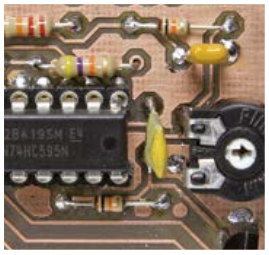
édition spéciale – numéro double

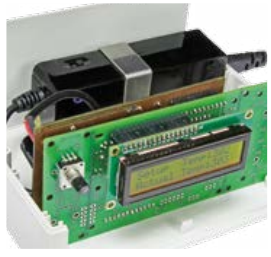
juillet/août 2015 | nº 445/446
www.elektormagazine.fr

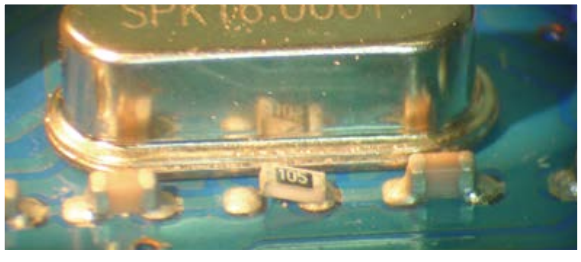
elektor

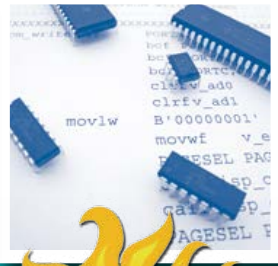
DÉCOUVRIR • CRÉER • PARTAGER

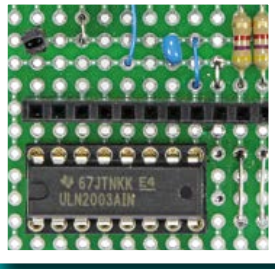


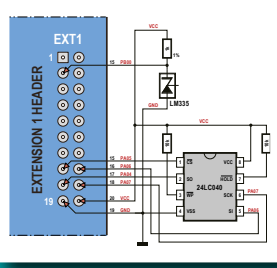
132 pages :
11 projets du labo
6 projets de lecteurs
4 cours de programmation
nouvelle carte de prototypage
nouvelle carte T-board
et bien plus !









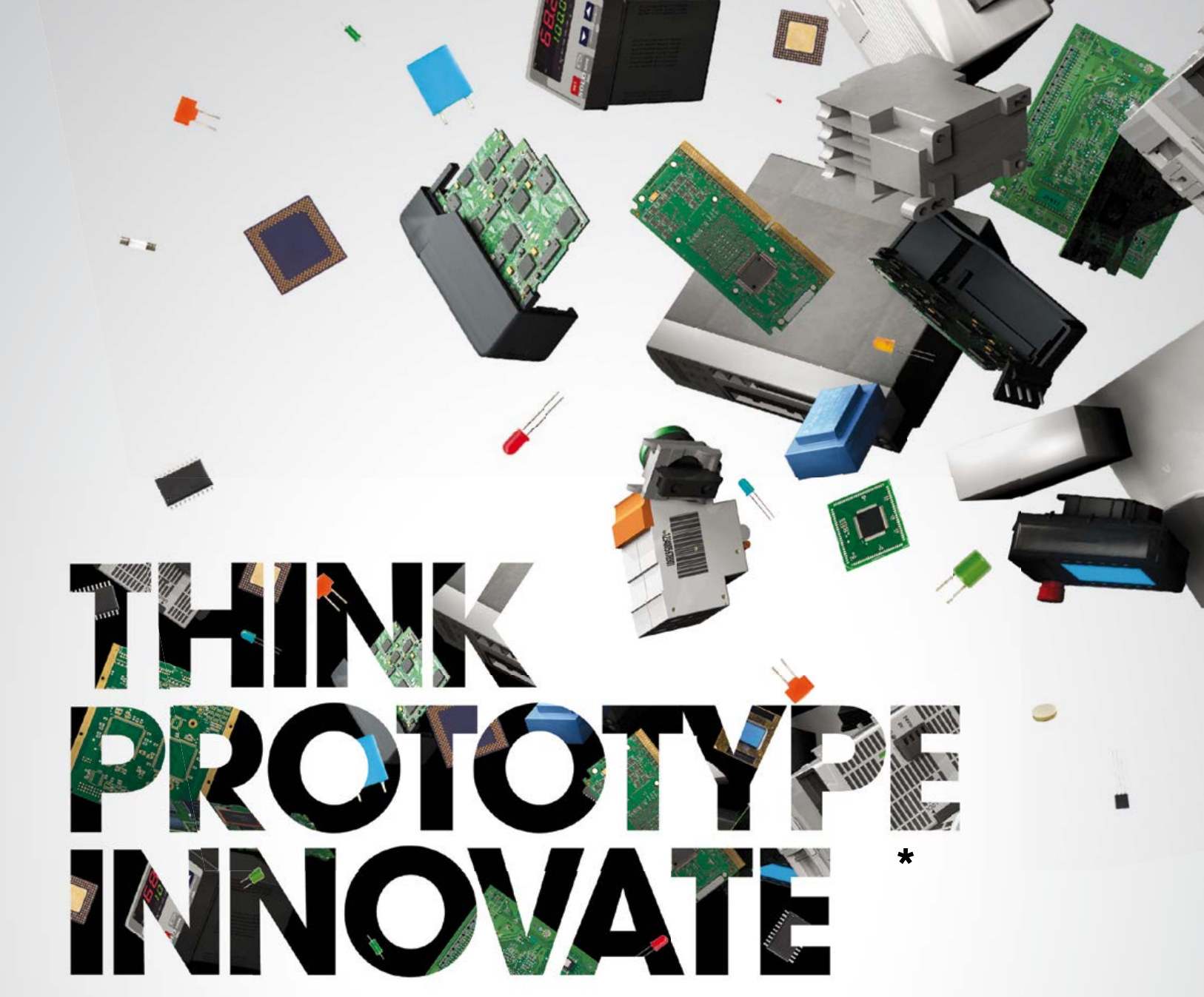


applis *Windows Store* • cours intensif d'assembleur (1) • **µC ARM pour néophytes (SPI)** • synthèse numérique directe (DDS) • **Arduino en esclave I²C** • interface LCD mono-fil • **sonde différentielle active 2 GHz** • **carte de prototypage ELPB-NG** • **sonnette musicale sans fil programmable** • convertisseur de tension de panneau solaire • **e-BoB BL600 (5)** • sirène de police • **ampli audio T-board à haut-parleur intégré** • **analyseur MIDI pour Arduino et Cie** • **vobulateur à Raspberry Pi** • poste de soudage de CMS avec Platino • émulateur de jeux vidéo Chip-8 • **wattmètre avec e-BoB ADS1115** • capteur I²C sans fil

ISSN 0181-7450

(B) 16,50 € • (CH) 29,00 FS • (CAN) 20.99 $ca • (And) 15,50 €
DOM surface 16,50 € • DOM avion 16,90 € • N Cal/S 2000 cfp

L 19624 -445- F: 15,50 € -RD



DESIGNSPARK

Tout cela est rendu possible grâce à RS

Replacez l'innovation au centre de votre processus de conception

Notre suite d'outils pour le prototypage rapide a été conçue pour vous aider à passer du concept au prototype en un temps record.

Grâce à ces outils adaptés aux besoins des concepteurs de produits et ingénieurs en électronique, retrouvez le temps de vous consacrer à votre passion pour l'innovation et concevez ainsi les produits qui changeront le monde de demain.

Téléchargez gratuitement nos outils de prototypage rapide sur designspark.com

#RapidPrototyping

* PENSEZ | CONCEVEZ | INNOVEZ

38ème année, n° 445/446 – juillet/août 2015

ISSN 0181-7450
Dépôt légal : juin 2015
CPPAP 1113 U 83713
Directeur de la publication : Donatus Akkermans

Elektor est édité par :
PUBLITRONIC SARL
c/o Regus Roissy CDG
1, rue de la Haye
BP 12910
FR - 95731 Roissy CDG Cedex

@ : service@elektor.fr
Tél. : (+33) 01.49.19.26.19
du lundi au vendredi de 10h à 13h

Fax : (+33) 01.49.19.22.37

www.elektor.fr | www.elektormagazine.fr

Banque ABN AMRO : Paris
IBAN : FR76 1873 9000 0100 2007 9702 603
BIC : ABNAFRPP

Publicité :
Fabio Romagnoli +32 485 65 40 90
fabio.romagnoli@eimworld.com

Imprimé aux Pays-Bas
par Senefelder Misset – Doetinchem
Distribué en France par M.L.P.
et en Belgique par A.M.P.

il n'y a pas de sot métier

Comme vous sans doute, tout dans la *branloire pérenne* du monde m'intéresse. Souvent, hélas, nous ne trouvons ni le temps ni l'angle d'approche adéquat pour approfondir nos connaissances. Pourtant ma curiosité et ma capacité d'enthousiasme se régénèrent inlassablement au contact de ce qu'il advient, sous le soleil, de nouveau et de renouveau. Aussi bien en astronomie qu'en gastronomie, en musique qu'en médecine ou en météorologie, en archéologie ou en recherche spatiale, et ainsi de suite. Comme roule l'ivresse du changement, je ne me lasse pas de ce que notre cerveau nous permet (d'essayer) de comprendre, y compris d'ailleurs le cerveau lui-même.
Chaque parcelle de l'univers ne mérite-t-elle pas, à un degré ou un autre, notre attention soutenue ? Cependant, en matière de progrès surprenants, et surtout pour la vitesse à laquelle ils s'y succèdent, je constate que l'électronique décroche le pompon de l'innovation à marche forcée. Quelle dynamique, quel bourgeonnement !
Et quelle chance d'être aux premières loges avec Elektor ! Nous venons de boucler ce numéro double de l'été 2015 où nous proposons une électronique accessible au plus grand nombre, avec une incroyable variété de sujets, une gradation de tous les niveaux, du débutant à l'expert confirmé…
Le travail achevé, je lève les yeux sur les dernières nouvelles et, où que se porte mon regard, je suis submergé par des annonces sensationnelles. Au hasard de mes lectures de ce début juin 2015, je relève pêle-mêle, par exemple, l'utilisation désormais généralisée du fil de cuivre à la place de l'or pour les liaisons internes des puces ; ou encore la banalisation de l'Arduino devenu une espèce de 555-à-tout-faire ; ou l'annonce de la fabrication de circuits multicouches ultraminces à plus de cent (sic) couches. Une telle énumération pourrait s'étendre sur des pages, mais mon intention est d'en finir ici avec l'annonce du jour, celle qui me laisse bouche bée d'admiration : les *textiles tactiles* de Google et Lévis Strauss. Si vous n'avez pas encore vu de démonstration de ces tissus à trame électronique, jetez-y un coup d'oeil. Je souhaite qu'Elektor puisse y consacrer bientôt des articles. Je ne ferai pas la sottise de demander à quoi ça va servir, mais une chose est sûre : le noble métier de chiffonnier, presque tombé en désuétude, a désormais un bel avenir.

Bonne lecture!
Denis Meyer

Notre équipe

Rédacteur en chef :	Denis Meyer (redaction@elektor.fr)
Rédaction internationale :	Harry Baggen, Jan Buiting, Jaime González-Arintero, Jens Nickel
Laboratoire :	Thijs Beckers, Ton Giesberts, Luc Lemmens, Clemens Valens (responsable), Jan Visser
Coordination :	Hedwig Hennekens
Ont coopéré à ce numéro :	Robert Grignard, Hervé Moreau, Kévin Petit, Guy Raedersdorf, Mariline Thiebaut-Brodier, Thierry Destinobles, Jean-Louis Mehren
Service de la clientèle :	Cindy Tyssen
Graphistes :	Giel Dols, Mart Schroijen, Patrick Wielders
Elektor *en ligne* :	Daniëlle Mertens

ce mois-ci

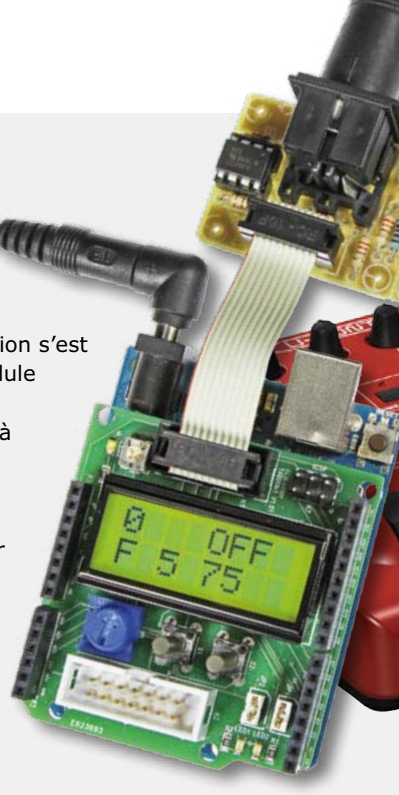
38e année – nº 445-446 juillet-août 2015

analyseur MIDI

module d'entrée/sortie pour Arduino et Cie

Le célèbre duo Arduino et *shield* d'extension s'est composé une suite musicale avec un module d'analyse d'entrée et sortie MIDI qui, par l'entremise du connecteur ECC, s'adapte à d'autres cartes à µC. Notre micrologiciel didactique décode les messages MIDI et les fait apparaître. Les modules logiciels utilisés offrent un album de variations sur le thème.

84



ampli audio T-board

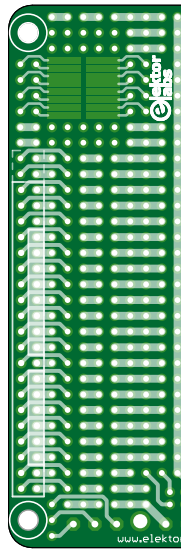
à haut-parleur intégré

Pour la mise au point de circuits analogiques sur une platine d'expérimentation, un petit amplificateur audio et un haut-parleur sont souvent utiles, afin de rendre les signaux audibles. Nous avons donc réalisé une carte *T-board*, qui reprend sur un seul circuit imprimé un amplificateur de classe D, un convertisseur CC/CC, et un haut-parleur.

80

DÉCOUVRIR | CRÉER | PARTAGER

DÉCOUVRIR | CRÉER | PARTAGER

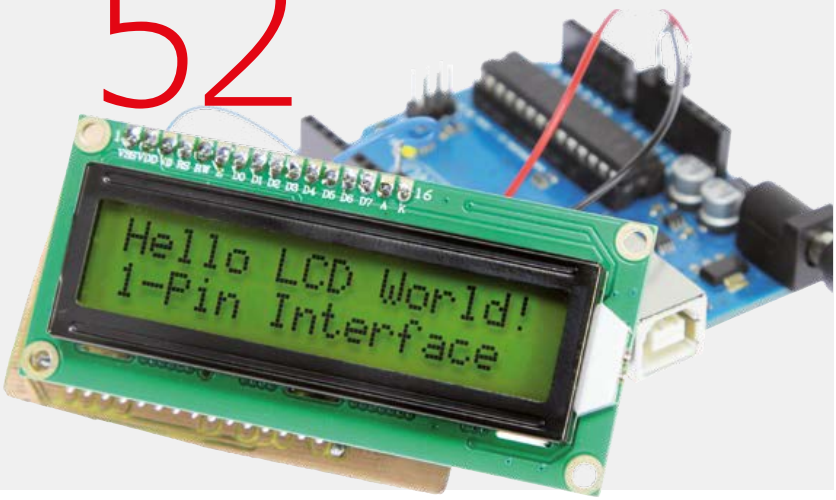


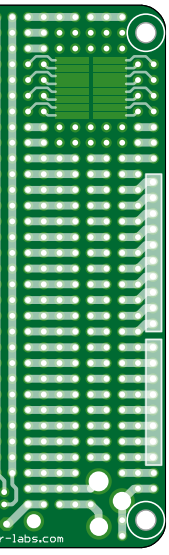
interface LCD mono-fil

tous les bits à la file

Si vous avez déjà travaillé sur différents projets à microcontrôleur, vous connaissez l'angoisse de la ligne d'E/S qui ne trouve plus de port pour se brancher. Que faire ? À regret, supprimer une fonction ou chercher une plus grosse puce, si elle existe, ou faire les frais d'une extension de port ? Rien de palpitant. C'est toujours plus facile avec les périphériques qui n'occupent qu'une broche de port. C'est ce que je propose de faire avec cette interface pour n'importe quel LCD.

52





elektor magazine

DÉCOUVRIR CRÉER PARTAGER

bientôt sur ces pages

afficheurs à tubes fluorescents sur shield Arduino

Quatre afficheurs à tube fluorescent et leur circuit de commande sur une extension Arduino, avec un logiciel qui permet de les utiliser comme horloge, comme voltmètre ou comme compteur. Ça a de la gueule !

mesure de puissance alternative

C'est la carte de liaison Elektor du convertisseur A/N ADS1115 qui est au centre de ce circuit de mesure de puissance à trois calibres, de 0,1 W à 2 kW. La sortie est isolée et l'affichage est assuré par un Arduino.

carte E/S Android

La commande des 22 entrées-sorties de cette carte d'extension universelle est assurée par une application Android. La communication avec le téléphone ou la tablette tactile Android se fait au choix par Bluetooth, WiFi ou USB.

Sous réserve de modification.
Le numéro de septembre paraîtra le 25 août.

Elektor : votre guide

Elektor, c'est bien plus qu'un simple magazine. C'est une communauté d'électroniciens, du débutant au professionnel, désireux d'apprendre, de concevoir et de partager une électronique qui frappe.

57 pays

246817 membres actifs

10 experts &

elektor.post

L'hebdo d'Elektor

Les signaux du changement fusent autour de nous. Avec son rythme hebdomadaire, la lettre électronique elektor.post permet de les suivre à une allure soutenue. Une semaine sur deux, un montage inédit !

www.elektor.com/newsletter

communauté Elektor

Devenez membre, Green ou Gold

Pro ou débutant, rejoignez la communauté. C'est le plus sûr moyen de ne rien rater, ni en électronique classique, ni en techniques embarquées modernes. Vos atouts : accès direct à elektor.labs, forums, lettres d'information hebdomadaires, projets inédits bimensuels, offres exceptionnelles, archives, moteurs de recherche. Les formules Green et Gold donnent droit à de nombreux avantages : **GREEN**, c'est le magazine sous forme numérique, sans papier. **GOLD**, c'est la formule complète avec la version imprimée.

www.elektor.com/memberships

elektor.TV

Pour y voir plus clair

L'image vidéo filmée sans façon est devenue un rival stimulant pour le texte typographié et mis en page ! Les anecdotes visuelles ne manquent pas dans la vie d'un labo d'électronique, surtout quand ça commence à fumer. Et souvent trois plans filmés remplacent efficacement de longs discours. Regardez elektor.tv !

www.youtube.com/user/ElektorIM

Elektor PCB Service

Des cartes à la carte

Adieu perchlorure de fer, bienvenue aux magnifiques circuits imprimés du service de gravure à la demande d'Elektor. Très professionnel et très rapide, très avantageux et tellement plus commode !

www.elektorpcbservice.com

elektor.labs

découvrir, créer & partager

Au cœur de la matrice, **elektor.labs** (le labo d'Elektor) est ouvert et accessible à tous, c'est l'incubateur où éclosent les circuits. Petits et grands, analogiques ou numériques, d'avant-garde ou nostalgiques, ils y sont tous transformés en matière première raffinée, prête à l'emploi, testée et documentée pour vos propres créations.

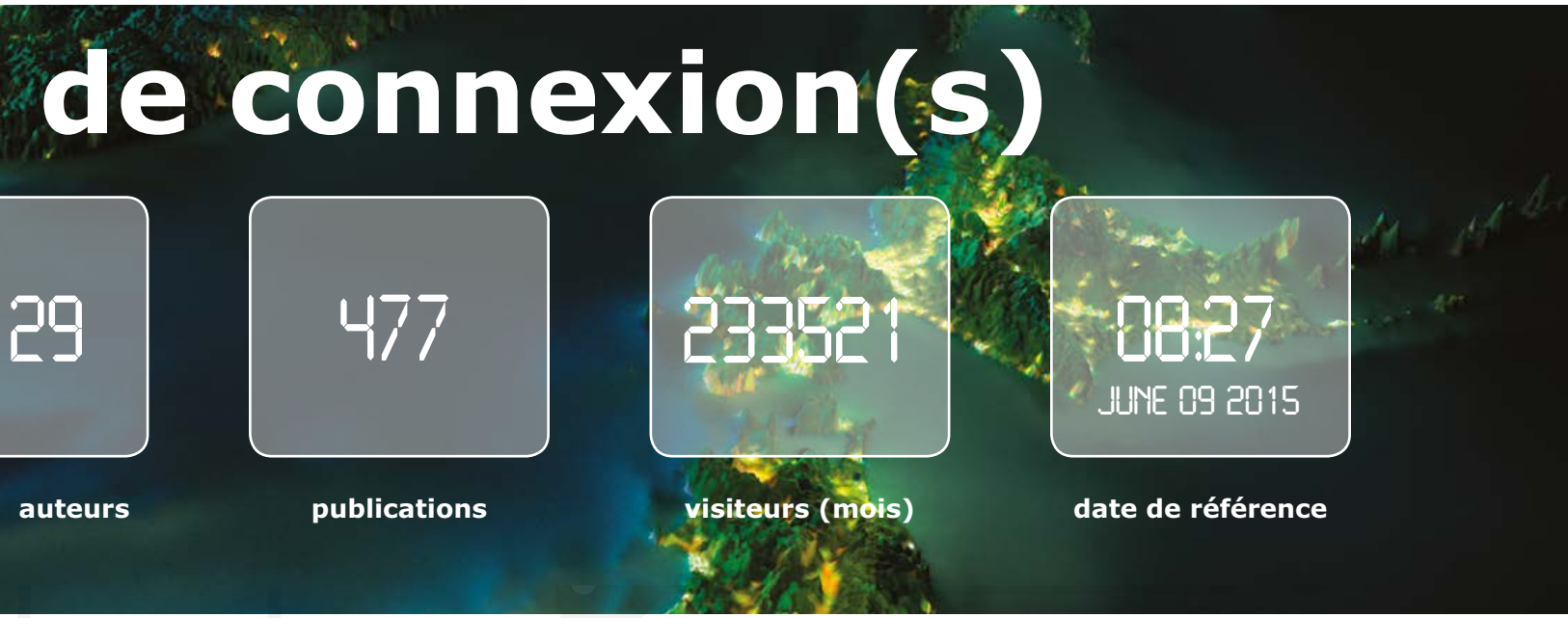
www.elektor-labs.com

elektor.academy

À cheval sur la courbe d'apprentissage

Webinaires, séminaires, cours, présentations, ateliers, lectures, formation en entreprise sont quelques-unes des méthodes pédagogiques utilisées par Elektor pour diffuser la connaissance de l'électronique à tous les niveaux aussi bien parmi les professionnels passionnés que pour les amateurs motivés.

www.elektor-academy.com



elektor.magazine

Plus de 500 pages d'électronique inédite chaque année

Le magazine est le vaisseau amiral, affrété tous les mois par la rédaction internationale d'Elektor pour vous embarquer vers des contrées électroniques nouvelles. Chaque édition, sur papier ou en format numérique, ne se contente pas de rester à la hauteur des précédentes, mais cherche à les surpasser.

www.elektormagazine.fr

e-choppe Elektor en ligne

Votre panier d'achats pour l'électronique

Le magazine et le labo d'Elektor proposent, en coopération avec des partenaires choisis, des produits et des services de haut niveau. Notre e-choppe, véritable caverne d'Ali Baba, est ouverte toute l'année sans interruption pour les électroniciens du monde entier.

www.elektor.fr

livres et DVD Elektor

La puissance de l'information

Elektor aborde tous les domaines de l'électronique : de la programmation des 8 bits aux ARM, des antennes aux diodes zener, des µC aux tubes... Nos ouvrages font référence et autorité dans le monde entier, aussi bien pour les techniques classiques que pour les innovations les plus récentes. Lire pour (mieux) comprendre.

www.elektor.fr

3 formules pour rester connecté avec Elektor !

Formule GREEN 92,50 € par an	Formule GOLD 127,50 € par an	Formule gratuite !
✗ 10 x magazine imprimé	✓ 10 x magazine imprimé	✗ 10 x magazine imprimé
✓ 10 x magazine numérique	✓ 10 x magazine numérique	✗ 10 x magazine numérique
✓ l'accès à l'archive d'Elektor	✓ l'accès à l'archive d'Elektor	✗ l'accès à l'archive d'Elektor
✓ 10 % de remise dans l'e-choppe	✓ 10 % de remise dans l'e-choppe	✗ 10 % de remise dans l'e-choppe
✓ l'accès à Elektor.labs	✓ l'accès à Elektor.labs	✗ l'accès à Elektor.labs
✓ un DVD annuel	✓ un DVD annuel	✗ un DVD annuel
✓ des offres exclusives	✓ des offres exclusives	✓ des offres exclusives
✓ 26 nouveaux projets inédits	✓ 26 nouveaux projets inédits	✓ 26 nouveaux projets inédits
www.elektor.fr/formule-greencard	www.elektor.fr/formule-goldcard	www.elektor.fr/inscription

Et bienvenue dans la machine

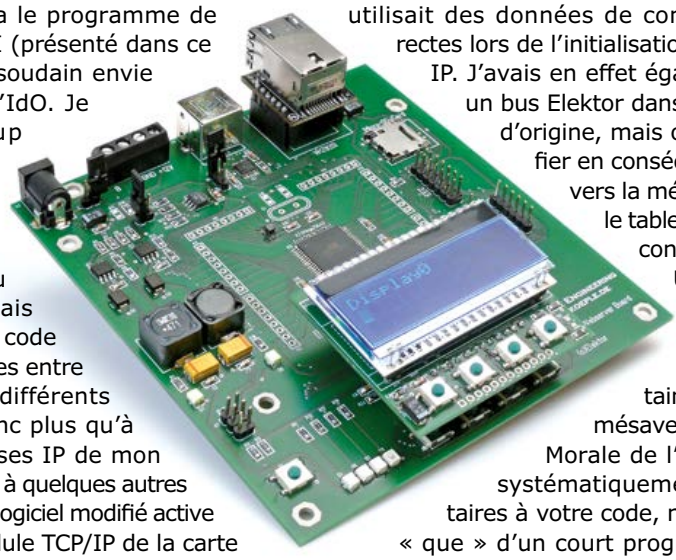
Les langages de programmation sont à l'honneur dans ce numéro : la série *ARM pour néophytes* se poursuit, un article explique la programmation d'applis pour PC et tablettes Windows, et un cours intensif sur l'assembleur débute. Comme vous le remarquerez vite, utiliser ce langage particulièrement proche du matériel se justifie encore pleinement. Outre une exploitation optimale des ressources, une de ses vertus est en effet qu'il astreint à comprendre le fonctionnement du matériel. Programmer un microcontrôleur en assembleur oblige ainsi à connaître son mécanisme de cadencement. J'ai moi-même appris quelques petites choses en préparant ce cours, en plus d'avoir rafraîchi mes connaissances. J'ai surtout été tenté de m'attaquer à une idée qui me hante depuis longtemps : créer une petite machine virtuelle qui exécuterait (le même) *bytecode* sur différents microcontrôleurs afin d'en commuter très rapidement telle ou telle sortie. Un jour, peut-être…

exemple de branchement avec CALL :

```
        movlw   H'FF'
        call    delay1
        movlw   H'20'
        ...
        movlw   H'A0'
        call    delay1
        movlw   H'00'
        ...
;-------------------------
delay1 movwf  H'20'
        ...
        return
```

L'erreur fut de croire qu'il n'y en aurait pas

Je possède chez moi une boîte dans laquelle je range notre *Xmega sur carte polyvalente* ainsi que quelques outils de développement.
Je ressors ma boîte hier avec l'intention de porter sur la Xmega le programme de mon analyseur MIDI (présenté dans ce numéro), mais j'ai soudain envie de me remettre à l'IdO. Je n'ai pas beaucoup de temps, mais je veux au moins réussir l'échange de quelques octets entre mon nouveau PC et la carte. J'avais écrit l'an dernier un code d'échange de données entre la carte Xmega et différents modules, je n'ai donc plus qu'à l'adapter aux adresses IP de mon réseau domestique et à quelques autres paramètres. Le micrologiciel modifié active correctement le module TCP/IP de la carte (je le vois aux LED de la prise réseau qui s'allument), je lance donc un scanner de réseau depuis mon PC.



Mauvaise surprise, le module TCP/IP n'est pas détecté. J'envoie quelques octets et tente d'autres incantations : même échec.
Finalement je trouve l'erreur : la carte Xmega utilisait des données de configuration incorrectes lors de l'initialisation du module TCP/IP. J'avais en effet également configuré un bus Elektor dans mon programme d'origine, mais oublié ici de modifier en conséquence le pointeur vers la mémoire qui contient le tableau des données de configuration.
Un commentaire sur la ligne de code en question m'aurait certainement évité cette mésaventure…
Morale de l'histoire : ajoutez systématiquement des commentaires à votre code, même s'il ne s'agit « que » d'un court programme écrit pour tester quelque chose à la va-vite ! ⏮

(150275 – version française : Hervé Moreau)

trucs et astuces

les lecteurs écrivent aux lecteurs

Des solutions futées qui facilitent la vie des électroniciens

Aimants de contact

Peter Bitzer

Lecteur d'Elektor depuis la première parution en 1970, je n'ai encore jamais rencontré nulle part ce truc-ci. Il existe bel et bien des accumulateurs au CdNi de 32,5 mm de diamètre et 90 mm de longueur, qui bien sûr ne s'accommodent d'aucun chargeur existant. Pour recharger pareille cellule, j'installais dans le temps de grandes pinces crocodile sur les cosses. C'est du bricolage et au moindre mouvement, le crocodile va voir ailleurs.

Quand on sait que les pôles des accumulateurs sont en acier, ça change tout ! En outre on peut trouver à bon compte des aimants en néodyme (Nd) de toutes dimensions. Leur surface est bonne conductrice de l'électricité. Exactement ce qu'il faut pour établir le contact sur les bornes, comme sur la photo.

Les avantages :

- solidité des contacts

- pas de bricolage

- pas de saut de croco

- plus besoin de coupleur de pile

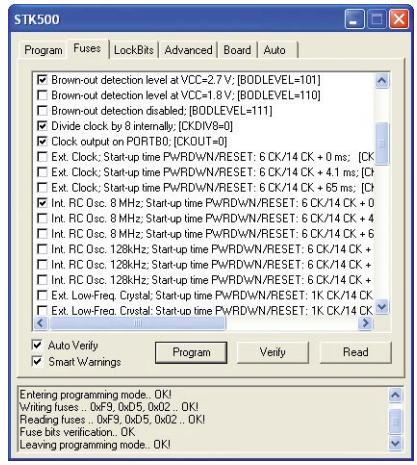
- utilisable sur les accus sans cosses à souder, les NiMH aussi

- plus besoin de câbles de liaison, on peut mettre les accus en série avec une bille de néodyme.

Sortie oscillateur sur l'ATmega168

Burkhard Kainka

Récemment, la programmation d'un Mega8 s'est arrêtée, mémoire pleine. Aucun souci pour monter sur un Mega168, j'y ai 16 Ko à disposition. Lors de la programmation des fusibles, mon attention est attirée sur un réglage qui n'existe pas sur le Mega8 : Clock output on PORTB0. Avec ça, on peut vérifier la précision de l'oscillateur RC sans logiciel supplémentaire, puisqu'il suffit de brancher un fréquencemètre sur PB0. Sitôt dit, sitôt fait : 11 059 kHz. Ah oui ! J'avais branché un quartz externe. Après lui, il y a en série l'oscillateur RC interne à 8 MHz avec un diviseur par 8. Le résultat est alors de 1 001 kHz, avec des variations sur les chiffres derrière la virgule. L'oscillateur tient parfaitement la tolérance garantie de 1 %. La même expérience avec l'oscillateur à 128 kHz présente une précision nettement moindre.



(150135 – version française : Robert Grignard)

Vous avez une solution futée pour arranger une bricole… Une façon bien à vous d'utiliser un composant ou un outil… Vous savez comment résoudre un problème plus facilement ou mieux qu'avec la solution actuelle… Écrivez-nous – chaque astuce publiée vous rapportera 40 € !

appli pour commander des circuits électroniques

la création d'applis Windows pour les électroniciens

Veikko Krypczyk (Allemagne)

Les applications pour appareils mobiles, les *applis*, ont déclenché une nouvelle vague : elles sont intuitives et répondent généralement à la commande tactile. Il est possible de créer des applis pour Windows 8.1 qui tournent aussi bien sur des tablettes que sur des ordinateurs de bureau. Cet article est une introduction *compacte* à la programmation d'applis ; dans la section CRÉER, nous vous montrons comment piloter du matériel externe avec une appli.

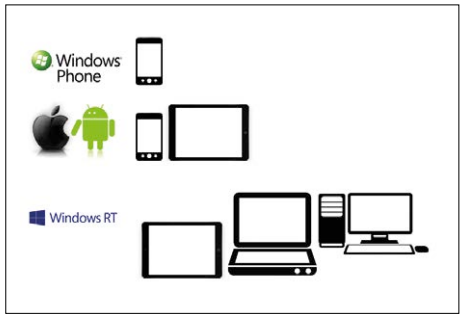


Figure 1. Systèmes d'exploitation pour les applis et leurs plates-formes cibles.

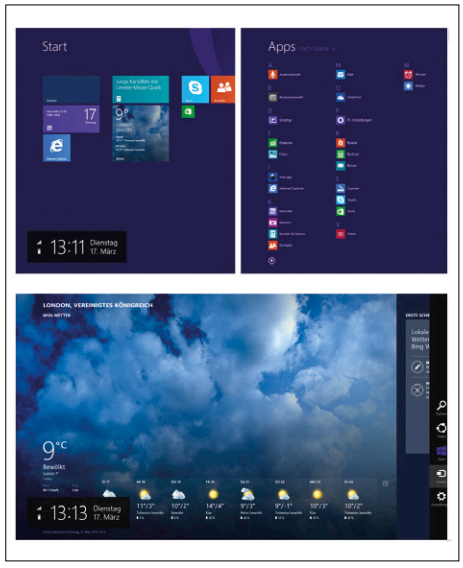


Figure 2. L'écran de démarrage de Windows 8.1 comme « point de rassemblement » de toutes les applis. L'utilisation est toujours intuitive et focalisée sur le contenu. Le logiciel passe au second plan. Ici : une appli pour la météo (montage photo).

Il existe des applis pour tous les domaines d'utilisation et de la vie. Avant de s'attaquer à la programmation de ces petites applications, il est utile de les classer en fonction des plates-formes cibles et des différents domaines d'application qui en résultent. En général, lorsqu'on parle d'applications, il s'agit de programmes pour appareils mobiles. Les systèmes Android (Google), iOS (Apple) et Windows Phone/Windows 8.1 RunTime (Microsoft) se partagent le marché. Alors qu'Android et iOS tournent tant sur ordiphones que sur tablettes, Microsoft fait la différence entre les deux. Windows Phone est prévu pour les ordiphones, Windows 8.1 RunTime pour les tablettes. Windows 8.1 RunTime est disponible sur tous les PC et ordinateurs portables avec comme système d'exploitation une version de Windows à jour, en parallèle en quelque sorte à la version de bureau de Windows *classique* (**fig. 1**). La caractéristique des applis est la concentration sur une seule tâche. L'interface utilisateur (UI) est toujours intuitive et conçue pour la manipulation tactile. En plus de la fonction, le *design* et le ressenti de l'utilisateur jouent un rôle prépondérant (**fig. 2**). Comme Windows 8.1 ne tourne pas seulement sur tablette, mais également sur ordinateurs de bureau ou portables, il est possible de relier du matériel à ces appareils fixes, notamment par USB. Grâce à sa focalisation sur le contenu et à ses concepts modernes de *design* et de commande, l'appli semble idéale pour piloter des circuits électroniques externes. Les utilisateurs finaux veulent déclencher une commutation (sortie) ou être informés de certaines caractéristiques de l'environnement au moyen de capteurs (entrée). Cela doit se dérouler de manière intuitive. Le logiciel devient secondaire. Microsoft prévoit pour Windows version 10 une extension du concept d'appli. L'objectif est de concevoir une appli pour tout type

d'appareil (de l'ordiphone à l'ordinateur de bureau).
Cet article est une introduction à la programmation d'applis pour Windows 8.1 (aussi appelées *Windows Store Apps*, par analogie au système de distribution). L'approche technique est fortement apparentée à celle des applis Windows Phone. Grâce aux applis universelles, le même code peut produire des applications pour les deux systèmes, le code spécifique à une plate-forme est donc minimisé.
Pour construire une telle appli, il y a différentes variantes d'un point de vue technique (**fig. 3**) :

- Définition de l'interface utilisateur en langage de description XAML (basé sur le XML) et utilisation de C# ou VB.Net comme langage de programmation. Une forme spéciale du framework .Net (.Net pour Windows Store Apps) est utilisée comme interface.
- Définition de l'UI grâce au XAML et programmation de la logique en C++. Les services du système sous-jacents sont directement adressés.
- Création de l'UI à l'aide de l'interface graphique DirectX. Cette combinaison est tout à fait adaptée pour les jeux.
- Création de l'UI grâce au HTML et CSS. La logique est implémentée en Javascript. L'accès aux bibliothèques système de Windows RunTime est possible grâce à des composants d'Internet Explorer.

Il est recommandé de recourir à une combinaison de XAML et C# pour les applis spécialisées. Si vous avez l'expérience du développement d'applications pour Windows (bureau) sur la base de la *Windows Presentation Foundation* (WPF, spécification graphique), vous serez en terre connue. Commençons par la mise en place de notre environnement de travail.

Préparatifs

Nous avons besoin d'un environnement de développement (IDE). Il s'agit de Visual Studio (VS), version 2013. Le fabricant Microsoft offre VS gratuitement pour le développement semi-professionnel, ce qui est réjouissant. En [2], on peut télécharger soit VS Community 2013, soit VS Express 2013. Cette dernière version nous suffit. La version Community permet de s'attaquer à d'autres types de projets (par ex. pour le bureau ou le web). Il est recommandé de procéder de la manière suivante après le téléchargement :

1. Mise à jour du système d'exploitation. Windows 8.1 est obligatoire pour développer des applis pour ce système. Windows 7 ne convient pas. Windows 8 peut être mis à jour gratuitement vers Windows 8.1 via le système ou le *Store*.
2. Installation de l'IDE Visual Studio, y compris (si nécessaire) les paquets linguistiques désirés.
3. Relance de la fonction de mise à jour du système d'exploitation.

