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

ELECTRONIQUE

ELECTRONIQUE

ET LOISIRS

magazine

www.electroniquemagazine.com

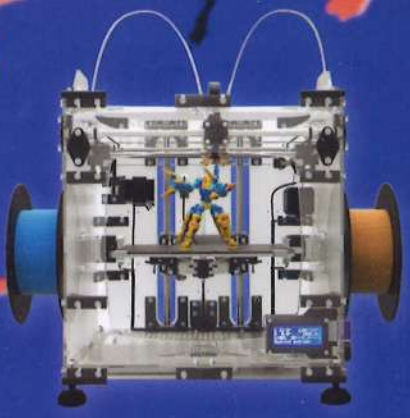
n°136 Automne 2016

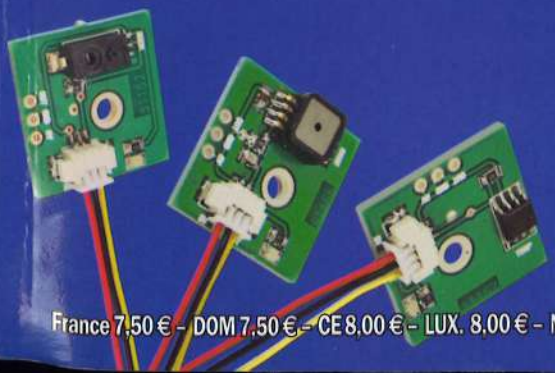
DEMOBOARD PIC18FXX

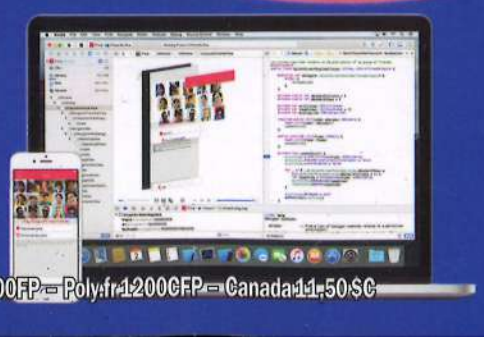
Télécommande UHF longue portée

- Cours Arduino VI
- Imprimante 3D bicolore - II
- Emetteur FM avec RaspberryPi
- Programmez sous iPhone, iPad - II

Breakout Board multifonctions







N° 136 Septembre 2016

M 04662 - 136 - F: 8,30 € - RD

France 7,50 € - DOM 7,50 € - CE 8,00 € - LUX. 8,00 € - NC (S)900FP - Poly.fr 1200CFP - Canada 11,50 $C

Sommaire

ARTICLES

Numéro 136
Automne 2016
100 pages

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

L'EDITO AUTOMNE 2016

Chères lectrices, chers lecteurs,

C'est la rentrée, hélas les vacances sont bien finies ! Pour ceux d'entre vous qui veulent se remettre rapidement dans le bain, nous décrivons une application dérivée de l'article consacré à la transformation du RaspberryPi en récepteur radio. Il s'agit ici de le transformer en émetteur FM. D'autre part, toujours dans les hautes fréquences, nous vous proposons un système émetteur/récepteur UHF longue portée (plusieurs kilomètres avec les antennes appropriées).

Les adeptes de l'impression 3D ne manqueront pas de travail avec la première partie du montage mécanique de la 3DVERTEX. La suite sera abordée dans le prochain numéro 137.

Nous continuons aussi l'étude des cartes de prototypage BREAKOUT BOARD avec 3 nouvelles cartes, les premières ont suscité un vif intérêt de la part des lecteurs.

Enfin de nombreux lecteurs nous ont interrogé sur la mise en œuvre des microcontrôleurs PIC 18FXX, nous proposons à partir de ce numéro une platine de développement pour PIC 18FXX avec quelques cartes additionnelles pratiques. Nous mettrons aussi l'accent sur la programmation dans les prochains numéros.

La Rédaction

L'énergie renouvelable en page 44

à l'adresse **www.electroniquemagazine.com** dans le sommaire de la revue 136 et à l'onglet « Télécharger »

La lampe renaît dans le transistor à vide

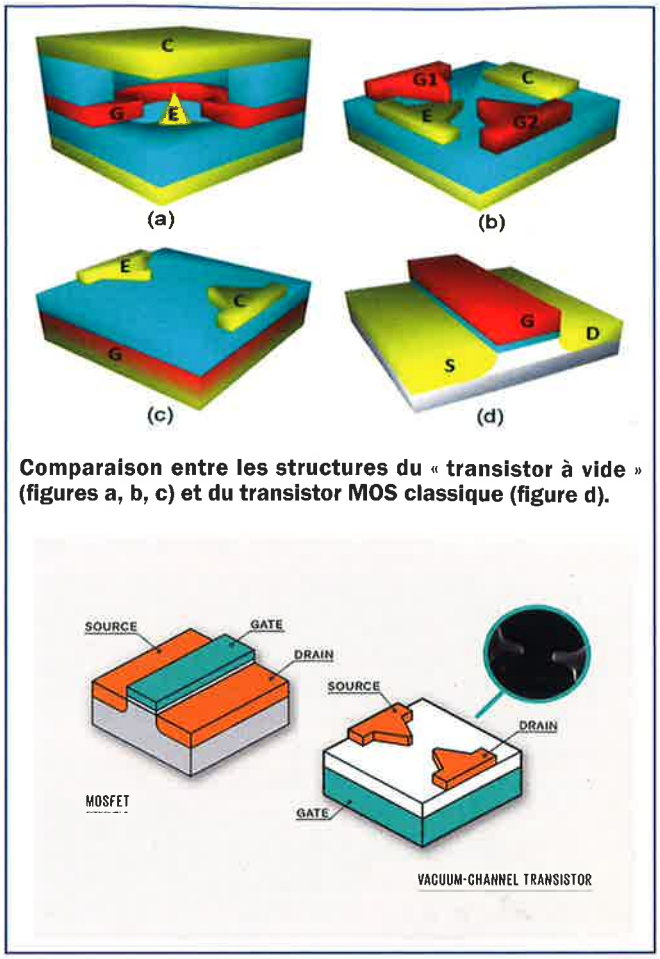
Les semi-conducteurs actuels semblent atteindre leurs limites en termes de fréquence de fonctionnement. C'est pourquoi les chercheurs testent constamment de nouveaux matériaux et des architectures innovantes pour le fonctionnement en commutation. Actuellement les composants sont constitués de matériaux à 2 dimensions, car ils contiennent une seule couche de molécules. Récemment des chercheurs ont mis au point des composants « à vide » qui fonctionnent comme des lampes, mais qui sont en fait des semi-conducteurs.

Des chercheurs de la NASA ont conçu et construit dans leur laboratoire une sorte de tube à vide qui ressemble conceptuellement à une lampe d'autrefois. Plus précisément, ils l'ont appelé « transistor à vide », car les ingénieurs de la NASA ont imaginé fabriquer un transistor FET dont la grille est faite d'hélium (He). Normalement le transistor devrait être contenu dans le vide absolu (difficile à obtenir) comme dans le cas des lampes. Cependant l'hélium est plus facile à manipuler, avec des résultats presque identiques. En d'autres termes, il s'agit d'une lampe fabriquée avec les méthodes modernes de production d'un transistor FET.

Pour expliquer à quoi ressemble et comment fonctionne ce composant, nous devons reprendre les fondements des traditionnels tubes à vide amplificateurs. La lampe la plus simple est la triode (son comportement est similaire à celui d'un transistor FET), elle dispose d'une grille de commande qui régule le flux de courant (le flux d'électrons) circulant entre la cathode émettrice d'électrons et l'anode réceptrice d'électrons. L'anode est maintenue à une polarité positive par rapport à la cathode, et dans des conditions normales en appliquant une certaine tension sur la cathode convenablement chauffée par un filament, des électrons sont libérés et « aspirés » par l'anode. En polarisant la grille négativement par rapport à la cathode, une partie des électrons est rejetée et donc le courant d'anode diminue jusqu'à l'interruption totale obtenue lorsque la tension négative atteint la valeur du pincement (pinch-off).

Un transistor FET à canal N fonctionne de manière identique. Les charges électriques (électrons) circulant entre la source S et le drain D sont contrôlées par la tension entre la grille G (Gate) et la source S. Dans les « transistors à vide » conçus par la NASA, le semi-conducteur est constitué d'une zone émettrice d'électrons (Source) polarisée négativement, et une autre zone (Drain) polarisée positivement par rapport à la Source. Pour faciliter le déplacement des électrons, la zone émettrice est en forme de pointe. Ainsi un faible champ électrique permet de déclencher un flux d'électrons. La grille est constituée par une électrode qui entoure le canal entre la « Source » et le « Drain ». La variation de la tension entre la « Gate » et la « Source » permet de rétrécir le canal et donc de faire varier le courant.

Mais pourquoi cette architecture ? C'est très simple, comme le « transistor à vide » est baigné dans de l'hélium, ce dernier permet un déplacement plus rapidement des électrons.



Comparaison entre les structures du « transistor à vide » (figures a, b, c) et du transistor MOS classique (figure d).

De plus, contrairement à ce qui se passe dans les transistors FET et bipolaires, la commutation entre un état passant et un état bloqué (ON/OFF) est beaucoup plus rapide. Dans les transistors classiques, la limitation de la vitesse de commutation est due à la recombinaison des charges dans la base, ce qui en fonctionnement commuté réduit la vitesse de passage de l'état bloqué à l'état saturé (passant), même en utilisant des diodes Schottky pour éviter un abaissement excessif de la tension entre le collecteur et l'émetteur à saturation.

Dans le cas des transistors FET, le problème provient des capacités parasites. Celles-ci deviennent particulièrement gênantes dans les transistors MOSFET, car elles ont tendance à laisser le transistor à l'état passant pendant un certain temps même après la disparition de la tension de polarisation de la grille.

Selon les chercheurs de la NASA, ces nouveaux transistors permettent d'obtenir une fréquence de transition (fréquence à partir de laquelle le gain « hfe » d'un transistor utilisé en amplificateur linéaire est égal à 1) de 460 GHz. Cela représente 100 fois la fréquence maximale des processeurs les plus rapides actuellement.

Pour le moment, le nouveau « transistor à vide » est à l'état de prototype dans le laboratoire, car avant de pouvoir être fabriqué à l'échelle industrielle il est nécessaire de surmonter les problèmes liés aux procédés de fabrication.

L'USB « Type-C »

Faisons un point sur l'évolution du bus USB. C'est une norme de communication série (USB signifie Universal Serial BUS) qui permet la connexion et l'envoi de données entre un PC et des périphériques.

Il est important de souligner la différence entre les spécificités et les connecteurs. Il y a souvent beaucoup de confusion entre ces deux aspects.

La norme « 1.0 » de janvier 1996 prend en charge les connexions à une vitesse maximale de 1,5 Mbit/s, vitesse appropriée pour les souris, claviers et autres dispositifs similaires. La version 1.1 (septembre 1998) ajoute le mode « full speed » qui augmente la vitesse jusqu'à 12 Mbit/s. Ce mode permettait d'effectuer des transferts de données dans de bonnes conditions entre un PC et des périphériques de stockage tels que des clés USB ou un disque dur externe.

Avec la norme USB 2.0 (avril 2000), la vitesse de transfert maximale atteint 480 Mbit/s, ce qui était similaire aux normes séries les plus rapides comme l'IEEE 1394, principalement utilisé pour les signaux vidéo.

Dans les années suivantes, une série de nouvelles fonctionnalités ont été introduites, par exemple la prise en charge de la recharge de périphériques, l'USB OTG (USB On The Go) qui permet à deux périphériques USB de communiquer entre eux sans la nécessité d'un hôte (PC). Par exemple, envoyer directement les photos d'un appareil photo à une imprimante ou un disque dur.

En 2008, apparaît la norme USB 3.0 qui peut transférer des données à une vitesse de 4,8 Gbit/s (soit 10 fois plus que l'USB 2.0).

Les premiers appareils font leur apparition sur le marché deux ans plus tard. La version 3.0 garde la rétrocompatibilité avec les normes précédentes c'est-à-dire avec l'USB 2.0 et l'USB 1.0.

Tableau 1 : brochage des connecteurs standard USB 1.1 - 2.0.

Pin	Nom du signal	Couleur du fil	Description
1	VBUS	rouge	+ 5 V
2	D-	blanc	data -
3	D+	vert	data +
4	GND	noir	GND

Tableau 2 : brochage des connecteurs micro/mini USB 1.1 - 2.0.

Pin	Nom du signal	Couleur du fil	Description
1	VBUS	rouge	+ 5 V
2	D-	blanc	data -
3	D+	vert	data +
4	ID	fuchsia	interconnection Mini/Micro A et B • Type A connecté à GND • Type B non connecté • OTG connecté à GND
5	GND	noir	GND

Tableau 3 : broches supplémentaires introduites avec le connecteur USB 3.0

Pin	Nom du signal	Description
6	SSTX+	Transfert des données de l'hôte au périphérique.
7	SSTX-	retour SSTX+
8	GND	GND
9	SSRX+	Transfert des données du périphérique à l'hôte.
10	SSRX-	retour SSRX+

Tableau 4 : dernières broches ajoutées avec le connecteur USB 3.0

Pin	Nom du signal	Description
11	VBUS	+ 12 V
12	VBUS	+ 20 V

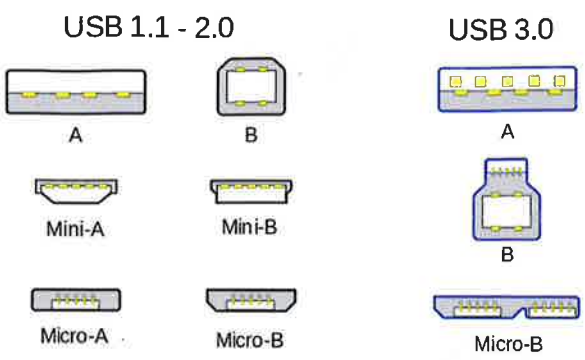
Au mois de juillet 2013, la version USB 3.1 apparaît avec un nouveau connecteur nommé « Type-C ». Cette norme atteint une vitesse maximale de transmission de 10 Gb/s.

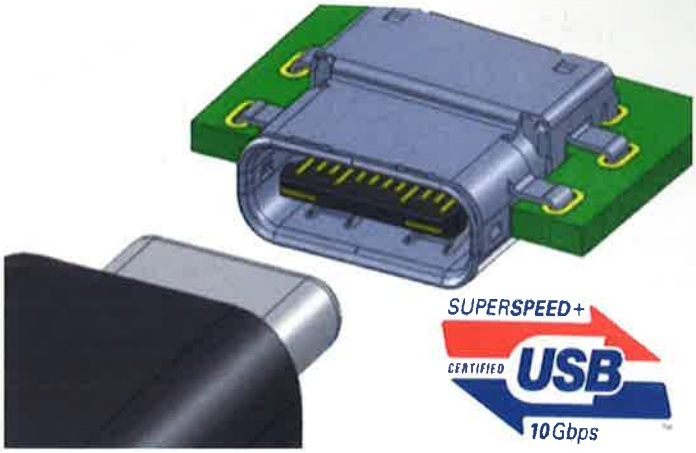
En ce qui concerne la recharge, ce protocole est en mesure de fournir 5 niveaux différents, uniquement avec le nouveau connecteur de « Type C » :

1. 5 V/2 A max (10W) ;
2. 5 V/2A max (10W) et 12 V/1,5 A max (18W) ;
3. 5 V/2A max (10W) et 12 V/3 A max (36W) ;
4. 5 V/2A max (10W), 12 V et 20 V/3 A max (36W, 60W) ;
5. 5 V/2A max (10W), 12 V et 20 V/5 A max (60W, 100W).

En ce qui concerne les différents types de connecteurs, les spécifications de la norme USB prévoyaient deux types de connecteurs pour relier les périphériques. Le connecteur A et le connecteur B. Plus tard, des variantes ont été introduites pour miniaturiser les connecteurs. C'est ainsi que sont nés les connecteurs « mini-USB » et « micro-USB », toujours de type A et de type B.

Avec les dernières spécificités de l'USB 3.1, le nouveau connecteur « Type C » a été introduit. Il est très différent des connecteurs précédents, de plus il est réversible c'est-à-dire qu'il n'a plus de sens haut/bas. Les différents niveaux de prise en charge de l'alimentation ont conduit à l'ajout de deux nouvelles broches « VBUS » (12 V et 20 V/5A), ce qui





Les différents types de connecteurs des standards USB 1.1 - 2.0 et USB 3.0.

représente un véritable avantage de cette norme par rapport aux précédentes.

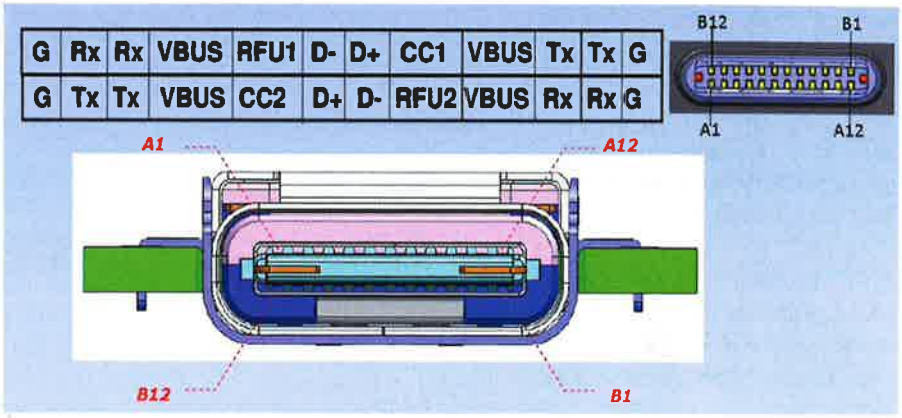
Dans les nouveaux appareils portables, le connecteur de « Type C », qui sert exclusivement à l'alimentation, est de couleur noire. Par contre celui des données de « Type C » (compatible A) est de couleur bleue.

Le câble de « Type C » est en passe de devenir le câble « universel », car il est capable de fournir un taux de transfert allant jusqu'à 10 Gb/s, une capacité d'alimentation de 100 W et la possibilité de transmettre un flux vidéo en continu avec un débit très élevé, le tout dans un seul câble.

Grâce à la petite taille du connecteur USB « Type C » (il est plus petit que la version « micro-USB »), il est possible de fabriquer des ordinateurs et des périphériques plus petits, plus minces et plus légers.

Dans l'avenir, nous pourrons recharger n'importe quel smartphone, tablette et ordinateur portable avec un seul câble qui sera également adapté pour transférer des données à des vitesses élevées.

Câble USB 3.1.

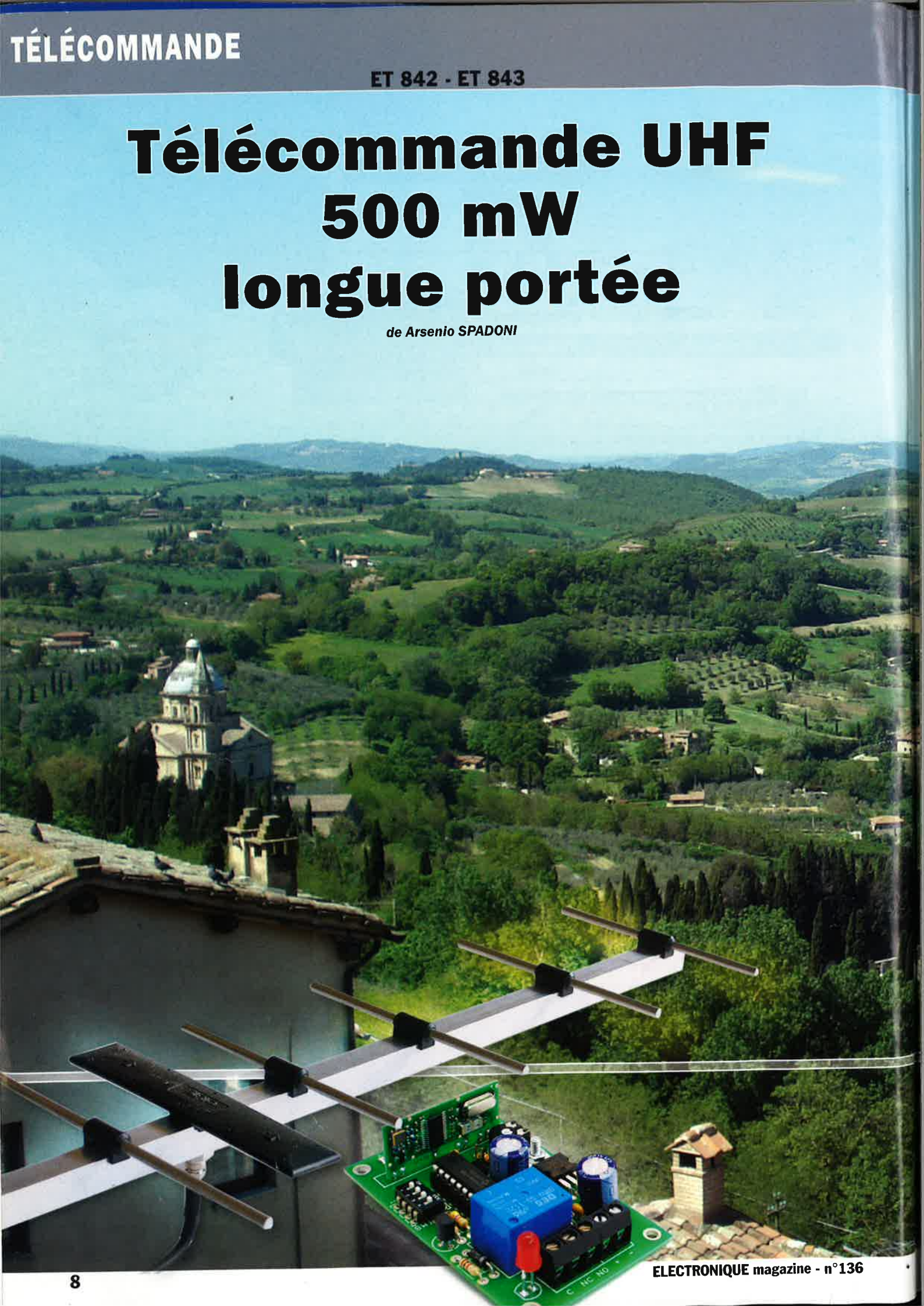


Brochage du connecteur USB 3.1.

ET 842 - ET 843

Télécommande UHF 500 mW longue portée

de Arsenio SPADONI



Nous vous proposons dans cet article de réaliser un ensemble émetteur/récepteur pouvant couvrir de grandes distances, grâce à une directive récente de l'Union Européenne qui nous permet d'utiliser la bande des 869 MHz jusqu'à une puissance maximale de 0,5 W.

Certains penseront qu'il semble dépassé de proposer la conception d'un système de commande à distance traditionnelle, à l'ère du smartphone qui utilise les réseaux GSM et Wi-Fi. Cependant il existe encore des endroits où ces technologies ne sont pas disponibles.

Si à cela nous ajoutons le fait qu'aujourd'hui, grâce à la nouvelle directive Européenne, il est possible d'utiliser une puissance d'émission HF assez importante pour contrôler une pompe ou un système d'alarme situés à plusieurs kilomètres, il est clair que ce projet n'est pas dénué de sens, car il peut résoudre de nombreux problèmes de contrôles à distance de manière économique (il n'est pas nécessaire d'avoir un abonnement SIM) et fiable.

Ce système **émetteur/récepteur fonctionne dans la bande des 869 MHz** qui peut être utilisée librement (il n'y a pas besoin de licence ou de frais à payer) pour la transmission de données avec une puissance maximale de **500 mW**.

La seule limitation est celle qui concerne le **temps d'occupation de la fréquence qui ne doit pas dépasser 10 % de la base horaire**. Cela ne pose pas de problème car dans la plupart des cas, les systèmes de commande à distance fonctionnent de façon brève et intermittente, donc le seuil des 10 % ne sera normalement jamais dépassé.

Cependant, afin d'effectuer les choses dans le respect des règles, notre système, après la transmission d'une chaîne de données, s'inhibe pendant une durée 10 fois plus grande que celle de la transmission. Avec ce type de fonctionnement nous sommes en conformité avec la loi.

En ce qui concerne la puissance d'émission HF de 0,5 W, même les non-experts savent qu'à ces fréquences il est possible de couvrir des distances de plusieurs dizaines de kilomètres, surtout si vous utilisez des antennes directives.

Pour être précis, la **loi interdit l'utilisation d'antennes directionnelles pour la transmission**, mais l'utilisation d'une bonne antenne de type « **Yagi** » pour la **réception** (ce qui est tout à fait légal) peut augmenter considérablement la portée.

Lors de nos tests, nous avons essayé une antenne « Yagi » en réception avec un gain de – 10 dB, le signal a parcouru **un peu moins de 5 km en rase campagne**.

A l'issue de nos tests, nous pouvons dire que la portée avec des antennes classiques omnidirectionnelles dépasse facilement le kilomètre. Avec une antenne directive performante en réception et en terrain dégagé il est possible d'atteindre 5 km en respectant la réglementation en vigueur.

À ce stade, de nombreux lecteurs vont se demander si la réalisation d'un tel système est complexe, difficile à assembler, à calibrer et à régler au niveau de la partie haute fréquence.

En fait rien de bien compliquer, car nous utilisons des modules AUREL professionnels. Il suffit d'alimenter le système, de brancher l'antenne et d'appliquer sur l'entrée le signal à transmettre.

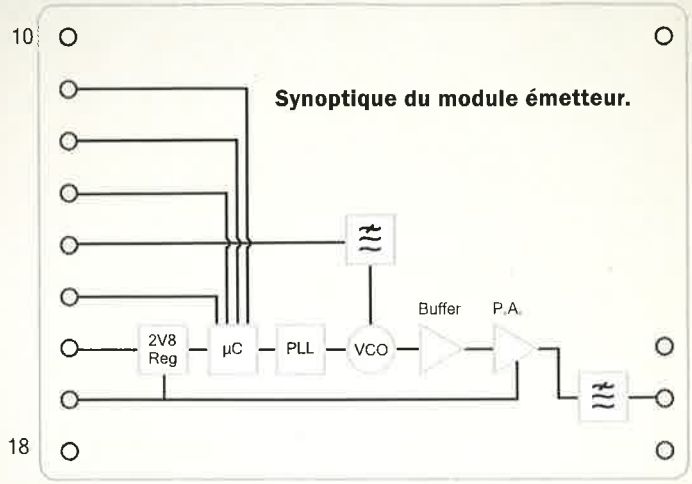
Plus précisément nous utilisons pour ce projet deux modules commercialisés par le fabricant AUREL.

Le module émetteur AUREL TX869BOOST

C'est un module hybride commercialisé par la société AUREL. Il dispose d'un émetteur radio UHF complet fonctionnant sur la bande des 869 MHz, avec 4 fréquences différentes (869,45 MHz ; 869,50 MHz ; 869,55 MHz ; 869,60 MHz) sélectionnables par l'intermédiaire de deux entrées équipées de résistances internes de « pull-up ». Reportez-vous au Tableau 1. L'oscillateur est basé sur un synthétiseur de fréquence à quartz et un oscillateur contrôlé en tension (VCO) qui génère un signal périodique dont la fréquence varie en fonction de la tension d'entrée. Pour changer la fréquence du canal, il suffit de configurer les broches de sélection (11 et 12).

En sortie de l'oscillateur se trouve un amplificateur HF linéaire d'une grande efficacité, capable de développer une puissance de +28 dBm (soit 500 mW) avec une tension d'alimentation de 5 V et une puissance de +24 dBm avec une tension d'alimentation de 3 V. La puissance de sortie se réfère à une charge de 50 Ω.

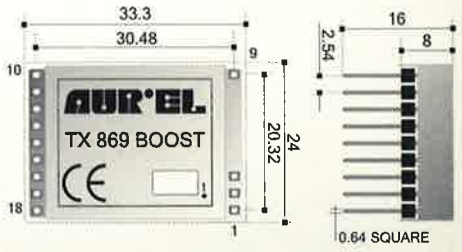
L'entrée des données à transmettre (qui doivent être codées de préférence avec un code Manchester), s'effectue sur la broche 14. Le débit maximum des données est de 19200 bps.



Synoptique du module émetteur.

Brochage

1) GND RF
2) Antenne
3) GND RF
9) GND
10) GND
11) FSEL0
12) FSEL1
13) AUX IN (N.C)
14) DATA IN
15) PA_CTRL
16) ENABLE
17) VCC
18) GND



Le module peut être mis en « stand-by » ou en veille (oscillateur et amplificateur HF désactivés) à l'aide d'une entrée de commande appropriée située sur la broche 16.

En mettant cette broche à un niveau logique haut (1) le module est actif, si elle se trouve à un niveau logique bas (0) le module est en stand-by.

S'il n'y a pas de pertes dans les données transmises et que la transmission s'effectue correctement avec une puissance et une modulation optimales, les données à transmettre sur la broche 14

Il s'agit des modules **TX869BOOST** et **RX-FM8SF**. Le premier est un **émetteur 4 canaux** capable de délivrer une puissance de **500 mW** sur la bande des **869 MHz** avec une modulation de type FM.

Le second est un **récepteur superhétérodyne** fonctionnant à **869,50 MHz** (un des canaux de l'émetteur TX) avec une **sensibilité** de **-105 dBm**. Reportez-vous aux encadrés respectifs de ces deux modules pour voir leurs caractéristiques détaillées.

Le projet décrit dans ces pages utilise une **transmission codée** (nécessaire pour éviter les fausses alarmes ou les fausses activations) gérée par un microcontrôleur.

L'application proposée dans ce projet est la plus courante, cependant nous voulons fournir à nos lecteurs toutes les informations nécessaires pour réaliser des circuits similaires mais différents. Par exemple avec plus de canaux ou avec une fonctionnalité légèrement différente.

Dans notre cas, vous pouvez activer l'émetteur avec de simples boutons poussoirs ou des interrupteurs statiques.

Le récepteur dispose d'un relais qui peut fonctionner aussi bien en mode monostable que bistable.

Voyons maintenant l'émetteur et le récepteur en détail.

L'émetteur

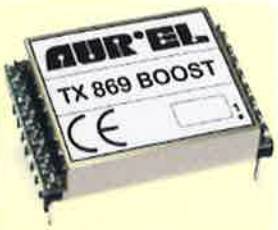
Commençons par étudier le schéma électrique de l'émetteur visible en figure 1. Il est conçu pour commander à distance le récepteur à l'aide de deux boutons poussoirs ou des sorties d'autres dispositifs aux niveaux TTL ou CMOS.

Le cœur de l'émetteur est le **microcontrôleur U1** (PIC16F88) qui est chargé de **surveiller en permanence l'état des entrées RB1** et **RB3** reliées respectivement aux boutons poussoirs **P1** et **P2**.

Dès qu'un bouton est pressé, le PIC génère une trame de données codées transmise à l'entrée du module TX869BOOST.

Tableau 1 - Fonctions des broches

Broche	Nom	Description
1 - 3	GND RF	Connexion du négatif de la sortie HF (plan de masse)
2	Antenne	Connexion de l'antenne de 50 Ω Sortie HF de l'émetteur.
9 - 10	GND	Connexion du négatif de l'alimentation (plan de masse).
11	FselO	Sélection de la fréquence d'émission. Active bas (pull-up interne).
12	FseL1	Sélection de la fréquence d'émission. Active bas (pull-up interne).
13	NC	Réservée (en interne connectée au µC pour des applications ultérieures).
14	Data In	Entrée des données de 0 V à 5,5 V max, avec une résistance de 10 kΩ à la masse.
15	PA_CTRL	Entrée analogique de régulation de la puissance de sortie. Ouvert = puissance maximale ; GND = puissance minimale.
16	Enable	Connectée au positif ou au négatif de l'alimentation comme suit : 0 = PWDN (module en veille avec une consommation maximale de 1 µA); 1 = Actif (module allumé).
17	Vcc	Positif de l'alimentation.
18	GND	Connexion du négatif de l'alimentation (plan de masse).



ne doivent pas être appliquées avant un délai de 100 ms lorsque la broche 16 passe d'un niveau « 0 » à un niveau « 1 ». La désactivation du module doit s'effectuer au minimum 30 ms après le dernier bit transmis.

En outre, les données doivent toujours être précédées de « bits de démarrage » constitués par une série d'impulsions rectangulaires d'une durée d'au moins 5 ms, qui préparent l'envoi des données réelles.

En ce qui concerne le réglage du canal sur lequel transmettre (c'est-à-dire la fréquence d'émission), il est réalisé par l'intermédiaire des broches 11 et 12 selon le Tableau 2. Après avoir réglé la valeur de la fréquence d'émission via les broches 11 et 12, le changement de canal devient effectif lorsque la broche 16 (ENABLE) est mise à un niveau bas puis à un niveau haut. Le niveau haut correspond à VCC et ne doit pas être inférieur à 2, 8 V. Les entrées de contrôle et des données acceptent des signaux TTL (0 – 5 V) ou CMOS (0 – 3 V).

Il est **nécessaire que les données soient codées afin d'éviter des perturbations avec d'autres dispositifs radiocommandés** dans le voisinage, mais aussi des perturbations pouvant interagir avec le récepteur.

Le microcontrôleur **fonctionne comme un circuit codeu**r de données à transmettre. Pour être exact, à chaque appui sur P1 (RB1) ou P2 (RB3), il génère une **trame** composée de 4 blocs :

- un bloc contenant les « **bits de démarrage** » ;
- un bloc contenant un **en-tête** (header) ;
- un bloc contenant le **code modifiable** à volonté ;
- un bloc contenant la **commande**.

Le bloc contenant les « **bits de démarrage** » correspond à une **onde carrée** d'une durée de **5 ms** et d'une fréquence de quelques kHz. Cela est nécessaire pour que le module TX869BOOST puisse exécuter correctement les transmissions précédant le code et le reste de la trame.

Plus exactement, le microcontrôleur envoie comme « **bits de démarrage** » **8 octets** de type « **01010101** ». Le bloc contenant les « bits de démarrage » permet de **préparer l'émetteur**.

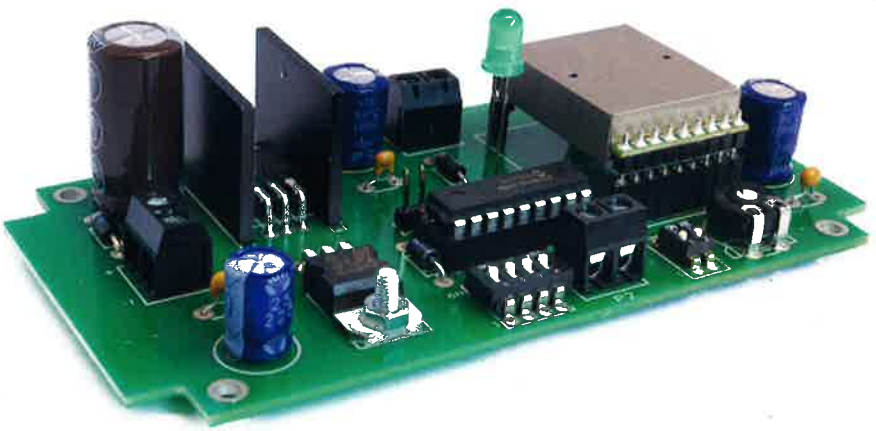
L'**en-tête** (header) correspond à un **code fixe** généré par le microcontrôleur afin d'indiquer au microcontrôleur du **récepteur qu'il doit se préparer** à **recevoir** des données.

Cette technique consiste essentiellement à diminuer le bruit électromagnétique présent dans l'atmosphère et dû aux différentes ondes radio, GSM, Wi-Fi, TNT.

Cela pourrait perturber l'oscillateur local du récepteur, les données en sortie de ce dernier pourraient être incorrectes et donc mal interprétées par le microcontrôleur du récepteur.

C'est pour cette raison qu'il serait impensable de demander au microcontrôleur du récepteur de reconnaître une trame sur la base d'une simple logique de commutation non codée (0 ou 1), le récepteur fonctionnerait de manière incohérente voire aléatoire.

Afin de nous assurer que le microcontrôleur du récepteur distingue la trame comportant les données réelles du bruit électromagnétique, l'émetteur génère dans l'**en-tête 2 octets** de type « **10101010 + 10110110** » envoyés en séquence, et qui sont faciles à distinguer des signaux présents dans l'atmosphère (bruit).



Le récepteur peut ainsi facilement interpréter les données de l'en-tête car elles ont la même cadence et la même fréquence, de sorte que cette technique est plus que suffisante pour préparer la séquence de décodage par le récepteur.

Un autre détail de ce projet est que pour faciliter le décodage des données par le récepteur, la transmission est effectuée en adoptant le **codage « Manchester »**.

Cette technique **consiste à transmettre 2 bits pour chaque bit de données.** Dans ce système de codage, les périodes de transmission des bits sont divisées en 2 intervalles égaux.

Chaque période de transmission comporte une transition en son milieu, ce qui facilite la synchronisation entre l'émetteur et le récepteur.

Afin d'assurer une synchronisation correcte pour les intervalles de temps utilisés pour l'échantillonnage du signal par le récepteur, le codage Manchester envoie une séquence de « bits de démarrage » constituée par 64 « 0 » et « 1 » alternés.

Cela produit une onde carrée qui permet au récepteur de déterminer la valeur des intervalles de temps. La combinaison de cette séquence de « bits de démarrage » et de la transition au milieu de la transmission de chaque bit permet de se passer d'une horloge externe pour synchroniser l'émetteur et le récepteur.

En d'autres termes, chaque donnée est représentée par 2 niveaux logiques, dont le premier est égal à celui des données, et l'autre est l'inverse.

Si la donnée est un « 1 » logique, la paire de bits qui la représente est « 10 » alors que si elle vaut « 0 », la donnée est représentée par les bits « 01 ».

En réalité, le codage Manchester a pour effet de sensibiliser le récepteur lors de la transition, soit sur le front montant du signal de commutation lors du passage d'un « 0 » logique

L'émetteur

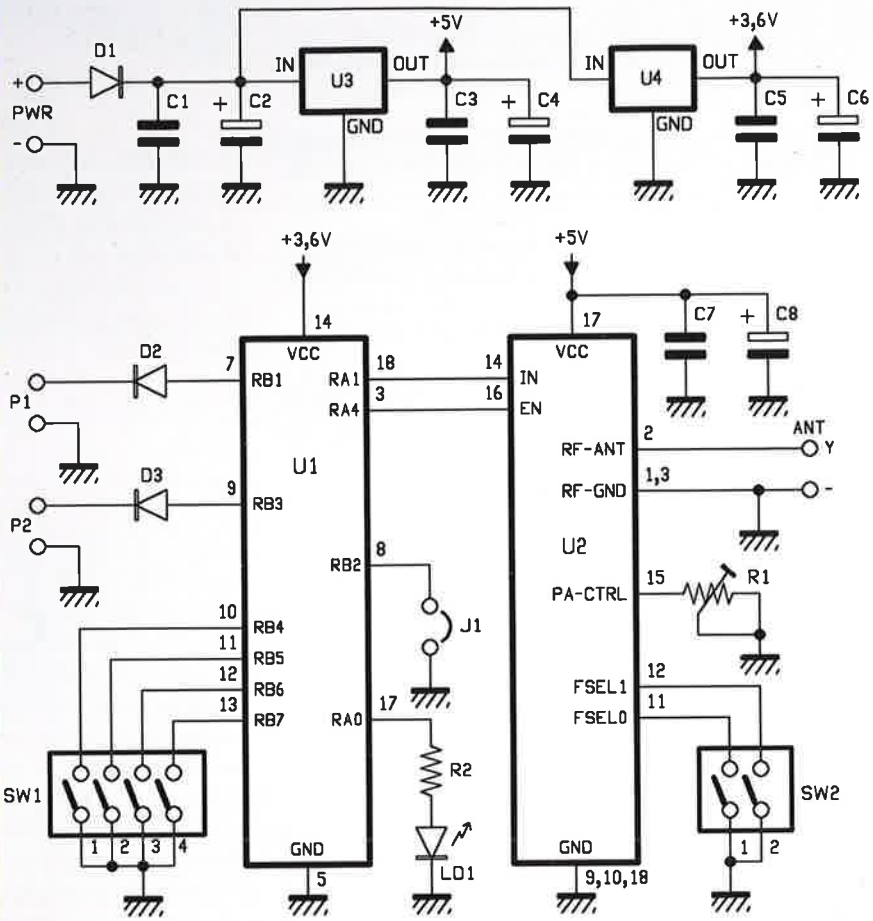


Figure 1 : schéma électrique de l'émetteur.

Liste des composants du ET 842

R1..... trimmer 2,2 kΩ
R2..... 330 Ω

C1..... 100 nF multicouche
C2..... 1000 µF/25 V électrolytique
C3..... 100 nF multicouche
C4..... 470 µF/25 V électrolytique
C5..... 100 nF multicouche
C6..... 220 µF/16 V électrolytique
C7..... 470 µF/25 V électrolytique
C8..... 100 nF multicouche

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

U1..... PIC16F88-I/P (MF842)
U2..... TX869B00ST
U3..... 7805
U4..... LD1086-36

SW1... DIP switch 4 voies
SW2... DIP switch 2 voies

LD1.... LED 5 mm verte

Divers

Support ci 2 x 9 broches
Vis 10 mm 3MA (x2)
Ecrous 3MA (x2)
Bornier 2 pôles (x3)
Barrette mâle 2 pôles au pas de 2,54 mm
Cavalier
Dissipateur pour TO220 (ML26)
Antenne 869 MHz

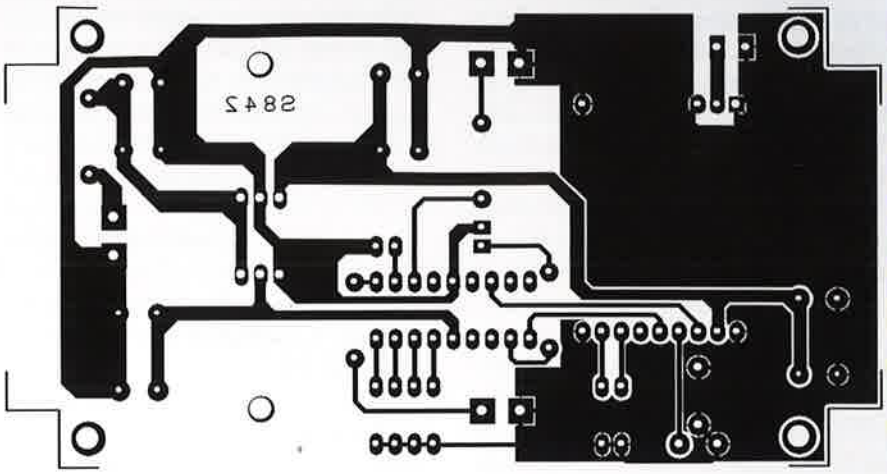


Figure 2 : circuit imprimé à l'échelle 1 : 1 côté soudures de l'émetteur.

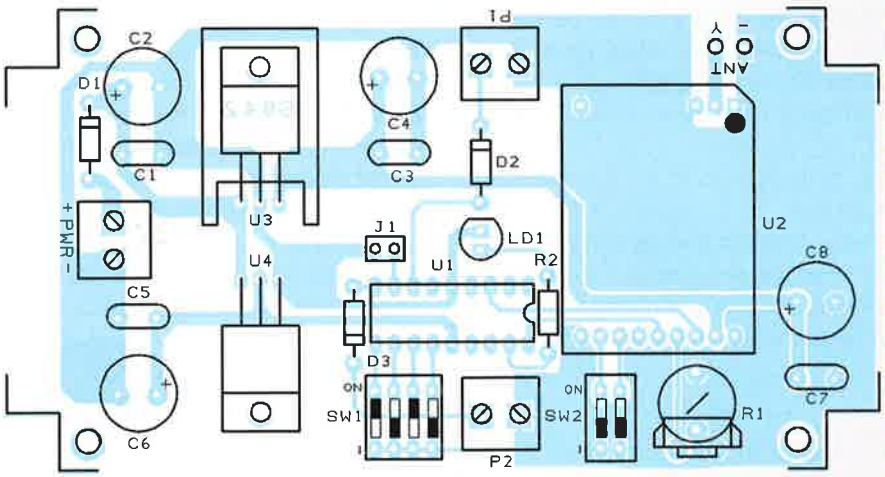


Figure 3 : plan de câblage des composants de l'émetteur.

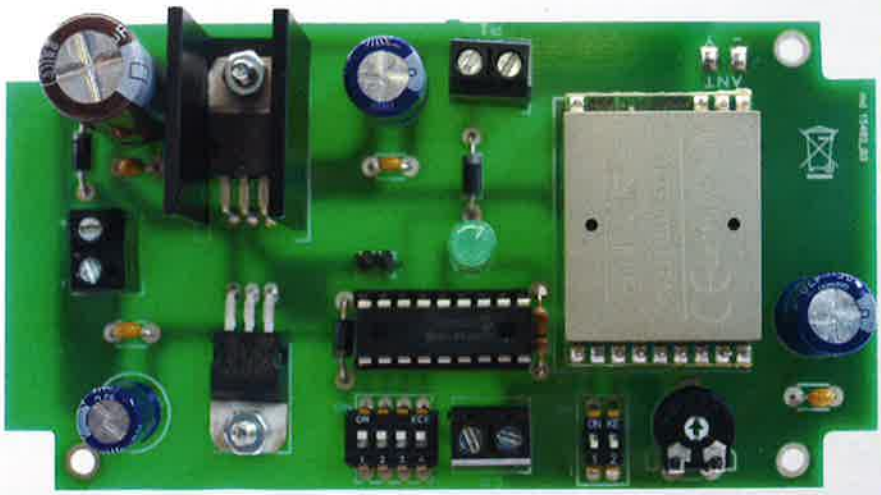


Figure 4 : photo de l'un de nos prototypes de l'émetteur. Sur la droite vous pouvez voir le module TX869BOOST.

vers un « 1 » logique (0 → 1) soit sur le front descendant d'un « 1 » logique vers un « 0 » logique (1 → 0).

Malheureusement, même s'il est vrai qu'il améliore l'immunité aux bruits, ce codage occupe une bande passante importante car pour chaque bit de donnée à transmettre, 2 bits sont envoyés. Le débit de données est réduit de moitié.

Par conséquent, comme les modules autorisent une transmission de **19200 bps**, avec le **code Manchester** l'information transmise sera de **9600 bps**. Le codage Manchester est utilisé dans les réseaux Ethernet et d'autres types de réseaux locaux.

Continuons par l'analyse de la trame transmise par l'émetteur et examinons le **code**. Celui-ci est un octet dont **4 bits sont définis par l'état du DIP switch SW1**, ce qui confère une sécurité. SW1 est constitué par 4 commutateurs qui permettent de définir **16 combinaisons possibles**.

Nous pouvons donc utiliser **dans le même environnement plusieurs paires d'émetteurs/récepteurs**. La **dernière partie de la trame contient la commande proprement dite** pour le récepteur, étant donné que, comme nous l'avons mentionné plus haut, c'est l'émetteur qui impose le mode de fonctionnement.

La **commande indique au récepteur** si le **relais doit s'enclencher ou retourner au repos**. De façon plus précise, en activant l'entrée P1 cela provoque l'enclenchement du relais, et en appuyant sur P2 le relais est désactivé. Le tout, indépendamment du mode de fonctionnement choisi pour le récepteur.

Chaque entrée est associée à un code particulier. **P1 correspond** à « **01010101** » et **P2 correspond** à « **10101010** ». Par conséquent, la dernière partie de la trame est « 01010101 » si P1 est pressé. Si maintenant P2 est pressé, la dernière partie de la trame est « 10101010 ».

Il va sans dire que pour utiliser la télécommande en mode bistable, il est

nécessaire d'utiliser les deux entrées (étant donné que l'une active et l'autre relâche le relais), tandis que dans le cas d'un mode monostable, vous devez utiliser uniquement le bouton P1.

Si vous appuyez sur P2 dans ce mode, cela n'a aucun effet sur le récepteur. Pour avoir la certitude que la commande a bien été reçue par le récepteur, chaque transmission de la trame est répétée 5 fois.

La partie la plus intéressante de notre circuit est certainement le module TX869BOOST d'une puissance de 500 mW, qui est utilisé ici dans une configuration classique. Il est alimenté par une tension continue stabilisée de 5 VDC provenant de la sortie d'un classique régulateur 7805.

Pour pouvoir alimenter le microcontrôleur en 3,6 VDC, nous faisons appel à un 2ème régulateur à faible chute de tension U4 (LD1086-36). Le module HF dispose d'un DIP switch (SW2) qui permet de sélectionner le canal sur lequel transmettre (fréquence d'émission).

Dans notre projet, le **récepteur ne dispose que d'un seul canal** c'est-à-dire une seule **fréquence de réception à 869,5 MHz**. Nous devons donc **paramétrer le DIP switch SW2 de l'émetteur comme suit** : nous laissons ouvert celui qui est connecté à la broche 11 et nous fermons celui relié à la broche 12.

Le module TX869BOOST dispose d'un réglage de la puissance d'émission HF grâce à la broche 15.

En reliant un trimmer de 2,2 kΩ, nous pouvons faire **varier la puissance d'émission** de **+24 dBm à +28 dBm**. Par exemple, dans les endroits où la puissance d'émission n'a pas d'importance (faible distance), le fait de pouvoir la réduire permet de réduire le courant consommé et donc dans le cas d'un système alimenté sur batterie, cela permet une plus grande autonomie.

Les **données à transmettre** doivent être dirigées vers la **broche 14** d'entrée (IN). Puisque l'émetteur peut être activé et désactivé (mode veille) au moyen d'un niveau logique, nous utilisons la **broche 3** (RA4) du microcontrôleur que nous relions à la **broche 16** (EN) du module afin de **réduire la consommation de courant et transmettre uniquement lorsque cela est nécessaire** (cela évite de perturber d'autres systèmes environnants et d'être conforme à la loi car la durée d'émission ne doit pas dépasser 10 %).

Par conséquent, le microcontrôleur maintient, la plupart du temps, à un **niveau logique bas la broche « EN »** du module et la met à un **niveau logique haut pendant toute la durée de chaque transmission**.

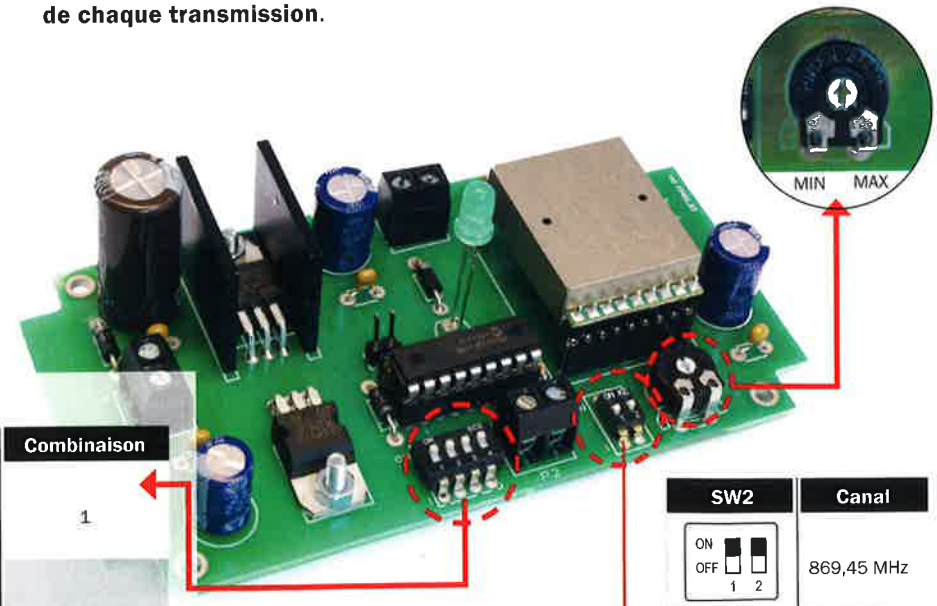
En réalité dès que le module est activé, il faut attendre un certain temps (100 ms) avant de pouvoir transmettre, la broche 16 est forcée à un niveau logique haut pendant 100 ms puis ensuite les données sont envoyées à la broche 14 du module.

Conformément aux spécification d'AUREL, le module doit être désactivé après un certain temps par rapport à la fin de la trame envoyée, soit au bout de 30 ms.

La **broche 16** (EN) du module reste donc à un **niveau logique haut pendant « 100 ms + la durée de la transmission + 30 ms »**.

Le module **émetteur TX869BOOST**, pour fonctionner correctement avec une bonne portée, doit être **équipé d'une antenne externe** reliée entre la **broche 2** (RF – ANT) et les **broches 1 et 3** (RF- GND).

Avant de détailler le fonctionnement du récepteur, nous devons éclaircir quelques points concernant la **directive « Decision 2006/771/EC**



SW1	RB4	RB5	RB6	RB7	Combinaison
ON OFF 1 2 3 4	0	0	0	0	1
ON OFF 1 2 3 4	1	0	0	0	2
ON OFF 1 2 3 4	0	1	0	0	3
. . .	. . .	. . .	. . .	. . .	. . .
ON OFF 1 2 3 4	1	1	1	1	16

SW2	Canal
ON OFF 1 2	869,45 MHz
ON OFF 1 2	869,50 MHz
ON OFF 1 2	869,55 MHz
ON OFF 1 2	869,60 MHz

Figure 5 : Les 2 commutateurs DIP switch permettent de régler le codage de la transmission (SW1) et la fréquence d'émission (SW2). Le trimmer R1 permet de régler la puissance d'émission HF. Dans notre cas, nous avons choisi la fréquence 869,50 MHz.

- **EUR-Lex - Europa.eu** » (la décision 2006/771/CE) relative à l'harmonisation du spectre radioélectrique en vue de l'utilisation de dispositifs à courte portée (quelques kilomètres).

Dans cette directive sont stipulées les conditions d'utilisation de la bande 869,4 - 869,65 MHz :

- puissance/intensité de champ maximale : **500 mW** ;
- coefficient d'utilisation jusqu'à **10 %** ;
- l'**espacement des canaux** doit être de **25 kHz**, sauf lorsque la totalité de la bande est utilisée comme canal unique pour la transmission de données à haute vitesse ;
- les applications **vidéo sont exclues**.

Nous devons donc faire en sorte qu'indépendamment des commandes, le **module émetteur ne doit transmettre que pendant 10 % du temps**, c'est-à-dire pendant **6 min par heure** de temps écoulé (60/10).

C'est pour cette raison que chaque fois que vous activez une entrée (pressez P1 ou P2), le **microcontrôleur génère 5 fois la trame pendant environ 1 seconde et désactive l'émetteur pendant 9 s**. Nous sommes sûrs que la transmission ne dure pas plus d'une seconde pour 10 secondes écoulées.

Cela signifie que même si P1 ou P2 sont pressés de façon continue sans être relâchés, l'émetteur ne transmettra la commande que pendant une seconde toutes les 10 secondes ou 6 min par heure.

Pour chaque transmission, ou à la suite de l'émission d'une série de trames, le microcontrôleur U1 active la broche RA0 ce qui allume la LED LD1 afin d'indiquer visuellement qu'une transmission est en cours.

Le récepteur

Nous allons voir maintenant le fonctionnement du récepteur. Il s'agit d'un circuit relativement simple dont le cœur du montage est un **microcontrôleur** (**PIC16F88**) et qui a pour tâche de **décoder les signaux provenant de la sortie du récepteur RX-FM8S** et **gérer en conséquence l'état du relais**. L'ensemble du circuit de réception est alimenté par une tension de 5 VDC obtenue par un classique régulateur 7805.

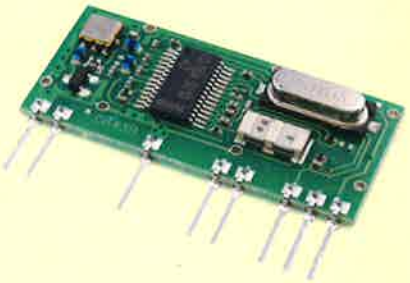
Le module RX-FM8S reçoit les ondes radio via l'antenne connectée entre les broches 3 (RF – ANT) et 2 et 7 (RF-GND). Il syntonise les ondes grâce à son propre étage accordé.

Le démodulateur **convertit la fréquence de la porteuse modulée et extrait les données** qu'il dirige vers sa sortie, c'est-à-dire la **broche 14** (OUT).

Par conséquent, la **trame ainsi décodée**, atteint l'entrée **RA7** du microcontrôleur, dont **le programme attend en boucle** non pas une série de signaux carrés mais **l'arrivée de l'entête** (header) de la trame émise. Lorsqu'une trame se présente, le PIC vérifie d'abord si le format des données est

Le module récepteur AUREL RX-FM8S

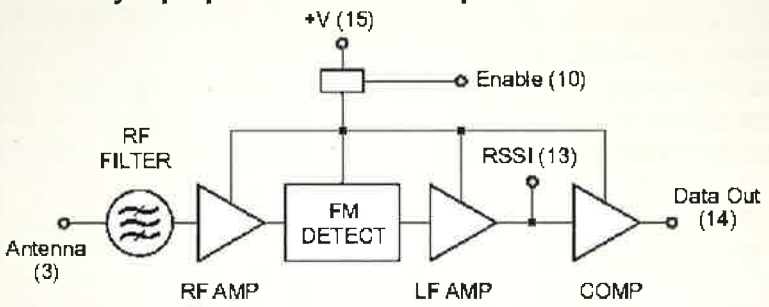
C'est un récepteur superhétérodyne FM avec une sensibilité élevée (-105 dBm) et une grande sélectivité (110 kHz de bande passante avec une antenne). Il dispose d'un filtre SAW à ondes de surface au niveau de l'étage d'entrée, ce qui le rend insensible aux interférences et optimise la réception.



