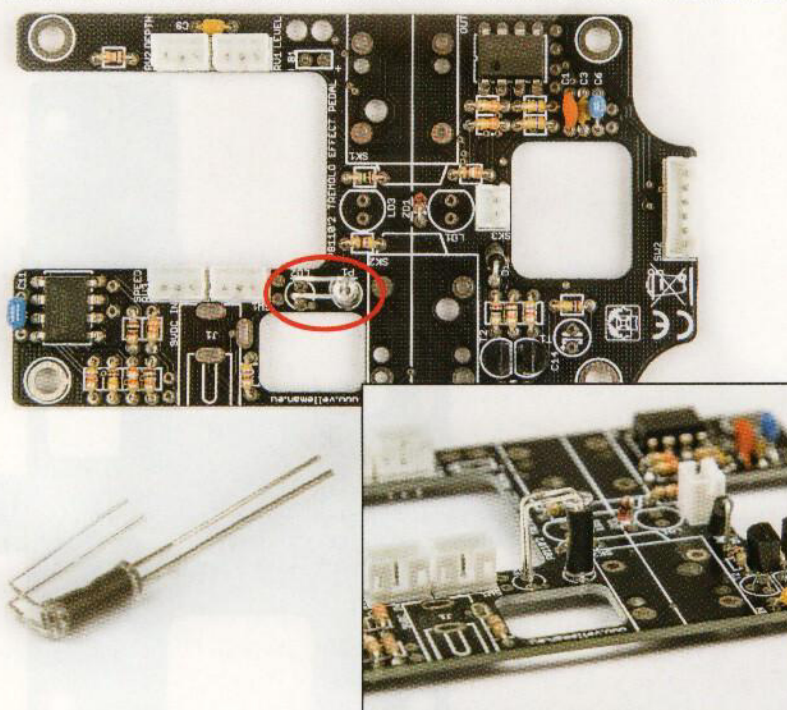


Figure 1 : assemblage du couplage optique du phototransistor de modulation et de la LED LD2 dans une gaine thermorétractable, l'ensemble étant ensuite soudé sur le circuit imprimé.

étage tampon. Le signal en sortie (broche 7) atteint le condensateur électrolytique C8 avec la même amplitude, pour passer à travers le commutateur SW2 et enfin vers la prise jack de sortie.

Le pont **diviseur de tension** constitué par les résistances **R1** et **R2** permet de **polariser** l'amplificateur opérationnel

U1b de manière à ce que sa tension de sortie au repos (broche 7) soit égale à environ la **moitié de la tension d'alimentation**. De cette manière, l'amplitude du signal peut varier entre des valeurs positives et négatives.

Le **signal modulé** a une **enveloppe** dont l'**amplitude** dépend du **flux lumineux**

généré par la LED, qui à son tour dépend de la tension avec laquelle elle est alimentée. Cette tension provient du générateur de signaux de l'effet trémolo.

Le générateur d'**effet trémolo** est constitué d'un générateur d'**ondes carrées** et **triangulaires unidirectionnelles**, obtenues à l'aide d'un **multivibrateur astable** à la sortie duquel est placé un circuit intégrateur. La sortie est reliée au réseau de contre-réaction positive du multivibrateur astable.

Pour comprendre le fonctionnement de l'ensemble, partons de la polarisation effectuée par le **diviseur de tension** constitué par **R13** et **R14**, ce dernier permet le **changement d'état** (oscillation), étant donné que les **2 amplificateurs opérationnels** (U2a et b) sont **alimentés par une tension unique** (mono tension).

Supposons que lors de la mise sous tension du circuit, les deux condensateurs **C12** et **C13** soient **déchargés**. Le premier fournit une impulsion de niveau haut sur l'entrée inverseuse de U2a. La sortie de ce dernier est amenée à environ 0 V et avec elle l'entrée inverseuse (broche 6) de U2b, étant donné que C13 présente une tension de 0 V à ses bornes (déchargé).

Dans ces conditions, **C11 se charge avec une polarité positive sur la**

Les connexions



Comme toutes les « pédales », le trémolo décrit dans cet article est connecté entre la guitare électrique et un amplificateur ou une table de mixage.

broche 7 de U2b, et avec une **pente constante** car la charge s'effectue à courant constant. C'est une des caractéristiques d'un circuit intégrateur de tension.

La **broche 7** présente un **potentiel évoluant selon une rampe croissante**, qui s'arrête lorsque la tension reportée par le réseau de contre-réaction formé par R16 et R17 sur l'entrée non-inverseuse de U2a est **supérieure à celle de l'entrée inverseuse** de U2a. Cette tension augmente pendant la charge de C12 pour atteindre en régime permanent 3,7 V.

À ce stade, la sortie de **U2a** passe d'un **niveau bas à environ 9 V** et **force**, à travers R18 (négligeable car sa valeur est de seulement 100 Ω), le potentiomètre RV3 et R25, la **décharge du condensateur C11** et sa **recharge avec une polarité opposée**, à savoir positive, sur la broche 6. Cette charge détermine la chute linéaire du potentiel de la broche 7 jusqu'à ce que le réseau de contre-réaction R16 et R17 ne puisse plus rétrocéder une tension suffisante pour vaincre celle imposée sur la broche 2, le **circuit revient aux conditions initiales** et la **sortie de U2a** **retourne à un niveau bas**.

Le cycle se répète tant que le montage est alimenté. La **broche 1 de U2** **présente une onde carrée** et la **broche 7** **une onde triangulaire**, **toutes deux unidirectionnelles** et qui sont appliquées aux extrémités du commutateur SW1 grâce auquel nous pouvons choisir le type de modulation de l'effet trémolo, c'est-à-dire soit avec une enveloppe « carrée » ou « triangulaire ».

Notez que la présence du condensateur électrolytique C13 lisse les fronts montants et descendants de l'onde carrée et réalise ainsi une modulation douce conforme à l'effet trémolo, qui, dans sa forme théorique serait sinusoïdale.

La fréquence des deux ondes est identique, car les deux sont issues du même cadencement, essentiellement celui dû à la charge et à la décharge de C11 et réglé par le potentiomètre RV3. Lorsque **RV3 atteint sa résistance maximale**, la modulation du trémolo est ralentie au maximum, soit aux environs de **1 Hz**.

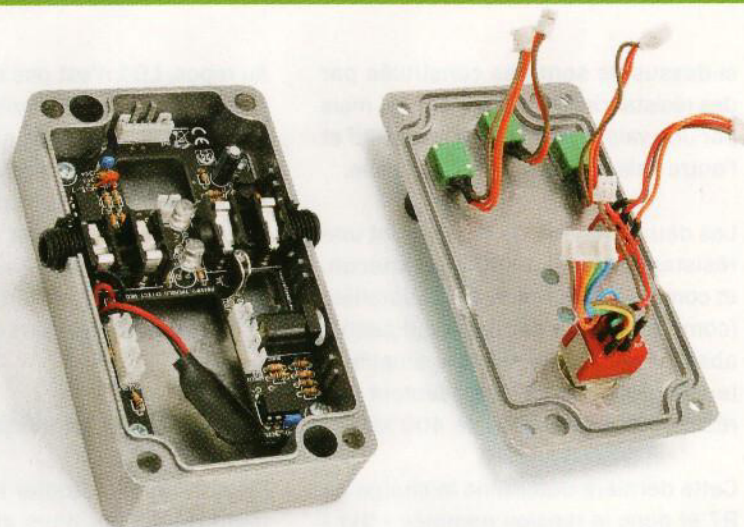


Figure 2 : le circuit imprimé est inséré dans un boîtier de forme rectangulaire. Toutes les commandes sont montées sur le couvercle et reliées par des connecteurs SIL au pas de 2,54 mm. Prévoyez deux trous pour les LED sur le couvercle.

En tournant le curseur de RV3 vers l'autre extrémité, il n'y a plus que R15 qui reste insérée dans le circuit, la fréquence de modulation atteint alors **50 Hz**.

Le signal sélectionné à l'aide du commutateur SW1 est appliqué à la LED blanche LD2 à travers la résistance R8 qui limite le courant. Le condensateur C10 lisse les pics de l'onde carrée. La diode LD3, alimentée de la même manière que LD2, indique visuellement que la modulation du signal audio de la guitare est active.

Il convient de noter que, malgré que la LED et le phototransistor n'aient pas un comportement linéaire, nous pouvons les utiliser pour transmettre le signal de modulation de manière tout à fait linéaire car l'excursion de la modulation de l'onde carrée ou triangulaire est limitée. Cela est dû au fait que la LED et le phototransistor fonctionnent dans une zone dont leurs caractéristiques sont quasi linéaires.

La **profondeur de la modulation d'amplitude** permise par le modulateur varie entre **10 %** (curseur du potentiomètre RV2 dirigé vers R6) et **50 %** (curseur vers le collecteur du phototransistor).

Concluons l'analyse du circuit avec l'étage d'alimentation, qui normalement au repos est maintenu hors circuit afin d'économiser la pile. Le circuit est activé lors de l'insertion d'un jack dans la prise SK1, l'extrémité arrière

du contact (jack de type stéréo) est normalement court-circuitée vers la masse (où le négatif de la pile est relié) et s'ouvre lorsque le jack est inséré, la **base** du transistor **T1** est alors **polarisée** par la résistance **R10**.

Cette dernière fait en sorte que le transistor entre en saturation, donc son collecteur relié à R12 se retrouve à la masse. Un courant circule alors dans R11 et R12 polarisant la base de T2 (PNP) qui fonctionne comme un interrupteur statique.

Le collecteur de **T2** fournit ainsi l'**alimentation** aux amplificateurs opérationnels. L'alimentation est filtrée par le condensateur électrolytique C2 ainsi que le condensateur céramique C4.

L'alimentation est également filtrée par un réseau de composants « RC » constitué de **R7** et **C14**. Le but de ce filtre RC est d'obtenir une alimentation « propre » (9Vr) pour que les diviseurs de tensions R1/R2, R3/R4 et R13/R14 fournissent des tensions de références les plus stables possible aux amplificateurs opérationnels.

La résistance R7 introduit une chute de tension appliquée aux diviseurs de tensions de référence par rapport à la tension de la pile qui est de 9 V. Cependant, les tensions de références fournies aux amplificateurs opérationnels valent environ la moitié de l'alimentation principale, étant donné que les diviseurs de tensions mentionnés

ci-dessus ne sont pas constitués par des résistances de valeurs égales, mais par une valant 330 k Ω vers le positif et l'autre valant 470 k Ω vers la masse.

Les deux diviseurs de tension ont une résistance totale de 800 k Ω chacun, et comme ils sont montés en parallèle (compte tenu du courant négligeable absorbé par les entrées des amplificateurs opérationnels) ils présentent une résistance équivalente de 400 k Ω .

Cette dernière détermine la charge de R7 et donc la tension nommée « 9Vr » vaut environ **7,2 V**. Par conséquent, la tension présente sur les entrées du **U1a** et **U1B** (broches 3 et 5) vaut environ **4,23 V** environ.

Par contre, le pont diviseur formé par R13 et R14 est alimenté directement à partir du 9 V, en raison des valeurs relativement faibles des résistances, si le pont était alimenté à partir de R7, la tension « 9Vr » chuterait excessivement.

Le jack d'alimentation J1 représente une alternative à la pile. Lorsqu'il est inséré, le contact central s'ouvre isolant ainsi du reste du circuit le positif de la pile.

Notez que, quelle que soit la source d'alimentation, la LED bleue LD1 s'allume pour indiquer la présence d'alimentation et l'activation de l'effet trémolo. La LED est connectée en série avec une résistance qui limite le courant direct et une diode zener, qui empêche l'allumage en dessous d'une certaine tension sur la ligne « +9V ». LD1 indique que la pile est faible lorsqu'elle est éteinte.

En fait, dans le cas d'une alimentation sur pile, lorsque la tension de cette dernière est faible, la chute de tension aux bornes de la zener s'ajoute à celle de la résistance de limitation (R5). Si la chute de tension est supérieure au seuil de LD1, celle-ci en fonctionnement normal ne s'allumera pas ou faiblement, indiquant qu'il faut changer la pile.

La LED LD1 est alimentée uniquement lorsque le commutateur SW2 est enclenché afin d'activer l'effet trémolo.

Au repos, LD1 n'est pas alimentée afin d'éviter une consommation inutile de la pile. Si le circuit est alimenté via la prise jack, LD1 ne sera pas allumée au repos (il est inutile de vérifier la tension de la pile si le circuit est alimenté avec une source extérieure), car dans ce cas nous n'avons pas besoin de savoir que l'effet trémolo n'est pas actif.

Réalisation pratique

Nous venons d'étudier le fonctionnement du circuit, nous allons maintenant le construire et l'utiliser avec une guitare.

L'ensemble de l'électronique tient sur un seul circuit imprimé double face dont vous pouvez télécharger les typons sur notre site www.electroniquemagazine.com dans le sommaire détaillé de la revue.

Une fois le circuit gravé et percé, commencez comme d'habitude par souder les composants ayant un profil bas : les résistances, les diodes en respectant l'orientation de la bague, les supports de circuits intégrés, les condensateurs non polarisés.

Les condensateurs polarisés doivent être soudés, en respectant la polarité, du côté soudures (face bottom) du circuit imprimé. Pour cela, aidez-vous du plan de câblage et de la photo dans l'encadré intitulé « Plan de montage du circuit trémolo ».

Préparez le montage mécanique des interrupteurs sur la face avant du boîtier en soudant des fils assez longs pour les relier au circuit imprimé. Continuez en soudant les LED tout en respectant la polarité. La cathode est généralement indiquée par un biseau et la patte la plus longue correspond à l'anode.

En ce qui concerne le phototransistor, la patte la plus longue correspond à l'émetteur. Vous devez coupler optiquement le phototransistor avec la LED LD2 à l'aide d'un morceau de gaine thermorétractable. Enfin, soudez toutes les prises, celle d'alimentation et les jacks femelles 6,35 mm pour circuit imprimé.

Notez que les LED LD1 et LD3 sont soudées à une certaine hauteur par rapport au circuit imprimé, de façon à ce qu'elles viennent à fleur des trous respectifs du couvercle. Elles doivent être visibles de l'extérieur.

Comme nous l'avons déjà mentionné, LD2 est couplée optiquement au phototransistor. Pour cela, vous devez plier les pattes des 2 composants qui doivent être soudés horizontalement après les avoir disposés dans de la gaine thermorétractable ou dans un petit morceau de tube en plastique noir (non translucide) de 8 mm de longueur et 3 mm de diamètre.

Eventuellement, chauffez la gaine afin de maintenir les composants, sinon utilisez de la colle. N'oubliez pas d'étanchéifier les extrémités de la lumière ambiante à l'aide de silicone.

En figure 1, vous pouvez visualiser la manière de réaliser l'assemblage optique de ces 2 composants.

Une fois que tous les composants sont soudés correctement, insérez les deux NE5532 dans leur support en faisant attention à l'orientation. Reportez-vous au plan de montage.

Les potentiomètres sont connectés au circuit imprimé à l'aide de connecteurs à trois pôles au pas de 2,54 mm. Aux autres extrémités, les fils doivent être soudés aux bornes des potentiomètres (voir la figure 2). Sinon, vous pouvez les relier au circuit en soudant 3 fils à chaque potentiomètre, si vous n'avez pas la possibilité d'utiliser des connecteurs. Choisissez la manière qui vous convient le mieux.

Le montage du circuit étant terminé, vous devez le disposer dans un boîtier, de préférence en métal. Dans ce cas, connectez la masse de l'alimentation en un seul point du boîtier à l'aide d'une vis et d'un écrou.

Pour éviter les bouclages de masse, il est conseillé d'utiliser des **prises jack en plastique** pour l'entrée IN et la sortie OUT, ou les isoler du boîtier en faisant des trous plus grands de manière à ce que les parties métalliques des jacks ne touchent pas le boîtier.

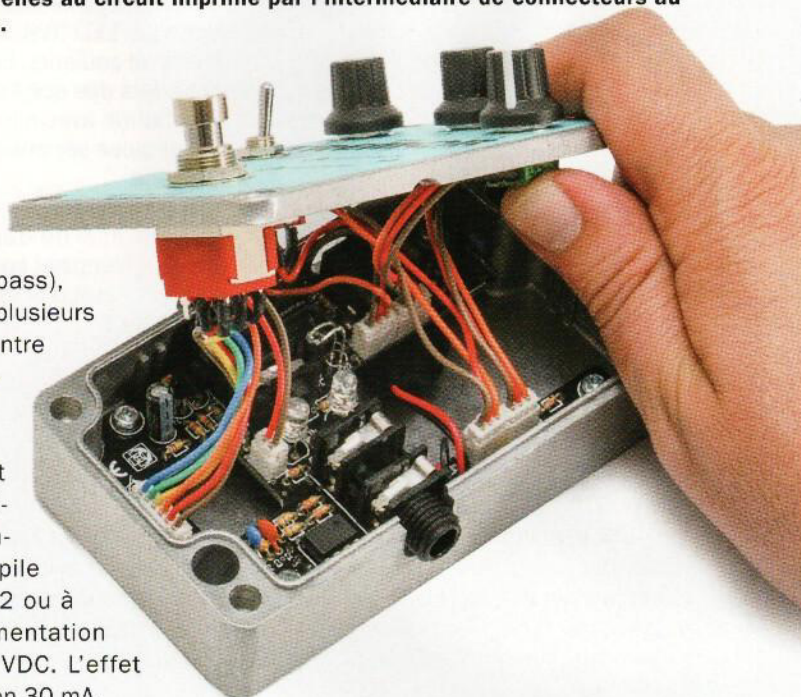
La figure 3 montre le boîtier métallique qui est très robuste, car il est placé sur le sol et actionné par un pied.