Après ces étapes, il faut redémarrer l'ordinateur. Cette démarche permet d'éviter ultérieurement des demandes de mise à jour ou des notifications de problème. Comme les logiciels à installer sont lourds (framework .Net, bibliothèques système, Visual Studio), le processus peut durer un moment. Il faut donc faire preuve de patience. Lors du premier lancement, il est nécessaire de s'enregistrer en tant que développeur auprès de Microsoft. Vous serez guidé directement depuis l'IDE. Il est nécessaire de s'inscrire, car seuls les développeurs peuvent installer directement une appli sur leur ordinateur (sauf exception, que l'on appelle *sideloading*). En général, il faut passer par le *Store*. La licence est gratuite et doit être régulièrement renouvelée. Un compte Microsoft est obligatoire, il sera créé à ce moment si vous n'en avez pas. Après les activités de préparation, vous pouvez démarrer Visual Studio 2013 (**fig. 4**).

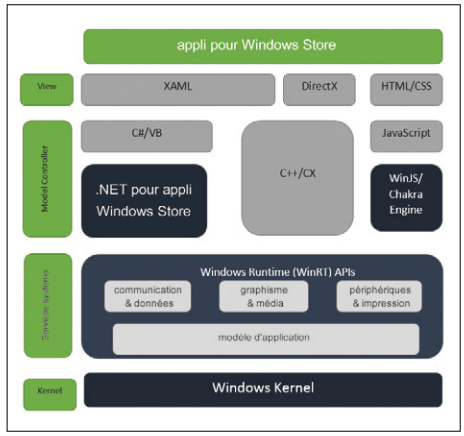


Figure 3. Variantes techniques du développement d'une appli pour Windows Store [1].

Première application

Pour notre première expérience, nous allons créer une appli (dans le jargon, une application « Hello World ») pour nous familiariser avec la structure/le processus. Après le démarrage de Visual Studio, nous cliquons sur *Fichier* | *Nouveau* | *Projet*... Dans la fenêtre de dialogue, les modèles sont classés par langage de programmation. Sous la section Visual C#, l'entrée *Store Apps*| *Windows* mène au but. On sélectionne ensuite *Projet vide*. Il reste ensuite à ajouter le nom de l'appli et l'emplacement de stockage. Nous laissons inactive l'option *Ajouter au contrôle de code source* pour nos premiers essais (**fig. 5**).

Après avoir cliqué sur OK, l'IDE crée le code cadre pour l'appli et il est possible de la lancer (flèche verte dans la barre d'outils). Comme les applis Store peuvent être exécutées directement sur l'ordinateur

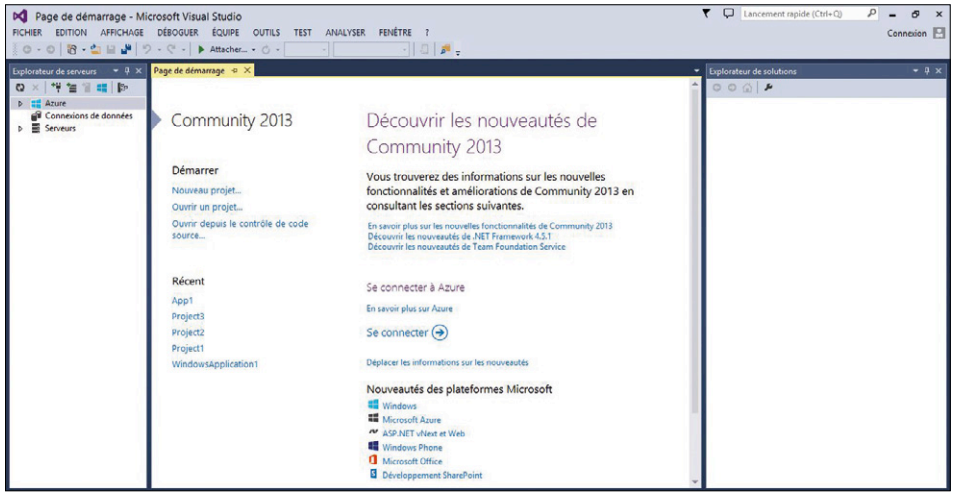


Figure 4. Écran de démarrage de l'environnement de développement Visual Studio 2013.

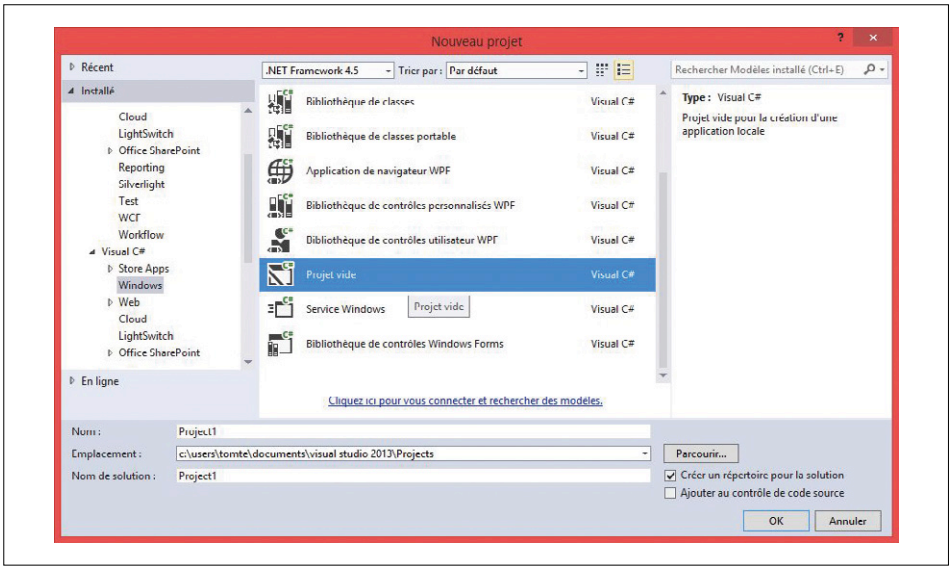


Figure 5. L'assistant de projet de Visual Studio propose des modèles pour créer une appli.

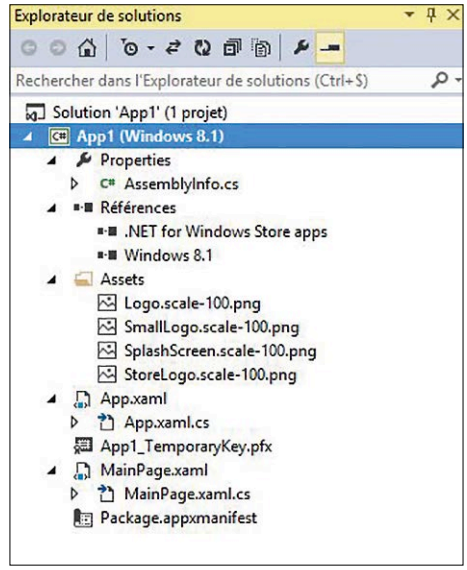


Figure 6. La structure de l'appli dans l'explorateur de solutions.

Tableau 1. Éléments d'une appli pour Windows Store

élément du projet	description courte
Properties	Contient dans le fichier AssemblyInfo.cs les propriétés générales du projet. Il n'est pas prévu de modifier directement ces informations. Les paramètres peuvent être modifiés par une fenêtre de dialogue (Projet| Paramètres de {Nom de l'appli}). Le numéro de version fait partie de ces informations.
Références	C'est ici que sont indiquées les bibliothèques que l'appli nécessite.
Assets	Dépôt des ressources pour une appli, par exemple les images pour les tuiles. D'autres ressources se trouvent dans ce dossier, éventuellement dans des sous-dossiers.
App.xaml | App.xaml.cs	Fichiers de code source pour le lancement du programme. En général, il n'est pas nécessaire d'effectuer de modifications manuelles.
{Nom de l'appli}_TemporaryKey.pfx	Les applis sont signées par un certificat unique. Celui-ci peut être établi directement dans l'IDE.
MainPage.xaml | MainPage.xaml.cs	Les fichiers XAML décrivent les écrans qui forment l'interface utilisateur. Chaque page correspond à un fichier de ce type. Le code relatif aux éléments de l'interface peut être déposé directement dans le fichier cs correspondant. Par ex., un gestionnaire d'événements qui est exécuté lorsqu'on touche un bouton.
Package.appxmanifest	Un *Package-Manifest* est nécessaire pour la distribution de l'appli sur le *Store*. Un double-clic sur cette entrée ouvre l'éditeur intégré. Font partie des paramètres notamment le nom de l'appli, le mode d'affichage (portrait, paysage) et les permissions demandées, comme l'accès à l'internet ou aux services de localisation.

Tableau 2. Conteneurs de mise en page essentiels

conteneur de mise en page	description
DockPanel	La position (*Top*, *Bottom*, *Left* et *Right*) des éléments de commande à afficher est spécifiée. Le *Control* ajouté en dernier complète la place restante automatiquement.
Grid	Les éléments sont disposés sous forme d'un tableau. Il faut encore définir le nombre de lignes (*Rows*) et de colonnes (*Columns*). On s'appuie sur la structure de la grille lors de la déclaration des éléments inférieurs, c'est-à-dire qu'il faut indiquer la ligne et la colonne désirées, où l'élément sera placé. Les paramètres *Grid.Column* et *Grid.Row* sont à définir pour chaque élément intégré.
StackPanel	Les éléments de commande inférieurs sont disposés les uns à côté des autres, ou les uns en dessous des autres. Le sens de positionnement est déterminé par la propriété *Orientation*.

du programmeur, nous n'avons besoin ni d'une tablette ni d'un émulateur pour tester l'appli. Celle-ci affiche naturellement un écran noir en mode plein écran. Observons la structure dans l'explorateur de solutions. (*Affichage* | *Explorateur de solutions*) (**fig. 6**). La solution contient dans ce cas uniquement un projet qui comprend l'appli Store. Il est recommandé de diviser les projets logiciels complexes en plusieurs projets. Les principaux éléments du projet sont décrits dans le **tableau 1**.

Agencer l'interface utilisateur

L'UI des applis Windows Store est basée sur un sous-ensemble de la WPF. Cette dernière est entièrement au format vectoriel. Les possibilités graphiques sont très variées en ce qui concerne les formes et les couleurs. La définition de l'interface utilisateur s'effectue grâce à un langage de description XML propre (XAML). Il est ainsi possible de la séparer totalement du code de programmation. La « mise en page relative » est fixée en fonction des autres éléments, plusieurs conteneurs peuvent être sélectionnés (**tableau 2**).

Comme les applis tournent sur des dispositifs avec des écrans très différents (taille et résolution), le positionnement relatif des éléments est incontournable. Un conteneur de mise en page permet d'inclure tous les éléments de commande d'un écran. La classe de base de tous les conteneurs de mise en page est la classe *Panel*. *Children* est une caractéristique importante dont héritent les éléments inclus dans un tel conteneur. Le positionnement des éléments n'est pas défini explicitement, mais est accordé de manière automatique. Les paramètres *Width* (largeur) et *Height* (hauteur) sont disponibles pour la plupart des éléments, mais devraient être utilisés avec parcimonie (par ex. pour des boutons).

La taille d'un élément est plutôt déterminée par son contenu en combinaison avec le conteneur qui contient cet élément. Il est possible de développer l'UI dans Visual Studio à l'aide de l'outil de dessin intégré (**fig. 7**). Les exigences relatives à la commande tactile sont respectées de manière exemplaire. Les éléments de commande contiennent les événements nécessaires. Comme les applis peuvent aussi être pilotées au clavier et à la souris, des éléments pour le traitement des signaux de navigation à la souris et au clavier sont disponibles.

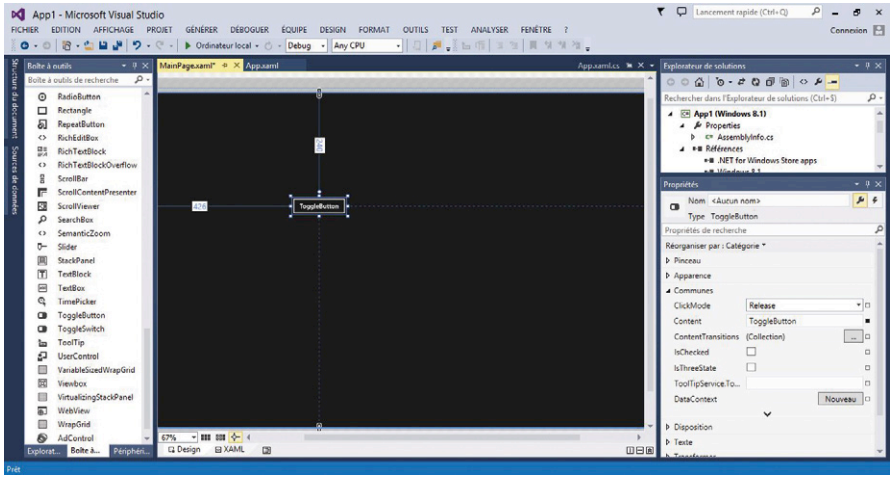


Figure 7. L'interface utilisateur peut être créée graphiquement, les éléments de contrôle sont simplement tirés avec la souris jusque dans la zone d'affichage.

La WPF pour les applis Windows Store propose un très grand choix d'éléments de commande pour concevoir une interface utilisateur attrayante et moderne. Ces éléments sont triés selon leur utilisation dans une arborescence. Pour une introduction à la conception d'interface utilisateur en XAML, reportez-vous au **listage 1** et aux remarques qui suivent. Notre exemple est étendu dans l'article qui traite de la commande de matériel externe.

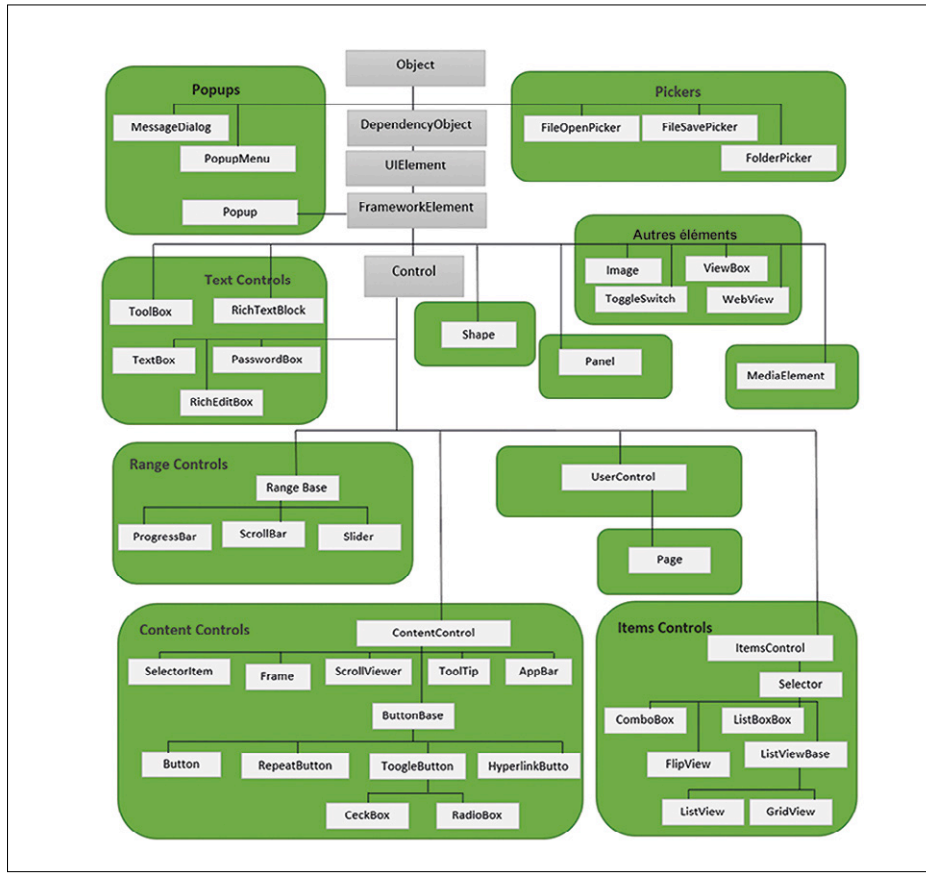


Figure 8. Hiérarchie des éléments de l'interface utilisateur pour les applis de *Windows Store*.

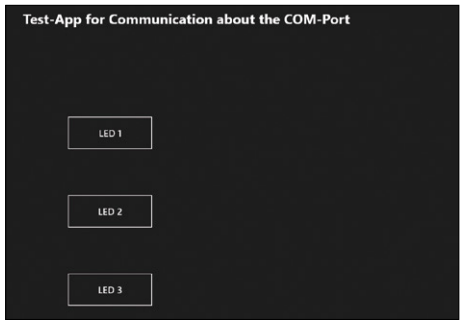


Figure 9. L'interface utilisateur de notre appli.

- L'élément racine s'appelle *Page*. Il comprend exactement une page.
- Suit l'en-tête avec l'inclusion de bibliothèques et de commentaires destinés aux concepteurs graphiques.
- Tous les éléments de commande de la page sont inclus dans un conteneur de mise en page supérieur. Ce conteneur est de type *Grid* (tableau), mais sans définition de lignes ni colonnes. Il est recommandé de se tenir à ce conteneur de mise en page. Des éléments inférieurs peuvent être inclus dans d'autres conteneurs de mise en page (imbrication libre). Ainsi, on obtient une structure facile à adapter. Le conteneur de mise en page le plus élevé peut servir à établir le fond d'écran (propriété Fond d'écran).
- Dans notre exemple, nous avons ajouté un élément de type *TextBlock* pour le titre et trois boutons de type *ToggleButton*. Ces derniers sont regroupés dans un conteneur (de type *StackPanel*). Ainsi, ce bloc de boutons peut être placé et déplacé de manière autonome. Les boutons *ToggleButton* sont déterminés par leurs taille (*Width*, *Height*), distance (*Margin*) et taille de police (*FontSize*). Comme la définition est la même pour les trois boutons, il est possible de créer un *Template* (modèle) pour la page ou l'appli.
- Si un *StackPanel* ne reçoit pas de paramètre pour l'orientation, l'agencement des éléments s'effectue de manière standard (à la verticale).

L'UI de cette simple appli ne comporte qu'un texte d'avertissement et trois interrupteurs pour allumer et éteindre des LED (**fig. 9**). La liaison avec le code à exécuter est établie lorsque les éléments de commande concernés sont reliés aux événements correspondants. Par exemple, l'événement *Click* sur un bouton de type *ToggleButton* est lié au code correspondant. Il sera déclenché par un clic de souris ou une pression sur l'écran tactile.

Créer le programme

Le programme est écrit en langage orienté objet C#. Comme mentionné auparavant, vous pouvez aussi utiliser le langage VB.Net. Les deux langages sont équivalents parce que le framework .NET permet de faire abstraction du langage et que des sous-ensembles sont disponibles pour les applis Windows Store. Il y a des différences dans les concepts supportés et la façon de formuler les programmes. Pour cette expérience, nous avons choisi le C#. Une introduction complète à ce langage dépasserait le cadre de cet article ; nous nous concentrerons donc sur quelques éléments clés (**tableau 3**).

Le langage a continuellement évolué, les éléments fonctionnels entre autres ont pavé le chemin qui mène au C#. Ces éléments en combinaison avec les nombreuses classes du framework .NET permettent d'exploiter pleinement ses performances.
De nombreuses solutions prédéfinies existent pour résoudre une multitude de problèmes. Il n'est donc pas nécessaire de réinventer la roue à chaque fois. L'inclusion de bibliothèques externes permet également d'augmenter les capacités. L'éditeur de code source offre toutes les fonctions que l'on attend d'un environnement de développement moderne, par ex. une reconnaissance complète et un formatage de la syntaxe ainsi que la transformation et l'amélioration intelligentes du code (*refactoring*).

Listage 1. Code XAML de l'interface de l'utilisateur de notre appli.

```
<Page
  x:Class="Elektor.MainPage"
  xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
  xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
  xmlns:local="using:Elektor"
  xmlns:d="http://schemas.microsoft.com/expression/blend/2008"
  xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006"
  mc:Ignorable="d">

  <Grid Background="{ThemeResource ApplicationPageBackgroundThemeBrush}">
    <StackPanel>
      <TextBlock Margin="50" Text="Test-App for Communication about the COM-Port" FontSize="32" FontWeight="Bold" />
      <StackPanel Margin="100">
        <ToggleButton Margin="50" Width="200" Height="80" FontSize="22">LED 1</ToggleButton>
        <ToggleButton Margin="50" Width="200" Height="80" FontSize="22">LED 2</ToggleButton>
        <ToggleButton Margin="50" Width="200" Height="80" FontSize="22">LED 3</ToggleButton>
      </StackPanel>
    </StackPanel>
  </Grid>
</Page>
```

Distribution des applis

Il est nécessaire de dire quelques mots à ce sujet. Comme pour toutes les applis, il n'est pas prévu que le développeur distribue directement son produit aux utilisateurs. En général, il faut que l'installation passe par le magasin des applis (*Store*). À des fins de test, il est possible d'autoriser l'installation d'une appli sur d'autres ordinateurs à l'aide d'une licence de développeur. Il est également possible, grâce au *sideloading*, de distribuer l'appli sur des appareils cibles, c'est-à-dire de passer outre le *Store*. Une fois que l'appli a quitté la phase de test et peut intéresser un plus large public, elle devrait être distribuée via le *Store*. Il faut pour cela remplir certaines conditions. Ainsi, les ressources de type image doivent être intégrées dans l'appli d'une certaine manière et au bon endroit (par ex. pour l'affichage des tuiles). Enfin, il faut passer un dernier test par un outil de certification de Microsoft, charger l'appli dans le Store, ajouter les données formelles (limites d'âge, protection des données, conditions de licence et prix). Après un court délai, Microsoft confirme la certification et met l'appli à disposition de tous les utilisateurs intéressés. Tout cela semble bien compliqué, mais ce processus est bien documenté en [3].

Conclusion et perspectives

En ce qui concerne la programmation d'applis pour le *Store* , il y a encore bien d'autres choses à dire ; reportez-vous à la littérature spécialisée. L'article dans la section Créer intéressera particulièrement les électroniciens, car il décrit comment piloter du matériel externe par un port COM virtuel via USB. Du côté matériel, la carte Arduino Uno et le *shield* d'extension Elektor font leur retour. Pour ce qui est de l'aspect logiciel, nous utilisons une astuce, car comme les applis tournent dans un « bac à sable », leurs capacités de communication sont restreintes pour des raisons de sécurité. Nous avons néanmoins trouvé une solution ! ⏮

(140411 – version française : Thierry Destinobles)

Tableau 3. Principales caractéristiques du C#

élément de langage	mots clés	syntaxe/exemple
commentaire dans le code source	Sur une ligne : // Sur plusieurs lignes : /*…*/	// voici un commentaire simple /* voici un commentaire sur plusieurs lignes */
types de données élémentaires	byte int float double decimal string	int Nombre = 5; string text = "Une chaîne de caractère";
opérations de base	+ - * /	int somme = terme1 + terme2; float produit = facteur1 * facteur2;

instructions de boucle	boucle for	for (int i = 0; i < limite sup; i++)
		{
		// corps de boucle
		}

	boucle while	while (condition == true)
		{
		// corps de boucle
		}

	boucle do	do
		{
		// corps de boucle
		} while (condition == true)

	boucle foreach	foreach (type variable in type énuméré)
		{
		// corps de boucle
		}

	break	for (int i = 0; i < limite sup; i++)
		{
		// corps de boucle
		if (condition2 == true) break;
		}

	continue	for (int i = 0; i < limite sup; i++)
		{
		// corps de boucle
		if (condition2 == true) continue;
		}

instructions de sélection	If else if else switch (…) case	If (a==3) a=5;

types de données définis par l'utilisateur	struct	struct ensemble de données
		{
		int Age;
		string Nom;
		}

classes propres	class	class MaClasse: ClasseDeBase
		{
		int UnAttribut;
		private void UneMéthode(…)
		{
		….
		}
		}

Liens

[1] Huber, T. C., Windows Store Apps mit XAML und C#, Galileo Press, 2013.

[2] www.visualstudio.com/fr-fr/downloads/download-visual-studio-vs.aspx

[3] https://dev.windows.com/fr-FR/publish

Q&R

presque tout ce que vous avez toujours voulu savoir sur le...

soudage par contact

Jan Visser et **Jaime González-Arintero**

Vous n'avez pas tous la possibilité de nous rejoindre en ligne, ou pas forcément le temps de nous revoir sur elektor.tv. Voici donc, pour les absents ou les pressés, la version imprimée de quelques trucs & astuces de soudage que nous avons présentés lors d'un Q&R en ligne.

En direct (pour le meilleur et pour le pire)

Elektor.tv a filmé certaines Q&R présentées par le duo Jan Visser (qui la plupart du temps explique) et Jaime González-Arintero (qui souvent apprend).

La vidéo de ce Q&R est ici : [po.st/soldersoldiers].

Q ***Le soudage par refusion des minuscules QFN peut subir l'effet « pierre tombale ». Existe-t-il un moyen de positionner correctement les QFN par soudage de contact ?***

R Nous parlerons du soudage sans contact dans un autre numéro, mais disons tout de suite que cette question intéressante implique aussi le bon vieux fer à souder ! La **figure 1** montre un exemple d'effet « pierre tombale » (*tombstoning*) sur une résistance CMS. Cet effet est dû aux temps de refusion différents qu'ont chacune des pastilles sur lesquelles repose le CMS (la pâte d'une pastille fond avant l'autre). Ces différences entraînent l'apparition d'un couple suffisant pour faire se redresser le composant.

L'effet pierre tombale peut être plus ennuyeux avec les QFN, mais on peut profiter de leur plage de refroidissement : si la pastille a un trou (**fig. 2**), on peut en effet fixer le composant sur la carte à l'aide d'adhésif Kapton, puis le souder en passant la panne par le trou et la face arrière. Cet adhésif supporte la température en jeu, le QFN est donc parfaitement fixé et on peut alors souder ses broches sur la face avant.

Q ***Une astuce quelconque pour le soudage à la traîne ?***

R Du gâteau pour les pros, le *drag soldering* n'est pas vraiment de la tarte pour les débutants, car il n'est pas évident d'éviter les ponts de soudure entre les broches, et si l'on n'est pas assez rapide, on peut « cramer » un composant. Une panne « cuillère » (p. ex. la *Mini Spoon* C245-931 de JBC) facilite le soudage à la traîne puisqu'on peut « charger » la soudure sur la panne et la relâcher graduellement. Il faut trouver le meilleur angle (**fig. 3**) et doucement tourner la panne durant le transfert de la soudure aux broches. Le flux doit être appliqué copieusement pour que la soudure reste fluide, et la panne ne doit pas être oxydée. La soudure sans plomb a en outre une viscosité différente et un débutant peut avoir plus de mal avec elle.
Il peut être difficile de trouver la bonne vitesse de fer (entre les broches) et le juste apport de soudure. Essayez en ne fondant pas la soudure directement sur l'arête de la panne, mais à un ou deux millimètres de son bord.

Figure 1. Le redressement de la résistance, un cas flagrant d'effet pierre tombale.

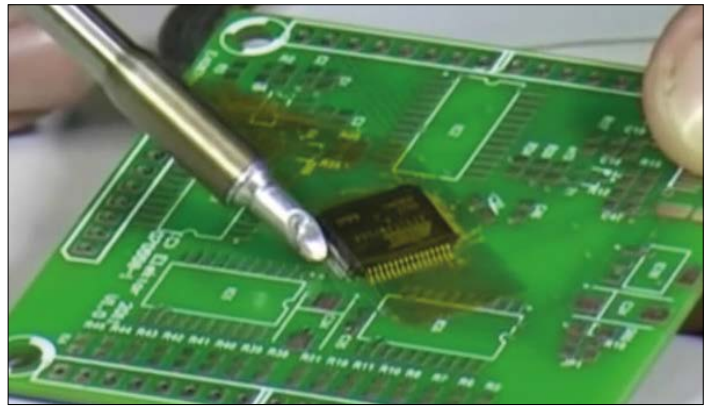


Figure 3. Une « panne-cartouche » de type « mini-cuillère ».

Q ***Qu'est-ce qu'un flux ?***

R Un flux est un agent chimique qui nettoie la soudure, facilite son écoulement et, plus important, réduit les oxydes métalliques et diminue la tension de surface pour faciliter le soudage. Le principal flux utilisé pour les travaux de soudage et dessoudage est un flux résineux — il faut éviter tout autre flux acide, p. ex. celui qu'utilisaient nos (grands-)pères pour souder des tuyaux de cuivre, car il peut corroder les connexions au point de détraquer le circuit.
Les fils de soudure ont souvent un cœur de résine qui suffit pour la plupart des applications, mais un flux supplémentaire peut être utile lorsqu'on modifie un circuit, et il est d'ordinaire indispensable pour souder des CMS à pas très fins. Parce qu'ils sont pratiques, nous recommandons les stylos ou seringues de flux liquide. Des flux spéciaux existent aussi pour la soudure sans plomb.

Note: Nous utilisons (presque) tous le verbe *souder* dans des circonstances où le terme exact serait *braser* ; le bord des pièces que nous assemblons ne sont pas modifiés par cette opération, comme c'est le cas quand, à proprement parler, on les soude.

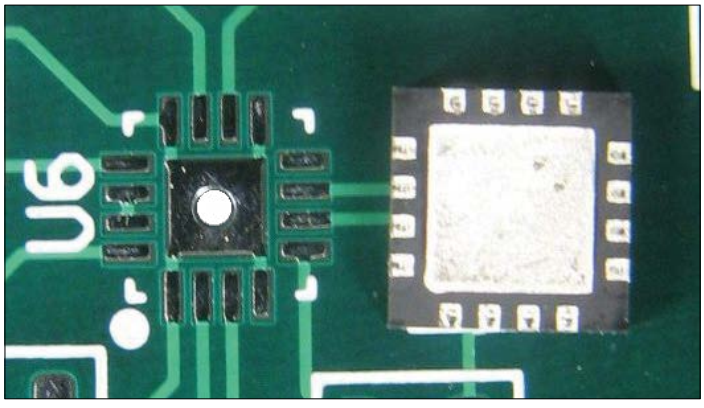


Figure 2. Trou dans une pastille pour un boîtier ***Q**uad **F**lat **N**o-lead*.

Q ***Que sont le soudage au trempé et le soudage à la vague ?***

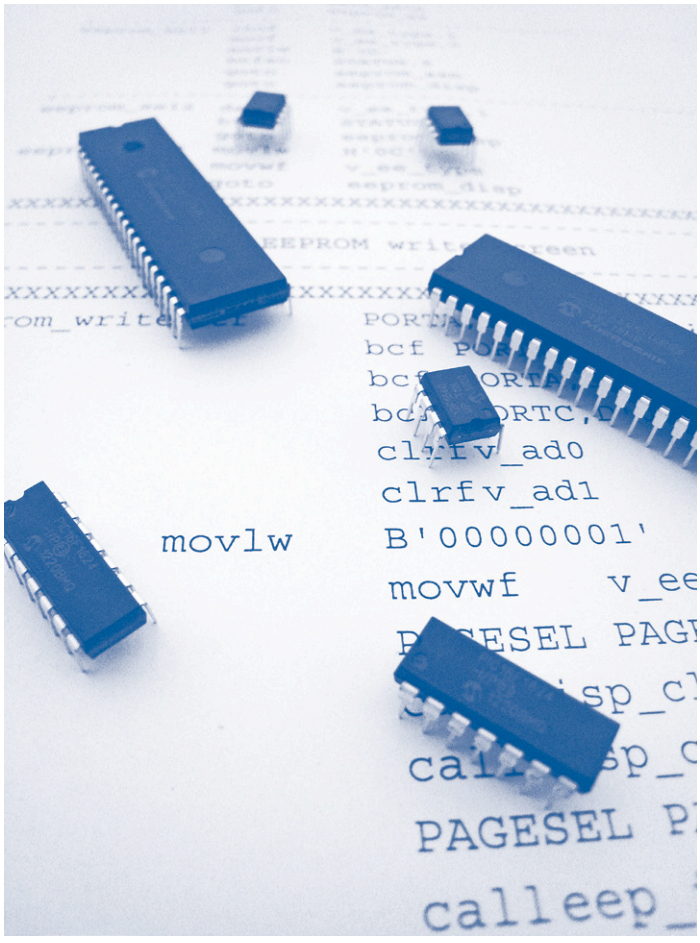
R Le soudage au trempé consiste à appliquer une couche de flux sur la carte où sont placés les composants (généralement des traversants). Le circuit est ensuite trempé — sans être totalement immergé — dans un bain de soudure que les broches « absorbent », établissant ainsi les connexions pour toutes les pastilles. Cette technique le plus souvent manuelle est pratique pour fabriquer des petites séries.
Le principe du soudage à la vague est le même, si ce n'est qu'il est automatisé et que les pastilles de la carte ne sont pas toutes soumises à la soudure en même temps : une vague de soudure fondue traverse la carte jusqu'à ce que toutes les pastilles soient recouvertes de façon adéquate. C'est un peu comme si la carte *surfait* sur la crête d'une vague de soudure en fusion, ce qui doit être cool quand on est une carte. ⏮

(150150 – version française : Hervé Moreau)

Vous n'avez pas trouvé réponse à la question que vous vous posez ?
Patience, la place nous manquait pour traiter en une seule fois ce vaste sujet qu'est le soudage, mais nous y reviendrons. N'hésitez pas à nous poser vos questions !

Restez connecté !

Les prochaines Q&R porteront sur…



cours intensif d'assembleur (1)

le premier programme

Miroslav Cina (Allemagne), miroslav.cina@t-online.de

De nos jours doit-on encore parler « assembleur » ? Notre réponse est un *oui* (franc et massif), car pour les microcontrôleurs à 8 bits en particulier, optimiser la longueur du code et la rapidité d'exécution peut s'avérer utile. Notre cours intensif autour d'un petit µC PIC allie comme d'habitude théorie et pratique.

L'assembleur est un langage de programmation très proche du matériel, ce qui présente des avantages et des inconvénients.
Avec un langage de programmation de haut niveau, c'est du compilateur que dépend l'efficacité de la compilation du code source. En assembleur, pour obtenir un résultat optimal, il faut s'occuper soi-même tant de la taille du code que de la vitesse d'exécution. La conversion de l'assembleur en langage machine (ou code hexadécimal) est en effet parfaitement définie.
La longueur du programme est un aspect très important en cas d'utilisation d'un *petit* µC (un PIC10F200 par ex. n'a que 256 mots de mémoire de programme). La performance peut jouer un rôle important dans les calculs complexes là où un compilateur C est capable parfois de produire des inepties. Dans de tels cas, il est alors logique de programmer directement en assembleur les parties de programme critiques. Autre avantage : pour les opérations à la chronologie critique, on peut calculer exactement le nombre de cycles d'horloge (c.-à-d. la durée) nécessaire au déroulement d'une routine en assembleur. L'assembleur est aussi le meilleur moyen de découvrir le matériel dans le détail. Les langages de haut niveau cachent (plus ou moins) aux yeux du programmeur ce qui se passe dans les registres & Co.
Cependant, en assembleur, tout est vraiment à faire soi-même – la simple multiplication de deux petits nombres peut requérir une certaine réflexion.

Matériel et logiciel

Comme je travaille depuis longtemps avec les µC PIC de Microchip, j'ai opté ici pour l'un des plus petits d'entre eux, le PIC12F675 [1]. Ce µC d'un euro ou moins fait parfaitement l'affaire pour de nombreuses applications simples.
Pour les expériences proposées, j'ai utilisé un programmateur PICkit2 et une petite carte d'expérimentation (l'ensemble est souvent vendu sous le nom de *PICkit2 Starter Kit*). Dans la prochaine partie du cours, nous verrons comment réaliser soi-même une carte d'expérimentation simple. Rien n'interdit bien entendu d'utiliser pour ce cours le programmateur PICkit3 plus récent.
Le logiciel utilisé est mis à disposition gratuitement. Sous Windows, nous pouvons utiliser un éditeur de texte (Notepad) pour saisir le code source, pour l'assembler ensuite, c.-à-d. le traduire en langage machine, à l'aide du programme MPASM.exe (inclus dans la Suite MPASM de Microchip). La Suite MPASM fait partie de l'IDE (***I**ntegrated **D**evelopment **E**nvironment*) MPLAB. On pourra télécharger l'IDE MPLAB gratuitement chez Microchip [2].
Finalement, le logiciel du PICkit2 assure le « transport » vers le µC du fichier hexadécimal résultant de l'assemblage. Voilà, c'est tout.

Les registres

Commençons par découvrir notre µC et apprendre quelques instructions d'assembleur. Pas de panique – le premier paquet de théorie n'est pas très compliqué ; ensuite nous pourrons démarrer dans la foulée.
Le PIC12F675 est un µC à 8 bits de la famille *midrange* (milieu de gamme) de Microchip. Ce µC est proposé entre autres en boîtier PDIP-8 (***P**lastic **D**ual*

***I**nline **P**ackage*). Six des huit pattes sont des broches de port numérique d'usage général (GP0 à GP5). GP3 ne peut être utilisée qu'en entrée, les cinq autres broches peuvent l'être soit en entrée, soit en sortie. Le PIC dispose de 1 024 mots de mémoire de programme flash, ainsi que de 64 octets de SRAM utilisables librement et de 128 octets d'EEPROM.

Lorsque l'on programme en assembleur, il faut travailler avec des registres ; ce sont, ici, des unités de mémoire de 1 octet qui ont une fonction logique spécifique. Certains registres font partie du noyau ; ils peuvent, par ex. lors de calculs, stocker des valeurs de 8 bits ou donner des informations quant à l'état du µC. D'autres registres sont responsables des blocs périphériques du µC, le registre GPIO qui donne l'état des entrées/sorties par exemple.