Il peut être alimenté, puis mis en veille en l'absence de signal afin de minimiser la consommation (0,5 µA). Cette fonction est obtenue en mettant la broche 10 (ENABLE) à VDD pendant la réception, puis à un niveau logique bas pour passer en mode veille (power-down).

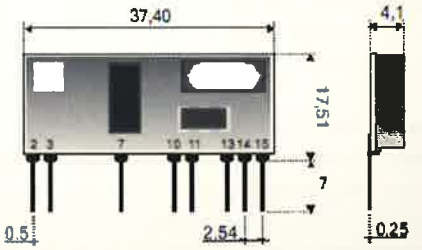
Le module fonctionne avec une tension d'alimentation comprise entre 4,5 VDC et 5,5 VDC et consomme un courant d'environ 6 mA. Les broches sont disposées en ligne (single-in-line), le module est construit sur un châssis en aluminium afin de réduire les perturbations.

Synoptique du module récepteur



Le brochage est le suivant :

2) Masse
3) Antenne
7) Masse
10) Enable
11) Masse
13) RSSI Out - Test point
14) Data Out
15) +V



correct, sinon il annule la procédure et retourne dans la boucle d'attente d'une nouvelle trame.

Le « firmware » du microcontrôleur ne tient pas compte des « bits de démarrage » qu'il considère comme une transmission aléatoire.

S'il reçoit et reconnaît l'en-tête (header), le programme va **lire** le « **prochain morceau** » de l'information contenue dans la trame, à savoir le **codage de l'émetteur** et le **compare** avec le paramétrage de ses lignes RB4, RB5, RB6, RB7 imposé par le DIP switch **SW1**.

À ce stade, **si le DIP switch du récepteur est configuré de la même manière que celui de l'émetteur, le programme lit alors la commande et l'exécute**. Ensuite il retourne dans la boucle d'attente d'une nouvelle trame.

Si le récepteur est réglé en **mode bistable** (J1 ouvert), le PIC met sa broche 15 (RA6) à un niveau haut lorsque les données de la commande reçue correspondent à « 01010101 » (P1) et à un niveau bas si l'octet de la commande correspond à « 10101010 » (P2).

Si le relais est déjà activé et qu'une commande de P1 arrive, rien ne change. De même si le relais est au repos et qu'une commande de P2 arrive, rien ne change.

En **mode monostable** (J1 fermé), le microcontrôleur met sa broche 15 à un niveau haut pendant une durée qui dépend du réglage du **trimmer R4**, la durée variant de **0,5 s** et **5 s**. Le trimmer fait varier une tension qui est appliquée sur l'entrée analogique **AN0** (17) du microcontrôleur.

Un convertisseur A/D interne échantillonne cette tension en 1024 valeurs qui peuvent être interprétées par le programme.

Par exemple si le trimmer est réglé d'une certaine façon qui provoque la présence d'une tension de 2,5 V sur la broche AN0, le programme lira la valeur de la variable correspondante comme égale à 512. Comme la tension maximale que nous pouvons appliquer sur une entrée est de 5 V, la plus petite tension qui peut être lue par le programme est donc de 5 /1024 = 4,9 mV.

Dans ce cas la variable du programme vaudra 1. La valeur 1024 provient de la documentation technique du PIC qui dispose de convertisseurs A/D de 10 bits soit 1024 possibilités ou valeurs. Lors de l'exécution d'une séquence d'activation monostable, le microcontrôleur ignore toutes les données arrivant du récepteur tant que le relais n'est pas retourné au repos.

Le **relais** de la sortie du récepteur est commandé par le microcontrôleur en utilisant un **transistor NPN qui fonctionne comme un interrupteur statique**. Le transistor T1, lorsqu'il devient conducteur, amène le côté de la bobine relié à son collecteur vers la masse.

Le récepteur

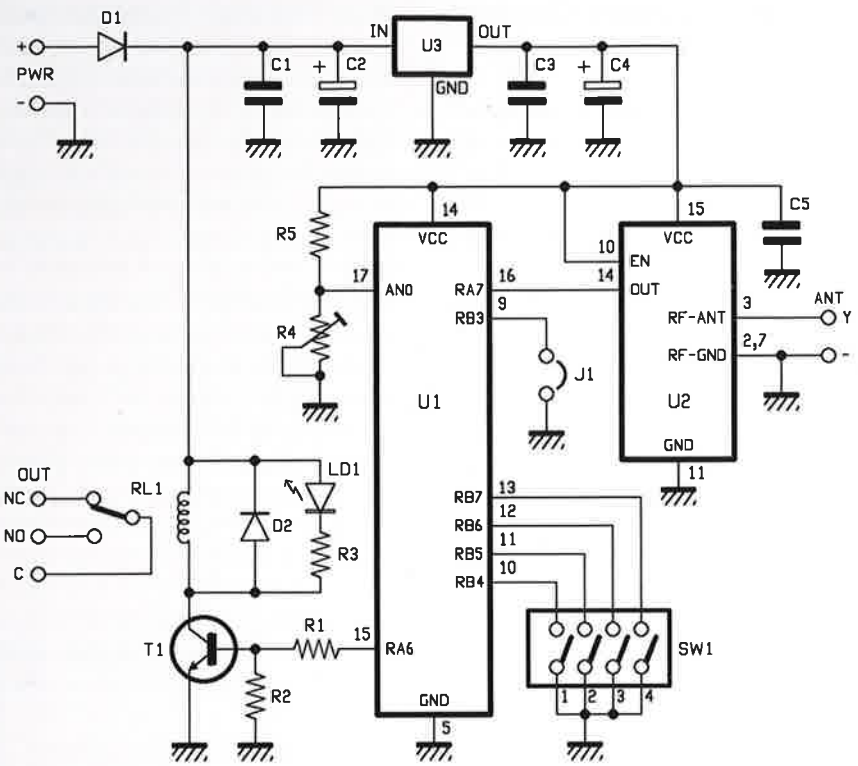


Figure 6 : schéma électrique du récepteur.

Liste des composants du ET 843

R1..... 4,7 kΩ
R2..... 10 kΩ
R3..... 1 kΩ
R4..... trimmer 2,2 kΩ
R5..... 330 Ω
C1..... 100 nF multicouche
C2..... 470 µF/25 V électrolytique
C3..... 100 nF multicouche
C4..... 470 µF/25 V électrolytique
C5..... 100 nF multicouche
D1..... 1N4007
D2..... 1N4007
U1..... PIC16F88-I/P (MF843)
U2..... RX-FM8SF869
U3..... 7805
SW1... DIP switch 4 voies
RL1.... relais 12V 220VAC/3A 1 contact
LD1.... LED 5 mm rouge

Divers

Support ci 2 x 9 broches
Vits 10 mm 3 MA
Ecrou 3 MA
Bornier 2 pôles
Bornier 3 pôles
Barrette mâle 2 pôles au pas de 2,54 mm
Cavalier
Antenne accordée à 869 MHz

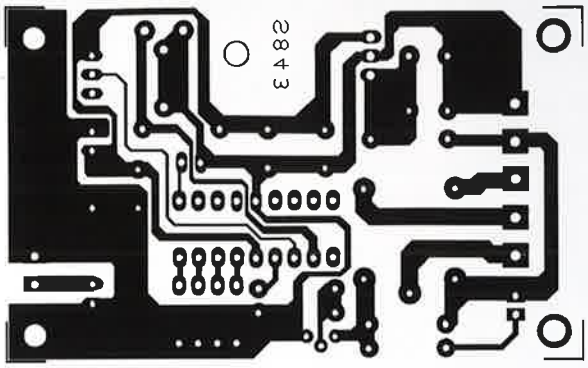


Figure 7 : circuit imprimé à l'échelle 1 : 1 côté soudures du récepteur.

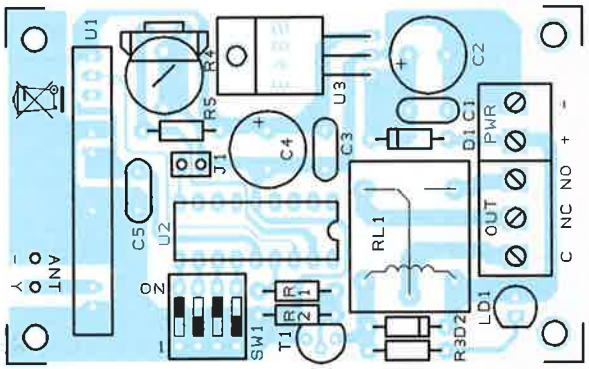
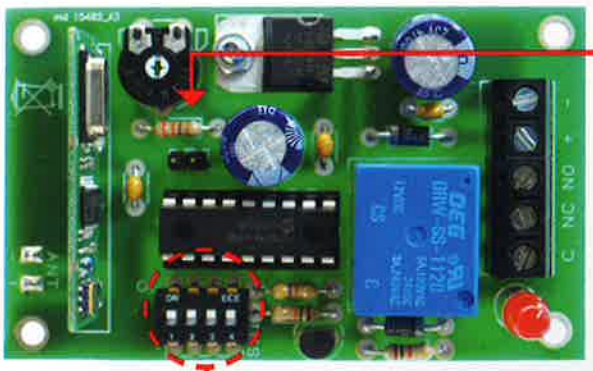


Figure 8 : plan de câblage des composants du récepteur.



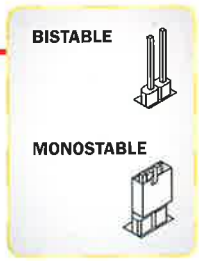


Figure 9 : photo de l'un de nos prototypes du récepteur.

Un **courant circule alors dans la bobine** du relais qui actionne le contact. La diode D2 protège la jonction « base – collecteur » du transistor pendant les phases transitoires (changement d'état de T1, bloqué ou conducteur) et évite que le relais se mette à osciller. La LED **LD1**, en série avec la résistance R3, indique que le **relais est activé** (contact collé).

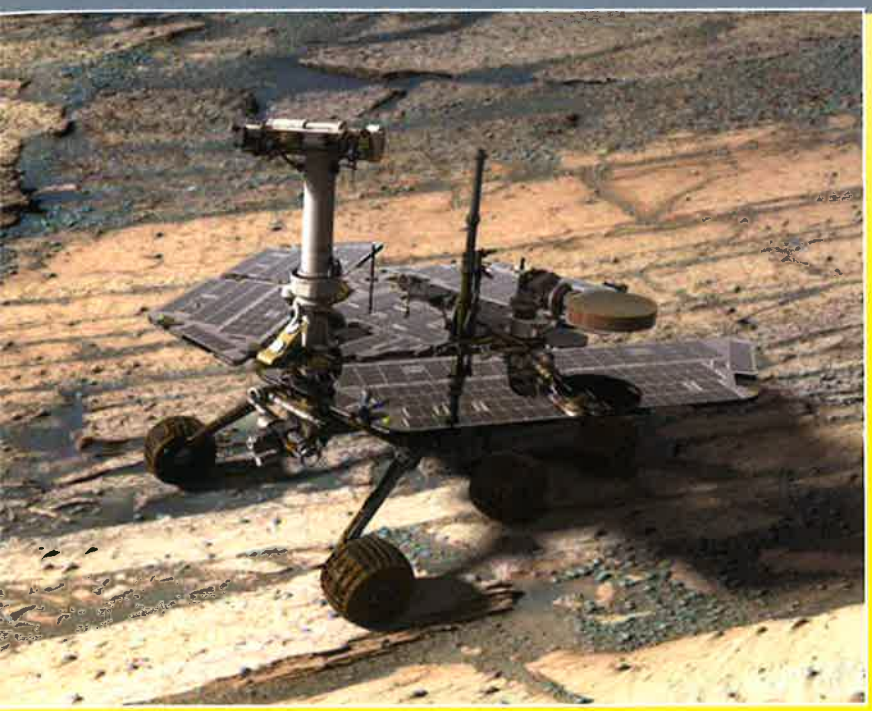
Réalisation pratique

Passons maintenant à la partie pratique de notre projet et voyons comment fabriquer l'émetteur et le récepteur. La première chose à faire est de préparer les circuits imprimés qui ne comportent qu'une seule face.

Ils sont donc relativement faciles à réaliser avec une gravure chimique traditionnelle. Pour cela vous devez télécharger les typons des circuits imprimés qui se trouvent sur notre site **www.electroniquemagazine.com** dans le sommaire détaillé de la revue **135** à l'onglet « **Télécharger** ».

SW1	RB4	RB5	RB6	RB7	Combinaison
ON OFF 1 2 3 4	0	0	0	0	1
ON OFF 1 2 3 4	1	0	0	0	2
ON OFF 1 2 3 4	0	1	0	0	3
. . .	. . .	. . .	. . .	. . .	. . .
ON OFF 1 2 3 4	1	1	1	1	16

Le codage du récepteur doit être identique à celui de l'émetteur, il est configuré à l'aide du DIP switch SW1. Le cavalier J1 permet de choisir le mode de fonctionnement du relais (monostable ou bistable).



Contrôle à distance de l'espace

Le contrôle à distance par les ondes radio a rendu possible des choses à la fois simples et plus complexes. Il permet non seulement d'actionner à distance des portes, des lumières ou des machines, mais aussi de piloter à distance un robot qui explore la planète Mars. En Janvier 2004, les sondes jumelles de la NASA (Spirit et Opportunity) se sont posées de chaque côté de la planète Mars. Leur mission devait durer plus de trois mois au cours desquels elles devaient tenter de rechercher des traces d'eau sous forme de glace.

Les deux robots ont été guidés par les ondes radio depuis la Terre, malgré les difficultés et les limites imposées par la distance et le donc le temps de latence important pour exécuter un ordre. En effet il faut considérer la lenteur de la propagation des ondes radio dans l'espace due aux grandes distances. Lorsque que nous sommes à quelques centaines de mètres voire quelques kilomètres sur Terre, la transmission des ondes radio est presque instantanée d'un point à un autre.

Par contre dès que la distance atteint des millions de kilomètres, cela entraîne des retards considérables. Nous savons que les ondes radio se propagent à très grande vitesse (300 000 km/s), mais dans le cas de distances interplanétaires pour arriver sur Mars qui se situe entre 56 à 100 millions de km de la Terre selon les périodes, les ondes radio mettent en moyenne 5 minutes pour arriver. Et autant de Mars vers la Terre. Donc entre chaque commande il faut attendre 10 minutes pour savoir si le robot a bien reçu l'ordre à exécuter.

Le transfert des données entre le robot et la Terre doit faire face à plusieurs contraintes. Le robot a un équipement de communication de puissance limitée. Il dispose d'une quantité d'énergie réduite due à la surface de ses panneaux solaires et l'équipement de télécommunications a dû être allégé au maximum. Les orbiteurs ont par contre dans ce domaine plus de capacité grâce à la taille de leurs panneaux solaires et de leur équipement de télécommunication. Le débit de la transmission est donc généralement faible, il est compris entre quelques bits par seconde avec l'antenne faible gain du robot et 32 bits par seconde en utilisant l'antenne UHF et le relais des sondes qui orbitent autour de Mars.

Pour que la transmission puisse avoir lieu, il faut que l'émetteur et le récepteur soient visibles l'un par l'autre. L'orbiteur « Mars Odyssey » qui a été utilisé pour le transfert de la majeure partie des données n'est visible que durant 10 minutes à chacun de ses passages au-dessus du robot. La communication n'est possible que durant 5 minutes.

Si le robot veut communiquer directement avec la Terre, il doit prendre en compte la rotation de la planète Mars et doit attendre si nécessaire de faire face à la Terre. Les communications directes vers la Terre sont, par ailleurs, limitées à 3 heures par jour martien pour limiter la consommation d'énergie et l'échauffement de l'électronique.

Les rares antennes suffisamment puissantes sur Terre pour recevoir les émissions des sondes sont celles du « Deep Space Network » de la NASA, qui peuvent suivre un grand nombre de missions planétaires simultanées. Des créneaux horaires d'une heure sont alloués à chaque robot pour chaque jour martien pour les communications montantes (envoi d'instructions aux robots, programme du jour) et descendantes (transfert de données scientifiques et de navigation).

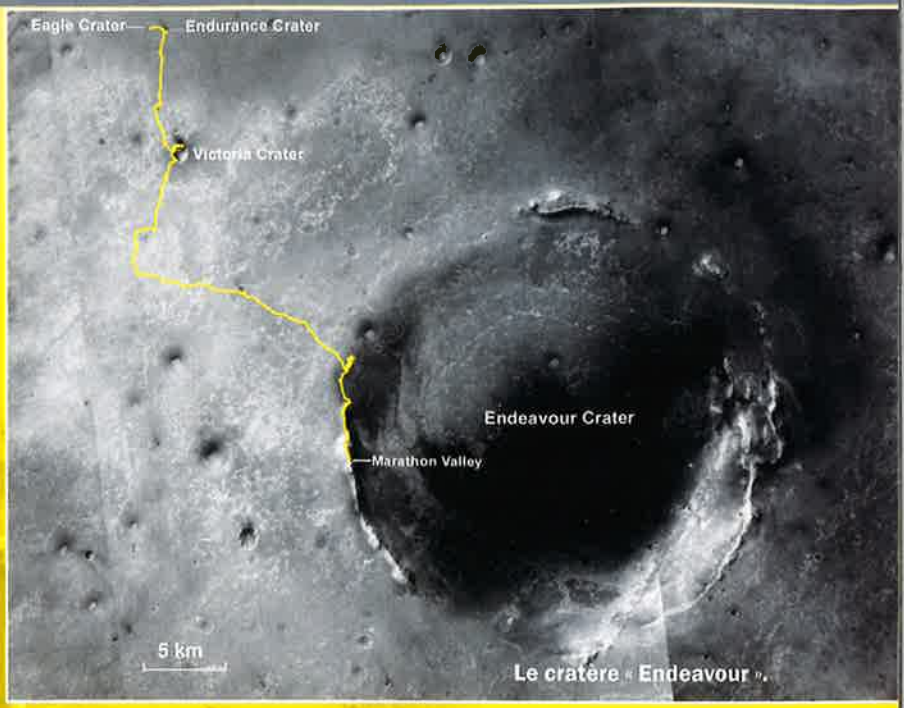
Une fois tous les deux ans la Terre et Mars se trouvent en opposition par rapport au Soleil. Il en résulte un black-out total des communications avec le robot qui dure deux semaines : durant cette période, le robot ne se déplace plus et exécute un programme d'observation au voisinage de sa position.

Le robot dispose, pour communiquer avec le contrôle au sol, de 3 antennes :

- une antenne grand gain (HGA) parabolique de 28 cm de diamètre, utilisée pour communiquer directement avec la Terre en bande X. La bande X est privilégiée pour les sondes spatiales (plage de fréquences d'ondes radio dite Supra Haute

Fréquence située aux alentours de 10 GHz), car elle permet le transfert de volumes importants de données sans nécessiter de grandes quantités d'énergie. L'antenne peut être pointée avec précision vers la Terre grâce à deux moteurs fournissant deux degrés de liberté. Son débit est de 11 kbit/s pendant 3 heures par jour au maximum ;

- une antenne faible gain omnidirectionnelle (LGA) fixe qui permet également de communiquer directement avec la Terre, également en bande X, mais avec un débit très faible ;
- une antenne UHF omnidirectionnelle de portée limitée et qui est utilisée pour communiquer avec les orbiteurs « Mars Global Surveyor » et « Mars Odyssey » lorsqu'ils se trouvent à la verticale du robot. Au cours d'un seul passage de 8 minutes, le robot peut transférer 7,5 Mo de données. Le transfert du même volume de données par émission directe vers la Terre durerait d'une heure et demie à cinq heures.



Le cratère « Endeavour ».

Depuis plus de 12 ans après leur arrivée sur la surface de Mars, les deux robots sont toujours opérationnels car les ingénieurs de la mission ont appris à gérer et à résoudre tous les dangers et difficultés d'un tel contrôle particulier. Plusieurs fois les robots ont été bloqués et plus d'une fois des dunes et des pierres les ont empêchés de suivre une trajectoire prédéfinie.

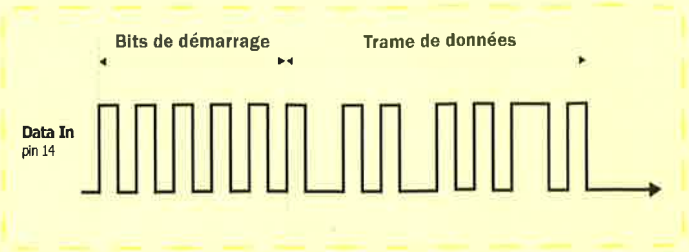
Toute l'expérience pratique acquise lors de cette mission a rendu possible de nouvelles explorations pour de futurs robots. « Opportunity », qui est le robot qui a eu le moins de problèmes, a envoyé le 25 janvier 2015 des images spectaculaires du cratère « Endeavour », objectif ambitieux car il a mis 2 ans pour l'atteindre (la vitesse moyenne est de 2 km par jour). Le 13 juillet 2015, « Opportunity » s'éloigne du cratère et, le lendemain, s'engage dans la « Marathon Valley » (voir les photographies ci-après).

L'autre robot « Spirit » a connu une mission plus difficile car ses panneaux solaires sont recouverts de sable, une roue est bloquée et un câble gêne le déplacement du bras robotisé.


Descente du robot « Opportunity » vers « Marathon Valley ».

Coucher de soleil sur Mars photographié par le robot Spirit.

Le protocole de communication



La trame générée par le microcontrôleur a la structure suivante :

bits de démarrage – header – codage – commande (du bouton pressé)

Les bits de démarrage correspondent à un ensemble de **8 octets** ayant la structure suivante :

01010101 01010101 01010101 01010101 01010101 01010101 01010101 01010101

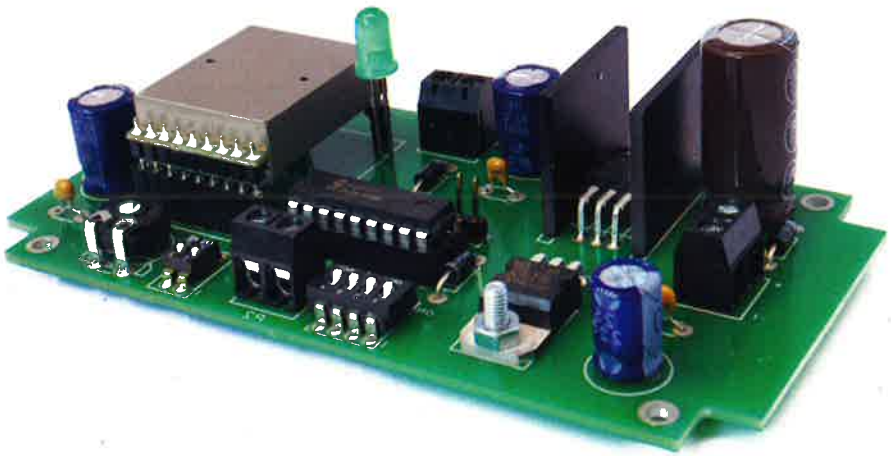
Ces 8 octets (bytes) servent à préparer le module TX pour la transmission.

L'**header**, qui comprend **2 octets** (10101010 et 10110110), sert à **synchroniser le récepteur avec l'émetteur** et le prépare au décodage. Le codage est utilisé pour identifier le récepteur et la commande correspond au code (1 octet) du bouton pressé. L'octet vaut « **01010101** » pour **P1** et « **10101010** » pour **P2**.

Le codage est effectué selon le protocole « **Manchester** « et chaque trame est envoyée 5 fois de suite pour garantir que la commande sera bien exécutée.

Notez aussi que ce projet de contrôle à distance est également disponible sous forme de kit complet (voir les annonces dans la revue).

La fabrication des 2 cartes n'est pas difficile, commencez comme d'habitude par souder les résistances, les diodes en respectant la polarité, puis continuez avec les supports des microcontrôleurs, les condensateurs non polarisés, les condensateurs électrolytiques (le moins est indiqué sur le boîtier).



Soudez les 2 DIP switch de l'émetteur, le marquage « ON » doit être positionné respectivement du côté de U1 pour SW1 et du côté de U2 pour SW2.

Le DIP switch SW1 du récepteur doit avoir le marquage « ON » vers le microcontrôleur. Soudez la LED verte.

Continuez en soudant les borniers des boutons P1 et P2 de l'émetteur, puis les 2 régulateurs en pliant leurs pattes à 90 ° sans les casser et en insérant un dissipateur d'environ 20 °C/W pour U3. Insérez et soudez la barrette femelle du module TX869BOOST. De même pour le récepteur soudez le transistor, la LED, les borniers, le régulateur, le relais et le module FM8SF869.

À ce stade connectez les fils des deux alimentations, c'est à dire celle de l'émetteur et celle du récepteur aux borniers correspondants, ainsi que la sortie du récepteur.

Ensuite, vous devez connecter aux deux point nommés « ANT » des antennes, des modèles 869 Mhz existent dans le commerce. Sinon vous pouvez utiliser **pour les antennes un fil rigide d'une longueur de 8,5 cm pour l'émetteur et le récepteur**.

Vérifier que tout fonctionne correctement **en faisant attention aux paramétrages des DIP switch**. Si la portée est insuffisante pour vos besoins, vous pouvez utiliser une antenne UHF omnidirectionnelle pour l'émetteur et, dans le cas du récepteur une antenne « Yagi » sur mat.

Pour l'alimentation de l'**émetteur**, vous devez utiliser une tension continue comprise entre **9 VDC** et **15 VDC** avec un courant d'au moins **600 mA**. Pour le **récepteur**, la plage de la tension d'alimentation est la même, mais le courant nécessaire pour son fonctionnement est beaucoup plus faible, de l'ordre de **100 mA** environ.

Nous vous recommandons de visiter le site **http://www.vk5dj.com/yagi.html**, qui propose de télécharger gratuitement un logiciel de calcul d'antennes Yagi, et bien plus encore. ■

DEMOBOARD- PIC18FXXX

Platine de développement pour PIC18F8XXX

Première partie

Suite aux nombreux courriers de nos lectrices et de nos lecteurs nous vous proposons dans cette série de 3 articles, une platine de développement dédiée à la famille très répandue des microcontrôleurs PIC18F8XXX. Elle se compose d'une carte de base contenant le microcontrôleur, l'interface de communication de type série, un étage de gestion de l'alimentation, et des cartes d'extensions de type afficheur LCD, clavier et module horloge temps réel (RTC) que nous étudierons par la suite. Toutes les lignes d'entrées/sorties de la carte de base sont accessibles par l'intermédiaire de borniers.

de Vincenzo Ligorio

Les microcontrôleurs de la famille des **PIC** de la firme **Microchip** ont toujours été appréciés des concepteurs de par leur facilité d'utilisation et de leur (re)programmation qui les distinguent des autres fabricants.

Ces microcontrôleurs sont devenus au fil du temps les plus largement utilisés dans la création d'applications simples mais aussi complexes, aussi bien par les amateurs que par les professionnels.

Depuis plusieurs années, la société Microchip a décidé de développer des microcontrôleurs performants et peu couteux visant ainsi le marché des amateurs dont les besoins sont de plus en plus importants.

Avec une quinzaine d'euros vous disposez d'une puissance de calcul, d'un nombre considérable d'I/O ainsi que des fonctions intégrées que bien des programmes spatiaux des années 60 auraient été contents de posséder.

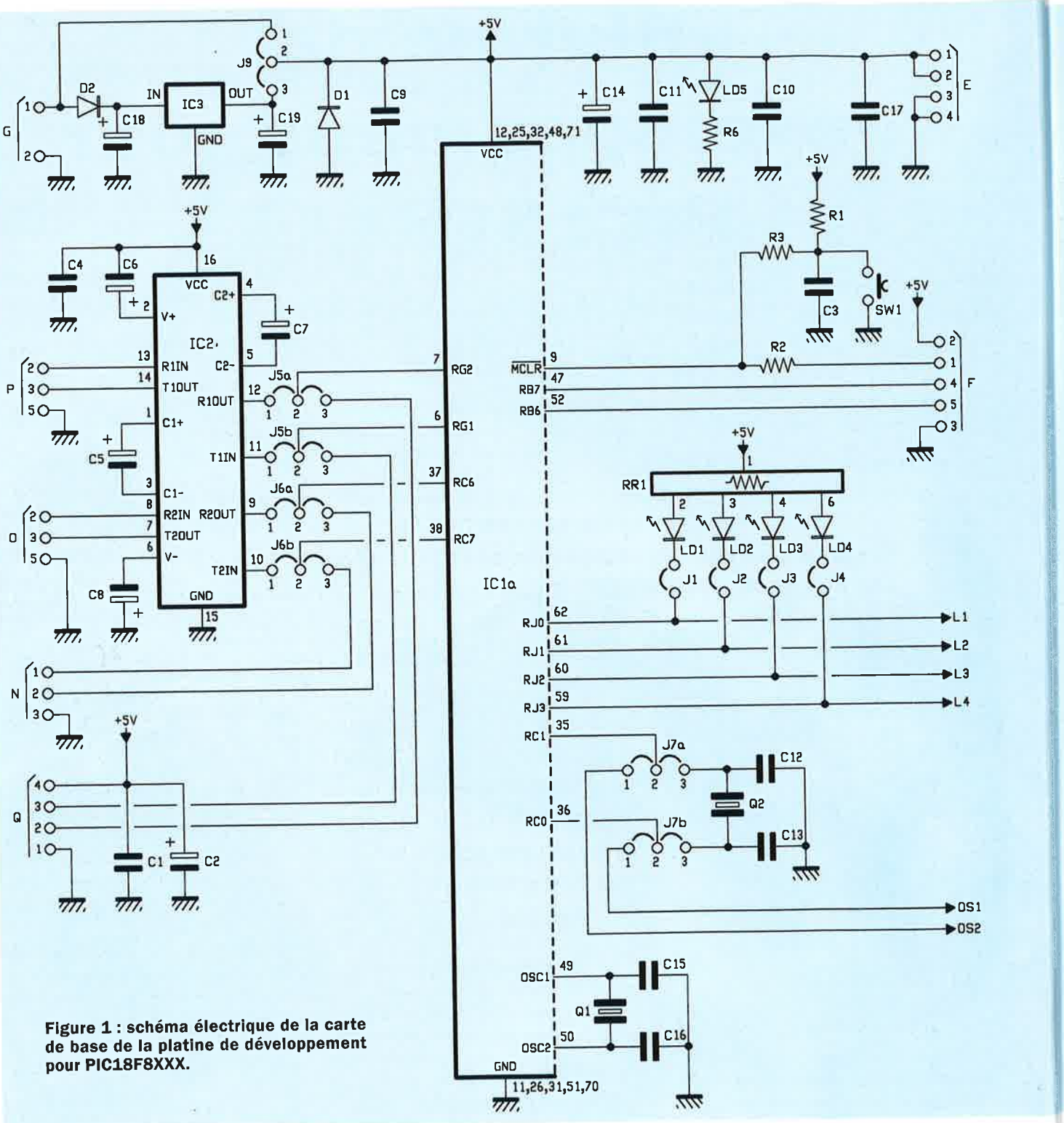


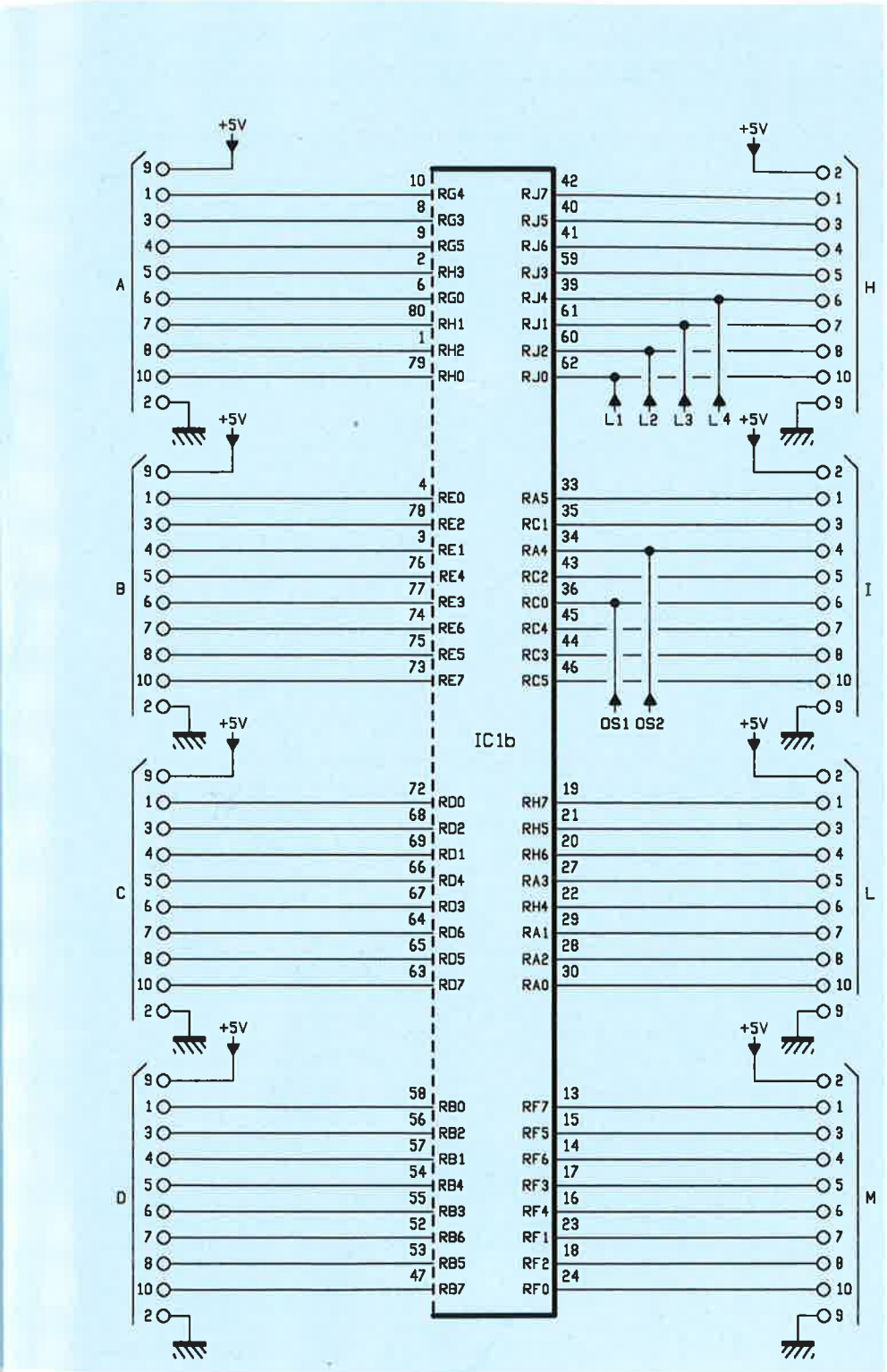
Figure 1 : schéma électrique de la carte de base de la platine de développement pour PIC18F8XXX.

Pour exemple l'ordinateur de guidage du LEM ou module lunaire des missions Apollo était composé d'une quantité de mémoire morte de 64 ko qui contenait l'ensemble des programmes, et de 4 ko de mémoire vive (effaçable) utilisée par les traitements. Les deux types de mémoires étaient constitués de tores magnétiques, le tout pesait environ 35 kg.

En comparaison, le **PIC18F8723** utilisé ici dispose de **128 Ko** de **mémoire flash** pour les programmes, de **4 Ko** de **RAM** et de **1 Ko** d' **EEPROM** et pèse seulement quelques grammes !

Le PIC18F8723 est disponible dans un boîtier TQFP de 80 broches (20 de chaque côté). Il dispose d'une **double UART**, de **16 canaux** pour la **conversion analogique/digitale** (A/D) dont la **résolution** est de **12 bits**, de **70 entrées/sorties** (I/O), de 128 ko de mémoire Flash (mots de 14 bits) extensible via une interface externe.

Avec une telle quantité de mémoire Flash, le programme peut fonctionner plus rapidement, car il accède à ses différentes routines (ou parties) qui se



situent dans la même page mémoire sans la nécessité de faire appel à des instructions contenues dans d'autres pages mémoires (et donc il s'affranchit des limites imposées par la segmentation de la mémoire).

Une augmentation supplémentaire de la vitesse de fonctionnement a été obtenue grâce à un **oscillateur interne** à **PLL** (« Phase Locked Loop » ou boucle à verrouillage de phase) qui, piloté de façon appropriée par un quartz externe, permet de faire fonctionner le microcontrôleur jusqu'à 40 MHz, assurant ainsi une puissance de traitement de 10 MIPS soit **10 millions d'instructions par seconde**. Il existe une **deuxième entrée d'oscillateur à 32 kHz**, elle peut être utilisée pour piloter une horloge temps réel. Si elle n'est pas utilisée, les deux broches, où normalement est relié un quartz de 32 kHz, peuvent être configurées comme une entrée ou une sortie.

Découvrant ainsi les grandes possibilités du PIC18F8723, qui est un microcontrôleur **haut de gamme** de la famille des **PIC18F**, nous avons décidé de concevoir et de proposer à nos lectrices et lecteurs une platine de développement pour un tel microcontrôleur (qui peut cependant être utilisée avec d'autres microcontrôleurs de la même famille).

Nous vous proposerons des exemples de programmes intéressants qui exploiteront toute une série de routines réutilisables pour diverses applications, cela permettra à chacun d'entre vous de réaliser des fonctions spécifiques pour vos besoins. Il suffira tout simplement d'effectuer quelques modifications simples pour adapter vos routines.

Cependant nous aborderons la programmation et les exemples de routines dans la deuxième et troisième partie.

Dans cette première partie nous allons étudier le matériel, qui consiste, comme nous l'avons mentionné, en une unité centrale et des cartes d'extensions qui viendront se connecter à la carte de base.

Cette dernière correspond à la carte principale (comme une carte mère de PC) dont son rôle est essentiellement de rendre accessible de l'extérieur toutes les lignes d'entrées/sorties et les périphériques intégrés du microcontrôleur.

Les cartes d'extensions sont des circuits relativement simples qui se connectent aux différents ports grâce aux borniers présents sur la carte de base.

Ces extensions sont très utiles dans la pratique, car elles permettent de vérifier rapidement les fonctionnalités de certaines applications spécifiques.

Les principales cartes d'extensions qui s'interfacent avec la carte de base contenant le PIC sont :

Plan de montage de la carte de base

La carte principale de base déporte toutes les broches du microcontrôleur vers plusieurs borniers externes, de manière à rendre extrêmement facile le câblage d'un prototype.

Les borniers et les cavaliers rendent inutile l'utilisation d'un fer à souder pour connecter la carte de développement à d'autres cartes ou prototypes de tests.

Pendant la phase de montage de la platine de développement, il faut veiller à ne pas détruire le microcontrôleur en le surchauffant, il doit être soudé avec un fer de 25 W et une pointe de 0,4 mm ainsi qu'avec de la soudure pour composants CMS.

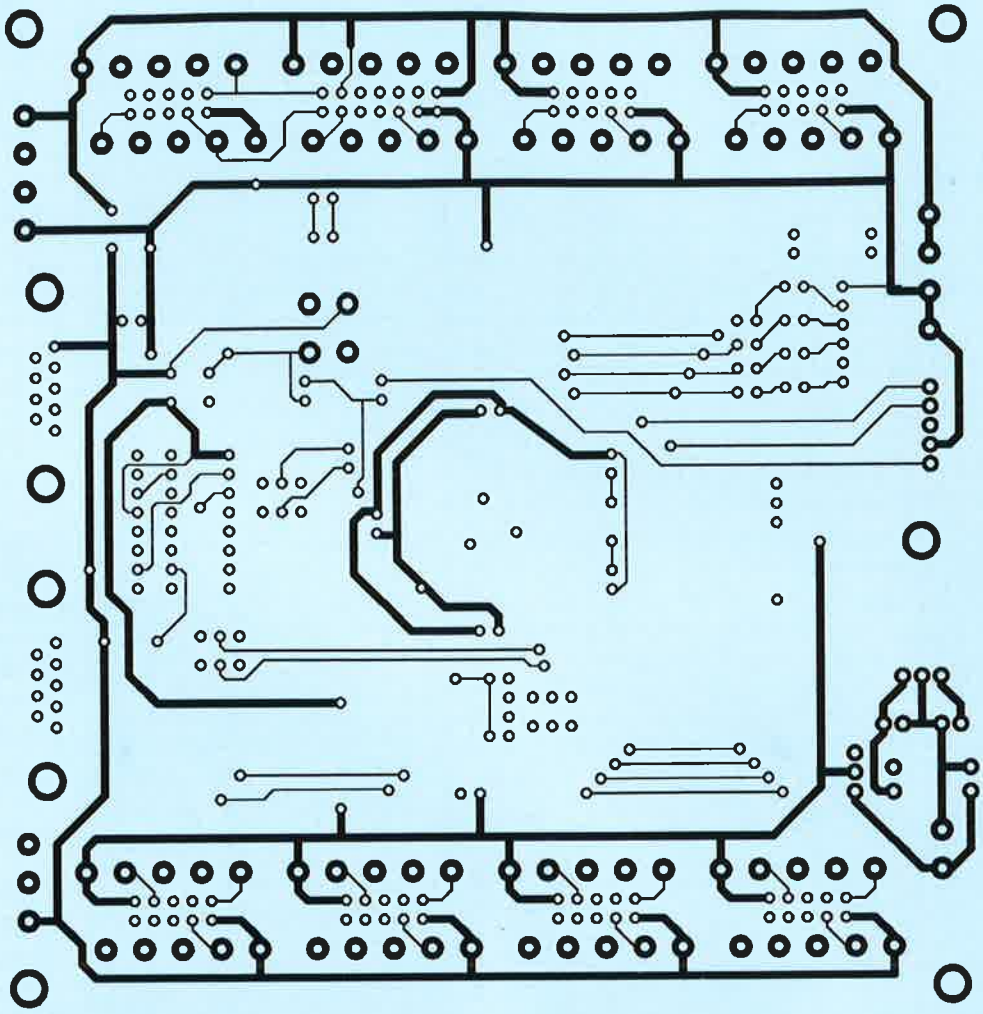


Figure 2 : circuit imprimé à l'échelle 1 : 1 côté soudures de la carte de base de la platine de développement pour PIC18F8XXX.

Liste des composants de la carte de base de la platine de développement pour PIC18F8XXX.

R1.....10 kΩ
R2.....160 Ω
R3.....1 kΩ
R4.....10 kΩ
R5.....10 kΩ
R6.....470 Ω

RR1....réseau de résistances 6 x 470 Ω + C

C1.....100 nF multicouche
C2.....22 µF/25 V électrolytique
C3.....100 nF multicouche
C4.....100 nF multicouche
C5.....10 µF/25 V électrolytique
C6.....10 µF/25 V électrolytique
C7.....10 µF/25 V électrolytique
C8.....10 µF/25 V électrolytique
C9.....100 nF multicouche
C10....100 nF multicouche
C11....100 nF multicouche
C12....22 pF céramique
C13....22 pF céramique
C14....22 µF/25 V électrolytique
C15....22 pF céramique
C16....22 pF céramique
C17....100 nF multicouche
C18....22 µF/35 V électrolytique
C19....22 µF/35 V électrolytique
IC1.....PIC18F8723-I/PT
IC2.....MAX232
IC3.....7805
D1.....1N4007

un module horloge temps réel (RTC), un afficheur à cristaux liquides et une carte gérant l'alimentation, utile pour des applications où il est nécessaire que le PIC puisse se mettre en veille afin de consommer le minimum d'énergie (cas des applications sur piles ou accus). Enfin nous avons développé une carte d'extension avec 8 boutons poussoirs assignables par logiciel à certaines lignes d'entrées.

La carte principale de base

Dans un premier temps, nous allons analyser la carte principale qui est la base du système de développement, ensuite nous étudierons les cartes d'extensions.

La carte de base, dont vous pouvez voir le schéma électrique en figure 1, contient donc le microcontrôleur ainsi que quelques composants permettant le fonctionnement du PIC. **Toutes les lignes d'entrées/sorties du PIC**

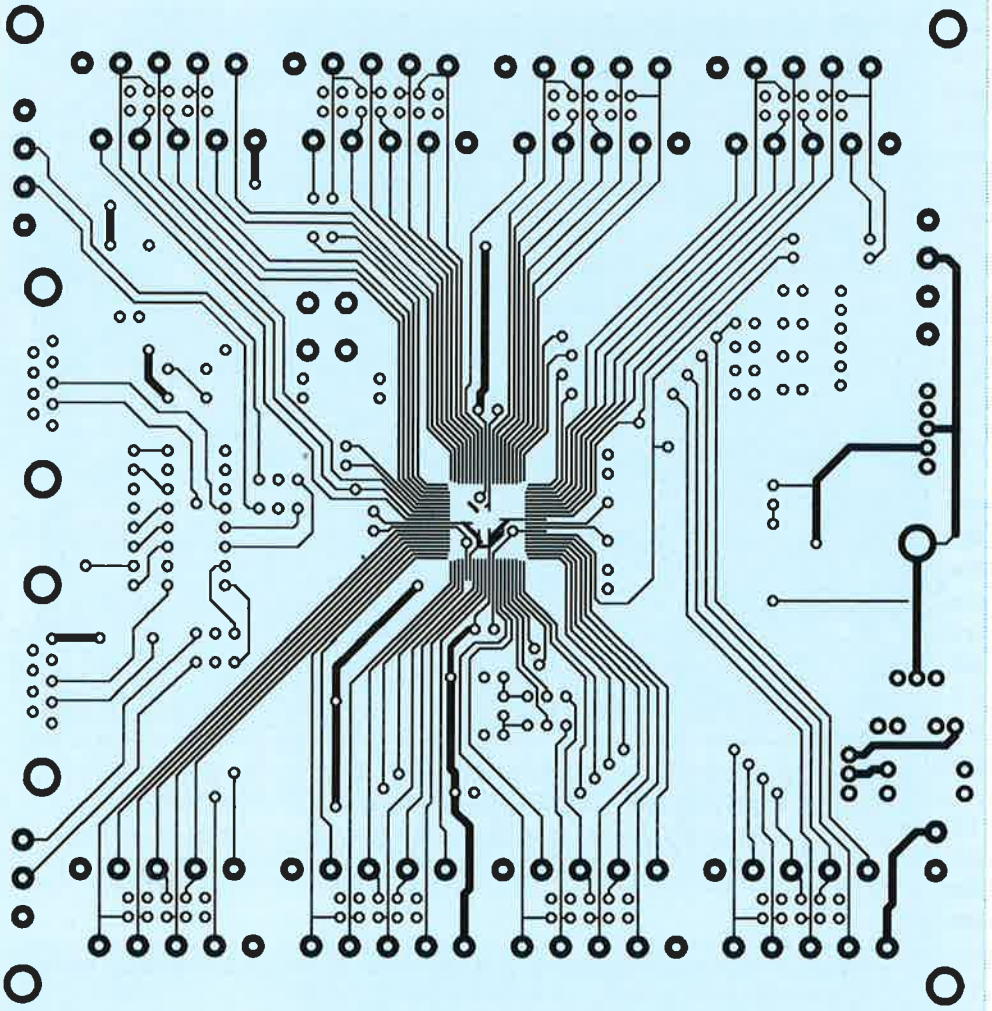


Figure 3 : circuit imprimé à l'échelle 1 : 1 côté composants de la carte de base de la platine de développement pour PIC18F8XXX.

D2..... 1N4007
LD1.... LED 3 mm rouge
LD2.... LED 3 mm rouge
LD3.... LED 3 mm rouge
LD4.... LED 3 mm rouge
LD5.... LED 3 mm verte
SW1... microswitch
Q1..... quartz 10 MHz
Q2..... quartz 32,768 kHz

Divers

Support ci 2 x 8 broches

Bornier double niveau 2 + 2 pôles (x 8)
Bornier double niveau 3 + 3 pôles (x 8)
Bornier 2 pôles (x 5)
Connecteur DB9 femelle pour ci
Barrette mâle 2 pôles (x 4)
Barrette mâle 3 pôles (x 7)
Barrette femelle 5 pôles
Cavaliers (x 11)
Dissipateur ML26
Vis 3 MA 10 mm
Ecrous 3 MA

sont reportées vers des borniers. La plupart de ces lignes (ou broches) peuvent être configurées au moyen de cavaliers en fonction de vos besoins, afin qu'elles puissent être utilisées avec vos applications.

Par exemple, si vous utilisez l'horloge interne du PIC, les broches RC0 (35) et RC1 (36), à l'aide du double cavalier J7, peuvent être dirigées vers l'extérieur de la carte afin d'être utilisées comme des lignes communes d'entrées/sorties.

Sinon à l'aide de J7, vous pouvez les relier à un quartz si vous utilisez une horloge externe.

Notez que les broches RC7 (38), RC6 (37), RG1 (6) et RG2 (7) qui correspondent aux lignes de communication série sont reliées aux 2 UART internes.

Plus précisément, RG1 correspond à RX et RG2 à TX du 1er UART, tandis que RC6 correspond à RX et RC7 à TX du 2ème UART.

Lorsque vous utilisez les ports de communication, vous devez fermer chacun des deux cavaliers J5a, J5b et J6a, J6b sur les contacts 1 et 2, de manière à relier les UART du microcontrôleur au convertisseur TTL/RS232 IC2 (MAX232).

Sinon dans le cas contraire, en reliant les deux cavaliers J5a, J5b et J6a, J6b sur les contacts 2 et 3, les broches RC7, RC6, RG1 et RG2 sont dirigées vers les borniers N et Q enfin d'être utilisées en tant qu'entrées/sorties.

Dans cette configuration, ces broches peuvent servir aussi de lignes communication série pour gérer des périphériques, mais seulement avec un niveau TTL.

Sur la carte de base, en plus du microcontrôleur, d'autres composants sont présents afin d'améliorer l'interface avec le PIC et de fournir l'alimentation.

Commençons par le **convertisseur TTL/RS232**, il s'agit d'un double convertisseur intégré de type **MAX232** contenant 2 sections qui convertissent des signaux au niveau TTL vers des signaux au niveau RS232 et vice-versa (RS232 vers TTL).

Il permet, par exemple, de **connecter la carte de développement à un périphérique équipé d'un port de communication respectant la norme RS232**, cela peut être un ordinateur.

Nous le verrons plus loin, ce convertisseur permet, grâce à un programme de type « terminal », de dialoguer directement avec le PIC afin d'exécuter des

procédures comme par exemple un débogage (via le port série).

Les lignes RC1, RJ0, RJ1, RJ2 et RJ3 peuvent être connectées à des résistances externes de pull-up (de tirage) constituées par un réseau de résistances « RR1 » à l'aide des cavaliers J1, J2, J3 et J4.

Nous n'avons pas prévu ce dispositif pour les autres entrées/sorties, car elles disposent de résistances internes de pull-up programmables par logiciel.

Nous avons doté le microcontrôleur de la **programmation en circuit** (**ICSP**) car il est soudé sur le circuit imprimé. Dans ce cas vous devez utiliser un **programmateur de type IC3D** de **Microchip** ou tout autre programmateur compatible ICSP.

Vous devez **relier le programmateur au connecteur « F »** (réservé pour la programmation). La ligne **MCLR** est connectée à la broche 1 du connecteur « F » et est utilisée comme **remise à zéro du PIC**. Le **reset** peut être effectué manuellement à l'aide du poussoir **SW1**.

L'alimentation électrique de la carte de base est appliquée au bornier « **G** ». Si le cavalier **J9** est en position « **1-2** », vous devez appliquer une source de **tension continue stabilisée de 5 VDC** avec un courant d'au moins 200 mA.

Attention dans ce cas **vous ne devez pas appliquer une tension supérieure à 5,5 VDC qui est la limite acceptable du PIC sous peine de destruction.**

Si vous disposez d'une source d'alimentation plus élevée et pas forcément stabilisée, par exemple 9 V, positionnez le cavalier **J9** dans la position « **2-3** ».

Dans ce cas **la tension est délivrée par le régulateur IC3** (7805) qui fournira le **5 VDC** à l'ensemble de la carte.

La diode **D2 protège contre une inversion de polarité**. Lorsque le cavalier **J9** est dans la position « **2-3** », vous devez appliquer sur le bornier « **G** » une tension **d'au moins 8,5 VDC** car il faut au moins une différence de potentiel (dropout) de **3 V** entre l'entrée et la sortie du régulateur.

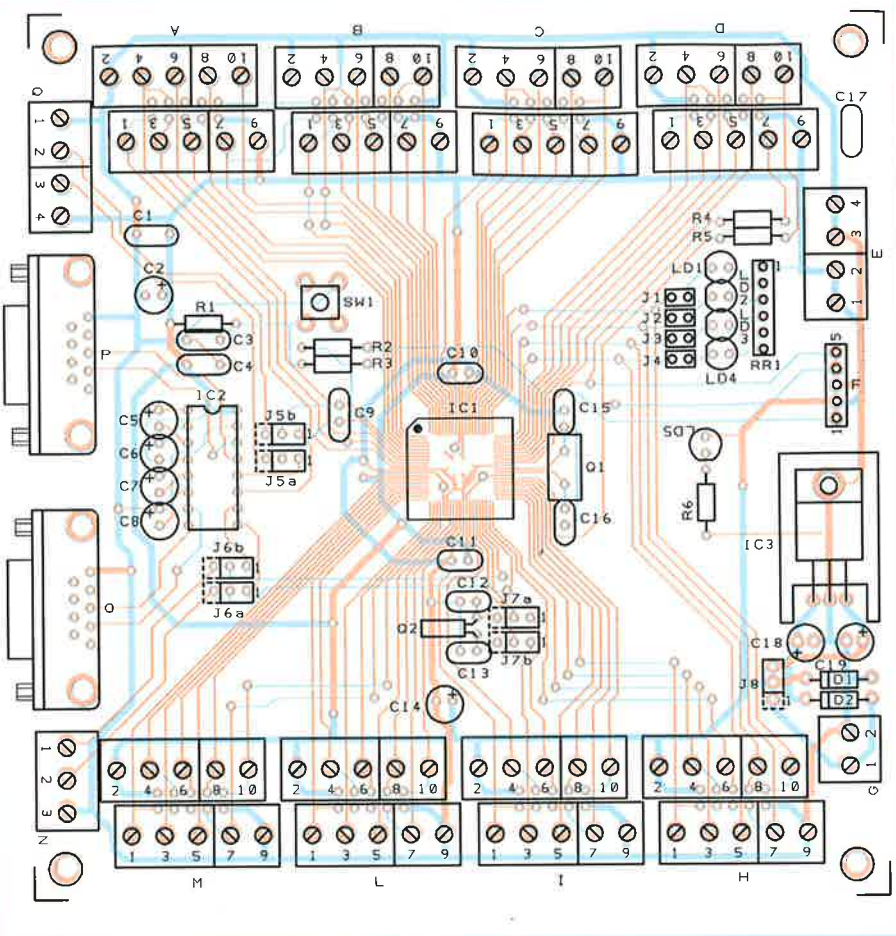


Figure 4 : schéma de câblage des composants de la carte de base de la platine de développement pour PIC18F8XXX.

Les cartes d'extension

Pour permettre la simulation de différentes applications, nous avons doté la carte principale de base de modules supplémentaires (ou cartes d'extension) afin d'obtenir certaines fonctions non indispensables, mais que certains lecteurs auront besoin de mettre en œuvre.

Ces modules supplémentaires sont les suivants : un **module d'allumage/extinction** (contrôle de l'alimentation), un **module horloge temps réel RTC** qui peut être combiné avec le module contrôle de l'alimentation, un module comportant un **clavier**, un **module d'affichage** à cristaux liquides alphanumérique.

Le module d'alimentation (allumage/extinction)

C'est un petit circuit contenant une logique très simple qui commande un **transistor MOSFET de commutation** et un **régulateur de type 7805**.

Ce module est généralement **connecté en amont** du bornier « **G** » de la carte de base. Dans ce cas, vous devez **fermer le cavalier J9** dans la position « **1-2** » car la tension d'alimentation est de 5 VDC stabilisés.

Ce module peut être utilisé lorsque vous voulez **activer/désactiver le PIC à l'aide de signaux logiques** compatibles TTL.

Par exemple, avec ce module vous pouvez tester une routine qui effectue l'arrêt et la mise en fonction automatiques du PIC grâce à un contrôle externe.