Choisissez des potentiomètres avec un axe et un pas de vis en plastique pour les fixer sur le couvercle du boîtier métallique, après avoir percé des trous de diamètre approprié.

De la même manière, l'interrupteur SW1 et le commutateur SW2 sont montés sur le couvercle du boîtier, isolez-les correctement du boîtier pour éviter des bouclages de masses. Ils sont reliés au circuit imprimé par des fils ou des connecteurs, selon votre convenance.

Une fois l'assemblage complet terminé, nous pouvons procéder au test. Comme toutes les « pédales », le trémolo est connecté entre une guitare électrique et un amplificateur ou une table de mixage. Il peut y avoir dans certains cas d'autres « effets à pédales » entre la guitare et le trémolo. Comme les « pédales » peuvent être

Figure 3 : les câbles du commutateur tripolaire, de l'interrupteur et des 3 potentiomètres sont reliés au circuit imprimé par l'intermédiaire de connecteurs au pas de 2,54 mm.



contournées (bypass), il peut y en avoir plusieurs reliées en série entre la guitare et l'amplificateur.

Branchez l'effet trémolo à l'amplificateur, alimentez-le avec une pile de 9 V type 6F22 ou à l'aide d'une alimentation stabilisée de 9 VDC. L'effet consomme environ 30 mA.

Mettez le volume de l'amplificateur au minimum, puis mettez sous tension le circuit sans activer l'effet, vous devez

entendre le son de la guitare. Activez l'effet trémolo et vérifiez que les notes vibrent plus ou moins rapidement et profondément.

Carte son à effet Trémolo



Particulièrement adapté à une utilisation avec une guitare, mais pas seulement, ce kit vous permet de réaliser un effet trémolo analogique à pédale.

Cet appareil dispose d'une entrée et d'une sortie sur prises jack audio de 6,35 mm de haute qualité de marque Neutrik, il dispose de LED indiquant l'état de l'effet trémolo et de l'alimentation.

Des potentiomètres permettent le contrôle du volume, la profondeur, la vitesse, la forme de l'onde (carrée ou triangulaire), le tout dans un boîtier en métal très robuste. Il s'éteint automatiquement lorsqu'il n'y a pas de câble connecté à la prise jack d'entrée.

Alimentation : pile de 9 V type 6F22 (non incluse) ou par une alimentation externe (non incluse).

Dimensions (mm) : 115 x 65 x 25.

réf : k8110.....Carte son à effet Trémolo.....63,60 €

Etoile de Noël

Cette décoration de Noël technologique comporte 56 LED de type Neopixel et se contrôle avec n'importe quelle carte Arduino (non incluse). Vous pouvez la disposer au sommet de votre sapin de Noël ou à tout autre endroit de votre maison afin de produire des effets lumineux spectaculaires. Les LED Neopixel sont connectées en cascade et alimentées en parallèle. Cette configuration permet d'adresser chaque LED individuellement et de choisir la couleur entre les 256 combinaisons possibles pour chaque couleur de chaque LED RVB. Le nombre de combinaisons de couleurs est de $256 \times 256 \times 256$ soit 16,7 millions de couleurs. Le programme est disponible en téléchargement et permet de générer différents effets dès que l'étoile est mise sous tension.

Notez que ce kit nécessite une carte Arduino (fonctionne avec n'importe quelle carte Arduino) **qui n'est pas comprise dans le kit**. Elle devra être acquise séparément (nous vous conseillons une carte Arduino Micro), sauf si vous en avez déjà une.

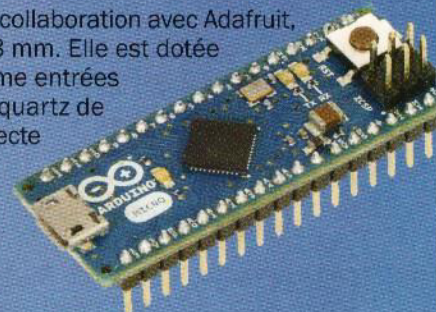
La communication entre Arduino et les LED Neopixel s'effectue via un bus de type « One-Wire » (un seul fil). L'alimentation est de 5 VDC et le débit maximal de transmission des données est de 800 kbps. Le protocole de commande du système Neopixel consiste à envoyer 3 groupes d'octets dans une chaîne de 24 bits, chacun contenant l'état d'éclairage de chaque couleur de base. Alimentation : 5 VDC. Dimensions (mm) : 148 x 148.



Référence : FT1300M
Prix : 33,80€

Carte Arduino Micro ATMEGA32U4

La carte Arduino Micro est basée sur le microcontrôleur ATMEGA32U4, elle a été développée en collaboration avec Adafruit, et dispose de toute la puissance d'Arduino dans un encombrement de seulement 48 mm x 18 mm. Elle est dotée de 20 entrées/sorties digitales (dont 7 peuvent être utilisées comme sorties PWM et 12 comme entrées analogiques), de 32 Ko de mémoire Flash (4 Ko utilisés par le bootloader), d'un oscillateur à quartz de 16 MHz, d'une connexion micro-USB, d'un connecteur ICSP et d'un bouton de reset. Elle se connecte simplement à un ordinateur via un câble micro-USB.



Microcontrôleur : ATmega32u4
Tension d'alimentation : de 7 VDC à 12 VDC
Entrées/sorties digitales : 20 (7 PWM)
Entrées analogiques : 12
Courant par broche I/O : 40 mA
Mémoire Flash : 32 ko (4 ko réservés par le bootloader)
SRAM : 2,5 ko
EEPROM : 1 ko
Fréquence de l'horloge : 16 MHz

Référence : ARDUINOMICRO
Prix : 30,30€

Boule décorative de Noël à LED intelligente RVB

Ce kit est composé de deux semi-sphères transparentes et de tous les composants nécessaires pour créer une boule décorative à LED intelligente RVB. Il est capable de créer des effets lumineux agréables tels que des dégradés de couleurs RVB et des flashes intermittents. La sphère s'allume automatiquement dans des conditions d'obscurité grâce à son phototransistor incorporé. Elle est facile à assembler.

Alimentation : pile bouton 3 V (CR2032 non incluse)

Consommation : 5 mA

Diamètre du circuit imprimé : 38 mm

Diamètre de la sphère : 50 mm



Référence : MK197
Prix : 8,20€

LED intelligente pour boule décorative

LED intelligente RVB de 5 mm de diamètre à changement de couleurs. Elle ressemble à une LED classique avec seulement 2 broches, mais dispose d'un microcontrôleur intégré. Longueur d'onde : rouge 620 nm ; vert 520 nm ; bleu 460 nm. Intensité lumineuse à 20 mA : rouge 750 mcd ; vert 1500 mcd ; bleu 750 mcd. Lorsqu'elle est sous tension, la LED change automatiquement de couleur en passant du rouge vers le lilas, puis le violet, puis le bleu, puis le vert, enfin le jaune pour revenir ensuite à la couleur rouge. Le cycle recommence ainsi indéfiniment.



Référence : LED5/RGB
Prix : 0,45€

COURS

MPLAB X

de Francesco Ficili et Vincenzo Germano

Cours MPLAB X
Première partie

Nous commençons notre voyage pour découvrir MPLAB X, le nouvel environnement de développement intégré développé et distribué par Microchip afin de remplacer l'environnement MPLAB IDE. C'est l'occasion d'apprendre à connaître les microcontrôleurs de la famille PIC32 de Microchip, et de passer dans la catégorie supérieure des microcontrôleurs fonctionnant en 32 bits. Première partie.

Notre longue expérience dans le domaine du développement d'applications intégrées nous a enseigné que l'une des problématiques les plus importantes du développement de telles applications est l'écriture du firmware ou programme du microcontrôleur.

Comme nous le savons tous, le développement d'un firmware pour microcontrôleur est très différent du développement de logiciels pour ordinateurs de bureau fonctionnant sous Windows. Dans le cas de l'écriture d'un logiciel, le système d'exploitation nous aide à organiser l'exécution des différents processus et simplifie l'accès aux ressources matérielles du PC. Lorsque nous développons une application pour un microcontrôleur, le problème est très différent.

Le système d'exploitation n'existe pas (ou s'il existe il s'agit d'un système d'exploitation en temps réel qui est très différent de Linux, Windows ou MacOS). Il nous appartient alors d'écrire le code qui nous permettra d'accéder aux différentes ressources du microcontrôleur. Cette tâche, qui est extrêmement complexe, devient de plus en plus difficile avec l'arrivée

des nouveaux microcontrôleurs et la complexité croissante des applications intégrées.

Si vous travaillez avec des périphériques analogiques simples ou des données séries, vous allez utiliser des microcontrôleurs simples à 8 bits. Cependant, ils possèdent déjà des périphériques d'une certaine complexité tels que l'USB et l'Ethernet.

Un smartphone moderne doit gérer au minimum une connexion GSM/GPRS pour la téléphonie et plusieurs périphériques comme le Wi-Fi, le Bluetooth et le GPS. En plus des périphériques sans fil, le smartphone dispose certainement d'une interface USB (qui, dans la plupart des cas, est un périphérique USB, mais les derniers modèles supportent l'USB OTG avec des fonctionnalités d'hôtes), d'un accéléromètre à 3 axes, d'un affichage tactile, etc. Tout cela est géré par un microcontrôleur 32 bits de type ARM Cortex ou similaire.

Pour permettre le développement de telles applications complexes, les outils de développement supportant ces microcontrôleurs 32 bits ont considérablement évolué,



Figure 1 : le logo du nouvel environnement de développement « MPLABX IDE » de Microchip.

allant des premiers IDE avec peu de fonctionnalités aux outils modernes de compilation et de débogage. De plus, ces environnements de développement (IDE) proposent de nombreux plug-ins et « framework » conçus pour répondre aux besoins les plus variés.

Par exemple, Microchip propose gratuitement un environnement de développement très moderne pour des applications utilisant ses produits, en particulier les nouveaux microcontrôleurs PIC32 (32 bits). L'outil mis à disposition par Microchip possède un IDE de base dont la dernière version est un environnement de développement basé sur la plate-forme « NetBeans » appelé « **MPLAB X IDE** ».

Cet environnement diffère considérablement de son prédécesseur, la version « MPLAB IDE 8.xx », ce qui constitue une avancée réelle dans les systèmes de développement fournis par Microchip. Les nouveautés les plus intéressantes qui différencient « MPLAB X IDE » de son prédécesseur sont les suivantes :

- **plateforme croisée** (Cross Platform) : contrairement à l'ancienne version, « MPLAB X IDE » fonctionne sous les principaux systèmes d'exploitation, à savoir Windows, Linux et MacOS ;
- **nouvelle plate-forme** : le nouvel IDE est basé sur une plate-forme dont le concept moderne (NetBeans) est à l'opposé d'Eclipse ;
- **nouvel outil de compilation** : avec l'arrivée de « MPLAB X IDE », des nouveaux outils de compilation apparaissent. Les anciens compilateurs C18, C30 et C32 sont remplacés par des nouveaux compilateurs dénommés respectivement XC8, XC16 et XC32 ;
- nouveaux « **framework** » sous forme de **plug-in** : contrairement aux anciennes versions, les « framework » ne sont plus fournis en tant que paquets externes (ou packages), mais sous forme de plug-ins importables, comme dans le cas du développement d'applications pour ordinateur de bureau.

Comme nous pouvons le constater à partir des points énumérés précédemment, cette nouvelle famille d'outils permet de réduire considérablement le temps de développement d'un projet.

Introduction à MPLABX IDE

Nous allons maintenant examiner en détail la composante fondamentale du nouvel outil de développement qui est MPLAB X IDE. Il s'agit d'un environnement de développement intégré (IDE) spécialement conçu pour les applications de développement utilisant les différentes familles de microcontrôleurs Microchip.

Il prend en charge toutes les familles : 8 bits, 16 bits et 32 bits.

MPLAB X IDE est un environnement de développement basé sur un IDE multi-langage et multiplateforme « NetBeans » développé par Oracle. Il bénéficie donc de toutes les fonctionnalités de l'IDE dont il est issu, comme par exemple la possibilité d'étendre ses fonctionnalités à l'aide de plug-ins.

La figure 2 illustre les principaux composants de ce puissant environnement de développement, qui comporte les éléments suivants :

- un **gestionnaire de projet** (Project Manager) qui est l'interface permettant la gestion d'un projet directement à partir de l'IDE. Grâce à ce composant, vous pouvez insérer et/ou supprimer des fichiers dans le projet lui-même et naviguer dans les différents fichiers qui le composent ;
- un **éditeur** (Editor) qui permet de modifier les fichiers du projet. L'éditeur de MPLAB X IDE, dérivé de l'éditeur de « NetBeans », est un outil puissant qui vous permet de naviguer dans le code à l'aide de la fonction évoluée « Go to » (Go to File, Go to Type, Go to Symbol, Go to Declaration...). Comme la plupart des éditeurs récents, il dispose de la fonction « Live Parsing » afin que les développeurs puissent voir les erreurs de syntaxe en temps réel tout en écrivant le code sans compilation. L'éditeur « parse » (analyse) le code en s'assurant qu'il peut être compilé. Cet outil de développement contient un instrument de visualisation de l'arborescence d'appel des fonctions, ainsi que d'autres caractéristiques intéressantes ;
- un **système de configuration** et de gestion intégré prenant en charge « Subversion », « CVS » et « Mercurial », ainsi qu'un support intégré de suivi des bugs. « Subversion » est un logiciel de contrôle des versions, « CVS » étant son prédécesseur. « Mercurial » est un autre logiciel de gestion des versions d'un logiciel ;
- un **ensemble d'outils** supportant **différents langages** (Language Tools) et donc par conséquent différents compilateurs/assembleurs qui composent la chaîne de compilation. Certains de ces outils sont présents de manière native (intégrés) dans le package de base de l'IDE (par exemple MPASM), tandis que d'autres, comme les compilateurs C, doivent être installés séparément pour être automatiquement intégrés à l'IDE. En plus des compilateurs Microchip spécialement développés pour MPLAB X IDE (XC8, XC16 et XC32), il est possible d'utiliser des produits tiers ;
- un **débogueur** (Debugger) au niveau du code, qui permet de déboguer l'application directement au niveau du code source en créant un lien entre ce dernier et l'application binaire téléchargée dans la mémoire flash du microcontrôleur ;
- une **interface de programmation** qui permet d'interagir avec différents programmeurs de chez Microchip, y compris les platines de développement dotées d'une programmation en circuit (ICSP) ou avec des programmeurs d'autres fabricants ;
- un **simulateur** dénommé **MPLAB SIM**, permettant de simuler le code source si vous ne disposez pas de débogueur. Il existe également la possibilité d'utiliser

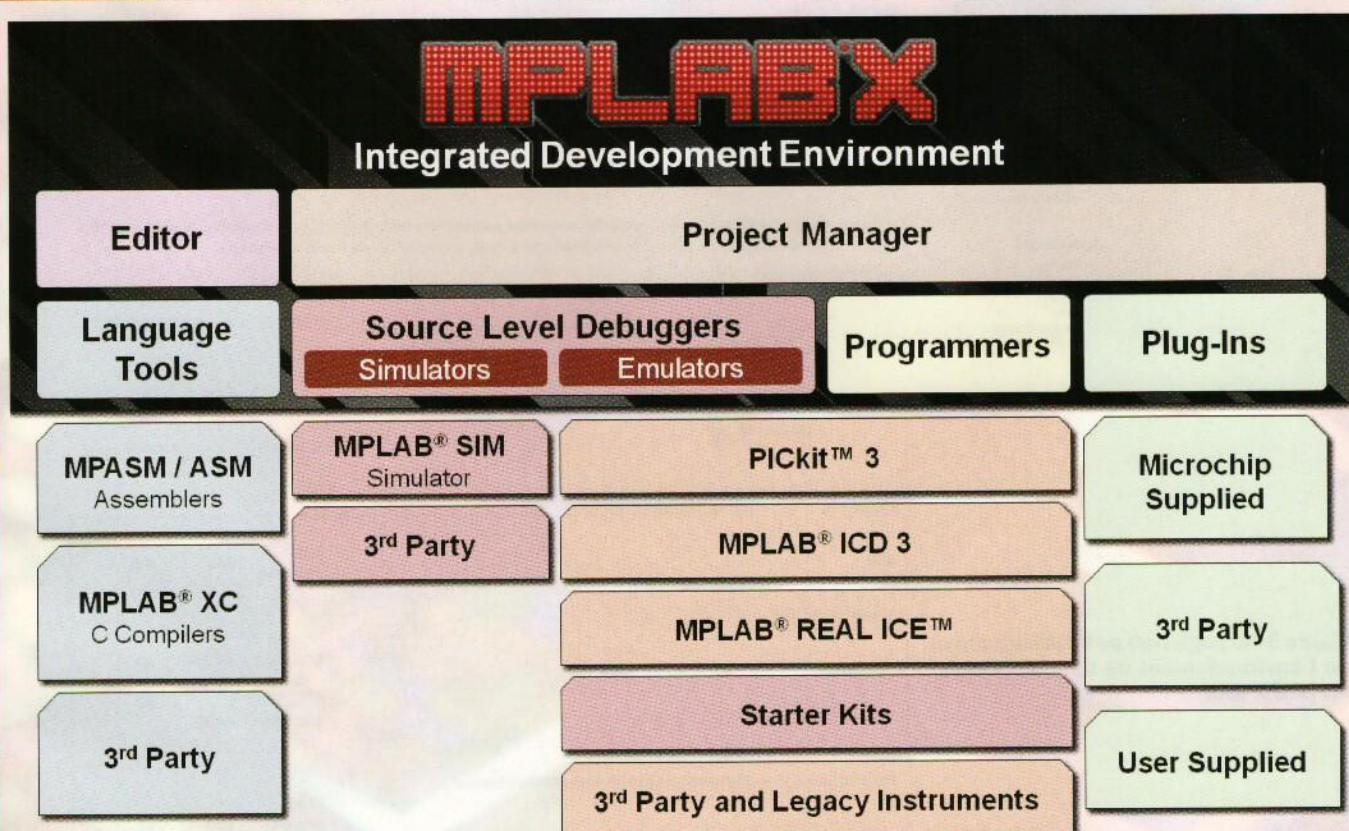


Figure 2 : schématisation de l'environnement MPLAB X IDE.

au sein de l'IDE d'autres simulateurs plus puissants, tels que celui de Proteus ;

- un **utilitaire de conversion** intégré qui permet de convertir automatiquement les anciens projets développés avec les versions 8.XX de MPLAB vers des projets de types MPLABX IDE.