Pour les PIC, il y a toujours au moins quatre registres dans le noyau :

- Le registre W
- Le registre de configuration (__CONFIG)
- Le registre des options (OPTION_REG)
- Le registre d'état (STATUS)

La mémoire SRAM de 64 octets utilisable librement, mais aussi la plupart des registres, sont accessibles à des adresses de mémoire bien définies. Une adresse sur 8 bits permet d'accéder à 256 emplacements de mémoire. Un coup d'œil à la cartographie de la mémoire (*memory map*) (**fig. 1**) nous apprend que l'on n'a pas réellement 256 emplacements de mémoire à disposition ; le registre d'état par ex. est accessible aux adresses 03h et 83h.
La cartographie montre que le registre GPIO (nous l'utiliserons bientôt) se trouve à l'emplacement 05h ; nous voyons aussi que nous pouvons accéder à la mémoire d'application (64 octets) sous les adresses de mémoire comprises entre 20h et 5Fh. Les registres sans emplacement de mémoire, comme le registre des options et le registre W, sont des exceptions.
Le registre W (W = *Work*, travail) est le registre principal du µC. Toutes les opérations arithmétiques par ex. se font par le biais du registre W.
Le registre d'état est subdivisé en bits individuels (cf. **fig. 2**). L'important, à cet endroit, est le bit n° 5 (RP0), le *Register Bank Select Bit*. Il est utilisé pour la sélection entre les deux banques de mémoire ; le bit doit être mis à 1 (levé) ou à 0 (effacé) avant que l'on ne puisse donner une adresse comme paramètre à une instruction subséquente. Nous avons accès aux emplacements de mémoire 80h à FFh lorsque le bit est levé ; si RP0 = 0, nous avons accès aux emplacements de mémoire 00h à 7Fh.
L'écriture du registre Config se fait lors du flashage du µC (cf. ci-dessous). Il définit par ex. l'utilisation (ou non) de l'oscillateur interne du µC.

Pour le moment, laissons de côté le registre des options. Maintenant nous allons nous intéresser brièvement à trois autres registres qui, à la différence de ceux qui ont précédé, s'occupent des périphériques.

TRISIO, GPIO et ANSEL

Les registres TRISIO, GPIO et ANSEL servent à commander les E/S. Pour le registre ANSEL, pour le moment il faut juste retenir que nous devons le mettre à 00h pour les premières expériences – ce qui désactive les fonctions analogiques.
Le registre TRISIO commande le sens de la communication. Les bits correspondants du registre TRISIO définissent l'utilisation d'une broche de port soit en entrée, soit en sortie. La valeur *0* en fait une sortie, un *1* une entrée. Si par ex. le premier bit est à zéro (on écrit `TRISIO<0> = 0`), la broche GP0 est utilisée en sortie.

La seule exception est, nous le disions plus haut, la broche de port GP3 (qui remplit également la fonction MCLR - *Master Clear*) ; elle ne peut être utilisée qu'en entrée. Le bit correspondant du registre TRISIO est donc toujours à 1.
Le registre GPIO nous permet de commander directement les entrées/sorties. Toute écriture dans le registre GPIO affecte directement les broches configurées en sortie, les entrées pouvant quant à elles être interrogées par une lecture du registre GPIO.

Premières instructions

Il est temps maintenant de découvrir quelques instructions d'assembleur (les *instructions de base* en quelque sorte). Le PIC12F675 fait partie des µC RISC (*Reduced Instruction Set* = à jeu d'instructions réduit) – il n'y a que 35 instructions à apprendre.

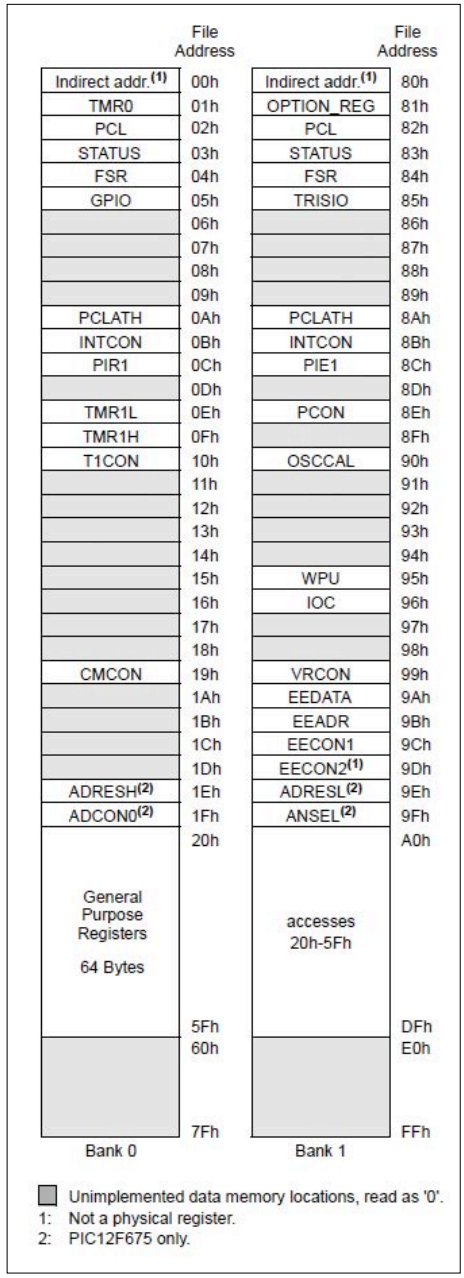


Figure 1. La cartographie de la mémoire donne les adresses des registres et des emplacements de mémoire utilisables librement.

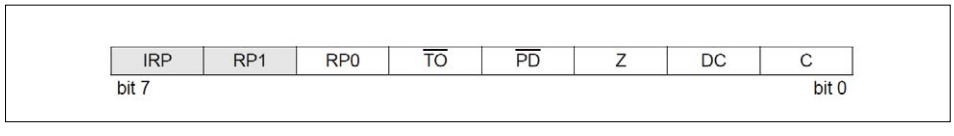


Figure 2. Les bits du registre d'état (*status*).

MOVLW et MOVWF

L'instruction `MOVLW` charge une valeur à 8 bits dans le registre W. La valeur elle-même peut être représentée en binaire, hexadécimal ou décimal.
La syntaxe de la commande est la suivante :

```
movlw   k
```

où *k* est une valeur se trouvant dans l'intervalle compris entre 00h et FFh (1 octet).

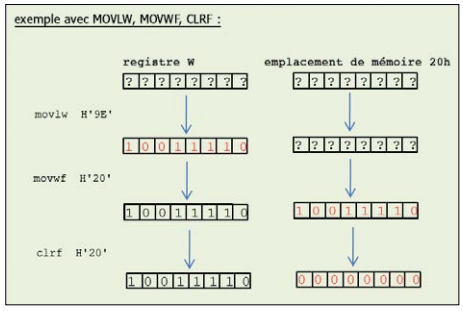


Figure 3. Mode opératoire des instructions *MOVLW*, *MOVWF* et *CLRF*.

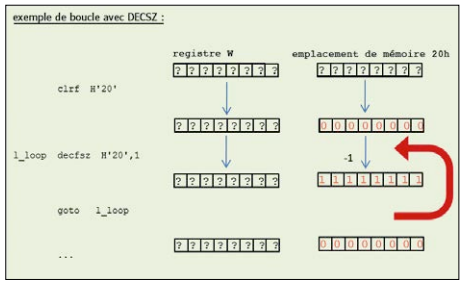


Figure 4. *DECFSZ* permet de réaliser des boucles.

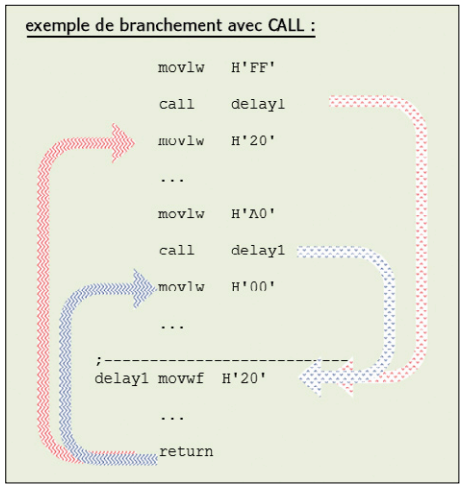


Figure 5. *CALL* sert à appeler un sous-programme.

D'ailleurs, la casse (écriture en majuscules ou minuscules) des instructions en assembleur est sans importance, nous pouvons donc également écrire :

```
MOVLW   k
```

Les exemples ci-après montrent comment écrire la valeur sous forme binaire (préfixe B), hexadécimale (H) et décimale (D) :

```
movlw   B'10011110'
movlw   H'9E'
movlw   D'158'
```

Les trois instructions sont identiques, elles chargent la valeur 9Eh dans le registre W.

L'instruction `MOVWF` copie le contenu du registre W vers un emplacement de mémoire :

```
movwf   f
```

où *f* est une valeur entre 00h et 7Fh qui représente l'adresse-cible (comme nous le disions plus haut, il faudra auparavant, selon le cas, lever ou effacer le 5e bit du registre d'état, pour commuter d'une banque de mémoire à l'autre).

Si nous voulons par ex. écrire la valeur 9Eh à l'adresse 20h, la séquence des instructions pourrait être :

```
movlw   H'9E'
movwf   H'20'
```

La première instruction charge la valeur 9Eh dans le registre W et la seconde écrit le contenu du registre W à l'emplacement de mémoire 20h.

CLRF

L'instruction met le contenu d'un emplacement de mémoire à 00h (CLRF = *Clear f*).
La syntaxe de l'instruction est :

```
clrf  f
```

où *f* est une valeur entre 00h et 7Fh qui représente l'adresse-cible.

Un exemple :

```
clrf  H'20'
```

se traduit par l'écriture de la valeur 00h à l'emplacement de mémoire 20h.

Pour résumer, l'exemple suivant reprend les instructions `MOVLW`, `MOVWF` et `CLRF` :

```
movlw   H'9E'
movwf   H'20'
clrf    H'20'
```

La **figure 3** montre l'effet sur les emplacements de mémoire (au départ, nous ne connaissons pas les valeurs qu'avaient le registre W et l'emplacement de mémoire 20h).

BSF et BCF

Contrairement aux instructions précédentes qui travaillent toujours avec des octets, les instructions `BSF` et `BCF` opèrent sur des bits. **`BSF`** est l'abréviation de *Bit Set f* et, par analogie, **`BCF`** l'abréviation de *Bit Clear f* où *f* représente l'adresse d'un emplacement mémoire. La syntaxe est la suivante :

```
bsf   f, d
bcf   f, d
```

où *f* est une valeur entre 00h et 7Fh et *d* un chiffre entre 0 et 7.

Il s'agit d'instructions qui mettent à un (`bsf`) ou effacent (`bcf`) un seul bit d'un octet se trouvant à un emplacement de mémoire indiqué. Un exemple :

```
bsf   H'03', H'05'
```

Cette instruction met à *1* le bit 5 (d = 05h) de l'emplacement de mémoire 03h (f = 03h).

GOTO

Tout comme avec bien d'autres langages de programmation, `GOTO` permet d'interrompre le déroulement linéaire du programme pour le faire se poursuivre à un autre endroit – spécifié dans l'instruction.
La syntaxe de l'instruction est :

```
goto  k
```

où *k* est une adresse de la mémoire de programme. En fait, *k* peut prendre une valeur entre 000h et 7FFh, mais comme notre µC ne possède que 1 024 mots de mémoire de programme, les valeurs au-delà de 3FFh sont hors des limites.
On pourra, au lieu de *k*, également spécifier une étiquette (*label*), c'est-à-dire un

nom attribué à un emplacement particulier de la mémoire de programme. Cette étiquette doit être définie ailleurs dans le code assembleur ; il suffit de l'intercaler juste avant l'instruction à partir de laquelle doit se poursuivre le programme. Ici, il faut tenir compte de la casse !

DECFSZ

Cette instruction est une abréviation de *DECrement F and Skip if Zero* (décrémenter F et sauter si zéro) ; nous pourrions également la nommer « Instruction de boucle » (*loop instruction*). Elle est chargée de l'exécution conditionnelle de l'instruction qui suit. On commence par décrémenter (diminuer de 1) la valeur de l'emplacement de mémoire *f* pour ensuite exécuter soit l'instruction suivante (si le résultat est différent de zéro), soit sauter ladite instruction et exécuter celle qui suit immédiatement après (si le résultat est égal à zéro).
La syntaxe est la suivante :

```
decfsz f,d
```

où *f* est l'adresse d'un emplacement mémoire et *d* définit où doit être enregistré le résultat de l'opération (décrémentation). Si *d* = 0, le résultat est stocké dans le registre W et l'emplacement de mémoire n'est pas modifié ; si *d* = 1, le résultat est écrit à l'emplacement de mémoire (le registre W reste inchangé).
La **figure 4** montre comment cette instruction permet de réaliser une boucle. Dans la première ligne, nous mettons à 00h le contenu de l'emplacement de mémoire 20h. Immédiatement après on a exécution de l'instruction `DECFSZ` suivie d'un 1 – le résultat écrase le contenu de l'emplacement 20h. Lors du premier passage, nous soustrayons un 1 de 00h ; le résultat est FFh, vu que nous calculons avec des octets non signés. Ensuite, le µC regarde si le résultat est égal à 00h. Ce n'est pas le cas ici, on a donc exécution de l'instruction suivante. Notons que la décision de savoir si le résultat est égal à zéro ou non est prise sur l'examen du bit dit zéro du registre d'état. Ce bit serait à 1 si le résultat de la soustraction avait été 00h.

Comme nous le montre la figure 4, la deuxième ligne est identifiée par une étiquette *l_loop*. À la ligne suivante, `goto l_loop` permet de revenir à l'instruction `decfsz`. Ici on a à nouveau une soustraction d'un 1 et réécriture en 20h. La valeur actuelle diminue ainsi et passe à FEh, et ainsi de suite.
Nous avons donc créé une boucle exécutée 256 fois dans notre exemple. Si la valeur en 20h atteint finalement 00h, on a saut de l'instruction `goto` ; nous sortons de la boucle. À noter que nous n'avons pas utilisé le registre W ici.
On peut ainsi utiliser une boucle lorsque, par ex., on souhaite intégrer une temporisation dans le déroulement du programme (pour faire clignoter une LED par ex.).

CALL / RETURN

Tout comme nous l'avons vu avec d'autres langages de programmation, `call` sert à appeler un sous-programme et `return` à terminer le sous-programme et reprendre le programme principal.
La syntaxe de l'instruction est très proche de celle de `goto` :

```
call k
```

où *k* est ici aussi une adresse de la mémoire de programme ; elle peut aller de 000h à 3FFh.
Ici à nouveau il va de soi qu'il est possible d'attribuer une étiquette. Nous préférons donner un nom explicite à notre sous-programme, *Delay* par exemple.

`return` ne comporte pas de paramètre additionnel :

```
return
```

L'instruction `call` permet d'appeler le même sous-programme depuis différents points dans le programme (cf. **fig. 5**). Une fois le traitement terminé et le `return` exécuté, le déroulement du programme continue toujours avec l'instruction qui suit le `call` en question.

NOP

Même ça, ça existe : cette instruction ne fait vraiment rien ; on pourra l'utiliser pour une temporisation. Nous nous en servirons dans notre exemple pratique.

Définitions de constantes

En assembleur, on peut utiliser le mot-clé `EQU` pour définir des constantes afin d'améliorer la lisibilité du code et sa portabilité. Si au début du code nous convenons par ex. que :

```
STATUS   EQU  H'0003'
```

Nous n'avons plus besoin, plus loin dans le code, d'indiquer explicitement l'emplacement auquel peut être trouvé le registre d'état et écrire par exemple :

```
bsf   STATUS,H'05'
```

au lieu de

```
bsf   H'0003',H'05'
```

pour mettre à un le 5e bit du registre d'état. Ne faisons pas les choses à moitié et convenons aussi que

```
RP0   EQU  H'05'
```

Nous pouvons dès lors, par

```
bsf   STATUS,RP0
```

et

```
bcf   STATUS,RP0
```

mettre à un et effacer le 5e bit du registre d'état et ainsi commuter entre les banques de mémoire.
Le code se laisse également plus facilement porter d'un type de µC à un autre (passage par ex. d'un petit PIC à un PIC plus grand). Le registre d'état peut, avec un autre type de µC, se trouver à un autre emplacement de mémoire ; il nous suffit dans ce cas de changer la définition des constantes, le reste du code est conservé.

__CONFIG

Cette commande (appelée directive) n'est pas, à l'inverse des instructions mentionnées précédemment, convertie en code machine exécuté lors du déroulement du programme. Elle est utilisée pour fixer le contenu du registre Config. Le registre de configuration est l'un des (très rares) registres, dans lequel on ne peut pas écrire à une adresse de la mémoire. La définition de la valeur du registre de configuration se fait dès le flashage du µC.

En général, nous retrouvons la directive `__CONFIG` au début du code source, par exemple :

```
__CONFIG        B'00000110000100'
```

Ces bits permettent de paramétrer l'utilisation de l'oscillateur interne sans quartz externe, la non-activation de la protec-

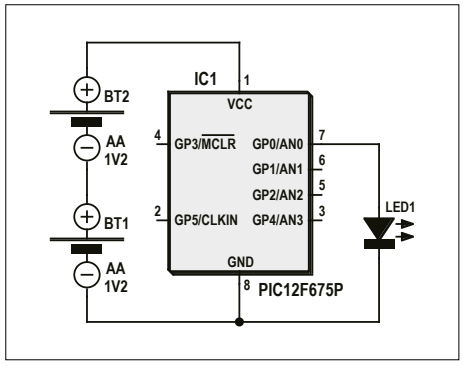


Figure 6. Difficile de faire plus simple pour obtenir le clignotement d'une LED.

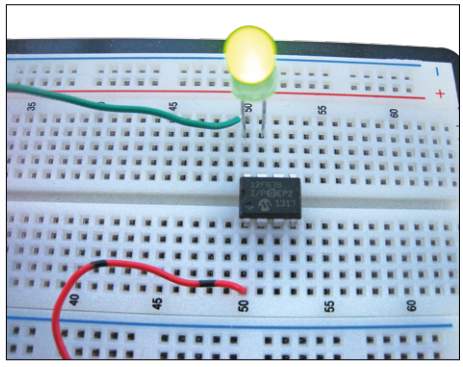


Figure 7. Le circuit peut être réalisé sur une plaque d'essais.

tion des données ou de la mémoire de programme, la désactivation de la fonction de chien de garde (*watchdog*), etc. Nous retrouvons, comme d'habitude, la description de chacun de ces bits dans la feuille de caractéristiques, souvent au chapitre « *Special Features of the CPU / Configuration Bits* » (les fiches en français sont rarissimes).

Commentaire

Un commentaire commence toujours par un point-virgule (;). Tout ce qui suit le point-virgule (jusqu'à la fin de ligne - ***End of Line***) est ignoré par le compilateur et considéré comme commentaire.
Si la ligne commence par un point-virgule, toute la ligne est traitée comme un commentaire.

Application « Hello World »

Nous savons maintenant tout ce qu'il faut pour créer une application simple. Allons-y.
Notre tâche est (relativement) facile. Nous allons, pour commencer, connecter une LED à la broche GP0 et la faire clignoter.
En termes de matériel, la solution est d'une mise en œuvre ultrasimple. Nous utiliserons l'oscillateur d'horloge interne (4 MHz) et n'aurons pas besoin de quartz externe. La LED sera connectée directement à GP0 (même sans résistance) – le µC limite lui-même le courant de sortie à environ 20 mA (alimentation de 5 V) par le biais de la résistance interne du port (**fig. 6**).
Le circuit peut être réalisé sur une plaque d'essais (**fig. 7**). La source d'alimentation pourra par ex. être constituée de deux piles de 1,2 V ; mais aussi n'importe quelle autre source qui peut fournir au moins 20 mA sous 2 à 5 V.

Code source

À quoi doit ressembler le programme en assembleur ? Que faut-il faire au minimum pour écrire un programme exécutable ?
Comme nous le disions plus haut, nous pouvons utiliser pour notre exemple un simple éditeur de texte tel que Notepad, y saisir le code et enregistrer le fichier. L'extension doit toujours être *.asm* (il faudra peut-être, après enregistrement, la modifier manuellement). Nous utiliserons comme nom de fichier : *01_LED_v_1p03.asm*. Vous pouvez télécharger un fichier tout fait sur la page de projet de cet article [3].

La **figure 8** reprend le début du code. Pour augmenter la lisibilité du programme et le rendre plus facile à comprendre par d'autres utilisateurs, nous l'avons doté de nombreux commentaires. Nous voulons de plus travailler avec des constantes. Des définitions telles que

```
STATUS EQU  H'03'
```

rendront le code plus portable si nous voulons plus tard nous essayer à un PIC un peu plus grand.
Heureusement il est inutile de réécrire à chaque fois ces définitions de constantes dans notre code. On retrouve, dans le fichier *P12F675.INC*, qui fait partie de la Suite MPASM, toutes les définitions essentielles (adresses de registres, positions de bits, etc.) pour notre type de µC.
Nous incluons ce fichier tout au début

```
;*****************************************************************************
;*                        Blinking LED                                       *
;*                      v 1.03 - 09.03.2015                                  *
;*****************************************************************************
;*         Hardw.: PIC12F675 used                                            *
;*         OSC.....: Internal osc. used     POWER.....:  5V                  *
;*****************************************************************************

;-------------------------------------------------------
;Pins connection:
;----------------------
;GP0 --> LED output
;GP1 --> N/C
;GP2 --> N/C
;GP3 --> N/C
;GP4 --> N/C
;GP5 --> N/C
;-------------------------------------------------------

 INCLUDE "P12F675.INC"

_U002                                   EQU     B'00000110000100'               ;MCLR = input
              __CONFIG        _U002

;=============================================================================
;       Variable Definition
;=============================================================================
TIMER1                  EQU             H'20'                   ;Used in delay routine
TIMER2                  EQU             H'21'                   ; "             "               "

;=============================================================================
```

Figure 8. Les commentaires sont importants, même en assembleur.

```
;*****************************************************************************
;
;       Register Definitions
;
;*****************************************************************************
;...
;-----Bank0-----------------
INDF            EQU  H'0000'
TMR0            EQU  H'0001'
PCL             EQU  H'0002'
STATUS          EQU  H'0003'
FSR             EQU  H'0004'
GPIO            EQU  H'0005'
PCLATH          EQU  H'000A'
INTCON          EQU  H'000B'
PIR1            EQU  H'000C'
TMR1            EQU  H'000E'
TMR1L           EQU  H'000E'
TMR1H           EQU  H'000F'
T1CON           EQU  H'0010'
CMCON           EQU  H'0019'
ADRESH          EQU  H'001E'
ADCON0          EQU  H'001F'
;-----Bank1-----------------
OPTION_REG      EQU  H'0081'
TRISIO          EQU  H'0085'
PIB1            EQU  H'008C'
PCON            EQU  H'008E'
OSCCAL          EQU  H'0090'
WPU             EQU  H'0095'
;...
;...
```

Figure 9. Définitions des registres dans le fichier *Include*.

Liens

[1] www.microchip.com/wwwproducts/Devices.aspx?product=PIC12F675

[2] www.microchip.com/pagehandler/en-us/family/mplabx

[3] www.elektor-magazine.fr/130483

de notre code par une directive INCLUDE. Cela fonctionne exactement comme avec tous les autres langages de programmation « classiques » ; le résultat est le même que si nous avions tout simplement copié, par un copier-coller, le code du fichier à inclure dans notre programme. La **figure 9** nous montre le contenu du fichier comportant les définitions de constantes typiques au µC.

La figure 8 nous apprend que nous avons en outre défini deux variables de type octet (c.-à-d. des emplacements de mémoire en SRAM), à savoir *TIMER1* et *TIMER2*. Nous les utiliserons dans notre boucle de temporisation.

La **figure 10** montre le reste du programme. Nous n'y utilisons que des instructions dont il a été question plus haut. L'illustration montre les différents registres avec tous leurs bits. Un point d'interrogation noir à la place du bit signifie que sa valeur est sans importance ici. Dans la boucle *Main* on voit un petit rond rouge ; il signale le moment d'allumage de la LED. Le rond gris identifie l'instruction qui éteint la LED. Les flèches représentent les sauts.

Le petit sous-programme *delay_r* introduit une temporisation. Vous voyez peut-être immédiatement comment cela fonctionne ? Note : on se trouve en présence de deux boucles imbriquées.
Tout à la fin du fichier on doit impérativement trouver la directive END.

Assembler

Une fois le programme saisi, nous pouvons assembler le code. Pour cela, nous appelons le programme MPASM.EXE (partie de l'IDE MPLAB).
Après le démarrage du programme (**fig. 11**), nous sélectionnons notre fichier texte (dans le champ *Source File Name*). Dans le champ *Processor*, il faut sélectionner le type de notre µC. Le reste peut être laissé avec les valeurs par défaut. Presser sur le bouton *Assemble* déclenche la compilation du fichier ASM.
Ce processus produit alors un fichier hex (et quelques autres fichiers que nous pouvons ignorer pour l'instant).

Le PICkit2 est accompagné du programme PICkit2V2.exe. Avant le lancement du programme, le kit doit être connecté à l'ordinateur via USB et le µC doit se trouver dans le support de programmation de la carte d'expérimentation. Lors du démarrage de PICkit2V2.exe, il y a détection automatique du µC (cf. **fig. 12**).
Ensuite, le point du menu *File – Import Hex* permet de charger le fichier hex et enfin le bouton *Write* de le flasher dans le µC.

C'est tout pour aujourd'hui – j'espère que vous avez apprécié vos premiers pas en assembleur. Dans le prochain article nous passerons en revue l'ensemble du jeu d'instructions du PIC12F675. Notre exercice pratique se matérialisera sous la forme d'un dé électronique.

(130483 – version française : Guy Raedersdorf)

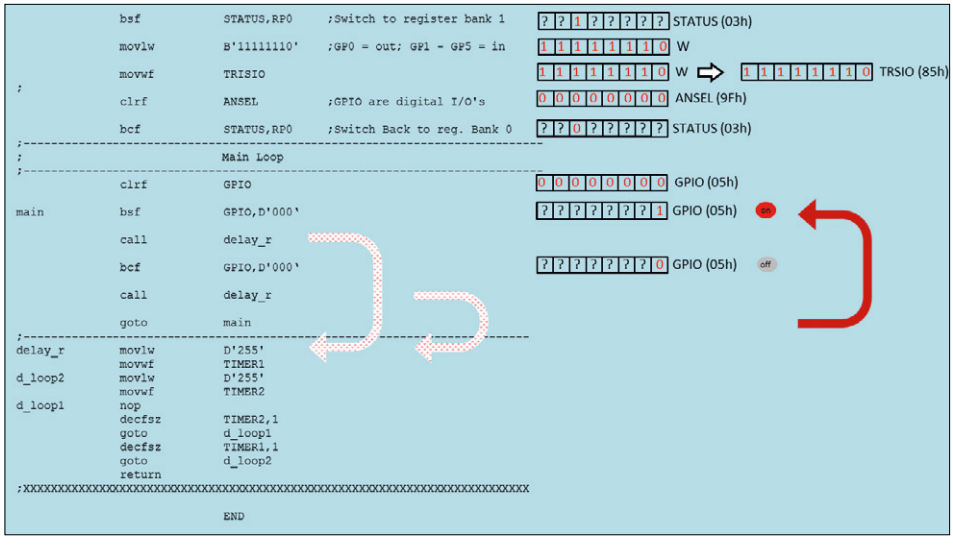


Figure 10. Pour la temporisation, nous utilisons un sous-programme.

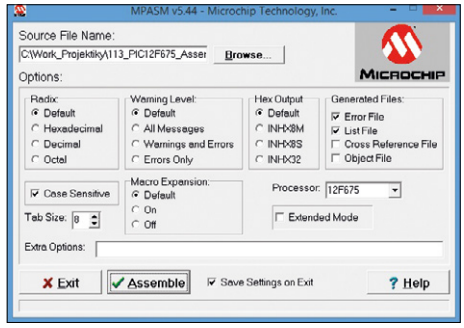


Figure 11. Capture d'écran de l'assembleur MPASM qui convertit le code en fichier Hex.

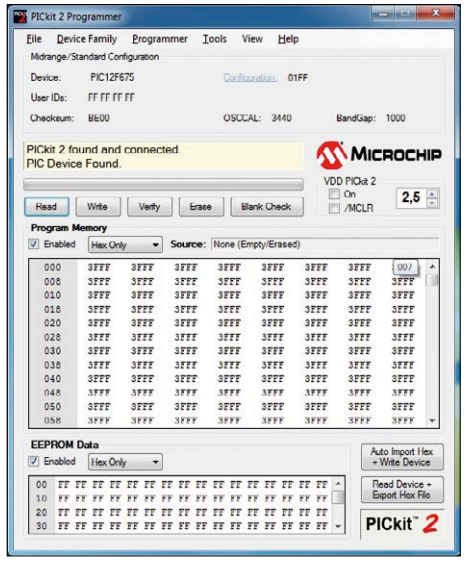


Figure 12. Le logiciel de programmation du PICkit nous permet de flasher le programme dans le µC.

trucs & astuces pour le logiciel DesignSpark Mechanical/CAD

3e partie : ajouter une liste de composants

Neil Gruending (Canada)

Apprenez à ajouter une liste de composants à un modèle DesignSpark Mechanical 3D.

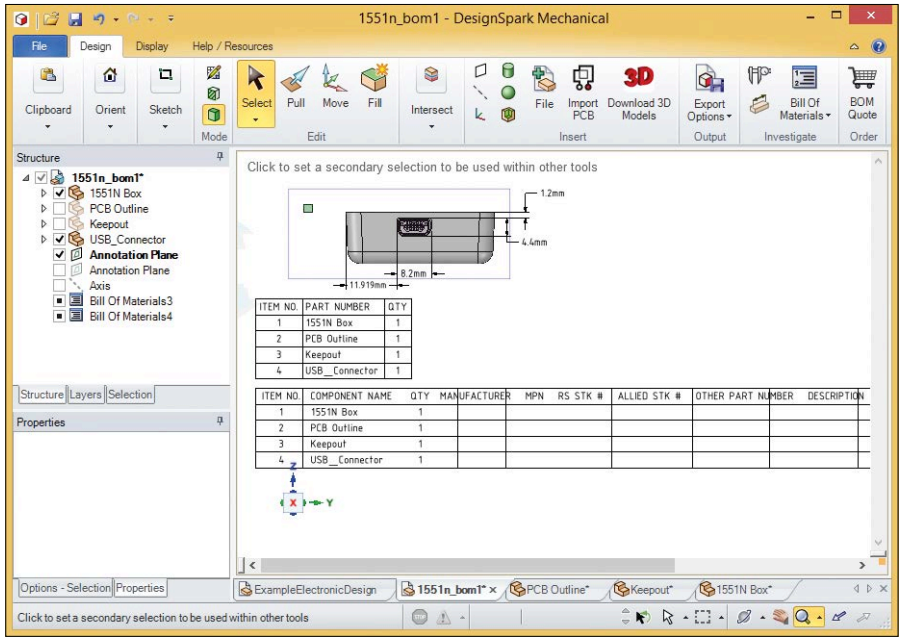


Figure 1. Exemples de BOM.

Il est facile d'ajouter une liste de composants ou BOM (*Bill of Materials*) à un modèle DesignSpark Mechanical. Ajoutons-en une à l'exemple que nous avions complété par des cotes la fois dernière.

Ajouter la liste de composants

Tout d'abord il faut un plan sur lequel ajouter la BOM : c'est un objet à deux dimensions. Ce peut être n'importe quel plan dans le projet, mais on choisit habituellement un plan d'annotation. Dans notre exemple, ce sera le plan utilisé pour les cotes, mais vous pourriez en créer un nouveau avec l'outil *Dimension* ou l'outil *Plane*. Ensuite ajoutez une BOM *Full Detail* ou *Top Level* à l'aide de l'outil *Bill of Materials* du menu *Investigate* (**fig. 1**).

Le tableau du dessus, une BOM *Top Level*, contient tous les composants du projet ainsi que leur nom dans la colonne *Part Number*. Le tableau du dessous est une BOM *Full Detail*, habituellement la plus utile des deux. Elle contient tous les composants du projet avec leur référence chez le distributeur, RS Components ou Allied Electronics. Comme vous le voyez, il faudra ajouter ces informations pour que la BOM détaillée soit utile. Si vous téléchargez les composants depuis RS, elles seront déjà présentes.

Ajouter les informations du fabricant

L'outil BOM de DesignSpark utilise des propriétés spéciales pour stocker les informations du fabricant sur un composant. Remplir tous les champs à la main à chaque fois que vous dessinez un composant peut être fastidieux, c'est pourquoi DS possède l'outil *Edit Ordering Information* (**fig. 2**) qui peut le faire pour vous. Il suffit de cliquer avec le bouton droit sur le composant et de choisir *Edit Ordering Information* dans le menu.

Lorsque toutes les informations ont été ajoutées, cliquez sur *Save* pour insérer les champs de BOM dans les propriétés du composant. La **figure 3** montre notre exemple après l'ajout des champs de BOM au composant du boîtier Hammond 1551. Le tableau de BOM a été mis

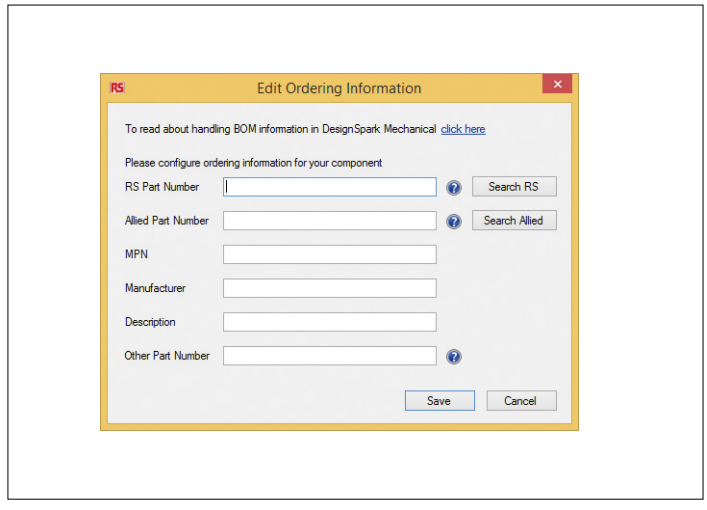


Figure 2. Fenêtre *Edit Ordering Information*.

à jour automatiquement : le coin inférieur gauche montre les champs ajoutés au composant. Si vous avez besoin d'éditer ces informations, utilisez l'outil *Edit Manufacturing Information* à nouveau ou éditez directement les champs dans la fenêtre de propriétés du composant.

Vous pouvez choisir de faire apparaître ou non un composant dans la BOM. Dans notre cas, ça sera utile pour cacher le composant nommé *keepout* de la BOM (sa présence dans le modèle reste justifiée). Commencez par cliquer avec le bouton droit sur le composant puis choisissez *Bill of Materials* pour lui ajouter la propriété @BomInclude. Changez maintenant sa valeur de *True* à *False* et il disparaîtra de la BOM (**fig. 4**).

Export en CSV

Une fois la BOM achevée, il est facile de l'exporter au format *Comma Separated Value* (CSV) pour l'ouvrir dans un tableur tel que Microsoft Excel. Cliquez d'abord sur le tableau de la BOM pour le sélectionner, DS dessinera un cadre en pointillé autour du tableau. Maintenant choisissez l'option d'export en CSV depuis le menu *Bill of Materials* et DS exporte la BOM dans un fichier .csv situé dans le dossier du projet, au même niveau que le fichier de conception .rsdoc. DS ouvrira également le programme par défaut associé aux fichiers CSV.

Devis de BOM en ligne

DesignSpark Mechanical possède également un outil de devis en ligne qui téléversera votre BOM sur le site de RS Components ou Allied Electronics, ce qui facilitera la commande de composants pour votre projet. Il suffit de choisir l'option *BOM Quote* depuis le menu *Order*. L'outil de devis produira alors sa propre BOM pour le téléversement et vous n'aurez pas à sélectionner un tableau de BOM avant d'exécuter l'outil. La **figure 5** montre la BOM téléversée pour notre exemple.

Le site, celui d'Allied Electronics puisque je suis canadien (www.alliedelec.com ; Allied est la branche US/canadienne/mexicaine de RS Components, *Ed.*), essaiera ensuite de faire correspondre les références de composants de votre projet aux composants disponibles. Ici, il a trouvé le boîtier Hammond pour lequel j'avais saisi les références RS et Allied. Vous n'aurez toutefois pas toujours le temps de saisir les références au préalable. Par exemple, j'ai seulement saisi le fabricant (Molex) et le préfixe de la référence (54819) pour le connecteur USB. DS vous avertira si vous ne saisissez pas de référence RS Components ou Allied, mais vous pourrez ignorer l'avertissement sans dommages. Le site d'Allied m'a ensuite proposé une liste de correspondances possibles pour la référence Molex dans laquelle j'ai choisi la référence complète.

Conclusion

DesignSpark Mechanical possède des outils de BOM très faciles à utiliser qui font gagner du temps. L'exemple proposé ici est très simple, mais la vraie puissance des outils de BOM s'exprime dans les projets plus gros et plus complexes. Essayez !

(150294 – version française : Kévin Petit)

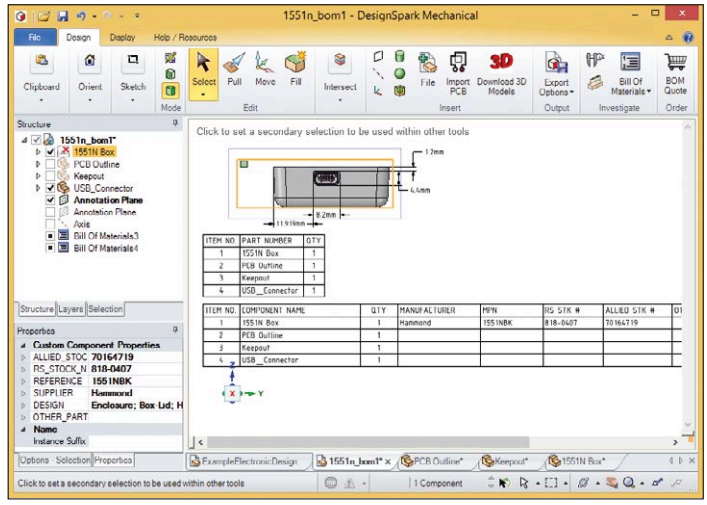


Figure 3. Informations du boîtier mises à jour.