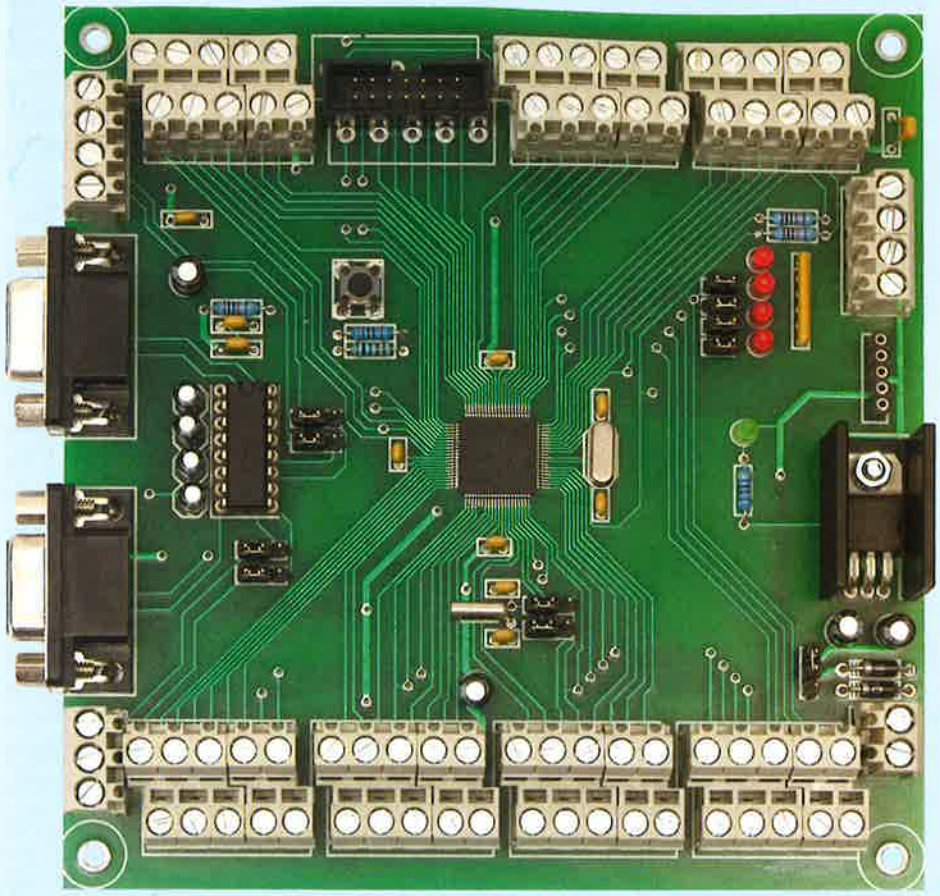


Figure 5 : photo de l'un de nos prototypes de la carte de base de la platine de développement pour PIC18F8XXX.

Le courant consommé par la logique de ce module est de quelques centaines de µA, le reste sert à la carte de base qui consomme environ 200 mA.

Dans les conditions de repos, les deux portes **NAND** « **IC1a** » et « **IC1b** » (broches 1, 2, 3 et 4, 5, 6) réalisent une **bascule bistable** dont les sorties sont respectivement à un niveau **1** et **0** logique. Un tel état correspond à « **éteint** », en appuyant sur le bouton OFF (SW1) la broche 1 se retrouve à un niveau logique 1 et 0.

Au contraire, en appuyant sur ON (SW2) ou en envoyant une impulsion d'un niveau logique 0 sur la broche 6, le circuit « s'allume ».

En condition de repos, les broches 12 et 13 de « **IC1d** » sont à un état logique 0, donc la broche 11 est à un niveau logique 1.

Le transistor **MOSFET T1**, qui est de type « canal P », se retrouve dans un état **bloqué** et aucun courant ne circule entre le drain et la source.

Nous retrouvons simplement la tension d'alimentation sur la cathode

Ces types de routines sont beaucoup utilisées dans la pratique, elles sont destinées à économiser l'énergie de systèmes fonctionnant sur batteries.

De par sa structure, ce module peut commander directement le microcontrôleur à l'aide d'une ligne qui doit être mise à un niveau logique 0, soit manuellement à l'aide des 2 boutons « **ON** » et « **OFF** » soit par une impulsion provenant d'un dispositif électronique externe.

En observant le schéma électrique visible en figure 7 du module d'allumage/extinction, une tension continue d'au moins 8,5 VDC doit être appliquée sur la broche 2 du bornier « CON » (de préférence stabilisée).

Cette tension sera utilisée pour alimenter à la fois la logique de commande du module et la carte de base principale du système de développement.

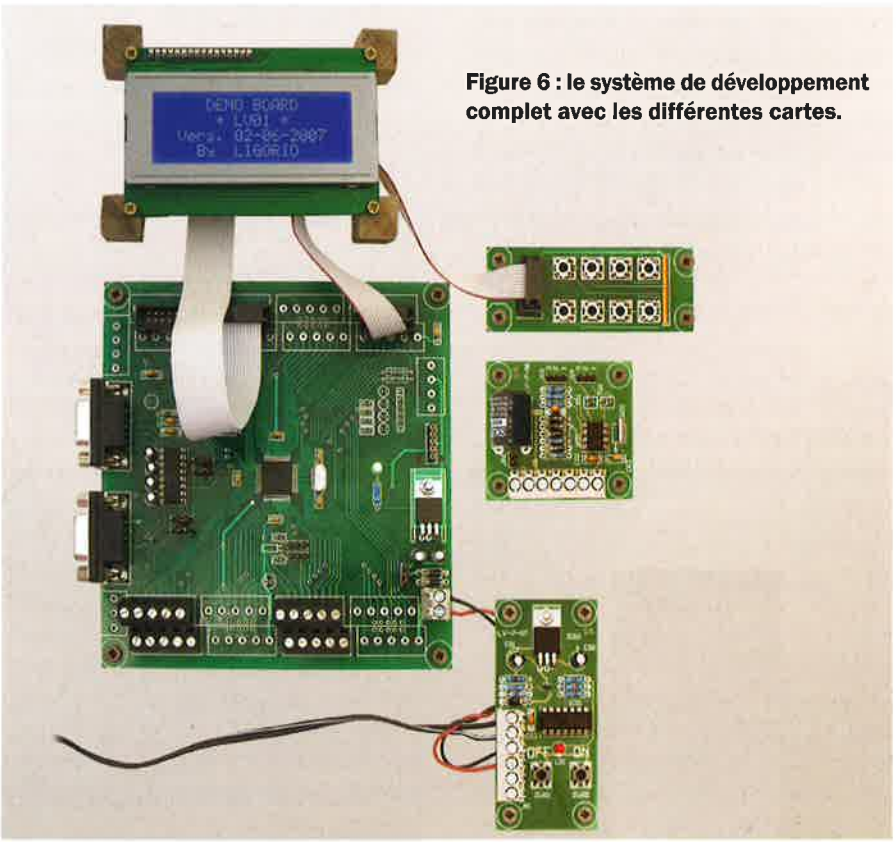


Figure 6 : le système de développement complet avec les différentes cartes.

Le module d'alimentation

Le module d'alimentation vous permet d'activer et désactiver la carte de base, soit manuellement au moyen des 2 boutons poussoirs SW1 et SW2, soit par une commande électronique provenant d'un circuit externe (comme le module RTC) connecté aux broches 5 et 6 du bornier « CON ». La bascule bistable composée par les portes « IC1a » et « IC1b » garantit l'allumage et l'extinction sans condition d'incertitude.

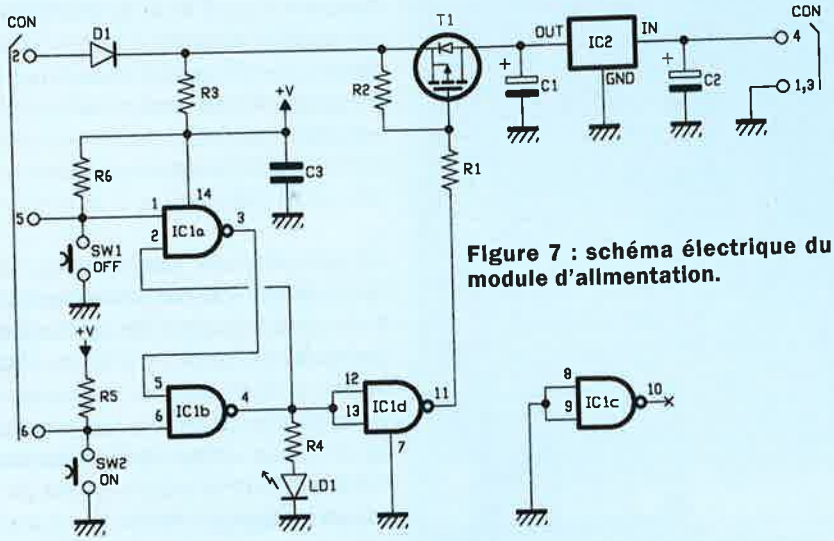


Figure 7 : schéma électrique du module d'alimentation.

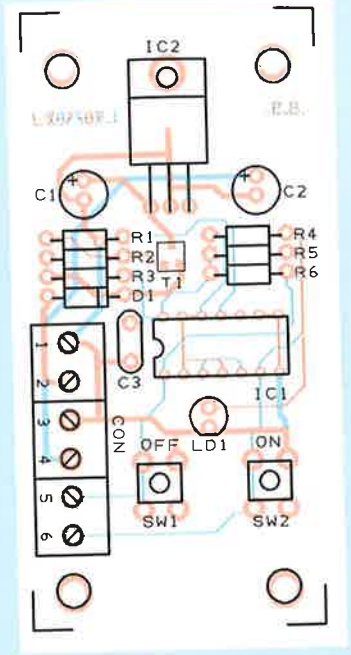


Figure 10 : schéma de câblage des composants du module d'alimentation.

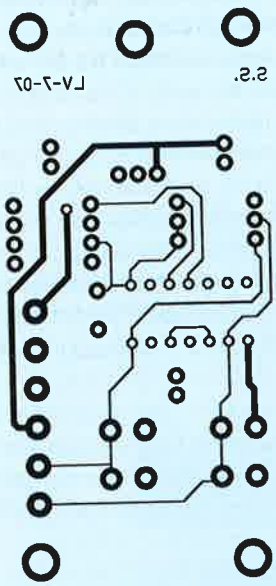


Figure 8 : circuit imprimé à l'échelle 1 : 1 côté soudures du module d'alimentation.

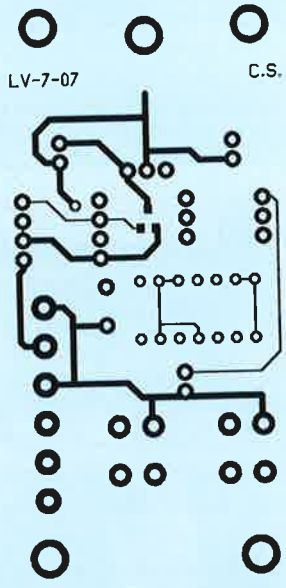


Figure 9 : circuit imprimé à l'échelle 1 : 1 côté composants du module d'alimentation.

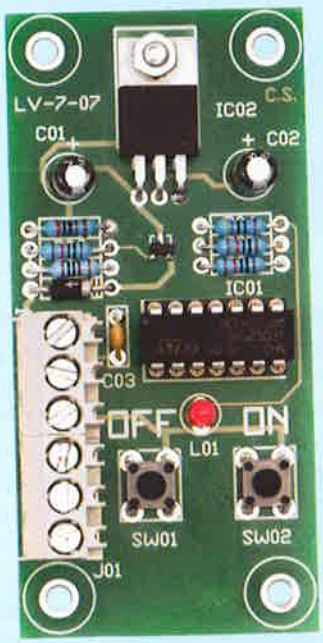


Figure 11 : photo de l'un de nos prototypes du module d'alimentation.

Liste des composants du module d'alimentation

R1.....10 kΩ 1 %
R2.....10 kΩ 1 %
R3.....1 kΩ 1%
R4.....499 Ω 1%
R5.....10 kΩ 1 %
R6.....10 kΩ 1 %

C1.....22 µF/25 V électrolytique
C2.....22 µF/25 V électrolytique
C3.....100 nF multicouche

1 .....1N4004
T1 .....IRLML6402
IC1.....CD4093

IC2.....7805

SW1...microswitch
SW2...microswitch
LD1....LED 3 mm rouge

Divers
Bornier 2 pôle (x 3)
Support ci 2 x 7 broches

de la diode D1 (protège le module contre une inversion de polarité) qui sert à alimenter les portes logiques du module. Le régulateur **IC2 n'est pas alimenté** et donc par conséquent le PIC est éteint.

Si la broche 6 de « IC1b » reçoit une impulsion d'un niveau logique 0 ou si le bouton ON est pressé, la sortie de la porte NAND « IC1b » (broche 4) passe à un niveau logique 1.

Ce 1 logique se retrouve sur la broche 2 de « IC1a », cependant la broche 1 est forcée à un niveau 1 à cause de R6 et donc la sortie de « IC1a » (broche 3) est à un 0 logique, par conséquent la bascule est bloquée dans la situation actuelle.

Le niveau logique 1 sur la broche 4 impose un niveau logique 0 sur la broche 11 de « IC1d », le transistor **MOSFET T1 devient conducteur** et **alimente le régulateur IC2** jusqu'à ce que le bouton « OFF » soit pressé ou que la broche 5 du bornier reçoive une impulsion à niveau logique 0.

Tant que le régulateur IC2 est alimenté, le module fournit une tension de 5 VDC stabilisée par l'intermédiaire des broches 4 (positif) et 1,3 (masse) à la carte de base et donc le microcontrôleur est alimenté.

Cette condition est signalée par la LED LD1 polarisée par la résistance R4 qui est mise à un niveau logique 1 par la sortie (broche 4) de IC1b.

Comme nous l'avons mentionné précédemment, le module d'alimentation peut être utilisé pour tester des routines qui permettent l'arrêt automatique du PIC.

Vous devez configurer une ligne du PIC en sortie et lorsque vous désirez l'éteindre, vous reliez cette ligne à la broche 5 du bornier « CON » du module d'alimentation et la mettez à un niveau logique 0.

Pour rallumer le PIC, vous devez envoyer une impulsion d'un niveau logique 0 sur la broche 6 du bornier « CON ». Il est évidemment impossible de demander cela au PIC car il est éteint, l'impulsion doit venir d'un système externe ou le bouton SW2 doit être pressé.

Habituellement, pour « réveiller » un microcontrôleur, lorsqu'il est en veille, lors de l'arrivée d'un événement tel que, par exemple, l'apparition d'une tension, l'arrivée d'un appel téléphonique ou à l'expiration d'un certain intervalle de temps nous utilisons généralement un module horloge temps réel que nous allons voir maintenant.

Le module RTC

Nous avons prévu comme carte d'extension, un **module horloge temps réel** ou **RTC** (Real Time Clock), qui peut être utilisé pour **gérer l'horloge du système pour des applications nécessitant une planification ou un calendrier**, afin d'activer le microcontrôleur de manière automatique lorsque cela est nécessaire.

Par exemple dans le cas d'un système d'arrosage automatique hebdomadaire, ou lorsque que vous devez gérer des alarmes à distance via GSM.

Comme vous pouvez le voir sur le schéma électrique du module RTC en figure 12, il dispose d'un bornier « CON » avec 6 broches permettant de le relier à la carte principale de base du système de développement, ou à d'autres circuits.

Le cœur du module RTC est le circuit **IC1** (**PCF8563**) qui est un circuit intégré encapsulé dans un boîtier ayant 4 broches de chaque côté.

Il fonctionne avec un quartz dont la fréquence est de **32,768 kHz** connecté entre les broches 1 et 2 et avec un condensateur relié entre la broche 1 et la masse. Le **PCF8563** se gère par l'intermédiaire du **bus I2C** via les broches 6 (**SDA**) et 7 (**SCL**) du bornier (respectivement broches 5 et 6 du composant).

Il dispose aussi d'une sortie d'interruption « **INT** » (broche 3) liée à un événement. Cette sortie dans notre circuit est reliée soit directement à la broche 5 du bornier lorsque le cavalier J3 est entre les points 2 et 3, soit à travers le **couple de transistors** constitué par **T1** et **T2** (J3 entre 1 et 2) montés en **amplificateur de courant**.

Il est aussi possible de doter la sortie 5 du bornier d'une **résistance de pull-up R5** grâce au cavalier J2 (points 1 et 2).

Notez que l'amplificateur de courant constitué par les transistors T1 et T2 **n'inverse pas la condition logique** de la broche « **INT** » (3).

Il permet un courant de sortie plus important mais également une tension de sortie beaucoup plus grande de 8,5 V à 15 V par rapport aux 5 V du niveau logique de la sortie « INT ». Cela permet de piloter un relais ou d'autres dispositifs avec un courant maximal de 50 mA.

La particularité de ce montage est qu'il ne consomme pas de courant lorsque la sortie 5 du bornier est au repos.

En effet si la broche « INT » (3) est à un niveau logique 1, T1 et T2 sont bloqués, aucun courant ne circule dans les résistances de polarisation.

Afin d'éviter la perte de l'heure et de la date en cas de panne de courant, une **batterie de secours** pour circuit imprimé de 3,6 V (65 mA/h) est prévue. Elle sert également à maintenir le fonctionnement du module lorsque l'alimentation principale est interrompue en même temps que celle du PIC.

Le module RTC fonctionne avec une tension de **5 VDC** stabilisée (avec au moins 10 mA) appliquée entre les points 4 (positif) et les points 1 et 3 (masse) du bornier.

Dans ce cas, la **batterie est rechargée à travers R2 et D2** (cette dernière évite que, lors d'une coupure d'alimentation au point 4, la batterie se décharge à travers R2), tandis que D4 fournit la tension au reste du circuit.

En cas de coupure de courant, la batterie (si J1 est fermé) alimente le reste du module à travers la diode D3. Celle-ci en fonctionnement normal, c'est-à-dire en présence de l'alimentation principale de 5 V, ne conduit pas et laisse le

Le module RTC

Le circuit intégré PCF8563 (IC1) est géré par la carte de base via les broches 6 et 7 (BUS I2C) du bornier. La batterie de secours est dotée de son propre circuit de charge et assure la sauvegarde de l'heure et de la date lors d'une coupure d'alimentation. La sortie « INT » (broche 5 du bornier) peut être utilisée pour piloter un circuit externe (comme le module d'alimentation) soit directement, soit par l'intermédiaire de l'amplificateur de courant constitué par T1 et T2, en fonction des positions des cavaliers J2 et J3.

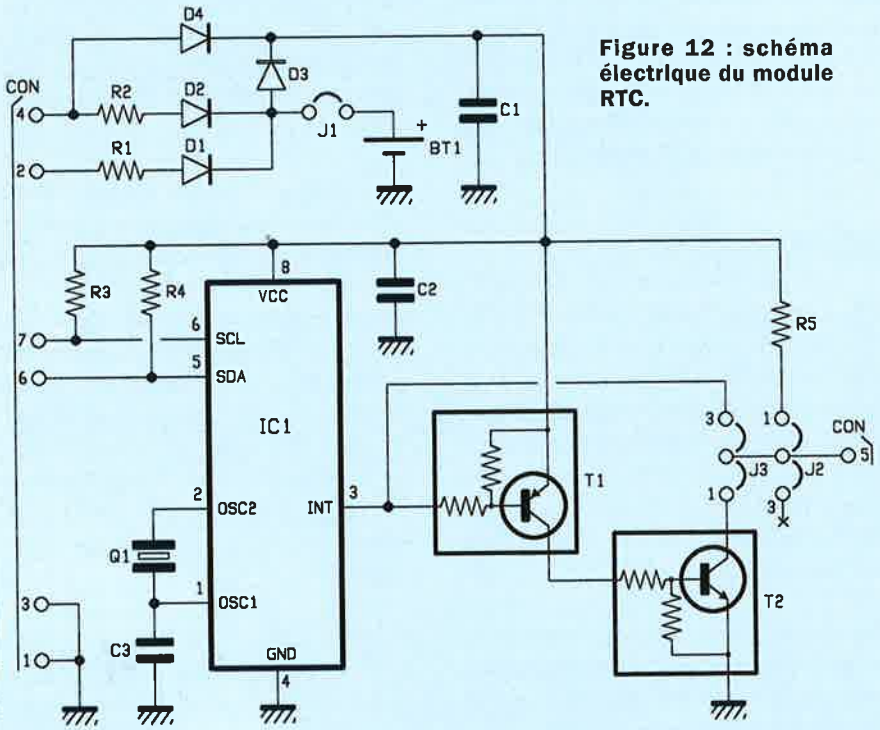


Figure 12 : schéma électrique du module RTC.

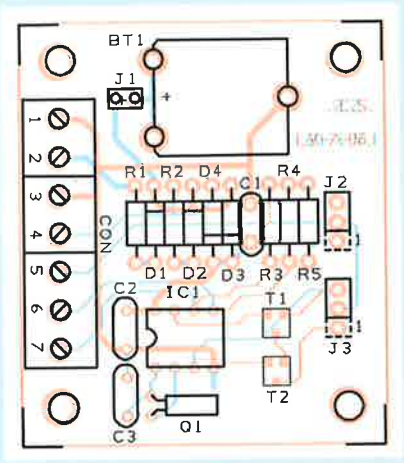


Figure 15 : schéma de câblage des composants du module RTC.

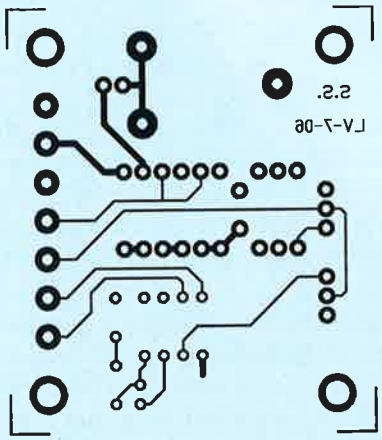


Figure 13 : circuit imprimé à l'échelle 1 : 1 côté soudures du module RTC.

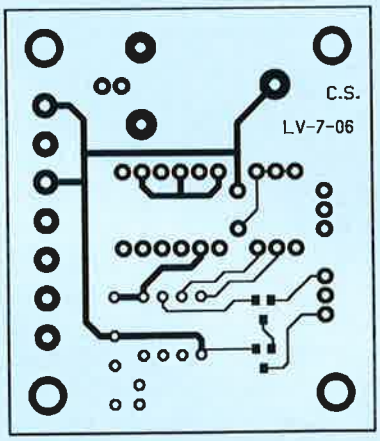


Figure 14 : circuit imprimé à l'échelle 1 : 1 côté composants du module RTC.

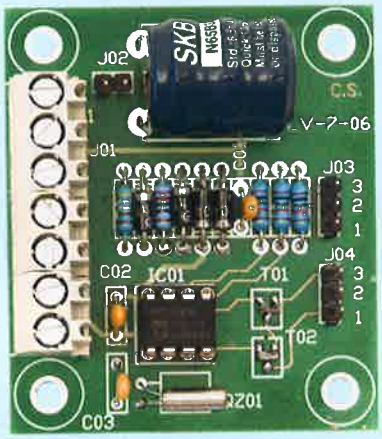


Figure 16 : photo de l'un de nos prototypes du module RTC.

Liste des composants du module RTC

R1.....4,99 kΩ 1%
R2.....220 Ω 1%
R3.....10 kΩ 1%
R4.....10 kΩ 1%
R5.....10 kΩ 1%
C1.....100 nF multicouche
C2.....100 nF multicouche
C3.....22 pF céramique
D1.....1N4004
D2.....1N4004
D3.....1N4004
D4.....1N4004
T1.....PDTA144ET
T2.....PDTC144ET
Q1.....quartz 32,768 kHz
IC1.....PCF8563P

BT1....batterie 3,6 V/65 mA

Divers

Bornier 2 pôles (x 2)
Bornier 3 pôles
Barrette mâle 2 pôles
Barrette mâle 3 pôles (x 2)
Cavaliers (x 3)
Support ci 2 x 4 broches

courant de charge de la batterie circuler à travers le réseau R2/D2.

Notez qu'il y a une 2ème façon de charger la batterie, par l'intermédiaire du point 2 du bornier « CON ».

Nous avons prévu cette possibilité pour des applications particulières dans lesquelles le module RTC est alimenté conjointement avec le microcontrôleur de la carte de base via le module d'alimentation. Dans ce cas, lors des phases d'allumage/extinction la batterie aura du mal à se charger correctement. Il est donc opportun de prévoir un système de charge continu à l'aide de R1 et D1. En effet la broche 2 doit être en permanence soumise à une tension de 5 V par rapport à la masse (broches 1 et 3 du bornier « CON »).

Le module afficheur LCD

Etudions maintenant le **module afficheur LCD**, dont vous pouvez voir le schéma électrique en figure 17. Il s'agit d'un afficheur ayant une interface classique pilotée par un **contrôleur HD44780 Hitachi** ou compatible.

L'afficheur peut-être **d'une ligne avec 16 caractères**, mais aussi de **2 lignes de 20 caractères** ou encore 4 x 20, 4 x 16, etc. Le module afficheur LCD se compose d'un adaptateur sous forme de circuit imprimé.

Les deux résistances **R1** et **R2 déterminent le contraste** tandis que **R3 limite le courant du rétroéclairage** de l'afficheur. Toutes les connexions du module LCD sont disponibles sur le bornier « CON ».

L'interface de communication prévoit l'utilisation des **8 bits de données** de

Le module afficheur LCD

Il s'agit d'un adaptateur matériel ne contenant que des composants passifs. Il permet facilement l'interfaçage des afficheurs alphanumériques communs comme par exemple un classique 16 caractères de 2 lignes.

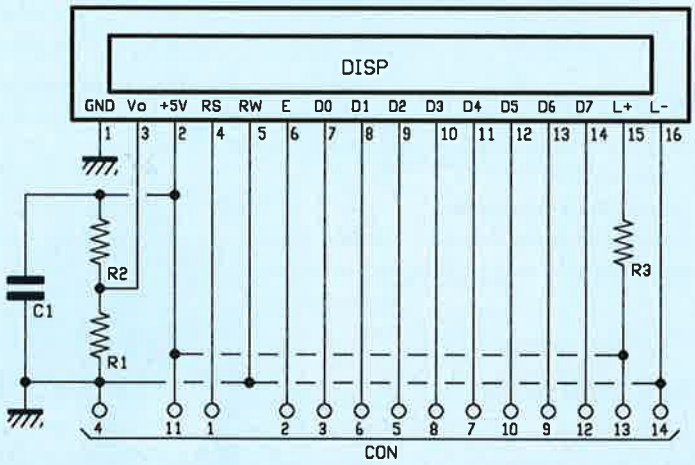


Figure 17 : schéma électrique du module afficheur LCD.

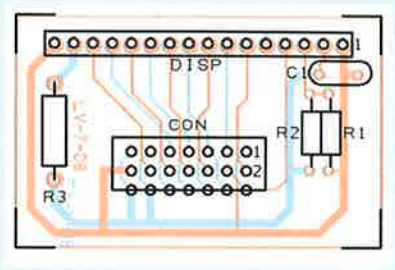


Figure 20 : schéma de câblage des composants du module afficheur LCD.

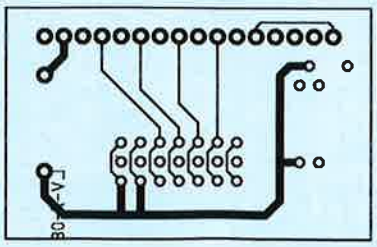


Figure 18 : circuit imprimé à l'échelle 1 : 1 côté soudures du module afficheur LCD.

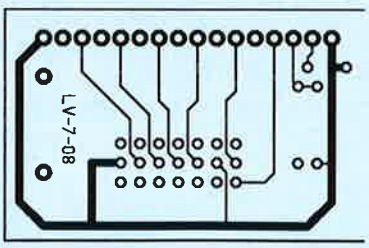


Figure 19 : circuit imprimé à l'échelle 1 : 1 côté composants du module afficheur LCD.

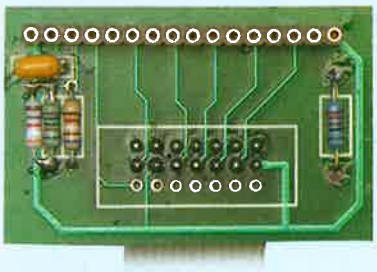


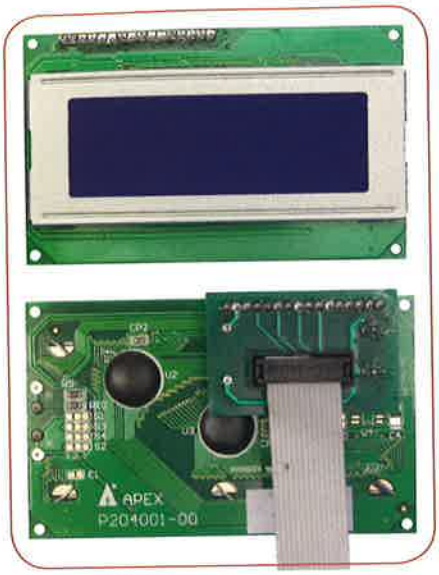
Figure 21 : photo de l'un de nos prototypes du module afficheur LCD.

Liste des composants du module afficheur LCD

R1.....3,92 kΩ 1%
R2.....10 kΩ 1%
R3.....221 Ω 1% 1/2W
C1.....100 nF multicouche
DISP...afficheur LCD 16 x2

Divers

Barrette femelle 16 pôles
Barrette mâle 16 pôles
Barrette mâle 7 pôles (x 2)



l'afficheur (**D0** à **D7**), configuration un peu inhabituelle. En effet la plupart du temps dans les montages que nous décrivons dans la revue, les afficheurs LCD sont gérés uniquement avec 4 bits de données (D4 à D7).

Nous avons opté, dans ce cas, pour cette configuration afin de donner la possibilité à chacun d'entre vous de choisir le mode de données le mieux adapté à vos applications.

Rappelez-vous, en effet, que la gestion d'un afficheur sur 8 bits mobilise 4 broches supplémentaires du microcontrôleur, mais réduit de moitié le temps de lecture et d'écriture.

En plus des lignes de données et de commandes disponibles sur le bornier, vous avez accès aux deux lignes de commande « **E** » et « **RS** ». La broche « **R/W** » est reliée à la **masse**, car dans notre cas **nous voulons simplement afficher** (écrire) quelque chose sur l'écran (mode « Write ») et non pas lire l'état du tampon de l'afficheur (mode « Read »).

Rappelez-vous que la broche « R/W » permet d'envoyer ou de recevoir des données vers/de l'afficheur. En d'autres termes, le contrôleur HD44780 peut recevoir des données correspondant aux caractères à afficher mais peut aussi envoyer des informations sur son état.

La ligne « **RS** » est utilisée pour indiquer à l'afficheur si les données

Le clavier

Notre clavier contient 8 boutons poussoirs afin d'être géré sur un octet entier ou un port complet du microcontrôleur. La connexion à la carte base est réalisée à l'aide d'un câble plat.

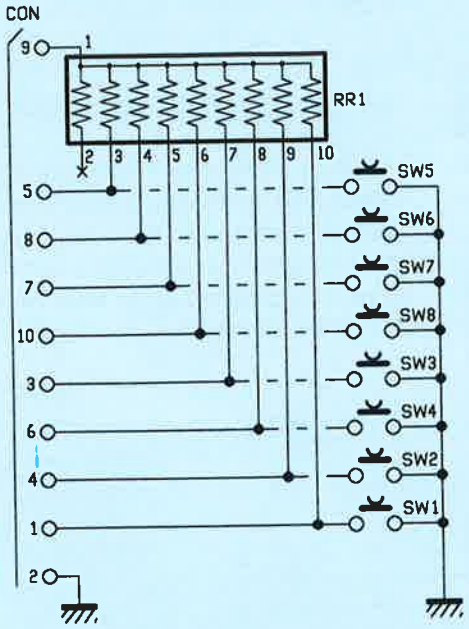


Figure 22 : schéma électrique du clavier.

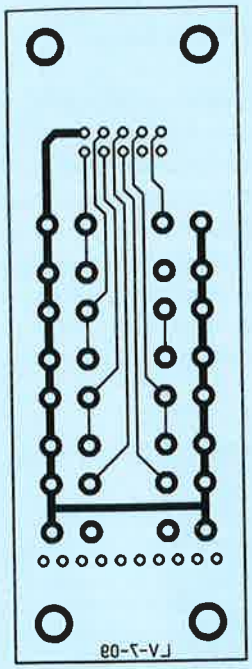


Figure 23 : circuit imprimé à l'échelle 1 : 1 côté soudures du clavier.

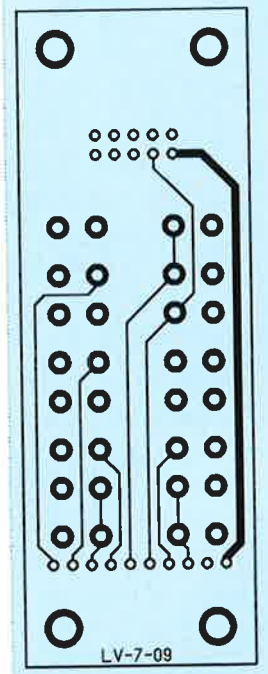


Figure 24 : circuit imprimé à l'échelle 1 : 1 côté composants du clavier.

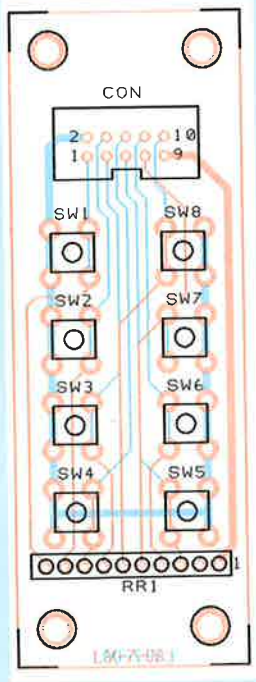


Figure 25 : schéma de câblage des composants du clavier.

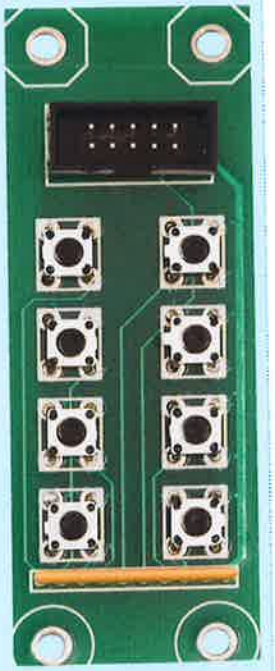


Figure 26 : photo de l'un de nos prototypes du clavier.

Liste des composants du clavier

RR1.... réseau de résistances 9 x 10 kΩ + C
SW1 à SW8 microswitch
Connecteur POD 2 x 5 broches

reçues sur le bus doivent être interprétées comme des **commandes** ou des **caractères à afficher**.

Une commande est composée d'un octet qui déplace le curseur ou efface l'écran, tandis que les données sont le texte (caractères) à afficher.

Enfin, la broche « **E** » de l'afficheur (broche 6) ou « **Entrée de validation** (**ENABLE**) » est active sur un front descendant.

Il faut lui envoyer une impulsion d'au moins 450 ns afin d'indiquer que des données valides sont présentes sur les broches **D0** à **D7**. L'afficheur lira les données sur le front descendant de cette entrée.

Le module d'affichage peut être connecté à n'importe quel port de la carte de base, il faudra cependant configurer correctement le programme en fonction des broches utilisées.

Le clavier

Le dernier module externe est un ensemble de touches formant un clavier dont le schéma électrique est visible en figure 22.

Il contient essentiellement un circuit imprimé sur lequel sont présents **8 boutons poussoirs** connectés sur **8 lignes** accessibles via le bornier « CON » de 10 broches.

Chaque bouton est doté d'une résistance de pull-up qui maintient la ligne correspondante à un niveau logique haut lorsque le bouton n'est pas pressé.

Le clavier permet de simuler des commandes manuelles pour vos applications. Il peut être relié à n'importe quel port de la carte de base, cependant vous devez configurer correctement le programme en fonction des broches utilisées.

Nous avons utilisé 8 boutons pour fonctionner avec un octet entier (ou un port complet) dans le programme, mais rien ne vous empêche de réaliser un clavier différent capable de gérer un plus grand nombre de boutons.

Réalisation pratique

À ce stade, nous devons analyser la construction des différentes cartes du système de développement.

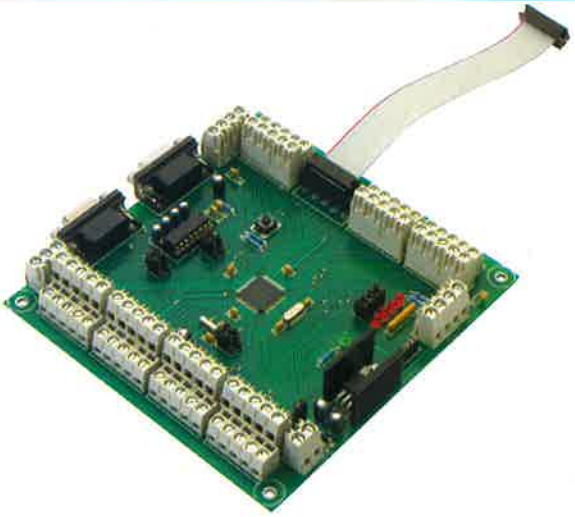
La première des choses à faire est de réaliser le circuit imprimé de la carte de base (celui contenant le microcontrôleur).

Attention **ce circuit est délicat à fabriquer par un processus de gravure classique**, notamment à cause de la finesse des pistes au niveau du microcontrôleur.

Nous vous conseillons d'**utiliser les fichiers GERBER et de passer par un professionnel**. Il en existe sur le web à des prix abordables.

Les modules par contre peuvent être facilement fabriqués par gravure chimique classique.

Vous trouverez tous les typons ainsi que tous les fichiers GERBER des circuits imprimés sur notre site **www.electroniquemagazine.com** dans le sommaire détaillé de la revue **135** à l'onglet « **Télécharger** ».

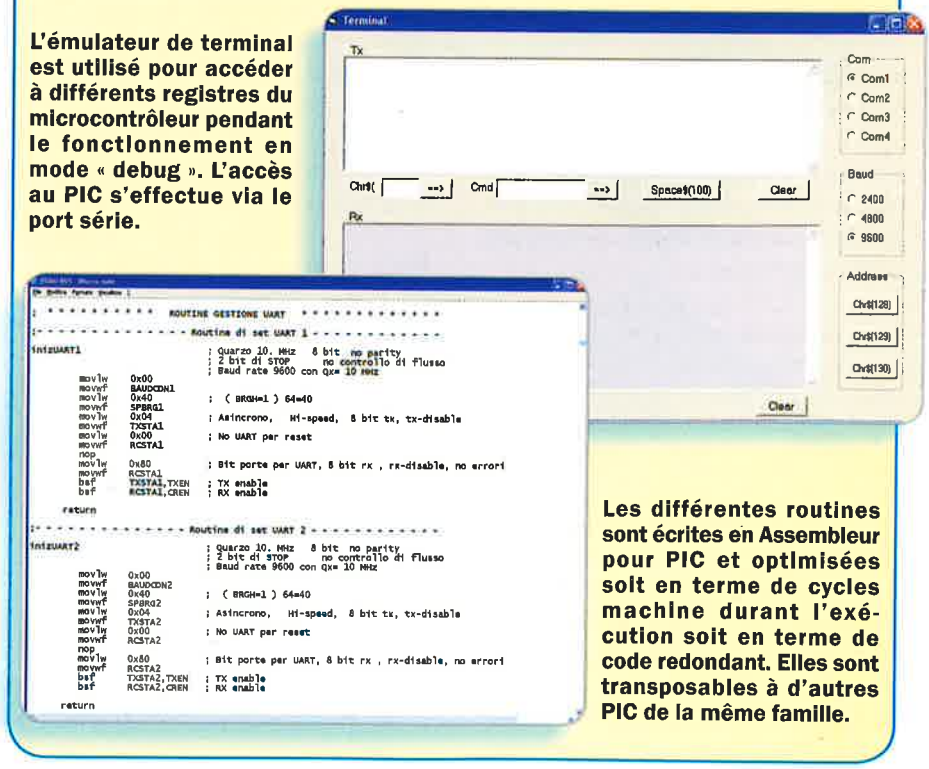


Si vous décidez de fabriquer par vos propres moyens la carte de base, une fois le circuit imprimé double face gravé, **vérifiez qu'il n'y a pas de cheveux d'anges** (ou courts-circuits) au niveau des pistes du microcontrôleur.

Si tout est correct commencez par souder le PIC, pour cela nous vous conseillons d'utiliser une **pointe de colle forte en tube que vous mettez sous la face du PIC**, puis positionnez-le correctement, le biseau ou le point doit être situé vers C9, en vérifiant que ses pattes viennent plaquer

À la découverte du logiciel

L'émulateur de terminal est utilisé pour accéder à différents registres du microcontrôleur pendant le fonctionnement en mode « debug ». L'accès au PIC s'effectue via le port série.



Les différentes routines sont écrites en Assembleur pour PIC et optimisées soit en terme de cycles machine durant l'exécution soit en terme de code redondant. Elles sont transposables à d'autres PIC de la même famille.

sur les pads (ou emplacements) des broches du circuit imprimé.

Au bout de quelques minutes commencez par **souder une broche d'un côté à l'aide d'un fer à souder de 25 W à 30 W doté d'une pointe fine de 0,4 mm pour CMS**. Utilisez de la s**oudure pour CMS de 0,5 mm de Ø**.

Ensuite continuez en soudant une broche du côté opposé et en laissant à chaque fois refroidir le PIC.

Vérifiez que de la soudure ne coule pas sous le PIC, mettez très peu de soudure. La colle vous permet de maintenir le PIC sans décalage au niveau des broches.

Une fois le microcontrôleur soudé, placez les jonctions entre les faces inférieure et supérieure du circuit à l'aide de morceaux de pattes de composants.

Manipulez délicatement le circuit imprimé car vous venez de souder le PIC, il ne faudrait pas l'arracher.

Ensuite comme d'habitude soudez les résistances, les condensateurs non polarisés, le support du MAX232, les condensateurs polarisés en respectant leur orientation.

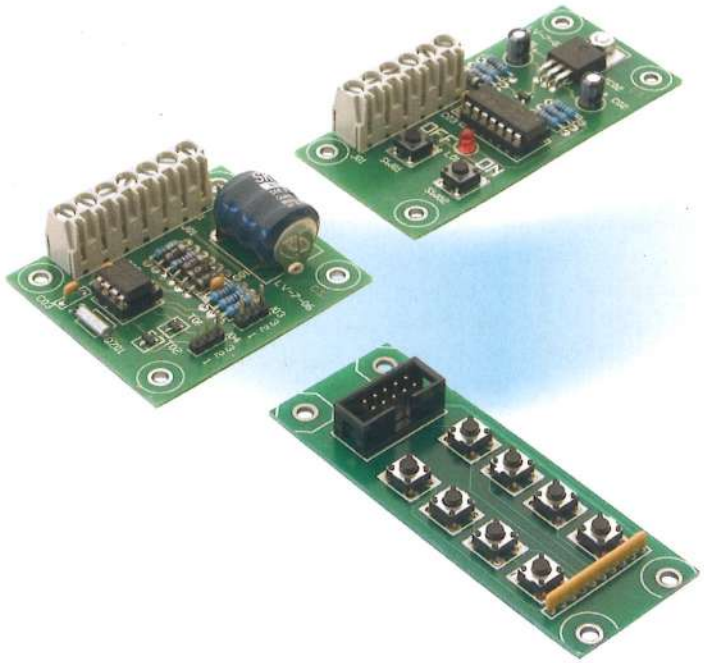
Continuez en soudant les LED, le quartz, le régulateur avec son dissipateur fixé sur le circuit imprimé, et enfin les barrettes mâles et femelles ainsi que les borniers.

Attention les borniers doivent être de type à deux niveaux, ils sont plus pratiques pour effectuer les connexions avec vos prototypes.

Une fois la carte de base terminée, procédez de la même manière pour les cartes d'extensions (module RTC, module d'alimentation, afficheur, clavier) qui sont beaucoup plus faciles à fabriquer.

Notez que sur le **module LCD**, les **composants sont soudés côté face « soudures »**. Utilisez une barrette mâle à 16 contacts afin d'interchanger facilement les différents types d'afficheurs LCD.

Pour rendre l'ensemble plus pratique, vous pouvez fixer les différentes cartes sur une planche de bois ou de plastique (évitez l'aluminium ou l'acier) avec des entretoises hexagonales de hauteur appropriée, en veillant à maintenir un espacement convenable pour effectuer les différentes connexions avec vos applications (cartes prototypes ou autres dispositifs).



Conclusion

Maintenant que vous avez réalisé la plate-forme de développement, vous pouvez télécharger à partir de notre site le code source complet qui comprend quelques routines couramment utilisées et le débogueur qui fonctionne sous Windows via le port série. Vous pourrez déjà vous faire une idée.

Nous analyserons en détail les routines et le programme Windows dans les prochaines revues. ■

NB : afin d'utiliser pleinement les ressources de la carte de développement décrite dans cet article (et les suivants) et notamment la programmation en circuit, nous vous conseillons d'utiliser le MPLAB ICD-3.

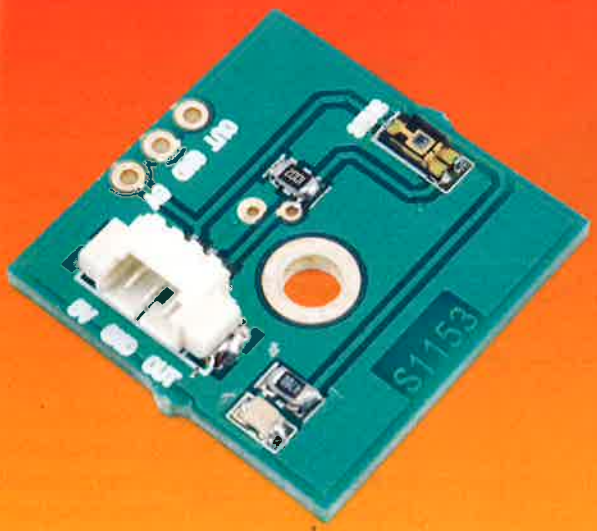
Il s'agit d'un programmateur en circuit et d'un émulateur pour les microcontrôleurs Microchip. Il permet aussi de déboguer en circuit les programmes grâce à son environnement de développement MPLAB (IDE).

Il est disponible sur les sites spécialisés tels que http://fr.farnell.com ou http://fr.rs-online.com.

Caractéristiques :

- Débogage en temps réel ;
- Connectivité standard RJ-11 Microchip ;
- Alimentation par USB ;
- Programmation ultrarapide ;
  - Emulation faible tension, prend en charge des tensions d'alimentation cibles allant de 2 V à 5,5 V;
  - Module d'interface de test pour permettre aux utilisateurs de tester les lignes d'E/S ;
  - Mise à niveau des nouvelles versions de l'IDE MPLAB et du firmware en téléchargement gratuit sur le site web de Microchip.

BREAKOUT BOARD - 2

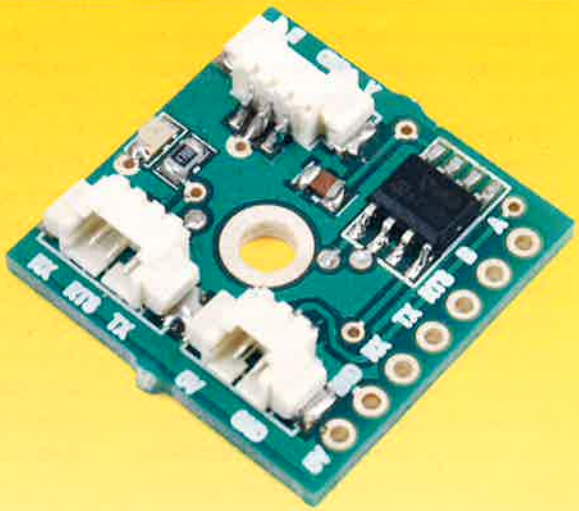


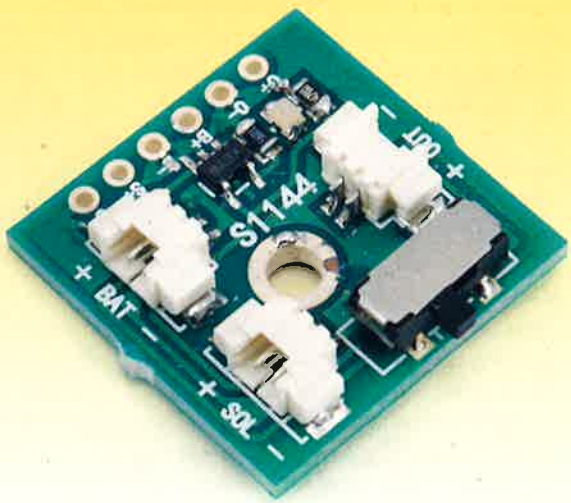
Nous continuons la présentation de différentes cartes de prototypage ou « Breakout board » avec trois nouvelles cartes dédiées à la recharge de batterie Lithium-Ion, à la communication RS485 et à la détection de lumière.

CARTES DE PROTOTYPAGE MULTIFONCTIONS

Deuxième partie

de Davide SCULLINO





De nos jours, les composants électroniques les plus intéressants ne sont malheureusement disponibles qu'en boîtiers CMS à cause de la miniaturisation des systèmes portables. Ils sont donc de plus en plus petits, et il devient difficile pour un particulier de les mettre en œuvre sans équipement particulier et donc très coûteux.

C'est pour cette raison que nous avons réalisé des cartes de prototypages disposant de composants offrant des fonctions très utiles dans de nombreuses applications (capteur, communication, gestion de charge, etc.).

Comme nous vous l'avons exposé dans la première partie (voir le numéro 135), ces cartes sont dotées de composants CMS spécifiques déjà montés avec toutes les connexions nécessaires pour une mise en œuvre immédiate.

Après vous avoir décrit dans le **précédent numéro** (135) les **3 premières cartes de prototypage**, à savoir une **carte capteur de température** (DS18B20), une **carte capteur d'humidité** (HIH-5030-001) et une **carte capteur de pression** (MPX6115A6U), nous continuons la description de 3 nouvelles cartes.

Une première carte basée sur un **capteur de luminosité** de type « **TEMT6000** », une deuxième carte dédiée à la **communication de type RS485** basée sur un circuit « **ST485BDR** » et enfin une troisième carte **chargeur de batterie Lithium-Ion** basée sur un circuit intégré « **MCP73831T** »

La carte capteur de luminosité

Cette carte comprend un circuit intégré capteur de lumière, le « **TEMT6000** » de la société **Vishay** (www.vishay.com). Elle permet de **détecter la luminosité** d'un environnement, il est possible de l'intégrer dans un système déjà existant.

Le « TEMT6000 » est un **phototransistor NPN** disponible **uniquement en boîtier CMS**, et dont l'agencement des contacts est adopté par de nombreux composants modernes, c'est-à-dire **qu'il n'y a plus de broches qui sortent du boîtier**, mais des **contacts** (ou plots) en forme de « **L** » qui se trouvent sur les côtés du boîtier. Il est donc très difficile de les souder sans équipement spécifique, de ce fait le soudage par un hobbyiste avec du matériel conventionnel n'est pas possible (il faut utiliser des systèmes à air chaud ou infrarouges).

Le capteur « TEMT6000 » est conforme à la norme « RoHS 2002/95/EC » ainsi qu'à la norme « WEEE 2002/96/EC » en ce qui concerne l'absence de plomb. Vous pouvez utiliser ce capteur dans des applications où la qualité est requise.

Ce capteur possède donc un phototransistor ayant une jonction « base-émetteur » disposée face à une fenêtre de surface supérieure et sensible au spectre de la lumière visible. La sensibilité est garantie pour un **angle d'incidence** de **± 60 °** par rapport à la perpendiculaire de ladite surface.

Le fonctionnement du phototransistor est très simple. La **jonction** « **collecteur** » est **polarisée par une tension positive** dans le cas d'un phototransistor **NPN**, ou **négative** dans le cas d'un phototransistor **PNP**, par rapport à l'émetteur.

Lorsque la lumière atteint la base, l'énergie du rayonnement lumineux brise les liens entre les charges électriques en excès et, le silicium dopé qui compose le transistor, libère ainsi les charges électriques qui donnent naissance à un **courant de base amplifié par le champ électrique présent entre le collecteur et la base**. Il en résulte un **courant de collecteur**, dont **l'intensité est directement proportionnelle** à la

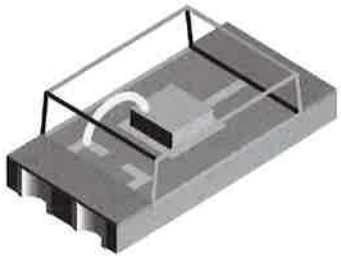


Figure 1 : structure du capteur de lumière « TEMT6000 ».

quantité de lumière frappant la base, même si la relation n'est pas parfaitement linéaire (elle s'en rapproche).

En ce qui concerne les connexions, seuls le collecteur et l'émetteur du « TEMT6000 » disposent de broches, la base n'est pas accessible. Ce choix est judicieux pour un capteur de luminosité, car il n'est pas nécessaire de détecter des variations rapides de luminosité.

Normalement, dans un phototransistor, il est utile de relier la base à l'émetteur avec une résistance de valeur relativement élevée. Cela permet d'accélérer la recombinaison des charges libérées par la jonction base-émetteur lorsqu'elle est soumise à un flux lumineux.

Cependant pour des applications où le capteur doit détecter de façon précise la luminosité, lorsque la fréquence des variations de l'intensité lumineuse sur la jonction est élevée, le temps nécessaire aux charges libérées par la lumière pour se recombiner et bloquer le phototransistor, devient comparable à la durée des impulsions. Ainsi le transistor ne peut plus « suivre » les variations lumineuses.

En connectant la base à l'émetteur avec une résistance, cela permet la circulation d'un courant qui accélère la recombinaison et donc augmente la vitesse de commutation du phototransistor. Cependant cette solution entraîne une diminution de la sensibilité du composant, en ce sens que pour une même quantité de lumière éclairant la base, le courant de collecteur est moins important.

Pour mettre en œuvre le phototransistor « TEMT6000 », il doit être polarisé comme un **transistor utilisé en amplificateur** (typiquement en « émetteur commun » ou en « collecteur commun »).

Dans notre montage, le capteur est configuré en « **collecteur commun** ».

Cela signifie qu'aux **bornes** de la résistance **R1**, connectée entre l'émetteur et la masse, la **différence de potentiel exprime l'intensité lumineuse** arrivant sur la base, c'est-à-dire l'éclairement lumineux en **lux**.

Le montage en « collecteur commun » permet un fonctionnement linéaire du phototransistor, donc cela permet d'obtenir une relation pratiquement linéaire (voir la figure 2) entre l'éclairement lumineux et le courant de collecteur et, par conséquent celui de l'émetteur.

Étant donné que ce dernier détermine la tension aux bornes de la résistance d'émetteur (R1 sur le schéma), nous obtenons une **tension de sortie** entre la broche « **OUT** » et la masse (GND) qui **varie linéairement** et de manière directement **proportionnelle à l'éclairement lumineux**.

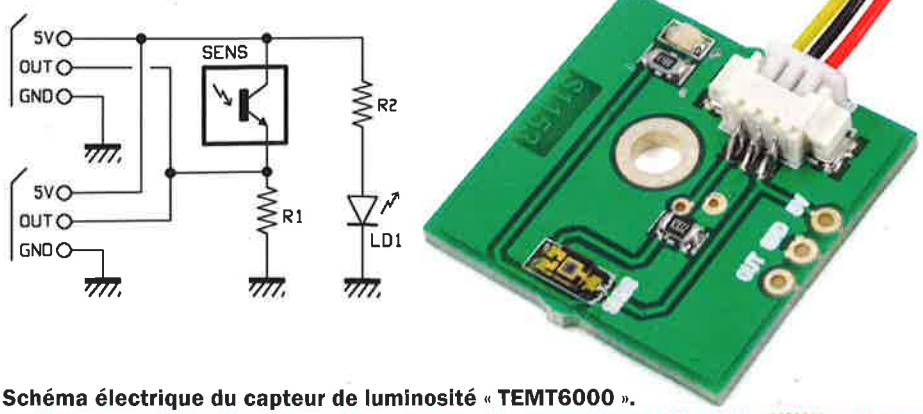


Schéma électrique du capteur de luminosité « TEMT6000 ».

Pour utiliser correctement le capteur « TEMT6000 », vous devez vous rappeler de ses principales caractéristiques, il présente, à une température ambiante de 25 °C, une tension « V_{CE} », entre le collecteur et l'émetteur, maximale, dans l'obscurité, de **6 V** et une tension inverse entre l'émetteur et le collecteur « V_{EC} » qui ne doit pas dépasser **1,5 V** avec un **courant de collecteur** maximal de **20 mA**.

Pour notre carte de prototypage ces caractéristiques sont satisfaites, puisque la tension d'alimentation est de 5 V et que la tension entre le collecteur et l'émetteur est inférieure, étant donné que la chute de tension aux bornes de R1 doit être retranchée des 5 V.

Le courant de collecteur I_c n'atteindra jamais 20 mA, car la valeur de la résistance R1 étant de 10 kΩ, le courant maximal de collecteur sera de 5/10 000 soit I_c = **0,5 mA**.

Dans des conditions de **faible luminosité**, la sortie « **OUT** » de la carte présentera une tension d'environ **0,5 V**. En utilisant une lampe dirigée vers le capteur, la tension de sortie atteindra près de **5 V**.

Pour une polyvalence maximale, toutes les lignes (alimentation, tension de sortie et masse) sont accessibles à deux endroits sur les bords de la carte. Vous disposez ainsi d'un connecteur et d'un report des lignes directement sur le circuit imprimé.

En ce qui concerne les applications de cette « breakout board », il est possible de réaliser un détecteur de lumière ambiante pour commander automatiquement l'éclairage d'un local, ou le rétroéclairage d'un afficheur LCD.

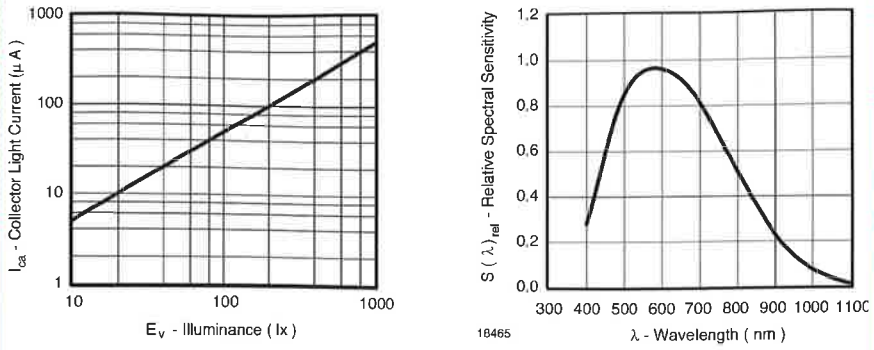


Figure 2 : courbes décrivant le courant de collecteur du « TEMT6000 » en fonction de l'éclairement lumineux (à gauche) et de la réponse spectrale (à droite).