Comme nous l'avons mentionné précédemment, l'IDE dérive directement de « NetBeans », de sorte qu'il est possible d'utiliser une multitude de plug-ins développés pour l'environnement natif, en plus de ceux développés par Microchip. MPLAB X IDE étant un environnement ouvert, il est possible de télécharger le code source et de développer vos propres plug-ins.

Installation de l'environnement de développement MPLAB X IDE

Maintenant, nous allons installer l'environnement de développement MPLAB X IDE que nous utiliserons pendant toutes les leçons de ce cours. Dans cette partie du cours, nous allons installer les composants de base pour bien débuter, c'est-à-dire l'environnement **MPLAB X IDE** et le compilateur **XC32**.

En effet, nous nous orientons vers des microcontrôleurs 32 bits qui offrent bien plus de possibilités que les 8 bits pour un prix à peine plus élevé. Tout d'abord, nous devons télécharger les deux logiciels sur le site web de Microchip aux adresses suivantes :

- MPLABX IDE : <http://www.microchip.com/mplab/mplab-x-ide>
- XC32 : <http://www.microchip.com/mplab/compilers>

Sur chacune de ces pages, vous devez cliquer sur l'onglet « **Downloads** » qui se situe vers le bas. Vous y trouverez les dernières versions en téléchargement gratuit pour les principaux systèmes d'exploitation Windows, Linux et MacOS.

Notez que dans ce cours, nous nous **référerons toujours à la version de Windows**. Cependant, comme l'environnement de développement MPLAB X IDE est multiplateforme, nos explications tout au long de ces leçons seront toujours valables pour les autres versions.

En figure 3, vous pouvez voir une capture d'écran de la page de téléchargement de l'environnement de développement MPLAB X IDE, en ayant cliqué au préalable sur l'onglet « Downloads ».

Au moment où ces pages sont écrites, les **versions 4.00 de MPLAB X IDE** et **1.44 du compilateur XC32** sont disponibles en téléchargement. Il se peut que les versions soient différentes lorsque vous les téléchargerez, prenez toujours les dernières versions.

Une fois les logiciels téléchargés, démarrez la procédure d'installation et suivez les instructions jusqu'à la fin. Cette procédure d'installation est décrite pour Windows 7, 8 et 10. Elle peut différer pour les autres systèmes d'exploitation tels que Linux ou MAC OS.

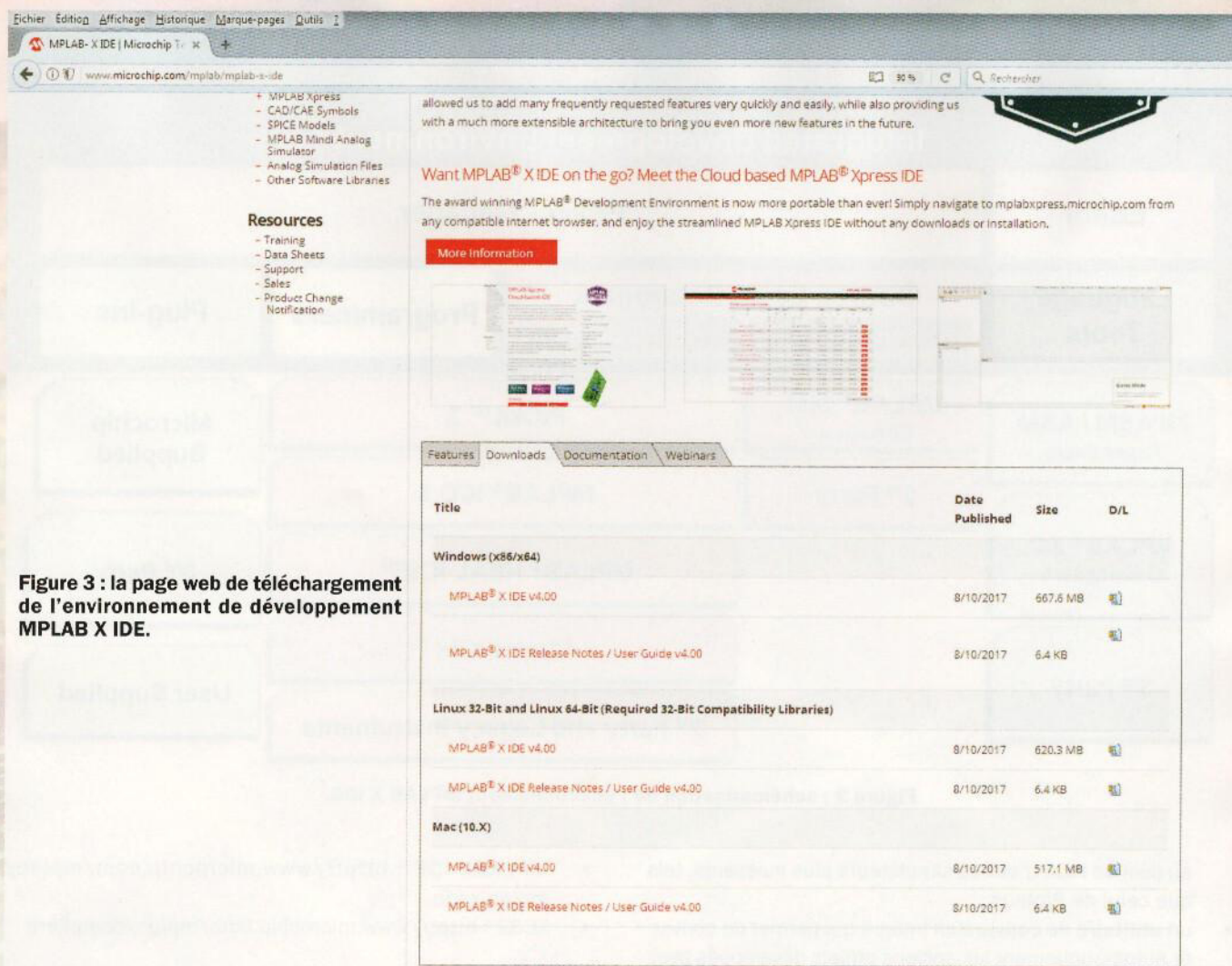


Figure 3 : la page web de téléchargement de l'environnement de développement MPLAB X IDE.

Commençons par installer l'environnement MPLAB X IDE, pour cela vous devez exécuter le fichier nommé « **MPLABX-v4.00-windows-installer.exe** » en tant qu'administrateur (son nom peut être différent en fonction de la version). La figure 4a illustre l'écran de démarrage de l'assistant d'installation de MPLAB X IDE version 4.00.

Cliquez sur le bouton « **Next** », l'écran de la figure 4b apparaît vous invitant à lire l'utilisation de la licence du logiciel et à l'accepter. Cochez la case « **I accept the agreement** », puis cliquez sur le bouton « **Next** ». Une nouvelle fenêtre apparaît correspondant à la figure 4c.

Elle indique le chemin par défaut de l'installation de l'environnement MPLAB X IDE, vous pouvez le modifier, mais nous vous conseillons d'utiliser le chemin par défaut. En dessous se trouve les réglages du « **Proxy** » (serveur) qui permet de télécharger les plug-ins.

Nous vous conseillons dans un premier temps de ne pas utiliser le « **Proxy** ». Sélectionnez la case « **No Proxy** », nous vous montrerons par la suite comment modifier cette option et obtenir une connexion avec le proxy. Cliquez sur le bouton « **Next** », la fenêtre de la figure 4d apparaît. Elle permet de sélectionner les programmes que vous désirez installer.

Cochez toutes les cases comme visible en figure 4d puis cliquez sur le bouton « **Next** ». L'écran de la figure 4e s'affiche en vous indiquant que l'installation de MPLAB X IDE peut commencer.

Cliquez sur le bouton « **Next** », le processus d'installation commence (figure 4f). Cela peut prendre plusieurs minutes. Au bout d'un certain temps, le programme d'installation

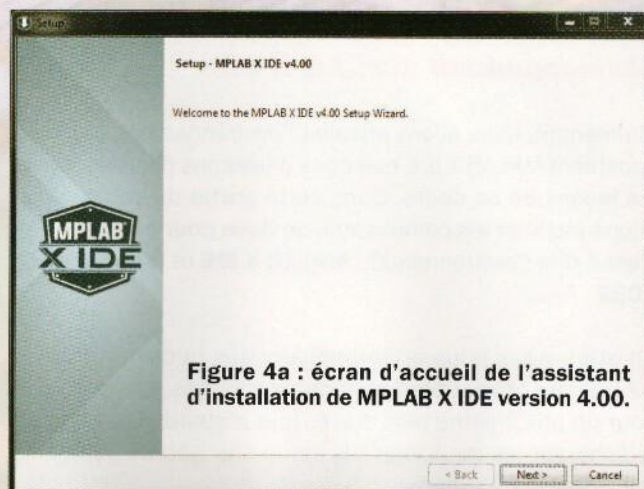


Figure 4a : écran d'accueil de l'assistant d'installation de MPLAB X IDE version 4.00.

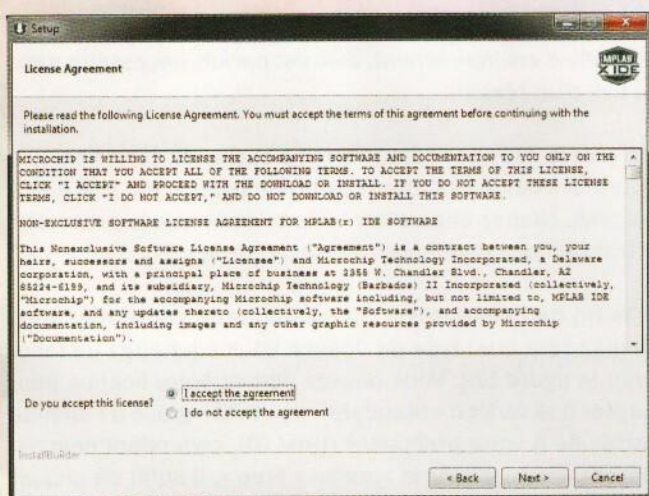


Figure 4b : termes de la licence de MPLAB X IDE, cochez la case « I accept the agreement » pour continuer.

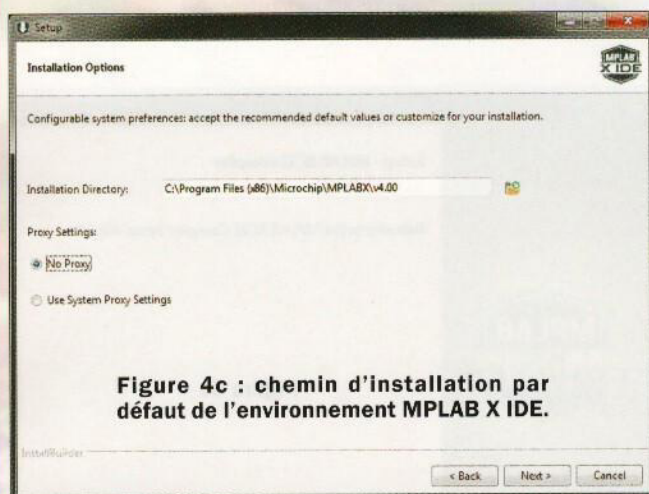


Figure 4c : chemin d'installation par défaut de l'environnement MPLAB X IDE.

vous demande d'installer un logiciel supplémentaire (voir la figure 4g). Il s'agit du logiciel « **Install MPLAB Communications Library** », cliquez sur installer.

Le processus d'installation continue jusqu'à l'affichage de la figure 4h qui indique que vous êtes arrivé à la fin du processus d'installation.

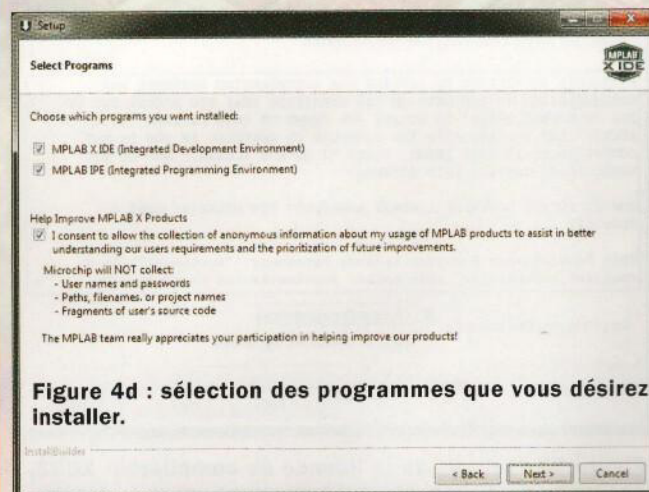


Figure 4d : sélection des programmes que vous désirez installer.

Décochez toutes les cases, elles représentent des liens vers des pages web du site de Microchip qui s'afficheront lorsque vous aurez cliqué sur le bouton « **Finish** », si les cases sont cochées. Cela ne présente que peu d'intérêt pour l'instant. Les cases étant décochées, cliquez alors sur le bouton « **Finish** » pour terminer l'installation.

Vous devez apercevoir 3 icônes sur votre bureau nommées « MPLAB X IDE v4.00 », « MPLAB driver switcher » et « MPLAB IPE v4.00 ». La première est l'environnement de développement, la seconde vous indique qu'il est possible d'importer vos anciens projets fonctionnant sous MPLAB 8.XX vers MPLAB X (si vous en possédez).

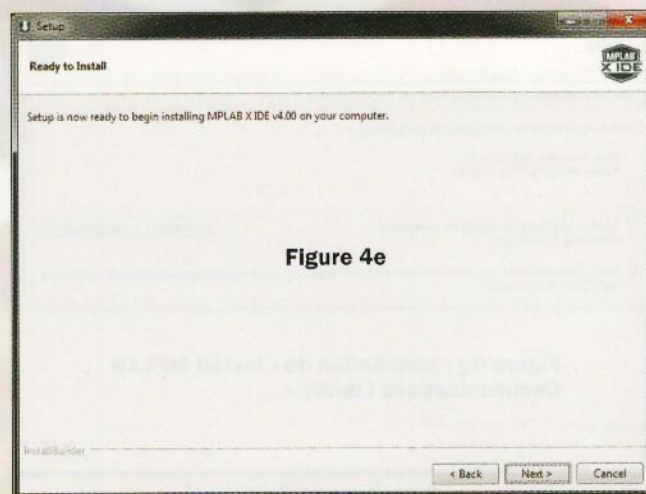


Figure 4e

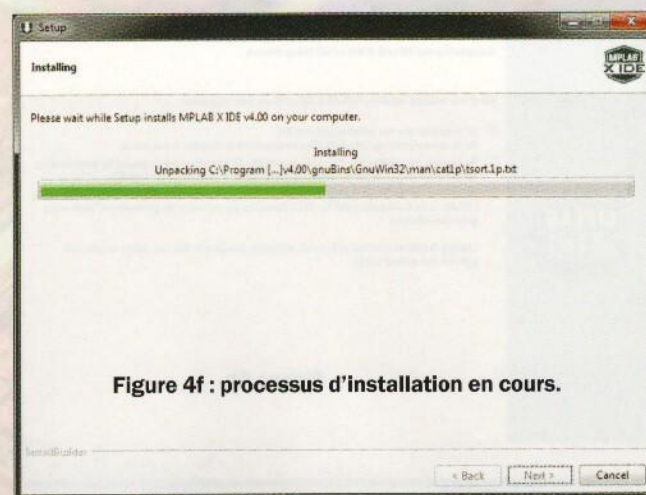


Figure 4f : processus d'installation en cours.

La 3^{ème} icône correspond à l'environnement de programmation et permet d'importer un fichier au format « **hex** » afin de programmer un microcontrôleur.

Cependant, il est possible de programmer un microcontrôleur directement à partir de MPLAB X IDE, comme nous le verrons dans la suite du cours.

Passons maintenant à l'installation du compilateur XC32, pour cela vous devez exécuter le fichier nommé « **xc32-v1.44-full-install-windows-installer.exe** » en tant qu'administrateur.