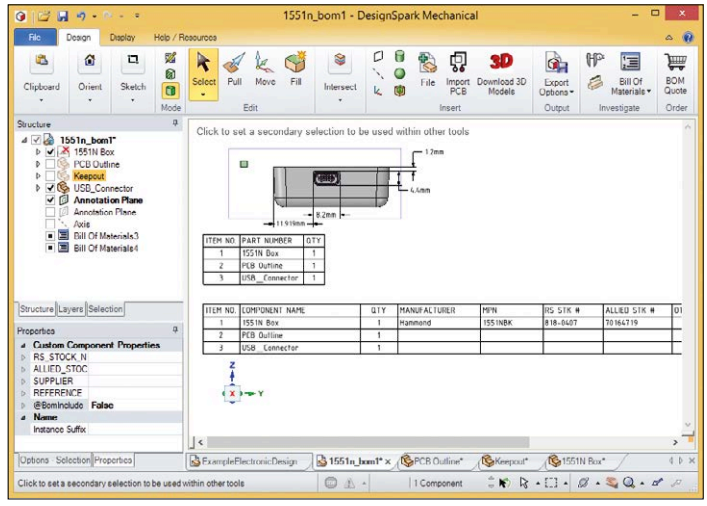


Figure 4. Exemples de BOM.

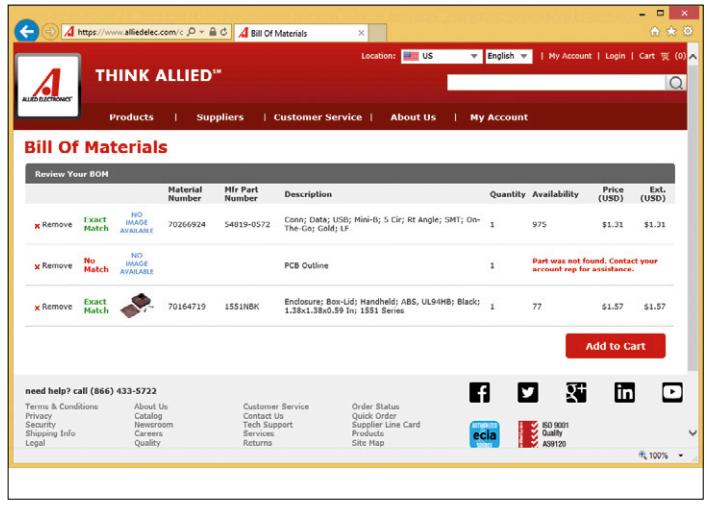


Figure 5. Outil de devis en ligne.

interface SPI

µC ARM pour néophytes pour passer de 8 bits à 32 bits 6e partie

Viacheslav Gromov (Allemagne)

Comme décrit précédemment, le SERCOM de notre SAM D20 peut se convertir en interface U(S)ART ou I²C. Cependant avant de maîtriser complètement le SERCOM, il reste à découvrir une autre interface : SPI. Nous réaliserons ici un projet plus ambitieux qui inclut, outre l'interface SPI, d'autres éléments périphériques déjà rencontrés.

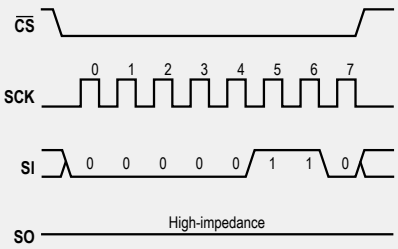
Dans l'épisode précédent, nous avons testé le CAN avec un capteur de température LM335. Nous allons réutiliser ce capteur dans un projet plus ambitieux avec interface SPI : un enregistreur de données de température. Il nous faut pour cela une mémoire EEPROM à interface SPI, issue d'une famille connue (25LC ou 25AA). Ces mémoires EEPROM de diverses capacités sont faciles à trouver. Avec les EEPROM il y a peu de risque de se fourvoyer ; en réalité seuls de très gros projets requièrent de la mémoire externe. Il nous faut en plus une minuterie précise, de préférence une version capable de mémoriser des heures et des minutes. N'avons-nous pas déjà utilisé une horloge en temps réel (RTC) en mode Calendrier ?

Le SERCOM en interface SPI

Lorsque le SERCOM du SAM D20 devient une interface SPI, sa structure rappelle celle de l'interface I²C. Le synoptique de la **figure 1** montre le SERCOM, à gauche, configuré en SPI maître et à droite en SPI esclave. Ces interfaces ont un registre à décalage (*shift*) pour l'émission et la réception immédiate des données ainsi qu'un registre TxDATA et RxDATA dans lesquels, selon le cas, la CPU écrit les données à envoyer ou peut y lire l'octet reçu. Chaque registre RxDATA est en outre doté d'un tampon ce qui permet un fonctionnement souple.
Le maître doit de plus produire le signal d'horloge du bus, fonction toujours assurée, avec le SAM D20, par le générateur de taux de transmission. La fréquence de l'horloge se règle dans le registre BAUD. En configuration SPI esclave, on a en outre comparaison de l'adresse reçue (s'il y en a une) avec l'ADDR/ADDRMASK de façon à ce que le contrôleur se sente aussi, le cas échéant, adressé. On retrouve, au milieu du synoptique, les quatre lignes SPI.
Il est bien sûr possible de jouer sur de nombreux paramètres (octet de poids fort ou faible en premier, par ex.). En esclave, le SERCOM peut même continuer à travailler en sommeil (quelle vie d'esclave !). Comme d'habitude avec le SERCOM, on peut multiplexer les broches à loisir et choisir une source d'horloge

Le protocole SPI

SPI est l'acronyme de ***S**erial **P**eripheral **I**nterface ;* comme le célèbre bus I²C, c'est un bus de données classique à protocole simple [5]. SPI utilise, à l'inverse de l'I²C, des lignes séparées pour les données émises ou reçues par le maître. Il n'y a ni adressage de l'esclave, ni bits de départ et d'arrêt, cela permet une vitesse de transfert élevée.



source : Microchip

C'est pourquoi les cartes SD actuelles et la plupart des programmateurs ISP supportent cette interface. En outre, il est possible, avec ce bus de données, de connecter sur un bus presque autant d'esclaves à un maître que l'on veut. L'interface SPI comprend les lignes suivantes :

Master Out Slave In (MOSI) : données maître -> esclave
Master In Slave Out (MISO) : données esclave -> maître
Serial Clock (SCK) : signal d'horloge de synchronisation du bus de données délivré par le maître.
Slave Select (SS) : chaque esclave possède sa propre ligne SS vers le maître. Si le maître la met au niveau bas, l'esclave sait que le maître s'adresse à lui.

À l'inverse du bus I²C, on se passe de résistances de polarisation haute ; cependant, en cas de système à plusieurs niveaux de tension (5 et 3,3 V), cela peut poser des problèmes.

Avec la SPI, les relations entre les niveaux des données et le signal d'horloge revêtent une importance capitale. La polarité de l'horloge (CPOL) ainsi que la phase du signal d'horloge par rapport au signal de données (lecture sur flanc descendant ou montant, CPHA) jouent un rôle. Les combinaisons de ces paramètres donnent quatre modes différents (0 à 3). Il est en outre possible de transférer les données avec l'octet de poids faible (LSB) ou fort (MSB) en premier. Ce paramétrage varie d'un circuit à un autre ; un coup d'œil aux caractéristiques s'impose. L'illustration montre un transfert de données SPI typique ; ici envoi de l'instruction WREN à la 25LC040.

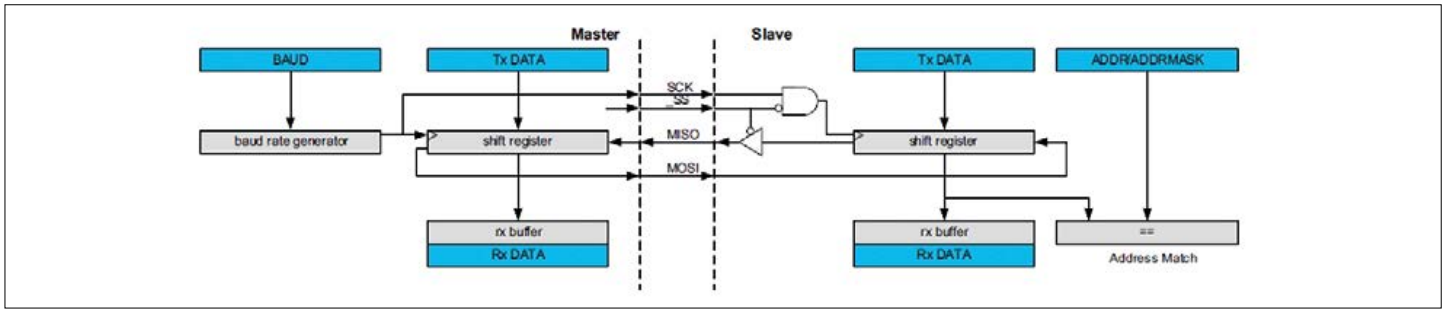


Figure 1. La structure de principe du SERCOM en tant qu'interface SPI (mode Esclave/Maître ; diagrammes et captures d'écran : Atmel).

quelconque. Plus d'infos à ce sujet à la page 369 de la feuille de caractéristiques [1].

L'enregistreur de données de température

Il est temps de réfléchir aux composants et éléments périphériques à utiliser et de penser à la réalisation globale du circuit. Comme la feuille de caractéristiques indique les éléments périphériques et leurs broches, on sait où connecter les composants externes. Il existe bien sûr diverses options de connexion, la MCU possède plusieurs périphériques identiques (SERCOM 1 à 4 par ex.) qui peuvent être multiplexés de différentes façons. Au début, utilisez les broches dans la configuration standard. Nous pouvons dessiner le circuit. Pour notre projet, nous avons raccordé à la carte le LM335 du dernier numéro et une EEPROM 25LC040 ; le câblage est minimal (**fig. 2**). La connexion du circuit à la plaque d'essai à l'aide de câbles pour Raspberry Pi (**fig. 3**) est rapide. Règle à respecter : utilisez des câbles les plus courts possible.

Pour le logiciel, on commence par créer un projet vide dans lequel on inclut les bibliothèques ASF requises, extraites de l'Assistant ASF. Plutôt en inclure trop que pas assez ; des bibliothèques de base telles que *Delay* ou *USART* sont souvent utilisées pendant le développement d'un projet, au débogage en particulier. Plus tard, on supprimera les bibliothèques inutiles. Il nous a fallu ici (**fig. 4**) inclure de nombreuses bibliothèques, les plus importantes sont celles des U(S)ART, SPI et RTC. Hormis celle de la RTC, toutes les bibliothèques sont rattachées dans leur version avec interrogation (*polled*) ; elles ne sont pas soumises aux interruptions, cela permet une meilleure lisibilité du fichier *Main*.

Les fonctions de configuration des UART, CAN, RTC, et SPI sont (comme souvent) recopiées de projets antérieurs ; pour la RTC, nous avons aussi repris l'ISR de type *Callback*. Comme l'EEPROM utilise déjà la broche PA04 (AREFB), nous devons, par rapport au projet précédent, passer la broche de référence du CAN sur GPIO PA03 (AREFA) et la configurer de manière adéquate.

Du nouveau ici : la fonction de configuration du SERCOM en tant que SPI maître ; regardons-la de plus près (**listage 1**) [3]. Première étape : création de deux structures de configuration appelées `config_spi_master` et `slave_dev_config`, cette dernière est dotée de son préparamétrage. Ensuite, l'instruction

```
slave_dev_config.ss_pin = PIN_PA05
```

permet d'attribuer la broche SPI-SS du SERCOM à la broche PA05 de la MCU. Puis, on transfère au SERCOM les paramètres

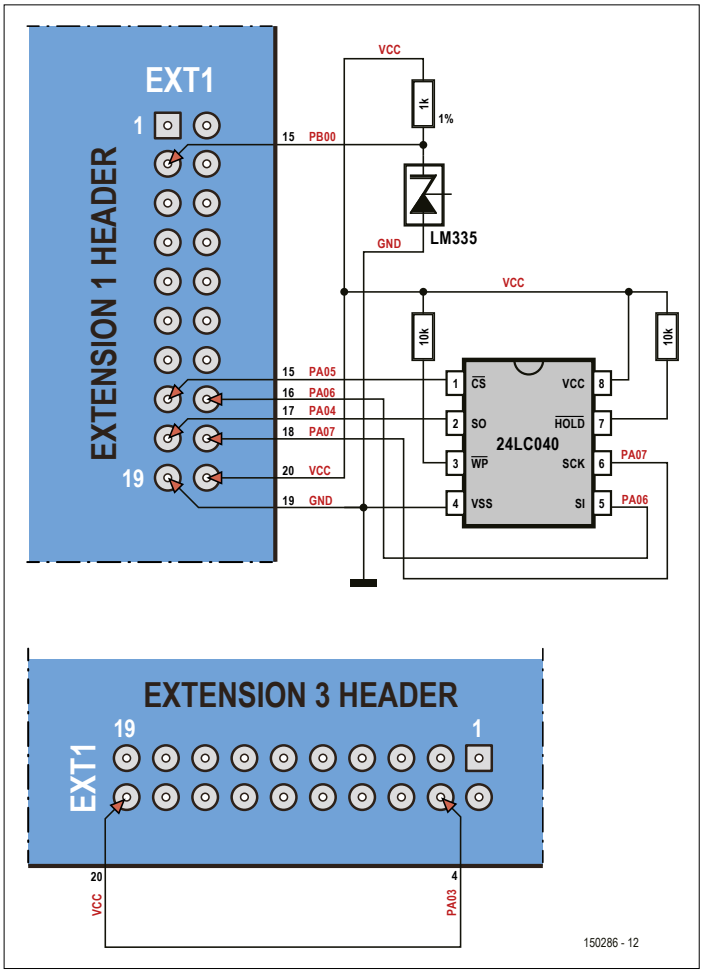


Figure 2. Le circuit simple de l'enregistreur de données de température.

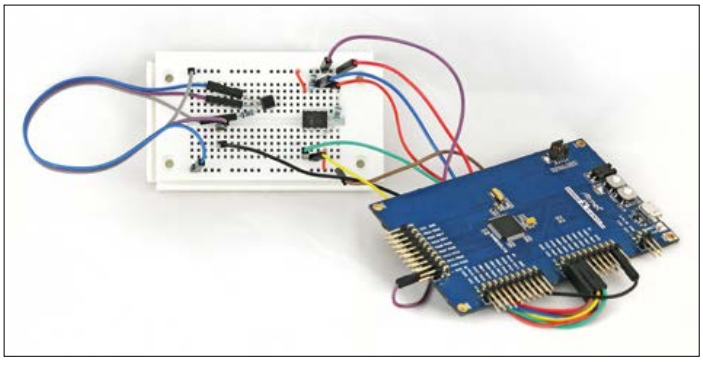


Figure 3. Voici à quoi ressemble la version sur plaque d'essais de l'auteur.

Listage 1. Fonction pour la configuration du SERCOM en tant que maître

```
void configure_spi_master(void)
{
  struct spi_config config_spi_master;
  struct spi_slave_inst_config slave_dev_config;
  spi_slave_inst_get_config_defaults(&slave_dev_config);
  slave_dev_config.ss_pin = PIN_PA05;
  spi_attach_slave(&slave, &slave_dev_config);
  spi_get_config_defaults(&config_spi_master);
  config_spi_master.mux_setting = EXT1_SPI_SERCOM_MUX_SETTING;
  config_spi_master.pinmux_pad0 = EXT1_SPI_SERCOM_PINMUX_PAD0;
  config_spi_master.pinmux_pad1 = PINMUX_UNUSED;
  config_spi_master.pinmux_pad2 = EXT1_SPI_SERCOM_PINMUX_PAD2;
  config_spi_master.pinmux_pad3 = EXT1_SPI_SERCOM_PINMUX_PAD3;
  config_spi_master.transfer_mode = SPI_TRANSFER_MODE_0;
  config_spi_master.data_order = SPI_DATA_ORDER_MSB;
  config_spi_master.mode_specific.master.baudrate = 250000;
  spi_init(&spi_master_instance, EXT1_SPI_MODULE, &config_spi_master);
  spi_enable(&spi_master_instance);
}
```

Listage 2. La fonction *Callback* relativement longue de la RTC en mode Calendrier

```
void rtc_match0_callback(void)
{
  adc_start_conversion(&adc_instance);
  while(adc_read(&adc_instance, &data) == STATUS_BUSY){}
  volt = data * 0.000805;
  measure_result = 25 + (volt - 2.945) / 0.01;
  rtc_calendar_get_time(&rtc_instance, &time);
  spi_select_slave(&spi_master_instance, &slave, true);
  spi_write_buffer_wait(&spi_master_instance, &write_enable, 1);
  spi_select_slave(&spi_master_instance, &slave, false);
  delay_ms(10);
  spi_select_slave(&spi_master_instance, &slave, true);
  spi_write_buffer_wait(&spi_master_instance, &write_command, 1);
  spi_write_buffer_wait(&spi_master_instance, &adress[i], 1);
  spi_write_buffer_wait(&spi_master_instance, &time.hour, 1);
  spi_write_buffer_wait(&spi_master_instance, &time.minute, 1);
  spi_write_buffer_wait(&spi_master_instance, &time.second, 1);
  spi_write_buffer_wait(&spi_master_instance, &measure_result, 1);
  spi_select_slave(&spi_master_instance, &slave, false);
  i++;
  alarm.mask = RTC_CALENDAR_ALARM_MASK_SEC;
  alarm.time.second += 10;
  alarm.time.second = alarm.time.second % 60;

  if(i < 10)
  {
    rtc_calendar_set_alarm(&rtc_instance, &alarm, RTC_CALENDAR_ALARM_0);
  }
  else
  {
    measure_ended = 1;
    port_pin_set_output_level(LED0_PIN, 0);
  }
}
```

chargés dans `slave_dev_config` par l'instruction

```
spi_attach_slave(&slave, &slave_dev_config).
```

Cette procédure devra être exécutée plusieurs fois s'il y a plusieurs esclaves et donc plusieurs broches SS.
Au tour maintenant de `config_spi_master`. Cette structure est dotée elle aussi d'un préparamétrage. On lui attribue les broches auxquelles sont connectés les circuits intégrés. Ensuite

```
config_spi_master.transfer_mode = SPI_TRANSFER_MODE_0
```

permet de sélectionner le mode SPI 0, avec

```
config_spi_master.data_order = SPI_DATA_ORDER_MSB
```

« MSB first », et avec

```
config_spi_master.mode_specific.master.baudrate =
250000
```

une fréquence d'horloge de 250 kHz. Pour finir, les paramètres définis dans cette structure sont transmis au SERCOM 0 (qui se cache derrière `EXT1_SPI_MODULE`) et l'élément périphérique est activé.
En outre nous avons déclaré dans le code différentes variables et divers tableaux. Dans ces derniers, nous déposons par ex. les octets d'instruction requis par l'EEPROM SPI (cf. **Encadré**) ainsi que les dix adresses dans le tableau `adress`. Nous y reviendrons.

La phase de développement est une période d'intenses modifications. D'où la déclaration de plusieurs tableaux, à une seule variable au départ, mais qui peuvent plus tard, s'en voir ajouter d'autres. La modification du programme en sera facilitée.
Les modules majeurs du programme, dont on retrouve l'organigramme en **figure 5**, sont la fonction *Main* et l'ISR du compteur RTC. En premier, on écrit par précaution dans le registre de commande de la 25LC0040 de manière à avoir accès à toute la mémoire. Parallèlement on configure les autres éléments périphériques du SAM D20 et on met, provisoirement, le tampon dateur de la RTC à *12.10.2015 13:45:34*.
Le µC commence alors, par le CAN, la mesure de la température toutes les 10 s et enregistre dans quatre registres, dans cet ordre, le temps sous forme d'heures (1), de minutes (2) et de secondes (3) et la température (4). On saisit le pourquoi du quadruple aiguillage d'adresses du tableau `adress`. Le tout se passe dans l'ISR de type *Callback* de la RTC (**listage 2**) appelée à chaque alarme – à compter de *13:45:35* puis toutes les 10 s. Là, la variable *i* est incrémentée à chaque passage de 0 à 9 jusqu'à ce que les mesures soient terminées. D'où une période de mesure de 10 s x 10 passages = 100 s.
Ensuite la LED s'allume en raison de l'interrogation *If* à la fin de l'ISR. L'alarme suivante (et l'ISR) ne sont plus activées et `measure_ended` est mis à 1. Il s'agit là d'une sorte d'indicateur (*flag*) logiciel qui permet la lecture de la mémoire dans la boucle sans fin. Ainsi, l'allumage de la LED0 de la carte signale la fin de l'enregistrement des données. Les valeurs de température enregistrées peuvent ensuite être transmises à l'interface UART, reliée à l'ordinateur par le biais de l'EDBG, aussi souvent que

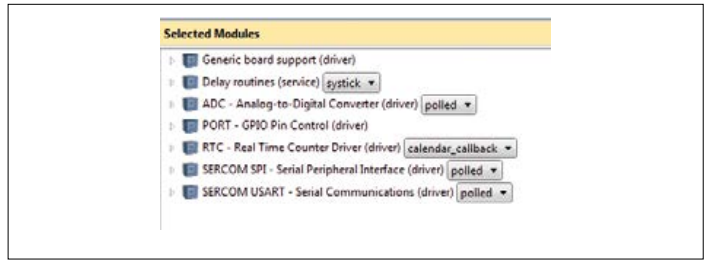


Figure 4. Ces bibliothèques ont été incluses dans notre projet. Dans l'Assistant ASF, la même bibliothèque se charge de l'esclave SPI et du maître SPI.

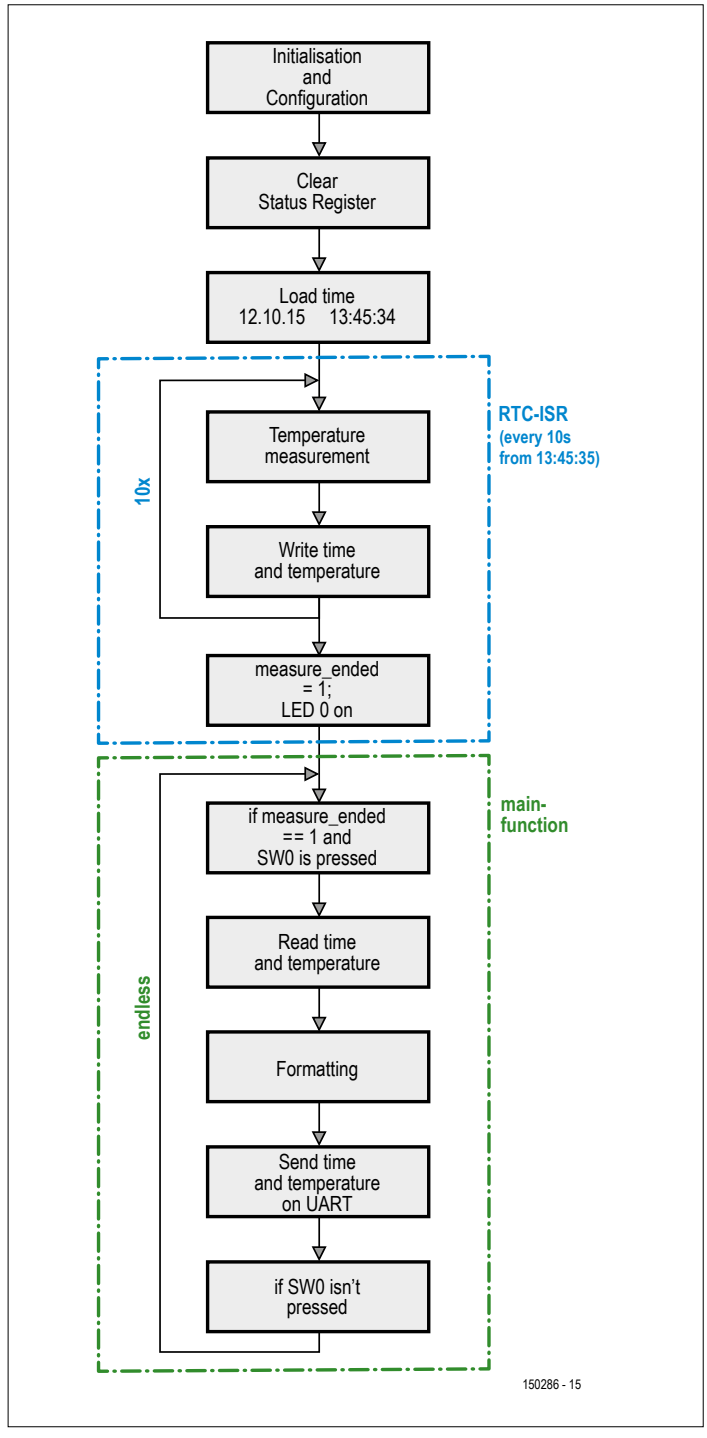


Figure 5. Ordinogramme du programme.

l'on veut. Pour cela, on appuie sur le bouton SW0 de la carte, action qui satisfait ainsi à la condition de l'interrogation *If* de la boucle sans fin (**listage 3**).

Dans la boucle *For* qui suit, la variable *i* est à nouveau incrémentée (de 0 à 9). À chaque incrémentation, on a lecture, dans quatre registres consécutifs de l'EEPROM, des données enregistrées puis transmission par le biais de l'U(S)ART. Il reste cependant à mettre les valeurs brutes au bon format avant leur envoi :

```
sprintf(usart_buffer_time, "Time: %d:%d:%d\n", read_
data[0], read_data[1], read_data[2]);

sprintf(usart_buffer_temperature, "Temperature: %d
degree\n", read_data[3]);
```

Comme on le voit, les valeurs lues se trouvent dans le tableau `read_data`.

Il n'y a plus, après la boucle *For*, qu'une petite boucle *While*

```
while(!port_pin_get_input_level(BUTTON_0_PIN)){}
```

qui ne permet l'envoi des données mémorisées suivantes, horodatage compris, qu'après relâchement de SW0. Cette mesure préventive évite, lors d'une action sur la touche, une relecture multiple des dix valeurs.

Nous n'avons pas encore vu le détail des instructions SPI, évidentes elles. Le programme complet ne connaît que deux types d'instructions :

```
spi_write_buffer_wait(&spi_master_instance,
&adress[i], 1);
```

envoie un tampon de longueur quelconque via la SPI. Tout ce qu'il nous faut est un pointeur vers la structure SPI, un pointeur vers le tableau de données et en dernier le nombre d'octets à envoyer.

```
spi_read_buffer_wait(&spi_master_instance, &read_data,
4, 0x00);
```

lit un tampon de longueur quelconque par le biais de l'interface SPI. Il nous faut encore un pointeur vers la structure SPI, un pointeur vers le tableau de données, le nombre d'octets et pour finir un « octet factice » (*dummy byte*). La valeur de l'octet factice est sans importance, il est envoyé parallèlement à l'opération de lecture pour que l'esclave sache exactement quand envoyer l'octet suivant.

Ces instructions sont généralement séparées par des pauses de 10 ms pour éviter de passer en deçà du temps d'écriture requis par l'EEPROM.
Au début et à la fin de chaque transfert de données il faut, comme le requiert le protocole SPI, mettre, avec l'instruction

```
spi_select_slave(&spi_master_instance, &slave, x); //
x = true/false
```

la ligne SS au niveau haut ou bas selon le cas.
Vous pouvez télécharger depuis [2] ce projet appelé « Temperature-Datalogger » et l'essayer. Pour cela, il vous suffit de démarrer un programme de terminal, de réinitialiser le contrôleur, d'attendre 100 s jusqu'à ce que la LED0 s'allume et donc que les mesures de température soient terminées, d'appuyer ensuite sur SW0 et de lire les dix valeurs de température enregistrées avec horodatage.

Il y a toujours moyen de peaufiner...

Avez-vous noté que les 215 lignes de code du projet, fonctions relativement sophistiquées comprises, ne nécessitent que 7,2% de la mémoire de programme de la MCU ? Bienvenue dans le monde ARM !
Vous avez découvert le plus gros projet proposé jusqu'à présent et devriez être en mesure de le comprendre. Vous maîtrisez en outre l'interface SPI du SAM D20 et savez piloter une EEPROM. Vous pouvez donc réaliser vos propres projets, voire peaufiner ce projet : modifier les périodes de mesure et allonger la durée de la mesure ; enregistrer la partie décimale

Listage 3. Coup d'œil au cœur de la boucle sans fin

```
if (measure_ended == 1 && port_pin_get_input_level(BUTTON_0_PIN) == 0)
{
for (i=0; i<10; i++)
  {
  spi_select_slave(&spi_master_instance, &slave, true);
  spi_write_buffer_wait(&spi_master_instance, &read_command, 1);
  spi_write_buffer_wait(&spi_master_instance, &adress[i], 1);
  spi_read_buffer_wait(&spi_master_instance, &read_data, 4, 0x00);
  spi_select_slave(&spi_master_instance, &slave, false);
  delay_ms(10);
  sprintf(usart_buffer_time, "Time: %d:%d:%d\n", read_data[0], read_data[1], read_data[2]);
  sprintf(usart_buffer_temperature, "Temperature: %d degree\n", read_data[3]);
  usart_write_buffer_wait(&usart_instance, (uint8_t *)usart_buffer_time, 15);
  usart_write_buffer_wait(&usart_instance, (uint8_t *)usart_buffer_temperature, 23);
  }
}
while(!port_pin_get_input_level(BUTTON_0_PIN)){}
```

de la température et/ou la date, par ex. Rappelez-vous qu'il n'est possible, selon le cas, de lire ou d'écrire qu'un maximum de seize octets (une page) d'un seul bloc.
Comme dans l'avant-dernier épisode, dont le sujet était l'interface I²C, nous placerons, bientôt dans le Forum Elektor [4] un projet qui permet d'essayer le SAM D20 en mode SPI esclave (avec une carte Arduino Uno).

Dans le prochain article, nous réaliserons des projets moins complexes qui auront recours à des bibliothèques négligées jusqu'à présent pour des raisons de place. Nous vous proposerons en outre des trucs & astuces pour le SAM D20 et Atmel Studio. Restez à l'écoute ! ⏮

(150286 – version française : Guy Raedersdorf)

Liens

[1] www.atmel.com/images/Atmel-42129-SAM-D20_Datasheet.pdf

[2] www.elektormagazine.fr/150286

[3] www.atmel.com/Images/Atmel-42115-SAM-D20-Serial-Peripheral-Interface-Driver-SERCOM-SPI_Application-Note_AT03255.pdf

[4] http://forum.elektor.com/viewforum.php?f=2698581

[5] https://fr.wikipedia.org/wiki/Serial_Peripheral_Interface

[6] http://ww1.microchip.com/downloads/en/DeviceDoc/22064D.pdf

La 25LC040

Ce membre de la famille 25LCxxxx, compact et à faible puissance, à huit pattes, d'une capacité de 4 Kbits (soit 512 octets) est facile à piloter par SPI à une fréquence d'horloge maximale de 2 MHz. L'EEPROM est une mémoire non volatile aux nombreux avantages, dont, ici une vitesse d'écriture élevée de 5 ms au maximum. La **figure 6** montre la structure de l'EEPROM. Elle possède les deux entrées actives au niveau bas $\overline{\text{HOLD}}$ et $\overline{\text{WP}}$. La première interrompt la transmission de données SPI en cours d'exécution, la seconde bloque l'accès en écriture via SPI. Si ces deux fonctions ne sont pas utilisées, les broches doivent être forcées au niveau haut par des résistances de polarisation haute. Les broches restantes servent à l'alimentation (2,5 à 5,5 V) ou sont utilisées par l'interface SPI.

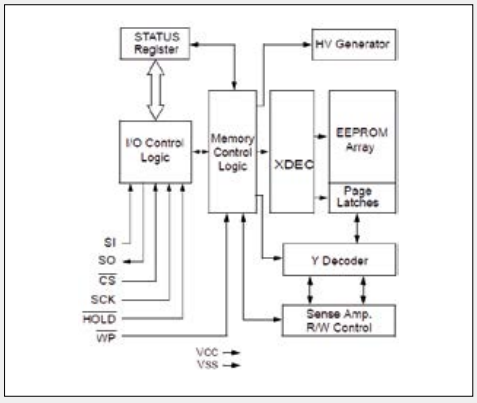


Figure 6. Structure de la 25LC040 (Source : Microchip)

La **figure 7** et le **tableau 1** montrent que les opérations d'écriture et de lecture sont simples. Envoi d'abord de l'instruction adéquate, puis de l'adresse de mémoire. Suivent les données à écrire ou lire, selon le cas. On peut envoyer ou recevoir jusqu'à 16 octets de données jusqu'à ce que le maître (MCU) remette la broche SS ($\overline{\text{CS}}$ ici) au niveau haut. Avant chaque opération d'écriture, il faut cependant activer l'accès en écriture par une instruction WREN unique. En outre, dans le registre d'état, les zones d'écriture concernées ne doivent pas être protégées en écriture.
La 25LC040 est compatible avec les modèles de plus grande capacité de cette famille ; celles-ci doivent cependant être adressées, à l'aide de plusieurs octets [6].

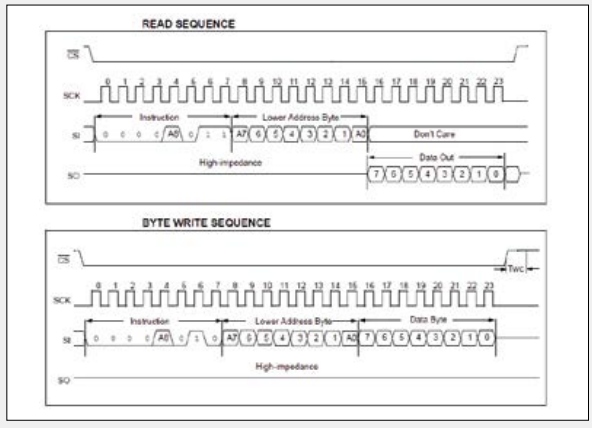


Figure 7. Dans la partie supérieure, une opération de lecture et en dessous, d'émission. Il est possible de lire ou d'envoyer séquentiellement jusqu'à seize octets (n'est pas visible ici). A8, le neuvième bit de l'adresse, n'a pas d'importance ici (donc bas). (Source : Microchip)

Tableau 1. Les instructions les plus importantes de la 25LC040 (version simplifiée)

instruction	description
READ (0x03)	lire des données depuis le circuit et à partir de l'adresse souhaitée
WRITE (0x02)	écrire des données dans le circuit et à partir de l'adresse souhaitée. Il faut auparavant que *Write Enable Latch* (cf. WREN) soit mis à un.
WRDI (0x04)	désactiver *Write Enable Latch*, rendant ainsi impossible toute écriture de données dans l'EEPROM
WREN (0x06)	activer *Write Enable Latch* afin de permettre l'écriture de données dans l'EEPROM
RDSR (0x05)	lire le registre d'état avec bits et drapeaux (*flags*) qui indiquent les zones de mémoire pour lesquelles est activée la protection en écriture
WRSR (0x01)	écrire le registre d'état avec bits et drapeaux (*flags*) qui indiquent les zones de mémoire pour lesquelles est activée la protection en écriture. Ceci a pour effet de modifier les paramètres de la protection en écriture.

synthèse numérique directe
une introduction

Robert Lacoste (Chaville)

L'électronicien doit garder à portée de main les briques essentielles que sont les techniques de production de signaux. La synthèse numérique directe (DDS) vient compléter ici la boucle à asservissement de phase (PLL) déjà présentée. Son principe est simple, mais gare au passage du numérique à l'analogique !

Dans mes deux précédents articles, j'ai détaillé la conception d'un oscillateur à quartz [1], puis présenté ce qu'on appelle une boucle à asservissement de phase ou PLL, pour *Phase Locked Loop* [2]. Pour mémoire, une telle PLL, associée à un oscillateur piloté en tension, permet de produire une fréquence quasi quelconque, mais verrouillée sur une fréquence de référence. Cette fréquence de référence provient en général d'un oscillateur à quartz. Une PLL permet donc d'obtenir une fréquence variable, mais aussi stable et précise que nécessaire.

Les PLL ont beau être incontournables dans beaucoup d'applications, en particulier en hautes fréquences, elles ne sont pas toujours la panacée en raison de deux inconvénients majeurs. D'une part leur résolution en fréquence, c'est-à-dire l'écart minimal entre deux fréquences pouvant être obtenues, est par nature limitée. Relisez mon dernier article pour plus de détails. D'autre part, une PLL comprend un filtre analogique : le filtre de boucle, dont le temps de stabilisation est non nul. Il passe donc un « certain temps » avant qu'une PLL se stabilise en cas de modification de la fréquence de sortie. Ce mois-ci je clôture cette minisérie d'articles par une autre technique de production de signaux : la synthèse numérique directe, ou *Direct Digital Synthesis* (DDS). La synthèse numérique directe n'a aucun des inconvénients d'une PLL.

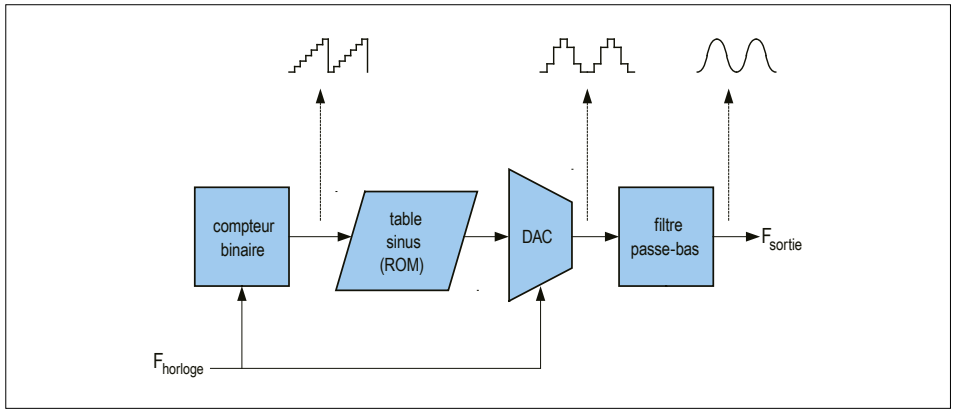


Figure 1. Un générateur de signal numérique élémentaire peut être construit autour d'un compteur binaire. Celui-ci balaye les lignes d'une ROM qui contient une période du signal. Il ne reste qu'à convertir le mot de sortie de la mémoire en un signal analogique et à le filtrer.