Vous pouvez aussi l'utiliser comme un luxmètre à condition que la précision ne soit pas primordiale. Vous pouvez aussi l'utiliser pour l'éclairage automatique d'une galerie ou d'un couloir.

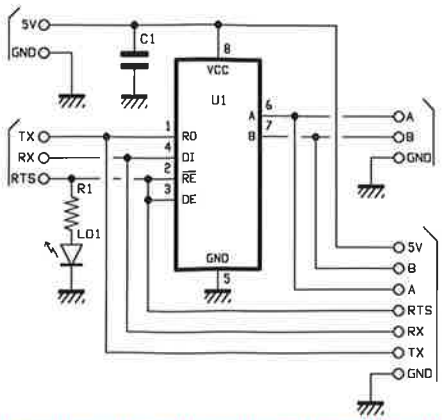
La carte de communication RS485

Nous allons étudier maintenant la 2ème « breakout board » de cet article, c'est une carte qui permet **d'interfacer une ligne série TTL** (par exemple la sortie d'un UART d'un microcontrôleur) avec un **bus de type RS485 série**.

Cependant le **bus RS485** présente une **ligne symétrique** (équilibrée) composée de **2 fils formant une paire torsadée** et référencés par rapport à la masse. Par contre **la ligne série TTL est de type asymétrique**, c'est-à-dire avec un seul conducteur référencé par rapport à la masse.

Avec sa **configuration symétrique**, le bus RS485 permet la communication sur de **longues distances**, de l'ordre du kilomètre, tout en étant insensible aux perturbations environnantes. Les informations sérielles sont véhiculées sur une même ligne soit dans une direction soit dans l'autre, nous obtenons un système émetteur/récepteur. Le bus RS485 peut supporter ainsi jusqu'à **32 émetteurs** et **32 récepteurs** sur une même ligne. Une ligne RS485 peut donc interconnecter plusieurs dispositifs, dotés chacun d'un émetteur et d'un récepteur.

De la sorte, une communication peut s'établir entre n'importe quelle paire de ces dispositifs avec seulement trois fils (2 fils conducteurs plus la masse).



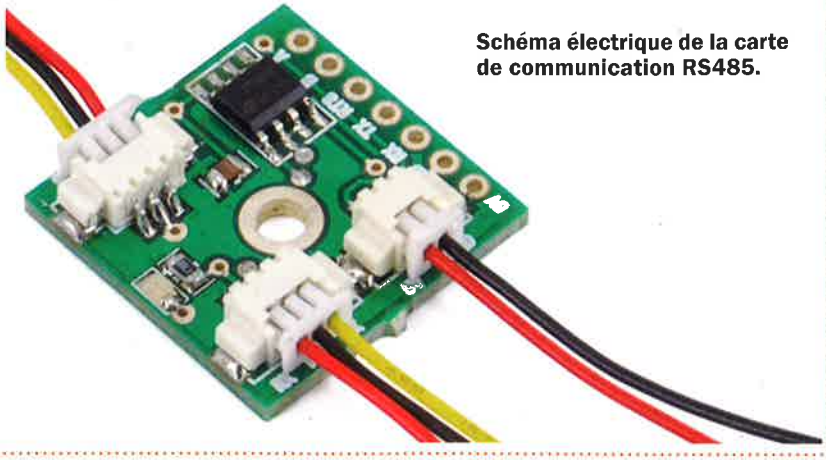


Schéma électrique de la carte de communication RS485.

La performance en termes **d'immunité au bruit électrique** et la **possibilité de couvrir de grandes distances** sont dues au fonctionnement en « **boucle de courant** » et à **une amplitude élevée de la tension du signal** des données caractérisant le bus RS485.

Lors d'une transmission, une tension différentielle référencée par rapport à la masse est présente sur les deux fils. Avec une tension d'alimentation égale, l'**amplitude du signal est doublée** par rapport à celle obtenue à partir d'une ligne asymétrique.

Le terme symétrique (équilibré) signifie que la ligne présente, à chaque impulsion, deux tensions d'amplitude égale mais de polarité opposée sur les deux fils (A et B) de la ligne RS485. Cela permet, dans le récepteur, d'utiliser un amplificateur différentiel pour additionner algébriquement les signaux et supprimer ainsi toutes les perturbations.

En effet, en reliant les deux fils transportant les signaux des données, un à l'entrée inverseuse et l'autre à l'entrée non-inverseuse d'un amplificateur différentiel (par exemple un amplificateur opérationnel configuré en comparateur de tension), les signaux sont additionnés algébriquement.

Lorsque la première entrée est négative, la seconde est positive, la sortie est alors négative c'est-à-dire à un niveau logique « 0 ». Dans le cas inverse, la sortie se trouve à un niveau logique « 1 ». Ainsi les signaux TTL (0 V / 5 V) sont « reconstitués » en sortie de l'amplificateur opérationnel.

Le bruit sur la ligne est supposé être d'amplitude égale sur les fils A et B. Il est ainsi annulé, car le bruit présent sur l'entrée inverseuse est « compensé » par celui de l'entrée non-inverseuse (opposition de phase).

Le bus RS-485 est un bus sur lequel plusieurs dispositifs peuvent être connectés en parallèle sur les fils de la ligne. Le dernier dispositif (le plus éloigné) doit comporter une résistance de terminaison de 120 Ω entre les fils A et B. La vitesse de transmission permise par le bus RS485 est de **35 Mbit/s** pour une distance de **10 m** et **100 kbit/s** pour une distance de **1200 m**.

Le montage utilise, pour la conversion analogique TTL/RS485, un circuit intégré « **ST485BDR** » disposant d'un émetteur/récepteur. Il s'agit de U1 sur le schéma électrique. Ce circuit est fabriqué par **STMicroelectronics** et est disponible en boîtier « **SO-8** » (boîtier CMS avec 8 broches) ou en boîtier « DIP-8 » traditionnel. Pour notre « breakout board », nous utilisons la version « SO-8 ».

Un circuit « émetteur/récepteur » est un composant (voir la figure 3), qui intègre une section « émission » (partie D) et une section « réception » (partie R). Dans notre cas, ces deux sections sont activées séparément à l'aide de broches spécifiques.

Plus précisément, la broche « **DE** » **active** la section « **émission** » lorsqu'elle est portée à un niveau logique **haut** (5 V). Par contre la broche « **RE** » **active** la section « **réception** » lorsqu'elle est mise à un niveau logique **bas**, c'est-à-dire à la masse (ou « 0 V »).

La logique de commande de ces deux entrées **est volontairement inversée** afin de pouvoir les **contrôler avec une seule broche.** Il est ainsi possible d'activer le récepteur, ce qui met hors service l'émetteur (fonctionnement en réception de données), ou d'activer l'émetteur ce qui met hors service le récepteur (fonctionnement en émission).

Il est également possible de gérer séparément ces deux broches pour garder un fonctionnement en réception même lorsque le « ST485DBR » émet, de manière à obtenir une rétroaction (feedback) sur l'état de l'émetteur.

Cependant, il n'est pas conseillé dans une communication de garder l'émetteur actif lorsque le « ST485DBR » reçoit des données d'une autre unité connectée au bus, car les deux émetteurs pourraient s'endommager mutuellement, bien que le « ST485DBR » intègre une protection.

Dans notre circuit, les broches « **DE** » et « **RE** » du « ST485DBR » sont gérées ensemble et contrôlées par le signal de

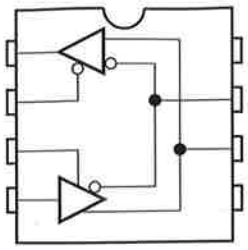


Figure 3 : brochage et connexions internes du ST485DBR.

commande « **RTS** » de type série provenant de l'UART. Les broches « **RX** » et « **TX** » de l'UART sont connectées respectivement aux broches « **RO** » (sortie de l'étage récepteur du « ST485DBR ») et « **DI** » (entrée de l'étage émetteur du « ST485DBR »).

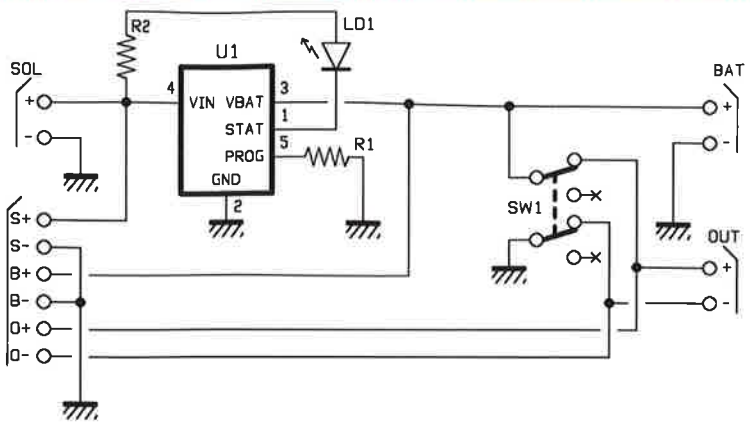
Les deux lignes du bus RS485 sont connectées aux broches A (6) et B (7) du « ST485DBR » et sont reportées sur le circuit imprimé de la carte afin d'y accéder pour effectuer des tests.

Pour offrir une polyvalence maximale, toutes les lignes (alimentation, RX, TX, RTS du signal TTL, lignes A et B du bus RS485) sont disponibles à deux endroits sur le circuit imprimé de la carte. Les pastilles proches de U1 permettent la connexion par empilement d'autres cartes.

Complétons la description du montage par la présence de la LED LD1 qui indique l'état de fonctionnement du circuit (émission ou réception), le condensateur C1 permet un filtrage des éventuels parasites présents sur l'alimentation.

Avant de passer à la description de la dernière « breakout board » pour cet article, résumons les caractéristiques du « ST485DBR » pour l'utiliser au mieux.

Le circuit intégré accepte **au repos des tensions sur les broches A et B** en mode commun (référencées par rapport à la masse) comprises entre **-7 V** et **12 V**, c'est-à-dire avec le driver et le récepteur éteints. La consommation dans ce cas est de seulement **300 µA**. L'émetteur, lorsqu'il est éteint, garde sa sortie dans un état de haute impédance afin de ne pas charger la ligne.



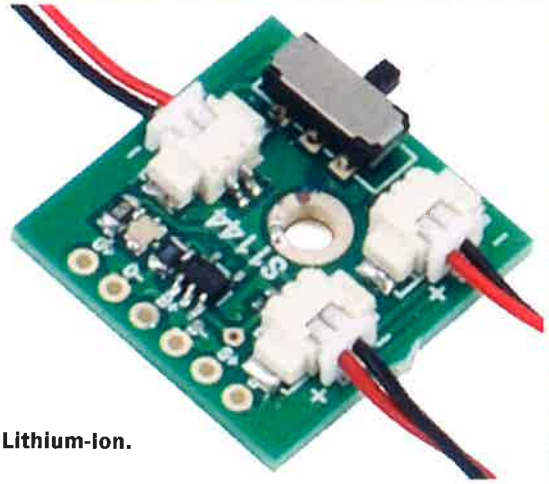


Schéma électrique de la carte chargeur de batterie Lithium-Ion.

Le temps de propagation, à savoir le **retard avec lequel un niveau logique appliqué** sur la broche « **DI** » **se présente** sur les broches **A** et **B**, ou le retard avec lequel un niveau appliqué sur les broches A et B se présente sur la broche « **RO** », n'est que de **30 ns**, ce qui permet une communication à haute vitesse.

Sur les lignes A et B du bus RS485 peuvent être raccordée jusqu'à 64 circuits intégrés du même type, cependant un seul circuit à la fois peut effectuer une transmission.

L'étage « **émetteur** » dispose d'un **limiteur de courant** qui, lorsqu'il intervient, met la sortie dans un état de haute impédance. La limitation est utile pour prévenir des dommages en cas de courts-circuits, ou si un autre périphérique connecté au bus RS485 émet en même temps.

La carte chargeur de batterie Lithium-Ion

Nous arrivons à la description de la dernière carte de prototypage (pour cet article). Elle intègre un chargeur de batterie pour un seul élément **Lithium-Ion** de **3,6 V** (4,2 V à pleine charge). La carte est basée sur un circuit intégré « **MCP73831T** » de type CMS en boîtier « **SOT-23** » et fabriqué par Microchip.

La carte accepte sur le connecteur « SOL » une alimentation typique de 5 V, qui peut provenir soit d'une alimentation classique, soit d'un petit panneau solaire, mais aussi d'un connecteur USB d'un ordinateur.

À partir des contacts du connecteur « SOL », la tension d'alimentation arrive sur l'entrée (VIN) du régulateur de charge U1, le « MCP73831T ». Ce circuit intégré accepte une tension en entrée variant de 3,75 VDC à 6 VDC. Il fournit en sortie un courant nécessaire pour la charge d'éléments Li-Ion ou Li-Po (lithium-polymère) allant jusqu'à 550 mA.

La batterie qui est raccordé au connecteur « BAT » peut avoir une capacité théoriquement illimitée car il n'y a pas de limitation dans le temps. Avec un courant en sortie de 550 mA, il faudra 1 heure pour charger un élément d'une capacité de 550 mAh, et 2 heures pour un élément de 1100 mAh. Pour un élément de 5,5 Ah, il faudra 10 heures et ainsi de suite.

Dans notre carte de prototypage, le circuit intégré « MCP73831T » fonctionne en configuration classique. La LED LD1 est alimentée à travers la résistance R2 de limitation de courant par la broche « **STAT** ».

Celle-ci se trouve à un **niveau logique bas** (0 V) pendant la **charge**, ce qui a pour effet de faire circuler un courant dans LD1 et ainsi de l'allumer. À la fin de la charge, la broche « **STAT** » prend un **niveau logique haut**, la tension présente est proche de celle de l'alimentation.

Il en résulte que l'anode et la cathode de la LED LD1 se retrouvent pratiquement au même potentiel, la **différence de potentiel aux bornes de LD1 est donc nulle**, aucun courant ne circule et LD1 est **éteinte**.

De même la broche « **STAT** » prend un **niveau logique haut** lorsque le circuit « **MCP73831T** » est en **veille** (**shutdown**) ou lorsqu'aucune batterie n'est connectée à la sortie « VBAT ». Cette dernière sortie (broche 3) est la ligne sur laquelle est connectée la batterie à charger.

Le circuit « MCP73831T » effectue **la charge à courant constant et à tension constante**. Le courant de charge (Ireg) est imposé (ou réglé) par la résistance R1 reliée à la broche 5, dont la valeur est calculée par la relation suivante :

Ireg = 1000 / R

où la **valeur de R est exprimée en ohms** (Ω) et la **valeur du courant Ireg est exprimée en ampères** (A).

Par exemple, avec une valeur de R1 = 4,7 kΩ soit 4700 Ω, le courant Ireg vaut :

Ireg = 1000 / 4700 = 1 / 4,7 = 0,212 A

donc Ireg = 212 mA

Avec une valeur de R1 égale à 2,2 kΩ, le courant Ireg vaut environ 454 mA.

Si la broche 5 n'est pas connectée (se retrouve « en l'air »), le circuit « MCP73831T » se trouve au repos (shutdown) et ne consomme que 2 µA. Cette broche peut aussi être utilisée pour commander le circuit (« ENABLE »).

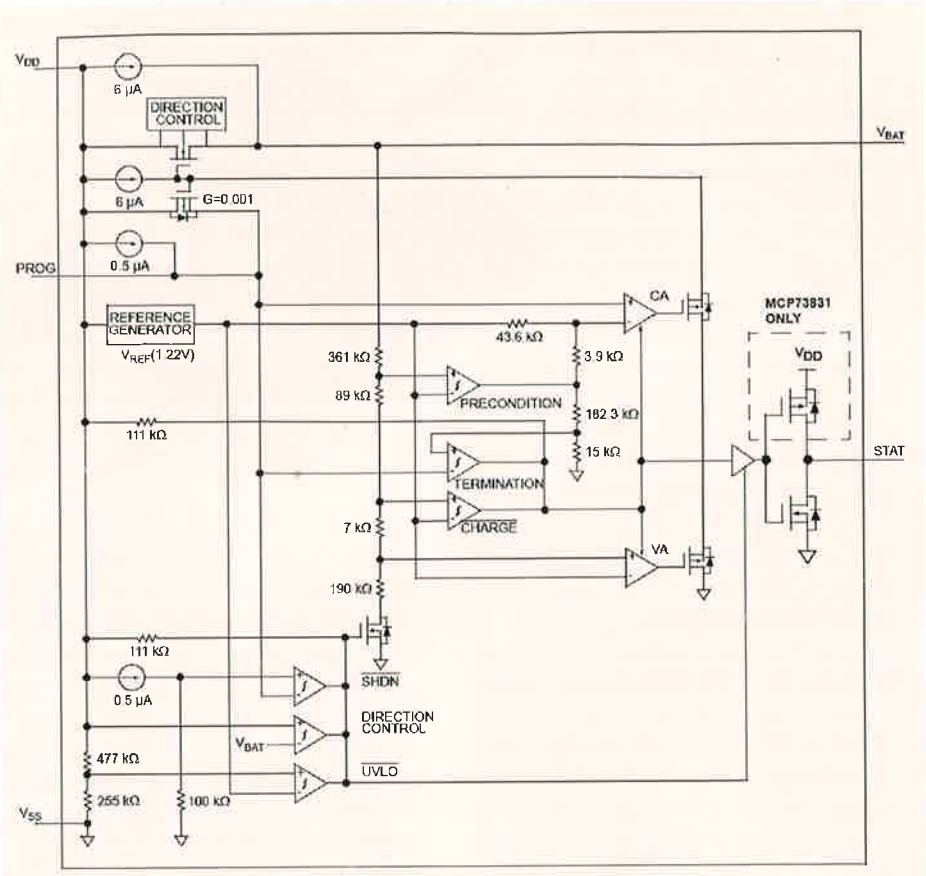


Figure 4 : schéma synoptique du MCP73831T.

La charge se déroule en plusieurs phases lorsque la batterie est reliée à la sortie « VBAT » (broche 1).

Si la **tension détectée est inférieure à un certain seuil**, généralement autour de **70 %** de la valeur de la pleine charge, la charge commence avec une **valeur de courant faible** afin de « préparer » la batterie à une charge rapide.

Cette dernière commence lorsque la tension aux bornes de la batterie dépasse le seuil de 70 % et la valeur du courant de charge est déterminée par la résistance reliée à la broche 5.

Dès que la **tension de la batterie atteint la tension de régulation** définie par le circuit « MCP73831T », celui-ci **arrête la charge rapide** et **commence une charge de maintien** qui permet de stabiliser la tension à 4,2 V, ou 4,35 V, ou 4,4 V ou 4,5 V, en fonction de la **version du circuit intégré utilisé**. Nous avons pris, pour notre projet, le « **MCP73831T-2** » qui régule à 4,2 V. Les différentes tensions de régulation du « MCP73831T » ont été définies par le constructeur afin de

Plan de montage de la carte capteur de luminosité à base de TEMT6000.

Figure A : circuit imprimé à échelle 1 : 1 côté face inférieure (plan de masse) de la carte capteur de luminosité.

Figure B : circuit imprimé à échelle 1 : 1 côté composants de la carte capteur de luminosité.

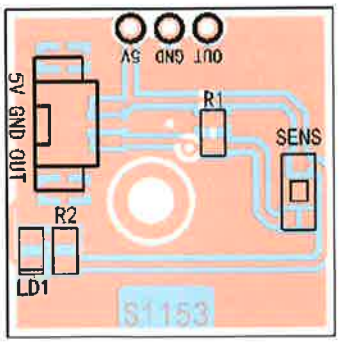


Figure C : schéma de câblage de la carte capteur de luminosité.

Liste des composants de la carte capteur de luminosité

R1..... 10 kΩ (0805)
R2..... 470 Ω (0805)
LD1.... LED verte (0805)
SENS . TEMT6000

Divers :

Barrette mâle 3 pôles
Connecteur mâle JST 1,25 mm 3 pôles pour circuit imprimé
Connecteur femelle JST 1,25 mm 3 pôles fils volants

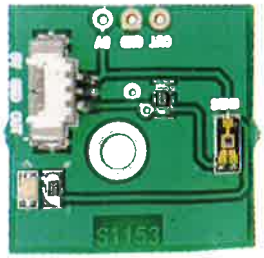


Figure D : photo de l'un de nos prototypes de la carte capteur de luminosité.

Plan de montage de la carte de communication RS485

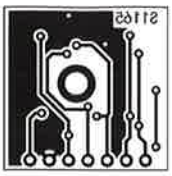


Figure E : circuit imprimé à échelle 1 : 1 côté face inférieure (plan de masse) de la carte de communication RS485.

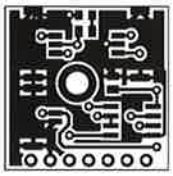


Figure F : circuit imprimé à échelle 1 : 1 côté composants de la carte de communication RS485.

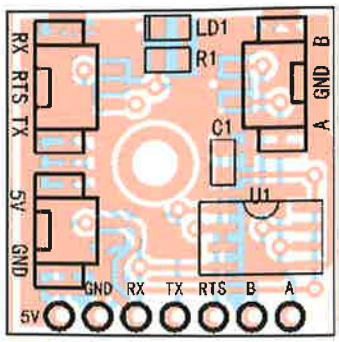


Figure G : schéma de câblage de la carte de communication RS485.

Liste des composants de la carte de communication RS485

R1..... 470 Ω (0805)
C1..... 100 nF céramique (0805)
LD1.... LED verte (0805)
U1..... ST485BDR

Divers :
Barrette mâle 7 pôles
Connecteur mâle JST 1,25 mm 2 pôles pour circuit imprimé
Connecteur mâle JST 1,25 mm 3 pôles pour circuit imprimé (x2)
Connecteur femelle JST 1,25 mm 2 pôles fils volants
Connecteur femelle JST 1,25 mm 3 pôles fils volants (x2)

Figure H : photo de l'un de nos prototypes de la carte de communication RS485.

Plan de montage de la carte chargeur de batterie Lithium-Ion

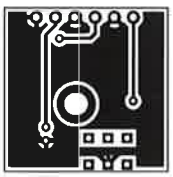


Figure I : circuit imprimé à échelle 1 : 1 côté face inférieure (plan de masse) de la carte chargeur de batterie Li-Ion.

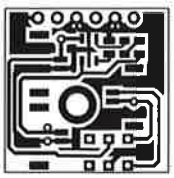


Figure J : circuit imprimé à échelle 1 : 1 côté composants de la carte chargeur de batterie Li-Ion.

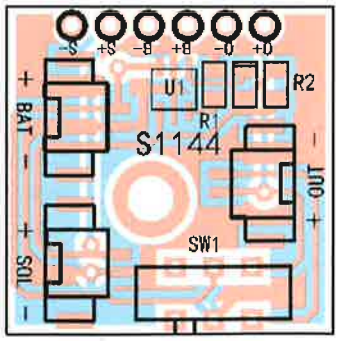


Figure K : schéma de câblage de la carte chargeur de batterie Li-Ion.

Liste des composants de la carte chargeur de batterie Li-Ion

R1..... 5,6 kΩ (0805)
R2..... 470 Ω (0805)
LD1.... LED verte (0805)
SW1... interrupteur à glissière coudé à 90°

U1..... MCP73831T
Divers :

Barrette mâle 6 pôles
Connecteur mâle JST 1,25 mm 2 pôles pour circuit imprimé (x3)
Connecteur femelle JST 1,25 mm 2 pôles fils volants (x3)

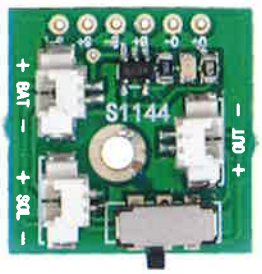
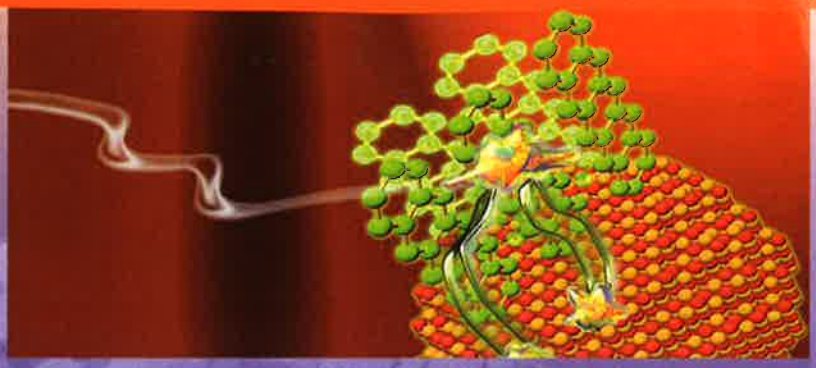


Figure H : photo de l'un de nos prototypes de la carte de communication RS485.

Le photovoltaïque au-delà de 33,7 %, est-ce possible ?



Le pourcentage de la lumière solaire convertie en électricité par les cellules photovoltaïques est théoriquement limitée à 33,7 %, cela correspond à la limite de « Shockley-Queisser ». En effet avec les technologies actuelles, il est impossible d'extraire les électrons contenus dans une zone importante appelée « état triplet » (« spin triplet »), où les électrons se trouvent dans un état excité engendré par l'absorption de la lumière par le semi-conducteur.

En résumé, il est très difficile d'extraire les électrons dans un état excité. Des chercheurs de l'Université de Cambridge ont réussi à extraire cette électricité « latente ». une de leurs plus récentes expériences a été d'utiliser un nanocristal à base de pentacène hybride (il s'agit d'un semi-conducteur organique) et de séléniure de plomb (PbSe, semi-conducteur inorganique).

Cette composition pourrait théoriquement convertir 100 % de la lumière reçue en énergie électrique.

Le groupe qui effectue les recherches développe un revêtement organique qui pourrait être appliqué à des cellules photovoltaïques à base de silicium afin d'augmenter le rendement de conversion des panneaux solaires.

répondre aux exigences des charges des nouveaux modèles de batteries au lithium, qui peuvent fonctionner avec des tensions supérieures à 4 V.

Pour compléter le chargeur de batterie, nous avons inséré un interrupteur à glissière SW1 qui permet de déconnecter la batterie de la sortie « OUT » du MCP73831. Cela permet de recharger la batterie tout en déconnectant l'utilisateur relié à la sortie « OUT ».

Comme pour les deux cartes décrites précédemment, **toutes les lignes sont reportées à la fois vers des connecteurs et vers des pastilles sur le circuit imprimé**. Il est ainsi possible de réaliser un montage par empilement avec d'autre cartes.

Réalisation pratique

Nous avons étudié les schémas et les fonctionnements des 3 « breakout board » à base de 3 capteurs différents, nous allons maintenant les construire.

Pour cela vous pouvez télécharger les typons des circuits imprimés sur notre site **www.electroniquemagazine.com** dans le sommaire détaillé de la revue **136** à l'onglet « **Télécharger** ». Vous y trouverez aussi gratuitement les fichiers GERBER des 3 cartes pour une réalisation professionnelle.

Une fois les circuits imprimés fabriqués, commencez par la première carte capteur de luminosité. Nous vous rappelons que pour une bonne soudure, il est conseillé d'utiliser un fer à souder de 25 W avec une pointe de 0,5 mm de diamètre et de la soudure pour composants CMS.

Munissez-vous d'une lampe ayant une fonction loupe, vérifiez l'orientation des composants et commencez par les souder en utilisant très peu de soudure afin d'éviter qu'elle coule en dessous du boîtier CMS een provoquant ainsi des courts-circuits.

Vous pouvez vous procurer les cartes déjà montées et testées, voir les annonces dans la revue.

La disposition et l'orientation des composants sont visibles sur les schémas de câblages (figures C, G et K). Aidez-vous des photos des prototypes que vous pouvez voir aux figures D, H et L.

Complétez le soudage des composants avec les résistances, l'unique condensateur C1 sur la carte RS485, les connecteurs, les LED CMS sur lesquelles l'encoche (ou biseau) indique la cathode.

Enfin terminez en soudant l'interrupteur à glissière pour la carte chargeur de batterie.

Notez que pour la carte capteur de luminosité contenant le « TEMT6000 », celui-ci ne possède pas de broches qui sortent du boîtier mais plutôt des « plots » sur les côtés. Il faut chauffer délicatement un plot à la fois et appliquer une pointe de soudure sans excès.

Notez aussi que, dans le cas où vous ne les trouviez pas dans le commerce, les connecteurs des 3 cartes sont facultatifs. Vous pouvez utiliser les pastilles du circuit imprimé en y soudant directement les fils.

Si vous envisagez d'empiler plusieurs « breakout board », utilisez des barrettes mâles/femelles au pas de 2,54 mm, cela simplifiera la connexion et l'utilisation des cartes.

Toutes les cartes doivent être alimentées avec une tension stabilisée de 5 VDC. Si vous les raccordez à des circuits existants, assurez-vous que vous disposez de signaux compatibles TTL et d'une alimentation stabilisée. La consommation de courant des cartes est très faible, si bien que vous pouvez utiliser des piles, des accumulateurs ou encore un petit panneau solaire pour les alimenter.

La voiture sans conducteur de Delphi à l'essai aux États-Unis

La société britannique Delphi Automotive, un fabricant de composants électroniques pour l'automobile, a développé un programme spécial de test d'une voiture autonome aux États-Unis.

En fait, Delphi a constitué une équipe d'ingénieurs qui sont partis de San Francisco jusqu'à New York à bord d'un SUV compact de type Audi SQ5. Celui-ci a la particularité d'être une voiture sans conducteur, le défi étant de parcourir 5600 kms en 8 jours et ce de manière entièrement autonome.

La voiture a été modifiée, des caméras spéciales, des radars et des logiciels ont été installés afin de tester tous les problèmes de la technologie « driverless » (sans conducteur). Pour des raisons de sécurité et afin de vérifier la fiabilité du SUV, un conducteur se trouve derrière le volant pour écarter tout risque de danger lors d'événements inattendus (comme cela s'est produit récemment avec l'Autopilot de Tesla bien que la voiture ne semble pas être en cause).

Pour Jeff Owens, directeur technique de Delphi Automotive, la technologie sans conducteur a fait d'énormes progrès ces dernières années, mais il faudra environ 20 ans pour qu'un véhicule autonome soit commercialisé, car il faut adapter la règlementation.

www.delphi.com/delphi-drive

Des nouvelles cellules photovoltaïques du MIT

Afin de réduire les coûts de l'énergie photovoltaïque, des chercheurs du Massachusetts Institute of Technology ont développé une cellule solaire dont l'efficacité énergétique atteint 35 %, dépassant ainsi la capacité des technologies actuellement sur le marché. Les cellules solaires sont composées de deux couches de matériaux différents, qui absorbent une grande quantité de lumière. L'un est composé de silicium et le second élément semi-transparent est formé de Pérovskite. Il existe déjà des cellules solaires en deux couches, mais avec des circuits électriques séparés.

WaterNest 100 : la maison flottante écologique

Cette maison flottante s'appelle WaterNest 100, elle dispose d'un design flexible qui s'adapte à tout type d'environnement. Elle est alimentée par des panneaux solaires situés sur le toit et fournissant l'énergie nécessaire pour toute la maison. Elle a été conçue par Giancarlo Zema pour la société EcoFloLife. La maison est accessible via un ponton pratique. La WaterNest 100 est construite avec du bois et de l'aluminium, elle est respectueuse de l'environnement. Elle possède toutes les commodités d'une maison traditionnelle moderne. La surface habitable est de 100 m2, ce qui est idéal pour une famille moyenne soucieuse de respecter l'environnement et faire des économies d'énergie. Les meubles présents dans la maison sont recyclables à 98 % et fabriqués à partir de matériaux recyclés.

L'énergie est fournie par 60 m^2 de panneaux solaires amorphes qui recouvrent le toit. La puissance électrique obtenue est d'environ 4 kW/h (selon l'ensoleillement). La lumière naturelle est diffusée dans la maison grâce à des puits de lumière situés dans le toit.

http://www.ecoflolife.com/floating-house

contrairement à cette nouvelle technologie qui n'a besoin que d'un seul circuit de contrôle.

Les deux couches différentes de la cellule solaire se comportent comme une cellule classique, facile à installer. Cependant les deux couches permettent d'absorber directement différentes parties du spectre lumineux du Soleil. Les premiers tests ont montré que la nouvelle cellule solaire pouvait atteindre une efficacité de 30 à 35 %, cependant la Pérovskite a tendance à se dégrader rapidement dans l'environnement.

http://news.mit.edu/2015/tandem-solar-cell-0324

L'éolienne volante de Google

Google a l'intention de fabriquer des éoliennes innovantes dans le cadre du programme « Project Makani ». Le laboratoire Google X a développé une éolienne volante « Airborne Wind Turbine » ressemblant à un avion dont l'envergure atteint 25 mètres. Elle est capable de capter le vent pour produire de plus grandes quantités d'énergie. Des tests sont effectués à la frontière entre la Californie et le Nevada. Durant le vol, le système de turbines, qui est relié à la terre par un câble, atteindra une hauteur de 450 mètres selon une trajectoire circulaire. Cette technique permet d'actionner les 4 turbines (éoliennes) présentes sur les ailes. Le principe retenu est celui du cerf-volant, qui est lancé dans l'air et retenu par un câble, exploitant ainsi les différents courants d'air pour faire tourner les éoliennes et produire de l'énergie.

L'idée de Google, de construire des éoliennes volantes, est dû au fait que la vitesse du vent est plus grande et plus constante à une altitude élevée par rapport au niveau du sol.

L'éolienne volante du projet « Makani » permet de capter une « quantité » de vent à un moindre coût, car elle se trouve à une altitude comprise entre 80 et 350 m. Cela élimine 90 % des matériaux d'une éolienne traditionnelle qui doit être montée sur un mat imposant et donc couteux. Chaque éolienne volante génère 50 % d'énergie en plus avec une puissance atteignant 600 kW.

www.google.com/makani/technology

L'énergie du mouvement des marées dans les lagunes : le projet

Le ministère britannique de l'énergie prévoit de financer le projet « Lagoon power plants ». Il s'agit de la construction des premières centrales au monde qui utilisent le mouvement des marées dans les lagunes. Depuis 50 ans, il existe des centrales qui exploitent le mouvement des marées, mais aucune n'est installée dans une lagune. Quatre centrales devraient être construites au Pays de Galles, une dans le Somerset et une à Cumbria. Elles captureront les mouvements des marées en entrée et les restitueront en sorties derrière de gigantesques barrages, en utilisant la masse de l'eau pour faire tourner des turbines.

Le projet de Swansea, au Pays de Galles, sera le plus couteux (1 milliard de livres) et devrez alimenter 150 000 foyers à des prix très bas. Le projet sera mis en œuvre par Tidal Lagoon Power et permettra de produire de l'énergie à des coûts très contenus. Les 6 centrales du projet devraient générer 8 % de l'électricité du Royaume-Uni.

Cependant le projet doit surmonter les craintes des écologistes qui restent sceptiques face à la grandeur des structures à mettre en œuvre. De même les pêcheurs craignent que ce projet fasse fuir les poissons ailleurs.

www.tidallagoonswanseabay.com

TRANSFORMEZ LE RASPBERRYPI EN ÉMETTEUR FM

Dans cet article, nous enrichissons le système de radiodiffusion en streaming via Internet à base de RaspberryPi décrit dans le précédent numéro 135, en lui adjoignant une carte faisant office d'émetteur FM avec des caractéristiques professionnelles.

de Marco MAGAGNIN

Par le passé, les radioamateurs commençaient leur formation en réalisant leur premier récepteur radio. De nos jours à l'ère digitale, nous avons suivi le même chemin en vous offrant dans le précédent numéro **135** d'**Electronique et Loisirs Magazine** un projet de **radiodiffusion en streaming via Internet** à base de **RaspberryPi**.

Le projet visait à diffuser à travers un haut-parleur soit les stations de radio disponibles en streaming sur le web, soit vos propres compilations de chansons enregistrées sur une clé USB. Nous reviendrons plus en détail sur cette dernière partie du projet.

L'interface « utilisateur » avait un double objectif, elle se composait d'un afficheur LCD et de boutons permettant d'interagir avec les fonctionnalités disponibles dans le menu. En particulier l'interface permettait le choix des stations, le réglage du volume, etc. comme avec une radio « traditionnelle », mais offrait en plus la possibilité de gestion à distance via une interface web simple d'utilisation.

Il nous est paru évident que les amateurs passionnés voulaient disposer d'un émetteur FM aux caractéristiques professionnelles, afin de propager leur voix dans le milieu radioamateur.

Pour rester fidèle à cette philosophie, nous vous proposons l'**intégration d'un transmetteur FM** au projet de radiodiffusion en streaming via Internet. Reportez-vous donc au numéro 135 pour l'étude détaillée de ce dernier, car nous réutilisons dans cet article la plupart des composants du système de radiodiffusion en streaming.

Dans un prochain article, nous vous décrirons une solution permettant de gérer l'ensemble du projet par l'intermédiaire d'Internet en Wi-Fi, avec une application pour smartphones. Le module que nous vous présentons est le **FLEXMOD-DB2** de la société TECNOROLL BMB. Ce module est monté sur une carte spécialement conçue pour le RaspberryPi.

Comme nous allons le constater, ce module a été conçu pour offrir aux amateurs de Hi-Fi une solution de qualité professionnelle pour un coût modéré.

Afin de garantir un haut niveau de qualité audio, nous avons prévu que le module fonctionne avec des signaux d'entrée au **format audio** « **I2S** » et/ou « **S/PDIF** ».

Oui, mais quel est le rapport entre la norme audio « I2S » et le RaspberryPi ?

Eh bien c'est l'occasion d'introduire une nouvelle fonctionnalité du RaspberryPi peu connue, il s'agit de la gestion de signaux audio de qualité.

Comme vous pouvez le voir en figure 1, le RaspberryPi comporte une série de broches indiquées par la flèche rouge. Cet endroit est conçu pour recevoir un connecteur à 2 x 4 broches, il s'agit du connecteur « **P5** » du RaspberryPi. Ce connecteur permet de **transmettre des signaux audio de haute qualité** à la norme « **I2S** ».

En figure 2, vous pouvez voir le brochage du connecteur « P5 ». Pour une utilisation en tant que sortie audio « **I2S** », les lignes « **CLK** », « **FS** » (Frame Sync) et « **DOUT** » sont utilisées. La broche « DIN » est une entrée microphone qui n'est pas mise en œuvre pour cette application.

La norme « **I2S** » signifie « **Inter IC Sound** » (Integrated Interchip Sound), c'est un **bus standard de communication** de type **PCM** (Pulse Code Modulation) inventé par **Philips** en 1986.

Une des caractéristiques du bus « I2S » est de maintenir séparés les canaux d'horloge et de données lors de la transmission du signal audio, de manière à **minimiser la gigue**.

C'est un phénomène typique des systèmes de communication qui comprennent un signal d'horloge dans un flux de données. La gigue (« jitter ») est le phénomène de **fluctuation d'un**

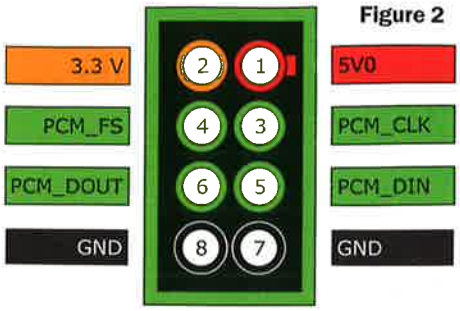


Figure 2

signal, celle-ci peut être un glissement de phase ou une dispersion temporelle. Elle entraîne des **erreurs** en sortie.

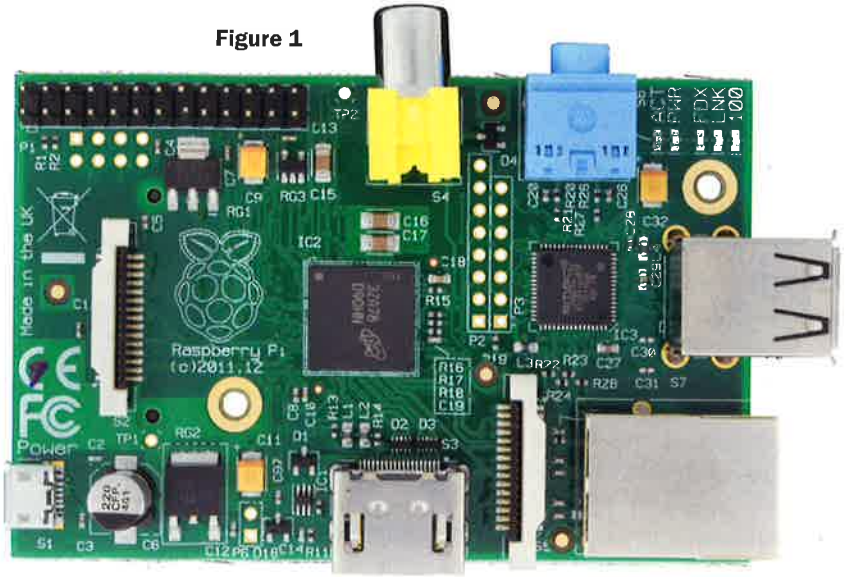
Pour rappel, le bus « **I2S** » est l'**interface utilisée dans les lecteurs CD**, et dans 99 % des appareils audio numériques professionnels ou non.

Le bit d'horloge (CLK) pulse une fois pour chaque bit de données contenu dans le flux. La fréquence du signal d'horloge « CLK » est le produit de la fréquence d'échantillonnage par le nombre de bits par canal (résolution) et par le nombre de canaux.

Par exemple, pour un lecteur de CD audio, avec une fréquence d'échantillonnage de 44,1 kHz et une résolution de 16 bits et 2 canaux (stéréo)), la fréquence d'horloge « CLK » est de : 44,1 kHz × 16 × 2 = 1,4112 MHz.

La ligne « FS » (Frame Sync) indique à quel canal les données transmises correspondent, car les données des 2 canaux sont transmises sur la même ligne. Pour être précis, un niveau logique bas (LOW) sur la ligne « FS » indique que les données transmises sont celles du canal gauche, alors qu'un niveau haut correspond au canal droit. La ligne « DOUT » transporte le signal audio échantillonné.

Figure 1



Le module FLEXMOD-DB2

C'est un module de très petites dimensions, malgré toutes les fonctions de traitement du signal, d'audio et de transmission FM dont il dispose (voir la figure 3). Vous pouvez y voir les principales sections du module qui seront décrites plus loin, le **processeur FPGA** (flèche n° 1), le **DAC** (flèche n° 2) et l'**oscillateur** fonctionnant à **1 GHz** (flèche n° 3).

En électronique analogique traditionnelle, les fonctionnalités équivalentes que fournit ce module auraient requis la construction d'au moins deux appareils dans un châssis de 19 pouces, à savoir un modulateur FM et un codeur RDS (Radio Data System).

Le module **FLEXMOD-DB2** est une version simplifiée de la famille des modules FLEXMOD destinés au marché professionnel et utilisés dans les transmissions de radio FM Broadcasting. La version que nous vous présentons dans cet article est d'un coût à la portée du monde amateur, c'est pour cela que ce module ne dispose pas d'entrée MPX (multiplex externe) et SCA (Subsidiary Comunication Autority).

Par conséquent, ces types d'entrées ne sont pas gérés par le firmware (programme) du module. Ces entrées ne sont absolument pas nécessaires pour une utilisation amateur et, au contraire, leur présence conduirait à augmenter le prix du projet. Pour les mêmes raisons, ainsi que pour une utilisation pratique, il n'est pas nécessaire d'avoir une entrée GPS.

Dans la série des modules professionnels, l'utilisation comme horloge du signal de référence 10 MHz généré par les satellites GPS garantit une précision du signal de sortie de l'ordre de 1 Hertz. Cela ne présente aucun intérêt pour un amateur.

L'avantage de choisir ce module est qu'en plus de son prix abordable, il offre la possibilité d'expérimenter et d'acquérir les connaissances de la transmission radio FM directement transposables au monde professionnel.

Le cœur « logique » de ce module est le processeur **FPGA Xilinx Spartan 3**. C'est un **DSP** (processeur de signal numérique) qui synthétise la majeure partie des processus logiques et en particulier les opérations mathématiques nécessaires pour générer les différents signaux qui composent la bande FM stéréo.

Le processeur FPGA dispose de 47000 unités logiques et de 126 blocs DSP. L'implémentation des fonctions occupe 86 % de la capacité du processeur.

Grâce au firmware particulièrement sophistiqué du processeur FPGA, ce module permet d'assurer la conformité avec les normes d'émissions, quelle que soit la qualité du signal analogique présent sur l'entrée.

La mise en œuvre de ces mêmes fonctionnalités avec une technologie analogique ou avec des modules numériques séparés ne comportant pas des fonctions de contrôle exigerait, pour une utilisation correcte, beaucoup plus de compétence que celles requises pour ce module. De plus il faudrait disposer d'outils de développement et d'instruments de mesure qui seraient très couteux et donc non accessibles pour un amateur.

Heureusement, l'intégration des fonctions de contrôle permet de réaliser un transmetteur avec des caractéristiques professionnelles sans avoir besoin de posséder une haute expertise et un équipement coûteux. De plus le processeur FPGA intègre des fonctions particulières comme la conversion du DAC et la génération du signal numérique d'horloge à 1 GHz (voir la figure 3).

Maintenant, nous allons étudier le diagramme de fonctionnement du module, représenté en figure 4, et dont la tâche fondamentale est de générer un signal FM stéréo à partir d'un signal d'entrée audio.

Le module dispose de **deux types d'entrées** : une entrée au standard « **I2S** » qui est utilisée pour **prélever le signal audio** du RaspberryPi, et une entrée pour **fibre optique** à la norme « **S/PDIF** » qui accepte la connexion d'une fibre optique sur le connecteur visible en figure 5.

« **S/PDIF** » est l'acronyme de « **Sony/Philips Digital Interface Format** ». C'est un format qui permet de transmettre et de recevoir des données audio numériques échantillonnées de 16 bits à 24 bits (24 bits pour ce qui concerne le FLEXMOD-DB2) sur un seul conducteur de type électrique ou optique. Le mode de transmission est très similaire à celui mis en œuvre dans le bus « I2S ».

La principale différence réside dans le fait que les signaux d'horloge (CLOCK)

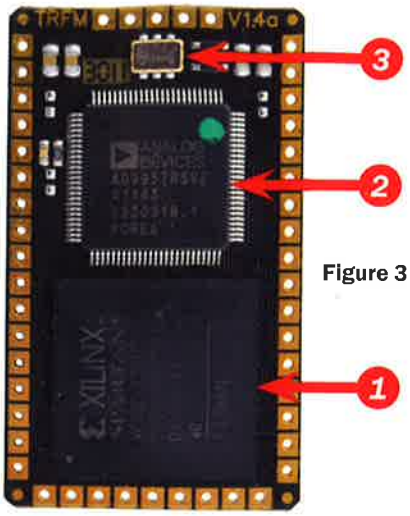


Figure 3

et les données audio (canal gauche et canal droit) sont multiplexés dans un seul conducteur.

Dans le dispositif récepteur, il est alors nécessaire d'implémenter des procédés de traitements supplémentaires afin d'extraire les différents signaux.

Le module dispose d'un bus série permettant la configuration et la personnalisation des fonctionnalités du logiciel du module, entre autre la gestion du canal RDS (Radio Data System), et les informations sous forme de texte que nous décrirons plus loin.

Revenons sur les entrées audio, en se référant au schéma de la figure 4. Les deux entrées « I2S » et « S/PDIF » sont dirigées vers le bloc nommé « **PCM** » (Pulse Code Modulator) numéroté 1 sur le synoptique.

Celui-ci est entièrement réalisé en logique FPGA. La fonction du « **PCM** » est de **traiter le signal** d'entrée (I2S ou S/PDIF) afin d'extraire les deux canaux audio stéréo échantillonnés sur **24 bits**, L (gauche) et R (droit).

Deux autres étages de traitement, nommés « **Resampler** » et numérotés 2 sur le synoptique, permettent le **ré-échantillonnage des signaux** d'entrée pour les adapter à la fréquence requise pour la transmission FM.

Les figures 6 et 7 représentent les diagrammes des signaux de sortie des modules de ré-échantillonnage, dans le cas d'une fréquence d'entrée identique à la fréquence de sortie (figure 6) et

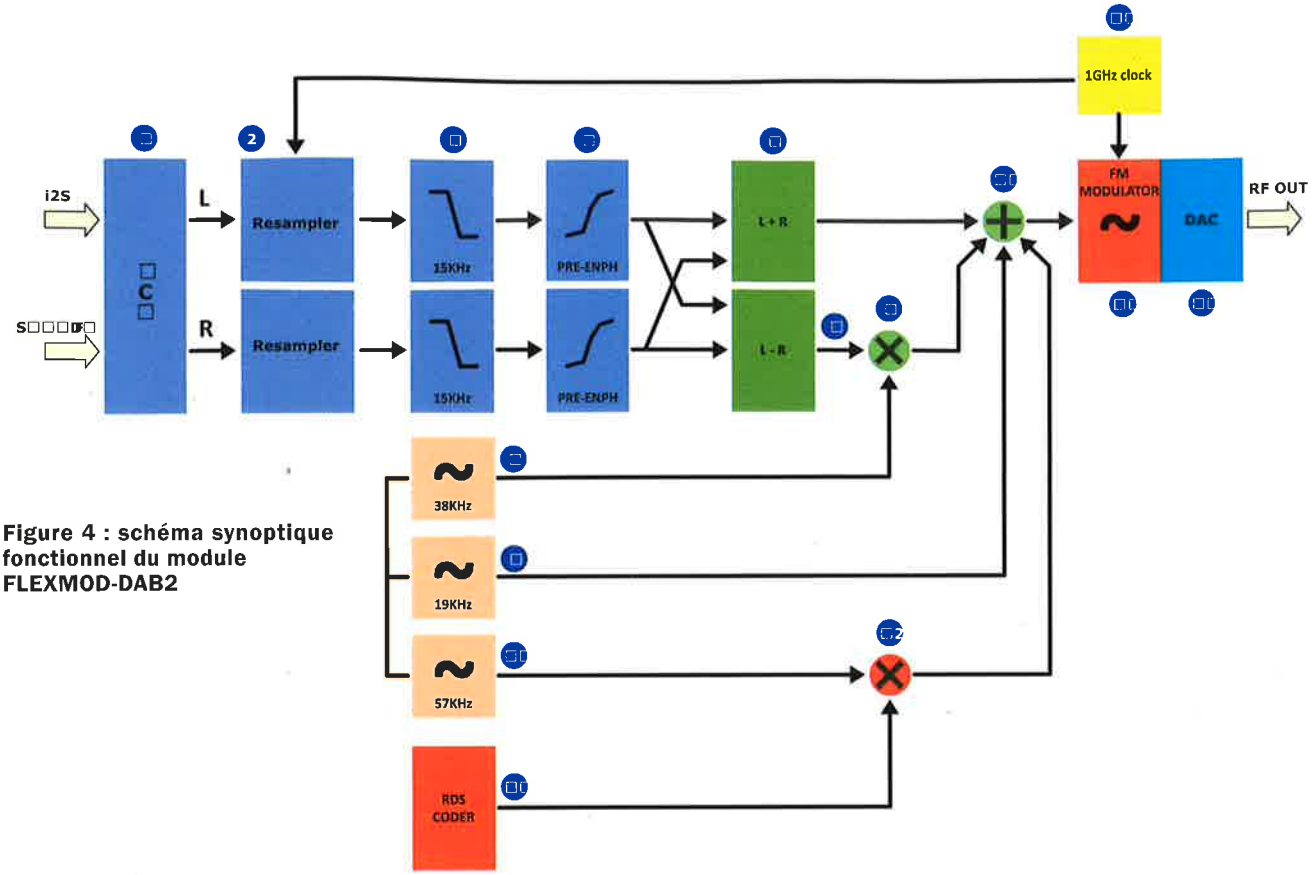


Figure 4 : schéma synoptique fonctionnel du module FLEXMOD-DAB2

dans le cas d'une fréquence d'entrée inférieure à celle de sortie (figure 7).

Les graphiques se réfèrent à un échantillonnage du signal réglé à 48 kHz. En réalité les modules de ré-échantillonnage, qui sont réalisés en logique FPGA, sont capables de stabiliser et de fixer la fréquence d'échantillonnage de manière automatique en fonction des caractéristiques du signal d'entrée.

Chaque canal est ensuite traité par un filtre passe-bas (blocs numérotés 3 sur le synoptique) qui transmet des signaux sans atténuation jusqu'à une fréquence de 15 KHz, et avec une atténuation de 96 dB au-dessus de 16 kHz.

Cette atténuation est impossible à réaliser avec des circuits analogiques traditionnels.

Figure 5

Cette forte atténuation élimine le risque d'interférence avec le signal de référence « pilote » de 19 kHz (inaudible et de faible amplitude) qui sert à restituer la sous-porteuse lors de la réception afin d'effectuer correctement la démodulation.

En figure 8 vous pouvez voir le diagramme de réponse du filtre passe-bas. Les deux signaux ainsi obtenus sont traités par deux blocs de préaccentuation (numérotés 4 sur le synoptique).

Ils permettent d'améliorer les hautes fréquences du signal audio afin d'augmenter le rapport « signal/bruit » du signal modulé. Cette accentuation sera ensuite réduite dans le récepteur.

Le rapport « signal/bruit » se détériore avec l'augmentation de la fréquence du signal, en rehaussant les hautes fréquences cela diminue l'incidence des bruits sur ces fréquences et donc sur le signal en général.

À ce stade, le module commence la mise en forme du signal FM stéréo MPX (Multiplex) dont le spectre est visible en figure 9.

Les diverses informations sont multiplexées dans le signal dont la largeur de la bande passante est de 100 kHz.

Dans la première partie du spectre, est inséré un signal audio monocanal, afin de maintenir la compatibilité avec les récepteurs FM mono plus anciens et qui sont précisément monophoniques. **Le signal est obtenu en additionnant les deux canaux L et R** pour obtenir un **signal mono** qui occupe la largeur du spectre MPX de 30 Hz à 15 kHz (numéro 5 sur le synoptique de la figure 4).

Ensuite, le signal sinusoïdal provenant du générateur à **19 kHz qui constitue la fréquence de référence** pour le traitement du signal stéréo est inséré (numéro 6 sur le synoptique de la figure 4).

Le **signal stéréo est alors composé par la différence** entre le canal gauche (L) et droit (R) transmis par une **porteuse à 38 kHz synchronisée** avec la fréquence de référence de **19 kHz**. La sous-porteuse en phase de 38 kHz est obtenue en multipliant par deux la fréquence du signal de référence de 19 kHz (numéro 7 sur le synoptique de la figure 4).

Figure 6

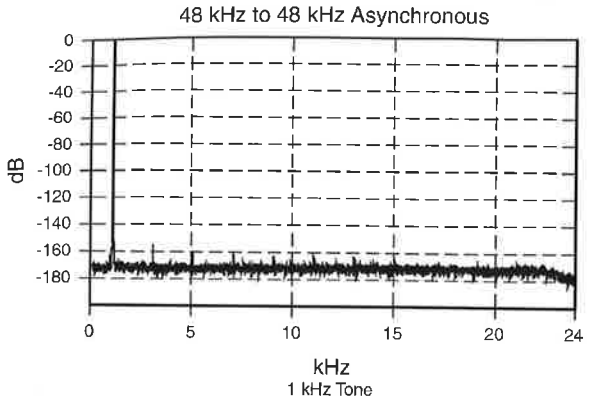
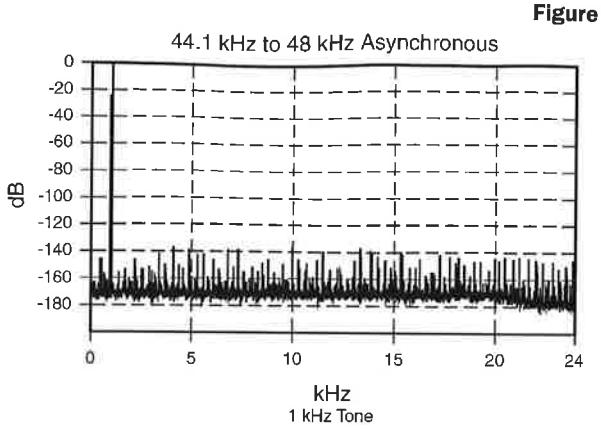


Figure 7



Dans cette partie du spectre MPX FM stéréo, qui s'étend de 23 kHz à 53 kHz, le signal obtenu par la soustraction du canal gauche et droit (L - R) est modulé.

La méthode utilisée pour moduler la bande passante autour de la fréquence de 38 kHz est la DSB-SC (double sideband suppressed carrier transmission), c'est-à-dire la « modulation d'amplitude à porteuse supprimée ». Cela permet d'augmenter la puissance d'émission (numéro 8 sur le synoptique de la figure 4).

Avec cette méthode, le signal final est obtenu à partir d'un mélangeur qui multiplie le signal modulé d'entrée par le signal de la porteuse (numéro 9 sur le synoptique de la figure 4).