La fenêtre de la figure 5a s'ouvre, cliquez sur le bouton « Next », l'écran de la figure 5b apparaît vous invitant à lire l'utilisation de la licence du logiciel et à l'accepter. Cochez la case « I accept the agreement » puis cliquez sur le bouton « Next ». Une nouvelle fenêtre s'ouvre (figure 5c) indiquant les différentes licences disponibles, par défaut la licence « Free » est cochée, les autres sont grisées car vous devez disposer d'un compte client chez Microchip.

Notez que la licence « Free » est complètement opérationnelle et conviendra parfaitement pour nos leçons. Ensuite, cliquez sur le bouton « Next », une nouvelle fenêtre apparaît

Pour information la case en dessous permet de modifier la variable d'environnement, cela est parfois nécessaire dans le cas d'un réseau.

Laissez la case par défaut cochée et cliquez sur « Next », la figure 5f s'ouvre indiquant que le compilateur est prêt à être installé, cliquez encore sur « Next » pour que le processus d'installation commence (voir la figure 5g).

À la fin de l'installation du compilateur, ce dernier vous demandera quel type de licence vous souhaitez installer (voir la figure 5h). Vous pouvez activer votre licence pour passer à la version « Standard » ou « PRO » grâce à l'identité attribuée à votre ordinateur (Host ID), cependant cela est payant. Pour utiliser la version « Free », il suffit de cliquer sur « Next » lorsque cet écran apparaîtra.

Vous verrez alors s'afficher la figure 5i qui indique que l'installation du compilateur XC32 est terminée.

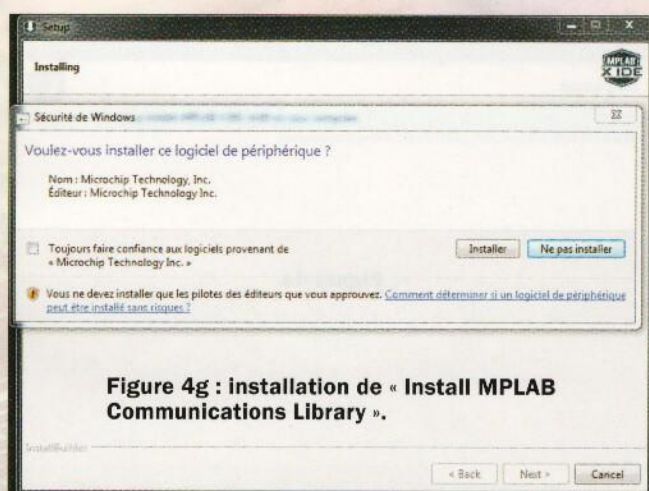


Figure 4g : installation de « Install MPLAB Communications Library ».

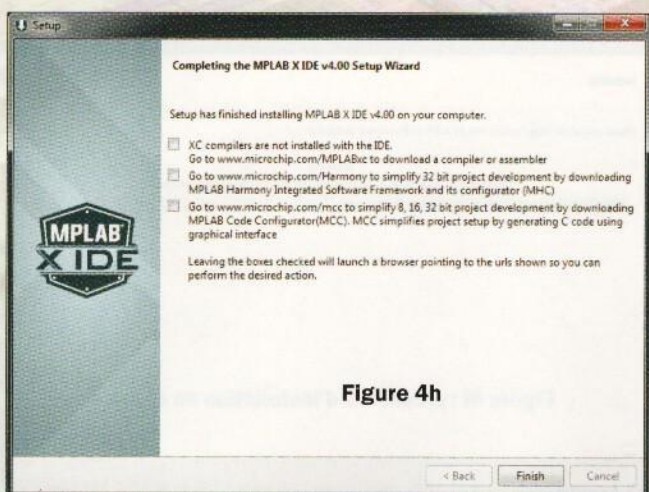


Figure 4h

correspondant à la figure 5d. Elle indique le chemin par défaut de l'installation qui est « C:\Program Files (x86)\Microchip\xc32\v1.44 ».

Vous pouvez modifier l'emplacement de l'installation en cliquant sur l'icône représentant un dossier avec une flèche verte (à droite) du champ. Si l'emplacement vous convient, cliquez sur le bouton « Next ». La fenêtre de la figure 5e apparaît, par défaut la case « Apply setting to all users of this machine » est cochée.

Cela veut dire que les paramètres d'installation seront appliqués à tous les utilisateurs de l'ordinateur.



Figure 5a

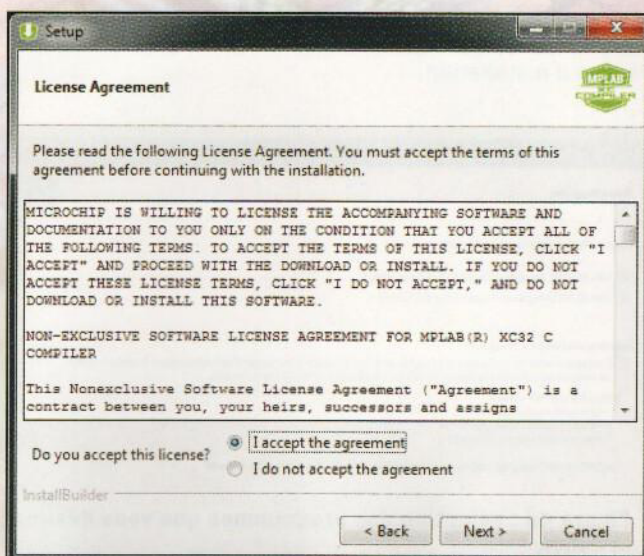


Figure 5b : termes de la licence du compilateur XC32, cochez la case « I accept the agreement » pour continuer.

Le compilateur Microchip est distribué sous 3 versions différentes : « Free », « Standard » et « PRO ». Elles diffèrent au niveau de l'optimisation du code effectuée par le compilateur.

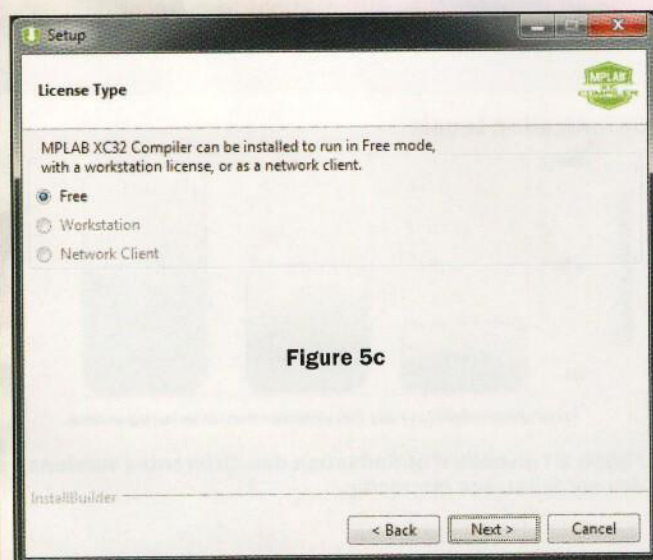


Figure 5c

En figure 6, vous apercevez les différences entre les versions. Cette caractéristique est extrêmement importante pour le développement d'applications industrielles, où les économies de mémoire FLASH (du programme) peuvent avoir un impact significatif sur les coûts de production, cependant pour le particulier dans un cadre didactique comme le nôtre cela a peu d'importance.

Il suffit simplement de choisir un microcontrôleur avec une quantité de mémoire FLASH supérieure.

Description de l'interface

Commençons maintenant par découvrir le nouvel environnement de MPLAB X IDE, en décrivant l'interface.

Lancez le logiciel en effectuant un double clic sur l'icône « MPLAB X IDE v4.00 ».

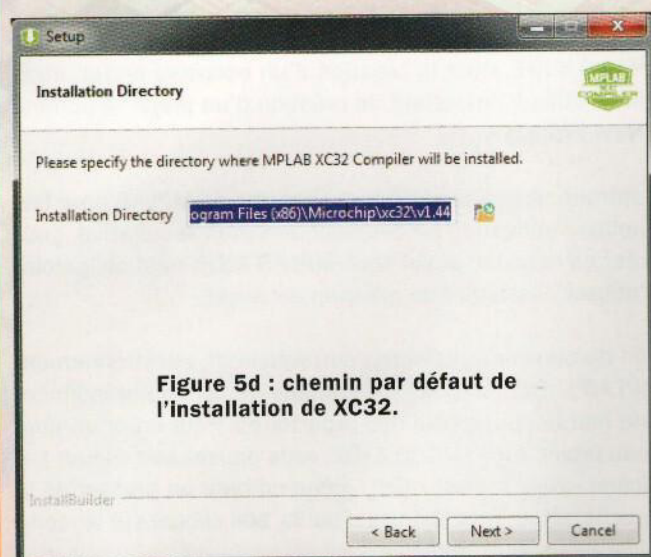


Figure 5d : chemin par défaut de l'installation de XC32.

Au démarrage, l'IDE affiche un écran qui se superpose à la fenêtre de l'éditeur (voir la figure 7) et fournit une multitude d'informations utiles sur l'environnement.

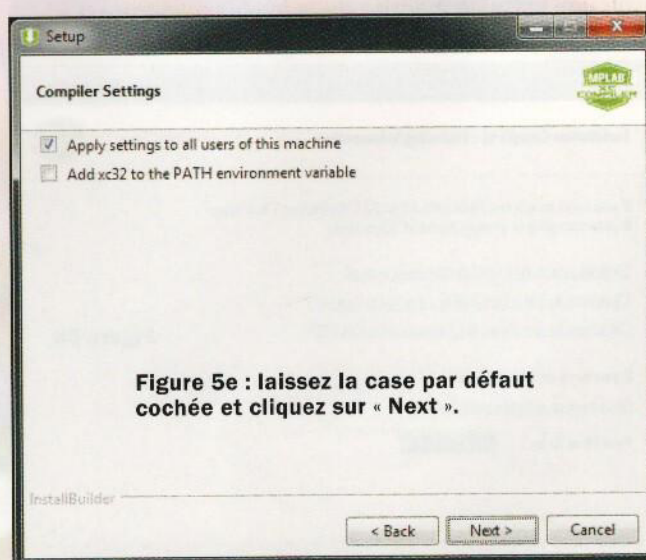


Figure 5e : laissez la case par défaut cochée et cliquez sur « Next ».

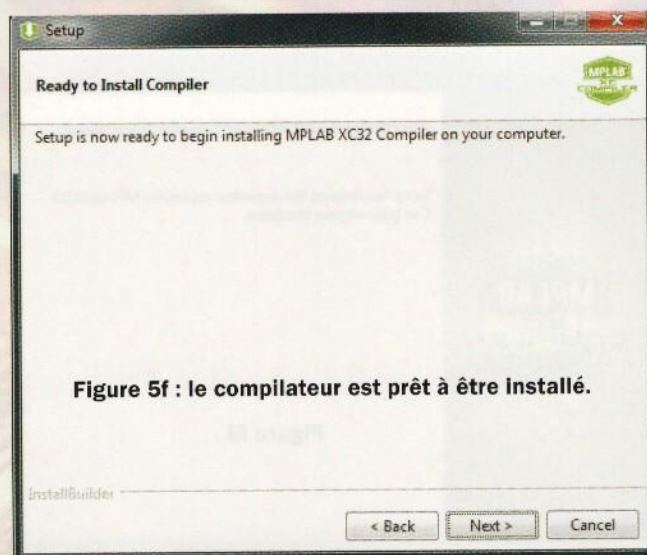


Figure 5f : le compilateur est prêt à être installé.

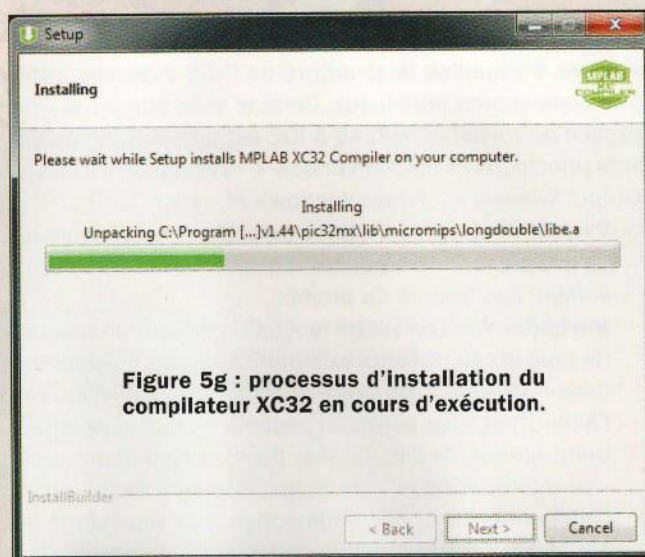


Figure 5g : processus d'installation du compilateur XC32 en cours d'exécution.

Il s'agit d'une fenêtre d'accueil disposant de raccourcis vous permettant de lancer l'éditeur afin de commencer un nouveau projet, de visualiser des didacticiels, de télécharger des plug-ins, de visiter les forums, etc. Cependant il faut avoir une certaine maîtrise de la langue anglaise.

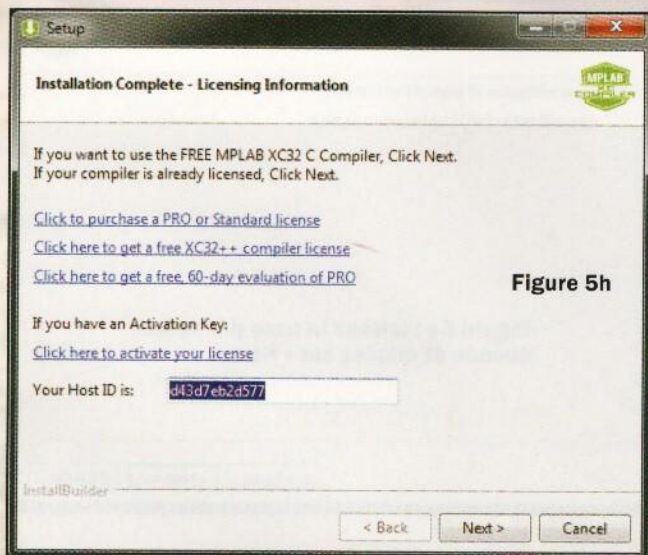


Figure 5h

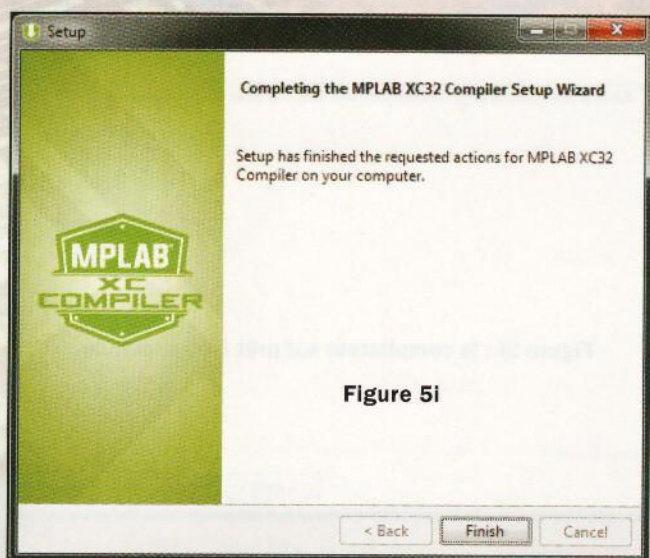


Figure 5i

La figure 8 visualise la structure de l'IDE avec ses différents composants principaux. Comme vous pouvez le voir, l'espace de travail de MPLAB X IDE est divisé en 5 composants principaux : « Project Window », « Navigator Window », « Output Window », « Editor Window » et « Main Toolbar » :

- **Project Window** : il s'agit de la fenêtre du gestionnaire de projet, à l'intérieur de laquelle est reporté l'ensemble des fichiers du projet ;
- **Navigator Window** : cette fenêtre de navigation répertorie tous les identifiants existants du fichier sélectionné dans l'éditeur (c'est-à-dire le fichier qui s'affiche dans l'éditeur) et vous permet d'accéder à chacun en effectuant un double clic dessus. Il existe également dans « Navigator Window » un onglet appelé « Dashboard » (Tableau de bord) qui résume tous les composants du projet ouvert. Par exemple, le type de microcontrôleur

utilisé (Device), le type de compilateur utilisé (Compiler Toolchain), la quantité de mémoire FLASH et RAM disponible, les ressources de débogage, etc. ;

- **Main Toolbar** : il s'agit de la barre d'outils située en haut de la fenêtre. Elle contient tous les principaux

Optimization Levels

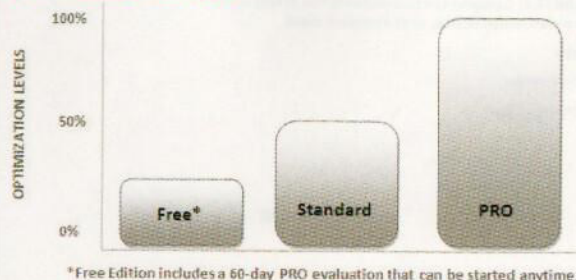


Figure 6 : niveaux d'optimisation des différentes versions des compilateurs Microchip.

contrôles de l'environnement de développement, tels que les boutons utilisés pour compiler le code source en mode débogage et en « release mode » (phase de test), les outils de débogage « run », « halt », « step into », « step over », etc. (respectivement : exécuter, arrêter, entrer dans, passer, etc.) et les raccourcis pour créer ou ouvrir des fichiers et pour accéder à toutes les options du système ;

- **Editor Window** : il s'agit de la fenêtre de l'éditeur dans laquelle le code source est écrit et donc développé. Elle contient sa propre barre d'outils. Il suffit de laisser le curseur de la souris dessus pour visualiser la fonction de chaque icône ;
- **Output Window** : c'est la fenêtre de sortie de l'IDE, elle fournit les informations de la compilation (erreurs ou compilation réussie), la connexion à des outils externes, le débogage, etc.