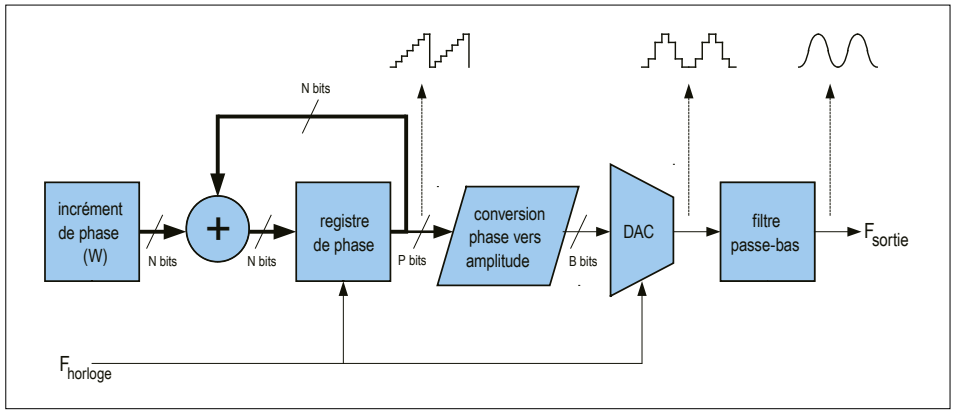


Figure 2. L'architecture de base d'un DDS est une variante du générateur de la figure 1, mais offre une résolution beaucoup plus fine de la fréquence, grâce à un registre de phase et un additionneur binaire.

DDS : les bases

Comme produire un signal périodique, par exemple une sinusoïde, avec des moyens aussi numériques que possible ? La solution la plus simple est illustrée sur la **figure 1**. Dans une mémoire numérique, par exemple une ROM, programmez une période de la forme d'onde souhaitée, pilotez ses lignes d'adresse avec un compteur binaire puis raccordez un convertisseur numérique/analogique (DAC) sur ses lignes de données. Il ne reste plus qu'à appliquer un signal d'horloge sur l'entrée du compteur : la mémoire sera lue à vitesse constante et vous aurez votre forme d'onde sur la sortie du DAC. Sa fréquence sera $F_{horloge}/2^N$, où N est le nombre de bits du compteur. Bien sûr, il ne faut pas oublier d'ajouter un filtre passe-bas afin de nettoyer le signal de sortie, c'est-à-dire transformer les marches d'escalier produites par le DAC en un signal lisse et continu. Comme vous le savez sûrement, la fréquence de coupure de ce filtre doit être un peu inférieure à la moitié de la fréquence d'horloge, c'est Monsieur Nyquist qui l'a dit. Ce

circuit est parfait, mais pas flexible. Par exemple, comment modifier la fréquence de sortie ? Il faudrait soit changer N, c'est-à-dire le nombre de bits du compteur et le nombre de lignes de la mémoire (pas simple), soit changer la fréquence d'horloge... ce qui nous ramène au problème initial et nécessiterait une PLL.

Qu'est-ce donc qu'un DDS ? C'est simplement une amélioration de cette conception originale, illustrée sur la figure 2. Plutôt que de lire séquentiellement toutes les lignes d'une table contenant une période du signal comme dans l'exemple précédent, un DDS est architecturé autour d'un registre baptisé *registre de phase*, de taille donnée, disons à N bits. À chaque coup d'horloge, le DDS ajoute à ce registre une valeur fixe (W), l'incrément de phase. Les bits de poids forts du registre de phase sont ensuite utilisés comme pointeur vers une table contenant ici encore une période complète du signal de sortie. La sortie de cette table est ensuite dirigée vers un DAC, puis vers un filtre analogique pour fournir le signal de sortie.

Comment ça fonctionne ? Supposons que l'incrément de phase W soit égal à 1. Dans ce cas, vous aurez besoin de 2^N impulsions d'horloge pour passer par toutes les valeurs de la table, c'est-à-dire pour générer une période du signal de sortie. La fréquence de sortie sera $F_{horloge}/2^N$.On est donc exactement dans le même cas que précédemment. Mais que se passe-t-il si W est égal à 2 ? La table sera lue deux fois plus vite, et la fréquence de sortie sera 2 x $F_{horloge}/2^N$. Et ainsi de suite. Un DDS permet donc de produire n'importe quelle fréquence de la forme : W x $F_{horloge}/2^N$.

Il y a quand même une limite : comme vous le savez, vous avez besoin d'au moins un peu plus de deux échantillons par période pour être en mesure de reconstituer un signal sinusoïdal. La valeur maximale de W est de là $2^{N-1} - 1$, et dans ce cas la fréquence de sortie est presque égale à la moitié de la fréquence d'horloge. Il est important de bien comprendre qu'un DDS n'est pas un simple diviseur de fréquence. Une petite feuille de calcul (**fig. 3**) peut vous aider à le comprendre : comme la valeur de l'incrément de phase W n'est en général pas un diviseur de 2^N, les valeurs successives du registre de phase d'une période à l'autre du signal ne sont pas les mêmes. Par contre, la fréquence de sortie est bien rigoureusement celle définie par la formule ci-dessus.

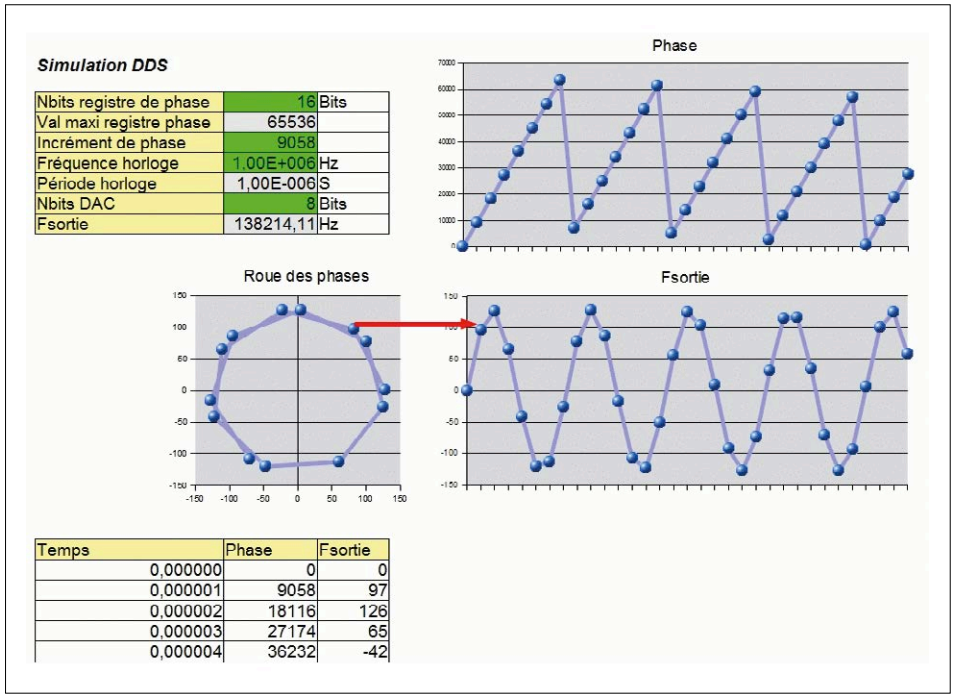


Figure 3. Cette simulation illustre le concept de *roue de phase* (*phase wheel*) d'un DDS. À chaque impulsion d'horloge, un angle fixe est ajouté au registre de phase. Comme cet incrément de phase n'est pas nécessairement un diviseur de 360°, les points générés à chaque période successive du signal ne sont pas exactement les mêmes, donnant de là une résolution très fine en fréquence.

À ce stade, des exemples numériques sont utiles pour montrer la puissance de l'approche DDS. Imaginons que le registre de phase ait une longueur de 32 bits et que l'horloge de référence soit de 20 MHz. Quelles fréquences peuvent être obtenues ? N'importe quelle fréquence comprise entre 1 x 20 MHz / 2^{32} soit 0,0046 Hz et ($2^{31} - 1$) x 20 MHz/ 2^{32} soit 9,999999 MHz, et ce avec une résolution de réglage de 0,0046 Hz ! Pas mal, non ? En fait, la fréquence maximale sera un peu plus faible en raison de contraintes sur le filtre passe-bas comme nous le verrons plus tard, mais vous avez saisi l'esprit...

La flexibilité d'un DDS...

Mis à part le filtre passe-bas, un DDS est purement numérique. De là, il est très facile de changer sa fréquence de sortie à la volée et sans aucun retard ni déphasage : il suffit de charger une nouvelle valeur dans le registre d'incrément de phase. De même, de petites améliorations de son architecture permettent facilement d'obtenir quasiment tout type de modulation et ce toujours entièrement en numérique (**fig. 3**). Par exemple la commutation entre deux valeurs pour l'incrément de phase donne une modulation FSK (*frequency shift keying*). On peut aussi ajouter une valeur fixe au registre de phase pour obtenir une modulation de phase ou une modulation PSK (*phase shift keying*). Enfin, rien n'interdit d'ajouter un multiplicateur numérique juste avant le DAC, ce qui donne une modulation d'amplitude.

Un DDS n'est pas non plus limité à des signaux sinusoïdaux : il suffit de programmer une autre forme d'onde dans la table pour obtenir n'importe quel signal périodique. Dans ce cas, la fréquence maximale sera significativement réduite à cause de l'obligatoire filtre passe-bas de sortie. En effet, un signal périodique, mais non sinusoïdal contient par définition des fréquences harmoniques, plus élevées que la fréquence fondamentale. La fréquence maximale de sortie devra de là être suffisamment basse pour que ces harmoniques soient inférieures à la fréquence de coupure du filtre passe-bas, qui doit rester inférieure à la moitié de la fréquence d'horloge. Tenez, un exemple : regardez la fiche d'un générateur de signaux de laboratoire numérique, par exemple un Agilent 33220A. C'est un générateur arbitraire cadencé par une horloge de 50 MHz. Sa fiche de caractéristiques indique qu'il peut produire un sinus jusqu'à 20 MHz, mais un triangle unique-

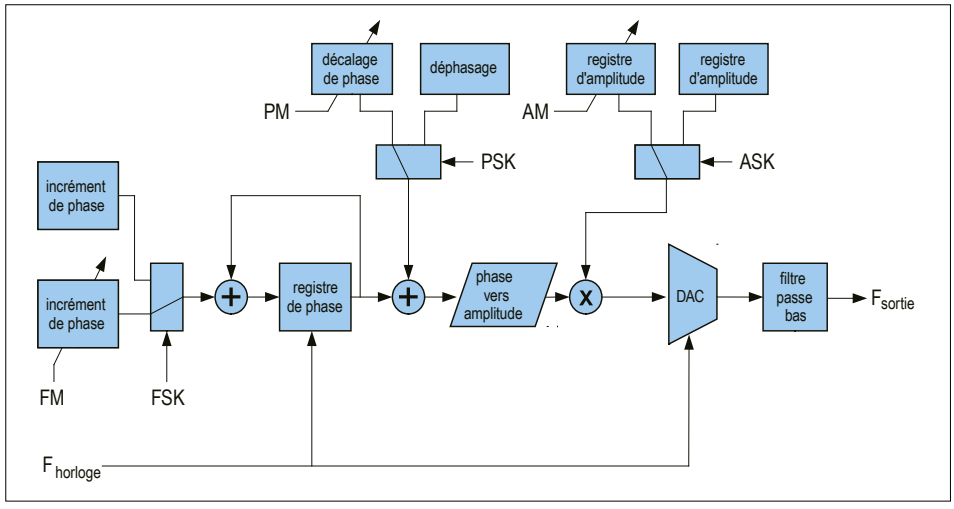


Figure 4. L'architecture d'un générateur DDS peut être facilement améliorée pour ajouter des fonctions de modulation numérique (en fréquence, en amplitude ou en phase)

ment jusqu'à 200 kHz. Maintenant, vous savez pourquoi !

Vous avez dit sin(x)/x ?

Maintenant que vous aurez compris qu'un DDS est fantastique, passons aux mauvaises nouvelles. Comme pour tout système numérique, le passage dans le monde analogique est critique, en particulier le filtre passe-bas de sortie. Pourquoi ce filtre est-il indispensable ? Tout simplement parce que la sortie du DAC est pas un signal sinusoïdal, mais une succession de marches d'escalier qui ne correspondent à une courbe sinusoïdale qu'à chaque impulsion d'horloge. Que se passe-t-il si l'on examine de signal dans le domaine fréquentiel ? Pour vous montrer ce qui se passe, j'ai simulé pour vous un DDS sans filtre passe-bas (**fig. 5**) sous SciLab, un outil *open source* de calcul numérique. Tout d'abord le signal de sortie contient des fréquences dites *images* : on trouve en sortie non seulement la fréquence F_{sortie}, mais aussi des signaux de fréquence $F_{horloge} - F_{sortie}$ et $F_{horloge} + F_{sortie}$, et même $2 \times F_{horloge} - F_{sortie}$, $2 \times F_{horloge} - F_{sortie}$, etc. Si vous regardez de près la figure 5, vous verrez que les amplitudes respectives des fréquences images suivent une courbe mathématique du type sin (x)/x.

Note : Cette fonction sin(x)/x, baptisée *sinus cardinal*, se retrouve partout en traitement du signal. Pour les personnes intéressées, c'est simplement parce que c'est la transformée de Fourier d'une impulsion de durée 1/Fhorloge, mais cela nécessiterait un autre article...

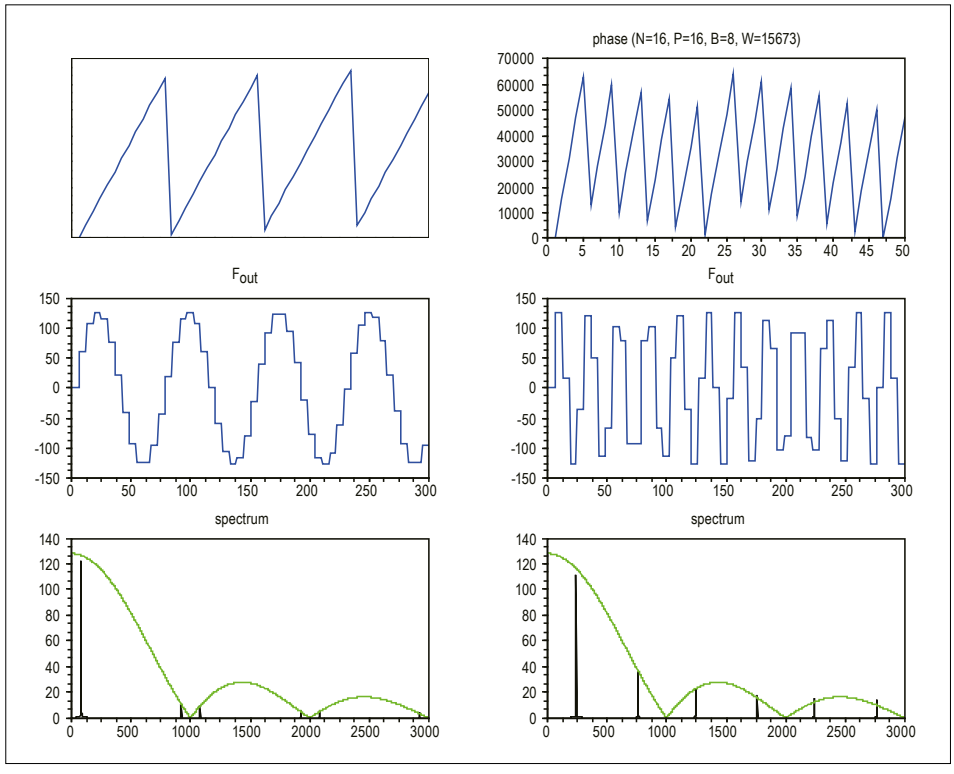


Figure 5. Cette simulation montre le spectre du signal délivré par un DDS doté d'un registre de phase de 16 bits cadencé à 1 GHz, avec deux valeurs différentes pour l'incrément de phase (F_{sortie} = 78 MHz à gauche, 239 MHz à droite). L'amplitude de la fréquence fondamentale baisse lorsque la fréquence augmente, en suivant une loi en sin (x)/x (courbe verte). Les fréquences images suivent la même loi.

Quelles sont les conséquences de la présence de ces fréquences images ? Premièrement un filtre passe-bas est indispensable pour supprimer tous les signaux images indésirables, et pour isoler la fréquence désirée, c'est-à-dire en général F_{sortie}. Malheureusement, plus la fréquence de sortie se rapproche de $F_{horloge}/2$ et plus la première fréquence image $F_{horloge} - F_{sortie}$ se rapproche de la fréquence voulue (**fig. 6**). Voilà pourquoi vous ne pouvez pas réellement produire de signal proche de cette limite, car le filtre passe-bas devrait être infiniment raide et donc impossible à construire.

Le second problème est que l'amplitude du signal de sortie ne reste pas constante lorsque la fréquence augmente, car elle suit la même loi sin(x)/x. Il faut donc compenser, si nécessaire, cette réduction de l'amplitude du signal lorsque la fréquence s'approche de plus en plus de la limite de Nyquist. Pour information, sans compensation l'atténuation de la puissance de sortie à cette fréquence est de 3,92 dB, ce qui n'est pas négligeable.

Un dernier mot : Rien ne vous empêche d'utiliser une des fréquences images plutôt que la fondamentale. Il suffit de remplacer le filtre passe-bas par un filtre passe-bande bien conçu et vous pourrez utiliser un DDS pour produire une fréquence supérieure à la limite de Nyquist. L'amplitude du signal de sortie sera simplement plus faible.

Quelques autres soucis...

Maintenant un bon filtrage élimine les fréquences images, mais signal de sortie est-il pour autant un sinus parfait ? Bien sûr que non : le nombre de bits

du registre de phase n'est pas infini, le nombre de lignes de la table de forme d'onde est fini, et le nombre de bits du DAC est également fini. Par exemple, le nombre limité de bits du DAC entraînera une erreur de quantification, ce qui se traduira par un bruit dans le spectre du signal de sortie. Pour illustrer le phénomène, j'ai modifié ma simulation de DDS sous SciLab pour montrer le spectre de sortie avec un DAC de 8 bits (**fig. 7**). La théorie dit que le rapport signal/bruit de quantification est de 1,76 + 6,02 B dB, où B est la résolution en bits de la DAC. Ainsi, avec un convertisseur N/A à 8 bits, vous pouvez vous attendre un rapport S/B du signal de 1,76 + 6,02 x 8 = 50 dB. Ce bruit de quantification peut être réduit soit en ajoutant un filtre passe-bande étroit autour de la fréquence de sortie désirée, soit en augmentant la fréquence d'horloge sans modifier la fréquence de coupure du filtre passe-bas.

Les DDS présentent un autre inconvénient, souvent plus critique que les erreurs de quantification : le nombre N de bits de l'accumulateur de phase n'est pas infini, et souvent la table de forme d'onde est également beaucoup plus petite que 2^N (seuls les bits de poids fort du registre de phase étant utilisés). Cela donnera une autre erreur sur le signal de sortie, mais cette fois ce ne sera pas un bruit à bande large, mais des fréquences parasites sur le spectre de sortie, ce qui est en général plus pénible. Une fois de plus, la théorie aide, et prédit que la puissance relative du signal parasite le plus grand est d'environ –6,02 P dBc, où P est le nombre de bits d'adresse de la table de forme d'onde. La difficulté est que les fréquences et les amplitudes de ces signaux parasites sont également fonction de la valeur de l'incrément de phase : si vous changez un peu la fréquence d'un DDS, les fréquences parasites seront tout autres, et ceci est en pratique fort pénible. La bonne nouvelle est que le comportement d'un DDS est prévisible et que les fabricants de circuits intégrés proposent de bons outils de simulation.

Une DDS par logiciel ?

Assez de théorie. Comment construire un générateur DDS ? Avant de vous présenter quelques circuits intégrés spécialisés, voyons comment le faire en *firmware*. Imaginez que vous souhaitez obtenir un signal sinusoïdal, disons de 7117 Hz, avec un p. ex. un PIC16F629A de Microchip cadencé à 20 MHz. Ce n'est qu'un exemple, bien sûr, mais représentatif de situations courantes où l'on a besoin de signaux DTMF, de signaux de commande, etc. La première idée qui pourrait traverser votre esprit serait d'utiliser un temporisateur du microcontrôleur pour produire un signal PWM de 7117 Hz, associé à un filtre passe-bas analogique. Sur cette variante de microcontrôleur, les temporisateurs sont pilotés par une horloge égale au quartz de la fréquence d'horloge, soit 20 MHz/4=5 MHz. Comme un temporisateur ne peut être programmé qu'à une valeur entière, vous pourrez obtenir soit 5 MHz/ 702 = 7122 Hz, soit 5 MHz/ 703 = 7112 Hz. Pas trop mal, mais assez loin de la cible. Comment faire mieux avec les mêmes ressources matérielles ? En programmant un DDS bien sûr. Configurez un temporisateur pour produire une interruption à une fréquence quelconque, mais nettement supérieure à 2 x 7117 Hz, disons 50 kHz. À chaque interruption, ajoutez une valeur fixe W à un registre de phase de 16 bits, convertissez-le en sinus avec une table en ROM avec p. ex 256 valeurs, et envoyez la valeur à un DAC. Filtrez enfin avec un passe-bas 10 kHz et voilà. À titre d'exemple, voici un schéma correspondant à ce concept (**fig. 8**). Le DAC est un simple réseau R-2R de 4 bits. Une paire d'amplificateurs opérationnels MCP6002 fait office de tampon et de filtre passe-bas. Comme le montre la simulation, le signal de sortie après filtrage est une belle sinusoïde à la fréquence voulue. J'ai simulé ce circuit grâce à l'outil VSM de la société *Labcenter* (disponible dans la suite logicielle *Proteus*). Cet outil n'est pas gratuit, mais permet de simuler non seulement un circuit analogique (résistances, filtre, etc.), mais aussi le logiciel exécuté sur le microcontrôleur, ce qui est magique dans un tel cas...

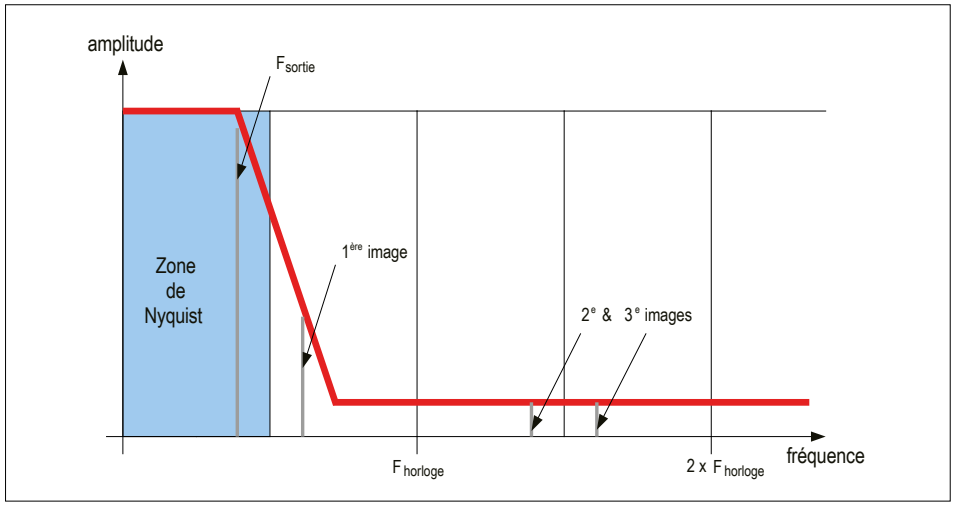


Figure 6. Le rôle du filtre passe-bas est fondamental. Il ne doit pas trop modifier l'amplitude du signal fondamental, mais atténuer drastiquement les fréquences images. C'est pourquoi des filtres très raides sont souvent nécessaires, surtout si la fréquence de sortie doit s'approcher de la fréquence de Nyquist.

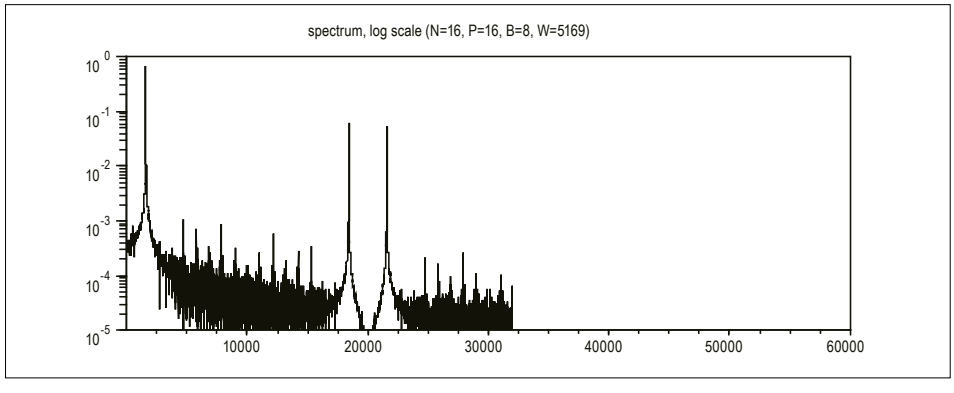


Figure 7. Cette simulation montre l'effet du bruit de numérisation du DAC, ici dans le cas d'un DDS 1 GHz réglé pour produire un signal de 78 MHz, et doté d'un DAC à 8 bits.

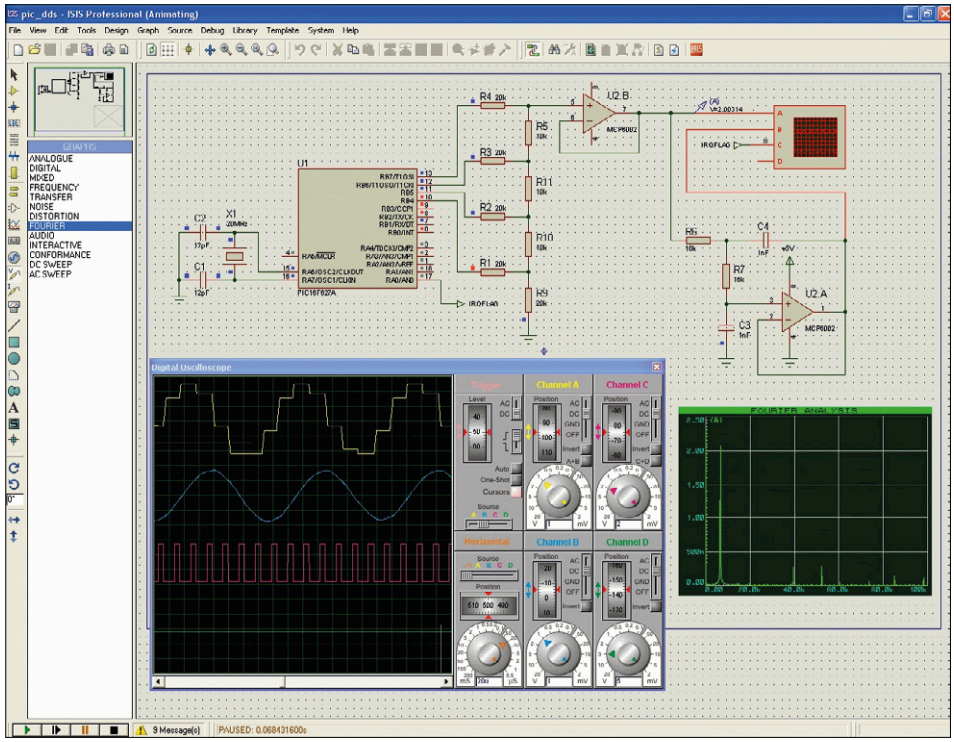


Figure 8. Une petite simulation d'un générateur DDS réalisé en firmware avec un PIC et un DAC à 4 bits. Cette simulation a été réalisée à l'aide du simulateur signaux mixtes VSM, de la suite *Proteus*.

Ce circuit est-il meilleur que l'approche PWM évoquée ? Comme on a construit un véritable DDS à 16 bits, on peut produire toute fréquence de la forme W x 50 kHz/65 536. Par exemple il suffit de choisir W = 9328 et vous obtenez une fréquence de 7116,69 Hz, beaucoup plus proche de la cible de 7117 Hz.

Un peu de silicium

L'approche logicielle est souvent utile, mais l'utilisation d'un circuit intégré DDS est également une option plus qu'intéressante. Le leader de ce domaine est sans conteste *Analog Devices* (www.analog.com/dds). Quelques exemples ? Pour quelques dollars et quelques millimètres carrés sur votre circuit imprimé, le AD9837 vous donne n'importe quelle fréquence jusqu'à quelques MHz avec une résolution de 0,06 Hz et une consommation infime (8,5 mW). Quelques registres à programmer via un port SPI et voilà ! Malgré sa simplicité apparente, un tel composant dispose de fonctions de modulation en fréquence, en phase ou en amplitude. Si l'anglais ne vous rebute pas trop, je vous recommande chaudement de télécharger la feuille de caractéristiques de cet AD9837, sa lecture devrait être limpide si vous avez compris les bases présentées dans cet article.

À l'autre bout de l'échelle, et à condition de débourser une somme un peu plus rondelette (euh, 150 $ la puce par 100 pièces...), vous pouvez vous offrir chez le même fabricant un AD9914 : fréquence d'horloge jusqu'à la bagatelle de 3,5 GHz et résolution en fréquence de 190 pHz (oui, ce sont des picohertz). Ce joujou consomme quand même jusqu'à 3 W, mais c'est plus qu'impressionnant, non ? Une autre famille de composants très intéressants et plus abordables est illustrée par le AD9102 : ce DDS, plus limité en fréquence (180 MHz quand même), présente un avantage colossal : sa table de forme d'onde n'est pas une ROM contenant un sinus, mais *une RAM de 4096 mots de 14 bits librement programmable*. Ce composant permet donc de réaliser un générateur de formes d'onde arbitraires tout en offrant la flexibilité d'un DDS ! À seulement 20 € de l'unité, cela donne des idées…

Pour conclure

En résumé, un DDS permet d'obtenir une résolution fine en fréquence, associée à des possibilités de modulation flexibles et ultra-rapides. Sa fréquence maximale reste limitée à environ 40% de sa fréquence d'horloge (à moins d'utiliser des fréquences images) et le signal de sortie peut contenir quelques signaux parasites désagréables.

Comparez ces avantages et ces inconvénients à ceux d'une PLL et vous en déduirez qu'ils sont assez complémentaires : une PLL est parfaite pour construire des oscillateurs locaux, où hautes fréquences et pureté sont impératives, tandis que les DDS sont parfaits comme sources de modulation, où l'agilité est clé. De là, DDS et PLL sont souvent les meilleurs amis du monde, et les modes de combinaison sont multiples : un DDS peut être utilisé pour produire l'horloge de référence d'une PLL, ou être intégré directement dans la boucle d'une PLL, etc. Il existe même des composants intégrant à la fois une PLL et un DDS.

Maintenant, à vous de jouer ! ⏮

Cet article a été publié dans la revue Circuit Cellar (nº 217, août 2009

(150315)

Liens

[1] Elektor nº 440, mars 2015, hors circuits : les quartz

[2] Elektor nº 442, mai 2015, hors circuits : boucle à asservissement de phase ou PLL

Sources

Proteus, Labcenter : www.labcenter.co.uk

AD9833, 9910, AD9912 : Analog devices

PIC16F629A, MCP6002 : www.microchip.com

Scilab : www.scilab.org/

Références

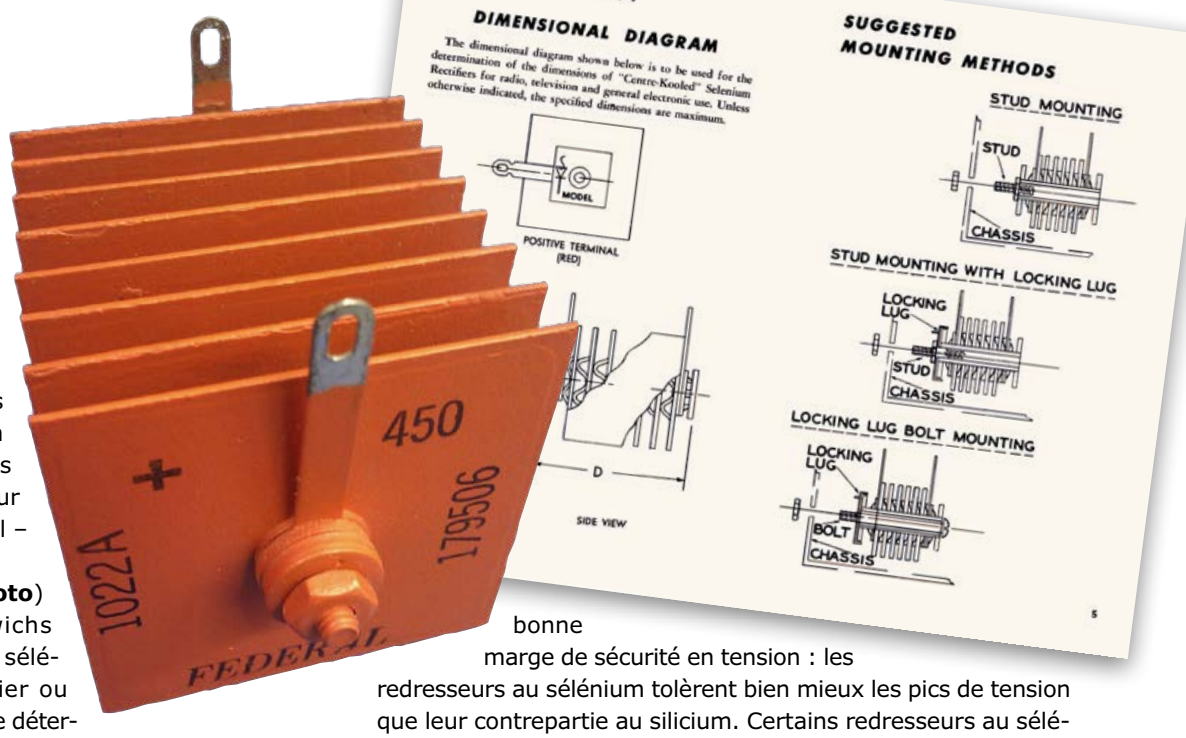
A technical tutorial on Digital Signal Synthesis, Analog Devices, 1999, www.analog.com/UploadedFiles/Tutorials/450968421DDS_Tutorial_rev12-2-99.pdf

DDS Primer, Analog Devices, 2003, www.ieee.li/pdf/viewgraphs_dds.pdf

redresseur au sélénium

drôle de composant n°17

Neil Gruending (Canada)



Tout le monde sait que lorsqu'on redresse une tension alternative on obtient une tension continue. Les redresseurs au silicium font ça très bien, mais dans les années 50 et 60, ils n'étaient pas disponibles pour les courants forts ; on utilisait alors des redresseurs au sélénium. Ces derniers sont un peu plus gros que leurs homologues au silicium mais cela ne les empêche pas d'être relativement efficaces pour des composants à base de métal – du moins pour l'époque.

Un redresseur au sélénium (**photo**) est un empilement de sandwichs constitués d'une fine couche de sélénium entre deux plaques d'acier ou d'aluminium. La surface de plaque détermine la capacité en courant du redresseur, leur nombre celle en tension. Chaque plaque ajoute environ 20 V à la tension inverse supportée et le redresseur de la photo possède par ex. une tension inverse de crête (PIV pour *peak inverse voltage*) de 160 V grâce à ses huit plaques.

Les redresseurs au sélénium étaient souvent utilisés dans les alimentations, sans doute grâce à leur rendement nettement meilleur que celui des tubes à vide de l'époque. Un redresseur au sélénium 120 V par ex. a une tension de chute typique de 5 V, il faut compter 10 à 25 V pour un redresseur à tube. Le tube nécessite également un courant de chauffe, ce qui réduit d'autant son efficacité.

Si vous possédez une radio, une télévision, un appareil de mesure ou toute autre antiquité des années 50/60, son alimentation contient probablement un redresseur au sélénium. Même s'ils fonctionnent correctement, il est conseillé de les remplacer par des diodes au silicium. Lorsqu'ils vieillissent, les redresseurs au sélénium voient leurs tension de chute et courant de fuite augmenter, non sans effet sur les performances des circuits. Dans le pire des cas, ils surchauffent et prennent feu, le tout accompagné d'une fumée toxique et nauséabonde qui pour beaucoup a une odeur d'ail ou d'oignon.

Par chance les redresseurs au sélénium, simple ou en pont, sont typiquement spécifiés pour un courant direct de 150 à 200 mA, leur remplacement par une diode est donc facile. Habituellement on utilise des diodes des séries 1N400x, 1N540x et BYxxx, disponibles dans beaucoup de gammes de tension ; leur courant nominal de 1 A est plus que suffisant. Toutefois gardez une bonne marge de sécurité en tension : les redresseurs au sélénium tolèrent bien mieux les pics de tension que leur contrepartie au silicium. Certains redresseurs au sélénium suivent cette nomenclature : B*xxx*C*yyy*, avec *xxx* = PIV et *yyy* = mA. Facile. Les modèles au silicium l'ont conservé.

Une fois la diode au silicium choisie, la prochaine étape est d'ajouter une résistance série pour simuler la limitation en courant et la plus grande tension de chute des redresseurs au sélénium. Le plus facile pour en déterminer la valeur est de diviser la chute de tension désirée (5 à 10 V) par le courant direct attendu. Assurez-vous également que la résistance soit spécifiée pour une puissance suffisante ; en général on utilise des résistances 5 W de 100 à 200 Ω.

D'habitude, je n'aime pas trop remplacer les composants d'origine mais malheureusement, c'est inévitable pour les redresseurs au sélénium. Il ne s'agit pas de savoir *si* ils vont tomber en panne mais *quand*. Et lorsque cela se produit, il faut des jours pour que la fumée et l'odeur se dissipent ! Ceux qui ne reculent devant rien pourront dénicher des informations dans le *Selenium Rectifier Handbook* [2].

(150283 – version française : Kévin Petit)

[1] http://en.wikipedia.org/wiki/Selenium_rectifier#/media/File:-Selenium_Rectifier.jpg

[2] www.tubebooks.org/Books/srhbst.pdf

Envoyez-moi vos idées en quelques lignes ou sous la forme d'un article déjà rédigé : neil@gruending.net

formats de fichier pour les circuits imprimés
qui gagnera la guerre des formats ?

Robert Huxel, Altium Europe GmbH

L'électronique est un secteur en évolution rapide, toujours à la pointe de la technologie. Toutefois l'étape clé de la préparation du circuit imprimé pour la fabrication est très conservatrice. Mais dans ce domaine la discussion à propos de la voie à suivre est actuellement pour le moins animée.