De cette façon, les fréquences de la modulation d'amplitude d'origine sont réparties symétriquement au-dessus et au-dessous de la fréquence porteuse (38 kHz), cette dernière étant pratiquement supprimée.

Il en va de même pour le **signal RDS** (Radio Data System). Le signal RDS (numéro 10 sur le synoptique) permet la **transmission de données sous forme de texte** au récepteur, qui sont ensuite affichées sur l'écran de la radio. Les informations concernent le nom de la station, le titre du morceau, etc.

Le signal RDS est multiplexé (numéro 12 sur le synoptique) dans le spectre MPX (numéro 11) avec la même méthode que celle utilisée pour le signal (L - R), à une fréquence d'environ 57 kHz. Cette sous-porteuse (numéro 13) est obtenue par le triplement de la fréquence du signal pilote de 19 kHz.

Le signal audio obtenu a une bande passante de 100 kHz, comme vous pouvez le voir en figure 10. Maintenant, nous allons utiliser ce signal pour moduler la transmission FM. Cette opération est réalisée par le bloc « FM Modulator » piloté par l'oscillateur à 1 GHz (numéro 14 sur le synoptique de la figure 4). C'est d'ailleurs la fréquence de référence pour les modules de ré-échantillonnage.

L'interpolation, qui est effectuée par le processeur FPGA (numéro 15 sur le synoptique de la figure 4), entre le signal audio, la fréquence de l'oscillateur et le canal FM (fréquence de sortie) est configurée par l'intermédiaire des fonctions de gestion du module. Cela permet d'obtenir un signal modulé à la fréquence souhaitée, qui est converti en un signal analogique par le dernier étage DAC (numéro 16) du module FLEXMOD-DB2.

Etant donné que la **sortie HF** présente un niveau de **0 dBm** (1 mW), un morceau de fil relié au connecteur

Figure 8

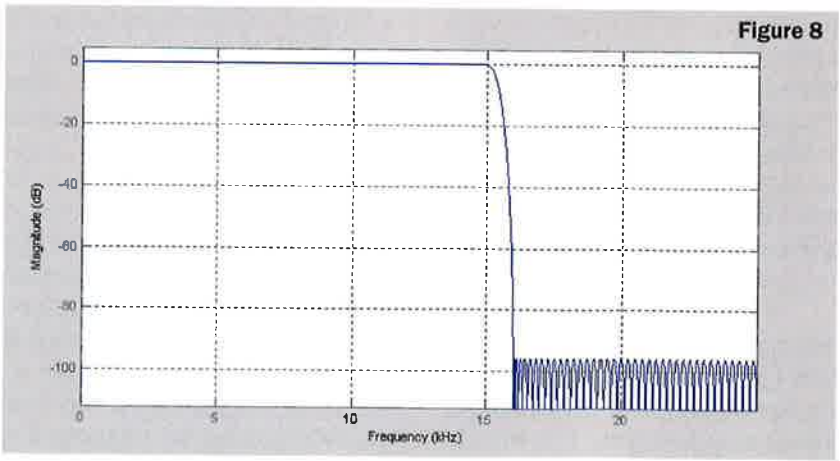
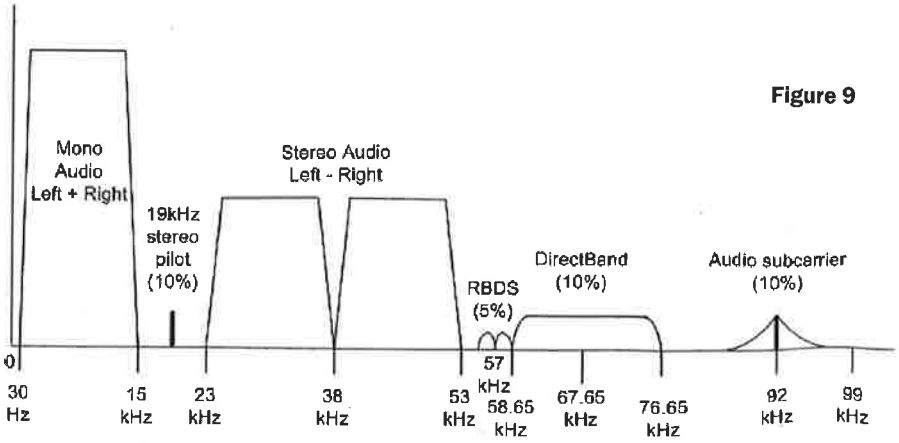


Figure 9



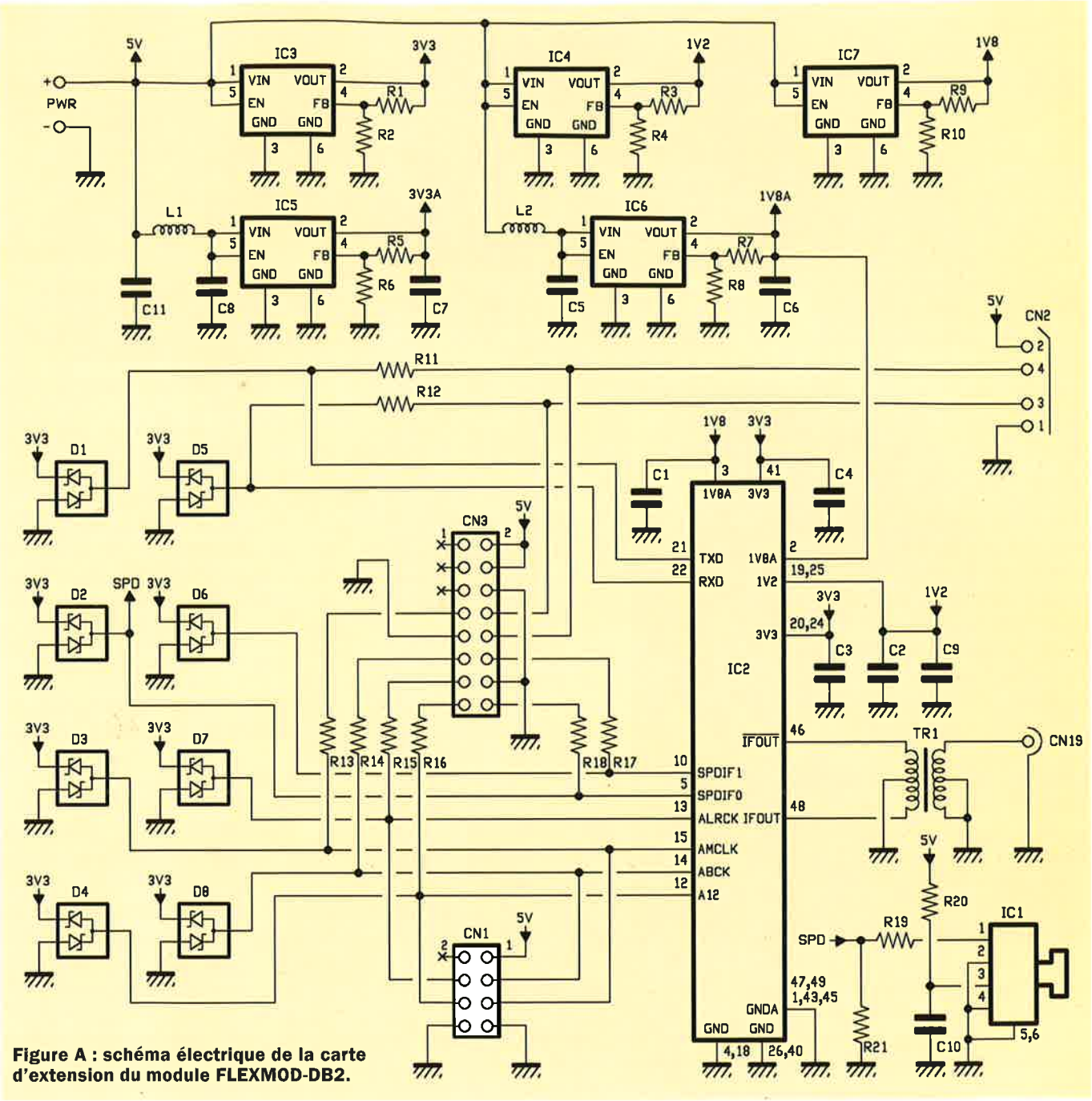


Figure A : schéma électrique de la carte d'extension du module FLEXMOD-DB2.

d'antenne est plus que suffisant pour transmettre un signal FM sur une distance de 30 à 50 m. Pour augmenter la portée de l'émetteur, il suffit de relier à la sortie un amplificateur HF externe.

Ces caractéristiques, bien que d'un niveau « amateur », permettent d'utiliser ce module pour créer une station « relais » de signaux FM, par exemple dans un relief montagneux où les émetteurs originels ne passent pas. Pour mettre en œuvre cette solution, il suffit d'une ligne ADSL à partir de laquelle est prélevée la diffusion en streaming d'une station donnée, et appliquer le signal à l'entrée de notre module positionné à l'endroit souhaité.

Une autre application possible est la diffusion d'informations de guidage, par exemple dans un musée ou un centre commercial. Dans ce cas, les capacités de gestion multifréquence du RDS sont très utiles, car elles permettent de syntoniser le même émetteur sur différentes fréquences, par exemple les différentes parties d'un musée ou les différentes zones d'un centre commercial.

Cela nous permet de dire que la gestion et la configuration du module FLEXMOD-DB2 s'effectuent au moyen d'une application intégrée dans le firmware et sont accessibles via le port série. Nous étudierons par la suite la gestion de l'application via le port série en mode terminal distant (effectuée via le RaspberryPi), afin de voir ce qui se passe du côté du récepteur en ce qui concerne le traitement du signal stéréo FM.

Après avoir obtenu le spectre MPX du signal FM, il faut extraire les signaux

Plan de montage de la carte d'extension du FLEXMOD-DB2

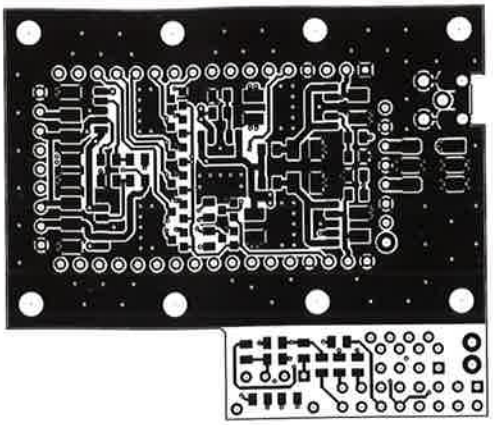


Figure B : circuit imprimé à l'échelle 1 : 1 côté soudures (face inférieure) de la carte d'extension du module FLEXMOD-DB2. Attention, les composants CMS sont soudés sur la face inférieure.

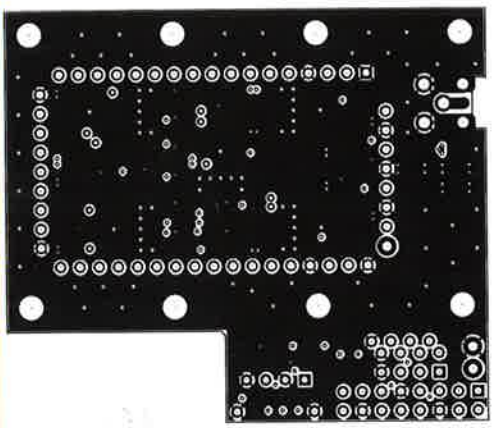


Figure C : circuit imprimé à l'échelle 1 : 1 côté face supérieure. Le module FLEXMOD-DB2 est soudé sur la face supérieure.

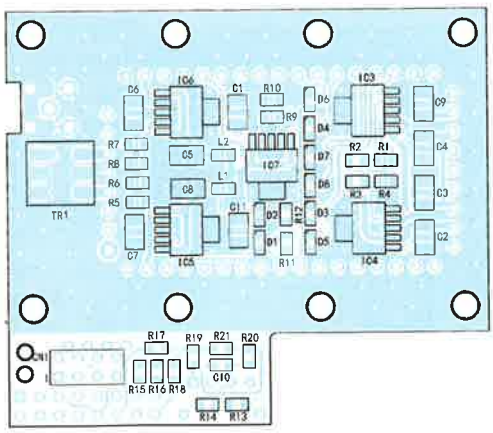
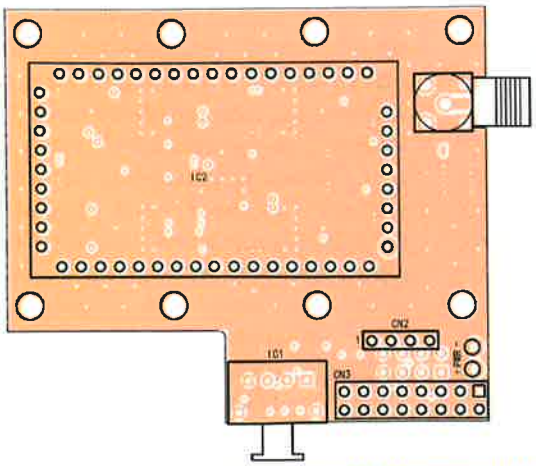


Figure D : plan de câblage des composants côté face inférieure.

Figure E : plan de câblage des composants côté face supérieure.



Liste des composants de la carte d'extension du FLEXMOD-DB2

R1..... 8,2 kΩ boîtier 0805
R2..... 4,7 kΩ boîtier 0805
R3..... 0 Ω boîtier 0805
R4..... non utilisée
R5..... 8,2 kΩ boîtier 0805
R6..... 4,7 kΩ boîtier 0805
R7..... 3,3 kΩ boîtier 0805
R8..... 6,8 kΩ boîtier 0805
R9..... 3,3 kΩ boîtier 0805
R10.... 6,8 kΩ boîtier 0805
R11.... 470 Ω boîtier 0805
R12.... 470 Ω boîtier 0805
R13 à R18 non utilisées
R19.... 1,2 kΩ boîtier 0805
R20.... 10 Ω boîtier 0805
R21.... 2,2 kΩ boîtier 0805

C1..... 47 µF/6,3 V céramique boîtier 1210
C2..... 47 µF/6,3 V céramique boîtier 1210
C3..... 47 µF/6,3 V céramique boîtier 1210
C4..... 47 µF/6,3 V céramique boîtier 1210
C5..... 47 µF/6,3 V céramique boîtier 1210
C6..... 47 µF/6,3 V céramique boîtier 1210
C7..... 47 µF/6,3 V céramique boîtier 1210
C8..... 47 µF/6,3 V céramique boîtier 1210
C9..... 47 µF/6,3 V céramique boîtier 1210
C10.... 100 nF 16 V céramique boîtier 0805
C11.... 47 µF/6,3 V céramique boîtier 1210

IC1..... TORX173
IC2..... FLEXMOD-DAB2
IC3..... TPS73601
IC4..... TPS73601
IC5..... TPS73601
IC6..... TPS73601
IC7..... TPS73601
L1..... 220 mH/100MHz
L2..... 220 mH/100MHz
D1..... BAT754S boîtier SOT23
D2..... BAT754S boîtier SOT23
D3..... BAT754S boîtier SOT23
D4..... BAT754S boîtier SOT23
D5..... BAT754S boîtier SOT23
D6..... BAT754S boîtier SOT23
D7..... BAT754S boîtier SOT23
D8..... BAT754S boîtier SOT23
TR1.... ADT1-1WT+ (www.minicircuits.com)

Divers

Connecteur SMA pour circuit imprimé
Barrette femelle 2 pôles
Barrette femelle 4 pôles (x2)
Barrette mâle 4 pôles (x2)
Barrette mâle/femelle 8 pôles (x2)
Dissipateur

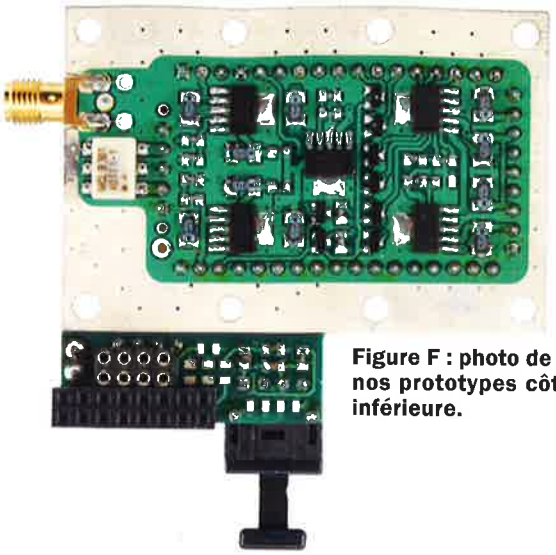


Figure F : photo de l'un de nos prototypes côté face inférieure.

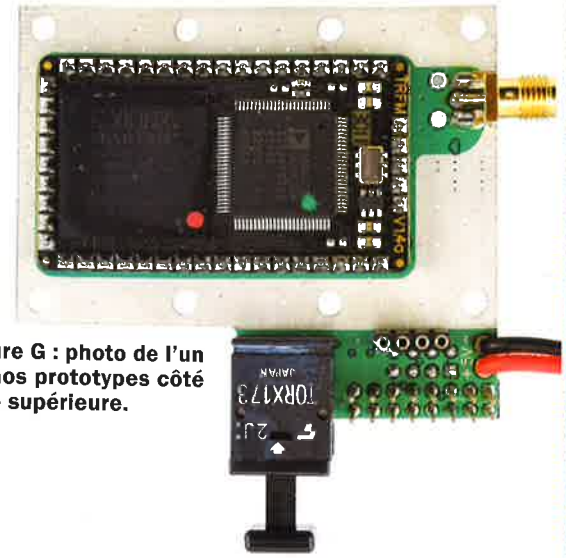


Figure G : photo de l'un de nos prototypes côté face supérieure.

(L + R) et (L - R) qui seront ensuite retraités pour obtenir le canal L et R. Le **canal gauche** est obtenu à partir de la **somme** des canaux **(L + R) + (L - R) = 2L**.

Le **canal droit** est obtenu à partir de la **soustraction** des mêmes **canaux (L + R) - (L - R) = 2R**.

La carte FLEXMOD-DB2 pour le RaspberryPi

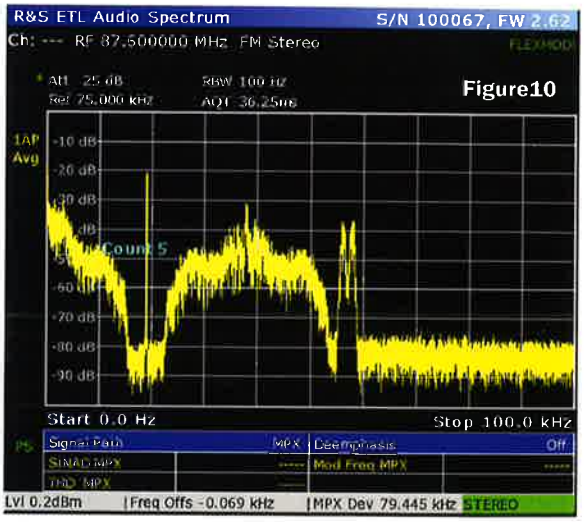
Afin d'intégrer facilement le module FLEXMOD-DB2 au RaspberryPi, nous avons réalisé une carte d'extension qui permet de loger le module, de l'alimenter et de le connecter correctement au RaspberryPi.

L'alimentation doit être stabilisée à **5 VDC** et doit être capable de délivrer un courant **d'au moins 3 A**.

Avec une telle alimentation, il est possible d'alimenter le module et le RaspberryPi en même temps, car le 5 V de l'alimentation est reporté sur les broches 2 et 4 du connecteur GPIO et sur la broche 1 du connecteur P5 (I2S). La tension de 5 V alimente aussi les autres circuits de la carte (IC1 et IC3 à IC7).

Les deux circuits intégrés, **IC5** et **IC6**, fournissent respectivement des tensions particulièrement bien filtrées qui sont le **3,3 V** et le **1,8 V**.

Celles-ci alimentent le **convertisseur DAC** et l'**oscillateur 1 GHz**. Les autres circuits intégrés fournissent des tensions de **1,2 V** (IC4), de **3,3 V** (IC3) et de **1,8 V** (IC7), nécessaires pour le processeur FPGA et les diverses sections du module.



Sur le connecteur **P5**, nommé « **CN1** » sur le schéma électrique de la figure A, sont présents en plus de l'alimentation 5 V sur la broche 1, les signaux du bus « **I2S** », le signal d'**horloge** (CLOCK) sur la broche 3, le signal « **FS** » sur la broche 4 et les données sur la broche 6 « **DOUT** ». La broche « DIN », dans cette configuration, reste inutilisée.

Les diodes **D1 à D8** sont utilisées pour **protéger toutes les entrées** contre d'éventuelles surtensions. Les **lignes séries** du module FLEXMOD-DAB2 sont reliées aux broches **8** et **10** du connecteur **GPIO** du RaspberryPi.

Une des particularités de cette carte est de permettre l'utilisation de l'ensemble des lignes du RaspberryPi.

En particulier, vous pouvez connecter les lignes I2S à travers des résistances de 0 Ω aux broches 7 (CLK), 13 (FS) et 15 (DOUT) du connecteur GPIO.

Cette option vous permet de monter le module sur n'importe quel dispositif, même en position verticale si vous utilisez un connecteur orienté à 90°. Dans cette configuration, bien entendu, le module ne peut plus être utilisé avec le RaspberryPi.

L'assemblage de l'ensemble impose que le module FLEXMOD-DAB2 soit inséré dans son logement prévu sur la carte d'extension. Le tout doit être disposé dans un boîtier en aluminium qui a une double fonction : il permet d'immuniser le module des fréquences parasites et de dissiper la chaleur (voir la figure 11).

Le boîtier en aluminium est constitué par 2 coques qui sont assemblées par 6 vis M3. Le radiateur est quant à lui fixé par 2 vis supplémentaires.

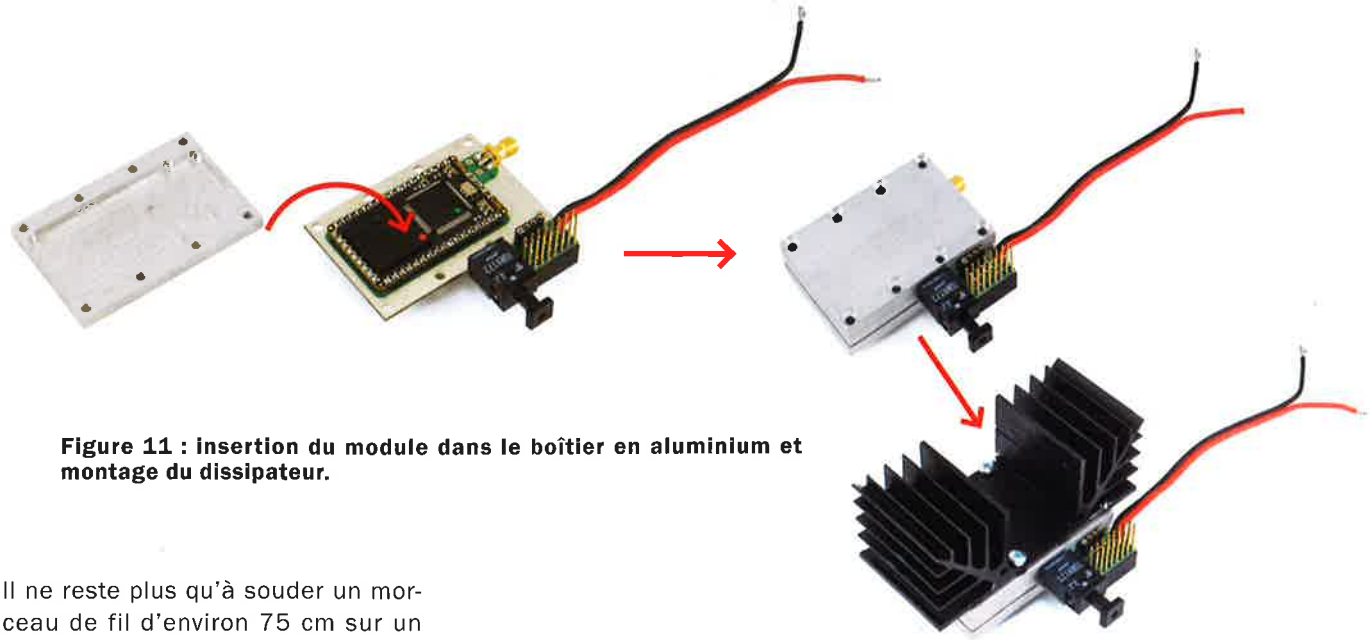


Figure 11 : insertion du module dans le boîtier en aluminium et montage du dissipateur.

Il ne reste plus qu'à souder un morceau de fil d'environ 75 cm sur un connecteur SMA mâle et de connecter l'antenne ainsi obtenue sur la prise SMA du module.

Préparation du RaspberryPi

Nous allons maintenant préparer le RaspberryPi afin d'accueillir la carte avec le module radio. En observant le RaspberryPi, vous remarquerez qu'à côté du connecteur GPIO se trouve un deuxième connecteur disposant de 2 rangées de 4 broches chacune. Il s'agit du connecteur sur lequel est reliée la sortie « I2S » du RaspberryPi (voir la figure 1).

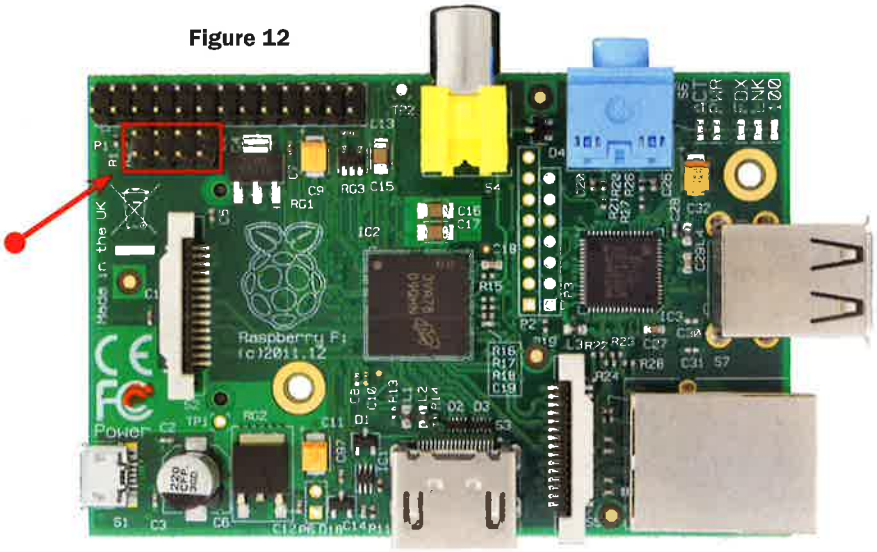
Tout d'abord, vous devez souder à cet emplacement, en faisant attention, un connecteur mâle à 2 rangées de 4 broches au pas de 2,54 mm (voir la figure 12).

Si la carte d'extension « afficheur LCD » est installée, vous devez la retirer avant de souder le connecteur mâle.

Insérez correctement la carte avec le module radio sur le RaspberryPi. Vérifiez qu'aucune surface de la carte d'extension entre en contact (risque de courts-circuits) avec la carte RaspberryPi.

Ensuite, insérez un câble plat sur le connecteur GPIO. Pour connecter le câble plat à la carte « afficheur LCD », vous devez démonter l'afficheur de la carte d'extension.

Figure 12



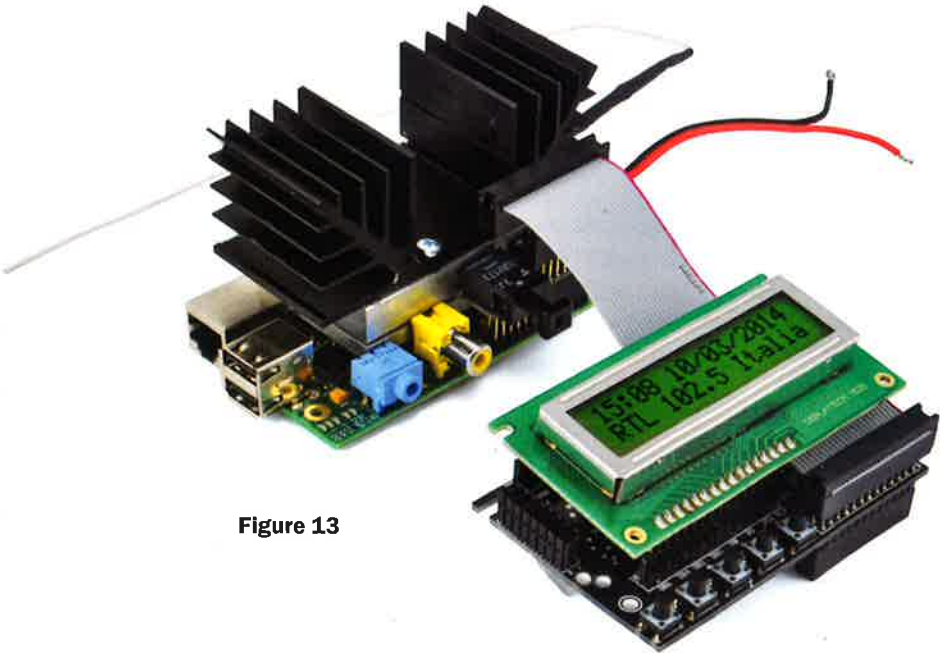


Figure 13

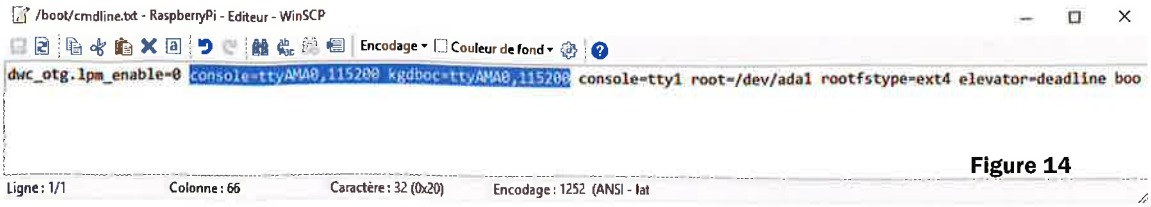


Figure 14

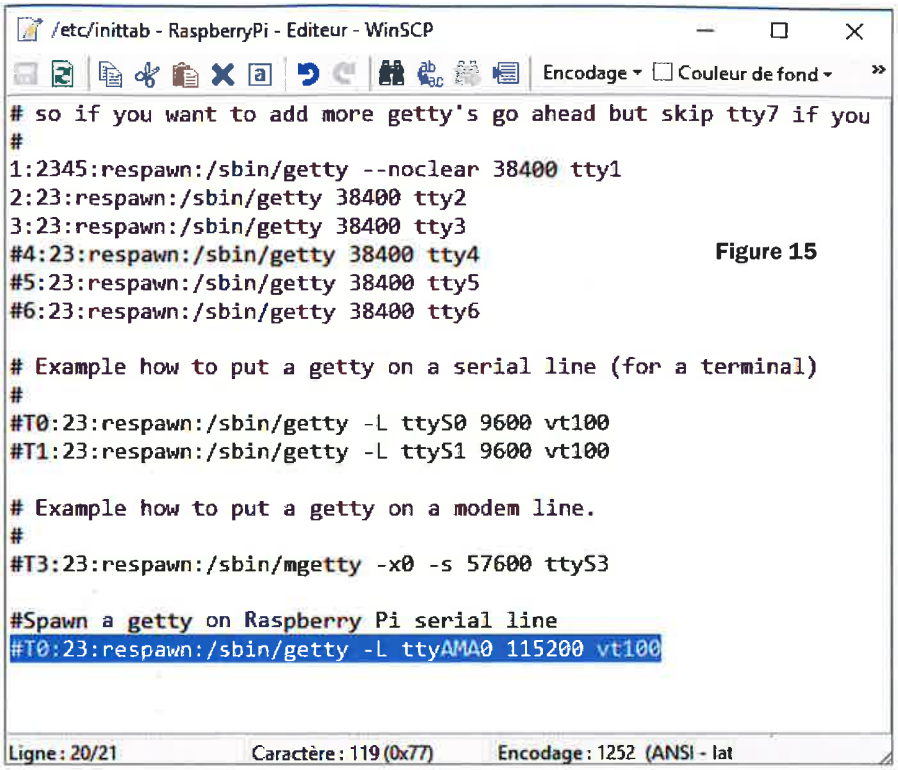


Figure 15

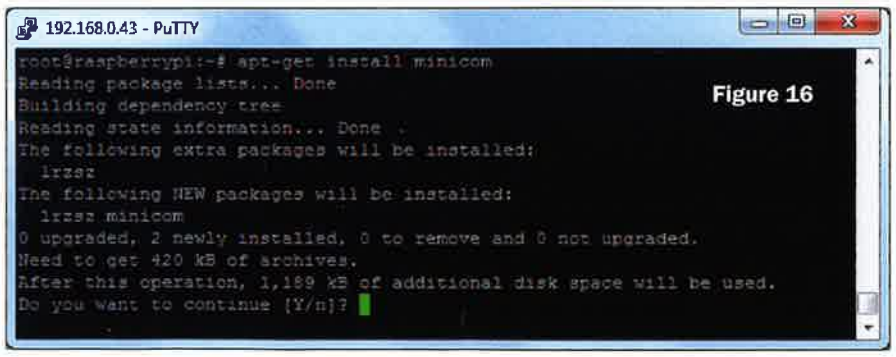


Figure 16

Branchez le câble en respectant l'orientation et remontez l'afficheur. L'ensemble doit correspondre à l'illustration représentée en figure 13.

De la carte d'extension comportant le module FLEXMOD-DAB2, sortent deux fils correspondant à l'alimentation.

Vous devez relier les 2 fils à une alimentation stabilisée de 5V/3A, vous aurez ainsi assez de puissance pour alimenter toutes les cartes y compris le RaspberryPi.

Configuration du RaspberryPi

Nous partons de la **configuration de l'article décrit** dans le précédent numéro **135** « **Ecoutez la radio avec RaspberryPi** ».

Appliquez la tension d'alimentation, et connectez le Raspberry Pi à distance en ouvrant une fenêtre de terminal et le gestionnaire de fichiers à l'aide des logiciels « Putty » et « WinSCP », comme décrit à plusieurs reprises dans les anciens numéros d'Electronique et Loisirs Magazine. Référez-vous aux numéros 123 à 126.

Mettez à jour la distribution à l'aide des commandes :

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

La première étape consiste à débrancher le port série car certaines de ses broches sont dirigées vers le connecteur GPIO, de sorte qu'il ne soit pas utilisé comme console système. Laissez les connexions libres pour pouvoir les relier au module FLEXMOD-DB2.

Ensuite allez dans le dossier « **/boot** » et ouvrez le fichier « **cmdline.txt** ». Dans ce fichier, vous devez supprimer les paramètres de configuration (voir la figure 14) :

```
console=ttyAMA0,115200
kgdboc=ttyAMA0,115200
```

Sauvegardez et sortez de manière habituelle. Enregistrez le fichier et allez dans le dossier « **/etc** », dans lequel se trouve le fichier « **inittab** ». Ouvrez-le et supprimez la ligne suivante :

```
#T0:23:respawn:/sbin/getty -L
ttyAMA0 115200 vt100
```

ou, tout simplement comme nous préférons, mettez la ligne en commentaire à l'aide d'un « **#** » en début de ligne.

Auparavant vous avez ouvert le fichier « **inittab** », vous pouvez optimiser les performances du RaspberryPi en éliminant certaines sessions de terminal entre commentaire, comme le montre la figure 15. Il s'agit des lignes suivantes :

```
#4:23:respawn:/sbin/getty 38400
tty4
#5:23:respawn:/sbin/getty 38400
tty5
#6:23:respawn:/sbin/getty 38400
tty6
```

Ensuite fermez le fichier. Effectuez un **redémarrage pour que les modifications apportées prennent effet.**

Maintenant, nous devons installer l'émulateur de terminal « **TTY** » pour se connecter au port série et atteindre l'application d'administration du module.

Utilisez le paquet « minicom » que vous installez avec la commande (voir la figure 16) :

```
apt-get install minicom
```

Ensuite, vous allez **configurer** le **serveur** « **MPD** » et **installer les pilotes audio** afin que la transmission du streaming audio vers la sortie « I2S » puisse avoir lieu. Vous devez d'abord activer les pilotes nécessaires à la gestion du bus « I2S ».

Tous les pilotes sont déjà inclus dans les dernières distributions de Raspbian, si la mise à jour a été effectuée correctement vous ne devriez rencontrer aucun problème.

Vous pouvez charger les pilotes en utilisant la commande « **modprobe** », **mais nous vous recommandons de configurer les pilotes afin qu'ils soient chargés directement lors du (re) démarrage**, c'est-à-dire lors du « **boot** ».

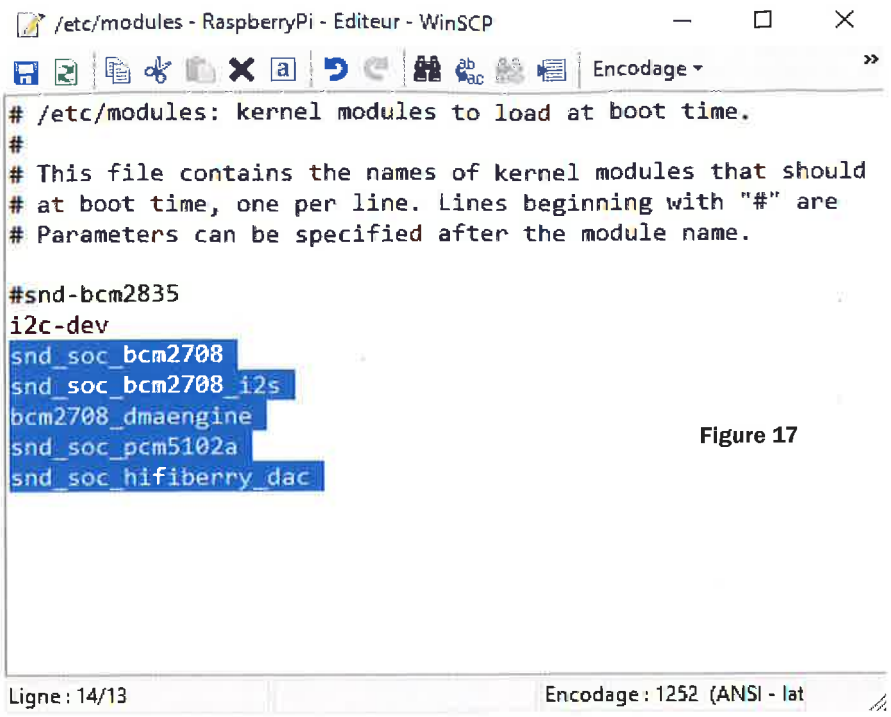


Figure 17

Pour cela, ouvrez le fichier « **/etc/modules** » et ajoutez les lignes suivantes (voir la figure 17) :

```
snd_soc_bcm2708
snd_soc_bcm2708_i2s
bcm2708_dmaengine
snd_soc_pcm5102a
snd_soc_hifiberry_dac
```

Si vous rencontrez des difficultés de reconnaissance de la source « I2S », vous pouvez mettre la ligne suivante en commentaire :

```
#snd_bcm_2835
```

Dans ce cas, vous ne verrez pas le pilote (driver) dans la fenêtre « **alsamixe** », comme décrit ci-après. Enregistrez et fermez le fichier. Redémarrez et ouvrez la fenêtre de terminal, tapez la commande :

```
alsamixer
```

Cela ouvre la fenêtre de gestion du mélangeur « **alsa** », avec la touche F6 vous pouvez faire défiler les différents pilotes reconnus. Vous devez trouver le pilote « PCM » et ensuite celui de gestion du bus « I2S » (voir la figure 18).

Figure 18

Pour trouver le nom de référence du pilote, tapez la commande suivante (voir la figure 19) :

```
aplay -l
```

Figure 19

Vous voyez apparaître le nom du pilote avec lequel il est identifié. Notez le nom du pilote ou copiez-le dans le presse-papier.

Pour « amener » le signal sur l'entrée du bus I2S du module, vous devez configurer le serveur MPD (Music

Player Daemon que nous avons installé dans le précédent numéro 135) en sortie ou « OUTPUT ».

Le fichier de configuration se nomme « **mpd.config** », il est situé dans le dossier « **/etc** ».

Ouvrez le fichier et allez à la ligne contenant le paramètre « **audio_output** ».

Modifiez la ligne (voir la figure 20) en utilisant le nom du pilote que vous avez noté ou copié précédemment.

```
name "sndrpihifiberry [snd_rpi_hifiberry_dac], device 0: HifiBerry DAC HiFi pcm5102a-hifi-0 []"
```

Enregistrez et fermez le fichier, redémarrez le serveur MPD avec la commande suivante :

```
service mpd restart
```

Après l'installation, avec la fenêtre de terminal, vous pouvez accéder à l'application en tapant la commande :

```
minicom -c on
```

Vous devez obtenir le résultat de la figure 22.

Maintenant, ouvrez le moniteur série et utilisez l'application de configuration pour définir la fréquence d'émission.

Donnez un nom RDS à votre station de radio, nous avons nommé notre radio « ELM ». Tapez la commande suivante :

```
minicom -c on
```

À l'invite (« prompt ») du moniteur série tapez :

```
help
```

Faites défiler (juste par curiosité) toutes les options concernant la gestion du module.

Nous étudierons par la suite les différentes possibilités. Pour aujourd'hui, limitez-vous à la définition du nom de la radio, à la fréquence d'émission et à une chaîne de texte défilante.

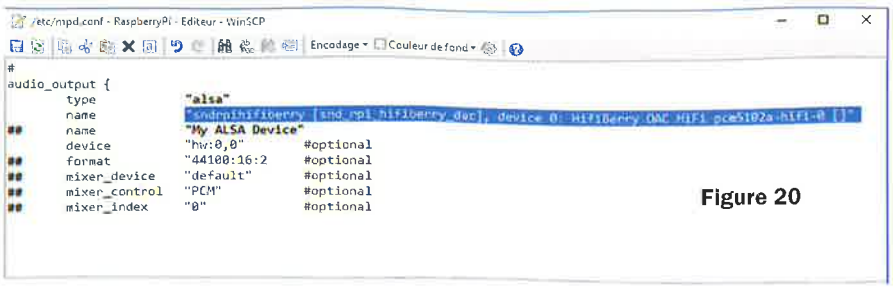


Figure 20

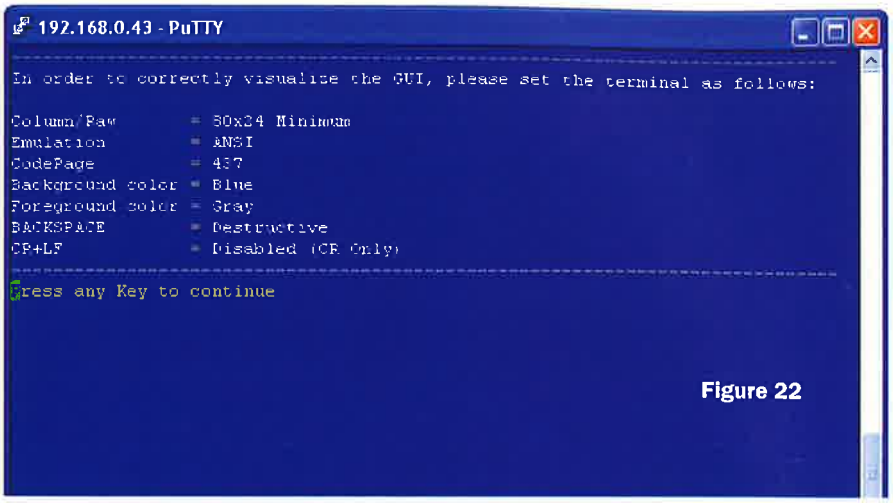


Figure 22

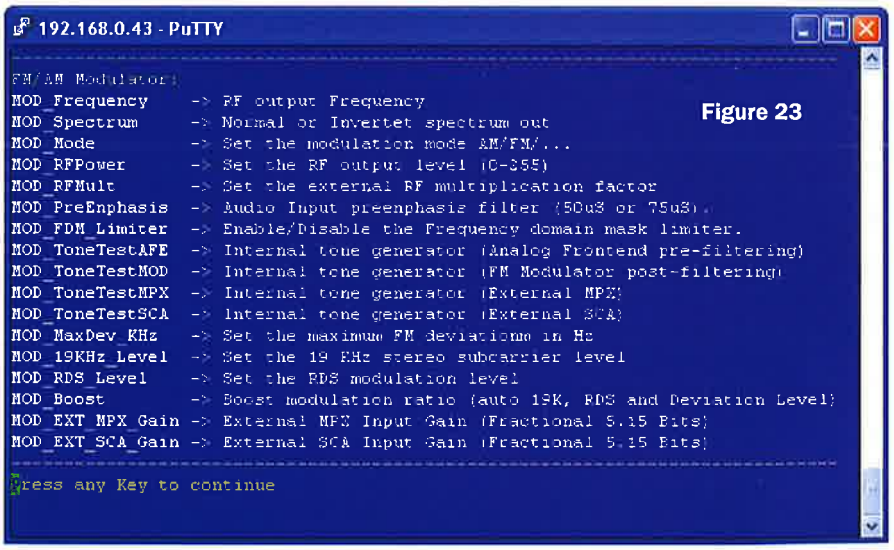


Figure 23

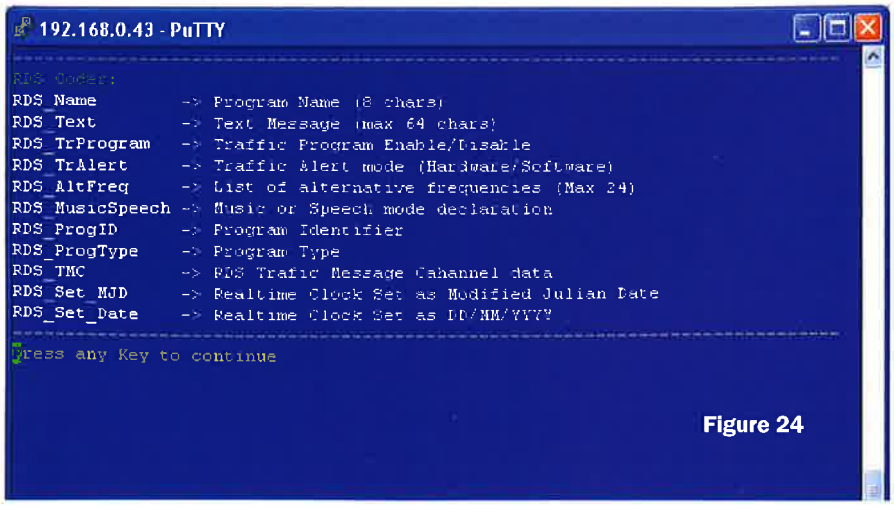


Figure 24

Dans les figures 23 et 24, vous pouvez voir quelques-unes des fonctions disponibles.

Pour modifier vos paramètres, appuyez sur la touche « **ENTER** » pour revenir à l'invite du terminal. À ce stade, tapez :

```
mod
```

Si vous appuyez sur la touche « **TAB** », vous obtenez la liste des commandes qui commencent par « **MOD** », y compris celle qui permet de modifier la fréquence.

À cet égard, nous rappelons que le module est réglé avec une **fréquence de 0 MHz** afin de protéger éventuellement l'amplificateur final, **il ne transmet donc aucun signal**. Tapez la commande suivante, attention au point décimal (voir la figure 25) :

```
MOD_Frequency 87.6
```

Cette fréquence est celle que nous souhaitons, mais vous pouvez choisir une autre valeur selon vos besoins.

Ensuite tapez « **RDS** » et appuyez sur la touche « **TAB** ». Pour définir le nom de la radio, tapez :

```
RDS_Name "ELM"
```

Pour définir le texte défilant, tapez :

```
RDS_Text "Radio privée ELM"
```

Le dernier paramètre à configurer est la source du signal (voir la figure 26) :

```
Source_Select 4
```

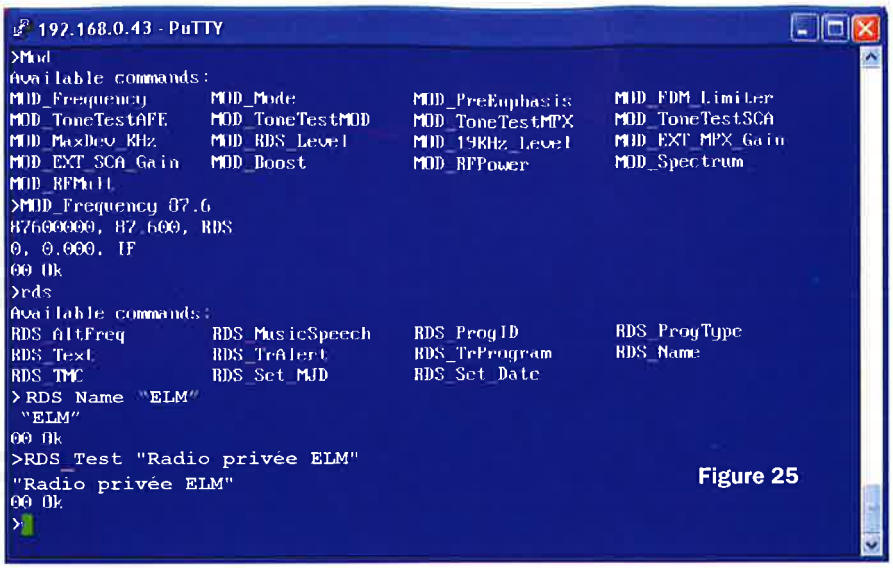
Tout sauvegarder avec la commande :

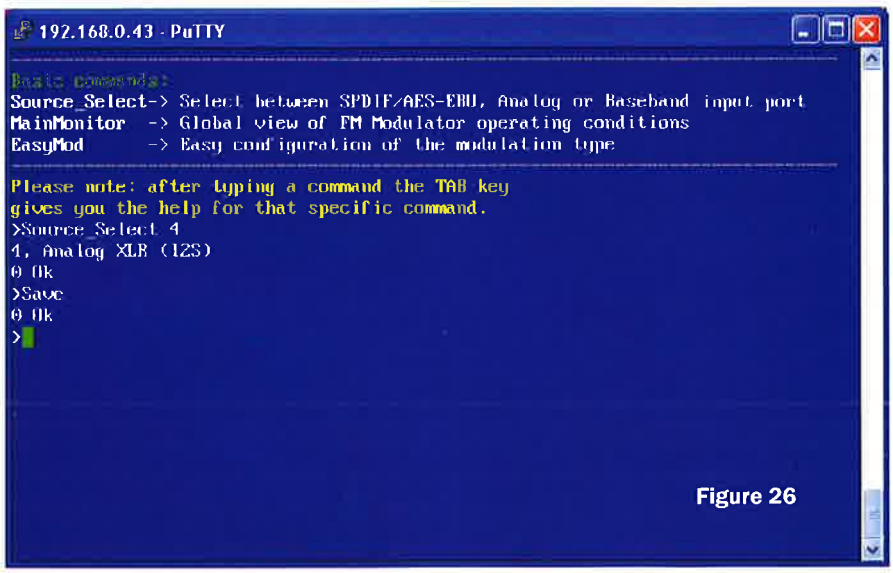
```
Save
```

Maintenant, lancez le programme de gestion de streaming audio en tapant :

```
home/pi/radio/ada_radio.py start
```

L'afficheur LCD indique l'initialisation du système, puis affiche en streaming la radio sur laquelle vous êtes syntonisé. Le flux audio est envoyé à la sortie « I2S » du RaspberryPi et à travers le module FLEXMOD-DAB2, celui-ci retransmet le signal sous forme d'ondes FM avec une fréquence définie par vous-même.





Réglez le récepteur sur la fréquence de votre « station radio ». Vous entendrez la diffusion du flux audio en streaming.

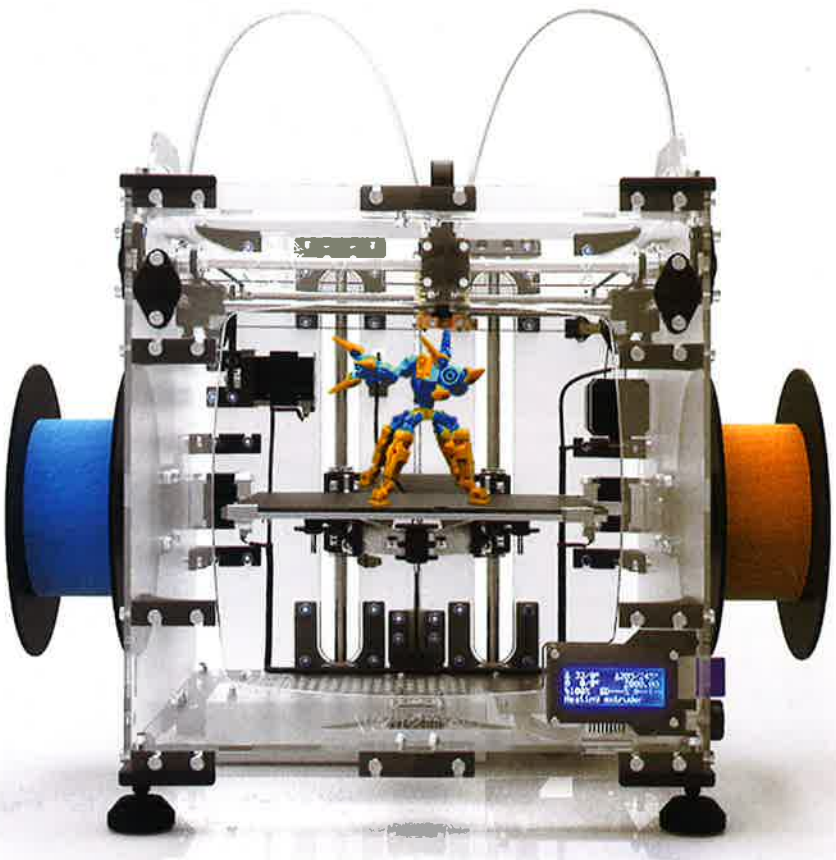
Si le récepteur est RDS, vous devez voir apparaître le nom que vous avez donné à votre station et le texte défilant que vous avez écrit précédemment.

Si cela n'est pas le cas, vous devez régler dans les paramètres de votre récepteur RDS celui de l'affichage des informations de la station (« Show Info Station »).

Dans les prochains articles, nous décrirons de manière plus approfondie l'application de gestion et de monitoring du module FLEXMOD-DB2, et nous intégrerons dans le projet une interface utilisateur graphique particulièrement attrayante et accessible à partir d'un smartphone.

3DVERTEX - 2

Dans le précédent numéro d'Electronique et Loisirs Magazine, nous vous avons présenté une nouvelle imprimante 3D plus compacte et plus polyvalente que la 3DRAG. Nous avons abordé ses caractéristiques techniques, dont la principale est d'imprimer en deux couleurs, ainsi que sa partie électronique. Dans cette seconde partie, nous allons aborder l'aspect mécanique en étudiant le montage pas à pas.



3DVERTEX

L'IMPRIMANTE 3D BICOLORE

Deuxième partie : La mécanique

de Boris LANDONI

Dans le précédent numéro **135** d'Electronique et Loisirs Magazine, nous vous avons présenté la **3DVERTEX**. Il s'agit d'une nouvelle imprimante 3D qui va compléter la famille de la 3DRAG que vous avez apprécié au fil du temps (il y a maintenant déjà trois ans).

L'imprimante 3DVERTEX est plus polyvalente et adopte toujours la même philosophie Open Source de la 3DRAG. Cependant, elle est différente car elle propose des solutions techniques intéressantes, telles que **l'impression native en deux couleurs**.

En fait l'imprimante 3DVERTEX est conçue pour accueillir deux extrudeuses, mais vous pouvez la faire fonctionner avec une seule extrudeuse dans un premier temps et l'upgrader par la suite selon vos moyens. Cette option a été pensée dans le sens où le client peut « mettre à jour » son imprimante 3D tout en maîtrisant son budget.

La **3DVERTEX** est une **imprimante 3D en deux couleurs** pour du filament de 1,75 mm de Ø. La mécanique a été entièrement repensée, le plateau d'impression se déplace vers le haut et le bas (monte et descend), tandis que

Figure 1 : équerres d'assemblage du châssis : A) externe ; B) interne grande ; C) interne petite.

l'extrudeuse se déplace uniquement horizontalement. L'électronique (que nous avons décrit dans le précédent numéro 135) dispose d'une nouvelle carte contrôleur spécifique à cette imprimante.

De plus elle est dotée d'un boîtier qui enveloppe la totalité de l'imprimante avec deux passages latéraux permettant d'installer les bobines de filament à l'extérieur. Les bobines sont montées sur des supports disposant de roulements afin de réduire les efforts lors de l'entrainement des filaments.

Après vous avoir expliqué les innovations technologiques et la description en détail de l'électronique, nous allons étudier maintenant la construction mécanique de la 3DVERTEX.

Commençons par le fait qu'il n'y a plus de cadre en aluminium comme dans la 3DRAG, mais une coque en polycarbonate translucide assemblée par des équerres en ABS renforcé. Cela permet de renforcer la structure du châssis afin de réduire les vibrations, d'éliminer les projections de matière plastique lors de l'impression sur la table de travail, mais aussi d'empêcher qu'un objet étranger pénètre dans l'imprimante car la coque fait office de boîtier.

Cette solution technique simplifie l'assemblage et le montage de l'imprimante. Cela permet d'installer la 3DVERTEX sur votre bureau, tout en maintenant votre espace de travail propre.