Créer un projet avec MPLAB X IDE

Nous allons maintenant créer notre premier projet sous MPLAB X IDE. Pour la création d'un nouveau projet, nous allons utiliser l'assistant de création d'un projet dénommé « New Project ».

Contrairement aux anciennes versions de MPLAB pour lesquelles l'utilisation de cet assistant était facultative, pour créer un nouveau projet sous MPLAB X IDE il est obligatoire d'utiliser l'assistant de création de projet.

Afin de pouvoir fonctionner correctement, l'environnement MPLAB X IDE nécessite un système de fichiers prédéfini et une hiérarchie précise des répertoires. Pour créer un nouveau projet dans MPLAB X IDE, vous pouvez soit cliquer sur l'icône « New Project » (2^{ème} icône en haut en partant de la gauche) situé dans la barre d'outils, soit cliquer sur le menu « File » → « New Project ».

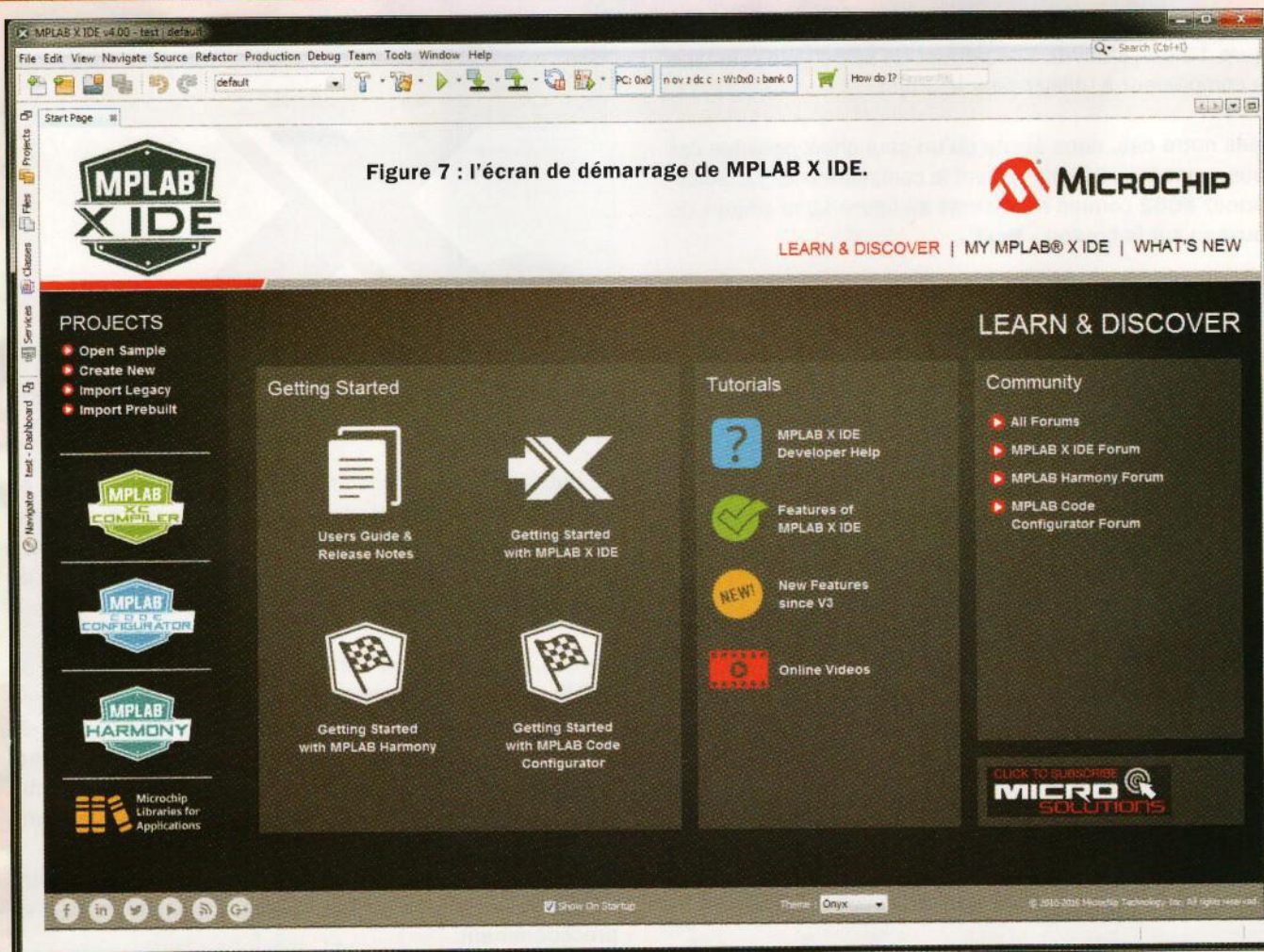


Figure 7 : l'écran de démarrage de MPLAB X IDE.

Une fenêtre s'ouvre semblable à celle de la figure 9. Dans cette fenêtre, sélectionnez l'option « **Microchip Embedded** » puis « **Standalone Project** » et cliquez ensuite sur le bouton « **Next** » pour continuer.

Notez que vous pouvez aussi sélectionner « **Samples** » (Exemples) et ensuite « **Microchip Embedded** ».

Vous voyez apparaître dans la partie de droite une liste de modèles de projets pour les différentes familles de microcontrôleurs Microchip (PIC18, PIC24, dsPIC et PIC32).

Pour l'instant, choisissons la première option « **Standalone Project** », en cliquant sur « **Next** » vous obtenez la fenêtre représentée en figure 10.

La liste déroulante « **Family** » permet de sélectionner une famille de microcontrôleurs afin de simplifier la recherche d'un microcontrôleur particulier que vous désirez utiliser dans votre projet. Pour notre premier projet, nous choisissons le modèle **PIC32MX795F512L**.

Donc dans la liste déroulante « **Family** », sélectionnez « **32-bit MCUs (PIC32)** » et dans la liste déroulante « **Device** », qui correspond à tous les microcontrôleurs de la famille PIC32 (c'est-à-dire la famille choisie dans la liste déroulante « **Family** »), sélectionnez le modèle PIC32MX795F512L et cliquez de nouveau sur le bouton « **Next** » pour continuer. Une nouvelle fenêtre s'ouvre semblable à celle de la figure 11, dans laquelle vous pouvez choisir le type de programmeur/débogueur que vous utilisez.

Dans le cas où vous ne disposez pas de programmeur présent dans la liste, vous pouvez choisir d'utiliser le **simulateur** (comme dans le cas de la figure 11).

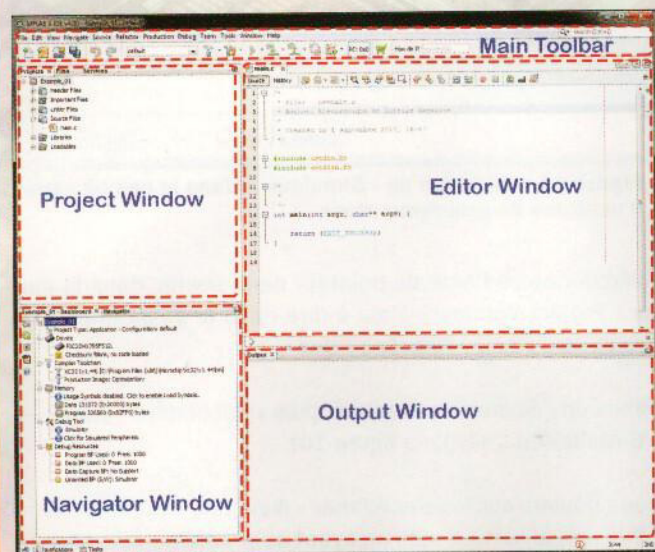


Figure 8 : l'interface de MPLAB X IDE.

Cliquez de nouveau sur le bouton « **Next** », la fenêtre de la figure 12 doit s'ouvrir. Elle permet de sélectionner le type de compilateur à utiliser pour le projet.

Dans notre cas, nous avons qu'un seul choix possible car nous avons installé uniquement le compilateur XC32. Sélectionnez **XC32** comme représenté en figure 12 et cliquez de nouveau sur le bouton « **Next** ».

Nous sommes presque à la fin du processus de création d'un nouveau projet. Vous devez obtenir une fenêtre semblable à celle de la figure 13, il suffit de taper un nom de projet dans le champ « Project Name ». Nous avons choisi comme nom de projet « **Exemple_01** », mais vous pouvez taper un autre nom de votre choix.

Notez que dans le champ « **Project Folder** », le nom de votre projet s'ajoute automatiquement au chemin d'accès du projet. Vous pouvez aussi modifier l'emplacement de votre projet selon votre convenance en cliquant sur le bouton « **Browse** » à côté du champ « **Project Location** ».

Cliquez sur le bouton « **Finish** » pour créer votre premier projet sous MPLAB X IDE comme visible en figure 13. À ce stade, MPLAB X IDE génère les fichiers et les répertoires nécessaires au projet.

Notez que par défaut le projet est placé dans le dossier suivant : « C:\Users\...\MPLABXProjects », où les points correspondent au nom de la session de votre ordinateur.

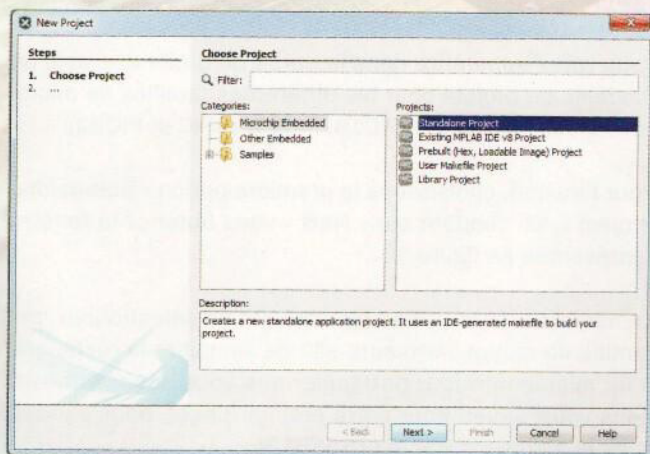


Figure 9 : fenêtre de l'assistant du choix du type de projet.

Le projet généré par MPLAB X IDE comporte l'extension « **.X** » pour faciliter l'identification de ce type de projets, de sorte qu'il n'y ait pas de confusion avec un ancien projet généré par MPLAB 8.XX (voir la figure 13a).

Notre premier programme

Nous allons maintenant écrire un premier programme basique qui servira d'application pratique. Pour cela, nous allons utiliser l'un des modèles (template) intégré à l'environnement de développement.

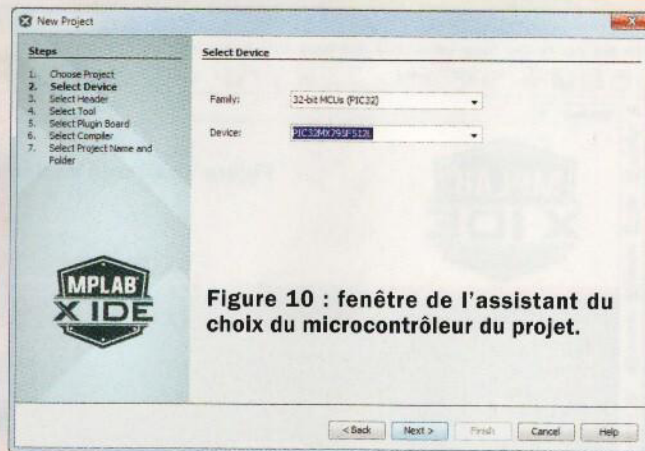


Figure 10 : fenêtre de l'assistant du choix du microcontrôleur du projet.

Le but de ce programme est de faire clignoter une LED en utilisant le simulateur, pour cet exemple. Nous n'aurons pas de LED physique qui clignotera, nous allons simplement inverser de manière cyclique l'état logique d'une broche de l'un des PORT du microcontrôleur.

Si vous disposez d'une carte de développement à base de PIC32, vous pourrez tester le programme en essayant de faire clignoter une LED. Cependant, il faut tenir compte de certains aspects concernant le passage d'un environnement de simulation à un environnement réel. Nous traiterons de cet aspect dans les prochaines leçons de ce cours.

Pour commencer, nous devons ajouter un fichier à notre projet. Ouvrez le projet « **Exemple_01** » que vous avez créé précédemment.

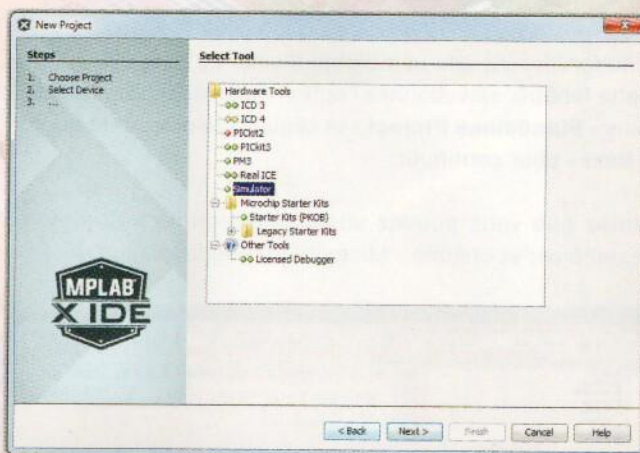


Figure 11 : sélection de « **Simulator** » dans le cas où vous n'avez pas de programmeur.

Sélectionnez à l'aide du pointeur de la souris, dans la partie « **Project Window** » c'est-à-dire dans le gestionnaire de projet, le dossier « **Sources Files** ».

Faites un clic droit de la souris, puis sélectionnez « **New** » → « **C Main File...** » (voir la figure 14).

Vous pouvez aussi sélectionner « **New File** » dans le menu « **File** » de la barre d'outil en ayant au préalable sélectionné le dossier « **Sources Files** », sinon le fichier risque d'être créé dans un autre dossier du projet.

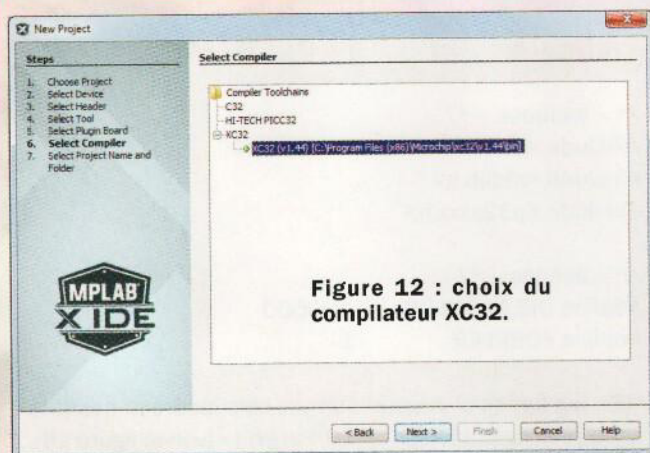


Figure 12 : choix du compilateur XC32.

Une fenêtre s'ouvre semblable à celle de la figure 14a. Sélectionnez avec la souris « **C** » dans la colonne « **Categories** » puis « **C Main File** » dans la colonne « **File Types** » puis cliquez sur « **Next** ».

À ce stade, une nouvelle fenêtre apparaît (voir la figure 15) dans laquelle vous pouvez choisir le chemin dans lequel enregistrer le nouveau fichier, sinon laissez le chemin par défaut et cliquez sur « **Finish** » pour créer le fichier nommé « **main.c** ».

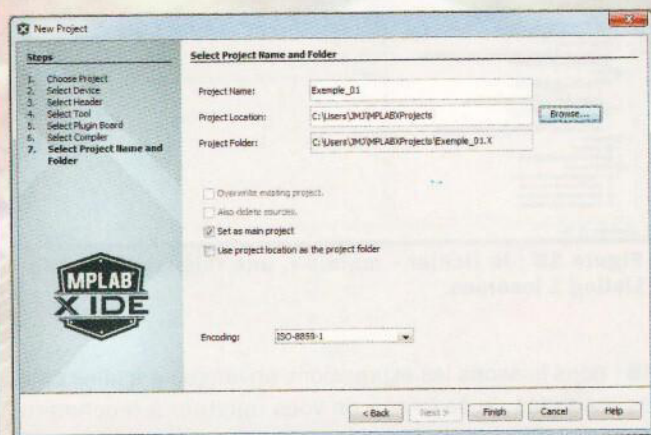


Figure 13 : cliquez sur le bouton « **Finish** » pour créer votre premier projet sous MPLAB X IDE.

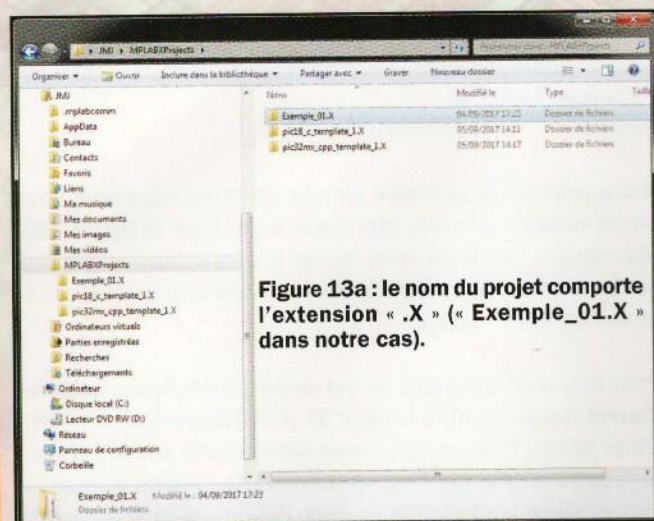


Figure 13a : le nom du projet comporte l'extension « **.X** » (« **Exemple_01.X** » dans notre cas).