Le point principal consiste à fournir des données à partir du processus de conception pour permettre la fabrication d'un circuit imprimé correct. Le format d'échange des fichiers de production a évolué au fil du temps et il n'est peut-être pas étonnant de constater qu'il n'existe aucune spécification totalement universelle ou *standard* de ce qui constitue un ensemble complet d'informations à remettre à la fabrication. Il y a eu, et il y a encore, une somme considérable d'interventions manuelles dans le processus ; le fabricant vérifie que ce qu'il a reçu est correct et qu'il sera possible de fabriquer. Souvent, les contacts personnels entre concepteur et fabricant facilitent le travail, changer de sous-traitant peut donc être perturbant.

Un nombre croissant de voix déclarent que ce paradigme ne suffit plus, que le secteur a besoin d'une nouvelle norme complète pour définir un circuit imprimé sans erreur, ni ambiguïté, une norme qui puisse être utilisée pour piloter un flux de fabrication automatisé pour des cartes toujours plus complexes. Comme souvent dans l'industrie électronique, lorsque les acteurs ont perçu la nécessité d'une nouvelle norme, un certain nombre de candidats qui se font concurrence ont émergé. Dans ce cas, le nombre (selon la façon dont on les classe et dont on retrace leurs antécédents) est de trois. Comme généralement dans l'industrie, il y a une scission philosophique entre ceux qui voudraient construire une nouvelle norme de bas en haut, et ceux qui opteraient pour poursuivre l'évolution de ce qui fonctionne déjà, en ajoutant des caractéristiques et en corrigeant les défauts en fonction des besoins.

Les candidats

La faction qui pense : « Si dans l'ensemble ça fonctionne, ne faisons pas de gros changements - développons ce dont nous disposons déjà pour traiter les zones à problèmes » - ce qui correspond à la doctrine du « changement minimum nécessaire » - est axée sur le format Gerber. Le format lui-même est à présent sous la garde d'Ucamco.

La révision la plus récente de l'ordre établi a ajouté des caractéristiques au format Gerber étendu largement utilisé, ce qui a donné naissance à « Gerber X2 » (**fig. 1**). Un élément clé de cette progression est que la quantité d'informations contenues dans les fichiers de géométrie de base est améliorée parce que les caractéristiques qu'ils en tirent portent à présent des attributs. Ces attributs peuvent spécifier quel type d'objet est représenté, et à partir de là, des aspects tels que la connectivité peuvent être déduits automatiquement. L'une des reven-

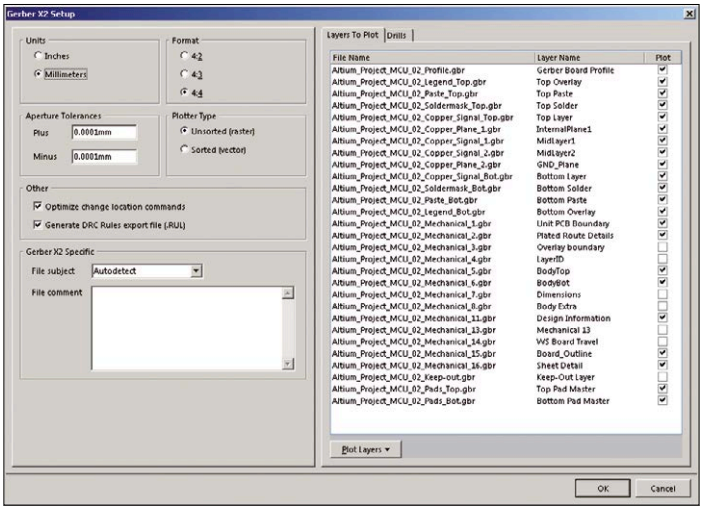


Figure 1. Boîte de dialogue de configuration Gerber X2 sous Altium Designer.

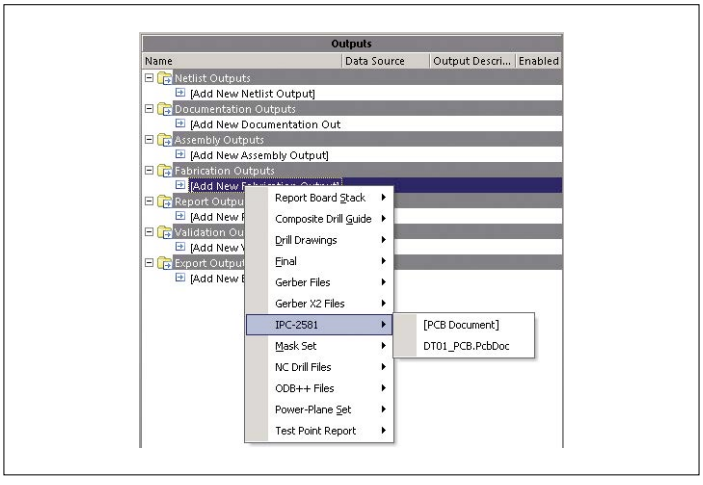


Figure 2. Faites facilement votre choix entre IPC-2581 et Gerber X2.

dications, et pas des moindres, émises en faveur de cette mise à niveau - qui correspond au conservatisme inné du secteur - est qu'elle est rétro-compatible avec la pratique répandue aujourd'hui ; si vous ne voulez pas utiliser les informations ajoutées, votre logiciel ignore tout simplement les données ajoutées et votre système peut continuer comme avant.
Pour tout ingénieur qui défend le progrès par l'amélioration continue, il y en a un qui défend l'approche de la feuille blanche, et ce secteur ne fait pas exception. En général, les camps opposés proposent une approche centrée sur les données, qui tire toutes les informations nécessaires d'un seul ensemble de données, plutôt que de la vue centrée sur la géométrie de Gerber. Avec sa propre histoire, un candidat est ODB++. Le double signe plus indique que ce format a suivi son propre processus évolutif. Le format a été créé par Valor, société par la suite acquise par Mentor Graphics, et le format est désormais maintenu, et promu, par Mentor.

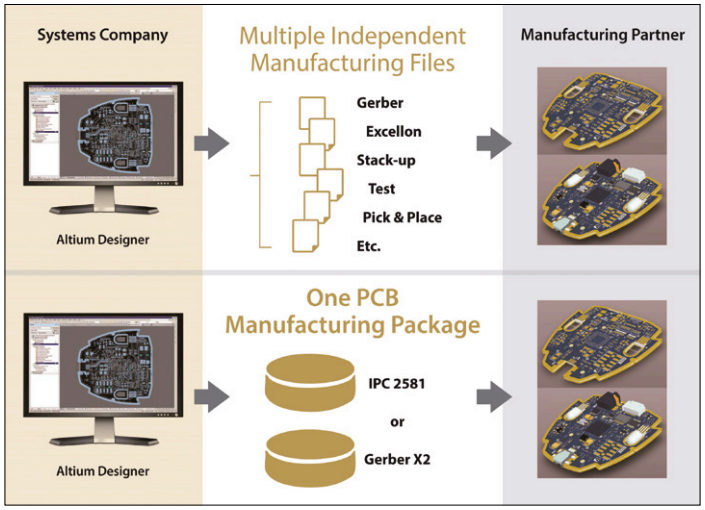


Figure 3. Le dossier de fabrication d'un CI devient un événement « en un seul clic » dans le processus de conception.

Le troisième camp est celui des passionnés du format IPC-2581 (**fig. 2**), description unique basée sur le langage XML (*eXtended Markup Language*). Comme dans d'autres domaines dans lesquels XML est utilisé, ses partisans font observer qu'il a été conçu comme un support de données ; c'est un format source cohérent qui permet de produire des ensembles de données dérivés (par ex. des images et des fichiers CNC). Il constitue par conséquent une base appropriée sur laquelle construire une nouvelle norme pour l'interface entre CAO/FAO et CI. IPC-2581 est sous la garde d'un consortium dédié, dont Cadence Design Systems est un élément moteur.

Le défi de l'adoption

Les utilisateurs sont souvent méfiants quand il s'agit d'adopter des formats qui sont considérés comme propriétaires, et de se laisser entraîner dans la « guerre des normes » du secteur de la CAO électronique. Ils examinent le rôle joué par les principaux leaders de la CAO électronique dans chaque norme. Dans le cas d'ODB++, Mentor a toujours positionné cette norme comme « ouverte », tous les aspects pouvant être pris en charge par les utilisateurs - et concurrents - de tout le secteur. Cadence souligne sa position en tant que l'un des principaux acteurs du consortium indépendant.
Toutes ces approches sont structurées de façon à englober les paramètres qui caractérisent les circuits imprimés à haute performance pour les conceptions d'aujourd'hui et de demain - par ex. les sections rigides/flexibles sur une même carte, l'identification des pistes et des vias comme étant appariés par l'impédance.
Quel est l'impact probable du passage à une autre stratégie ? De toute évidence, le concepteur et le fabricant doivent adopter les mêmes normes et assimiler les coûts d'acquisition des outils nécessaires ; les produits présents sur le marché sont-ils suffisamment mûrs pour que le « plug and play » soit assuré, ou un processus d'apprentissage des deux côtés est-il inévitable ? En résumé, s'il est équipé d'un progiciel de conception de CI qui peut délivrer un format de prochaine génération donné, tout ce qu'un équipementier devra faire sera de veiller à ce que le fabricant de sa carte puisse accepter ce format, et de commencer à l'utiliser pour transférer les spécifications de la carte à la fabrication. Ces changements des pratiques de travail sont rarement aussi simples. Mis à part les problèmes relatifs au format proprement dit, et l'interprétation de celui-ci par toutes les parties, les changements apportés à l'environnement dans lequel est conçu le circuit imprimé ont également une incidence.

Les structures évoluent

Aujourd'hui, le concepteur de circuits peut utiliser un logiciel puissant et complet qui porte son projet de l'architecture et de la capture de schémas jusqu'à la conception physique 3D du produit, en passant par la conception de tous les aspects du CI. En effet, avec une conception à grande vitesse, dense et critique pour l'intégrité du signal, il est probablement essentiel que le concepteur de circuits contrôle la géométrie de la carte au fur et à mesure qu'il développe cette dernière. Dans ce flux, l'exportation des données de fabrication du CI devient une étape « en un clic » du processus (**fig. 3**).
Il s'ensuit que le niveau de standardisation doit être élevé, et que la confiance dans le flux aval pour interpréter l'ensemble des données et exécuter en fonction de cela, automatiquement, doit être tout aussi élevée. Les mêmes pressions sur les délais de mise sur le marché qui poussent l'évolution vers des normes plus avancées font qu'une fois le bouton appuyé pour télécharger l'ensemble des données du CI, l'attention de l'ingénieur sera dirigée ailleurs. La « zone de confort » du fabricant d'un ingénieur en CI dédié, disponible immédiatement pour résoudre les problèmes, est susceptible de diminuer. Pour ces raisons, il semble peu probable que la base des utilisateurs qui conçoivent/fabriquent des CI abandonne son conservatisme à la hâte.
Il semble également peu probable que l'un des formats en concurrence devienne totalement dominant. Pour ce qui est du format (une méthodologie centrée sur les données ou sur la forme ne devra en aucun cas imposer des restrictions sur le processus de conception ou, à terme, le flux de fabrication), cela se résoudra sûrement de la manière dont bon nombre de « guerres » de format l'ont été. Autrement dit, par un concours de popularité de longue haleine dans lequel les fournisseurs d'outils du côté conception devront prendre en charge de multiples variantes, un « gagnant » finira par émerger lentement - le cas échéant. ⏮

(140525)

fiabilité des condensateurs électrolytiques

Les condensateurs électrolytiques à l'aluminium sont vitaux pour beaucoup d'appareils électroniques. Le besoin toujours croissant d'efficacité énergétique, l'utilisation grandissante des énergies renouvelables, et la présence de plus en plus forte de l'électronique dans les automobiles modernes ont été des moteurs significatifs de la démocratisation de ces composants dans les dernières décennies.

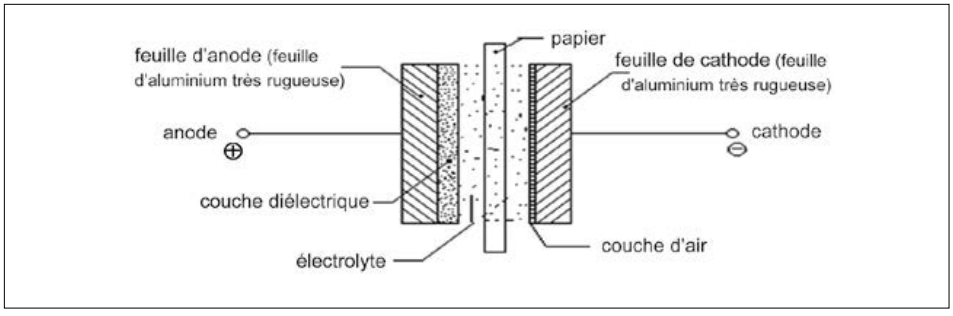


Figure. 1. Structure d'un condensateur électrolytique à l'aluminium.

Structure et construction

Les condensateurs électrolytiques à l'aluminium combinent tension maximale, de quelques volts à env. 750 V, et capacité élevée, de 1 µF jusqu'à plus de 1 F, sous forme compacte. Une feuille d'anode à la rugosité fortement augmentée est recouverte d'une fine couche diélectrique et mise en contact avec une cathode qui l'épouse parfaitement, via une l'électrolyte liquide (**fig. 1**). Le procédé de fabrication de ces condensateurs comprend les étapes majeures suivantes :

1. **Gravure** — des feuilles d'aluminium de grande pureté, de 20 à 100 µm d'épaisseur, sont le matériau de base pour les anodes et les cathodes. La gravure augmente la surface totale des feuilles d'un facteur de 140 (**fig. 2**) par rapport à leur surface géométrique.

2. **Formage** — la feuille d'anode supporte le matériau diélectrique constitué d'oxyde d'aluminium (Al2O3). Il y est déposé par un procédé électrochimique appelé oxydation anodique ou formage. La qualité du formage, c'est-à-dire l'homogénéité et la bonne couverture de la surface, est essentielle à une bonne fiabilité des composants en fonctionnement. Plus la tension de formage est éloignée de la tension maximale spécifiée, plus la probabilité de rupture du diélectrique est faible. Les électrolytiques de Jianghai ont un rapport entre tension de formage et tension maximale compris entre 1,25 pour les faibles tensions et 1,60 pour les hautes tensions. L'épaisseur du diélectrique est d'env. 1,4 nm/V, soit 900 nm pour un modèle spécifié pour 450 V (< 1/100 de l'épaisseur d'un cheveu).

3. **Découpe** — la feuille gravée et formée est produite en rouleaux de 50 cm de large, qui sont découpés pour obtenir les largeurs nécessaires aux anodes et cathodes.

4. **Enroulage** — des contacts sont attachés aux feuilles (couture ou soudure à froid) avant que celles-ci et le papier qui les sépare ne soient enroulés (plusieurs couches si nécessaire).

5. **Imprégnation** — les pores du papier de séparation et toute la surface de la feuille d'anode sont recouverts d'un électrolyte, la cathode liquide.

6. **Assemblage** — la cellule bobinée est placée dans son boîtier, les connexions électriques entre les contacts et les broches sont réalisées avant le scellement du boîtier par rivetage.

7. **Post-formage** (*burn-in*) pour « cicatriser » les bords découpés des feuilles.

8. **Contrôle** des paramètres électriques vitaux **sur la ligne de production pour 100% des composants** (capacité, facteur de dissipation et courant de fuite).

La figure 2 montre une vue au microscope de la surface d'une feuille d'anode haute tension après gravure. La distribution homogène et le grand diamètre des trous créés par gravure permettent une bonne couverture par la couche d'oxyde et un accès total à la surface pour l'électrolyte. À ce stade précoce de la production, il est déjà possible de déterminer si la feuille pourra être utilisée pour un

Dr. Arne Albertsen, Jianghai Europe Electronic Components GmbH

La durée de vie et la fiabilité de nombre d'applications sont directement liées aux paramètres correspondants des condensateurs électrolytiques [4]. Dans un précédent article [1], l'auteur s'intéressait à l'estimation de la durée de vie de ces condensateurs. Cet article traite de leur fiabilité.

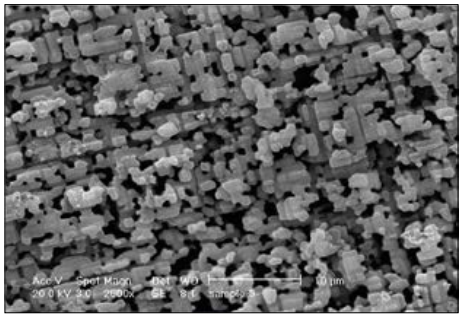


Figure 2. Vue au microscope d'une feuille d'anode après gravure.

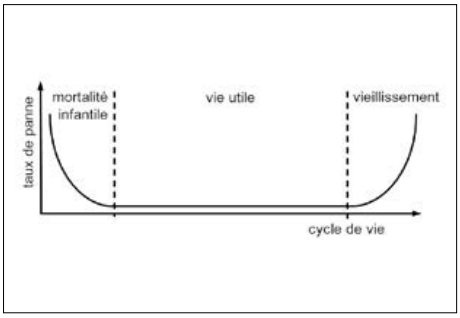


Figure 3. Taux de panne en fonction du temps : la courbe en baignoire.

condensateur convenant aux applications les plus exigeantes qui imposent haute fiabilité, tolérance au courant ondulatoire et durée de vie longue.

Les étapes 2 et 7 ont une grande influence sur la fiabilité des condensateurs électrolytiques en fonctionnement. Jianghai s'efforce de maintenir une grande distance entre la tension de formage et la tension spécifiée ainsi qu'un temps de séjour raisonnable lors du post-formage pour garantir une fiabilité élevée. Comme la tension de formage n'est pas toujours indiquée dans les feuilles de caractéristiques, le client final ne dispose pas de ce paramètre comme indicateur de performance. En demandant au fournisseur et en comparant les courants de fuite spécifiés, l'utilisateur en déduira les choix qu'a faits le fabricant. Alors que les prix des matériaux et de l'énergie augmentent, même certains fabricants connus n'hésitent pas à diminuer des tensions de formage de séries en production. Jianghai considère cette « optimisation » inacceptable pour la qualité.

Durée de vie contre fiabilité

Les mécanismes électrochimiques du vieillissement des condensateurs électrolytiques limitent leur durée de vie à une valeur qui peut être estimée en fonction de la température, du courant ondulatoire et de la tension en fonctionnement. Des pannes peuvent toujours survenir aléatoirement. Le nombre absolu de ces pannes dépend de la taille totale du lot observé. L'existence de pannes aléatoires n'est en général pas liée au processus de vieillissement, mais c'est plutôt la conséquence de points faibles internes et cachés (p.ex. dans le papier de séparation, la feuille, ou à proximité des connexions électriques). Souvent, ces pannes se produisent sans signes avant-coureurs et se terminent en court-circuit. L'augmentation des courants de fuite, à cause d'un diélectrique endommagé, entraînera une déformation telle (accompagnée d'une accumulation d'hydrogène gazeux) que la surpression ouvrira l'évent de sécurité. Le condensateur se dessèche alors et sa capacité devient très faible.
Les mesures de capacité, courant de fuite et ESR sur tous les composants produits et la conduction de tests supplémentaires sur des échantillons sélectionnés aléatoirement garantissent la qualité des produits. Les pannes précoces dans les applications sont donc rares [2].
Il existe beaucoup de définitions du terme « fiabilité », chacun (statisticien, mathématicien, ingénieur…) a la sienne. Une définition dictée par le bon sens pourrait être : « la probabilité qu'un appareil électronique remplisse sa fonction de manière satisfaisante durant une période donnée ».
Le taux de panne des condensateurs électrolytiques suit la fameuse « courbe en baignoire » [3]. Le taux de FIT (*Failures in Time*) λ représente le nombre de pannes par unité de temps (densité des pannes, unité de mesure 1 FIT = 10^{-9} pannes/h). La courbe en baignoire (**fig. 3**) fait apparaître trois phases distinctes ;

- la période de pannes précoces (« mortalité infantile ») où le taux de FIT λ décroît
- la période de vie utile où il est constant (pannes aléatoires)
- la période de fin de vie durant laquelle le taux λ augmente à cause de l'usure ou de changement au-delà des limites acceptables à la fin ou après la fin de la durée de vie normale

Le taux de panne indiqué en fonction de la « durée de vie utile » fait référence au pourcentage de composants dont les paramètres ont évolué au-delà des spécifications lors du test de durée de vie ; ne le confondez pas avec le taux de pannes aléatoires.

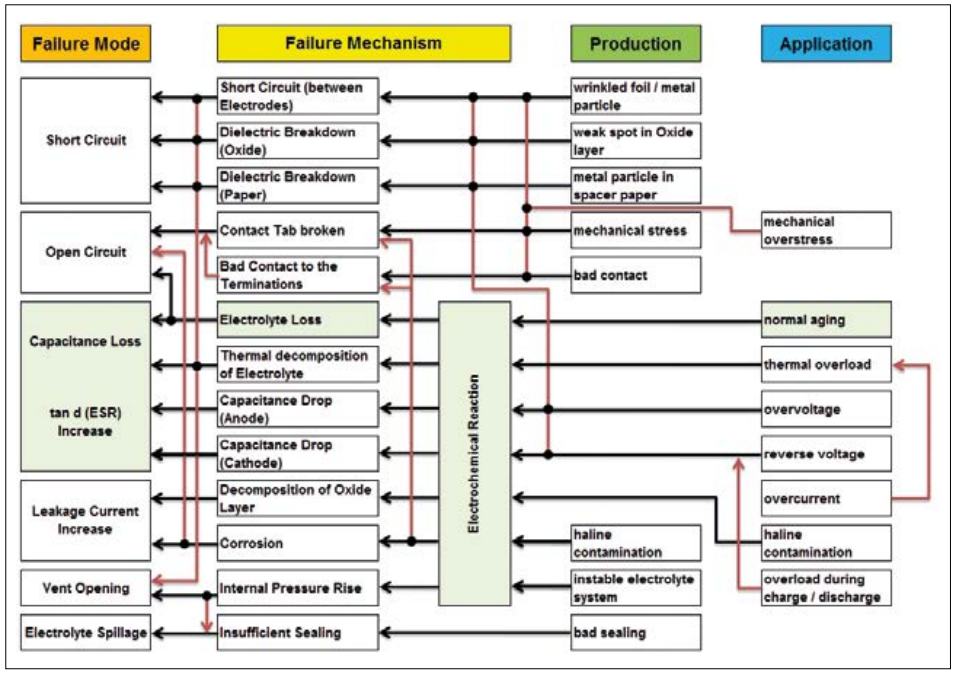


Figure 4. Types et mécanismes de panne.

Types et mécanismes de panne

La panne classique d'un électrolytique qui a vieilli normalement est de type paramétrique : faible capacité ou ESR trop élevée (**fig. 4**, en vert clair).

La panne (fig. 4) peut être causée par l'application ou un défaut de production (pannes rares sur le terrain). En effet la pureté des matériaux de base et le niveau de qualité des procédés de production ont été continuellement améliorés ces dernières années. Souvent, les pannes peuvent être expliquées par un changement défavorable de l'environnement de fonctionnement (température ambiante, courant ondulatoire, tension de fonctionnement, vibrations, contrainte mécanique, etc.) qui parfois ne peut être ni prédit ni empêché.

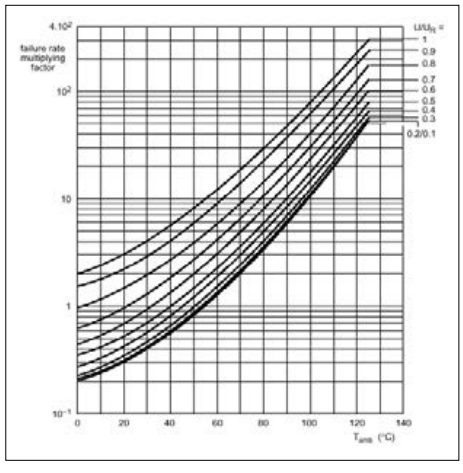


Figure 5. Facteurs multiplicatifs des taux de panne (MIL HDBK-217F).

Estimation du taux de panne

L'utilisation des meilleurs matériaux avec des procédés de fabrication de première qualité couplée à un système de contrôle de qualité efficace n'empêche pas les pannes aléatoires des composants sur le terrain. Pour estimer les taux de panne, on se réfère souvent au manuel militaire MIL-HDBK-217F, même s'il s'appuie sur des données anciennes. Les taux de panne y dépassent d'un facteur de 10 à 100 ceux observés sur le terrain pour les condensateurs Jianghai. Malgré cela, les données du MIL-HDBK-217F et ses calculs nous informent sur la dépendance des taux de panne à la température ambiante et la tension de fonctionnement (**fig. 5**). Les taux de panne sont normalisés pour une température ambiante de 40 °C et une tension de fonctionnement égale à 50% du maximum spécifié.

Pour obtenir des données sur la fiabilité à partir d'essais en laboratoire, il faudrait un effort phénoménal : des milliards d'heures de fonctionnement et le test d'un million de composants ; le coût de la main-d'œuvre serait élevé. À la place, Jianghai utilise des données sur les pannes de terrain chez les clients, couplées aux paramètres de l'application (température, courant ondulatoire, tension de fonctionnement). À l'aide de ces données, des données de production et des essais en laboratoire, les taux de panne par unité de temps sont estimés avec un effort raisonnable. L'ordre de grandeur du taux de panne sur le terrain est de 0,5 à 20 FIT.

Il est facile de calculer le MTBF (*Mean Time Between Failures,* NdT : temps moyen entre les pannes) à partir du taux FIT : MTBF = 1 / FIT. Le MTBF ne garantit pas une durée minimale avant la première panne : c'est le temps moyen avant qu'il ne reste qu'environ 37% des composants en état de fonctionnement (la fonction de distribution des pannes est de forme exponentielle).

Facteurs affectant la fiabilité

La fiabilité (ainsi que la durée de vie) des condensateurs électrolytiques de n'importe quelle marque ou type dépend de manière non linéaire de la température, du courant ondulatoire et de la tension de fonctionnement. De petites variations de ces paramètres affectent fortement les caractéristiques globales de ces composants. Il faut un circuit conçu avec soin pour atteindre le niveau de fiabilité requis d'un appareil :

- **Complexité** – réduire le nombre de composants améliore la fiabilité.
- **Contraintes** – la température, le courant ondulatoire et la tension de fonctionnement, parfois combinés à des contraintes mécaniques (vibrations…), nécessitent des compromis quant au coût ou la taille. Autant que possible, minimisez les contraintes thermiques : pour chaque augmentation de température de 10 K, le taux de panne des électrolytiques double !
- **Fiabilité des composants individuels** – lorsque vous sélectionnez des composants, comparez les prix ET la fiabilité. Les composants très fiables sont plus chers.

Pour une bonne utilisation

La majorité des pannes de condensateurs électrolytiques observées sur le terrain ne sont pas dues aux pannes aléatoires classiques. Au-delà de l'influence qu'a le fabricant, c'est l'utilisateur qui doit garantir de bonnes conditions de fonctionnement à travers une conception robuste, des processus de manipulation et fabrication précautionneux et une influence

modérée de l'environnement. Pour bien utiliser les condensateurs électrolytiques, la liste suivante est un bon début :

Transport et stockage

Les boîtiers des condensateurs électrolytiques (tout en aluminium) et leurs joints d'étanchéité (caoutchouc) sont mous et élastiques. N'utilisez pas de composants avec des dommages évidents. Il est malheureusement courant de constater une contamination aux halogènes (particulièrement les bromures utilisés pour stériliser les expéditions à l'international). Cela s'applique à la fois à l'expédition des composants individuels et au transport de produits finis.

Montage et assemblage

Il faut éviter de pousser, tirer ou tordre les broches des condensateurs électrolytiques (particulièrement les modèles radiaux). Vous endommageriez les contacts internes des feuilles d'anode ou de cathode.

Les colles, résines et laques ne doivent pas contenir d'halogènes. Il faut conserver, près des joints d'étanchéité des électrolytiques, un contact avec l'air ambiant afin d'empêcher la formation d'un microclimat confiné au-dessous du condensateur (risque de corrosion). Ne *routez* pas de pistes conductrices sous un condensateur électrolytique. N'utilisez jamais d'électrolytiques pour saisir un circuit imprimé (ce ne sont pas des poignées) !

Soudure

Respectez les températures limites de soudure recommandées par le fabricant afin d'éviter tout dommage (gonflement, perte de durée de vie ou destruction thermique de l'électrolyte). Cela s'applique en particulier au traitement des CMS avec des procédés sans plomb (les températures des profils de soudure sont plus élevées).

Fonctionnement

Lors des mises sous et hors tension, des surtensions transitoires supérieures à la tension de formage ou inverse, dues aux charges inductives, peuvent se produire. Même appliquées une seule fois, elles endommageront un condensateur électrolytique de manière permanente, une conception appropriée permet de les éviter.

Les excès de contraintes mécaniques en fonctionnement (p.ex. auto-résonance) peuvent causer la rupture des contacts. Solution : collez ou déplacez les condensateurs sur le circuit imprimé.

Toute augmentation de 10 K de la température ambiante double le taux de panne et diminue de moitié la durée de vie. Il vaut mieux placer les condensateurs électrolytiques loin des sources de chaleur (dissipateurs, inductances de puissance, etc.).

En résumé

La fiabilité individuelle des condensateurs électrolytiques à l'aluminium influence celle des appareils électroniques qui les utilisent. Il faut avoir une connaissance étendue de certains des paramètres clés de ces composants pour garantir une conception fiable des appareils électroniques.

Nous avons défini la fiabilité et les principaux facteurs qui l'affectent. Suivez nos conseils pratiques pour utiliser de façon optimale les condensateurs électrolytiques. Leur mise en pratique dépend du produit et de l'application. N'hésitez pas à consulter votre fournisseur pour vous aider pendant la phase de conception. ⏮

(150250 – version française : Kévin Petit)

Profil de l'entreprise

Jianghai Europe Electronic Components GmbH dont les bureaux et entrepôts se trouvent à Krefeld (Allemagne) comprend le service clientèle européen de *Nantong Jianghai Capacitor Co., Ltd.* (*Jianghai*) à Nantong, Chine. Jianghai a été fondée en 1958 à l'emplacement du siège actuel — à environ deux heures de route au nord de Shanghai. Dans ses premières années, Jianghai a développé et produit des produits chimiques spéciaux (p.ex. solutions d'électrolyte). En 1970, la production de condensateurs électrolytiques a démarré et lors des années suivantes, des installations de production de feuilles d'anode pour basses et hautes tensions ont complémenté la gamme de Jianghai. Étant le premier producteur en Chine, Jianghai est l'un des plus gros fabricants mondiaux de condensateurs électrolytiques radiaux, *snap-in (*NdT : qui s'encliquettent) et à visser.

L'auteur

Le Dr. Arne Albertsen a étudié la physique, plus particulièrement la physique appliquée à l'université de Kiel en Allemagne. Après son diplôme (1992) et sa thèse de doctorat (1994), tous deux sur l'analyse stochastique de séries temporelles dans un système de transport biophysique à membrane, il a suivi une carrière industrielle dans la construction d'usines spécialisées de traitement des eaux usées et technologies de production d'énergies renouvelables. En 2001, il a commencé à travailler avec les principaux fabricants de composants électroniques tels que *BCcomponents*, *Vishay*, et *KOA*. Il a occupé des postes de *manager* en conception, vente et marketing pour les composants passifs et actifs discrets jusqu'à ce qu'il rejoigne *Jianghai Europe Electronic Components* en novembre 2008. À son poste actuel, *Manager Sales and Marketing*, le Dr. Albertsen s'occupe du service clientèle pour les comptes OEM et distributeurs européens.

Références

[1] Albertsen, A., *Lebe lang und in Frieden! Hilfsmittel für eine praxisnahe Elko-Lebensdauerabschätzung*, Elektronik Components 2009, 22-28 (2009).

[2] Both, J., *Aluminium-Elektrolytkondensatoren, Teil 1 — Ripplestrom and Teil 2 — Lebensdauerberechnung*, BC Components, 02/2010, 2000.

[3] Stiny, L., *Handbuch passiver elektronischer Bauelemente*, Franzis Verlag, Poing, 2007.

[4] Venet, P., A,. Lahyani, G. Grellet, A,. Ah-Jaco, *Influence of aging on electrolytic capacitors function in static converters: fault prediction method*, Eur. Phys. J. AP 5, 71-83 (1999).

Bienvenue dans Elektor Labs

C'est dans Elektor Labs que les projets grands et petits, analogiques et numériques, vieux jeu et *méga cool* prennent forme pêle-mêle pour vous permettre de les réaliser à votre goût.

Notre offre : La célébrité

La plupart des électroniciens créatifs sont modestes. Trop discrets. Ce n'est pas parce qu'elle est griffonnée sur un rond de bière qu'une trouvaille épatante ne mériterait pas l'attention. Elektor Labs vous aide à affiner le fruit de vos études jusqu'à la perfection. Notre rédaction et nos illustrateurs donneront à votre création l'ampleur éditoriale qu'elle mérite, mais c'est toujours votre nom qui figurera en tête de l'article, car c'est *vous* l'auteur du projet. En plus, vous serez rémunéré honorablement, même si le plus grand honneur restera celui d'être publié en plusieurs langues et lu dans le monde entier. Auteurs de livres, de blogs ou de vidéos, vous ne serez pas moins bien traités. Étudiant(e)s et jeunes électronicien(ne)s, une publication dans une revue comme Elektor ne déparerait pas votre C.V. !

Notre histoire

Elektor Labs existe depuis les années 1970. En ce temps-là, c'est la même personne qui soudait les circuits et écrivait les articles. Le labo n'a pas seulement vu arriver et passer le transistor, le circuit intégré, le microprocesseur et les composants montés en surface, mais il s'est toujours porté à l'avant-garde des nouvelles vagues technologiques pour diffuser la bonne parole et les rendre accessibles au plus grand nombre.

Notre équipement

Il y a l'eau courante, le gaz, l'électricité et des machines à café. Trois des plus belles pièces du château d'Elektor sont réservées au labo, mais nous avons bien du mal à ne pas envahir le reste. Nos paillasses débordent de cartes, de composants en vrac, d'accessoires mécaniques, de prototypes et d'un fatras ingérable. L'appareillage lourd est heureusement dans un local séparé.

Nos produits

Nos produits sont dans le magazine et sur nos sites. Nous produisons texte et illustrations utilisés par la rédaction pour assembler les articles, mais surtout les circuits imprimés, certains assemblés prêts à l'emploi, ainsi que le logiciel et les composants programmés, des kits, des modules, des outils, des accessoires, des vidéos et un service d'info technique.

Nos principes

Les réalisations et les produits sortis des tuyaux de Labs répondent tous à des exigences sévères. Les produits des articles du magazine estampillés LABS doivent fonctionner avec l'appareillage étalonné disponible au labo. La correspondance entre schéma et liste de composants doit être parfaite. Les kits sont soumis à des tests par échantillonnage périodique. Nous suivons la directive ROHS et les autres normes de sécurité applicables à notre situation. Les erreurs constatées font l'objet d'une publication.

Nos webinaires

Autrefois nos ingénieurs les plus bavards testaient leurs prototypes en discutant. À la vue d'un micro, ils se taisaient. Pour les faire parler sur elektor.tv, il a fallu un talent de reporter. Elektor Labs présente aussi des webinaires où l'on parle bien et beaucoup. Ils sont annoncés dans notre lettre d'information Elektor.POST!

Nos experts et nos concepteurs

L'équipe est formée d'électroniciens expérimentés de tout plumage qui, avec ou sans prestigieux diplômes, cumulent quelque 200 années d'expérience en électronique. En plus, Labs est au cœur d'un réseau d'experts consultés ponctuellement, notamment quand ça coince.

elektor labs

Sharing Electronics Projects

Search

Home Proposals ➡ In Progress ➡ Finished

Log in

Plus on est de fous d'électronique, plus on rit !

Notre site communautaire elektor-labs.com est le port I/O bidirectionnel idéal pour mettre en vedette **votre projet** et pour suivre ceux des autres. Il est partagé par des milliers d'électroniciens comme vous. Avec eux et comme eux, ouvrez votre atelier personnel sur le monde. En circulant, vos idées s'enrichiront, vos circuits s'amélioreront. Les projets les plus suivis sur **www.elektor-labs.com** sont mûris par l'équipe d'Elektor-Labs et certains deviennent des articles (rémunérés !) dans le magazine.

Read/Write ?

Si vous souhaitez publier sous votre nom et pour votre bénéfice un projet dans ce magazine, en quatre langues et avec la présentation soignée d'Elektor, pour être lu par des dizaines de milliers d'électroniciens dans le monde, **rejoignez la communauté avec une carte de membre GREEN ou GOLD** (www.elektor.fr/membres). Les membres peuvent publier des projets sur le site elektor-labs.com, les autres se contentent de regarder.

bienvenue dans la section CRÉER

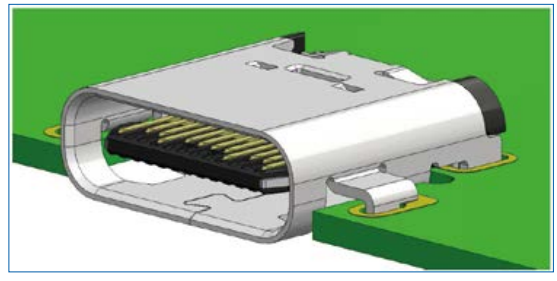
Clemens Valens, Elektor Labs

Créer pour communiquer — communiquer pour créer

D'après l'auteur d'un de mes livres de chevet consacré à l'innovation, les Bell Labs, les anciens laboratoires de R&D de la compagnie américaine de téléphonie AT&T, étaient si inventifs qu'ils auraient même inventé le mot innovation. L'auteur donne comme exemple de la créativité sans limites des Bell Labs (notez le pluriel) l'invention du transistor et des satellites. Bien que ce livre ne m'ait pas appris grand-chose sur l'innovation, il m'aura fait réaliser que la plupart des progrès, pour ne pas dire tous, proviennent des télécommunications. La technologie d'aujourd'hui a été créée parce les gens veulent et ont besoin de communiquer. Ils aiment se parler (parfois même ils s'écoutent), veulent partager toujours plus d'informations avec toujours plus de monde, et pour cela nous dépensons quantité d'énergie et de ressources tout en repoussant sans cesse les frontières de la technologie. Le partage fait naître de nouvelles idées, ces idées sont reprises ailleurs, et c'est ainsi que se développe la technologie. Si vous pouvez enfourner un surgelé dans votre micro-ondes avant de souffler dessus devant votre télé, c'est bien parce qu'il y a quelques décennies quelqu'un de la côte Ouest voulut un jour communiquer avec quelqu'un de la côte Est. Il en va de même des projets de ce numéro : ils ont été rendus possibles par des avancées techniques nées d'un désir et d'un besoin tout humain de communiquer. D'une certaine façon ces projets communiquent aussi entre eux. Et nous les publions, car nous souhaitons communiquer avec vous.