La construction mécanique

Nous allons voir maintenant comment assembler et exploiter la 3DVERTEX. Elle est livrée sous forme de kit (voir les annonces dans la revue). Pour des raisons d'espace, nous ne pouvons pas aborder dans le détail le montage mécanique, mais nous vous montrerons les grandes lignes de l'assemblage mécanique.

L'imprimante se compose de **21 blocs** de matériaux, divisés en catégories. Le **châssis** se compose de **4 côtés**, d'une **base** et d'un **cadre supérieur** pour rigidifier l'ensemble.

Les pièces sont assemblées à l'aide de boulons hexagonaux et d'**équerres** en matière plastique renforcée qui sont de deux types : **internes** et **externes**. Les différents types d'équerres sont visibles en figure 1.

La première des choses à faire est d'assembler le châssis, pour cela faites glisser deux grandes équerres sur un panneau latéral comme représenté en figure 2.

Vérifiez que l'orientation des pièces corresponde à celle de la figure 2.

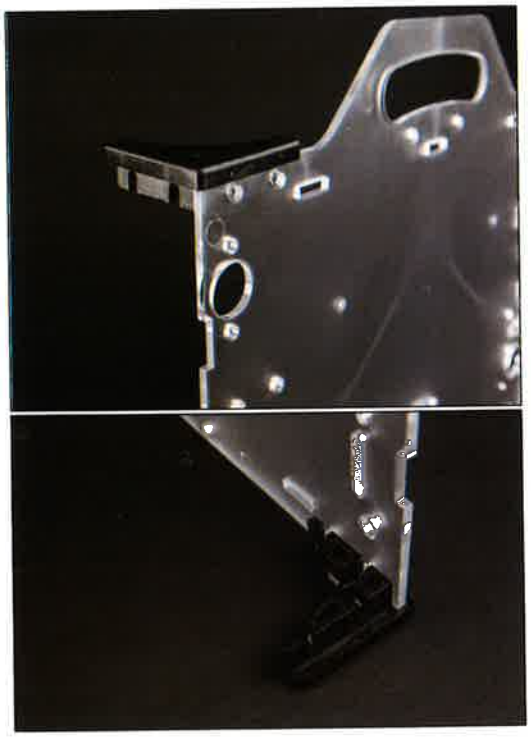
Maintenant, vous pouvez assembler les panneaux latéraux avec la face avant, comme le montre la figure 3, à l'aide des vis et des écrous hexagonaux. Utilisez une clé Allen.

Avant de serrer complétement les vis, il est conseillé de vérifier que les pièces soient positionnées correctement, sinon le châssis peut ne pas s'assembler correctement.

Avant de monter la partie arrière du châssis, vous devez **assembler le fond** sur les panneaux latéraux ainsi que le cadre supérieur (voir la figure 4).

Ces deux dernières pièces s'assemblent grâce à des encoches présentes dans les panneaux sur les côtés.

Figure 2 : ici les équerres assemblées sur les côtés du châssis.



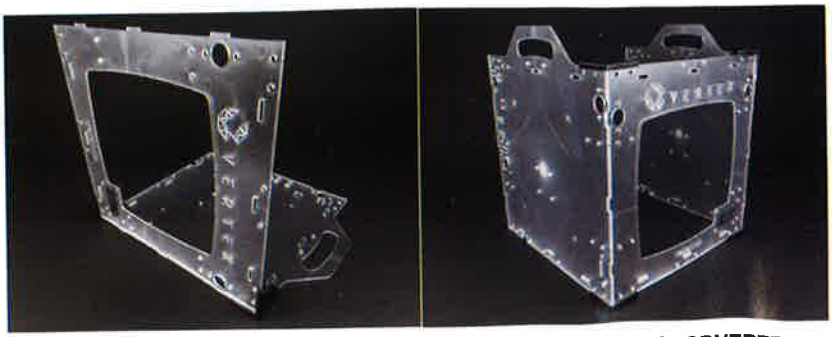


Figure 3 : assemblage des panneaux formant le châssis de la 3DVERTEX.

Notez cependant que le **panneau arrière n'est pas symétrique**, il faut donc l'installer dans la position adéquate.

Une fois le fond et le cadre supérieur encastrés, insérez les vis et les écrous en commençant par la grande équerre supérieure arrière gauche, et le panneau latéral gauche. Serrez les écrous autobloquants de type M5.

Prenez un serre-câble et insérez-le sur la partie émergente du boulon M5, utilisez un écrou M5 pour serrer le tout (voir la figure 5).

Chaque équerre doit être serrée avec deux vis et des écrous autobloquants de chaque côté, les 4 de la partie supérieure et les 4 de la partie inférieure.

Chaque équerre doit également être fixée à l'aide d'une vis dans le sens vertical par rapport au coin (ou angle) du châssis.

Après le montage des équerres externes, vous pouvez passer à l'installation des pieds qui sont vissés sur les grandes équerres de la partie inférieure à l'aide d'écrous (voir la figure 6).

Chaque pied possède un écrou qui permet d'ajuster la hauteur au moyen d'une clé appropriée, de façon à ce que l'imprimante soit de niveau sur le plan de travail.

Le réglage du niveau est fondamental, sinon il peut se produire une déformation de l'imprimante, car les pièces la constituant seraient soumises à des efforts anormaux et provoqueraient des dommages mécaniques, d'où une mauvaise impression des objets.

À ce stade, vous devez installer les grandes équerres internes, elles sont au nombre de deux pour chaque angle. Elles permettent de donner une rigidité à la structure afin de stabiliser la mécanique et en particulier les tiges filetées sur lesquelles se déplacent l'extrudeuse et le plateau d'impression.

Chaque équerre est montée à l'aide de 4 boulons M5 x 12 mm et de 4 écrous autobloquants M5 (voir la figure 7).

Positionnez en premier les boulons sur les panneaux, insérez les équerres dans les boulons puis fixez-les à l'aide des vis.

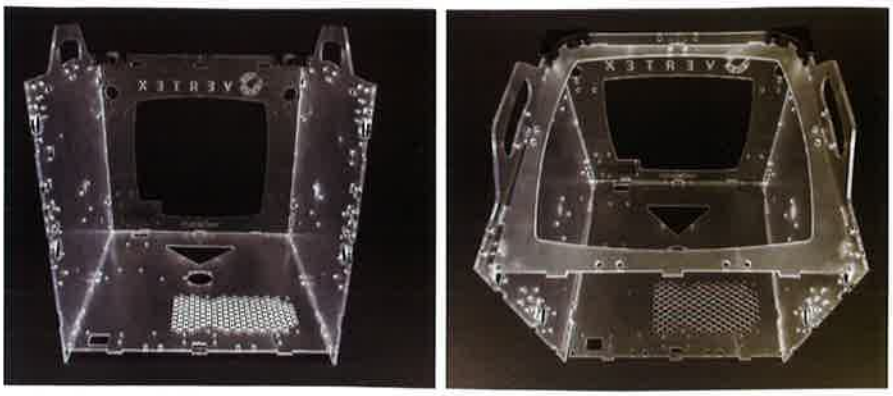


Figure 4 : assemblage du fond et du cadre supérieur du châssis de la 3DVERTEX.

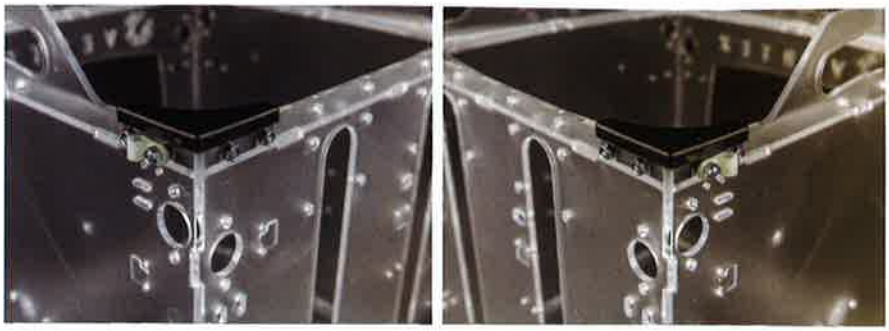


Figure 5 : à l'aide des mêmes vis de fixation que pour les panneaux, fixez les serres-câbles.

Les positions des deux équerres dans chaque angle sont représentées en figure 8.

Ensuite, vous devez installer dans chaque angle de la partie supérieure du châssis des grandes équerres internes en les retournant comme illustré en figure 8.

Une fois les panneaux assemblés et renforcés par les grandes équerres internes, vous devez installer dans la partie supérieure et inférieure 3 petites équerres internes situées au niveau des poignées et vers le milieu de la partie avant (voir la figure 9).

Ces éléments mécaniques sont indispensables pour assurer la stabilité nécessaire à la structure, sinon les zones intermédiaires des panneaux latéraux auraient tendance à s'écarter.

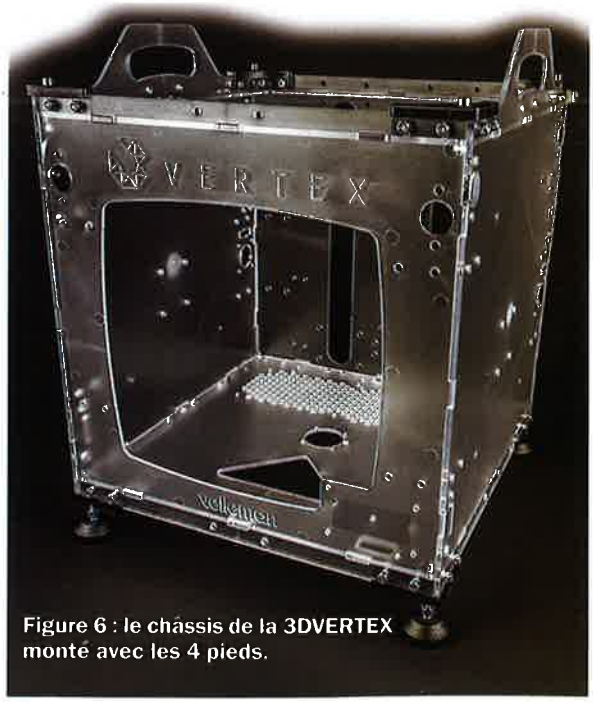


Figure 6 : le châssis de la 3DVERTEX monté avec les 4 pieds.

Après l'installation des petites équerres internes, vous devez faire pivoter la structure et fixez sur la partie arrière les supports en plastique visibles en figure 10.

Ceux-ci supporteront les tiges filetées en acier sur lesquelles se déplacera le mécanisme d'entraînement et de support du plateau d'impression.

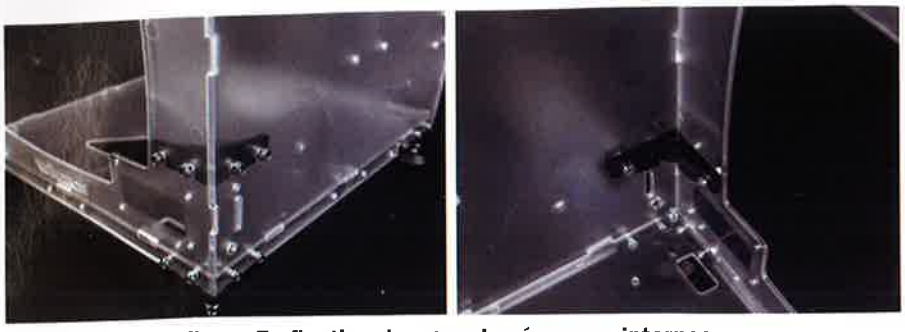


Figure 7 : fixation des grandes équerres internes.

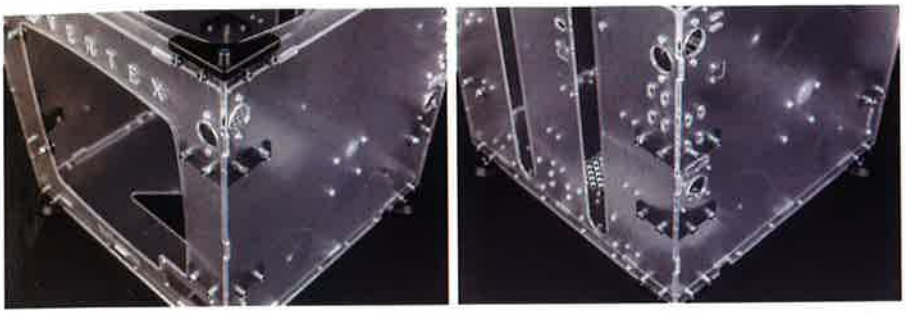


Figure 8 : fixation des grandes équerres internes sur chaque angle de la partie supérieure et inférieure. Elles doivent être retournées.

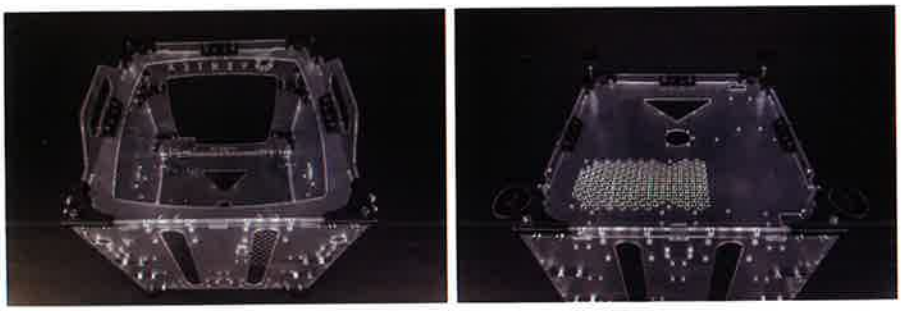


Figure 9 : fixation des petites équerres internes au niveau des poignées et la partie supérieure de la face avant.

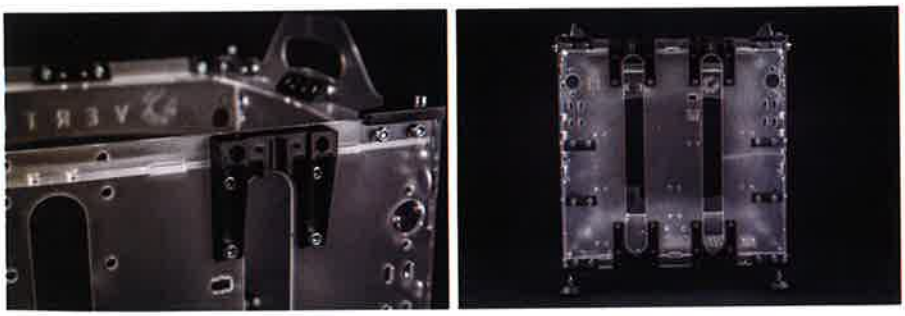


Figure 10 : montage du support supérieur du guide de la tige pour le bras du plateau d'impression.

Les tiges sont au nombre de deux et chacune est fixée au moyen de supports situés dans la partie supérieure et inférieure. Les supports sont alignés selon les fentes (voir la figure 10).

En réalité, ce sont seulement les parties de base des supports (la moitié) qui sont fixées, car lorsque les tiges seront montées, elles seront maintenues par les autres moitiés des supports.

Nous le verrons quand nous serons à l'étape de montage du plateau, car les tiges ne peuvent être fixées qu'après avoir inséré les supports du plateau d'impression.Derrière chaque support des tiges, il est nécessaire d'installer une petite équerre interne, qui va serrer l'ensemble avec les mêmes boulons.

Cette solution permet de **stabiliser davantage le châssis** de l'imprimante et permet de renforcer les propriétés mécaniques de déplacement du plateau d'impression.

En effet les boulons des supports ont une zone de pression plus grande, ce qui évite les risques de cassure de la zone du panneau où se situe le passage du boulon.

Passons maintenant au montage des supports rotatifs des bobines de filament. Vous devez assembler un seul support si vous avez une 3DVERTEX avec une seule extrudeuse, sinon dans le cas contraire il faut monter deux supports rotatifs (cas de 2 extrudeuses).

Si vous avez la version à une seule extrudeuse, vous devez monter le support rotatif du côté droit en regardant l'imprimante de face (à l'avant).

La première chose à faire est de prendre un socle en plastique doté de 4 trous (socle de support bobine). Insérez un boulon M8 x 70 mm en son centre et faites glisser un roulement de Ø 8 mm comme représenté en figure 12. Ensuite serrez un écrou autobloquant jusqu'au roulement.

Vissez un 2ème écrou autobloquant M8 à une distance de 14,25 mm (voir la figure 12), puis insérez un 2ème roulement de

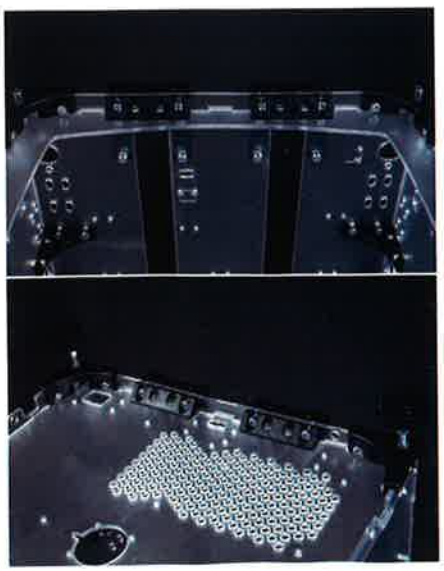


Figure 11 : chaque support des tiges de guidage du plateau d'impression est fixé à l'arrière à l'aide d'une petite équerre intérieur qui vient en butée contre la paroi arrière en polycarbonate.

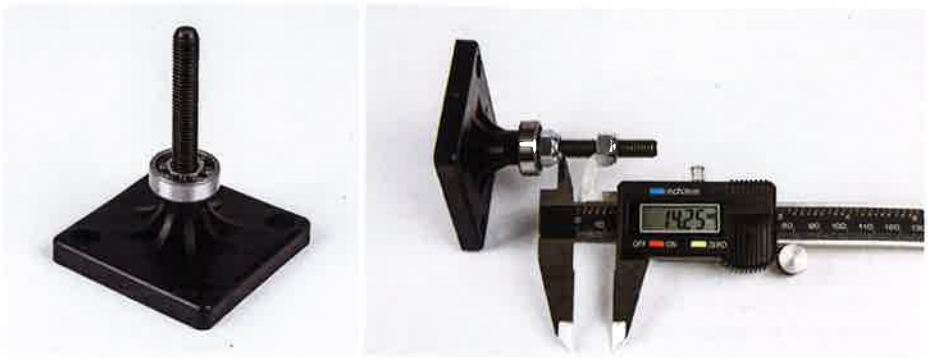


Figure 12 : Pour réaliser le support rotatif de la bobine, insérez sur un socle une tige filetée avec une paire de roulements séparés de 14,25 mm à l'aide d'écrous autobloquants.

Figure 13 : montage du 2ème roulement et de la 1ère moitié du support rotatif de la bobine.

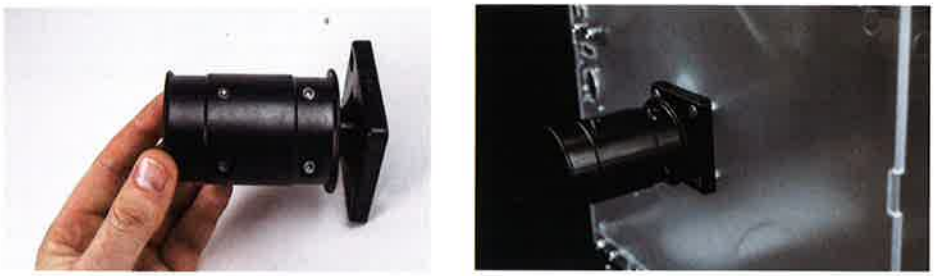


Figure 14 : montage terminé du support rotatif de la bobine.

Figure 15 : montage de la carte autonome de la 3DVERTEX.

Ø 8 mm jusqu'au 2ème écrou que vous venez d'installer. À l'aide d'un 3ème écrou, serrez le second roulement, vous devez obtenir le résultat de la figure 13.

Ensuite placez l'assemblage (socle + écrous + roulements) dans une moitié d'un support rotatif (support de grande bobine) comme indiqué en figure 13.

Si la position des roulements ne coïncide pas parfaitement avec celle des logements prévus dans le support rotatif, corrigez la position du roulement à l'aide d'écrous autobloquants M8.

Une fois le positionnement correct, placez l'autre moitié du support rotatif. Fermez l'ensemble au moyen de quatre boulons M3 x 25 mm munis de leurs écrous M3. Le support rotatif est prêt et peut être installé sur le côté de l'imprimante (voir la figure 14).

Le cas échéant, si vous possédez une deuxième extrudeuse, vous devez assembler le deuxième support en effectuant la même procédure que précédemment.

La prochaine étape consiste à monter la carte autonome de l'imprimante, qui comprend un afficheur LCD, un lecteur de SD-Card et l'encodeur rotatif. Cette carte s'installe à l'intérieur de la paroi frontale (avant), en correspondance avec une fenêtre prévue à cet effet, au moyen de 4 boulons M3 x 30 mm et d'entretoises en plastique M3 de 10 mm (voir la figure 15).

Les vis doivent s'insérer dans les trous correspondants des circuits imprimés (celui de l'afficheur LCD et de la carte de base). Les circuits imprimés doivent être espacés à l'aide d'entretoises en plastique (montage en sandwich).

Faites attention à l'axe de l'encodeur et au **bouton** de réinitialisation (RESET). Ceux-ci doivent **s'insérer dans les trous du panneau latéral.**

Assurez-vous également que toutes les broches de l'afficheur LCD soient alignées avec le connecteur de la carte de base et connectées correctement.

Après avoir fixé l'ensemble à l'aide d'écrous adéquates, insérez le câble plat reliant la carte autonome à la carte contrôleur de l'imprimante. Le **fil rouge doit se situer à gauche**, faites passer le câble dans la fenêtre prévue à cet effet (voir la figure 16).

Pensez à monter le bouton sur l'axe de l'encodeur sur le côté droit de l'afficheur lorsque vous le regardez de face.

Nous allons maintenant installer les capteurs optiques de fin de course (au nombre de 3).

Chaque capteur est constitué par un petit circuit imprimé comprenant un optocoupleur doté d'une cavité dans laquelle coulisse une pièce mécanique.

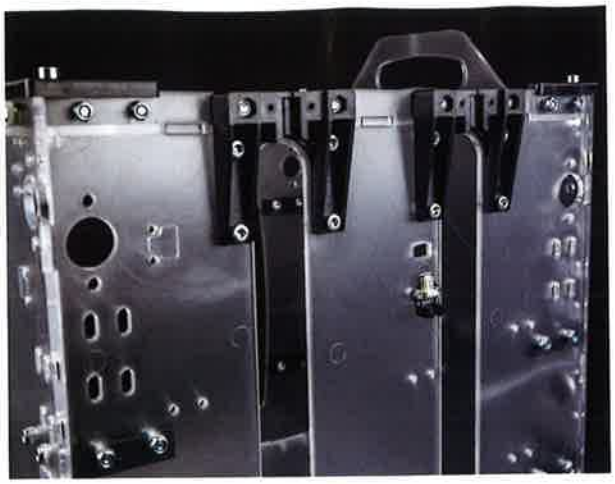
Chaque optocoupleur est fixé aux panneaux relatifs de l'imprimante à l'aide de rondelles isolantes en plastique, afin

Figure 16 : ici le câble plat reliant la carte autonome à la carte contrôleur de l'imprimante. Le fil rouge se situe à gauche.

Figure 18 : montage des optocoupleurs de fin de course des axes X et Y sur les parois internes du châssis.

Figure 19 : montage du ventilateur inférieur sur le fond du châssis. Le flux d'air doit être dirigé vers le bas.

Figure 17 : montage de l'optocoupleur de fin de course de l'axe Z avec les rondelles isolantes.



d'éviter que le circuit imprimé entre en contact avec les vis métalliques.

Les optocoupleurs de fin de course sont au nombre de 3 :

- un fin de course pour l'axe Z qui s'installe sur le panneau arrière à l'extérieur comme représenté en figure 17 ;
- un fin de course pour l'axe X ;
- un fin de course pour l'axe Y.

Pour chaque optocoupleur de fin de course placez 2 rondelles d'isolation M3 dans les trous du circuit imprimé, puis insérez 2 boulons M3 x 10 mm à travers le circuit imprimé et le panneau relatif.

Faites attention à l'orientation de chaque fin de course et à la disposition sur le panneau relatif.

Utilisez 2 écrous autobloquants pour fixer chaque fin de course.

Notez que les fins de course des axes X et Y sont fixés sur les parois internes du châssis et doivent être orientés comme indiqué en figure 18, tandis que l'optocoupleur de fin de course de l'axe Z se trouve à l'extérieur.

Chaque optocoupleur est connecté via un câble spécifique (fourni dans le kit) au connecteur correspondant de la carte contrôleur.

Les emplacements exacts de tous les fins de course sont déterminés par des trous prévus à cet effet sur les panneaux.

Après l'installation des fins de course, vous pouvez passer au montage du ventilateur inférieur (fond du châssis) comme indiqué en figure 19.

Faites passer les fils du ventilateur (40 mm x 40 mm) à travers le trou central du panneau inférieur du châssis, de sorte qu'ils puissent ensuite être reliés à la carte contrôleur qui viendra se fixer sur le fond de l'imprimante.

Notez l'orientation du ventilateur, l'**autocollant doit être vers le bas**. Le **flux d'air doit être dirigé vers le bas**, et les fils du ventilateur du côté de l'encoche.

Utilisez 4 boulons M3 x 20 mm, 4 écrous autobloquants M3 et n'oubliez pas d'installer la grille de protection.

Passons maintenant à l'installation des moteurs pas à pas des axes X et Y avec leurs supports. Insérez 2 poulies **T2,5/19T** de Ø 5 mm sur les arbres des

Figure 20 : préparation des moteurs des axes X et Y avec leurs poulies et leurs supports.

2 moteurs. Insérez une vis de réglage M3 x 4 mm dans chaque poulie.

Figure 20a : orientation des moteurs des axes X et Y par rapport à leurs supports.

Figure 20b : l'orientation du moteur de l'axes Z par rapport au support est différente.

Figure 21 : positionnement des moteurs des axes X et Y dans le châssis.

Figure 22 : montage du moteur de l'axe Z.

Faites attention que le **côté plat de l'axe s'aligne avec l'emplacement de la vis de réglage** et que celle-ci entre en contact avec l'axe lors du serrage.

Faites glisser la poulie au ras du moteur sans le toucher. Serrez la vis de réglage, répétez l'opération pour l'autre moteur (voir la figure 20).

Vérifiez que les poulies et les axes peuvent tourner librement.

Munissez-vous de 4 boulons M3 x 8 mm pour monter les supports moteurs. **Positionnez les connecteurs des moteurs des axes X et Y par rapport aux supports comme** illustré en figure 20a.

Le premier moteur doit être fixé à l'intérieur du châssis sur le côté gauche, l'autre moteur sur le côté droit. Ne serrez pas complètement les écrous, cela s'effectuera par la suite.

Les positions exactes des moteurs pas à pas des axes X et Y sont celles illustrées en figure 21.

Le moteur de l'axe Z doit être équipé d'un support, mais il ne dispose pas de poulie. **L'orientation du connecteur du moteur de l'axe Z par rapport au support est différente de celle des moteurs des axes X et Y** (voir la figure 20b).

Le moteur de l'axe Z doit être fixé dans la partie inférieure du panneau arrière. Utilisez 4 boulons M5 x 12 mm et 4 écrous autobloquant M5, ne serrez pas complètement.

Nous abordons maintenant l'assemblage de l'extrudeuse, qui prend place sur le côté latéral du châssis à proximité des bords arrières.

Si vous disposez d'une 3DVERTEX en version double extrudeuse, répéter cette opération pour la seconde extrudeuse.

Figure 23a : assemblage du levier de desserrage.

Figure 23 : montage du moteur d'alimentation en filament de l'extrudeuse

La première étape consiste à **chercher les pièces correspondantes à la figure 23a.** Elles doivent être assemblées selon l'ordre dans lequel elles se trouvent sur l'image (figure 23a).

Vous devez obtenir le résultat de la figure 23b, il s'agit du levier de desserrage. Serrez fermement l'écrou de blocage M5.

Ensuite prenez un moteur, une poulie et une vis de réglage. Insérez la vis de réglage dans la poulie à l'aide d'une clé Allen (voir la figure 23).

Positionnez la poulie (de filament) jusqu'à ce que la distance entre le dessus de la poulie et le corps du moteur soit de 19,5 mm (voir la figure 23c).

Une fois le réglage effectué, serrez fortement la vis de réglage.

Figure 23b : ici le levier de desserrage une fois monté.

Figure 23c : la distance entre le dessus de la poulie et le corps du moteur doit être de 19,5 mm.

Insérez ensuite le moteur dans l'emplacement prévu dans le panneau latéral du châssis (si vous montez la version comportant une seule extrudeuse, installez seulement le moteur sur le côté où vous avez placé le support de bobine).

Faites glisser le socle de droite (c'est-à-dire le socle correspond au côté droit) de l'extrudeuse sur la poulie de filament.

Vissez un boulon M3 x 16 mm dans le trou inférieur droit du socle de droite. Serrez le boulon de sorte que le moteur et le socle soient fermement fixés ensemble (voir la figure 23).

Attention toutefois à ne pas serrer de manière excessive, cela pourrait endommager le socle en plastique. **Le connecteur du moteur doit se situer vers le bas de l'imprimante**.

Le socle en plastique correspond en fait à la moitié de la coque qui entoure la poulie comprenant la roue dentée, elle permet l'entraînement du filament.

Insérez un écrou M5 dans l'ouverture située sur le haut du socle. Insérez le couvercle droit (l'autre moitié du socle) sur la partie déjà installée et à l'aide de 2 boulons M3 x 30 mm fixez le couvercle.

Prenez le raccord pour tube, celui-ci est constitué d'un corps hexagonal recouvert d'un revêtement en plastique rouge (voir la figure 24) et vissez-le dans l'écrou M5 que vous avez précédemment inséré dans le haut du socle (qui est maintenant fermé).

Ne serrez pas fortement le raccord, cela pourrait l'endommager.

Pour compléter le mécanisme d'alimentation en filament de l'extrudeuse, vous devez installer les éléments visibles en figure 24, c'est-à-dire le ressort qui comprime le filament sur la roue dentée de la poulie.

Placez le ressort de 27,5 mm dans la fente à droite du bloc de l'extrudeuse (voir la figure 25).

Insérez le levier de desserrage assemblé au début de cette étape. Utilisez un boulon M3 x 30 mm pour maintenir le levier de desserrage.

Vous devez le maintenir fermement vers le haut afin d'aligner les trous et pouvoir mettre en place le boulon.

Ne serrez pas trop fort le boulon car le levier de desserrage doit pouvoir se déplacer de haut en bas sans efforts. Vous devez obtenir le résultat de la figure 25 (à droite de l'image).

Figure 24 : fixation du raccord pour tube.

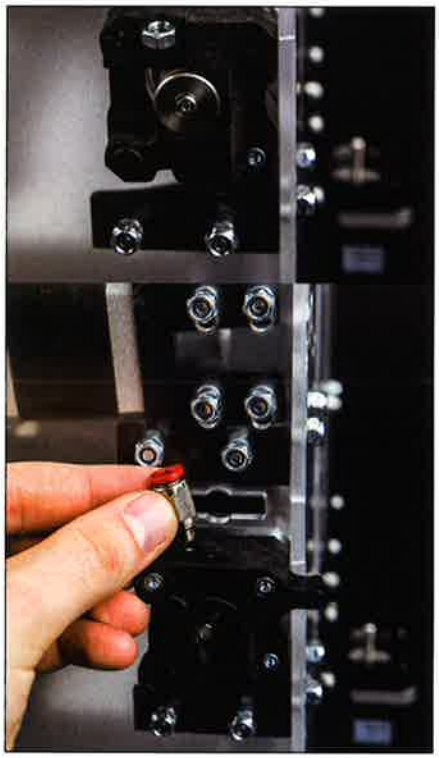
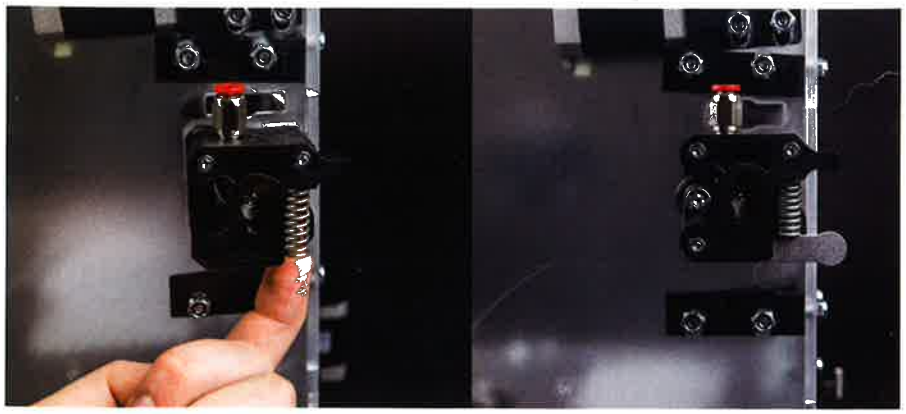


Figure 25 : montage de l'ensemble ressort + levier de desserrage. À droite vous pouvez voir le levier en place, il doit se manœuvrer sans effort.



Préparation de la carte contrôleur

Cette étape demande une attention particulière, vous devez manipuler délicatement les différentes cartes électroniques.Ces différentes cartes sont au nombre de 5, à savoir la carte principale qui est la carte contrôleur et 4 petites cartes qui correspondent aux drivers des moteurs.

Pour le **câblage général**, reportez-vous au précédent numéro **135** d'**Electronique et Loisirs Magazine**.

La carte contrôleur s'installe **dans le fond** de l'imprimante mais du **côté extérieur**. Avant de fixer la carte contrôleur sur le châssis de l'imprimante, commencez par **installer les 4 cartes drivers pour les moteurs sur la carte contrôleur** aux emplacements prévus. Faites attention à leur orientation.

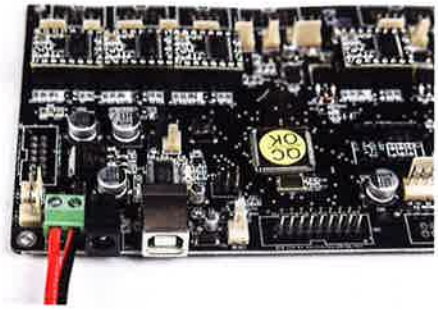


Figure 26a : câblage des fils de l'alimentation.

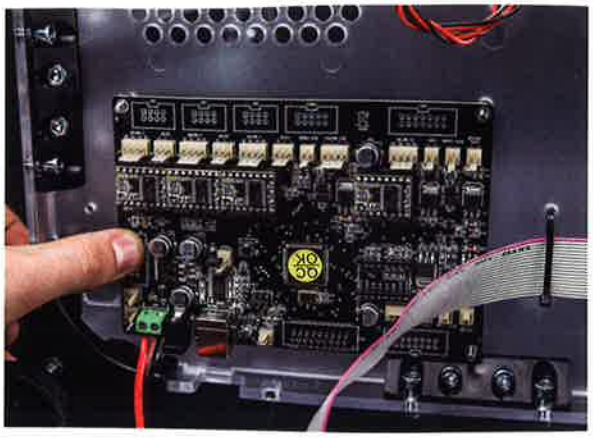


Figure 26b : fixation de la carte contrôleur au fond du châssis à l'extérieur.

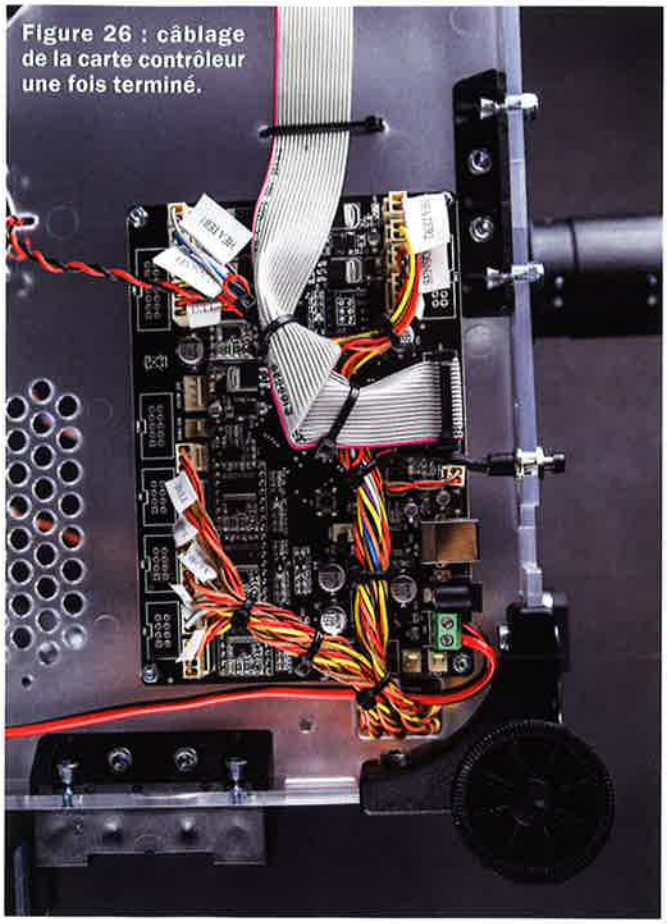


Figure 26 : câblage de la carte contrôleur une fois terminé.

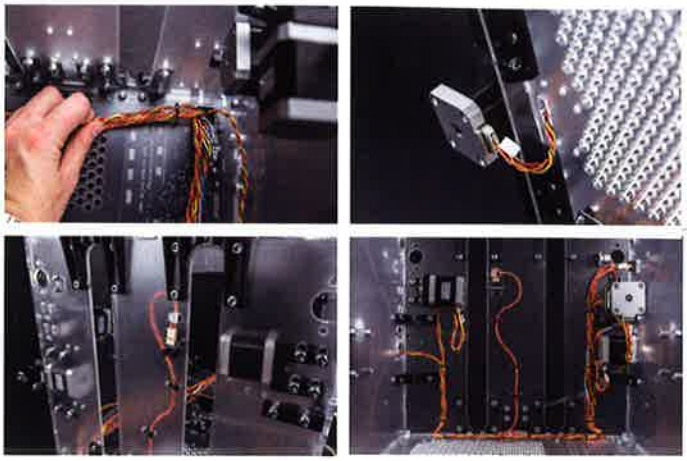


Figure 27 : passages des câbles à l'intérieur et à l'extérieur du châssis.

Dénudez les 2 fils (rouge et noir) du câble d'alimentation, et reliez les 2 fils sur le bornier d'alimentation (à côté du connecteur USB).

Faites attention à la polarité, le **fil rouge** se situe **à gauche** si vous positionnez la carte contrôleur comme indiqué en figure 26a.

Maintenant, montez la carte contrôleur sur le fond de l'imprimante à l'extérieur, pour cela placez 4 boulons M3 x 16 mm dans le panneau inférieur, puis insérez des entretoises en plastique sur chaque boulon. Disposez la carte contrôleur sur les boulons selon la disposition de la figure 26b. Fixez alors la carte à l'aide de 4 écrous autobloquants M3.

Commencez par brancher le câble plat, le **fil rouge doit se situer du côté de la prise USB**. Ensuite continuez le câblage en vous aidant des figures 26 et 27.

Vérifiez plusieurs fois l'exactitude des connexions. Utilisez ensuite des colliers pour maintenir les faisceaux de câbles après avoir **vérifié que les câbles n'entrent pas en contact avec les mécanismes du plateau d'impression et de l'extrudeuse**.

Ceci est fondamental pour les câbles situés à l'intérieur de châssis.

Passons maintenant au montage de l'alimentation AC/DC, elle doit être installée sur le fond du châssis à l'extérieur.

Avant de fixer l'alimentation, il faut **sélectionner la tension d'entrée** adéquate, elle est conçue pour fonctionner sur les plages 100 VAC à 120 VAC et 200 VAC à 240 VAC.

Sélectionnez la gamme de tension de **200 VAC à 240 VAC** à l'aide du **commutateur situé à l'intérieur du boîtier et indiqué par une étiquette jaune**, en utilisant un tournevis. Attention **cette opération doit s'effectuer avec l'alimentation débranchée du secteur**.

Figure 28 : positionnement et câblage de l'alimentation AC/DC.

Insérez 4 boulons M3 x 12 mm à travers le panneau inférieur, puis insérez 4 entretoises en plastique M3 de 5 mm par-dessus les boulons. Positionnez l'alimentation et serrez tous les boulons.

Reliez les 3 fils du câble secteur aux borniers de l'alimentation comme indiqué en figure 28. Le fil marron doit être connecté à la borne « L », le fil bleu à la borne « N » et le fils jaune/vert à la borne de terre.

Ensuite insérez le câble d'alimentation (rouge/noir) dans les bornes « +V » et « -V ». Le **fil rouge** étant bien évidement relié à la borne « **+V** ». Vous devez obtenir un résultat semblable à celui de la figure 28.

Nous abordons à ce stade du montage de la 3DVERTEX la partie la plus complexe à réaliser, il s'agit du montage de l'extrudeuse. Vous devez procéder par étape dont la première consiste à chercher et ranger sur votre plan de travail les pièces suivantes (voir la figure 29a) :

- 1 ventilateur 50 mm x 50 mm ;
- 1 ventilateur 25 mm x 25 mm ;
- 2 boulons M3 x 12 mm ;
- 4 boulons M3 x 10 mm ;
- 1 boulon M3 x 6 mm ;
- 1 boulon M3 x 8 mm ;
- 1 plaque de support de l'extrudeuse ;
- 1 raccord pour tube ;
- 1 support pour le petit ventilateur ;
- 9 rondelles isolantes M3 ;
- 1 vis de réglage M3 x 4 mm ;
- 2 entretoises en métal M3 x 25 mm ;
- 1 support pour tube supérieur Ø 8 mm ;
- 1 support pour tube inférieur Ø 8 mm ;
- 1 circuit imprimé de l'extrudeuse ;
- 1 buse 0,35 mm ;
- 1 bloc chauffant ;
- 1 guide d'isolation ;
- 1 support « hot end » ;
- 5 écrous autobloquants M3 ;
- 4 vis à tête fraisée M3 x 50 mm ;
- 2 chariots « XY 2 » ;
- 1 support pour le grand ventilateur ;
- 1 chariot « XY 1 » ;
- 1 résistance chauffante ;
- 1 capteur de température ;
- 1 morceau de tube PTFE de longueur 23 mm.

Commencez avec le morceau de tube PTFE, vous devez le couper avec un instrument propre, la coupe doit être perpendiculaire.

La **longueur du tube doit être de 23 mm ± 0,25 mm**, c'est-à-dire **au minimum 22,75 mm** ou **au maximum 23,25 mm**.

Cela demande donc une grande précision, soyez vigilant. **Evitez d'introduire des saletés dans le tube**.

Ensuite insérez le morceau de tube PTFE dans le guide d'isolation sans l'enfoncer complètement.

Figure 29a : les différentes pièces mécaniques constituant la partie de l'extrudeuse.

Placez le support « hot end », il ressemble au boîtier d'un transistor TO3 (creux au centre).

Vissez la buse de 0,35 mm dans le guide d'isolation, ce qui a pour effet d'enfoncer complètement le tube.

Vous devez obtenir un ensemble correspondant à l'image de gauche de la figure 29. L'ensemble doit mesurer normalement **47 mm**.

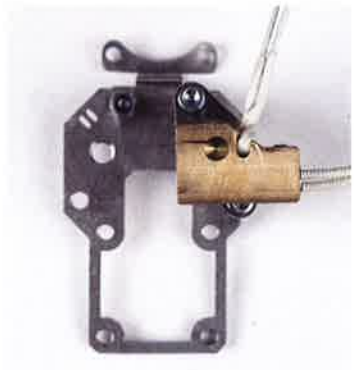
Figure 29b : la plaque de support de l'extrudeuse avec une rondelle isolante.

Figure 29c : la plaque de support de l'extrudeuse avec les 2 rondelles isolantes de l'autre côté.

Figure 29d : disposition du support du petit ventilateur par rapport à la plaque de support de l'extrudeuse.

Figure 29e : fixation du support du petit ventilateur sur la plaque de support de l'extrudeuse.

Figure 29f : le bloc chauffant est orienté horizontalement.



Insérez 2 rondelles d'isolation M3 de chaque côté (donc 4 rondelles) du support « hot end ». Insérez 2 boulons M3 x 10 mm dans le support, et vissez-les dans 2 entretoises M3 x 25 mm (disposées derrière le support).

Prenez le bloc chauffant et insérez la résistance chauffante, ensuite insérez une vis de réglage M3 x 4 mm pour maintenir la résistance chauffante.

Faites glisser le capteur de température sur le côté latéral du bloc jusqu'à ce que la partie de détection du capteur se trouve en contact avec le centre du bloc, pliez les fils du capteur de température.

Vous devez obtenir le résultat correspondant à l'image du milieu de la figure 29.

Insérez, en le tournant délicatement, le bloc chauffant sur la buse. Ne desserrez pas la buse lors de la manœuvre (voir l'image de droite de la figure 29).

Prenez la plaque de support de l'extrudeuse, positionnez-la de la manière représentée en figure 29b.

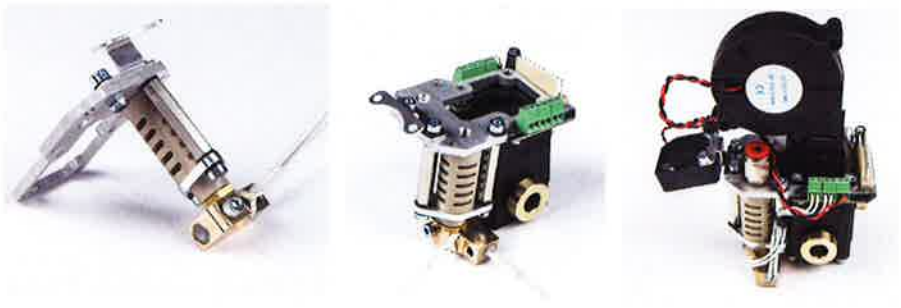
Insérez une rondelle isolante M3 comme indiqué en figure 29b (vers le bas et le devant de l'image).

Retournez la plaque et positionnez-la comme indiqué en figure 29c. Placez 2 rondelles isolantes M3 selon la figure 29c (points noirs sur l'image).

Munissez-vous du support du petit ventilateur, insérez 2 rondelles isolantes M3 et disposez-le par rapport à la plaque comme indiqué en figure 29d.

Figure 29 : assemblage du guide d'isolation avec la buse et du bloc chauffant (dans l'ordre de gauche à droite).

Figure 30 : assemblage du guide d'isolation avec le bloc chauffant et des ventilateurs à la plaque de support de l'extrudeuse.



Insérez un boulon M3 x 8 mm et un boulon M3 x 10 mm à travers le support du petit ventilateur et la plaque de support de l'extrudeuse.

À l'aide d'un écrou autobloquant M3, serrez le boulon M3 x 8 mm comme visible en figure 29e.

Ensuite prenez l'autre boulon M3 x 10 mm et insérez-le sur la plaque de support de l'extrudeuse, fixez l'ensemble constitué par la buse et le guide d'isolation, assurez-vous que le bloc chauffant est orienté horizontalement comme indiqué en figure 29f.

À l'aide d'un boulon M3 x 6 mm, fixez le bloc chauffant tout en maintenant la position (voir l'image de gauche de la figure 30).

Maintenant fixez le support du tube supérieur Ø 8 mm dans un charriot « XY 2 », placez le charriot « XY 1 » par-dessus. Puis fixez le support du tube inférieur Ø 8 mm et insérez dessus le deuxième charriot « XY 2 ».

Vous obtenez un bloc. Retournez-le, vous devez voir 4 petits trous où vient s'installer le circuit imprimé. Fixez ce dernier, puis l'ensemble « buse + guide d'isolation + plaque de support » comme indiqué en figure 30 (image du milieu).

Prenez le grand ventilateur 50 mm x 50 mm et appliquez de la colle sur chaque côté de la tubulure, puis fixez le ventilateur sur le grand support.

Une fois la colle sèche, assemblez le ventilateur sur le bloc « buse + guide d'isolation + plaque de support ».

Vous obtenez le résultat visible sur l'image de droite de la figure 30. À l'aide de 4 vis à tête fraisée M3 x 50 mm et 4 écrous fixez l'ensemble.

Ensuite fixez le petit ventilateur sur son support (celui de la figure 29d) et le raccord pour tube (rouge).

Enfin, connectez les fils aux différents borniers du circuit imprimé de l'extrudeuse, utilisez des colliers pour agencer les fils.

Vous devez obtenir un résultat équivalent à celui de l'image de droite de la figure 30.

Nous venons d'assembler la tête d'impression (extrudeuse) qui est une étape longue et minutieuse.

Nous aborderons dans le prochain numéro le montage de la tête d'impression, de l'axe des Z, du plateau d'impression et enfin de la seconde extrudeuse qui est une option. ■

Cours ARDUINO - 6

Cours ARDUINO

sixième partie

de Mirco SEGATELLO

Dans cette sixième partie du Cours Arduino, nous allons apprendre à mettre en œuvre les modules « XBee » afin de réaliser des connexions sans fil (Wireless) pour nos applications avec Arduino.

Dans cette partie du Cours Arduino, nous allons équiper une carte Arduino d'un **module de communication radio** afin de communiquer avec une autre carte Arduino ou un PC. Parmi les possibilités infinies offertes par les produits disponibles sur le marché, Arduino a adopté la norme « **ZigBee** », les applications qui en découlent sont donc orientées vers un matériel spécifique, mais disponible dans le commerce en plusieurs versions.

La norme « ZigBee »

Le terme « **ZigBee** » vient des mots anglais « zigging bee » qui signifient « danse avec l'abeille ». Il a été adopté car il a été inspiré par un mode de communication essentiel et rapide que les insectes utilisent pour la survie de leurs colonies.

Le **protocole** « **ZigBee** » a été mis en œuvre par la « **ZigBee Alliance** » (www.zigbee.org), qui est un consortium à but non lucratif pour la fabrication de semi-conducteurs. Son objectif était de créer un protocole de communication très polyvalent et largement diffusé afin d'être en mesure de fonctionner sur des réseaux ayant des structures différentes. Ce protocole est caractérisé par une faible consommation et est adapté à des systèmes fonctionnant sur batterie pendant de longues périodes. Par exemple, dans les systèmes de télémétrie, d'alarmes, domotiques, les détecteurs de fumée, etc.

Le **protocole** « **ZigBee** » a été conçu pour être en mesure de mettre en œuvre différents types de réseaux, statiques, dynamiques, en étoile, maillés. Ce protocole permet de gérer jusqu'à 65000 nœuds et de veiller à l'absence de collisions, il dispose d'un contrôle très avancé des erreurs.

Tableau 1 : comparaison des protocoles de communication

Protocole	ZigBee 802.15.4	Wi-Fi 802.11b	Bluetooth 802.15.1
Portée en mètres	1 à 100	1 à 100	1 à 10
Durée de la batterie	100 à 1000	0,5 à 5	1 à 7
Nombre de noeuds dans le réseau	65000	32	7
Applications	Monitoring, contrôle, télémétrie	Web, Email, Video	En remplacement d'un câblage
Taille de la pile (Kb)	4 à 32	1000	250
Vitesse de transmission (kb/s)	20 à 250	11000	720

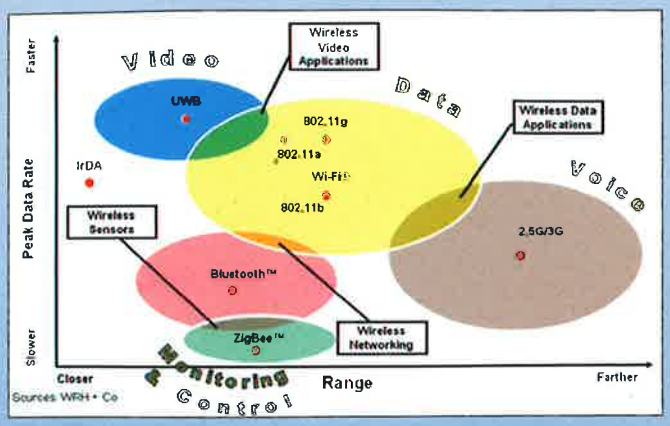


Figure 1 : comparaison entre les différents protocoles de communication.

Le **protocole** « **ZigBee** » présente des caractéristiques de **sécurité pour l'accès au réseau**, y compris un **système de cryptage à 128 bit**s basé sur le standard « NIST » Certified Advanced Encryption Standard (AES).

Si nous comparons le protocole « ZigBee » à celui du Wi-Fi et du Bluetooth, nous allons remarquer immédiatement quelques différences :

- le **Bluetooth** est essentiellement utilisé pour relier entre eux deux appareils sur une courte distance, sans utiliser des câbles. Il est généralement implémenté sur les smartphones afin d'utiliser des périphériques externes tels que des casques, des hauts parleurs, des récepteurs GPS, ou pour échanger des données avec un PC. Il est capable de gérer plusieurs appareils en même temps mais son débit est plutôt limité ;
- le **Wi-Fi** présente l'avantage d'un débit plus élevé, de sorte qu'avec des périphériques et des antennes spécifiques, il est possible de couvrir plusieurs centaines de mètres. Son taux de transfert est important, et il a été encore amélioré grâce aux dernières normes, ce qui permet de faire passer de la vidéo entre les périphériques et un PC par exemple. Cependant, le coût de la partie matérielle est élevé, de plus la consommation de courant est importante, ce qui n'en fait pas un protocole idéal pour des systèmes fonctionnant sur batterie pendant de longues périodes (voir le Tableau 1).

Au niveau commercial, la norme « ZigBee/IEEE 802.15.4 » a été mise en œuvre par la firme **DIGI** (www.digi.com) avec des modules appelés « **XBee** ». Les premiers de ces modules mis sur le marché se dénommaient tout simplement « XBee ». Ils étaient donc dotés du protocole « ZigBee », leurs caractéristiques en faisaient des modules idéaux pour des applications où il était nécessaire de réaliser des réseaux à faible coût et à faible consommation.

Ces modules sont simples à utiliser, ils ne nécessitent que très peu d'énergie et sont une solution efficace et fiable pour la transmission de données critiques. Les modules « **XBee** » fonctionnent dans la bande « ISM » à une **fréquence de 2,4 GHz**, ils permettent de réaliser des circuits extrêmement compacts.

Ils sont aussi compatibles avec les règlementations en vigueur dans des pays tels que les USA, le Canada, l'Australie, et l'Union Européenne.

Tableau 2 - Caractéristiques techniques des modules XBee Série 1.

Paramètres	XBee	XBee-PRO
Fréquence de fonctionnement	ISM 2,4GHz	ISM 2,4GHz
Puissance d'émission	1 mW	63 mW
Sensibilité de réception	-92 dB	-100 dB
Portée	30 m (à l'interieur) 100 m (en extérieur)	90 m (à l'interieur) 1.600 m (en extérieur)
Consommation	TX 45 mA RX 50 mA Veille < 10µA	TX 250 mA RX 55 mA Veille < 10µA
Vitesse de transmission HF	250 kbps	250 kbps
Vitesse de transmission de l'UART	1,2 à 115 kBaud	1,2 à 115 kBaud
Extensions	Lignes A/D et digitales	Lignes A/D et digitales
Périphériques adressables	65 000	65 000
Nombres de canaux disponibles	16	12
Alimentation	2,8 à 3,4 V	2,8 à 3,4 V

Chaque module contient un **émetteur/récepteur radio** et un **microcontrôleur**. Le programme (firmware) de ce dernier est évolutif (possibilité de mise à jour), et gère, en plus de la communication radio, les lignes d'entrées/sorties numériques et analogiques, les fonctions d'interruption (interrupt) et la mise en veille (sleep).

Cela permet **de connecter directement au module des capteurs** ou des **contacts secs** et ainsi d'implémenter un **système autonome** sans avoir besoin de l'interfacer avec d'autres éléments extérieurs mise à part une batterie pour l'alimentation. Toutes les tâches à effectuer sont programmées par l'utilisateur en fonction de ses besoins.

Les modules « **XBee** » existent aussi en **version professionnelle** en conservant les mêmes caractéristiques

et le même brochage mais avec une puissance d'émission supérieure. Cela permet des communications sur une longue distance. Vous pouvez voir un résumé des caractéristiques techniques dans le Tableau 2.

Il existe **plusieurs versions** de modules « **XBee** » en fonction du **type d'antenne** utilisé, les modules les plus performants permettent l'utilisation d'une antenne extérieure de type « **UFL** », cependant le coût engendré ainsi que les dimensions globales plus importantes du système ne justifient pas toujours leur utilisation.

Une seconde version de modules « XBee » fournit une antenne extérieure déjà installée et permet d'approcher le débit maximal. Une version avec une antenne intégrée existe, très pratique, mais dont la portée est réduite d'environ 30 %.

La série « **PRO** », compatible avec la série standard, a une **puissance d'émission** et une **sensibilité de réception plus importantes**. Ces caractéristiques lui permettent d'être utilisée dans des communications sur de longues distances. La série « PRO » se distingue par un encombrement supérieur par rapport à la série standard.

La « **série 2** » de modules « **XBee** » met en œuvre toutes les possibilités du protocole « ZigBee ». Cela permet de créer des **réseaux complexes** avec les fonctions d'auto-configuration, d'auto-routage, d'auto-dépannage, mais aussi des **réseaux maillés**. L'aspect extérieur reste inchangé, mais la mention « série 2 » est sérigraphiée. Par contre les modules de la « **série 2** » **ne sont pas compatibles** avec les modules de la « **série 1** ».