Maintenant, ouvrons le fichier « **main.c** » en allant dans le gestionnaire de projet et en cliquant 2 fois sur « **Sources Files** » pour développer le dossier. Cliquez ensuite 2 fois sur le fichier « **main.c** » afin qu'il s'ouvre dans l'éditeur.

Modifiez le fichier en insérant les lignes de code présentes dans le listing 1. Ecrivez les lignes avant « **int main(int argc, char** argv) {** ». Vous devez obtenir un résultat semblable à celui de la figure 16.

Ensuite, nous définissons une variable de type compteur, que nous appelons génériquement un compteur, de type long non signé (**unsigned long**) et initialisée à la valeur de **DELAY_VALUE** :

```
/* delay counter */
unsigned long counter = DELAY_VALUE;
```

Ensuite, avec la ligne suivante, nous définissons le registre directionnel du « **PORT D** » afin que toutes ses broches soient configurées en tant que sorties :

```
/* set ddr register */
TRISD = 0x0000;
```

Ces 4 lignes ci-dessus doivent être par contre insérées après la ligne « **int main(int argc, char** argv) {** ».

Le registre de direction (DDR = Data Direction Register) est un registre du microcontrôleur ayant une fonction spécifique. Il s'agit d'un registre de type « **SFR** » (Special Function Register), dont la fonction est de paramétrer la configuration des broches d'entrées/sorties de l'un des PORT du microcontrôleur.

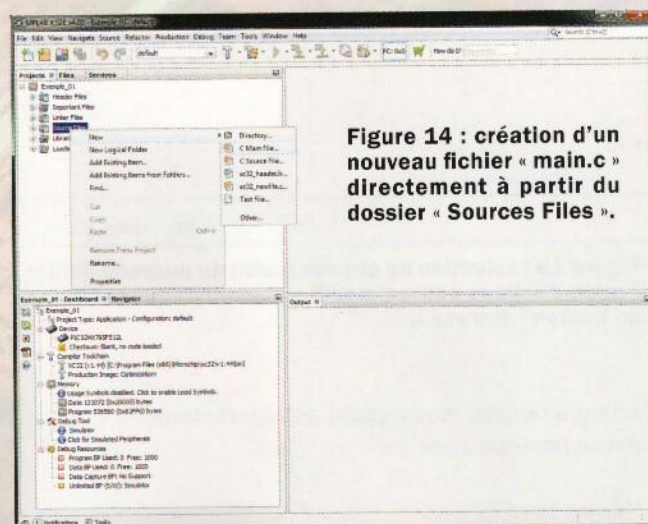


Figure 14 : création d'un nouveau fichier « **main.c** » directement à partir du dossier « **Sources Files** ».

Il existe donc un registre « **SFR** » pour chaque PORT physique du microcontrôleur. En fixant la valeur de l'un des bits du registre à « **0** », la broche correspondante du PORT associé au registre se configure en tant que sortie digitale (0 = output ou sortie). Si le même bit est fixé à la valeur « **1** », la même broche fonctionnera comme une entrée digitale (1 = input ou entrée).

Comme nous devons piloter une LED, nous configurons le registre afin que toutes les broches du PORT correspondant fonctionnent en sorties.

À ce stade, nous ajoutons au code source une boucle « **while** » infinie dans laquelle se trouve une seconde boucle « **while** » dont la condition de sortie est donnée par la valeur du compteur (si le compteur vaut 0 alors on sort, sinon on continue).

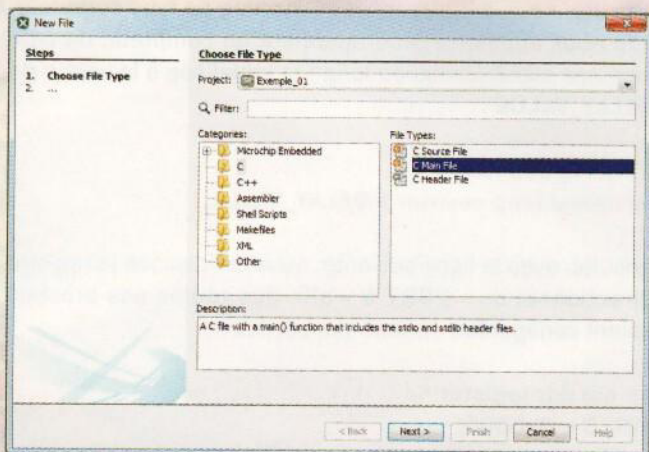


Figure 14a : autre méthode de création d'un nouveau fichier « main.c ».

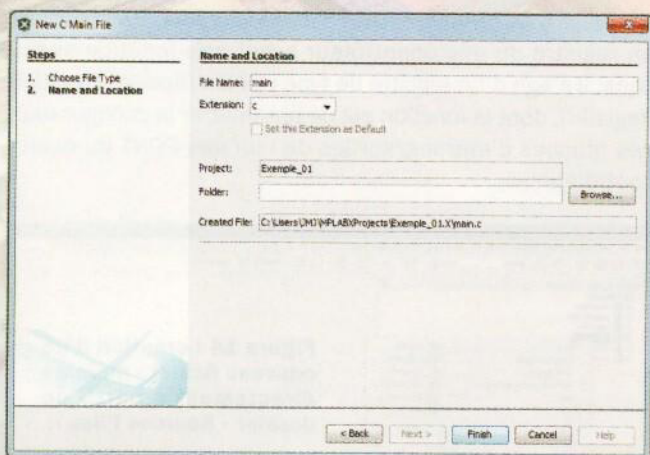


Figure 15 : sélection du chemin (path) du nouveau fichier « main.c ». Vous pouvez modifier l'emplacement à l'aide du bouton « Browse ».

À chaque boucle, le compteur est décrémenté, il s'agit de l'instruction « counter--; ».

```
/* delay loop */
while (counter)
{
    /* decrement counter */
    counter--;
}
```

C'est le cycle réel de retard. Une fois que le programme sort de ce cycle, le compteur est rechargé pour le prochain cycle et l'état précédent de la broche RD0 est inversé.

Listing 1

```
/* -- includes -- */
#include <stdio.h>
#include <stdlib.h>
#include <p32xxx.h>

/* -- defines -- */
#define DELAY_VALUE    100000
#define FOREVER        1
```

NB : les lignes ci-dessus doivent être insérées avant la ligne « int main(int argc, char** argv) { » (voir la figure 16).

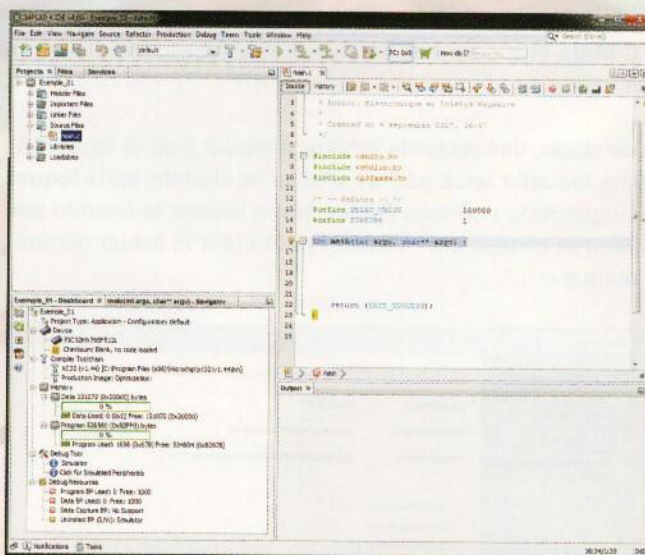


Figure 16 : le fichier « main.c », une fois les lignes du Listing 1 insérées.

NB : nous laissons les expressions en langue anglaise dans les exemples de code afin de vous habituer à reconnaître les instructions et les fonctions dans d'autres programmes, notamment dans les exemples de codes fournis par Microchip.

```
/* set delay value to DELAY_VALUE */
counter = DELAY_VALUE;
```

```
/* toggle RD0 */
PORTDbits.RD0 = ~PORTDbits.RD0;
```

Notre premier programme simple est donc complété, vous devez obtenir un code identique à celui de la figure 16a. Maintenant, nous pouvons lancer la simulation en cliquant sur le bouton « **Debug Main Project** » dans la barre d'outils et voir ce qui se passe.

Pour mieux comprendre ce qui se passe, insérons un **point d'arrêt** (break point) à la ligne 37 (PORTDbits.RD0 = ~PORTDbits.RD0;). Pour cela, cliquez simplement sur le bord de la fenêtre de l'éditeur sur le numéro de la ligne et ensuite cliquez sur le bouton « Debug Main Project » (voir la figure 17).

Après un laps de temps de vérification de l'intégrité du programme, l'exécution s'arrête à la ligne 37 (figure 18), où nous avons placé le point d'arrêt.

Si nous redémarons le programme en appuyant sur « **Continuer** » (ou encore sur la touche F5), nous répétons de nouveau la boucle « **while** » principale (FOREVER).

La ligne surlignée en vert à l'arrêt devient rose lorsque le programme s'exécute puis redevient verte.

Dans la fenêtre « **Output** » → « **Simulator** », vous devez voir les commentaires suivants (surlignés en bleu en figure 18) :

Single breakpoint: @0x9D000130
Simulator halted
Running

Nous avons créé dans notre premier programme une boucle infinie qui inverse l'état de l'une des broches de sortie. Si nous connectons une LED à cette broche, nous verrons la LED clignoter à une certaine fréquence.

Conclusion

Dans cette première leçon, nous avons présenté l'environnement de développement MPLAB X IDE, en énumérant les caractéristiques principales et en décrivant l'interface utilisateur.

De plus, nous avons utilisé l'assistant pour créer notre premier projet et nous avons illustré un premier projet pratique, facile à utiliser.

Dans la prochaine leçon, nous allons entrer dans les détails en étudiant les périphériques de base du microcontrôleur. En particulier, nous étudierons le périphérique « **Timer** » et celui de la gestion des interruptions.

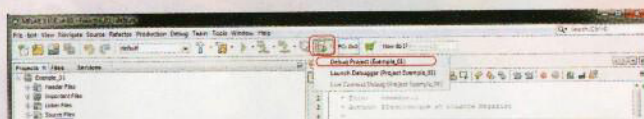


Figure 17 : le bouton « **Debug** » dans la barre d'outils.

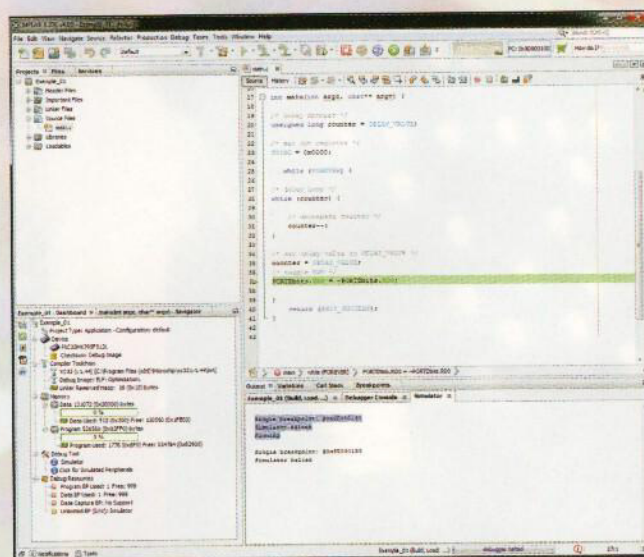


Figure 18 : débogage du programme, en bas dans la fenêtre « **Output** » vous apercevez les commentaires surlignés en bleu.



Le MPLAB® ICD4 est un débogueur/programmeur en circuit, c'est l'outil de débogage et de programmation le plus rapide et le plus économique de Microchip pour les familles des microcontrôleurs PIC, dsPIC et PIC32.

Cette vitesse est atteinte grâce à un microcontrôleur de 32 bits cadencé à 300 MHz avec 2 Mo de RAM et un FPGA haute vitesse pour accélérer les phases de programmation et de débogage. Il s'utilise avec l'interface graphique MPLAB X IDE.

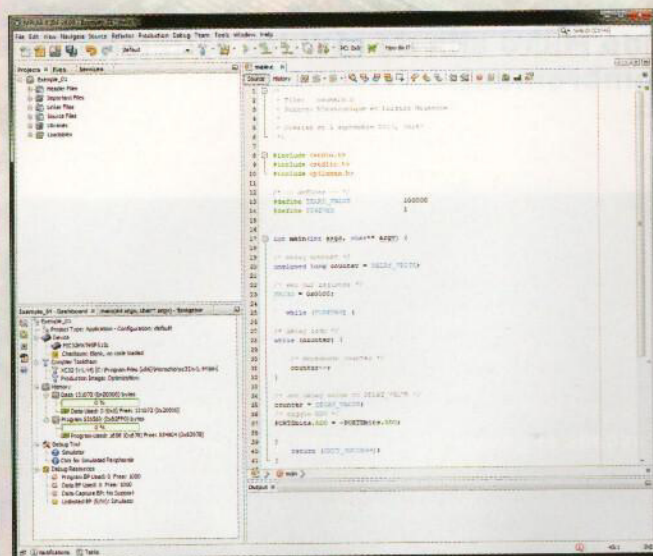


Figure 16a : le code source complet du fichier « **main.c** ».



Étoile de Noël

..... de Boris Landoni

Nous vous proposons dans cet article de réaliser une étoile de Noël innovante, en utilisant la technologie « LED Neopixel ». Cette étoile est contrôlée par une carte Arduino Micro afin de réduire au minimum l'encombrement.

Dès l'arrivée des fêtes de fin d'année, la Rédaction cherche des idées de montages qui allient tradition et innovation, pour se distinguer des gadgets traditionnels décoratifs de Noël disponibles dans le commerce, mais qui sont toujours utiles pour décorer un sapin, une porte, etc.

Pour cela, les LED RVB et les rubans lumineux à LED sont devenus incontournables, nous ne pouvions donc pas nous exempter de vous proposer des décorations classiques lumineuses (car Noël est une fête pleine de lumières, de vie et

d'espoir) remaniées avec les nouvelles technologies d'éclairage à LED Neopixel.

Avec un certain nombre de **LED Neopixel** connectées **en cascade et gérées par une carte Arduino**, nous avons créé une étoile de Noël à 5 branches qui, placée au sommet du sapin, produira une animation du plus bel effet. Examinons le schéma électrique du circuit qui comprend en grande partie des LED et une carte à base d'Arduino Micro, afin de réduire l'encombrement de l'étoile.

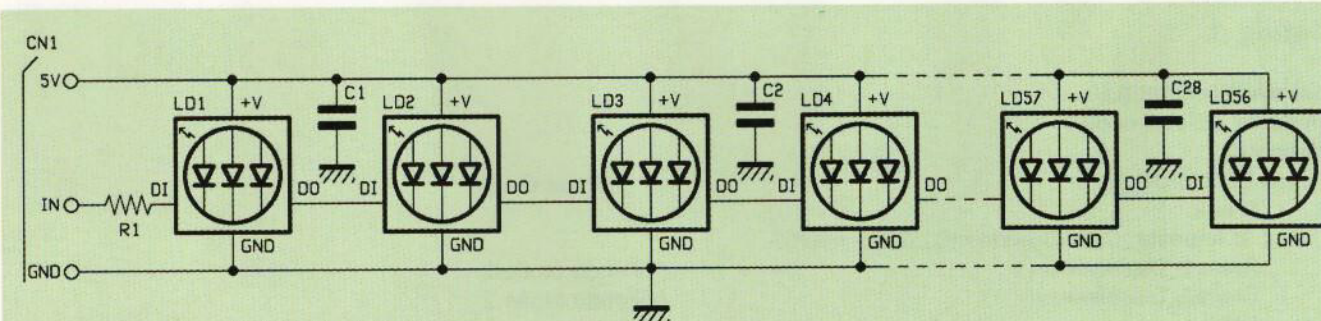


Schéma électrique de l'étoile de Noël.

Schéma électrique

La partie électronique du projet consiste essentiellement à l'utilisation d'un ensemble de **56 LED Neopixel** disposées pour former **deux étoiles concentriques**. Les 35 premières LED RVB constituent la plus grande étoile (vers l'extérieur), tandis que les 20 autres forment la plus petite étoile (vers l'intérieur). La 56^{ème} LED se situe exactement au centre du circuit imprimé, qui forme une étoile à 5 branches.

Les **LED Neopixel** sont connectées en **cascade** mais alimentées en **parallèle**. Cette configuration permet d'**adresser de manière individuelle chaque LED** et de choisir sa couleur individuellement parmi les nuances possibles, et cela parmi les 256 combinaisons possibles pour chaque couleur de base (donc nous avons $256 \times 256 \times 256$ combinaisons !), soit un total de **16,7 millions de couleurs** ou « couleurs vraies » !