Universal Universal Parallel Peer-to-Peer

La nouvelle norme USB de type C prend en charge le mode USB *SuperSpeed* à 10 Gb/s (USB 3.1), peut délivrer jusqu'à 100 W, et la forme physique du connecteur est nouvelle. Sa prise et sa fiche ne peuvent pas être accouplées directement avec celles des autres connecteurs USB.
USB signifie *Universal Serial Bus*, soit Bus Série Universel. *Bus* ? Deux périphériques par câble, pair à pair, peut-on vraiment appeler ça un bus ? *Série* ? L'USB de type C a trois paires de signaux, n'est-ce pas une sorte de parallèle ? *Universel* ? Soit, mais uniquement si vous convenez qu'une solution universelle c'est une norme différente pour chaque application. USB-C n'a en fait d'universelle que sa compatibilité avec les types A et B. Et puisqu'elle est aussi supposée être « prête pour le futur », ne devrait-on pas l'appeler *Universal USB* ? Ou U-U-P-P-2-P ? U²P²2P ? Prête pour le futur, jusqu'à ce que débarque quelque chose de mieux, un type D par exemple…



Concepteurs, téléchargez la spécification avant qu'elle ne (re)change :
www.usb.org/developers/docs

(150293-I – version française : Hervé Moreau)

Arduino en esclave I²C

Clemens Valens, Elektor.Labs

Le vénérable bus I²C n'a jamais été aussi populaire qu'aujourd'hui. La plupart des microcontrôleurs modernes possèdent une interface matérielle I²C ou permettent son implantation logicielle, et de nombreux capteurs, convertisseurs AN/NA, mémoires, horloges en temps réel et afficheurs, en sont également dotés.

Un bus I²C relie deux ou plusieurs dispositifs dont un, et un seul à la fois, est le maître ; les autres sont les esclaves. Seul le maître, en général un microcontrôleur, s'adresse aux esclaves qui ne peuvent donc pas parler entre eux directement.

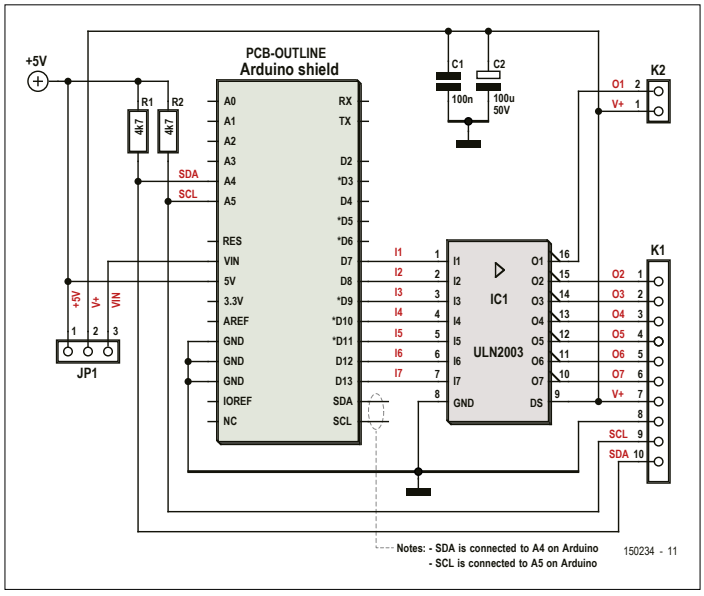


Figure 1. Le circuit d'essai utilisé pour jouer avec mes esclaves.

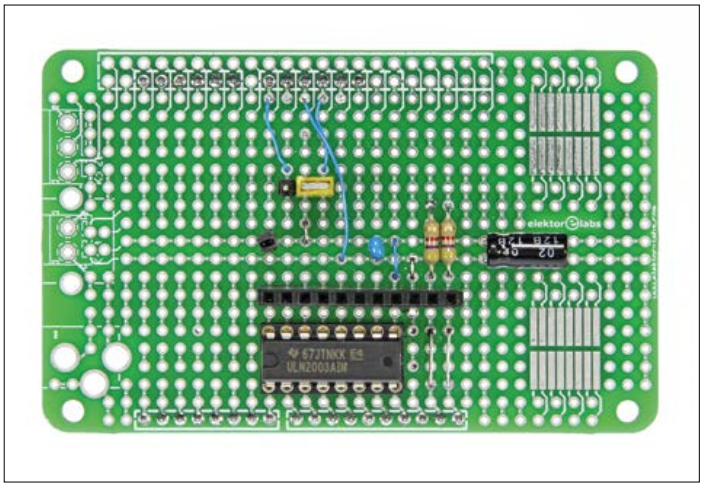


Figure 2. J'ai assemblé le circuit sur notre nouvelle carte de prototypage ELPB-NG (présentée ailleurs dans ce numéro).

Alors, comment faire communiquer deux ou plusieurs microcontrôleurs reliés par un bus I²C ? Ils ne peuvent pas tous être maîtres en même temps, et de toute façon un maître ne peut dialoguer qu'avec des esclaves. Eh bien, puisqu'il ne peut y avoir qu'un seul maître, les autres doivent être esclaves (ou être passifs, c.-à-d. ne pas interférer avec le bus). Pour maîtriser le concept d'esclave, le mieux est d'aller se documenter sur « l'aire de jeu » Arduino.

On y découvre que configurer Arduino en esclave est simple : il suffit de passer à la fonction `Wire.begin` de la bibliothèque *Wire* (qui traite des communications I²C) l'adresse d'esclave choisie. Et pour recevoir des données ou des instructions du maître, et éventuellement en envoyer, on dispose de deux fonctions de rappel (*callback*), une pour l'envoi, l'autre pour la réception. Soit, dans le croquis :

```
Wire.begin(I2C_SLAVE_ADDRESS);
Wire.onReceive(receiveEvent);
// appelée lorsque le maître envoie des données
Wire.onRequest(requestEvent);
// appelée lorsque le maître demande des données
```

C'est à vous qu'il revient de fournir les fonctions `receiveEvent` et `requestEvent` (noms arbitraires bien sûr, choisissez ceux que vous voulez). La première est appelée par le périphérique I²C du µC lorsque le maître envoie des données à l'esclave, la seconde lorsque le maître sollicite des données de l'esclave. Après l'appel de `receiveEvent`, `Wire.read` permet de lire les données reçues. Lorsque `requestEvent` est appelée, il faut envoyer des données avec `Wire.write`.

L'ennui ici est que l'esclave ne sait pas combien d'octets veut lire le maître. On peut résoudre ce problème de deux façons :

1. définir un protocole de communication de haut niveau d'après la fiche technique concernée ;
2. écrire des données jusqu'à ce que le maître interrompe la connexion.

La première solution est bien sûr la meilleure et celle qui est recommandée, mais elle demande un certain effort. La seconde ne fonctionne que si le périphérique I²C du µC possède les qualités requises.
Gardez à l'esprit que les fonctions `receiveEvent` et `requestEvent` doivent être aussi rapides que possible (ce qui ne veut pas forcément dire être aussi courtes que possible) pour ne pas risquer de ralentir le bus, provoquer des pertes de données, voire les deux.

J'ai préparé deux croquis afin que vous puissiez expérimenter : un contrôleur de moteur pas à pas esclave I²C, et un pont série-vers-I²C. Ils font communiquer par bus I²C deux cartes Arduino et commandent le moteur depuis le moniteur série. Vous les trouverez sous www.elektor.fr/150243.

(150243-I – version française : Hervé Moreau)

capteur I²C sans fil

pour le thermomètre/hygromètre à tubes Nixie

Hans Oppermann

Sous l'égide d'Elektor, le thermomètre/hygromètre à Nixie célébrait en juin 2012 les noces de la lampe Nixie russe et du microcontrôleur PIC américain [1]. Il ne lui manquait qu'une liaison radio pour parfaire sa précision et sa mobilité. Hans Oppermann avait promis d'y travailler. Le voici de retour.

Les tubes Nixie engendrent la fascination. En 2012, Hans Oppermann n'avait pas caché que c'était le thermomètre à Nixie de Dieter Lues [2] du numéro d'Elektor de janvier 2011 qui lui avait donné envie de réaliser son projet. Sur sa version, il avait ajouté l'hygrométrie et affiné les caractéristiques. Il a su que ces deux articles avaient rencontré un vif succès auprès des lecteurs.

Cependant, le thermomètre/hygromètre souffre d'un défaut, la localisation du circuit ne correspond pas nécessairement à l'endroit où l'on voudrait prendre la température. En outre, si le capteur est installé sur le circuit imprimé au milieu de l'électronique, la chaleur qu'elle dégage fausse les mesures. Quand on a, en ce temps-là, dépensé vingt euros pour le capteur SHT21 [3], on s'attend au moins à des mesures précises, belle apparence ou pas ! On pourrait utiliser l'interface I²C du capteur pour le poser plus loin, mais le bus ne tolère pas une longueur de câble de plus d'un mètre ou deux, et ce n'est jamais très esthétique. Non, ce qu'il nous faut, c'est une liaison aérienne, sans fil, par radio.

Réorganisation avec module radio

Conditions préalables majeures pour cette refonte : conserver le capteur et le circuit imprimé câblé, et ne pas toucher au logiciel du contrôleur Microchip PIC16F876. Ni le capteur ni le contrôleur ne doivent donc être affectés par la modification. La liaison radio sera vue du côté du microcontrôleur de la carte à Nixie comme un capteur SHT21 (esclave I²C) et du côté du capteur comme un contrôleur (maître I²C). L'émetteur lira les données du capteur SHT21 et les transmettra par radio au récepteur, qui doit décoder les données et en refaire des signaux I²C.

Comme module radio, eu égard à ces exigences, pourquoi ne pas embaucher le RFM12B [4] ? Ce module, à la **figure 1**, est petit, bon marché, simple à mettre en œuvre et les lecteurs d'Elektor le connaissent bien, puisqu'il s'est déjà manifesté dans d'autres projets, donc disponible dans l'e-choppe [5]. Le module radio est un trancepteur, ce qui signifie qu'il fonctionne aussi bien comme transmetteur que comme récepteur, il opère dans la bande de fréquence de 433 MHz. Le RFM12B jouit encore d'un autre avantage très appréciable, sa consommation de courant est extrêmement faible. Enfin, l'ensemble émetteur et capteur reste mobile et sa pile CR2023 dure très longtemps. D'ailleurs, des deux côtés, émetteur comme récepteur, on utilise des contrôleurs PIC LF très économes en énergie. Par des réglages particuliers de l'état du port et des périodes de pause assez longues, en pratique, la pile du montage de l'auteur a une durée de vie d'environ huit mois. La qualité de la pile CR2032 a aussi son influence, évidemment.

L'émetteur

La **figure 2** montre que l'émetteur se compose du processeur PIC16LF1847, du module RFM12B et du capteur SHT21 de l'appareil d'origine (il a été débranché puis installé sur le circuit imprimé de l'émetteur). Le contrôleur ne se distingue pas seulement par son extrême sobriété en courant, mais aussi par l'infaillibilité de son interface I²C, excellence à laquelle le PIC16LF819 que j'ai utilisé précédemment

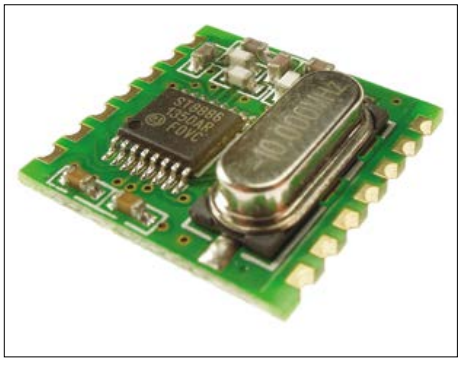


Figure 1. Petit, économe, performant : le module radio RMF12.

ne peut certes pas prétendre.
Le logiciel de l'émetteur prend les données du capteur SHT21 et les envoie par le module radio au récepteur. Le cycle d'émission est réglé sur trois minutes par souci d'économie d'énergie pour la pile, puisqu'en silence radio, la consommation tombe à 3 µA.
Les résistances R1 et R3 forment un diviseur de tension pour vérifier la tension de la pile lors de la mise en marche de l'émetteur. Les clignotements de la LED renseignent alors sur l'état de la pile : 1x : >2,5 V ; 2x : >2,7 V et 3x : >2,9 V. Après quoi la LED ne s'allume que brièvement, 50 ms, quand des données sont transmises. Aucun souci à se faire donc, il n'y a pas de gaspillage du courant de la précieuse pile bouton. Si, en cours de fonctionnement, la tension de la pile tombe sous 2,5 V, l'émetteur transmet une valeur de température de 77,7 si, du moins, son état de faiblesse le permet encore. Voir les fonctions essentielles dans l'encadré.

Le récepteur

Ne cherchez pas le schéma du récepteur, il n'y en a pas ! Non, il n'a pas été oublié, il se cache dans la figure 2. Les circuits de l'émetteur et du récepteur sont tellement semblables qu'il n'est pas nécessaire de reproduire le schéma. Les composants grisés appartiennent à l'émetteur, ils sont absents du circuit de récepteur.
Ainsi dépouillé, le récepteur se contente du PIC16LF1847 avec LED et du module radio RFM12B avec antenne. Le circuit se branche à la place du capteur sur le connecteur K2 du circuit imprimé à Nixie. La LED sert ici de témoin du temporisateur de type chien de garde (*watchdog*), qui démarre chaque fois qu'un paquet de données est reçu. Si pendant une période de 3 à 5 min aucun paquet de données n'est arrivé, la LED s'éteint. Cet état peut être temporaire, mais il est définitif si la pile de l'émetteur est à plat. La vérification est aisée, on éteint l'émetteur et on le rallume pour compter les clignotements de la LED (trois coups, la pile est bonne).
Le logiciel du récepteur imite la présence d'un capteur SHT21 et se fait voir comme esclave I²C. Bien qu'ici le traitement des données soit centré en particulier sur le SHT21, on pourrait fort bien adapter le

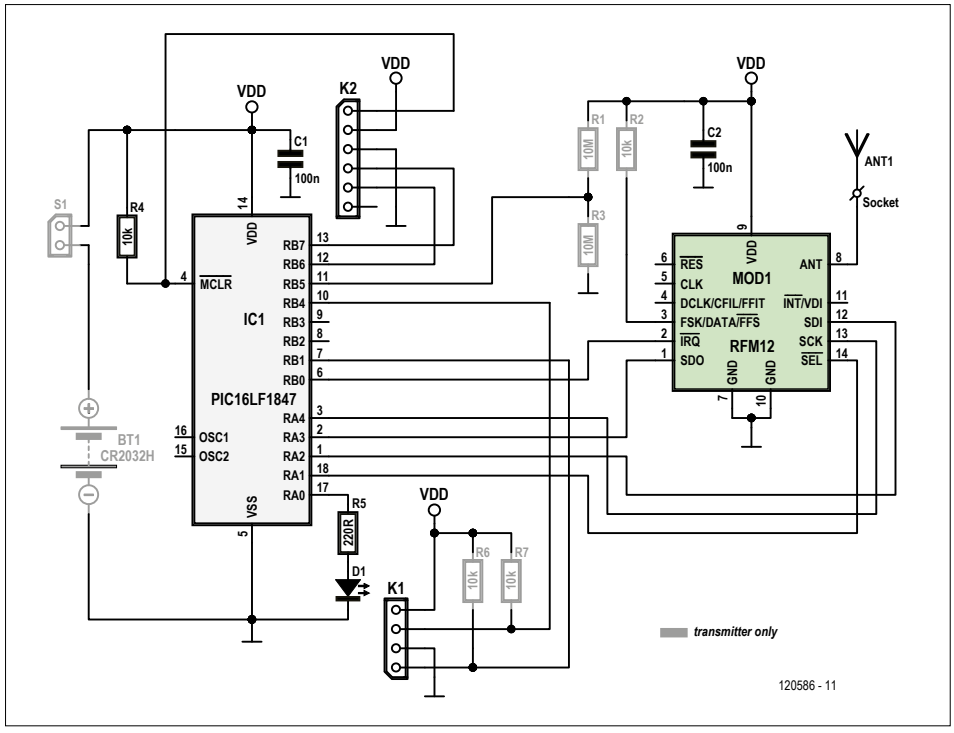


Figure 2. Des schémas pratiquement identiques pour l'émetteur et le récepteur.

Principales fonctions du logiciel de l'émetteur

- Initialisation du PIC, du RFM12B et du SHT21
- Boucle While
  - SHT21_measure:
    - --- mesure de la température avec « Hold Master Mode (HM) »
    - --- mesure de l'humidité avec « Hold Master Mode (HM) »
    - tri des octets comme s'ils venaient directement du SHT21 et non par radio.
  - Stockage des 6 valeurs à transmettre dans le tableau mres (2 octets température, CRC, 2 octets humidité, CRC)
  - rfm12_snd_msg:
    - transmission des 6 valeurs de mres par RFM12B
  - Check_voltage:
    - test de tension de la pile. Si elle est < 2,4 V, transmettre 77,7 (si c'est encore possible).
  - état de veille pendant TX_CYCLE minutes
- Fin de la boucle While.

logiciel à d'autres capteurs.
On peut aussi modifier la récurrence de la boucle dans l'émetteur avec la constante TX_CYCLE. Il va de soi qu'il faut alors changer aussi la constante du même nom dans le récepteur pour adapter le temporisateur de type chien de garde.
Le micrologiciel du récepteur est encore plus simple que celui de l'émetteur (voir encadré).

Si vous voulez programmer vous-même les deux PIC16LF1847, il faut régler les fusibles comme dans le **tableau 1**.

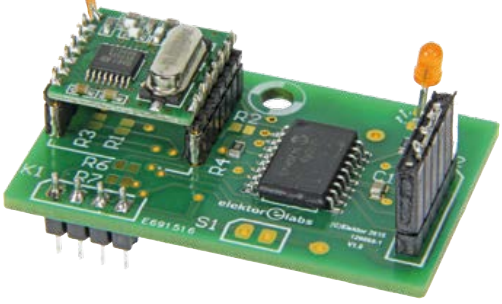


Tableau 1. Réglage des fusibles pour la programmation des contrôleurs

réglage des fusibles	120586-41 récepteur	120586-42 émetteur
Oscillator	internal RC, I/O on oscillator pins	
Watchdog timer	ON	
Protect	OFF	
LVP	OFF	
PUT	ON	OFF
Brownout detection	OFF	
Config word 1	39DC	39FC
Config word 2	1EFF	

La construction

Même schéma, même circuit imprimé : puisque les circuits sont quasi identiques avec un PIC (dont le micrologiciel est différent, lui) et un module radio, le tracé des pistes est identique pour l'émetteur et le récepteur. La paire de cartes nues est disponible dans l'e-choppe [6]. Il en va de même pour les contrôleurs PIC programmés avec le logiciel adapté à leur fonction. L'implantation ne varie que de quelques composants passifs entre l'émetteur et le récepteur. N'oublions pas le connecteur à six contacts K2 qui fournit une interface de programmation *in situ* des PIC16LF1847, compatible broche à broche avec le connecteur du PICkit 2/3.

Le garnissage des deux cartes, dont le dessin est à la **figure 3**, n'a rien de spectaculaire. Tous les composants, CMS pour la plupart, se soudent sur la face qui porte la sérigraphie, seul le coupleur de pile pour l'émetteur vient en face arrière, comme on le voit clairement à la **figure 4**.

Comme la carte du récepteur doit se brancher dans l'embase femelle sur le circuit imprimé des Nixie, on soude sur K1 une embase à picots droits, mais du mauvais côté, donc sur le dos de la carte. Sur la carte de l'émetteur, en revanche, c'est une embase femelle que l'on installe et du bon côté, cette fois, de manière à pouvoir y brancher la carte du capteur par son embase à picots coudés, comme on le voit sur la photo.

Pour le module radio, on utilise des embases à picots au pas de 2 mm sur la carte, pour correspondre au connecteur du module. Deux raisons à cela : d'abord parce qu'il est situé sous le module, ensuite parce que la programmation du microcontrôleur sur la carte ne peut se faire que quand le module radio et le capteur en sont absents. C'est que la tension d'alimentation de ces deux composants est de 3,3 V, alors que celle de l'interface de programmation est de 5 V. Ce n'est pas beaucoup plus, mais suffisant pour les endommager ou les anéantir ! D'où la nécessité de brancher le module radio par l'intermédiaire d'un connecteur et de ne pas oublier de le débrancher, ainsi que le capteur et la pile, évidemment, avant toute programmation. On peut régler le PICkit 2/3 sur 3,3 V aussi, il est vrai, et d'ailleurs il passe automatiquement sur 3,3 V lors du choix du PIC16LF1847, mais avec un autre appareil de programmation ou en cas d'erreur sur le choix du contrôleur, cette précaution est fortement recommandée.

Des modules radios RFM12 existent pour 433 MHz, mais aussi pour les deux autres bandes de fréquence ISM de 868 MHz et 915 MHz (uniquement pour les Amériques). Chaque version demande une initialisation particulière, parce qu'il faut écrire la fréquence correspondante dans un des registres internes du module RFM12 lors de la mise en marche.
Le logiciel est préparé pour toutes les fréquences, il n'y a qu'à choisir celle que vous utilisez en mettant les broches suivantes au niveau haut (1) ou bas (0) :

RB6 (K2-5)	RB7 (K2-4)	fréquence
0	0	433 MHz
1	0	868 MHz
0	1	915 MHz

Lors de l'allumage, le logiciel lit les réglages et se met sur la bonne fréquence. La tension VDD (K2-1) et la masse GND

Micrologiciel du récepteur

- Réception des 6 octets mres de l'émetteur dans la fonction isr0 et stockage dans la chaîne str ou str1
- Envoi des 6 octets de str1 à la demande du maître dans la fonction sspinterupt. Cela représente le noyau du logiciel esclave.

```
sspiinterrupt:
state = i2c_isr_state();   // Donne l'état de la transmission
(read/write et quelles données ont été transmises)
Détails : voir documentation du compilateur C CCS.

Selon que le code de la fonction est 0xE5 ou 0xE3, les données sont
stockées dans l'une ou l'autre chaîne str.
```

(K2-3) se trouvent aussi sur le connecteur K2. Le plus simple est de prendre une petite embase à picots (on peut le voir facilement à la figure 5), d'y souder deux bouts de fil pour disposer des niveaux voulus sur RB6 et RB7, et puis de brancher le tout sur l'embase à picots. D'ailleurs, sur la carte du récepteur il y a une embase à picots droits (ce devrait être le cas), mais à picots courbés sur l'émetteur. Cette dernière n'est pas indispensable, vous pouvez l'oublier si vous voulez.

Encore une petite chose, mais qui a son importance : les antennes. Il est facile de les fabriquer avec des morceaux de fil d'à peu près 17 cm de long (8,5 cm pour 868 MHz et 7,8 cm pour 915 MHz), ce qui correspond au brin quart d'onde. Une pastille est prévue sur chaque carte pour les souder dans un coin.

(120586 – version française : Robert Grignard)

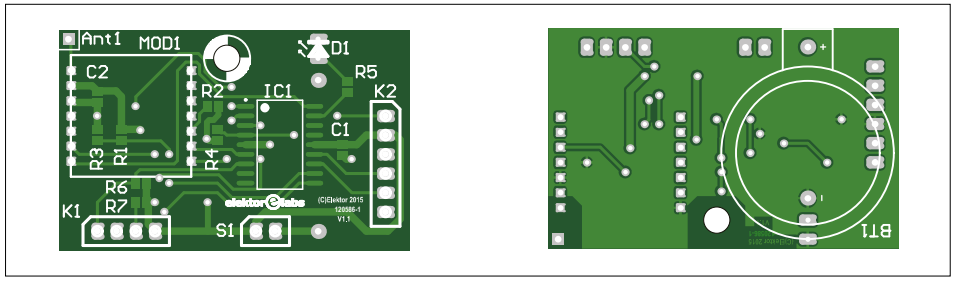


Figure 3. La même petite carte pour l'émetteur et le récepteur. Pile installée au verso pour l'émetteur.

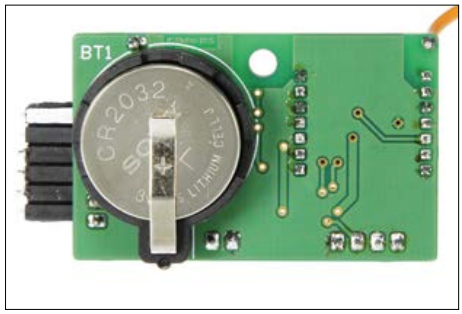


Figure 4. L'émetteur se distingue par sa pile au dos de la carte.

Figure 5. Le récepteur installé sur le circuit imprimé des Nixie en remplacement du capteur.

Liste des composants de l'émetteur

Résistances :
(toutes les résistances à 5 %, 0,125 W, CMS 0805)
R1, R3 = 10 MΩ
R2, R4, R6, R7 = 10 kΩ
R5 = 220 Ω

Condensateurs :
C1, C2 = 100 nF céramique, CMS 0805

Semi-conducteurs :
D1 = LED rouge 3 mm
IC1 = PIC16LF1847-I/SO (programmé : e-choppe réf. 120586-42)

Divers :
Mod1 = transcepteur 433 MHz RFM12B-433-S (Hope RF Electronic, dans l'e-choppe réf. 130559-91)
avec 17 cm de fil de cuivre comme antenne
S1 = interrupteur ou cavalier
K1 = embase verticale femelle à 1x4 voies
embase à 1x7 picots au pas de 2 mm (Farnell 605-8796)
embase femelle à 1x17 voies au pas de 2 mm (Farnell 547-3239)
Bt1 = coupleur de pile CR2032 (Multicomp CH25-2032LF)
pile bouton au lithium CR2032

Liste des composants du récepteur

Résistances :
(toutes les résistances à 5 %, 0,125 W, CMS 0805)
R4 = 10 kΩ
R5 = 220 Ω

Condensateurs :
C1, C2 = 100 nF céramique, CMS 0805

Semi-conducteurs :
D1 = LED rouge 3 mm
IC1 = PIC16LF1847-I/SO (programmé : é-choppe réf. 120586-41)

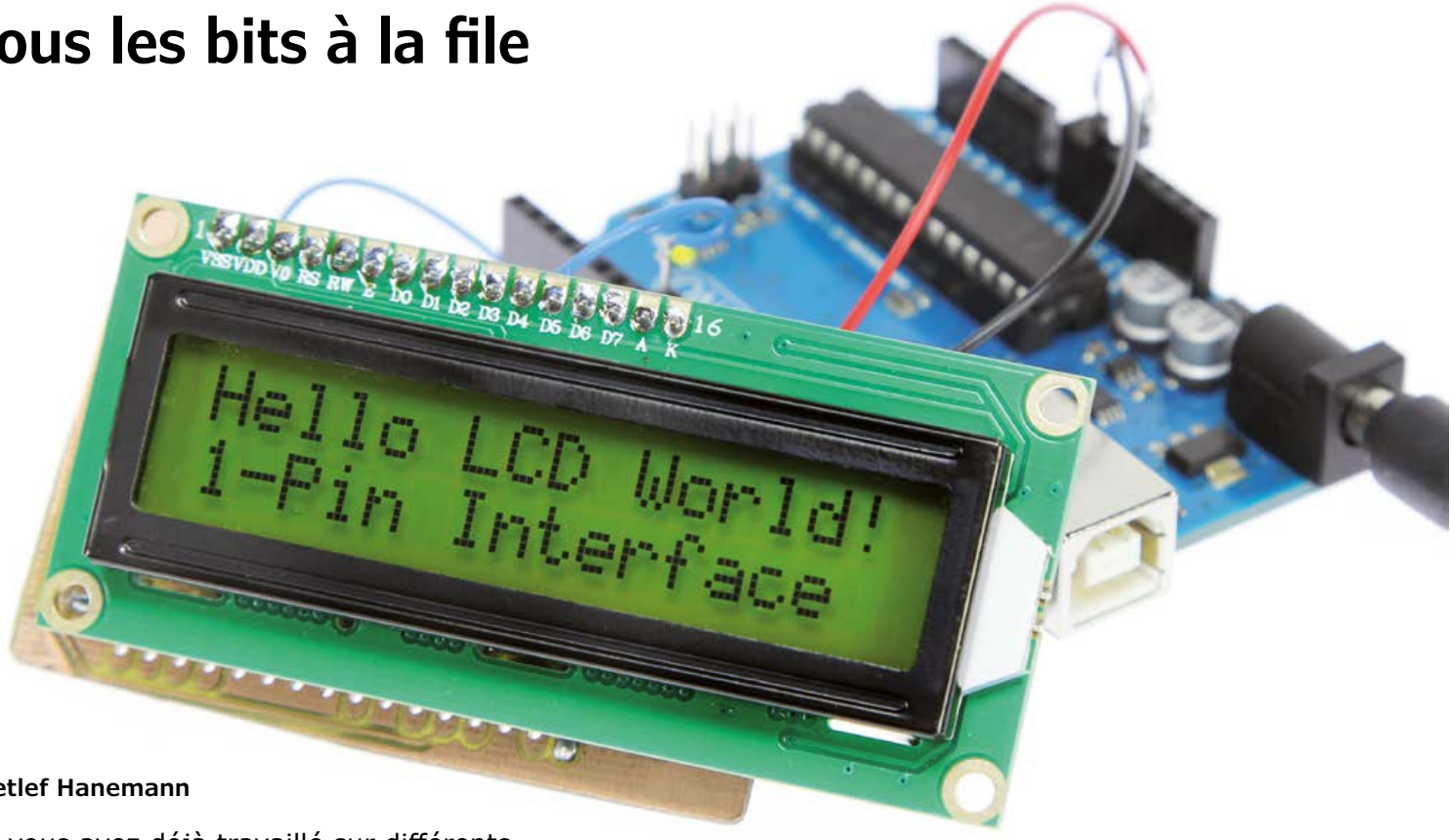
Divers :
Mod1 = transcepteur 433 MHz RFM12B-433-S (Hope RF Electronic, dans l'e-choppe réf. 130559-91)
avec fil de cuivre de 17 cm de long comme antenne
carte réf. 120586 par paire à l'e-choppe
embase à 1x7 picots au pas de 2 mm (Farnell 605-8796)
embase femelle à 1x17 voies au pas de 2 mm (Farnell 547-3239)

Liens

[1] Thermomètre/hygromètre à tubes Nixie, Elektor 06/2012, www.elektor.fr/110321
[2] Thermomètre à tubes Nixie, Elektor 01/2011, www.elektor.fr/090784
[3] Capteur SHT-2x : www.sensirion.com/de/produkte/feuchte-und-temperatur/feuchte-temperatursensor-sht2x/
[4] Module radio : www.hoperf.com/upload/rf/RFM12B.pdf
[5] www.elektor.fr/130559-91
[6] www.elektormagazine.fr/120586

interface LCD mono-fil

tous les bits à la file



Detlef Hanemann

Si vous avez déjà travaillé sur différents projets à microcontrôleur, vous connaissez l'angoisse de la ligne d'E/S qui ne trouve plus de port pour se brancher. Que faire ? À regret, supprimer une fonction ou chercher une plus grosse puce, si elle existe, ou faire les frais d'une extension de port ? Rien de palpitant. C'est toujours plus facile avec les périphériques qui n'occupent qu'une broche de port. C'est ce que je propose de faire avec cette interface pour n'importe quel LCD.

Prenons un petit microcontrôleur comme l'ATtiny d'Atmel à huit broches, enlevons les deux broches d'alimentation et celle de la mise à zéro, il n'en reste que cinq pour les entrées/sorties, ce n'est pas beaucoup. Une plus grande puce en compte davantage, mais occupe plus de place et s'il vous faut quatre boutons-poussoirs, des LED et autres périphériques, vous serez vite à court de broches, surtout quand il y a des périphériques qui accaparent quatre lignes de données en parallèle, comme un LCD. Ajoutez à cela que le soudage est d'autant plus compliqué qu'il y a plus de broches, quand ce n'est pas la distance entre elles qui diminue. Donc même avec un plus grand microcontrôleur, pour ne pas devoir laisser tomber certaines fonctions, on ajoute une extension de port, toujours faite de registres à décalage à commander par I²C (lentement) ou SPI (ce qui occupe une ligne de plus). Ces registres ont aussi besoin de deux ou même trois broches du microcontrôleur pour convertir les données parallèles et les envoyer comme il faut à l'interface du module LCD. Il y a mieux à faire.

Caractéristiques techniques

- interface universelle pour LCD parallèle
- écriture sérielle par une seule ligne GPIO
- module LCD enfichable avec différents brochages
- compatible avec les modules LCD bon marché
- durée minimale de transfert par caractère de 0,8 ms
- compatible avec tous les microcontrôleurs

Une meilleure solution

Difficile de descendre à moins d'une seule ligne de données, mais ce serait déjà parfait. Or, un LCD reçoit des données, il n'en émet pas. Pour trouver la solution optimale, nul besoin de réinventer la roue, la conversion de série en parallèle entre le microcontrôleur et le LCD est indispensable, on peut la confier à un module qui rassemble l'électronique de conversion et l'afficheur, le tout à faible coût. Ce dispositif d'affichage peut alors se brancher sur n'importe quel microcontrôleur à peu de frais.

La solution à un seul fil

On n'a pas attendu l'internet pour savoir comment un registre à décalage peut astucieusement transmettre des données avec moins d'entrées que ce qu'il en faudrait normalement. Tout registre à décalage qui se respecte doit avoir au moins deux entrées, l'une pour les données, l'autre pour l'horloge. Cette dernière est indispensable pour déterminer à quel moment l'entrée de données est au niveau voulu. Mais en créant un mélange spécial de données et d'horloge, il est possible de les séparer avec un simple réseau RC. La **figure 1** montre comment marche le procédé. Il faut pour cela que le registre à décalage soit déclenché par un flanc et qu'il réagisse au flanc montant, c'est du moins le cas pour la plupart des puces. Le niveau de repos, sans transfert, est haut. Chacune des deux entrées est au niveau haut et rien ne se passe. Lors d'une impulsion courte par rapport à la constante de temps du passe-bas RC, l'entrée de données reste haute et ensuite, sur le flanc montant de l'impulsion négative, un niveau haut est introduit dans le registre. Si l'on veut glisser un niveau bas dans le registre, il faut simplement rendre l'impulsion négative d'entrée nettement plus longue, de sorte que l'entrée de donnée soit tombée au niveau bas. Le signal d'entrée est donc formé d'une suite d'impulsions négatives courtes et longues en fonction de la séquence de bits à transmettre. En choisissant bien les durées, la méthode est fiable.

Du fait que maintenant les données arrivent aux sorties parallèles à un rythme indéterminé, il n'est pas possible de savoir quand il faut les transférer. C'est pourquoi les registres à décalage disposent normalement d'un tampon de sortie dans lequel les données du registre sont stockées « sur ordre », à un moment précis. Il faut pour cela une troisième entrée. J'ai trouvé sur l'internet, par le lien [1], quelqu'un qui avait eu l'idée d'utiliser un second réseau RC, avec une très longue constante de temps. Les impulsions courtes et longues ne suffisent pas pour faire passer au niveau bas l'entrée qui déclenche le transfert des données. Il en faut une très longue en plus. Au total, le signal de commande se compose d'impulsions courtes (haut), longues (bas) et très longues (transfert). Cette formule marche bien, mais elle a une limitation : le dernier bit doit toujours être bas et ne peut donc pas servir à la conversion de série à parallèle.

Mais pour la communication avec un module LCD, il y a encore un autre problème : les données sont généralement envoyées en mode à 4 bits vers un LCD de deux lignes ordinaire à contrôleur Hitachi HD44780 ou compatible, donc 7 bits suffiraient amplement. Mais quand le LCD reçoit les données du registre à décalage, il attend une impulsion positive. Pour en produire une telle, il faut transmettre deux fois chaque mot de donnée [3], c'est lourd et ça prend du temps. Je me suis souvenu d'un circuit qui, avec un signal de commande un peu différent, réalise d'une autre manière le transfert

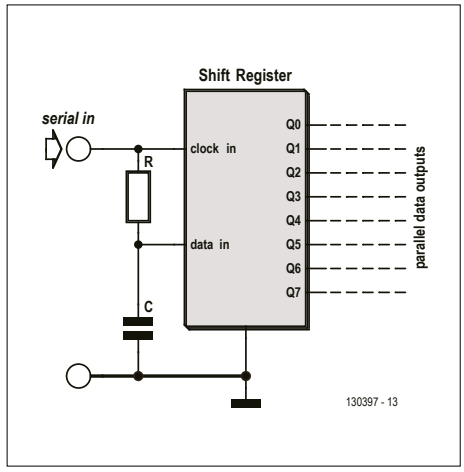


Figure 1. Le principe de la commande par une seule ligne d'E/S d'un registre à décalage au moyen d'un filtre passe-bas RC.