La « série 2 » introduit la possibilité de **configurer un module en tant que routeur** qui, inséré dans un réseau, se comporte comme un pont entre le **coordinateur** et le **terminal**. Ainsi avec d'autres routeurs cela vous permet d'étendre dynamiquement le réseau.

Il existe aussi une version « PRO » de la « série 2 » avec une puissance d'émission beaucoup plus importante. Reportez-vous au Tableau 3 dont les principales caractéristiques y sont reportées.

Étudions, ci-après, les éléments de base qui constituent un réseau de type « ZigBee » :

- **Nœud** : ce terme désigne tout dispositif du réseau qui échange des données par radio et qui est identifié par un numéro (Identify Device) ;

Figure 2 : module « XBee » avec une antenne intégrée.

Figure 3 : module « XBee » avec une sortie pour antenne extérieure de type UFL.

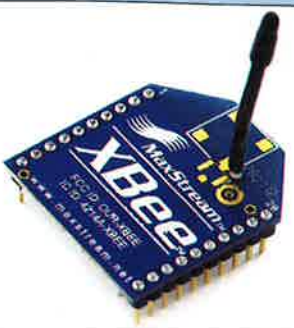


Figure 4 : module « XBee » avec une antenne extérieure intégrée.

Figure 5 : module « XBee PRO » avec une antenne intégrée.

Figure 6 : module « XBee PRO » avec une sortie pour antenne extérieure de type UFL.

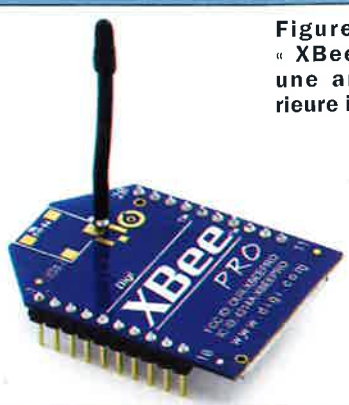


Figure 7 : module « XBee PRO » avec une antenne extérieure intégrée.

- **Coordinateur - ZC (controller)** : il représente le nœud racine du réseau, il initialise le réseau, gère les différents nœuds, exploite les données collectées et est capable de fonctionner comme un pont pour d'autres réseaux. Pour chaque réseau un coordinateur, qui contient des informations importantes de sécurité du réseau, est désigné. Il est capable de configurer les autres modules du réseau ;
- **Routeur – ZR** : aussi dénommé « Full Function Device » (FFD), c'est un dispositif client qui génère des informations et les adresse au nœud central. Il peut également servir d'intermédiaire pour l'envoi vers le nœud central d'informations provenant d'autres dispositifs ;
- **End Device – ZED** : également appelé « Reduced Function Device », RFD ou simplement nœud. C'est un dispositif client qui collecte des informations et les adresse au coordinateur ou à un routeur, cependant il ne joue pas le rôle d'intermédiaire pour d'autres dispositifs. C'est l'élément le plus simple, en général il est destiné à des objets tels que les interrupteurs, la télévision, la radio, les lampes, les appareils électroménagers, etc. Normalement, il fait partie d'un sous-réseau en étoile de type « point à multipoint » vers un routeur ou vers le coordinateur.

Tableau 3 - Caractéristiques techniques des modules XBee Série 2.

Paramètres	XBee Serie2	XBee-PRO Serie2
Fréquence de fonctionnement	ISM 2,4 GHz	ISM 2,4 GHz
Puissance d'émission	2 mW	63 mW
Sensibilité de réception	-92 dB	-100 dB
Portée	40 m (à l'interieur) 120 m (en extérieur)	90 m (à l'interieur) 1.600 m (en extérieur)
Consommation	TX 40 mA RX 40 mA Veille < 1µA	TX 250 mA RX 55 mA Veille < 4µA
Vitesse de transmission HF	250 kbps	250 kbps
Vitesse de transmission de l'UART	1,2 k à 1 MBbaud	1,2 k à 1 MBbaud
Extensions	Lignes A/D et digitales	Lignes A/D et digitales
Périphériques adressables	65000	65000
Nombres de canaux disponibles	16	15
Alimentation	2,1 à 3,6 V	2,7 à 3,6 V

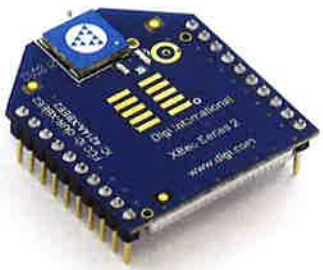


Figure 8 : module « XBee série 2 » avec une antenne intégrée.

Figure 9 : module « XBee série 2 » avec une sortie pour antenne extérieure.

Figure 10 : module « XBee série 2 » avec une sortie pour antenne extérieure de type UFL.

Figure 11 : module « XBee PRO » avec une antenne extérieure intégrée.

Réseaux implémentables avec les modules « XBee »

Nous allons voir maintenant quels sont les types de réseaux sans fil qui peuvent être mis en œuvre avec les modules « XBee » :

- **Réseau point à point (PPP)** : dans ce type de réseau, deux dispositifs dialoguent entre eux. Pour améliorer l'immunité aux bruits et éviter les conflits qui pourraient affecter les communications, il est possible de paramétrer un canal de transmission différent d'un module à l'autre. Ce type de réseau, même s'il est seulement constitué par deux éléments, peut contenir un maître et un esclave (voir la figure 12) ;
- **Réseau point à multipoint (PMP)** : dans ce type de réseau un dispositif communique avec plusieurs autres dispositifs. La topologie en étoile de ce type de réseau est d'une plus grande complexité et impose de résoudre des problèmes de conflits. C'est pour cela qu'à un seul dispositif est assigné la fonction de maître tandis que les autres sont tous des esclaves. Dans le cas du protocole « ZigBee », il y a au moins un coordinateur et un « End Device » (voir la figure 13) ;
- **Réseau pair à pair (peer-to-peer ou P2P)** : il s'agit d'un réseau utilisant le principe de la parité dans le sens où les données peuvent être transférées directement entre deux ordinateurs connectés au réseau. Ce modèle de

réseau est proche du modèle « client/serveur », mais chaque client est aussi un serveur. Les dispositifs qui y sont connectés sont équivalents (peer) dans le sens où ils remplissent les fonctions de serveurs et de clients. De cette façon, chaque nœud est capable d'initier et de terminer une communication. Si nous prenons le cas de 2 portables ou d'un PC et d'un smartphone, chacun peut communiquer l'un avec l'autre pour échanger des fichiers sans qu'il soit nécessaire d'utiliser un routeur. Sous Windows, ce réseau est aussi appelé « réseau ad hoc », c'est-à-dire un réseau chargé d'une mission particulière pour un temps limité :

- **Réseau maillé** : ce type de réseau exploite pleinement le potentiel du protocole ZigBee. Dans ce réseau sont présents un coordinateur et différents « End Device » pris en charge par le Routeur (Full Function Device) qui est utilisé pour étendre et acheminer les communications de manière dynamique. Ce type de réseau peut être implémenté uniquement avec les modules de la « série 2 ». Dans les réseaux maillés, le coordinateur est en mesure de configurer dynamiquement d'autres modules du réseau, chaque dispositif de type Routeur peut effectuer des opérations d'aiguillage et à son tour communiquer avec d'autres Routeurs. Cela permet une extension dynamique du réseau, il est possible d'insérer « à la volée » des nouveaux nœuds en laissant le réseau s'auto-configurer et en augmentant ainsi la distance parcourue. Si la communication avec des dispositifs est perdue, il est possible de la récupérer en la restaurant automatiquement par le biais d'autres nœuds. Le réseau devient alors très robuste, il est pratiquement insensible aux perturbations (voir la figure 15).

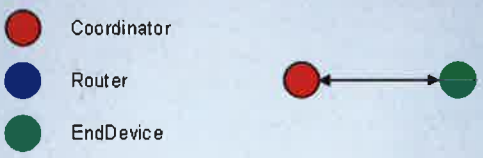


Figure 12 : exemple de réseau point à point (PPP).

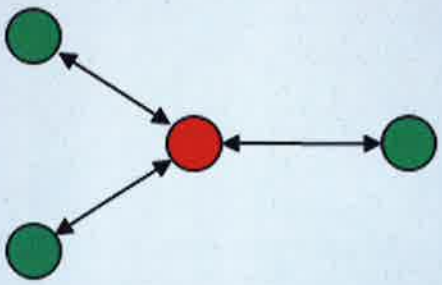


Figure 13 : exemple de réseau point à multipoint (PMP).

Figure 14 : exemple de réseau pair à pair (peer-to-peer ou P2P).

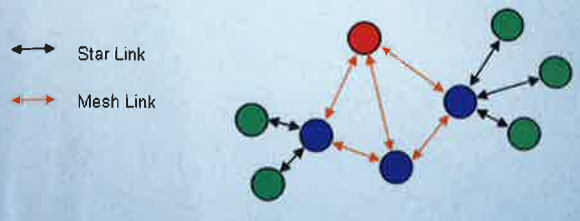


Figure 15 : exemple de réseau maillé.

Comparaison entre les modules 433 MHz et 868 MHz

Une autre grande famille de modules radio, disponible sur le marché et utilisée par de nombreux amateurs pour des applications sans fil avec des microcontrôleurs, utilise les fréquences de transmission 433 MHz ou 868 MHz. Par rapport aux modules « XBee », ces modules radio offrent certains avantages tels qu'une tension d'alimentation de 5 V directement compatible avec celle des microcontrôleurs, un coût légèrement inférieur et une connectique au pas standard de 2,54 mm.

Par contre les modules « XBee » occupent moins d'espace, car leur connecteur est au pas de 2 mm, de plus ils permettent une communication bidirectionnelle, pas toujours disponible sur les modules économiques à 433 MHz ou 868 MHz. Un autre avantage réside dans la capacité à adresser des modules « XBee » de manière unique, car chaque module est doté d'un numéro de série unique. Cela permet par exemple à deux modules de communiquer uniquement entre eux, sans tenir compte des signaux provenant d'autres modules dans le voisinage. Enfin les modules « XBee » disposent d'une logique interne permettant la correction des erreurs et d'un cryptage des données sur 128 bits.

Les différents modes de communication mis en œuvre par les modules « XBee » sont au nombre de 3 et sont décrits ci-après :

- **Mode transparent (pprz)** : ce mode exige que chaque module soit configuré en tant que « End Device » et que l'association entre « End Device » dans tous les modules soit désactivée. Chaque périphérique (dispositif) du réseau doit avoir les mêmes paramètres « ID » et « CH ». Ce mode est appelé « Transparent Mode », deux périphériques sont utilisés comme un modem sans fil, ils se comportent comme un câble série ordinaire. Les données envoyées au module RX arrivent directement au module TX, sans devoir programmer ou paramétrer quoi que ce soit, à l'exception de la vitesse de transmission de la communication. Le compactage des données et l'ajout d'une somme de contrôle (checksum) pour le contrôle des erreurs permettent d'envoyer les données :
- **Mode de commande (Command mode)** : en envoyant les 3 caractères spéciaux « +++ », le module « XBee »

passe en mode de commande et peut être configuré par des commandes simples de type AT. Ces commandes permettent de modifier la vitesse de transmission, l'adresse de destination, etc. :

- **API mode** : ce mode est le plus avancé, il peut configurer tous les paramètres d'un module à l'aide de commandes AT sans entrer dans le « Command mode ». Il est possible de gérer les communications avec les autres modules, mais aussi de recevoir des notifications relatives aux nœuds associés, dissociés ou réinitialisés. Ce mode permet d'obtenir des informations sur les adresses des nœuds du réseau, ou de recevoir des informations sur l'état d'une transmission (réussie) et d'identifier l'adresse source d'une réception. En « API Command Mode », le module peut être configuré avec un nombre de paramètres plus grand qu'avec le réseau. Il peut être programmé pour un fonctionnement autonome (stand-alone) avec la gestion des lignes d'entrées/sorties et du convertisseur A/D sans aucun support extérieur. Par exemple, il est possible de configurer une broche analogique afin d'acquérir automatiquement à des intervalles réguliers l'état d'un capteur et d'envoyer les données par radio. Avec le mode API, le module peut être configuré directement pour une application « Host » (le microcontrôleur dans notre cas).

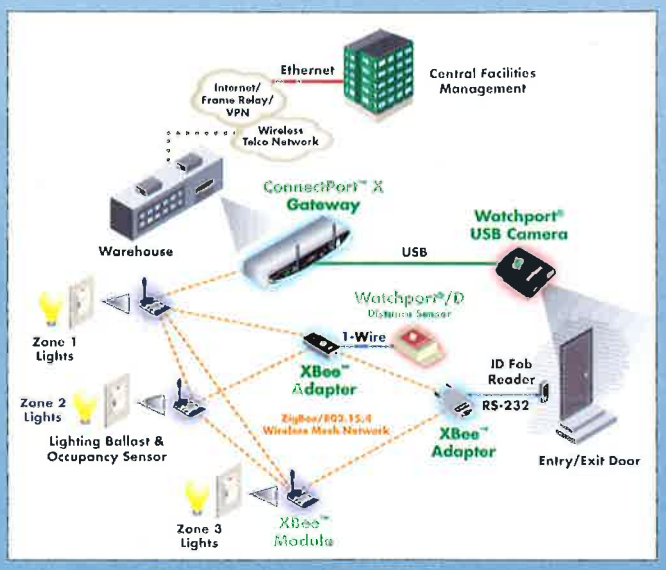


Figure 16 : exemple d'un réseau maillé pour la gestion de l'éclairage d'un grand bâtiment.

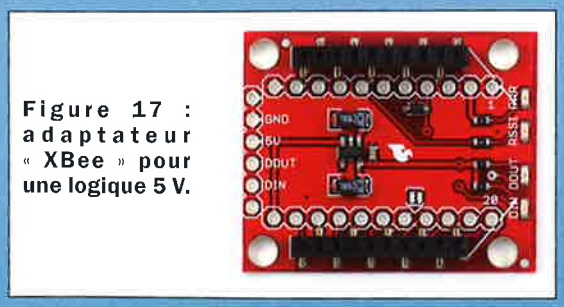


Figure 17 : adaptateur « XBee » pour une logique 5 V.

Figure 18 : adaptateur « XBee » pour une logique 5 V avec le module « XBee » inséré.

Utilisation pratique

Comme vous l'avez deviné, les modules « XBee » sont complets et sont conçus pour être utilisés dans des réseaux ayant une certaine complexité, dont l'étude ne sera pas abordée dans ce cours Arduino.

Pour nos applications, il suffit d'un microcontrôleur qui communique avec un autre ou avec un PC dans un petit réseau personnel (ou Personal Area Network pour PAN), avec un petit nombre de périphériques.

Le premier problème concerne la connexion matérielle (hardware) entre le module « XBee » et le microcontrôleur. Les modules « XBee » fonctionnent avec une tension de **3,3 V** et disposent d'un connecteur **au pas de 2 mm**, donc ils ne peuvent pas être utilisés directement sur une plaque d'essai.

En ce qui concerne le câblage entre le microcontrôleur et le module « XBee », nous pouvons considérer que la broche TX du « XBee » peut être connectée directement à la broche RX du microcontrôleur, car ce dernier interprète une tension de 3,3 V comme un niveau logique haut (1).

La broche TX du microcontrôleur peut être raccordée à la broche RX du module « XBee » par l'intermédiaire d'un diviseur de tension qui fait chuter la tension de 5 V à 3,3 V.

Par exemple en interposant 3 diodes en série soit (3 * 0,6 V = 1,8 V), la cathode étant orientée vers la broche RX du module « Xbee » et l'anode vers la broche TX du microcontrôleur, Il est nécessaire que les résistances de « pull-up » du module « XBee » soient activées. Il existe dans le commerce des interfaces appropriées pour ces fonctions, elles sont représentées en figures 17 et 18.

Vous remarquerez que ces modules comportent un grand nombre de broches. Cependant pour une utilisation pratique seulement quelques broches sont nécessaires, à savoir : VCC, GND, DOUT et DIN, ainsi que les broches utilisées pour la mise à jour du firmware. Donc, pour une mise en œuvre minimale nous aurons besoins des broches : VCC, GND, DIN, DOUT, RTS et DTR.

Chaque module comporte une résistance de pull- up de 50 kΩ sur la broche de réinitialisation (RESET). Il n'est pas strictement nécessaire d'utiliser cette broche, nous pouvons la laisser déconnectée. Certaines entrées peuvent être configurées avec une résistance de pull-up, les broches non utilisées doivent être de préférence déconnectées.

Tableau 4 - brochage des modules XBee.

Broche	Nom	Direction	Description
1	VCC	-	alimentation
2	DOUT	sortie	sortie des données de l'UART
3	DIN / CONFIG	entrée	entrée des données de l'UART
4	DIO12	entrée/sortie	entrée/sortie digitale 12
5	RESET	entrée	réinitialisation du module (impulsion minimale 200 ns)
6	PWM0 / RSSI / DIO10	entrée/sortie	sortie PWM 0 / Indicateur de puissance du signal RX / entrée/sortie digitale 10
7	PWM / DIO11	entrée/sortie	entrée/sortie digitale 11
8	[reserved]	-	non connectée
9	DTR / SLEEP_RQ/ DIO8	entrée/sortie	broche de contrôle du mode veille ou entrée/sortie digitale 8
10	GND	-	masse
11	DIO4	entrée/sortie	entrée/sortie digitale 4
12	CTS / DIO7	entrée/sortie	efface pour envoyer le Flow Control ou entrée/sortie digitale 7
13	ON / SLEEP / DIO9	sortie	indicateur d'état du module ou entrée/sortie digitale 9
14	[reserved]	-	non connectée
15	Associate / DIO5	entrée/sortie	indicateur des modules associés, entrée/sortie digitale 5
16	RTS / DIO6	entrée/sortie	demande d'envoi du Flow Control, entrée/sortie digitale 6
17	AD3 / DIO3	entrée/sortie	entrée analogique 3 ou entrée/sortie digitale 3
18	AD2 / DIO2	entrée/sortie	entrée analogique 2 ou entrée/sortie digitale 2
19	AD1 / DIO1	entrée/sortie	entrée analogique 1 ou entrée/sortie digitale 1
20	AD0 / DIO0 / Commissioning Button	entrée/sortie	entrée analogique 3 ou entrée/sortie digitale 3 ou bouton de mise en service

Configuration des modules « XBee »

La configuration des modules « XBee » peut s'effectuer à l'aide de **commandes AT**, mais il est plus facile d'utiliser l'environnement de développement « **XCTU** » que vous pouvez télécharger gratuitement sur le site **www.digi.com**. Voici le lien exact :

http://www.digi.com/products/xbee-rf-solutions/xctu-software/xctu#productsupport-utilities.

Vous y trouverez les versions pour Windows, Linux et MacOSX.

Ce logiciel vous permet également de tester le réseau radio en mesurant l'intensité du signal et la qualité (erreurs), il permet également la mise à jour du firmware des modules.

Pour interfacer un module « XBee » avec un PC, il est nécessaire de se procurer un adaptateur, disponible en version USB. Ce dernier est équipé d'un **convertisseur USB/série** « **FDTI** », vous devez d'abord installer le driver (pilote) que vous pouvez télécharger à l'adresse suivante :

http://ftdichip.com/Drivers/VCP.htm

Si vous utilisez une carte Arduino avec une interface USB, ces pilotes sont normalement déjà installés.

L'adaptateur USB pour module « XBee » comporte 2 LED qui permettent de surveiller les transmissions au niveau des broches TX et RX. Une 3ème LED est reliée à la ligne « RSSI » du modue « XBee » et une 4ème indique la présence de l'alimentation. Insérez le module « XBee » dans l'adaptateur et connectez-le à votre PC. Si vous utilisez la version USB, allez dans le « Gestionnaire de périphériques » de Windows pour connaitre

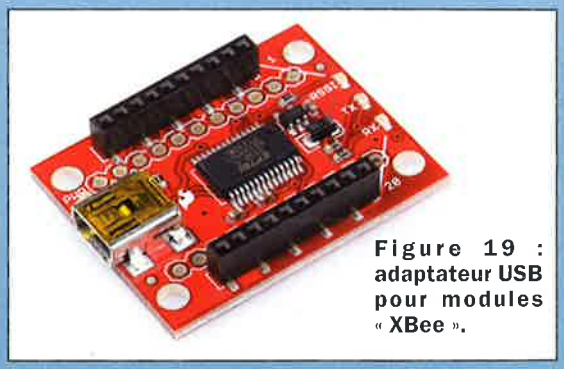


Figure 19 : adaptateur USB pour modules « XBee ».

Figure 20 : adaptateur USB-XBee en version « Dongle ».

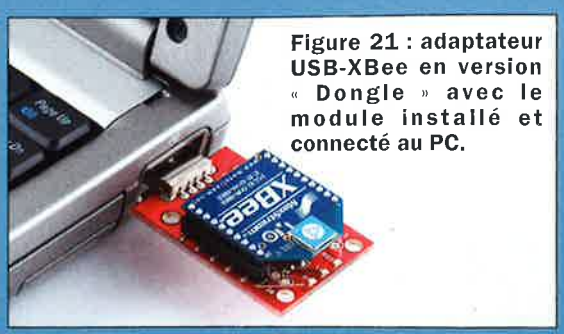


Figure 21 : adaptateur USB-XBee en version « Dongle » avec le module installé et connecté au PC.

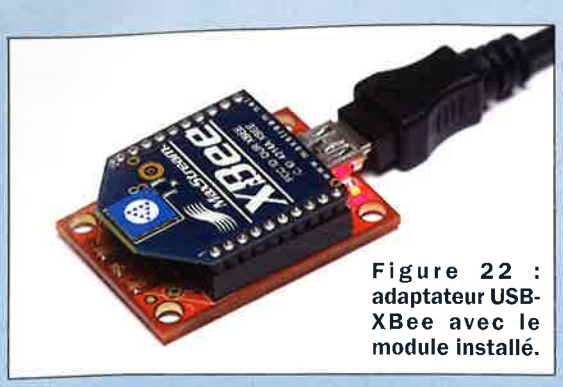


Figure 22 : adaptateur USB-XBee avec le module installé.

Figure 23 : adaptateur RS232-XBee avec le module installé.

le port COM sur lequel il est installé (voir la figure 25). Lancez le programme « XCTU » et sélectionnez l'onglet « PC Settings », puis sélectionnez le port COM utilisé par l'adaptateur.

Si le port COM n'est pas présent dans la liste, cliquez sur l'onglet « User Com Ports » (situé vers le bas de la fenêtre) et entrez manuellement dans le champ « Com Port Number » le numéro de port puis cliquez sur « Add ». Normalement le port COM doit apparaître dans la liste au-dessus (voir la figure 26).

Par défaut, les modules sont configurés pour une communication avec le protocole suivant : 9600 bauds, 8 bits de données, 1 bit d'arrêt et pas de bit de parité.

Cliquez sur le bouton « Test/Query » et laissez le logiciel identifier le module. Si tout se passe bien, vous devriez voir le type de module et la version du firmware (voir la figure 27). Par exemple, en insérant un module « XBee PRO », nous avons obtenu la réponse suivante :

```
Type de modem = XBP24
Version du firmware = 10E6 (version 1.x)
```

Maintenant cliquez sur l'onglet « Modem Configuration », ensuite sur l'onglet « Modem Parameters and Firmware » et enfin cliquez sur le bouton « Read » pour obtenir la configuration de votre module.

Si le module est plus récent que le logiciel, vous aurez besoin de mettre à jour la base de données des fonctions. Pour cela connectez-vous à Internet et cliquez sur « Download New Versions » (voir la figure 28).

Avec la commande « Read », vous obtenez l'ensemble de la configuration présente dans la mémoire du module (voir la figure 29).

Analysons ci-après, en détail, les paramètres de base :

- « **CH** » est le **canal de communication** (c'est-à-dire la fréquence du module radio). Seuls les modules ayant le même canal peuvent communiquer entre eux. Les modules avec un « CH » différent créent un sous-réseau dans lequel il peuvent communiquer et coexister avec d'autres réseaux ;
- « **ID** » est l'**identifiant** du **PAN** (Personal Area Network) sur lequel le module fonctionne. Seuls les modules ayant le même identifiant (ID) et le même canal (CH) peuvent communiquer entre eux. Dans ce cas, ils font partie du même PAN ;
- « **DH - DL** » est l'**adresse de destination du message**. « DH » représente la partie « haute » de l'adresse. En prenant DH = 0 et DL inférieur à 0xFFFF, les messages envoyés par ce module sont reçus par tous les autres modules du PAN qui ont le paramètre « MY » égal à DL. « MY » étant l'adresse de ces modules. Si DH = 0 et DL = 0xFFFF, la transmission de ce module sera reçue par tous les modules. Si DH > 0 et DL > 0xFFFF, la transmission de ce module sera reçue exclusivement par les modules dont le numéro de série SH (récepteur) sera égal à DH (émetteur) et SL (récepteur) égal à DL (émetteur). Tout ceci est valable pour les modules qui font partie du même PAN ;
- « **SH - SL** » représente l'**identifiant ou numéro de série** (32 bits au total) unique paramétré en usine. Chaque module a un identifiant différent des autres, cet identifiant ne peut pas être changé ;
- « **MY** » est l'**adresse source**, à ne pas confondre avec le numéro de série unique de chaque module. Comme nous le verrons, ce paramètre est utile lorsque nous devons adresser des transmissions uniquement vers certains modules. En paramétrant MY = 0xFFFF, cela désactive la réception avec une adresse de 16 bits ;
- « **RANGE** » est compris entre 0 et 0XFFFF ;
- « **PL** » est un paramètre qui permet de régler la **puissance de transmission**. Plus la puissance est faible, plus la consommation est faible ;
- « **BD** » permet de régler la **vitesse de transmission** (baud-rate) pour la communication série. Il est important de connaitre la valeur réglée sur le module, sinon vous ne pourrez pas y accéder, ou lire la configuration ou la définir. Par défaut dans le logiciel « XCTU », la vitesse est réglée à 9600 bauds, cela est également le paramétrage à utiliser sur Arduino pour la communication avec un module « XBee ».

Les modules de la « série 1 » sont déjà configurés pour une communication en « Mode transparent », fonctionnant dans un réseau de type « peer-to-peer » dans lequel chaque module est utilisé comme un « End Device ». Nous pouvons donc utiliser deux modules « XBee » (série 1) pour remplacer une connexion de type filaire RS232 entre deux périphériques (Mode transparent) sans qu'il soit nécessaire de les programmer.

Nous utilisons ensuite un second module « XBee série 1 » (XBP24) qui est inséré dans un deuxième adaptateur et

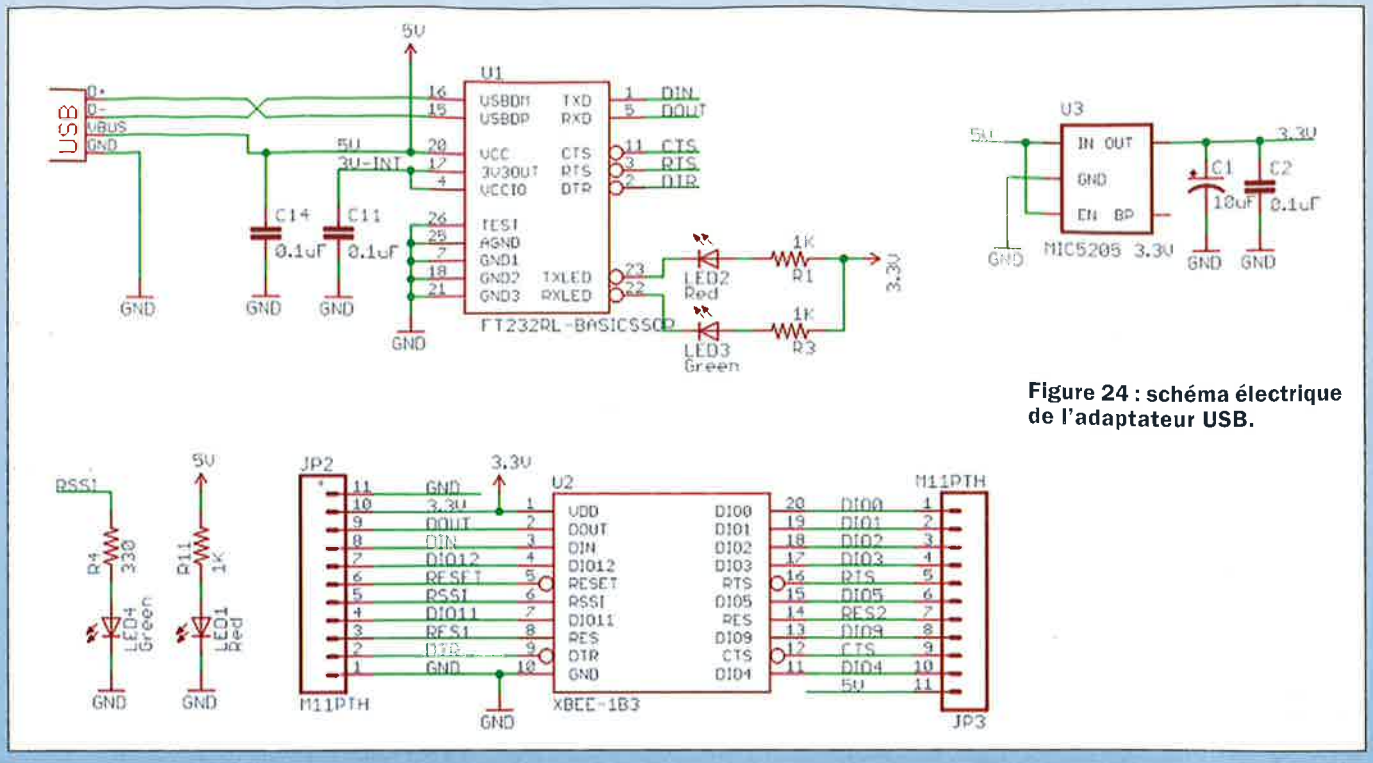


Figure 24 : schéma électrique de l'adaptateur USB.

interfacé à un second PC, ou comme nous allons le faire, utiliser le module « XBee » avec une carte Arduino pour établir une connexion sans fil entre un PC et Arduino.

Pour connecter le module « XBee » à Arduino, nous utilisons l'adaptateur générique (voir les figures 17 et 18). Nous réalisons les connexions suivantes :

- +5 V d'Arduino au +5 V de l'adaptateur XBee ;
- GND d'Arduino à GND de l'adaptateur XBee ;
- broche TX d'Arduino à la broche RX de l'adaptateur XBee ;
- broche RX d'Arduino à la broche TX de l'adaptateur XBee.

Rappelons que la broche TX d'Arduino doit être dirigée vers la broche RX du convertisseur FDTI, tandis que la broche RX d'Arduino doit être dirigée vers la broche TX du convertisseur FDTI. Ceci pose un problème car le module « XBee » et le convertisseur FDTI envoient des données sur la même ligne. La solution est d'utiliser la carte « **XBee Shield** » qui permet de réaliser toutes les connexions facilement et rapidement. Cette carte est **disponible dans le commerce** (sans le module Xbee) **déjà montée**.

La version actuelle de cette carte est la version 1.1, c'est celle que nous avons utilisée (voir la figure 31).

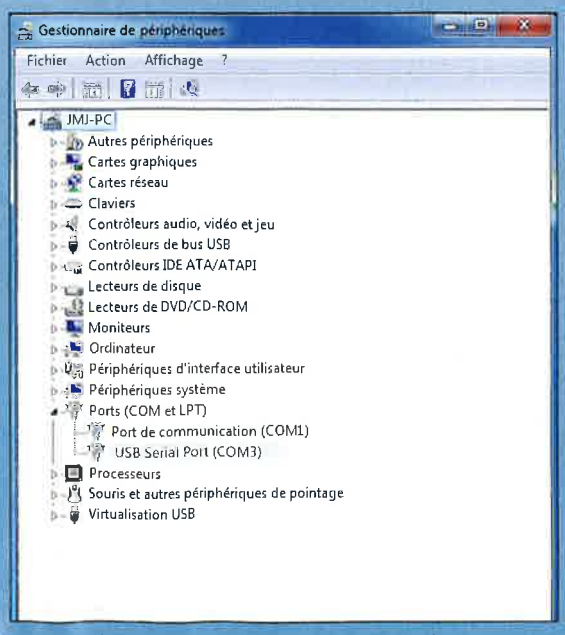


Figure 25 : visualisation des périphériques série disponibles.

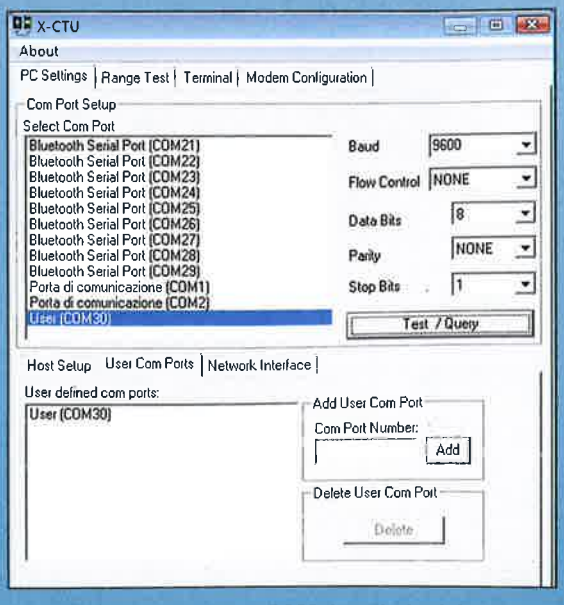


Figure 26 : sélection du port COM dans le logiciel « XCTU ».

Elle dispose d'emplacements appropriés pour accepter des composants traditionnels et des composants CMS.

D'après le schéma électrique de la carte « XBee Shield » (voir la figure 32), la tension d'alimentation du module « XBee » est produite en interne par un régulateur approprié à partir de la tension +5 V de la carte Arduino.

Les seules broches utilisées du module « Xbee » sont « DIN » (RX) et « DOUT » (TX). Elles sont reportées respectivement sur deux cavaliers (jumpers) « JP3 » et « JP2 ». La broche « DIN » comporte un diviseur de tension constitué par les résistances R1 et R2 de manière à être commandée par une

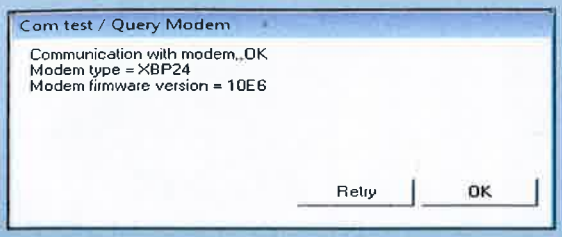


Figure 27 : réponse de la fonction Test/Query dans « XCTU ».

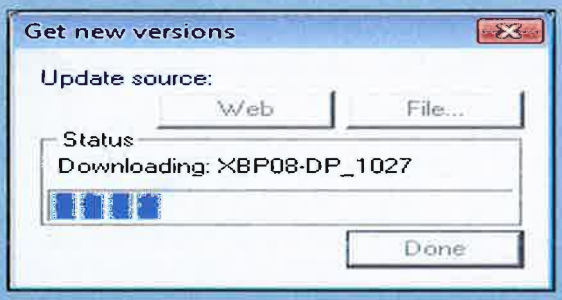


Figure 28 : phase de téléchargement et de mise à jour des modules.

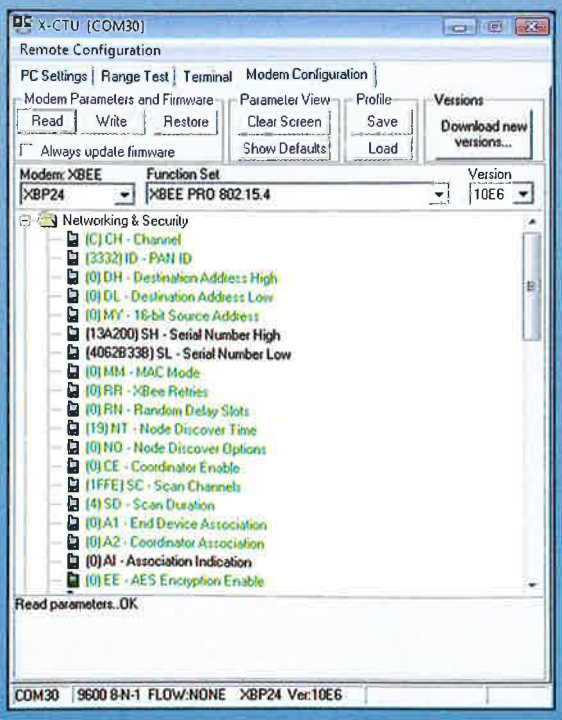


Figure 29 : lecture des paramètres par défaut du module « XBee ».

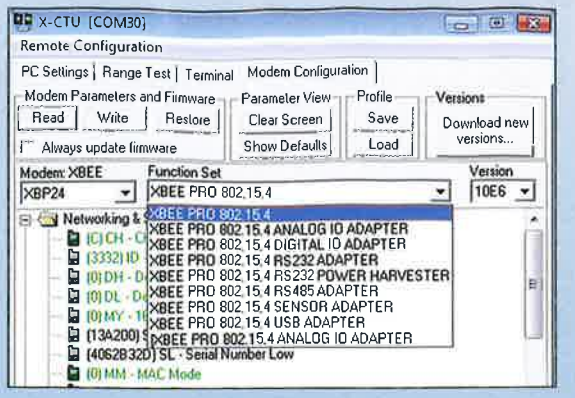


Figure 30 : liste des options du menu « Function Set » dans « XCTU ».

logique ayant une tension de 5 V. Les cavaliers permettent de configurer l'utilisation de la carte « XBee Shield » avec des microcontrôleurs Atmel ou bien avec un port USB.

Avec le cavalier en mode « XBee », la broche « DOUT » du module est reliée à la broche RX du microcontrôleur, qui à son tour est reliée de façon permanente avec la ligne TX du circuit FDTI. De la même façon, la broche « DIN » du « XBee » est connectée à la broche TX du microcontrôleur, qui à son tour est reliée de façon permanente à la ligne RX du FDTI. Ainsi, l'envoi de données à partir du microcontrôleur se fait soit par l'intermédiaire du module « XBee », soit par le port USB.

Le microcontrôleur est capable de recevoir des données provenant du module « XBee » et non du port USB, car les données provenant de l'USB et adressées au microcontrôleur seraient en conflit avec la broche « DOUT » du module « XBee ». De cette manière le microcontrôleur Arduino utilise le module « XBee » pour transmettre et recevoir des données.

Avec le cavalier en mode USB, la broche « DOUT » du module « XBee » est reliée à la broche RX du FDTI ainsi qu'à la broche TX du microcontrôleur. La broche « DIN » du « XBee » est connectée à la broche TX du FDTI et à la broche RX du microcontrôleur. Ainsi le microcontrôleur Arduino peut alors communiquer normalement avec un PC via le port USB. Utilisez ce mode pour programmer le microcontrôleur via le port USB.

En cas de retrait du microcontrôleur, le PC peut communiquer directement avec le module « XBee » via le port USB mais sans les lignes « RTS » et « DTR », car elles ne sont pas gérées par la carte adaptateur USB-XBee.

Cela permet d'utiliser le module « XBee » directement à partir d'un PC, par exemple pour l'acquisition de données provenant d'un réseau de capteurs disposés dans différents endroits.

Si le microcontrôleur est relié à la carte, il pourra toujours communiquer normalement via le port USB, mais ni le PC et ni le microcontrôleur ne pourront communiquer avec le module « XBee ».

Tableau 5 - Liste des principaux paramètres par défaut du moduleXBP24 (XBee serie1).

Paramètre	Défaut	Fonction
CH	C	Configure/lit le numéro de canal (utilise les numéros de canaux 802.15.4). RANGE : 0XC - 0X17
ID	3332	Configure l'identifiant du PAN (Personal Area Network). Si ID = 0xFFFF envoie le message à tous les PAN. RANGE : 0 - 0XFFFF
DH	0	Configure/lit les 32 bits supérieurs des 64 bits de l'adresse de destination. Si DH = 0 et DL inférieur à 0xFFFF, la transmission utilise une adresse de 16 bits. 0x000000000000FFFF est l'adresse de diffusion (broadcast address) pour le PAN. RANGE :0 - 0XFFFFFFFF
DL	0	Configure/lit les 32 bits inférieurs des 64 bits de l'adresse de destination. Si DH = 0 et DL inférieur à 0xFFFF, la transmission utilise une adresse de 16 bits. 0x000000000000FFFF est l'adresse de diffusion (broadcast address) pour le PAN. RANGE : 0 - 0XFFFFFFFF
MY	0	Configure/lit les 16 bits de l'adresse source pour le modem. Si MY = 0xFFFF désactive la réception des données avec une adresse de 16 bits. Les 64 bits de l'adresse source correspondent au numéro de série et sont toujours activés. RANGE :0 - 0XFFFF
SH	13A200	Lit les 32 bits supérieurs des adresses sources uniques IEEE 64-bit des modems. 64-bit adresse source est activé.
SL	4062B32D	Lit les 32 bits inférieurs des adresses sources uniques IEEE 64-bit des modems. 64-bit adresse source est activé.

Nous allons maintenant mettre en œuvre un exemple simple, dans lequel une donnée est envoyée par le PC à la carte Arduino qui répond avec la même donnée reçue. Pour ainsi dire, cela s'apparente à une sorte d'écho qui nous permet de vérifier si la connexion sans fil fonctionne.

Le **Listing 1** montre le code source correspondant. Le programme attend simplement la réception d'un caractère. Le caractère « h » allume la LED sur la carte, tandis que le caractère « s » l'éteint.

Dans tous les cas, les informations sont renvoyées par le module « Xbee » et par le port USB.

Si Arduino est connecté à votre PC, ouvrez l'environnement de développement Arduino et dans le menu « Outils » activez le « Moniteur Série » (réglé à 9600 bauds). Le caractère reçu doit s'afficher.

Connectez ensuite le module « Xbee » à la carte « Arduino Shield », qui à son tour doit être insérée sur une carte Arduino (dans notre cas une Duemilanove). Nous avons configuré les cavaliers sur le mode USB. Connectez la carte au PC et programmez-la normalement (attention au port COM utilisé). Maintenant, positionnez les cavaliers sur le mode « XBee » de manière à ce que le microcontrôleur ATmega utilise le module « XBee ».

Listing 1

```
/*
  XBee_01
  Test du module Xbee
  Réception et émission d'un caractère
  Allume la LED 13 si le caractère h est reçu
  Eteint la LED 13 si le caractère s est reçu
*/

byte ChRX =0;                          // caractère reçu
int Led = 13;                          // LED connectée sur la broche digitale 13

void setup()
{
   pinMode(Led, OUTPUT);               // broche LED configurée en sortie
   Serial.begin(9600);                 // vitesse de transmission série = BD XBee!
}

void loop()
{
   while (Serial.available() > 0) {
                                       // attend l'arrivée d'un caractère :
      ChRX = Serial.read();
      if (ChRX == 'h')
        digitalWrite(Led, HIGH);       // allume la LED

      if (ChRX=='s')
        digitalWrite(Led, LOW);        // éteint la LED

      Serial.print(ChRX);              //affiche le caractère
      delay(100);
    }
}
```

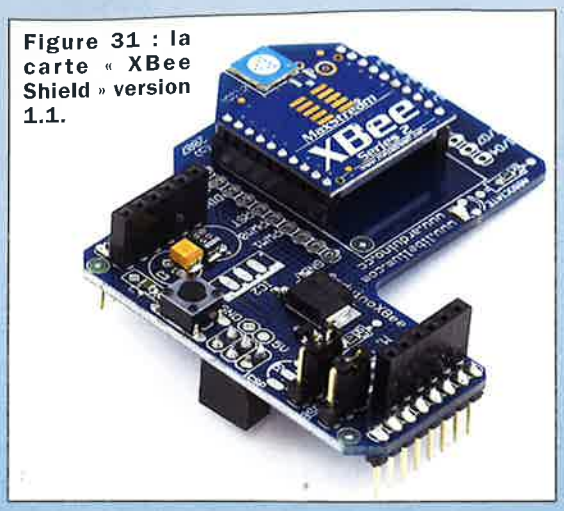


Figure 31 : la carte « XBee Shield » version 1.1.

Sur la carte « XBee Shield », la LED doit clignoter pour indiquer l'association avec le 1er module « XBee ». Sur l'adaptateur de cette dernière la LED « RSSI » doit s'allumer pour indiquer la réception des données (présence d'un signal).

Lancez « XCTU », il détectera de manière automatique l'adaptateur USB-XBee, puis cliquez sur l'onglet « **Terminal** ». Commencez à taper des caractères, un à la fois. Il suffit d'appuyer sur la touche du caractère correspondant et il est immédiatement envoyé. Si tout fonctionne correctement, le même caractère doit être retourné.

Dans l'environnement de développement d'Arduino (IDE), vous verrez apparaître les caractères arrivant sur la carte « XBee Shield ».

Dans « XCTU », cliquez sur l'onglet « **Range Test** », puis en bas à droite dans le champ situé à côté du bouton « **Create Data** » tapez 1 et cliquez sur « **Create Data** ». Le caractère « **0** » apparaît dans la fenêtre du haut, et correspond au caractère à envoyer. Si vous tapez 16, vous verrez apparaître les caractères « 0123456789:;<=>? », et ils seront envoyés.

Cliquez maintenant sur le bouton « **Start** » (en haut vers la gauche) pour que le caractère soit envoyé en continu. Le logiciel vérifie si la réception est correcte ainsi que le niveau relatif.

Pourquoi envoyer un seul caractère à la fois ? Cela est dû au fait que dans le sketch, il est prévu justement l'envoi et la réception d'un seul caractère. Cependant si nous connectons les broches RX et TX avec un second module, nous pourrions transmettre plusieurs caractères à la fois.

La ligne « RSSI » est programmée en tant que sortie PWM (paramètre P0 = 1) avec un rapport cyclique variant en fonction du niveau du signal qui est évalué à chaque réception des données.

Avec un filtre passe-bas, nous obtenons une tension proportionnelle à la puissance du signal reçu.

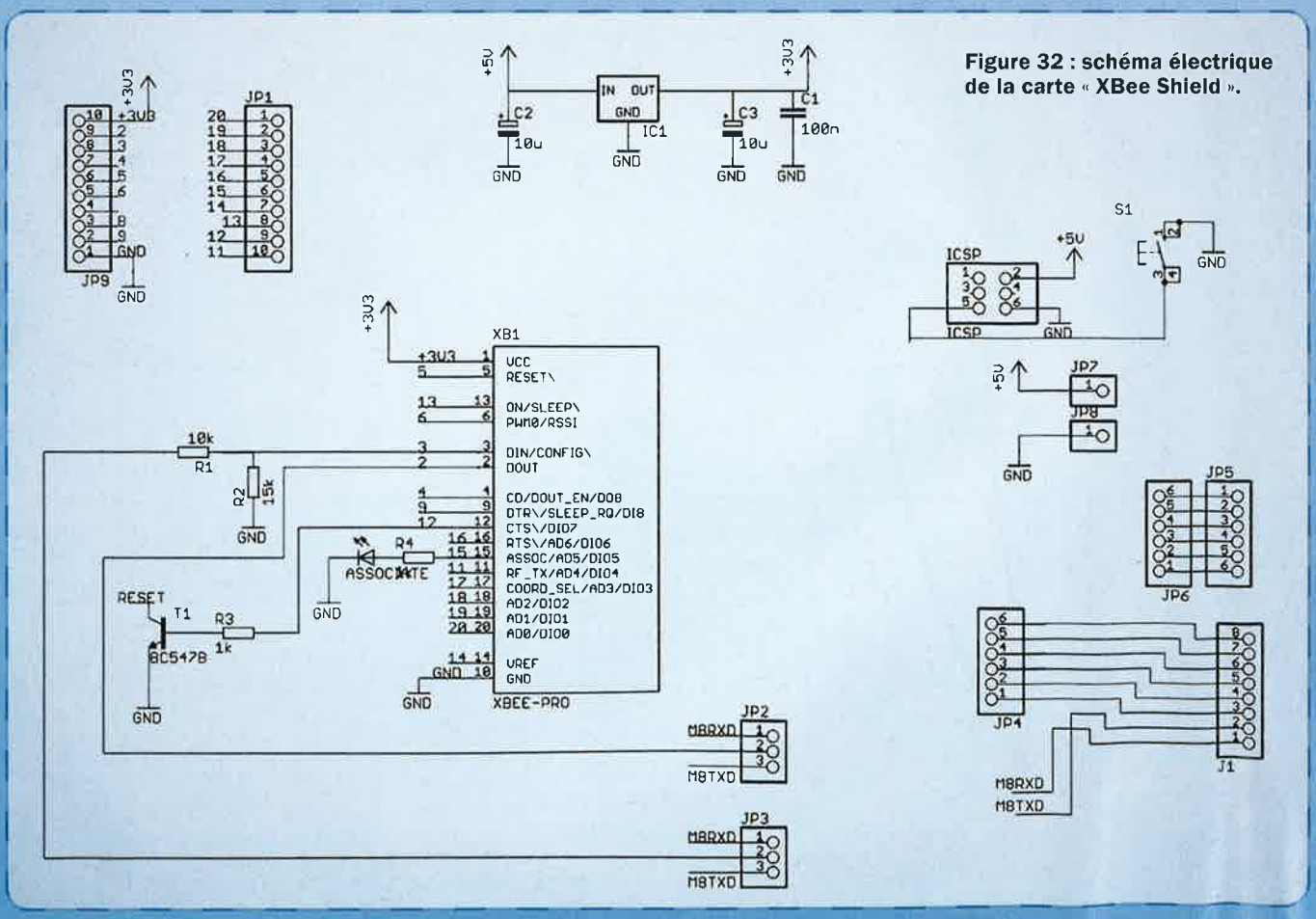


Figure 32 : schéma électrique de la carte « XBee Shield ».

La sortie PWM reste active pendant la durée correspondante à la valeur du paramètre « RP » (100 ms).

Le schéma général adopté est représenté en figure 37. Le second PC n'est pas strictement nécessaire, car le module « XBee » est géré par Arduino. Vous pouvez utiliser le même PC, cependant il faudra prêter attention au port COM utilisé.

À la place du logiciel « XCTU » pour envoyer et recevoir des données avec la carte Arduino, vous pouvez également utiliser le « Moniteur Série » de l'IDE d'Arduino.

Par exemple, la carte Arduino pourrait être insérée dans un robot pour gérer ses mouvements à distance, mais aussi pour acquérir des données sur le terrain.

Personnalisation de la communication

Si nous devions utiliser 3 modules avec une configuration par défaut, les données transmises par l'un d'entre eux atteindraient certainement les deux autres. Nous allons étudier maintenant la possibilité de configurer les modules de manière différente.

Par exemple, nous voulons contrôler un robot à distance, de sorte qu'il n'y ait pas d'interactions avec d'autres robots qui peuvent être présents dans le même environnement et qui utilisent la même technologie sans fil. Le module « XBee A » doit seulement communiquer avec le module « XBee B ».

Comme mentionné auparavant, les deux modules doivent avoir le même canal (CH) et le même identifiant (ID), ils appartiennent donc au même réseau PAN. Ils doivent donc être configurés de la même façon.

Module A : $DL_A=SL_B$ $DH_A=SH_B$
Module B : $DL_B=SL_A$ $DH_B=SH_A$

Après avoir inséré comme adresse de destination dans chaque module le numéro de série de l'autre, la communication ne peut s'effectuer qu'entre ces deux modules. Et si personne d'autre ne connait les numéros de série de vos modules, personne ne pourra interférer dans la communication entre vos modules.

Exemples d'utilisation

Examinons maintenant quelques exemples de configuration des systèmes sans fil avec des modules « XBee ». Le « Module A » ne peut transmettre qu'au « Module B », alors que ce dernier peut transmettre à tout le monde. La configuration est la suivante :

Module A : $DL_A=SL_B$ $DH_A=SH_B$
Module B : $DL_B=0$ $DH_B=0$

Le « Module A » utilise comme adresse de destination le numéro de série du « Module B », qui est donc le seul à recevoir les messages provenant du « Module A ». À l'inverse le « Module B » a comme adresse de destination « 0 », il peut donc transmettre à tous les modules.

Voici un second exemple que nous pouvons réaliser, le système comprend 3 modules A, B et C qui communiquent de la manière suivante :

- A envoie des données à C ;
- B envoie des données à C ;
- C envoie des données à A et à B.

Donc les modules A et B ne communiquent pas directement entre eux, la configuration est la suivante :

Module A : $DL_A=2$ $DH_A=0$ $MY_A=1$

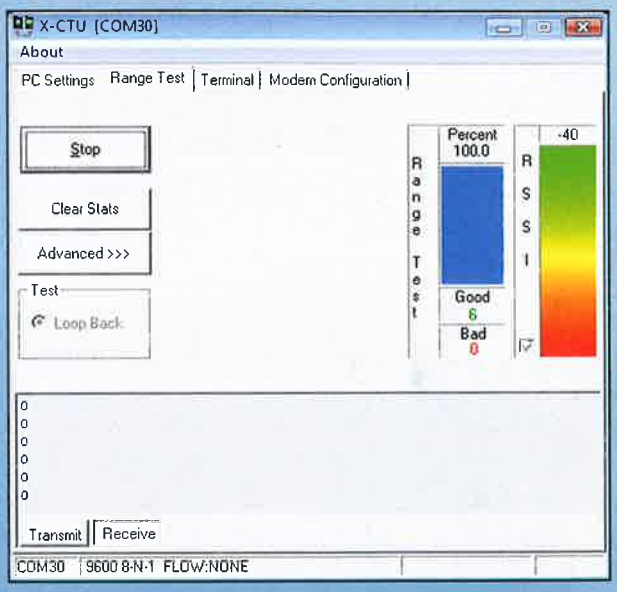


Figure 35 : fonction « Terminal » du logiciel « XCTU ».

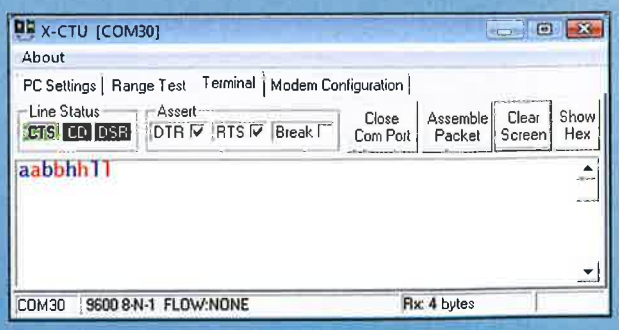


Figure 36 : fonction « Range test » du logiciel « XCTU ».

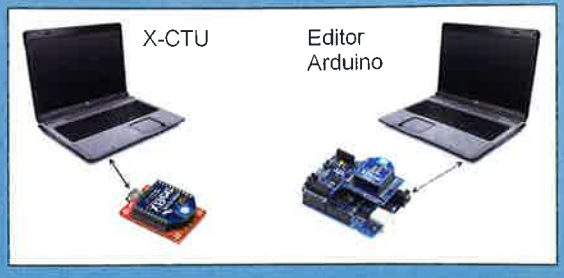


Figure 37 : schéma des éléments utilisés pour l'exemple.

Module B : DL_b=2 DH_b=0 MY_b=1
Module C : DL_c=1 DH_c=0 MY_c=2

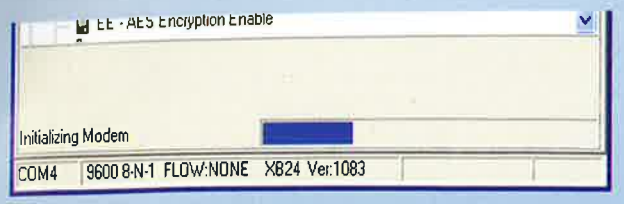


Figure 38 : début de la procédure d'écriture des nouveaux paramètres dans le module.

Figure 39 : écriture des nouveaux paramètres dans le module.

Figure 40 : écriture terminée avec succès.

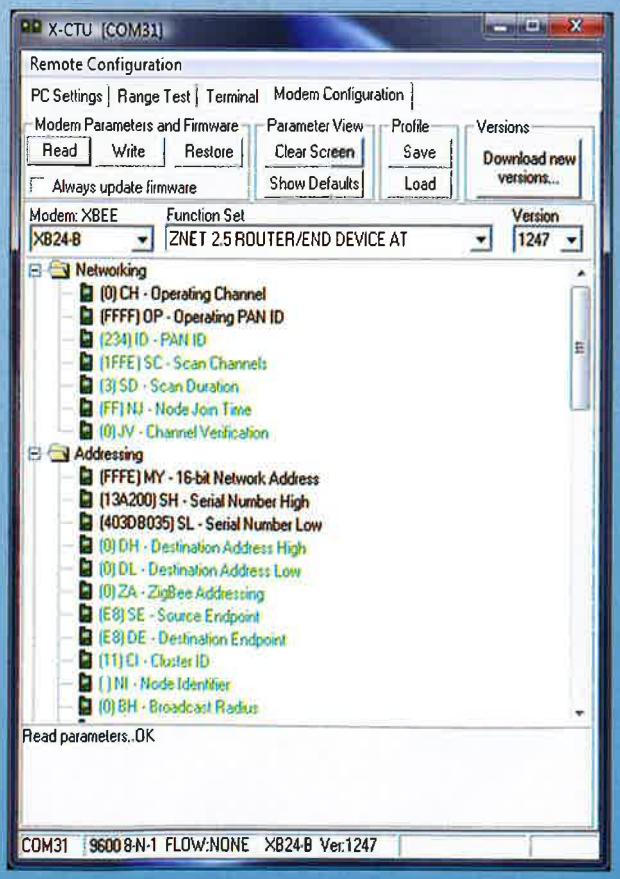


Figure 41 : lecture des paramètres par défaut d'un module « XBee » série 2.