Le circuit de l'étoile n'a besoin que de 3 connexions : l'alimentation positive et négative (que nous pouvons tirer de la carte Arduino) et l'entrée des données reliée à la broche numérique 9 (D9). Une alimentation externe de type micro USB de 1 A fera fonctionner l'ensemble.

Nous rappelons que la technologie Neopixel est une solution qui intègre dans chaque LED RVB un driver et leurs LED R, V et B respectives au format CMS. Nous pouvons donc contrôler directement (individuellement) chaque LED. La **communication des données** s'effectue via le bus « **oneWire** », qui est de type **série à un seul canal** alimenté en 5 V. Le bus « oneWire » permet donc de connecter et de faire **dialoguer entre eux des circuits sur un seul fil**.

Les LED Neopixel se connectent en cascade, ce qui signifie que la ligne de données en entrée arrive sur les contrôleurs respectifs et est reportée vers la sortie. La **vitesse de communication** sur le canal de données peut atteindre **800 kbps** et dispose d'une horloge intégrée.

Le contrôle série est indispensable, car sinon pour adresser individuellement toutes des LED qui composent

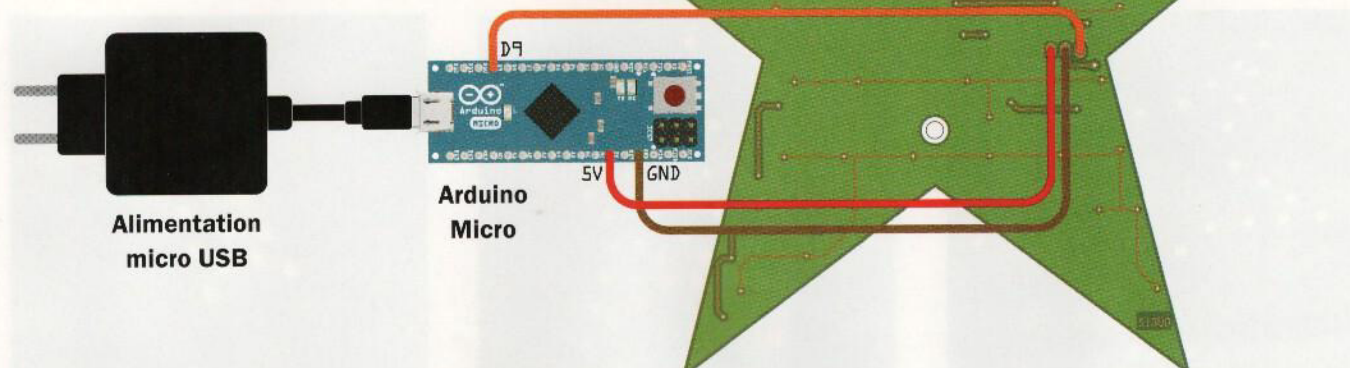
l'étoile, nous devrions utiliser une commande de type matricielle avec multiplexage, ce qui exigerait beaucoup de lignes du microcontrôleur (ici la carte Arduino).

Au lieu de cela, nous n'avons besoin que d'un seul canal de données que nous gérons avec une seule ligne de la carte Arduino, pour plus de simplicité et de clarté. La solution adoptée est l'utilisation d'une petite carte **Arduino Micro**.

Si nous avions dû implémenter un multiplexage matriciel pour les 56 LED, le nombre de lignes aurait imposé l'adoption d'une carte de la famille Arduino comprenant un grand nombre d'entrées/sorties, comme par exemple une carte Arduino Mega.

S'il est vrai que les LED Neopixel peuvent être connectées en cascade de sorte que la ligne de données d'un étage passe au

Le câblage



Listing 1

```

void startShow(int i) {
  Imposta_Colore(modo);
  switch(i){
    case 0: colorOff(); // Tout est éteint
            break;
    case 1: Imposta_Colore(random(0,colore_max)); // Fondu étoile 1
            Stella1_Dissolvenza(); // Fondu étoile 2
            Stella2_Dissolvenza();
            break;
    case 2: Stella1(); // Allumage séquentiel de l'étoile 1 et extinction
            Stella2(); // Allumage séquentiel de l'étoile 2 et extinction
            break;
    case 3: Stella_Alternata(); //Allumage alterné des étoiles
            break;
    case 4: Imposta_Colore(random(0,colore_max));
            for (int ciclo=0; ciclo<3; ciclo++)
            {
              colorCarica(); // Effet de charge
              delay(pausa_effetti*2000);
              colorScarica(); // Effet de décharge
              delay(pausa_effetti*1000);
            }
            break;
    case 5: Stella_Casuale();
            break;
    case 6: Imposta_Colore(random(0,colore_max)); // Effet stroboscopique
            Strobo();
            break;
    case 7: Imposta_Colore(random(0,colore_max)); // Effet de défilement
            barraScorrevole(pausa_sfumatura);
            break;
    case 8: rainbowCycle(12);
            break;
    case 9: colorMix (10);
            break;
  }
}

```

suivant, l'effet collatéral est **qu'au-delà d'un certain nombre de LED, la vitesse de gestion est considérablement réduite.**

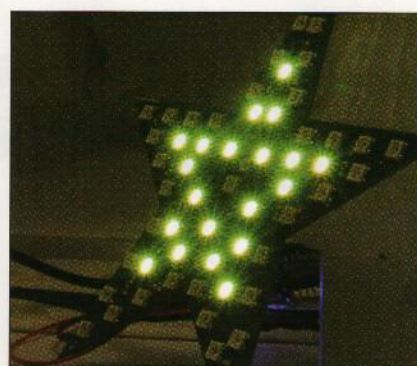
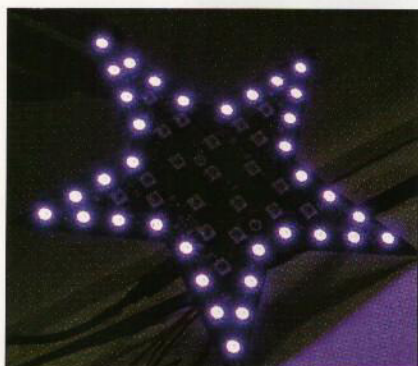
Pour cette raison, vous devez utiliser un adressage matriciel pour obtenir des affichages rapides, il est préférable

dans ce cas d'utiliser plusieurs lignes avec quelques LED connectées sur chacune d'elles.

Cela n'est pas le cas pour notre étoile de Noël, bien que les animations impliquent des variations de la luminosité de chaque LED, **la vitesse des**

différents effets n'atteint pas les limites de débit des données des LED Neopixel.

La définition de tous les effets, prévus par le programme, s'exécutent dans Arduino. Vous pouvez voir le code dans le **listing 1.**



Listing 2

```

void Stella1_Dissolvenza(void)
{
    int pixel = 0;
    int i=0;
    uint32_t R=0;
    uint32_t G=0;
    uint32_t B=0;
    num_pixel=strip.numPixels();
    colorOff();
    for (int ciclo=0; ciclo<7; ciclo++)
    {
        if (R_ON==true) R=0;
        if (G_ON==true) G=0;
        if (B_ON==true) B=0;

        //Incrémente la luminosité

        for(int i=0; i<luce_max; i++)
        {
            if (R_ON==true)
            {
                R=i;
            }
            if (G_ON==true)
            {
                G=i;
            }
            if (B_ON==true)
            {
                B=i;
            }

            for(pixel=0; pixel<num_pixel-21; pixel++)
            {
                strip.setPixelColor(pixel, strip.Color(R, G, B));
            }
            strip.show();
            delay(pausa_sfumatura/2);
        }

        // Décrémente la luminosité

        for(int i=luce_max; i>0; i--)
        {
            if (R_ON==true)
            {
                R=i;
            }
            if (G_ON==true)
            {
                G=i;
            }
            if (B_ON==true)
            {
                B=i;
            }

            for(pixel=0; pixel<num_pixel-21; pixel++)
            {
                strip.setPixelColor(pixel, strip.Color(R, G, B));
            }
            strip.show();
            delay(pausa_sfumatura/2);
        }
    }
}

```

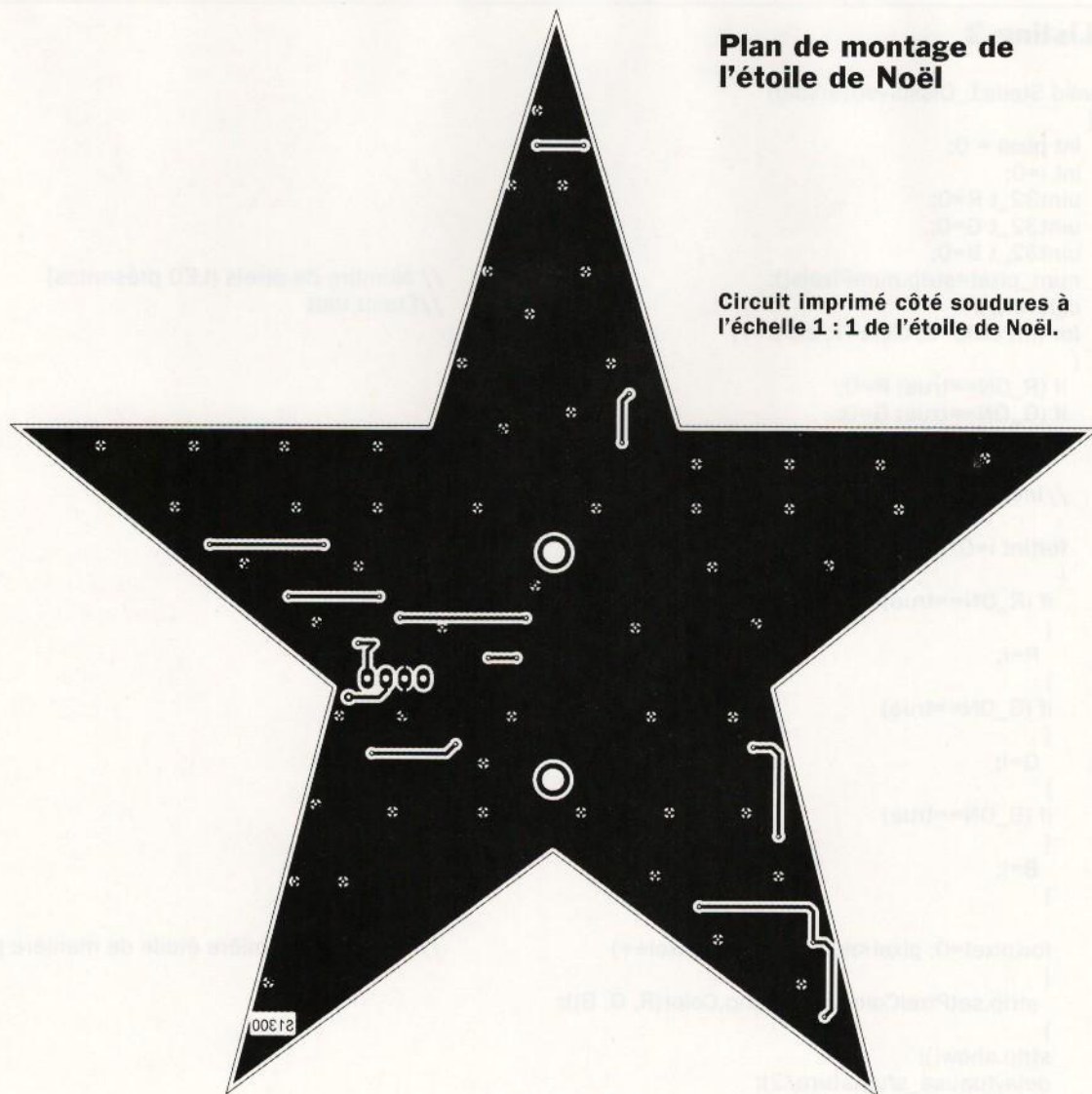
// Nombre de pixels (LED présentes)
//Éteint tout

// Allume la première étoile de manière progressive

// Met à jour (rafraîchit) l'état de la première étoile

Plan de montage de l'étoile de Noël

Circuit imprimé côté soudures à l'échelle 1 : 1 de l'étoile de Noël.



Liste des composants de l'étoile de Noël

C1 100 nF céramique (0805)
C2 100 nF céramique (0805)
C3 100 nF céramique (0805)
C4 100 nF céramique (0805)
C5 100 nF céramique (0805)
C6 100 nF céramique (0805)
C7 100 nF céramique (0805)
C8 100 nF céramique (0805)
C9 100 nF céramique (0805)
C10 100 nF céramique (0805)

C11 100 nF céramique (0805)
C12 100 nF céramique (0805)
C13 100 nF céramique (0805)
C14 100 nF céramique (0805)
C15 100 nF céramique (0805)
C16 100 nF céramique (0805)
C17 100 nF céramique (0805)
C18 100 nF céramique (0805)
C19 100 nF céramique (0805)
C20 100 nF céramique (0805)
C21 100 nF céramique (0805)
C22 100 nF céramique (0805)
C23 100 nF céramique (0805)
C24 100 nF céramique (0805)

C25 100 nF céramique (0805)
C26 100 nF céramique (0805)
C27 100 nF céramique (0805)
C28 100 nF céramique (0805)
R1 470 Ω (0805)
LD1 à LD56 LED Neopixel
WS2812B (5050)

Divers

Connecteur 4 pôles (fil + embout femelle)
Barrette mâle 4 pôles coudée à 90°

Si vous souhaitez apporter des modifications au programme pour obtenir des effets différents (le listing 2 montre un exemple de modification qui produit une disparition des deux étoiles ...), rappelez-vous que la fréquence de rafraîchissement et donc la vitesse d'allumage/extinction de chaque LED,

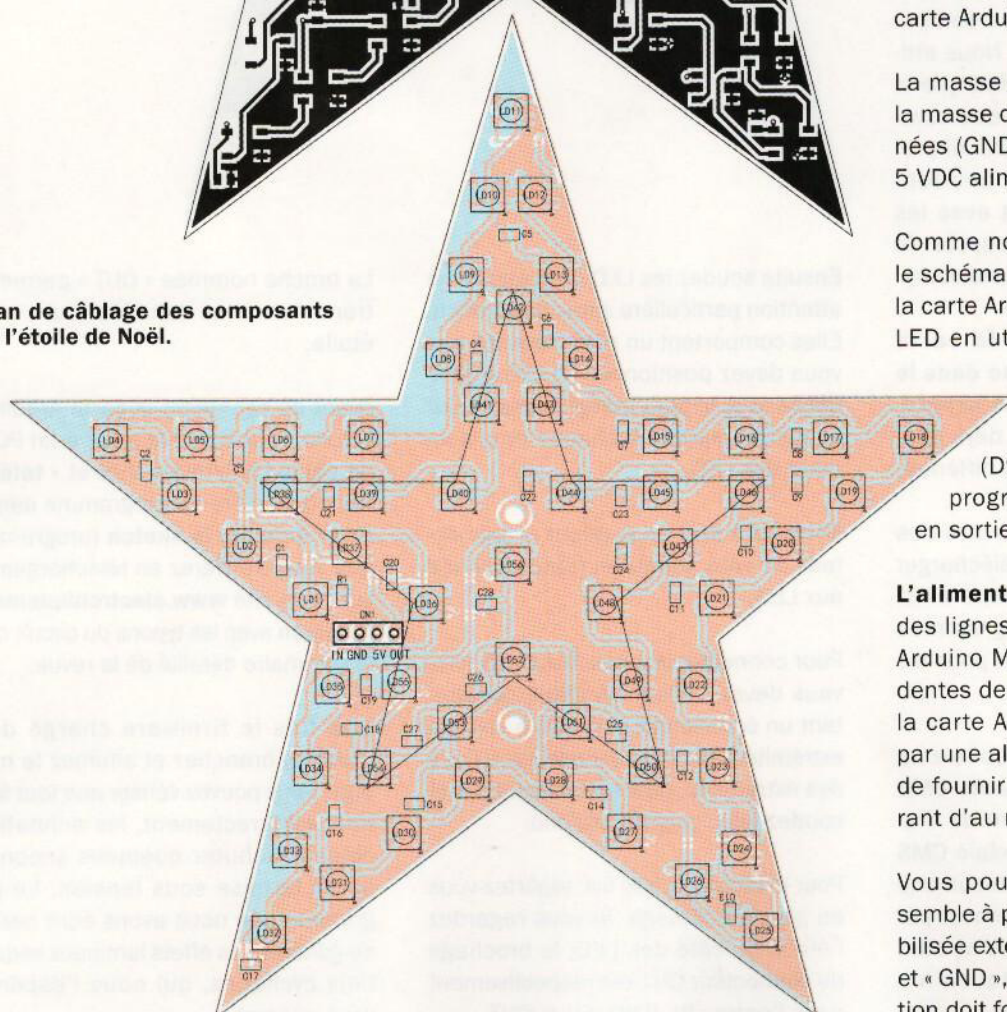
est inversement proportionnelle au nombre de LED à gérer. Pour chaque groupe de LED, vous pouvez définir la fréquence de rafraîchissement à volonté afin de rendre certains effets de lumière imperceptibles. Dans notre cas, la **fréquence de balayage des LED est de 400 Hz**.

La vitesse maximale de transfert des données conditionne la fréquence de rafraîchissement pour un nombre de LED donné. Cela signifie que **plus le nombre de LED à gérer est grand, plus la fréquence de rafraîchissement est petite et donc la vitesse de déplacement de l'effet représenté**.

Circuit imprimé côté composants
à l'échelle 1 : 1 de l'étoile de Noël.



Plan de câblage des composants
de l'étoile de Noël.