Le « Module C » a comme adresse de destination la même valeur que l'adresse source des modules A et B, de sorte que toutes les données envoyées arriveront sur A et B.

Le « Module A » et le « Module B » ont comme adresse de destination l'adresse source du « Module C » (qui est différente de celle de A et de B), donc les transmissions arriveront seulement à ce module.

Évidemment, si vous ne disposez que d'une seule station de programmation, vous devez programmer un module puis l'autre. Utilisez le logiciel « XCTU », définissez les différents paramètres, puis appuyez sur le bouton « Write » pour écrire dans la mémoire du microcontrôleur les nouvelles valeurs.

Programmation des modules « XBee » série 2

La série 1 n'implémente pas la fonction « routeur », il est donc impossible de configurer des réseaux maillés. Avec l'introduction de la série 2, différentes méthodes de communication peuvent être mises en œuvre, elles dépendent de la version du firmware installé. Le mode de fonctionnement par défaut n'est donc pas connu à l'avance, ni même si le module supporte le « Mode transparent ».

Les versions des firmwares disponibles des modules ZigBee (OEM Modules RF) sont :

1.0xx - Coordinateur, Transparent Operation
1.1xx - Coordinateur, API Operation
1.2xx - Router, End Device, Transparent Operation
1.3xx - Router, End Device, API Operation

Vous ne pouvez pas utiliser immédiatement les modules, vous devez d'abord les programmer.

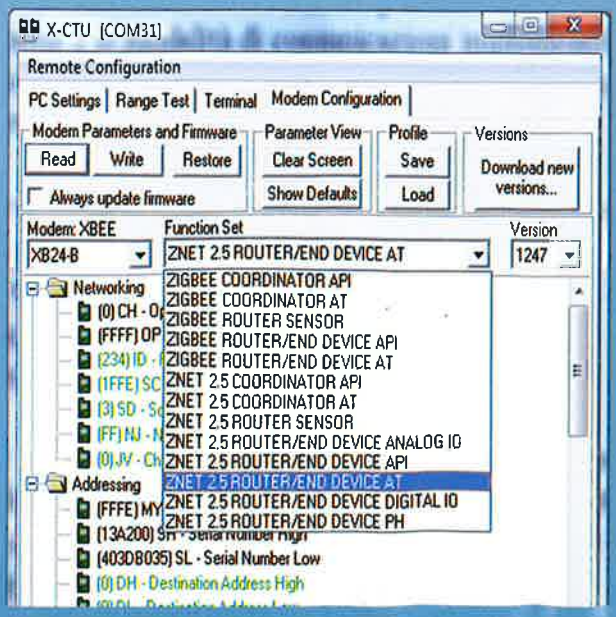


Figure 41a : les différentes possibilités du menu déroulant de « Function Set » pour un module « XBee » série 2.

Si vous voulez utiliser deux modules pour réaliser un simple réseau « point à point » en « Mode transparent », vous devez configurer un module en tant que « Coordinateur » et l'autre en tant que « End Device ».

Comme précédemment, vous devez interfacer le module à un PC, puis vous lancez « XCTU ». La lecture des paramètres par défaut pour un module « XB24-B » est visible en figure 41.

Vous remarquerez la référence du module (XB24-B), en dessous de « **Modem : XBee** », qui indique qu'il s'agit de la « série 2 ». À côté, dans le menu déroulant « **Function Set** », vous pouvez lire les différents paramètres du module. Par défaut, le canal de transmission est « **D** » et le « **PAN ID** » vaut **234**.

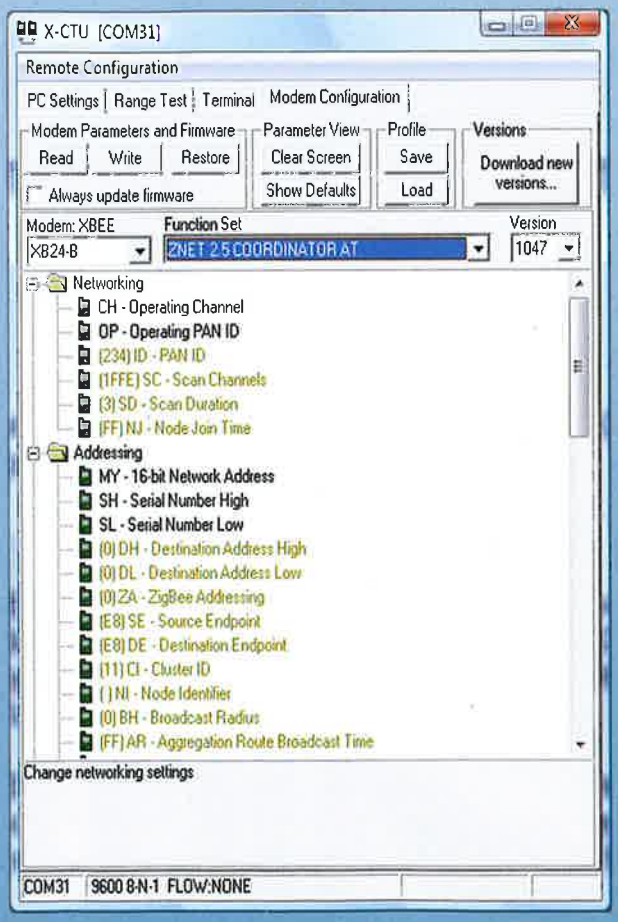


Figure 42 : configuration du module « XBee » série 2 en tant que « Coordinateur ».

Deux modules identiques, avec les mêmes paramètres par défaut, ne peuvent pas communiquer entre eux.

Configurez un module en tant que « **ZNET 2.5 ROUTER/END DEVICE AT** » et l'autre en tant que « **ZNET 2.5 COORDINATOR AT** » à l'aide du menu déroulant « **Function Set** » (voir les figures 41a et 42).

Une fois le paramètre sélectionné dans le menu déroulant, cliquez sur le bouton « **Write** ». À ce stade, le module « **End Device** » peut envoyer des données au module « **Coordinateur** », **mais pas l'inverse**.

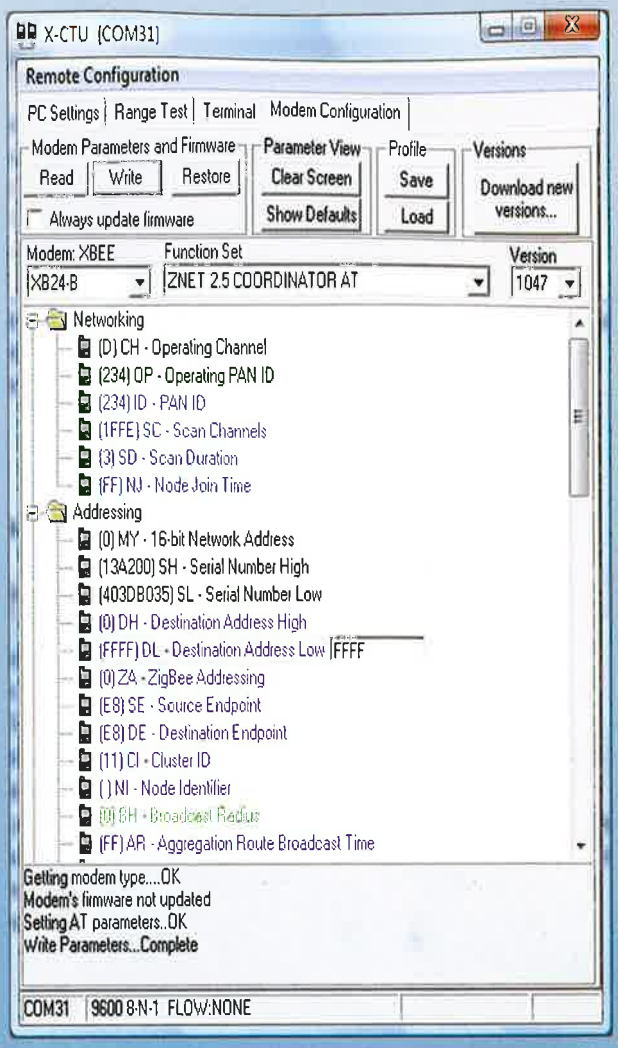


Figure 43 : configuration du paramètre « DL » du module « XBee » série 2.

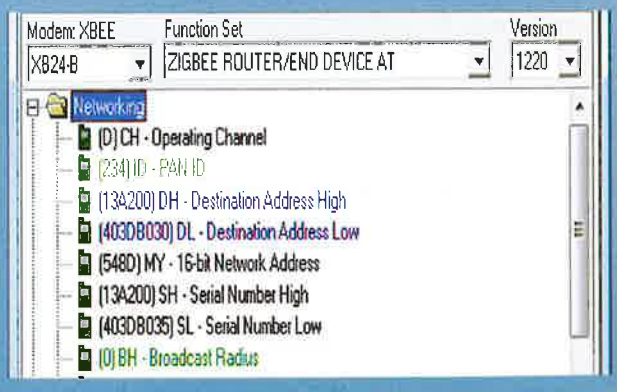


Figure 44 : module « XBee » série 2 configuration A.

Pour permettre au « Coordinateur » d'envoyer des données au « End Device », vous pouvez par exemple configurer le paramètre **DL = 0xFFFF**. Ceci habilite la transmission à tous les modules (broadcast mode), tel que représenté en figure 43.

Si vous désirez que votre « Coordinateur » envoie uniquement des données à votre « End Device », vous devrez effectuer la configuration suivante :

DHcoordinateur = SHenddevice
DLcoordinateur = SLenddevice

Il existe une troisième possibilité qui consiste à utiliser les deux modules en tant que « **ZIGBEE ROUTER/END DEVICE AT** » (voir les figures 44 et 45) avec les paramètres suivants :

$DH_A = SH_B$
$DL_A = SL_B$
$DH_B = SH_A$
$DL_B = SL_A$

Maintenant, les modules sont en mesure de dialoguer entre eux. Dans ce mode, les données envoyées sur la broche TX d'un module arrivent directement sur la broche RX du module distant et vice versa.

Le logiciel « **XCTU** » pour les modules « **XBee** » ainsi que les **drivers FTDI** sont téléchargables gratuitement sur notre site :

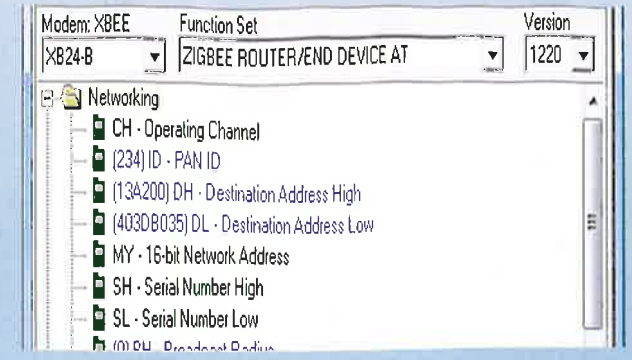


Figure 45 : module « XBee » série 2 configuration B.

www.electroniquemagazine.com dans le sommaire détaillé de la revue **136** à l'onglet « **Télécharger** ».

Dans la prochaine leçon du Cours Arduino, nous apprendrons à utiliser les SD-Card avec une carte Arduino. ■

Wattway, la route photovoltaïque Française

La société Colas effectue des tests en France sur une route photovoltaïque, appelée Wattway. Elle est constituée de panneaux solaires basés sur une technologie qui les rend indestructibles grâce à une résine spéciale qui recouvre le revêtement sur lequel circulent les véhicules. Ainsi la surface de la route produit de l'énergie sans occuper d'espace supplémentaire.

Les cellules photovoltaïques sont recouvertes d'une résine spéciale formant un revêtement qui protège le système sans dégrader le comportement routier des véhicules sur la route (adhérence dans les virages, distances de freinage, etc.).

L'énergie électrique ainsi produite est acheminée sous terre vers le réseau électrique pour y être injectée. Avec seulement 15 mètres carrés de surface photovoltaïque, il est possible de produire assez d'énergie pour alimenter un feu de circulation. En couvrant 2,5 % de la surface des routes Françaises, il serait possible de couvrir 10 % des besoins énergétiques nationaux.

La commercialisation des panneaux Wattway a commencé en début d'année 2016 en France, pour s'étendre dans le futur en Amérique du Nord.

www.wattwaybycolas.com

Cours de programmation sur IPhone

de Walter Dal Mut et Francesco Ficili

Dans cette deuxième partie du cours de programmation pour iPhone, nous allons installer et apprendre à utiliser l'environnement de développement « Xcode », grâce auquel nous pouvons écrire, déboguer et tester des applications pour iPhone et pour iPad.

Après la première partie, qui était une introduction au monde de la programmation sous le système d'exploitation d'Apple, il est temps de passer aux choses sérieuses en commençant par l'installation de l'environnement de développement « **Xcode** » qui nous permettra d'écrire nos propres applications pour iPhone et iPad. Cet environnement de développement permet également le débogage, le test et plus généralement tout ce qui concerne la gestion du développement d'applications dédiées.

Il est important de rappeler que pour installer l'environnement de développement « Xcode », il est impératif de disposer d'un ordinateur Macintosh à processeur Intel, sinon vous ne pourrez pas configurer le SDK.

Installation de « Xcode »

Pour installer « Xcode », vous devez télécharger sur votre Mac l'image du disque « **.dmg** » directement à partir du site Apple pour développeurs. La figure 1 montre la page web dédiée au système iOS, où vous pouvez télécharger le SDK.

L'adresse de ce site est : **http://developer.apple.com**. Il vous faudra un compte Apple, sinon vous pouvez en créer un en vous enregistrant.

Les outils de développement pour lesquels l'enregistrement est obligatoire, sont réservés uniquement aux clients titulaires d'un compte valide.

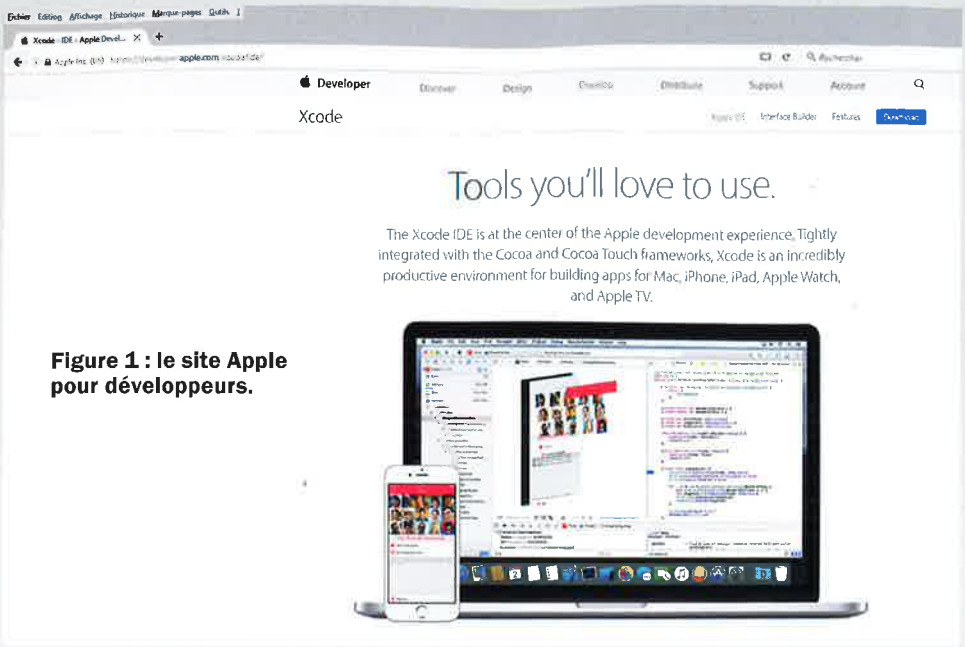


Figure 1 : le site Apple pour développeurs.

L'achat de la licence de développement n'est pas obligatoire (du moins pour l'instant) pour tester nos projets sur les périphériques. Le prix est d'environ 100 €. Les choses semblent être en train de changer au regard des modifications qui viennent d'être faites dans le programme de développeurs.

Désormais, sans acheter de compte développeur payant, Apple autorise la création, la compilation et le déploiement d'une application iOS sur vos appareils !

Cependant si vous souhaitez proposer votre application sur l'App Store, un compte développeur payant sera obligatoire. Si vous voulez simplement apprendre « Xcode » et tester vos applications directement sur votre appareil et non sur le simulateur, cela est maintenant possible.

En résum, si vous voulez utiliser le simulateur et/ou proposer votre application sur l'App Store il faudra payer !

Comme vous pouvez le voir en figure 3, dans la section dédiée aux développeurs, vous pouvez lire la documentation (en Anglais), télécharger les versions de développement d'iOS et bien plus encore. Nous vous proposons de parcourir le site des développeurs pour en apprendre davantage.

L'installation du package « Xcode » ne nécessite pas de configuration « ad-hoc », vous pouvez la faire sans aucun problème, tout simplement en effectuant un double clic sur le fichier et en suivant les étapes d'installation.

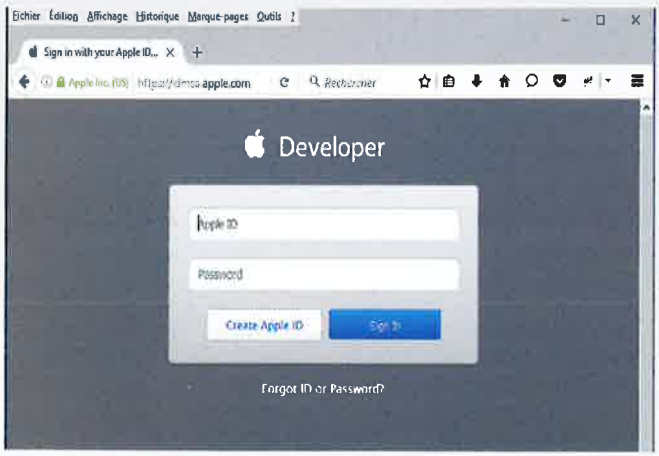


Figure 2 : la fenêtre de « login » (connexion).

Une fois votre enregistrement effectué, vous pouvez accéder à la zone réservée à partir de laquelle vous pouvez télécharger « Xcode » qui est l'outil indispensable pour développer sous iOS.

Pour valider l'enregistrement, il suffit de cliquer sur le lien en haut à droite, visible en figure 1, et identifié par le label « **Member Center** ».

Ensuite, le système accède à la section de « **Login** » (connexion) dédiée au développement (comme vous pouvez le voir en figure 2).

Dans le cas où vous n'êtes pas titulaire d'un compte, vous pouvez en créer un en cliquant sur le lien à droite intitulé « **Need to register ? Join now** ». Ce qui veut dire « Besoin de s'enregistrer ? Rejoignez-nous maintenant ».

Dès que vous cliquez sur ce lien vous ouvrez une autre page où vous pouvez commencer votre enregistrement.

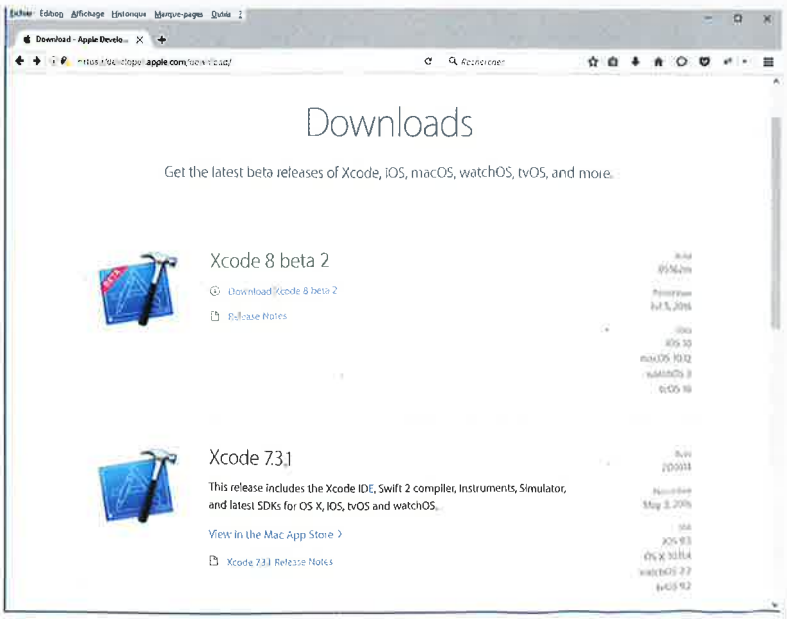


Figure 3 : le site consacré aux développeurs, à partir de cette section vous pouvez télécharger « Xcode ».

Figure 4 : recherche de l'application « Xcode » après l'installation, vous devez trouver une icône représentant un « marteau ».

Ce guide permet de découvrir les astuces contenues dans l'outil de développement.

Le 3ème bouton est en fait un lien hypertexte qui permet d'aller sur le site de référence pour les développeurs (http://developer.apple.com).

Dans la section de droite de la fenêtre d'accueil de « Xcode », vous trouverez la liste de vos dernières applications.

Démarrer « Xcode »

Une fois l'installation terminée, l'icône de « Xcode » ne figure pas parmi les applications, mais se trouve à un autre endroit du système de fichiers.

C'est pour cette raison que nous devons le localiser (le trouver) grâce au « Finder » (l'équivalent de la recherche sous Windows) comme le montre la figure 4.

Le résultat de la recherche nous indique que l'application « **xcode.app** » a été trouvée. Faites un double clic sur l'icône représentant un marteau pour lancer l'outil de développement « Xcode », vous êtes maintenant en mesure de commencer le développement d'applications dans le monde de l'iPhone.

En figure 5, vous pouvez voir la fenêtre d'accueil de l'outil de développement « Xcode », sur la gauche se trouvent 3 boutons. Le premier, nommé « **Create a new Xcode project** », permet de créer un nouveau projet.

Le second appelé « **Getting started with Xcode** » est un guide de démarrage et d'utilisation pour « Xcode », c'est une ressource très intéressante pour les nouveaux utilisateurs.

Figure 5 : fenêtre d'accueil de « Xcode ».

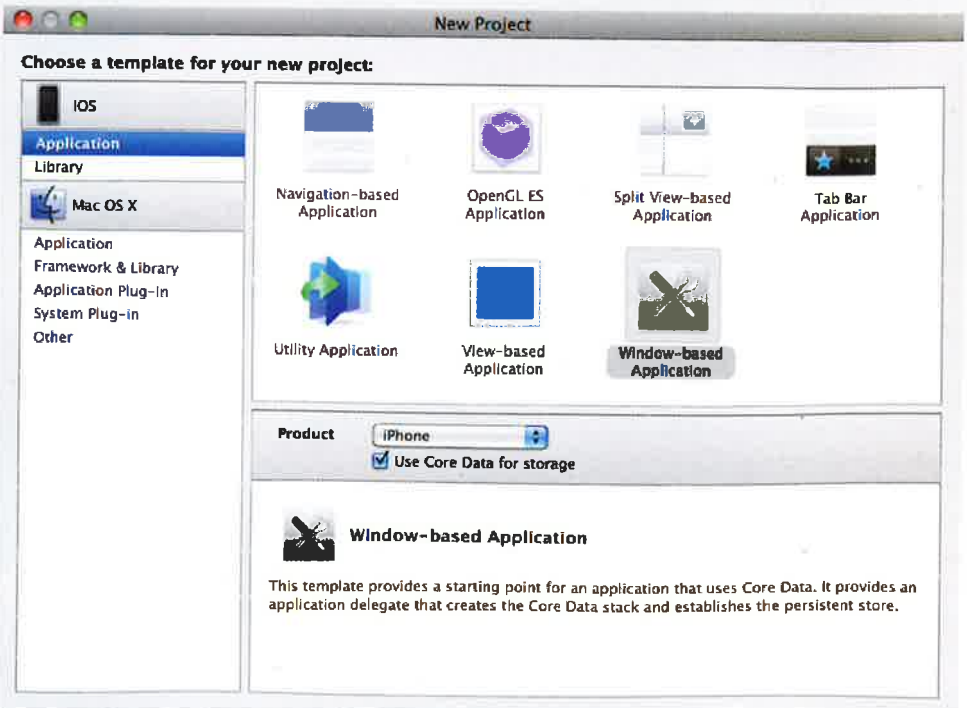


Figure 6 : navigation dans les différentes ressources disponibles sous « Xcode ».

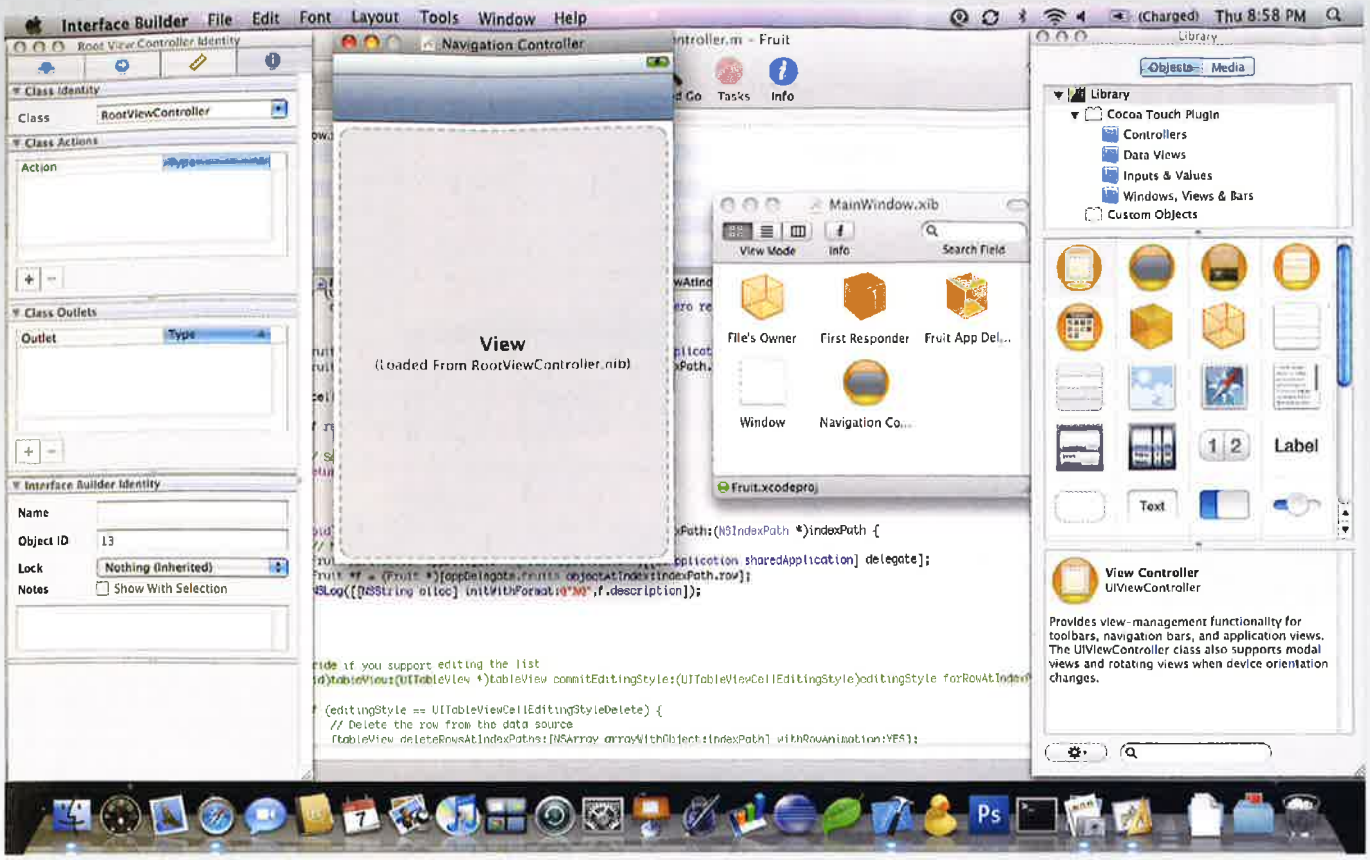


Figure 7 : Ici « L'interface Builder », c'est un outil de développement d'interface graphique. Il fait partie de l'environnement de développement « Xcode ».

Lors du 1ère lancement de l'application « Xcode » (après l'installation), la liste sera vide.

En bas à gauche de la fenêtre d'accueil (figure 5), vous trouverez le bouton « Open Other... » utilisé pour ouvrir un projet qui ne figure pas dans la liste de droite, bien souvent il s'agit d'un projet qui n'a pas été utilisé récemment.

Le bouton qui nous intéresse le plus est certainement le premier « **Create a new Xcode project** », car nous allons maintenant créer un nouveau projet. Cliquons sur le bouton, une seconde fenêtre s'ouvre où nous pouvons choisir en détail quel type d'application nous voulons réaliser.

En figure 6, vous pouvez voir la fenêtre qui permet de réaliser une application pour iPhone. Sur la gauche sont représentées deux sections, la première se rapporte au développement d'applications pour iPhone sous iOS, tandis que la seconde, dénommée « Mac OS X », est dédiée au développement d'applications pour les ordinateurs Mac.

Dans ce cours, nous traitons uniquement de la première partie, c'est-à-dire le développement d'application pour iPhone. Nous n'étudierons pas le développement d'applications pour ordinateurs Mac, et nous laissons de côté pour le moment le développement de librairies.

En figure 6, vous apercevez un grand rectangle dans la partie droite supérieure où sont représentés 7 modèles d'applications (App) disponibles pour le développement de vos projets.

Figure 8 : le simulateur d'iPhone.

En bas, une description s'affiche en fonction du modèle sélectionné.

Tous ces modèles sont disponibles pour l'iPhone et l'iPad, à l'exception de quelques-uns qui sont réservés à un seul des deux périphériques.

Par exemple « Navigation Based Application » est uniquement dédié à l'iPhone, « Split View-Based Application » est seulement réservé à l'iPad, « Utility Application » uniquement pour l'iPhone.

En règle générale, lorsque vous avez une bonne connaissance du système, il est presque exclusivement consacré à « Window-based Application » c'est-à-dire le développement d'une application à base de fenêtres. C'est le seul modèle qui permet d'obtenir une « mise en page » (template) universelle de l'application aussi bien pour l'iPhone que pour l'iPad.

« Xcode » n'est pas le seul outil de développement pour applications sous les systèmes Apple, un autre outil de développement existe, nous l'utiliserons pour la création de notre application. Il s'agit de « **Interface Builder** » (voir la figure 7).

C'est un **outil de développement d'interface graphique** pour des applications tournant sur **MacOSX**, il fait partie de l'environnement de développement « Xcode ». « Interface Builder » permet de créer une interface graphique pour une application.

L'interface graphique ainsi générée par « Interface Builder » est contenue dans un fichier de type « **.nib** » (NeXT Interface Builder) et plus récemment dans un fichier dont l'extension est « **.xib** ».

« Interface Builder » fournit des composants d'interface graphique qui sont des palettes d'objets tels que des champs de texte, des tableaux, et des menus. Il est possible de créer de nouveaux objets et de les ajouter à « Interface Builder ».

Pour créer une interface, il suffit de faire glisser un objet de l'interface depuis la palette vers une fenêtre ou un menu. « Interface Builder » sauvegarde l'interface d'une application dans un fichier qui contient les objets de l'interface et les relations avec ses objets utilisées dans l'application. Ces objets sont archivés dans un fichier de type « .nib ».

Le dernier programme à notre disposition est le **simulateur**, représenté en figure 8. Cet outil de développement permet la **simulation d'applications sans nécessairement devoir les installer sur un périphérique physique.** Nous pouvons simuler une application sur différentes versions d'iPhones (systèmes d'exploitations différents). Il en est de même pour l'iPad, avec ses différentes versions.

Pour chaque périphérique, nous pouvons simuler différentes conditions de travail, comme un appel entrant ou un problème de mémoire, faire pivoter le périphérique, arrêter et redémarrer l'application, provoquer une secousse et bien d'autres choses encore.

Rappelez-vous, cependant, qu'il s'agit d'un simulateur et non d'un émulateur, de nombreuses caractéristiques sont différentes par rapport au périphérique physique, en particulier le temps d'exécution. Prenons l'exemple d'une connexion Internet, si vous utilisez une connexion GSM, la vitesse de téléchargement et l'affichage des pages web est beaucoup plus élevée en particulier sur les anciens modèles tels que l'iPhone 3G.

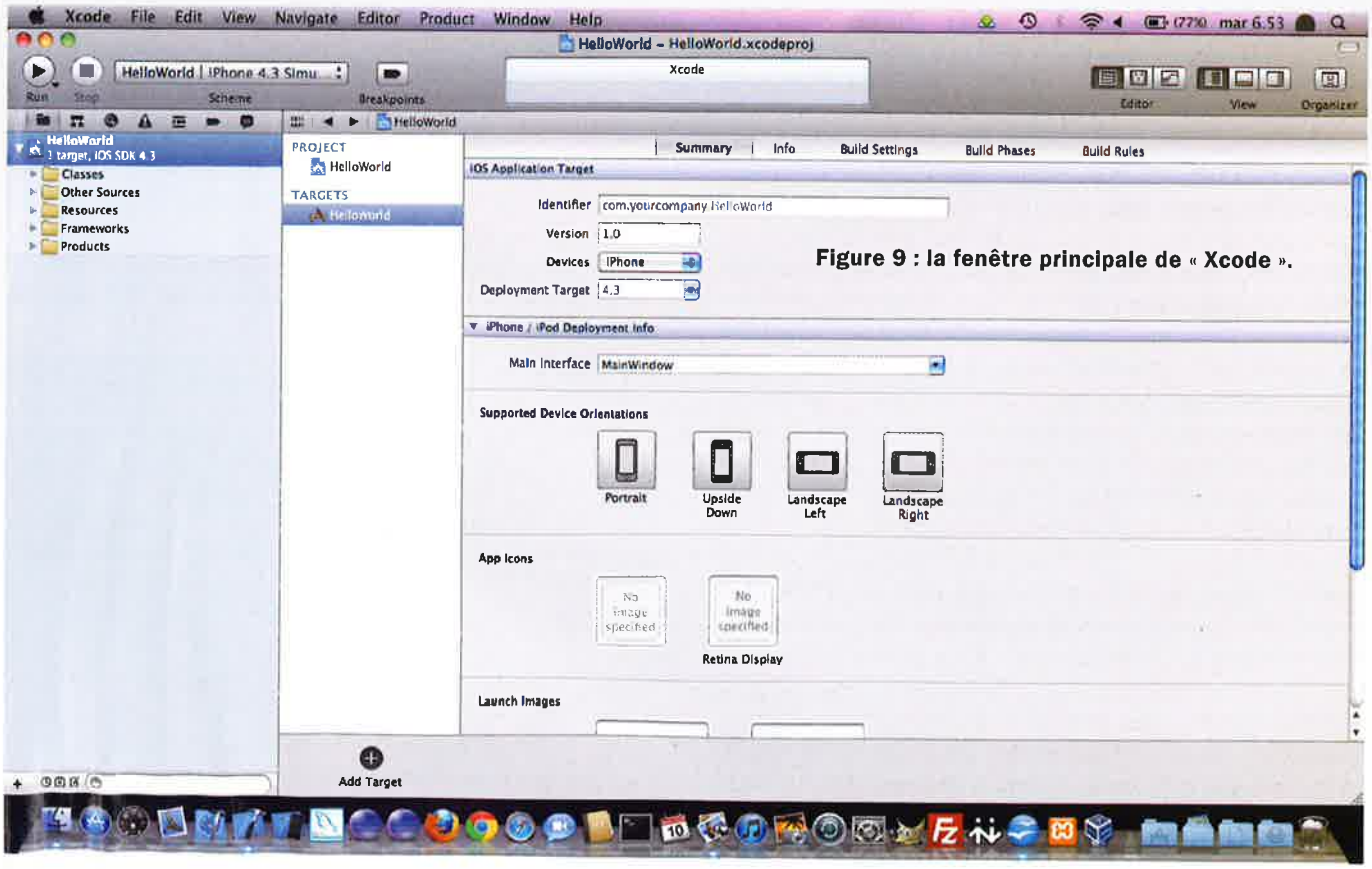


Figure 9 : la fenêtre principale de « Xcode ».

Création d'un projet avec « Xcode »

Dans la fenêtre de sélection des modèles « Xcode », que nous avons déjà vu en figure 6, sélectionnez l'icône « **View-based Application** ». Nommez le nouveau projet, par exemple nous commencerons avec le classique « Hello World » (sinon le nom que vous souhaitez), et continuez. Une nouvelle fenêtre s'ouvre alors et normalement elle doit être semblable à celle représentée en figure 9.

La fenêtre principale de « Xcode » est divisée en plusieurs parties : une pour l'écriture du code source et 3 autres pour l'intégration de l'environnement « Interface Builder », nécessaire pour le développement graphique de l'application.

Notez que selon la version de « Xcode » dont vous disposez, il se peut que la fenêtre principale soit légèrement différente de celle de la figure 9, néanmoins les principes de bases restent identiques.

Dans la partie gauche de l'environnement de développement, vous apercevez le gestionnaire de projet ou « **Project Manager** », dans la partie centrale de la fenêtre se trouve l'éditeur avec lequel vous écrirez votre code source.

Mais avant d'entrer dans le monde de la programmation pour iPhone, nous avons besoin de faire une parenthèse sur le langage ObjectiveC, afin de comprendre les différents concepts présentés dans la première leçon en utilisant la syntaxe exacte.

Le langage ObjectiveC

Le langage ObjectiveC, comme tous les langages de programmation, a besoin d'un « **point d'entrée** » (« entry-point »). C'est l'endroit du code source **où un programme commence**, c'est-à-dire où notre application va commencer l'exécution de son code.

Dans le cas de l'ObjectiveC, ce point d'entrée se trouve dans le fichier appelé « **main.m** », qui se trouve dans le dossier « **Other Sources** » à l'intérieur du dossier « ressources » du projet.

Une première différence essentielle avec la langage « C » est l'extension du fichier. L'extension « **.m** » (main.m) **indique au compilateur que le code est de type ObjectiveC** et doit donc être traité en tant que tel. Ce fichier nous montre immédiatement un premier aperçu du langage ObjectiveC.

Avant d'analyser le code et de commencer à comprendre les fondements, analysons la structure du fichier représentée dans le listing 1.

Il est important de souligner, pour ObjectiveC comme pour le langage « C », que la **déclaration explicite** des « **prototypes** » est absolument nécessaire, c'est-à-dire les **signatures des méthodes** pour tout ce que vous voulez utiliser dans votre code.

Pour rappel, en programmation orientée objet, une méthode est une fonction membre d'une classe, la signature définit les types de données acceptables pour une fonction ou une méthode.

Pour cela, **nous devons créer deux fichiers** pour **chaque ressource** à ajouter. Un fichier **en-tête** appelé « **header** » dont l'extension est de type « **.h** », et le second qui implémente les déclarations faites dans le fichier « header ». Ce 2ème fichier porte l'extension « **.c** » pour le langage « C » et « **.m** » pour le langage ObjectiveC.

Le fichier « header » (ou aussi appelé fichier « interface ») est utilisé pour définir toutes les méthodes et propriétés qui seront utilisées dans votre programme. Le fichier d'implémentation est le fichier dans lequel vous allez écrire votre code qui exécutera la description de votre fichier header.

Revenons dans le listing 1, la première ligne implémente la première directive du préprocesseur. La directive « **import** » permet d'inclure un autre fichier, ici il porte l'extension « **.h** ».

Listing 1 : Introduction aux fichiers ObjectiveC.

```
#import <UIKit/UIKit.h>

int main(int argc, char *argv[]) {
        ... // code objective
}
```

Notez que « Xcode » vous proposera de créer automatiquement le fichier « header » lorsque vous créerez le fichier d'implémentation.

La directive « import » de l'ObjectiveC est considérablement plus efficace que celle de son homologue en « C », à savoir la directive « include » qui sert à importer dans les fichiers les définitions courantes.

En fait, dans le langage « C », afin d'éviter le problème des définitions multiples, il est nécessaire d'introduire des directives de préprocesseur appropriées qui influent sur le processus de compilation, cela évite les erreurs ou les omissions du développeur.

Le listing 2 montre un exemple de fichier « header » en « C ». Les directives « **ifndef** » et « **endif** » sont de type « compilation conditionnelle ». Dans la pratique, le fichier exécute des directives de contrôle sur une constante, qui dans notre cas est appelée « EXEMPLE_H_ _ _ _ ».

Si elle n'existe pas, elle est créée par la directive « **define** » et son contenu est également testé. Avec cette méthode, même si nous incluons plusieurs fois ce fichier « .h », cela ne produira pas de problèmes de compilation, car après la première compilation le fichier sera ignoré car le compilateur l'aura déjà testé.

En ObjectiveC, il n'est pas nécessaire d'utiliser ce type de syntaxe, car nous utilisons la directive « import » qui résout le problème de l'inclusion multiple.

Revenons maintenant au code du listing 1, la **deuxième ligne correspond au point d'entrée** (entry-point) typique du langage « C », avec deux paramètres pour obtenir des informations supplémentaires qui pourraient arriver de la console pour lancer le programme.

Listing 2 : Exemple de déclaration d'un fichier header C.

```
//exemple.h
#ifndef __ EXEMPLE_H __
#define __ EXEMPLE_H __

        void salutation(void);
        int testFunction(char *string);

#endif
```

Listing 3 : corps de la fonction main en C.

```
NSAutoreleasePool *pool = [[NSAutoreleasePool alloc] init];
int retVal = UIApplication(argc, argv, nil, nil);
[pool release];
return retVal;
```

Pour ceux d'entre vous qui n'avez jamais vu la structure de la ligne 2, la variable « **argc** » contient le nombre de paramètres transmis à notre application, tandis que la variable « **argv** » est un **pointeur** de chaînes de caractères contenant les chaînes qui sont passées de la console vers l'application.

Voici un exemple pratique lorsque différents paramètres sont passés à un programme.

Les paramètres sont : « bonjour-nom électronique-langage anglais », où « bonjour » est le nom du programme, la variable « argc » aura une valeur de 5, tandis que la variable « argv » sera constituée par un tableau de 5 éléments représentant chacun une chaîne : « bonjour », « -nom », « électronique », « -langage » et « anglais ».

Analysons maintenant le listing 3, qui intègre le listing 1 avec en supplément la partie manquante du code de la fonction « **main()** ». Comme vous pouvez le voir dans la première ligne du listing 3, nous avons une variable pointeur de type « **NSAutoreleasePool** ».

Le fait que la variable soit un pointeur implique la présence de l'astérisque devant le nom de la variable « **pool** », exactement comme en « C ». Avec l'instruction « **NSAutoreleasePool *pool** » nous déclarons une variable qui est de type « **NSAutoreleasePool** ».

Un autre exemple pourrait être « **NSString *chaine** », qui identifie une variable dont le nom est « chaine » et qui est de type « **NSString** ».

L'instruction suivante est « **[[NSAutoreleasePool alloc] init];** », cette expression peut être décomposée en deux parties, la première « [NSAutoreleasePool alloc] » et la seconde « [pool init] ». Les crochets en ObjectiveC sont le moyen syntaxique d'invoquer des méthodes sur des objets. Ce qu'il y a entre crochets est le contenu du message, séparé en deux parties : à gauche, le **receveur est l'objet** qui va recevoir le message et **exécuter la méthode** appelée ; à droite, c'est la **méthode elle-même**.

Le type « **NSAutoreleasePool** » est donc notre premier **objet** dans ObjectiveC. Nous ne l'avons pas écrit, mais nous savons maintenant que c'est un objet.

La méthode « **alloc** » n'est pas vraiment un constructeur (comme nous l'avons mentionné dans la première leçon), car elle ne sert qu'à allouer de la mémoire pour un certain type d'objet. La méthode « alloc » est un proche parent de la méthode « malloc » du langage « C ».

Avec cette procédure, nous avons créé physiquement l'objet, mais nous ne l'avons pas encore initialisé (opération qu'un

Listing 4 : Exemple de réservation et de libération successive de la mémoire.

```
NSObject *chaine = [[NSObject alloc] init];
// À ce stade, le compteur de référence est à 1,
// cela est géré par la création de l'objet.
[chaine retain];
// Le compteur de référence (Reference Counter) est maintenant à 2
[chaine retain];
//reference counter = 3
[chaine release];
//reference counter = 2
// l'objet ne disparaît pas de la mémoire, il existe encore 2 références.
[chaine release];
//reference counter = 1
[chaine release];
// Le compteur de référence vaut 0, il est supprimé de la mémoire.
```

constructeur d'un langage OOP ferait automatiquement). Pour l'initialiser correctement, utilisons la méthode « **init** ».

En deuxième ligne, vous pouvez voir le code suivant en langage « C » :

```
(int retVal = UIApplication(argc, argv, nil, nil));
```

Grâce à cette ligne de code, nous appelons une fonction « **UIApplication(...)** » avec les paramètres nécessaires, et sa valeur de retour est de type entier en attribuant cette valeur à la variable « **retVal** ».

Cette fonction est définie à l'intérieur du **framework** « **Foundation** » de « **Cocoa Touch** » et est utilisée pour initialiser et lancer l'application. « Cocoa Touch » est une API native d'Apple pour le développement orienté objet spécifique à l'iPhone.

Par contre, à la 3ème ligne, nous appelons une méthode sur un objet, elle se trouve entre crochets et l'objet est « NSAutoreleasePool » que nous avons déclaré auparavant.

La méthode « **release** », qui est parente de la fonction « free » du « C », permet d'**effacer la mémoire de travail** (nous étudierons plus en détail la gestion de la mémoire dans les prochaines leçons).

La dernière ligne du listing 3 correspond au code de sortie de notre application, elle permet de revenir au système d'exploitation de la même manière que pour le langage « C ».

Gestion de la mémoire de l'iPhone

Dans la première leçon de ce cours, nous avons déjà évoqué le fait que l'iPhone nécessitait une gestion de la mémoire de la part du programmeur. Cela se fait sur tous les objets à travers la méthode « release », qui est notre pseudo-destructeur.

En « C », lorsque nous appelons la fonction « free » sur un type de données, elles sont immédiatement effacées de la mémoire. Toute autre pointeur qui conserve la référence de cette partie libérée de la mémoire, lors d'une tentative d'une opération de « **read** » (lecture) ou « **write** » (écriture) va « planter » l'application, car le pointeur se réfère à une partie de la mémoire qui n'appartient plus à l'application.

En ObjectiveC, ce problème a été résolu. Lorsque nous invoquons la méthode « **release** » sur un objet, **il n'est pas immédiatement retiré** de la mémoire, à la place un **compteur de référence** appelé « **reference counter** » est **décrémenté**. Lorsque ce compteur arrive à 0, alors l'objet est effectivement effacé de la mémoire.

Cette méthode nous donne plus de sécurité, à condition de toujours libérer les références des objets lorsque ceux-ci ne sont plus utilisés. Car sinon les références restent toujours en mémoire et provoquent un problème appelé « **memory leakage** » que nous pouvons traduire par « **fuite de mémoire** ».

Pour comprendre ce concept, prenons un exemple concret. En ObjectiveC, pour augmenter le nombre de références nous devons utiliser la méthode « **retain** » tout en décrémentant le compteur avec la méthode « **release** ».

Pour nos exemples, nous n'avons pas vraiment besoin de la méthode « **retain** », car il existe des procédés qui font appel à cette méthode, alors que nous allons largement utiliser la méthode « **release** ».

Dans le listing 4, nous utilisons un objet nommé « **NSObject** » pour deux raisons. La première est que pour le moment nous n'avons pas écrit notre premier objet, la seconde est que nous voulons soulever une question concernant l'espace occupé par les noms des applications.

Le problème qui est présenté est très bien connu et se résout avec les « **namespaces** » ou l'espace des noms, de manière à éviter la création de classes ou de fonctions ayant le même nom.

Cela rendrait impossible la compilation, car l'ObjectiveC n'admet pas d'« **overload** » (surcharge) pour les méthodes et le langage « C » n'admet pas de surcharge pour les fonctions.

En outre, il ne peut pas y avoir des classes avec des noms identiques, car le compilateur ne saurait pas laquelle utiliser.

Pour résoudre ce problème en Objective C, ainsi que dans les langages ne disposant pas de système de type « **namespaces** », on **introduit un préfixe avant le nom** comme préfixe de la classe.

Cette solution est utilisée par Apple, en effet dans le nom « NSAutoreleasePool » le préfixe « **NS**» a été choisi de la manière suivante : en fait, il signifie « NextStep ».

Cela correspond au fait que Steve Jobs quitte Apple en 1985, et fonde la société « Next » qui développe un système d'exploitation appelé « NextStep ».

Ce système est devenu plus tard la base du système d'exploitation actuel « MacOS X » d'Apple.

Nous vous conseillons d'utiliser votre propre espace de noms, afin d'éviter des problèmes avec les codes d'Apple ou de tiers. Dans ce cours, nous allons utiliser le préfixe « **EI** » comme contraction du mot « électronique ».

Listing 5 : instructions fondamentales « If-else ».

```
if (1 == 1) {
        printf("égaux");
} else {
        printf("différents ");
}
```

Listing 6 : instruction fondamentale « while ».

```
while(i < 5)
{
        printf("%d", i);
        ++i;
}
```

Commençons à travailler en Objective C

Pour pouvoir écrire nos premières lignes de code en Objective C, nous avons encore besoin d'étudier les instructions de base du langage, qui sont essentiellement issues du langage « C », ainsi que les structures de données de base qui sont fournies directement par Apple.

Nous allons commencer par les fondements de base de l'Objective C, pour ensuite parvenir à une véritable application. Les instructions de base « **if** », « **for** », « **while** » sont exactement celles du langage « C », avec en plus **la possibilité de déclarer des variables directement dans les cycles.**

Listing 7 : instruction fondamentale « for ».

```
for(int i=0; i<10; i++)
{
        printf("%d", i);
}
```

Une des particularités de l'ObjectiveC est qu'il n'utilise pas le « NULL » mais plutôt le « **nil** ».

Dans le listing 5, vous pouvez voir l'instruction « **if** » qui est identique à celle du « C ». Dans cet exemple, l'égalité des expressions est vérifiée. Les instructions « **while** » et « **do/while** » sont également identiques à celles du « C ». Le listing 6 montre un exemple d'utilisation de l'instruction « **while** ».

L'instruction « **for** » dont un exemple d'utilisation est visible dans le listing 7, qui en plus d'être équivalente à celle du « C » offre également des solutions de gestion de flux. Analysons cette dernière possibilité, mais vous pouvez toujours utiliser la méthode standard du « C ».

Dans cet exemple, la **variable « i » a été déclarée à l'intérieur de la boucle « for »**. Prêtez attention à cette déclaration, elle ne fait pas partie de la norme « ANSI C », mais il est possible de l'utiliser dans le langage ObjectiveC sans poser de problème particulier.

Comme vous le savez peut-être pour ceux d'entre vous qui ont programmé en « C », les chaînes de caractères n'existent pas, en fait ce sont des vecteurs de caractères.

Ceci est également vrai pour le langage ObjectiveC, mais Apple a mis à disposition la classe « **NSString** » qui simplifie grandement la programmation.

Voyons un exemple d'une chaîne constante :

```
NSString *chaine = @"je suis un objet";
```

Notons que, contrairement au langage « C », le symbole **arobase « @ »** devant les guillemets sert à **indiquer une référence à un objet**. En fait, nous déclarons un **pointeur** d'objet « **NSString** ».

Listing 8 : exemple complet en ObjectiveC.

```
for (int i=0; i<[liste count]; i++) {
        NSLog(
                @"Index: %d - Valeur: %@",
                i,
                [liste objectAtIndex: i]
        );
}
```

Lorsque nous utilisons la classe « **NSString** », nous devons faire attention car il s'agit d'une classe immuable. Le terme « immuable » indique que **cette classe ne peut pas être modifiée** lorsqu'elle est synthétisée comme un vecteur de caractères, il n'est plus possible d'augmenter ou de réduire l'espace une fois que le vecteur a été alloué.

C'est pour cette raison que si nous travaillons avec des chaînes muables, nous devons plutôt utiliser la classe « **NSMutableString** ». Celle-ci permet de modifier la structure de la chaîne et donc de réduire les problèmes de mémoire, car les chaînes ne sont plus continuellement réallouées.

Même pour les tableaux (array), il existe des classes déjà prêtes à l'emploi (NSArray, NSMutableArray, NSSet, NSMutableSet) qui permettent d'enregistrer à l'intérieur nos objets.

Dans le listing 8, vous pouvez voir notre premier exemple de programme complet en ObjectiveC. Tout d'abord, nous avons créé une chaîne qui est « **NSString *chaine = @"je suis un objet";** ». Ensuite nous créons un ensemble d'éléments appelé « liste ».

Comme nous l'avons déjà expliqué, nous avons besoin d'allouer de l'espace de type tableau « **NSArray** » dans la mémoire à travers la méthode de classe « **[NSArray alloc]** », puis nous appelons notre constructeur « **initWithObjects** ».

Ce constructeur nécessite une **liste d'objets séparés par des virgules**. Le dernier objet de cette liste doit être « **nil** ». Par contre un « NSArray » ne peut pas contenir des variables de type int, float, etc.

Après avoir alloué le tableau, ou plutôt un objet « **NSArray** » contenant le tableau réel, nous allons étudier les différents éléments de la boucle constituée par le constructeur « **for** ».

Pour rappel, en programmation orientée objet un constructeur est une fonction particulière appelée lors de l'instanciation.

Listing 9 : premier exemple concret sur Iphone.

```
#import <UIKit/UIKit.h>
#import "helloworldViewController.h"

int main(int argc, char *argv[]) {

        NSString *chaine = @"je suis un objet";
        NSArray *liste = [[NSArray alloc] initWithObjects:
chaine, nil];

        for (   int i=0; i<[liste count]; i++) {
                NSLog(@"Index: %d - Valeur: %@",
                        i,
                        [liste objectAtIndex: i]
                );
        }

  NSAutoreleasePool * pool = [[NSAutoreleasePool alloc]
init];
   int retVal = UIApplicationMain(argc, argv, nil, nil);
   [pool release];
   return retVal;
}
```

Elle permet d'allouer la mémoire nécessaire à l'objet et d'initialiser ses attributs.

Dans cette déclaration, nous appelons la méthode « **count** » sur l'objet « liste » qui renvoie le nombre d'éléments enregistrés. Dans ce cas, il y a un seul élément.

Pour chaque élément de la liste nous utilisons la fonction « **NSLog** », celle-ci écrit dans la console de l'iPhone notre message contenant l'index et la valeur de l'élément stocké dans la position courante de la liste.

La structure de « **NSLog** » est la suivante : « **NSLog(NSString *, ...)** » où les points indiquent un nombre variable de paramètres.

La première partie doit être une chaîne qui doit **commencer** par le caractère « **@** », car c'est un **pointeur**. Ensuite après l'arobase vient un nombre variable de paramètres, tels que décrits précédemment.

Le contenu du premier paramètre est le suivant : « **Index: %d - Valeur: %@** ». Le premier type que nous voulons représenter est un **nombre décimal normal** (**% d**), tandis que le second (**% @**) représente un **objet** comme si c'était une chaîne.

Maintenant, prenons notre projet que nous avons nommé « HelloWorld » et que nous utiliserons encore dans la prochaine leçon. Ouvrons le fichier « **main.m** » situé dans le dossier « **Other Sources** » et écrivons à l'intérieur la chaîne suivante : « **NSString *chaine = @"je suis un objet"** ».

Le fichier « **main.m** » doit ressembler à celui du listing 9. Pour exécuter ce code, nous devons cliquer sur le bouton « Run » dans la partie supérieure gauche de la barre de contrôle de l'IDE, comme le montre la figure 10.

Cette opération effectue la compilation et démarre la console dans laquelle nous pouvons visualiser les messages provenant de la fonction « **NSLog** ». Le résultat est représenté en figure 11.

Après avoir visualisé le message, nous pouvons fermer le simulateur en appuyant sur les touches « **POMME + Q** ».

Nous arrivons au terme de cette deuxième leçon. Dans le prochain numéro, nous étudierons le concept de « Framework MVC » (Model View Controller ou Modèle Vue Contrôleur) qui très utilisé dans le développement d'applications pour iPhone. ■

Figure 10 : la barre de contrôle de « Xcode ».

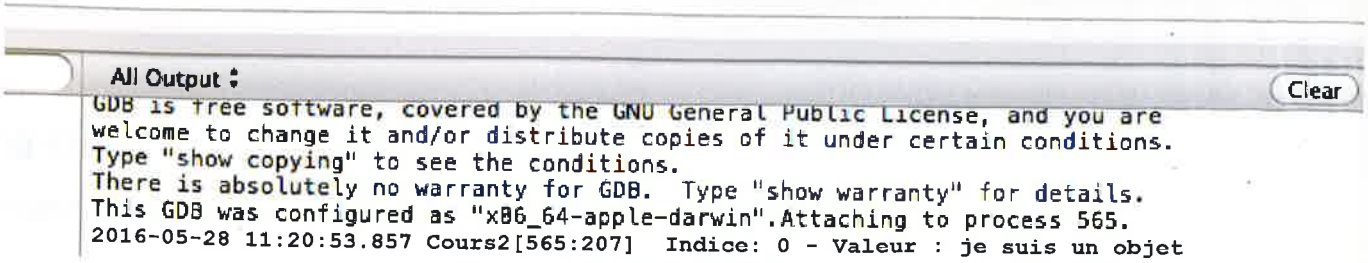


Figure 11 : la console « Xcode » après que nous ayons lancé notre application.