Le **protocole de commande** du système Neopixel prévoit l'envoi de manière cyclique (à une fréquence qui dépend de la fréquence de rafraîchissement désirée) de **chaînes de données** composées de **24 bits divisées en 3 groupes d'octets** (donc 1 chaîne = 3×8 bits).

Chacun des octets d'une chaîne de 24 bits contient l'état d'allumage de chaque couleur de base dans cet ordre : d'abord les 8 bits bleus, puis 8 bits rouges et enfin 8 bits verts.

Le schéma électrique montre la connexion en cascade des LED Neopixel. La ligne de données entre dans la broche « DI » de la 1^{ère} LED et ressort par la broche « DO » pour entrer de nouveau dans la broche « DI » de la 2^{ème} LED, et ainsi de suite.

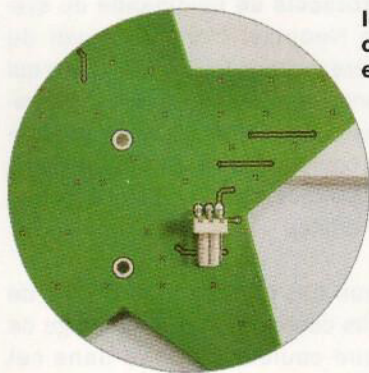
La broche de la 1^{ère} LED est reliée au bus de données à travers la résistance R1 au contact « IN », ce dernier étant connecté à la ligne de données de la carte Arduino.

La masse de l'alimentation est aussi la masse de référence du bus de données (GND sur le circuit imprimé). Le 5 VDC alimentant le montage.

Comme nous l'avons mentionné, sur le schéma électrique nous voyons que la carte Arduino pilote l'ensemble des LED en utilisant une seule ligne de données. Celle-ci, nommée « IN », est reliée à la **broche numérique 9** (D9) d'Arduino, donc dans le programme elle est configurée en sortie.

L'alimentation de l'étoile provient des lignes **5 VDC** et **GND** de la carte Arduino Micro. Pour des raisons évidentes de consommation de courant, la carte Arduino doit être alimentée par une alimentation externe capable de fournir 5 V stabilisés avec un courant d'au moins 800 mA.

Vous pouvez aussi alimenter l'ensemble à partir d'une alimentation stabilisée externe entre les broches « Vin » et « GND », mais dans ce cas l'alimentation doit fournir au moins 7 VDC.



Ici, côté soudures, vous pouvez voir le connecteur acheminant l'alimentation et le bus de données.



Réalisation pratique

Maintenant, construisons notre étoile. Elle nécessite un circuit imprimé double face, la plupart des composants sont au format CMS. Nous attirons votre attention sur le fait que la fabrication et le soudage des composants nécessitent du matériel professionnel. Les néophytes pourront se procurer le kit complet avec les composants déjà soudés auprès de la société Comelec (www.comelec.fr).

D'autre part, **notez que la carte Arduino n'est pas comprise dans le kit**, elle devra être acquise séparément sauf si vous en avez déjà une, même s'il s'agit d'un modèle différent.

Pour les plus courageux d'entre nos lecteurs, commencez par télécharger les typons du circuit imprimé ainsi que les exemples de programmes incluant la librairie Adafruit pour les LED Neopixel.

Une fois le circuit fabriqué, utilisez un fer à souder pour composants CMS avec une pointe de 0,2 mm de diamètre et de la soudure spéciale CMS (0,5 mm de Ø), ou encore mieux une station à souder CMS. Commencez par souder les condensateurs, ils ne sont pas polarisés donc l'orientation n'a pas d'importance.

Ensuite soudez les LED en prêtant une attention particulière à leur orientation. Elles comportent un côté biseauté que vous devez positionner en correspondance avec le petit triangle dessiné sur le circuit imprimé à chaque emplacement des LED.

Enfin, terminez en soudant le connecteur du côté soudures (face opposée aux LED).

Pour connecter la carte Arduino Micro, vous devez utiliser un câble comportant un connecteur (4 pôles) à chaque extrémité. Coupez le connecteur à l'une des extrémités, puis dénudez les fils et soudez-les à la carte Arduino.

Pour le brochage des fils, reportez-vous au plan de câblage. Si vous regardez l'étoile du côté des LED, le brochage du connecteur CN1 est respectivement dans l'ordre : IN, GND, 5V et OUT.

La broche nommée « OUT » permet de transmettre les données à une autre étoile.

Avant de connecter la carte Arduino à l'étoile, vous devez la relier à un PC via un câble USB/micro USB et « téléverser » (transférer le programme dans la carte Arduino) le **sketch** (programme) que vous trouverez en téléchargement sur notre site www.electroniquemagazine.com avec les typons du circuit dans le sommaire détaillé de la revue.

Une fois le firmware chargé dans Arduino, branchez et allumez le montage. Vous pouvez vérifier que tout fonctionne correctement, les animations doivent débiter quelques secondes après la mise sous tension. Le programme que nous avons écrit permet de générer des effets lumineux séquentiels cycliques, qui nous l'espérons vous raviront.

Une boule de Noël à changement de couleur



Voici une boule de Noël qui ornara votre sapin d'une manière originale. En effet, cette boule a la particularité de changer de couleur en produisant des effets lumineux lors du passage d'une couleur à l'autre, de plus elle ne nécessite pas d'unité de contrôle externe.

de Boris Landoni



Les boules utilisées pour décorer l'arbre de Noël existent en plusieurs modèles passifs ou lumineux. Pour ces derniers, elles sont généralement éclairées de l'intérieur au moyen de petites lampes incandescentes de diffé-

rentes couleurs ou par des LED, qui ont remplacé les ampoules traditionnelles.

Ces boules peuvent fonctionner de différentes façons, en fonction de leur puissance, avec un éclairage classique clignotant lentement, ou avec un éclairage fixe et donc constamment allumées. Pour ces dernières, il est alors possible de faire varier leur luminosité en les pilotant avec des unités de contrôle externes.

Dans ces pages, nous vous proposons quelque chose de différent, qui va au-delà de ce que vous pouvez trouver dans le commerce.

Il s'agit d'une boule de Noël à LED qui produit ses propres effets lumineux. Elle est donc autonome. C'est un concentré de technologie utilisant une **LED RVB spéciale qui intègre un microcontrôleur** et donc qui peut générer des effets lumineux tels que la dissolution des couleurs.

Mais procédons par ordre en analysant le schéma électrique qui est très simple, il comporte que très peu de composants.

Schéma électrique

Le circuit, comme vous pouvez le voir sur le schéma électrique ci-contre, se compose simplement d'une LED intelligente à 3 couleurs et d'une section d'alimentation automatique.

Lorsque la luminosité ambiante est suffisamment basse, la boule s'illumine en produisant des effets lumineux.

Non seulement la LED s'allume en prenant une variété de couleurs et elle le fait toute seule, mais de plus la boule décorative dispose d'une pile interne (donc pas de câblage externe à effectuer) et s'allume que lorsque cela est nécessaire afin d'économiser la pile bouton de type CR2032.

Donc, nous n'avons besoin pas d'alimentation externe, ni de fil gênant qui envelopperait le sapin de Noël.

L'allumage de la LED est contrôlé par un capteur de lumière. Cela permet de consommer l'énergie de la batterie uniquement lorsque cela est nécessaire, afin d'ouvrir le moins possible la boule constituée par deux hémisphères encastrés l'un dans l'autre. Cela permet une autonomie suffisante pour toute la durée des fêtes de fin d'année.

Commençons par le composant le plus intéressant, qui est la LED intelligente, c'est-à-dire une LED changeant de couleur et dont le diamètre est de 5 mm. Elle se présente sous la forme classique d'une LED dotée d'un boîtier transparent, il n'a que deux broches (anode et cathode) pour l'alimenter.

À l'intérieur du boîtier transparent, se trouve une puce bien visible en forme de parallépipède noir dotée de 4 fils qui pilotent une vraie LED RVB.

En regardant le composant par-dessus, vous pouvez voir avec une loupe à travers la résine transparente, les 4 conducteurs très fins en aluminium qui partent de plusieurs endroits de la puce de contrôle pour atteindre les broches internes de la LED RVB.

Cette dernière est constituée de 3 jonctions PN, chacune d'une couleur différente.

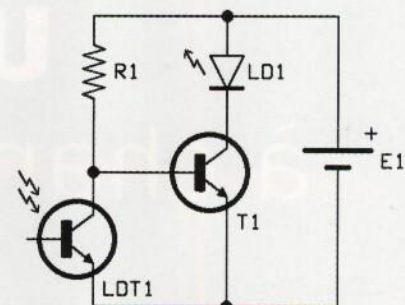


Schéma électrique de la boule décorative.

Les longueurs d'ondes sont de 625 nm pour le rouge, 520 nm pour le vert et 470 nm pour le bleu.

Les intensités lumineuses des 3 jonctions, pour un courant continu $I_f = 20$ mA, s'élèvent à 1600 mcd pour le rouge, 2800 mcd pour le vert et 1300 mcd pour le bleu.

NB : l'unité « mcd » veut dire « milli-candela », c'est une unité de mesure de l'intensité lumineuse. Dans le système international (SI), $1 \text{ mcd} = 10^{-3} \text{ candela}$.

Lorsque la LED est alimentée, elle change automatiquement de couleur passant du rouge au lilas, puis au violet, au bleu, au vert, au jaune, puis revient au rouge. Le cycle redémarre ainsi.

La tension d'alimentation de fonctionnement du composant est comprise dans une plage allant de 3 V à un maximum de 4,5 V.



Le circuit a été conçu pour être inséré dans une sphère en plastique transparente ou légèrement colorée, mais ayant une bonne transmittance ou facteur de transmission. Aucune fixation n'est requise, car le circuit se positionne automatiquement à cause de la gravitation.



Kyocera allume la LED

Le géant des imprimantes et des photocopieurs se lance dans le monde de l'éclairage à LED en développant des dispositifs d'éclairage en lumière blanche dérivée de la lumière violette, qui est convertie par des luminophores. Il s'agit de la technologie « VP3 » (violet emission three phosphor).

Cette technologie assure un rendu parfait des couleurs et de la blancheur. Elle utilise toutes les couleurs de l'arc-en-ciel, notamment les émissions de rouge foncé, la technologie « VP3 » permet un rendu magnifique et précis des tons chauds, atteignant un indice de rendu des couleurs (IRC) de 95 et un rendu du rouge foncé (R9) de 95, contrairement aux LED blanches basées sur le bleu et sans émissions de violet.

Les nouvelles LED de Kyocera utilisent cette technologie basée sur une jonction émettant une lumière violette, qui est ensuite convertie par trois types de luminophores pour produire tout le spectre de la lumière et même le blanc. Ces LED se distinguent des solutions existantes, car elles ne contiennent pas de substrat à base de nitrure de gallium (GaN).

Comme nous l'avons mentionné précédemment, l'allumage de la LED intelligente est commandé par un interrupteur crépusculaire statique, formé par le transistor T1 et le phototransistor LDT1. Ce dernier est un « L-7113P3C » encapsulé dans un boîtier rond de 3 mm de diamètre.

Il possède 2 broches reliées à son collecteur et à son émetteur. La base capte la lumière et en fonction de la quantité de lumière reçue, le phototransistor devient alors plus ou moins conducteur.

La base du phototransistor est exposée à la lumière ambiante qui traverse le boîtier transparent. Dans des conditions de **lumière sombre** ou de **faible intensité lumineuse**, le **phototransistor LDT1 est bloqué**. Aucun courant ne circule entre le collecteur et l'émetteur.

Ainsi, la tension présente sur la **base du BC547B** est **déterminée par le courant circulant** dans la résistance **R1**, qui a une valeur suffisante pour amener ce transistor en **saturation**.

Dans cette condition, le **potentiel du collecteur** du transistor BC547B

devient identique à celui de son émetteur, c'est-à-dire qu'il est **proche de 0 V** (masse) et donc un **courant de collecteur circule dans le transistor provoquant ainsi allumage** de la LED intelligente **LD1**.

Celle-ci se met à changer de couleur en commençant son cycle d'animations prédéfinies dans la puce de contrôle. Les animations consistent essentiellement en la variation de la couleur obtenue par la dissolution des couleurs de base rouge, verte et bleue.

Pour être exact, initialement les trois couleurs sont allumées une à la fois, puis elles sont dissolues.

Ensuite, des couples de couleurs s'allument (rouge/vert, rouge/bleu, vert/bleu) formant ainsi des combinaisons de dissolutions de couleurs. Le cycle se répète tant que la LED est alimentée.

Lorsque la lumière ambiante augmente (par exemple dans la journée), la jonction « base-collecteur » du phototransistor libère des électrons, qui deviennent disponibles pour la conduction, ce qui donne naissance à un courant de base dans le phototransistor LDT1.

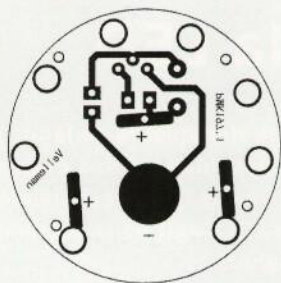
Ainsi, en raison du champ électrique « collecteur-base » de LDT1, les électrons libérés se mettent à circuler dans la jonction du collecteur, générant un courant de collecteur qui permet d'abaisser la tension V_{CE} (du phototransistor) à quelques centaines de millivolts (proche de la masse) en raison de la chute dans la résistance R1.

La **base de T1** se trouve alors à un **potentiel proche de 0 V** et donc le transistor **T1 est bloqué**. **Aucun courant circule dans le collecteur de T1** et par conséquent **la LED s'éteint** (elle n'est plus alimentée). Vous voyez donc que la boule pour le sapin de Noël s'allume et effectue des animations uniquement lorsqu'il y a peu de lumière dans la pièce, c'est-à-dire à partir du début de soirée.

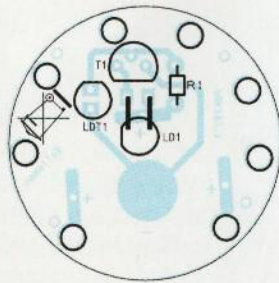
Réalisation pratique

Nous avons étudié le fonctionnement du circuit, nous allons maintenant passer à la construction. Nous rappelons à ceux qui veulent construire rapidement la boule décorative avant la Noël, qu'elle est disponible sous forme de kit à monter. Tous les composants ainsi que le circuit imprimé sont compris dans le kit.

Plan de montage



Circuit imprimé de la boule décorative à l'échelle 1 : 1 côté soudures.

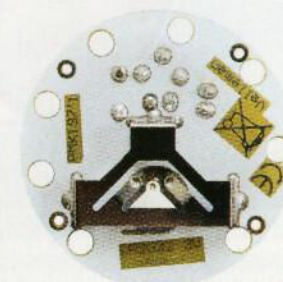


Plan de câblage des composants de la boule décorative.

Liste des composants

R1..... 100 kΩ
T1 BC547
LD1.... LED RVB intelligente Ø 5 mm

référence LOR5RGBC-MF-RHS
LDT1.. phototransistor L-7113P3C
E1 support de pile pour circuit imprimé type CR2032



Par contre, pour ceux qui veulent fabriquer une ou plusieurs boules pour décorer leur maison, ils doivent dans un premier temps réaliser le circuit imprimé qui comporte une seule face.

L'ensemble doit être intégré dans une sphère en plastique transparente, cela peut se trouver chez les distributeurs d'éléments décoratifs.

Une fois le ou les circuit(s) gravé(s) et percé(s), commencez par souder la résistance puis le support de pile CR2032 côté soudures.

Enfin terminez en soudant les composants actifs en prêtant une attention particulière à leur orientation.

NB : le circuit imprimé est circulaire, son diamètre est de 38 mm.

Rappelez-vous que l'anode (positif) de la LED est la patte la plus longue (le négatif correspond au côté biseauté du boîtier). Pour le phototransistor, la patte la plus longue correspond à

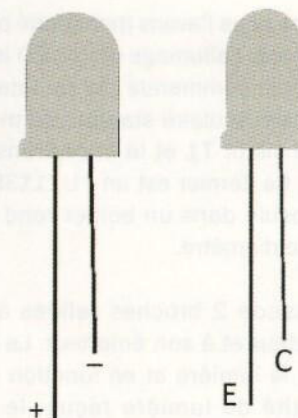
l'émetteur (le collecteur correspond au côté biseauté).

Pour le transistor BC547, en le regardant de face (côté plat où se situe l'inscription), les broches correspondent respectivement de gauche à droite au : collecteur, base, émetteur.

Une fois les soudures terminées, vous pouvez insérer la pile CR2032 dans le support en orientant la borne positive vers le haut (c'est-à-dire du côté opposé au PCB).

Ensuite, insérez le circuit dans une sphère transparente ou même colorée, mais avec une certaine transparence, sinon le phototransistor « comprendra » qu'il se trouve dans une ambiance lumineuse sombre et le montage fonctionnera en continu avec une autonomie très réduite.

Fabriquez autant de boules que vous désirez et décorez votre sapin de Noël, ou votre crèche ou encore l'intérieur de votre maison selon votre goût.



À gauche, le brochage de la LED LOR5RGBC-MF-RHS et à droite celui du phototransistor L-7113P3C.

NB : le kit complet ainsi que la LED intelligente (à l'unité) sont disponibles chez Comelec (www.comelec.fr).

Concernant la LED, elle dispose d'un microcontrôleur intégré, attention, vous ne pouvez pas la remplacer par une simple LED RVB disponible chez d'autres fournisseurs.