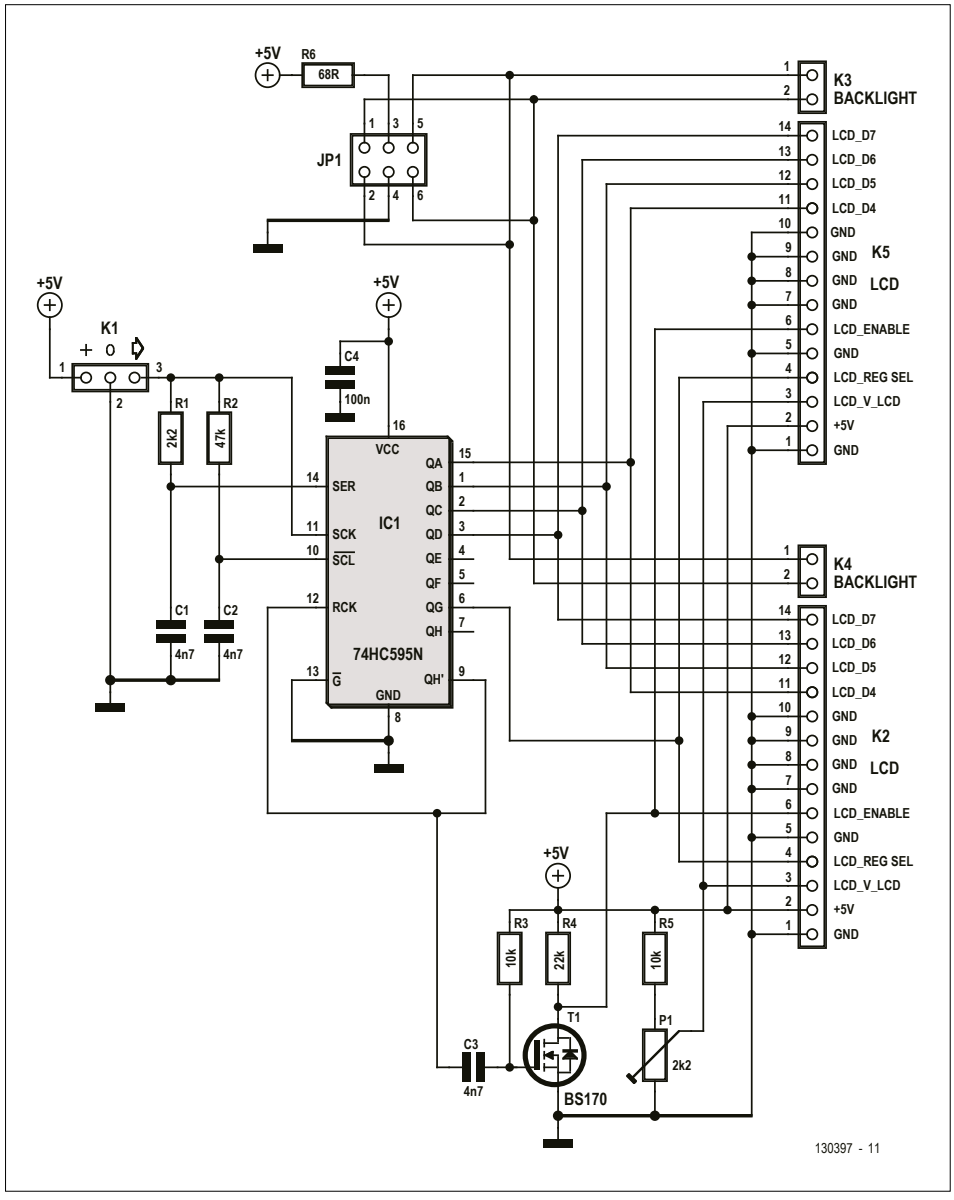


Figure 2. La solution choisie consiste à utiliser deux filtres passe-bas RC, mais aussi une particularité du registre à décalage 74HC595.

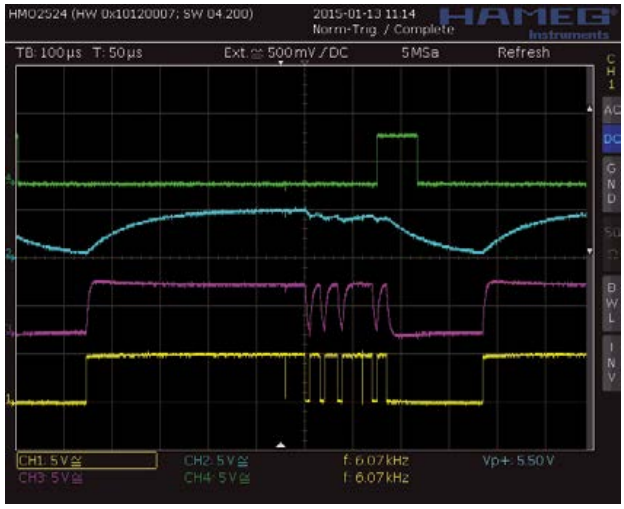


Figure 3. Oscillogramme des signaux : jaune = signal de commande, violet = entrée de données (broche 14), bleu = entrée *Reset* (broche 10) et vert = signal sur QH' (broche 9 de IC1).

de données dans le tampon de sortie et de là, dans le module LCD.

Le circuit à un seul fil

Comme on le voit à la **figure 2**, le signal de commande passe de la broche 3 de K1 à deux filtres passe-bas distincts. Pour le plus rapide avec R1 et C1, la constante de temps vaut 10 µs pour l'entrée de données, broche 14 de IC1. Le lent, composé de R2 et C2, a une constante de 200 µs environ, mais contrairement à la solution de [1], ne commande pas le transfert de données de IC1, il provoque en revanche une mise à zéro par sa broche 10. Le transfert de données est ainsi résolu d'une autre façon.

Le transfert de données dans le tampon de sortie de IC1, c'est la broche 12 qui s'en occupe, et vers le module LCD, c'est son entrée LCD_Enable par l'intermédiaire de K2 et K5. L'impulsion positive nécessaire pour le LCD est produite à la fin de chaque transfert de mots de données par un multivibrateur monostable construit avec T1. C'est une spécificité du 74HC595 qui le permet. Le tout premier niveau haut sur le bit 7 fournit le dernier coup d'horloge pour l'introduction des données dans le tampon de sortie de IC1, en raison du fait que la broche 12 de IC1 est reliée à sa sortie QH', laquelle est prévue pour la mise en cascade de ces puces et donc reliée directement au registre à décalage. L'entrée de mise à zéro, $\overline{SCL}$ en réalité, de IC1 n'efface que l'état du registre à décalage, mais pas son tampon de sortie. Quand, après l'introduction de tous les bits au moyen d'une longue impulsion négative de commande sur la broche 10, on provoque la mise à zéro, QH' est mise au niveau bas, mais QH et les autres sorties restent intactes. Par C3 et R3, le monostable T1 est activé et assure le transfert des données au LCD. Après un temps de repos pour R2/C2, on peut envoyer le mot de données suivant. La figure 3 montre la chronologie des différents niveaux de signal aux endroits concernés du circuit.

J'ai choisi les durées des impulsions pour le bas, le haut et la mise à zéro pour que la transmission d'un mot de données de 8 bits au LCD prenne environ 1,3 ms. On voit qu'à part les quatre bits de donnée QA à QD (bits 0 à 3), il ne faut que QG pour commander le choix du registre du LCD. Deux autres broches sont disponibles dans un autre but : QE et QF. Lors de la finition et du développement du circuit imprimé au laboratoire Elektor, Luc Lemmens a prévu deux rangées de broches pour le raccordement de différents LCD. Les modules LCD habituels à deux lignes se distinguent par la position de leur connecteur, dessus, dessous ou pas du tout, ainsi que par leur sens de progression. Autre différence, la polarité de l'éclairage et parfois l'ordre des broches.

Il n'est pas possible de savoir si l'interface s'adapte réellement à tous les modules. Luc assure qu'avec K2 plus K4 ainsi que K3 et K5 et le choix de la polarité de l'éclairage avec JP1, il a amélioré souplesse et compatibilité. En outre, Luc a réduit les constantes de temps des réseaux RC pour stabiliser le transfert de données. La largeur de l'impulsion « basse » ne doit pas être

Liste des composants

Résistances :
(toutes 5 % 0,25 W)
R1 = 2,2 kΩ
R2 = 47 kΩ
R3, R5 = 10 kΩ
R4 = 22 kΩ
R6 = 68 Ω
P1 = pot. ajust., 2 kΩ,
grand modèle horizontal

Condensateurs :
C1, C2, C3 = 4,7 nF au pas de 5 mm
C4 = 100 nF, au pas de 5 mm

Semi-conducteurs :
T1 = BS170
IC1 = 74HC595, DIL

Divers :
K1 = embase à 3 picots au pas de 2,54 mm
K2+K4, K3+K5 = embase à 16 picots
ou femelle à 16 voies* au pas de 2,54 mm
JP1 = embase à 2x3 picots
au pas de 2,54 mm
2 cavaliers pour JP1 au pas de 2,54 mm
module LCD à 2 lignes de 16 caractères*
circuit imprimé réf. 130397-1*

* voir texte

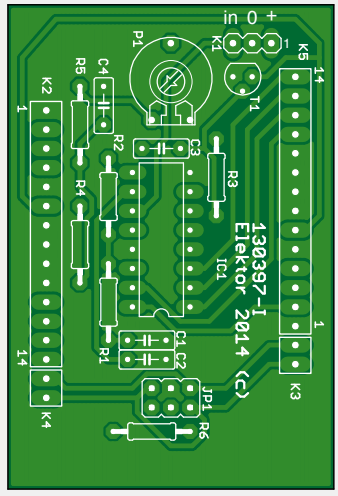


Figure 4. Le côté composants du circuit imprimé de l'interface LCD à un seul fil.

réduite dans le programme pour éviter des soucis de stabilité. Ce sont les pauses « hautes » qu'il faut allonger. Dans le logiciel d'expérimentation, Luc a ajouté un signal de déclenchement sur la broche 9 numérique d'Arduino (voyez le prototype), que vous pouvez bien sûr abandonner dans votre réalisation pour récupérer une broche.

Construction et mise en service

La **figure 4** montre la position des composants à implanter, une puce à pattes et une dizaine de composants passifs à fils, aucun CMS, pas de quoi faire trembler un fer à souder. Il faut simplement porter son attention sur l'ordre des broches lors du raccordement du LCD et savoir où poser des barrettes mâles ou femelles. Le brochage est normalement inscrit sur la carte du LCD, au besoin, consultez la fiche technique. Les **figures 5** et **6** présentent le prototype avec LCD seul et en compagnie de l'Arduino.

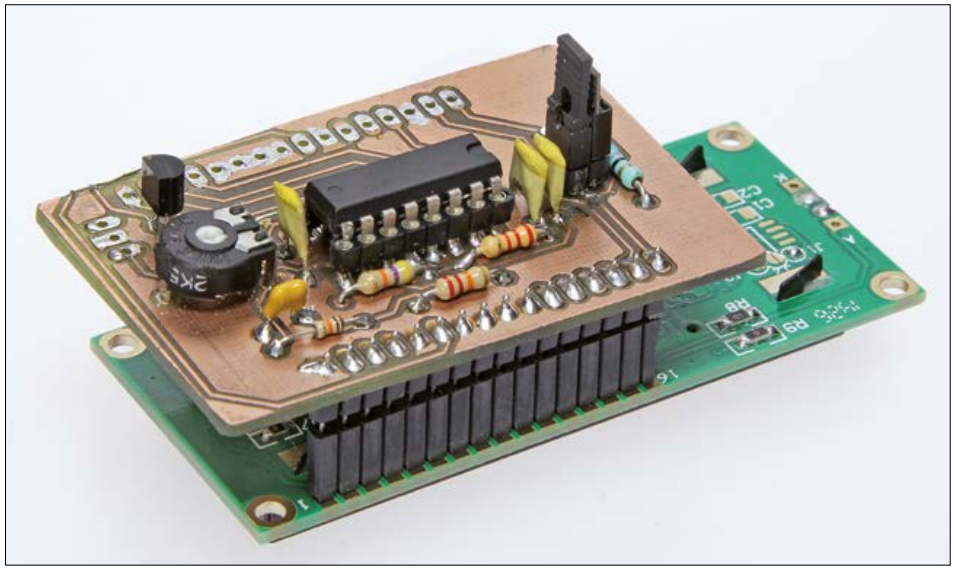


Figure 5. Le prototype de l'interface mono-fil, le module LCD est branché de l'autre côté.

À surveiller aussi le brochage et la polarité de l'éclairage du LCD. Sur JP1, deux cavaliers proches de K3 (connexion 1 avec 3 et 2 avec 4) font en sorte que sur K3 l'anode soit à l'extérieur et sur K4 à l'intérieur. C'est le contraire avec les cavaliers vers K4. C'est K1 qui assure la liaison entre la carte à microcontrôleur et son extension, la combinaison de l'interface et du LCD. Il importe ici que les tensions d'alimentation soient compatibles. Si le microcontrôleur est correctement programmé, mais que vous ne voyez rien s'afficher, pensez à régler le contraste avec P1.

Comme toujours, les données pour la gravure du circuit imprimé et le code pour Arduino sont disponibles gratuitement sur le site d'Elektor à la page de cet article [2]. Les commentaires du code sont là pour vous faciliter la vie si vous voulez l'adapter à un autre microcontrôleur. Le circuit imprimé à double face est bien entendu disponible à l'e-choppe, gravé, perforé, monté et testé. ⏮

(130397 – version française : Robert Grignard)

Liens

[1] La solution de Roman Black : www.romanblack.com/shift1.htm
[2] www.elektormagazine.fr/130397
[3] www.elektor-labs.com/node/3598

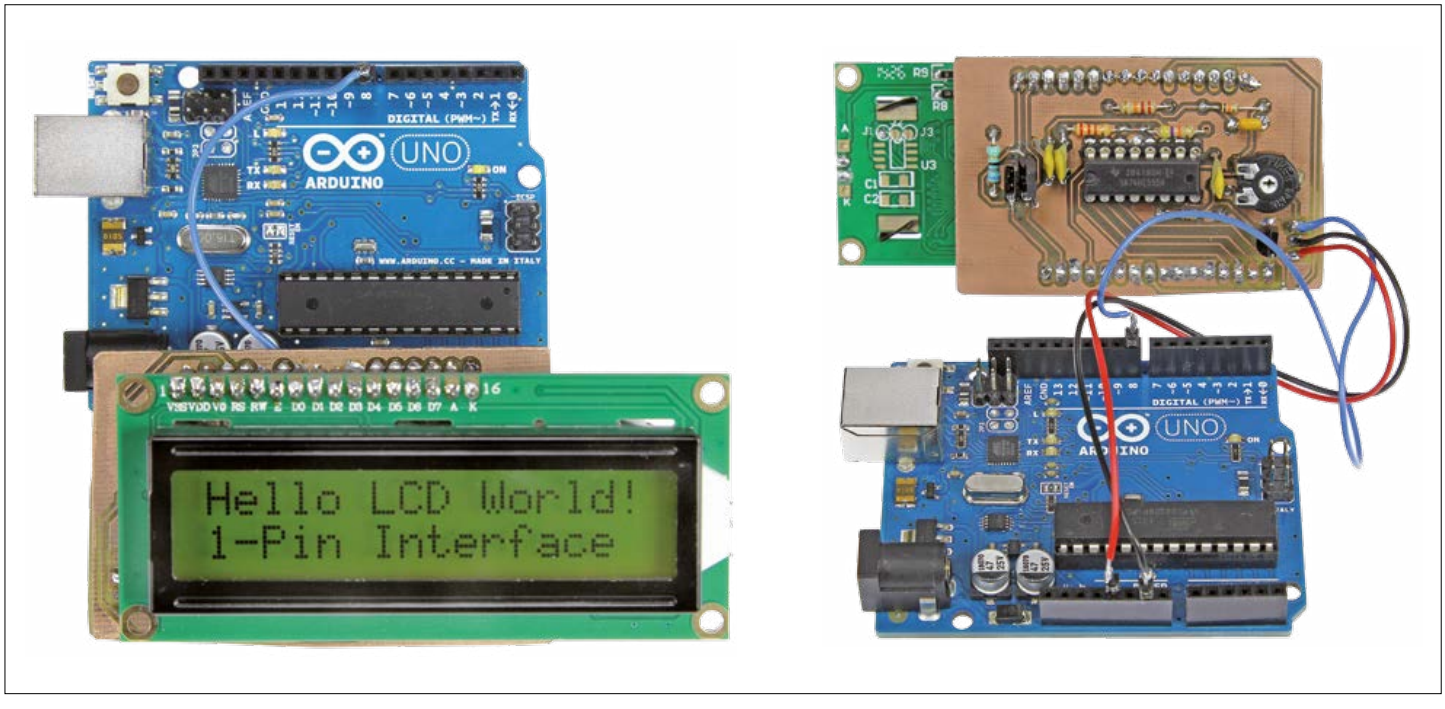


Figure 6. La carte Arduino avec l'interface mono-fil pour LCD, le tout en action.

Bougi(gu)e à LED

Victor Hugo Pachon (Colombie)

Ce circuit simple module la luminosité d'une LED pour mimer le vacillement d'une flamme de bougie. Pas de cire ici, juste de l'électronique alimentée par deux piles AA ou AAA. Pas non plus de risque de brûlure ou d'incendie, ni de fumée exaspérante lorsque vous l'éteignez.

Le circuit (**fig. 1**) comprend un petit microcontrôleur PIC12F675 programmé pour produire un signal MLI (modulation de largeur d'impulsion) dont la variation du rapport cyclique commande la luminosité de la LED. Produites de façon aléatoire en utilisant une suite binaire pseudo-aléatoire (SBPA), ces variations permettent de reproduire le vacillement d'une flamme. Chaque changement de luminosité intervient à intervalles suffisamment courts pour que l'œil soit aussi charmé que s'il regardait la gigue d'une vraie flamme dans un souffle d'air (♪ *Like a Candle in the Wind* ♪♪).

Ce petit projet se passe aisément d'un circuit imprimé. On peut par exemple monter le PIC, la LED et la résistance sur un morceau de plaque d'essai que l'on insèrera avec la pile dans un court tube de PVC, fermé ensuite par deux couvercles en plastique. Peignez le tube en blanc cassé, ajoutez quelques symboles cabalistiques, et vous obtiendrez une vraie fausse bougie.

Une grosse LED jaune de récup (5 mm) fait l'affaire, mais rien ne vous empêche d'essayer des LED de couleur différente. Une LED orange reproduit de façon assez fidèle l'aspect d'une flamme.

Côté programme, comme l'intensité lumineuse et la chute de tension aux bornes de la LED dépendent du modèle utilisé, vous devez définir les niveaux maximum et minimum du signal MLI. Le minimum est défini par la plus petite valeur produite par la SBPA (0), le maximum par la plus grande valeur (255). Le micrologiciel du PIC (à télécharger en [1]), plus précisément le fichier *asm* contenant le code assembleur (!), permet d'ajuster les valeurs MLI min/max afin d'obtenir l'intervalle adéquat. Il faudra peut-être également adapter la valeur de la résistance-talon R1 afin d'obtenir l'intensité lumineuse souhaitée. ⏮

(150072 – version française : Hervé Moreau)

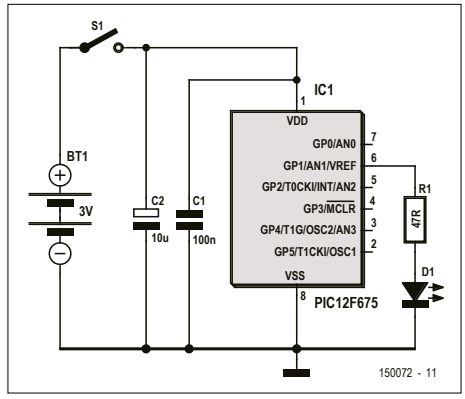


Figure 1. Un concurrent sérieux pour le concours du plus simple des circuits à PIC.

Lien

[1] Micrologiciel : www.elektormagazine.fr/150072

http://youtu.be/JKh8L83FJV8

instrument de surveillance pour batterie de 12 V

précis et peu coûteux

Ramkumar Ramaswamy (Inde)

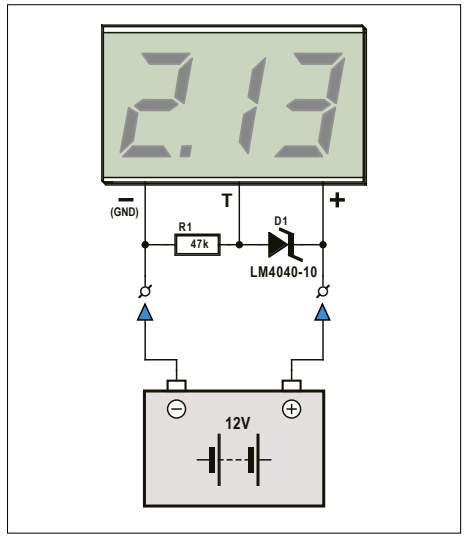
J'ai eu cette idée lorsque je cherchais un moyen de mesurer de manière précise et permanente la tension des différentes batteries de ma banque photovoltaïque. Pendant la décharge, la tension d'une batterie au plomb ne varie que peu. On voit sur le **tableau** qu'une tension (sans récepteur) de 12,6 V correspond à une charge de 100%, mais qu'avec 12,2 V, l'état de charge tombe à 60%. Il serait donc bon d'avoir un moyen pas cher et suffisamment précis pour surveiller chaque batterie. « Pas cher » aiguille vers les afficheurs proposés sur eBay pour moins de $2 ; mais leur précision de 1 à 2% est insuffisante. Tenter d'atteindre une précision de 0,1% alourdit considérablement la facture.

Le circuit de la **figure 1** est bon marché **et** précis. Ces afficheurs ont trois bornes : GND (−), Vcc (+) et une entrée de mesure (T). Le LM4040-10, une référence de tension de précision, induit une chute de 10 V très exactement, avec une précision supérieure à 0,1% ; le reste de

tension (au-delà de 10,00 V) est appliqué à la borne de mesure d'un afficheur 0-10 V qui montre le surplus de tension de la batterie au-delà de 10 V. Une valeur de 2,5 V correspond à une tension de batterie de 12,5 V. Même si sa conception est rustique, cet instrument permet une erreur acceptable de 2% de 2,5 V, soit 0,05 V, le LM4040 lui-même ajoute au maximum 0,01 V ; l'erreur totale ne dépasse donc pas 0,06 V. Parfait pour notre application.

La résistance de 47 kΩ assure l'alimentation du LM4040-10 à un courant d'au moins 60 µA, valeur requise pour un fonctionnement fiable.

(150077 - version française : Guy Raedersdorf)



tension	état de la charge
12,60+	100%
12,50	90%
12,42	80%
12,32	70%
12,20	60%
12,06	50%
11,90	40%
11,75	30%
11,58	20%
11,31	10%
10,50	0%

Figure 1. Grâce la référence de précision de 10 V, D1, le voltmètre numérique à vil prix peut servir à surveiller une batterie de 12 V avec une bonne précision.

bipeur de panne de courant

pour camping-cars et bateaux

Joachim Schröder

Quand un navigateur ou un camping-cariste ne dispose d'aucune source d'énergie mobile adéquate, ni cellules solaires ni piles à combustible, il recharge sa batterie sur une borne temporisée à pièces (ou jetons) telle qu'on en trouve sur les aires de service pour camping-cars ou sur les quais dans les ports de plaisance. Ces prises, commandées par la consommation, se coupent automatiquement une fois débitée la quantité d'électricité payée. Dans bien des cas, cette quantité d'énergie est inconnue, d'autant plus qu'il est difficile d'évaluer sa propre consommation. Il peut arriver que le courant de charge soit interrompu sans que l'on ne s'en aperçoive et qu'au lieu de démarrer une nouvelle journée avec une batterie de bord chargée à bloc, on parte avec une batterie à moitié vide.

D'où mon idée de réaliser ce petit circuit qui bipe durant quelques secondes en cas de coupure (volontaire) ou de panne de l'alimentation en 230 V. Il fonctionne grâce à un simple bloc d'alimentation de 5 V (non représenté ici) comme on en trouve au fond des tiroirs. Tant que la tension du réseau est présente, le bloc l'alimentation fournit +5 V sans broncher. Sur le circuit de la **figure 1**, le gros condensateur de 1 000 µF se charge à travers D1 et la résistance de 220 Ω. Parallèlement, par le biais de la résistance de 1 kΩ, cette tension de 5 V attaque la base du transistor PNP de sorte que ce dernier est bloqué.

En cas de disparition de la tension du réseau et par conséquent du +5 V, le transistor, épaulé par la résistance de 68 kΩ, devient passant. Le condensateur peut alors se décharger au travers du bipeur et du transistor. Le bipeur se manifeste pendant quelque 5 s, cette durée (que l'on pourra allonger par le choix d'une capacité de condensateur plus élevée) dépend du condensateur et des caractéristiques techniques du bipeur.

L'ensemble du circuit prend place, comme on le voit ci-dessus, sur un morceau de platine à trous (dimensionné au mieux), monté avec le bloc d'alimentation à l'intérieur d'un boîtier à fiche secteur intégrée, ce qui évite ainsi tout risque de contact accidentel. Une fois que l'on s'est raccordé à une borne, il reste à enficher l'appareil dans une prise libre du camping-car ou du bateau ; on sera alors averti immédiatement de la disparition (intentionnelle ou non) de la tension secteur.

(140281 – version française : Guy Raedersdorf)

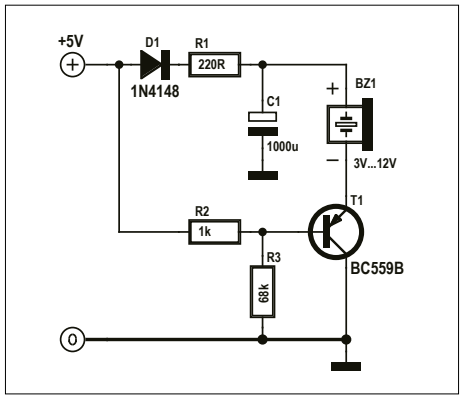
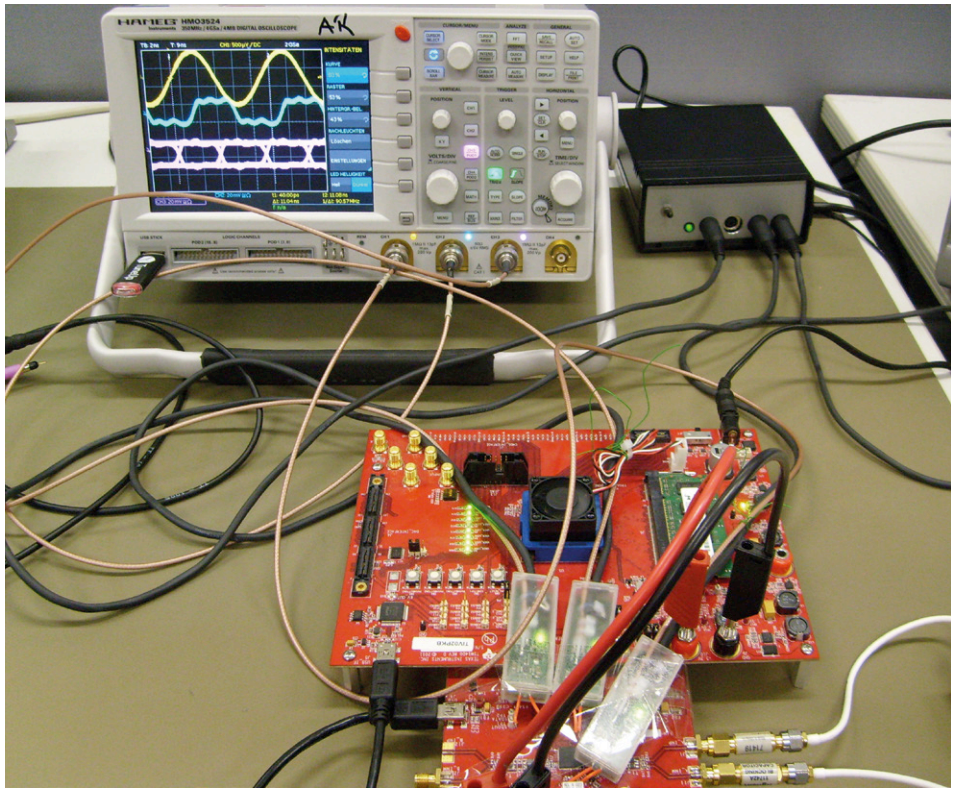


Figure 1. Ce petit circuit est raccordé à un bloc d'alimentation de 5 V et déclenche une alarme en cas de disparition de la tension de réseau.

sonde différentielle active 2 GHz

faites-la vous-même, encore mieux et moins cher



Alfred Rosenkränzer (Allemagne)

HDMI, DVI, USB, les signaux numériques rapides, on a l'habitude de les transmettre en mode différentiel. Pour les mesurer, il faut des sondes différentielles actives. Elles sont hors de prix pour l'amateur ou la petite entreprise. Alors, faites-les vous-même ! Le modèle présenté ici, dans un boîtier style clé USB, offre une très grande largeur de bande, près de 2 GHz.

Caractéristiques techniques

- atténuation : 10:1 pour des signaux différentiels et bouclage de 50 Ω sur l'oscilloscope
- impédance différentielle : 5 kΩ
- impédance d'entée asymétrique : 2,5 kΩ
- impédance de sortie : 50 Ω
- bande passante : 1,9 GHz (–3 dB)
- temps de montée/descente : 300 ps
- alimentation : ±8 à 12 V CC

On rencontre de plus en plus de signaux numériques rapides : FireWire, Panel Link, SATA, USB, DisplayPort, DVI, HDMI… Bien que différents protocoles et codages soient utilisés comme TDMS et 8b/10b, il y beaucoup de similitudes sur le plan électrique, aussi appelé couche physique ou *Physical Layer*. Les signaux sont transmis sur deux fils en différentiel, c'est-à-dire en opposition de phase. Leur amplitude n'est à peine que d'une centaine de millivolts, alors que le décalage peut atteindre plusieurs volts. Les deux conducteurs du signal sont généralement bouclés sur 100 Ω.

Le récepteur n'utilise que la différence de tension entre les deux fils, les parasites de mode commun induits sur les conducteurs sont ainsi atténués. La faiblesse des variations des signaux aide à réduire les interférences électromagnétiques. C'est ce qui permet d'atteindre de hauts débits binaires avec une vitesse de balayage modérée. Cette technique a déjà été utilisée il y a des décennies sur les puces ECL (*emitter-coupled logic*) ; de nos jours, on l'applique aux CML (*current-mode logic*) ou LVDS (*low voltage differential signaling*), mais aussi aux FPGA.

Bien entendu, on peut aussi observer ces signaux avec les sondes de mesure habituelles à haute impédance. Quand on veut voir le signal de différence, il faut deux sondes et la fonction mathématique de l'oscilloscope. Mais pour cela, les sondes doivent être convenablement équilibrées non seulement en atténuation, mais aussi en temps.

Pour les professionnels, il existe des sondes différentielles actives pour ce genre de mesure, par ex. Keysight (ex-Agilent), Tektronix, Rohde & Schwartz... Leur bande passante va de quelques centaines de mégahertz jusqu'à 10 GHz. Comme elles sont actives, elles ont besoin d'une alimentation, elle est fournie par l'oscilloscope (**fig. 1**) sur des contacts supplémentaires.

Évidemment, les accessoires d'un fabricant sont rarement compatibles avec les appareils des autres. Dans la gamme

inférieure de prix, on trouve des sondes à brancher sur un port USB, celui d'un oscilloscope par exemple. Quand il faut plusieurs sondes, comme pour mesurer le signal d'horloge et celui de données, on atteint vite la saturation du système USB. L'alimentation sur pile de 9 V n'est pas non plus une bonne solution, elle ne tient pas longtemps, surtout si on oublie d'éteindre la sonde. Alors, le prix de sondes adéquates, à partir de 800 euros, les rend inaccessibles aux amateurs comme aux petites entreprises. Il faut changer tout cela.

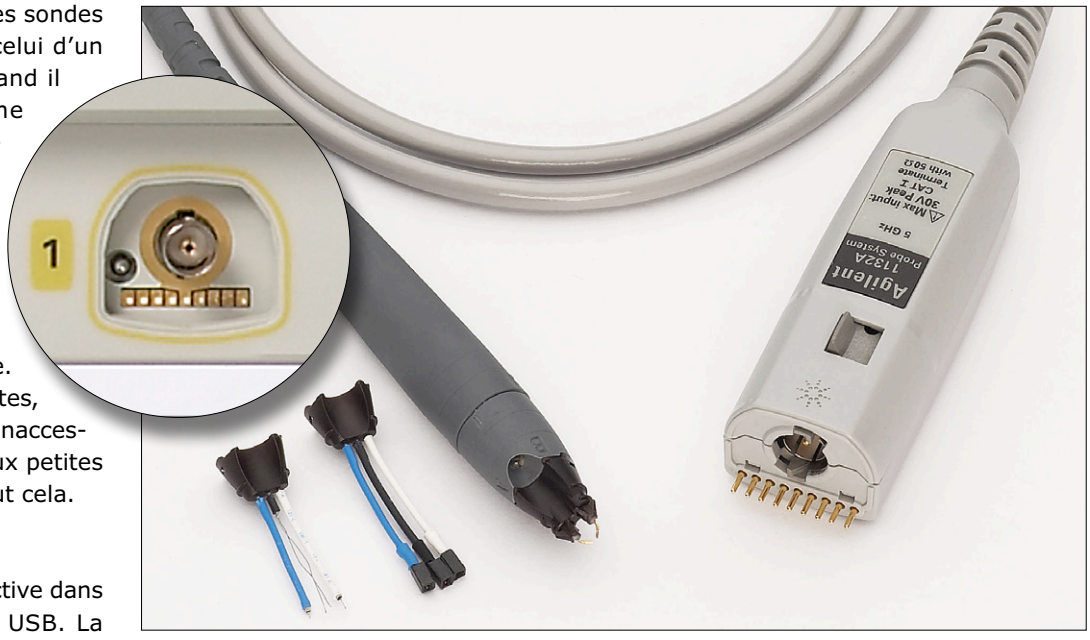


Figure 1. Entrée d'un oscilloscope avec les contacts prévus pour la sonde de mesure différentielle adaptée (source : Keysight).

Le circuit

Voici une sonde différentielle active dans un boîtier similaire à une clé USB. La **figure 2** révèle l'étonnante simplicité du circuit.

Un amplificateur opérationnel totalement différentiel ADA4927-1 d'Analog Devices suffit. Il a deux entrées et deux sorties et les deux branches sont câblées à l'identique. L'amplification vaut 1/5 selon le rapport R7/(R3+R4). La valeur de la résistance de R7 et de R8 est choisie de manière à obtenir la courbe de réponse la plus plate le plus loin possible. Elle se prolonge ici jusqu'à environ 1,9 GHz, assez pour la plupart des signaux et des oscilloscopes. Le signal d'entrée est appliqué aux contacts IN+ et IN–. Seule la sortie négative (–OUT) est utilisée pour attaquer l'entrée de l'oscilloscope avec une résistance de source de 50 Ω, à boucler sur 50 Ω, de préférence avec la fonction interne de l'appareil, sinon directement à l'entrée. Par rapport à l'amplitude de l'un des signaux d'entrée (pas leur somme), la sonde présente donc un rapport de

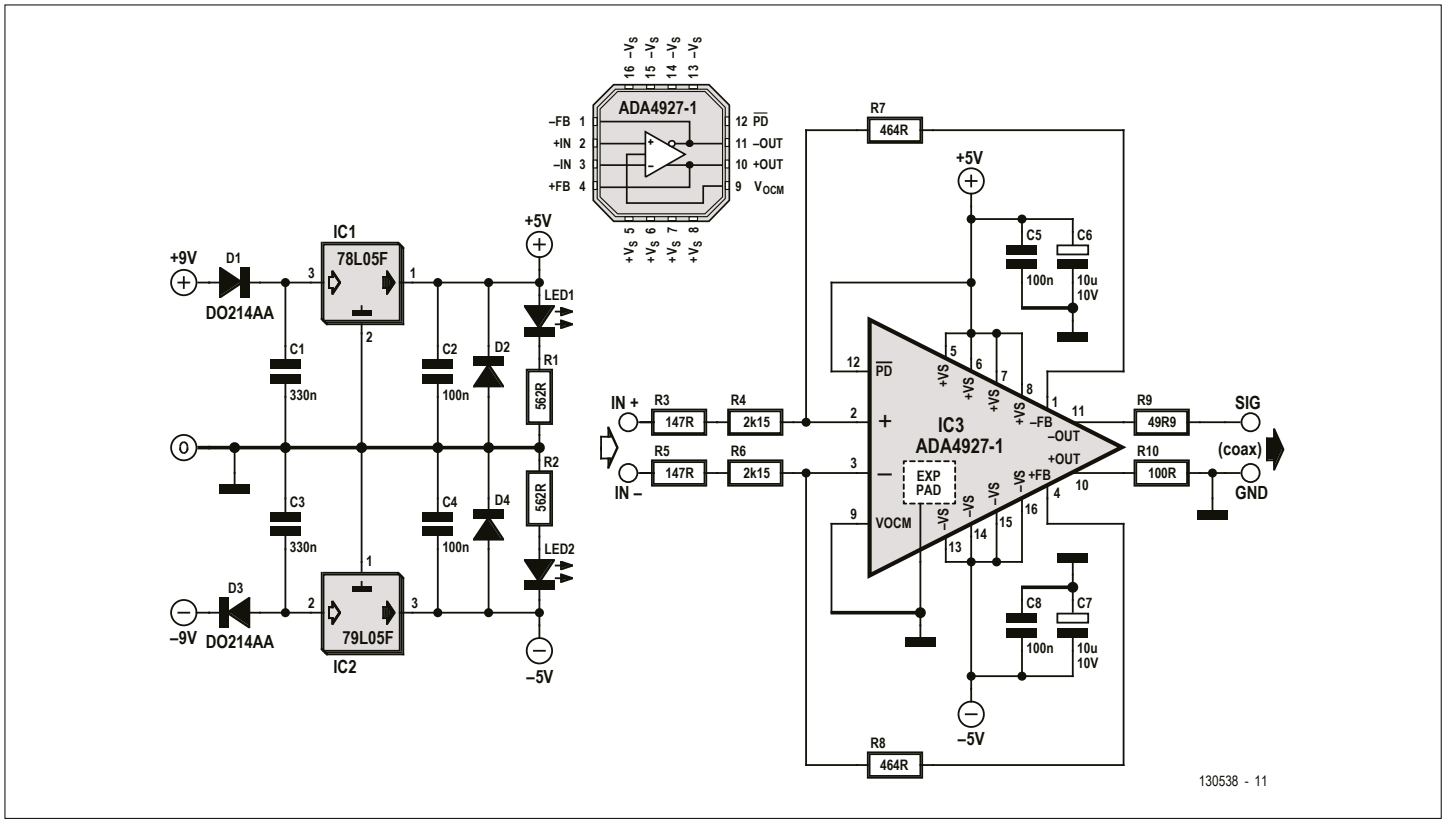


Figure 2. Schéma de la sonde différentielle active.

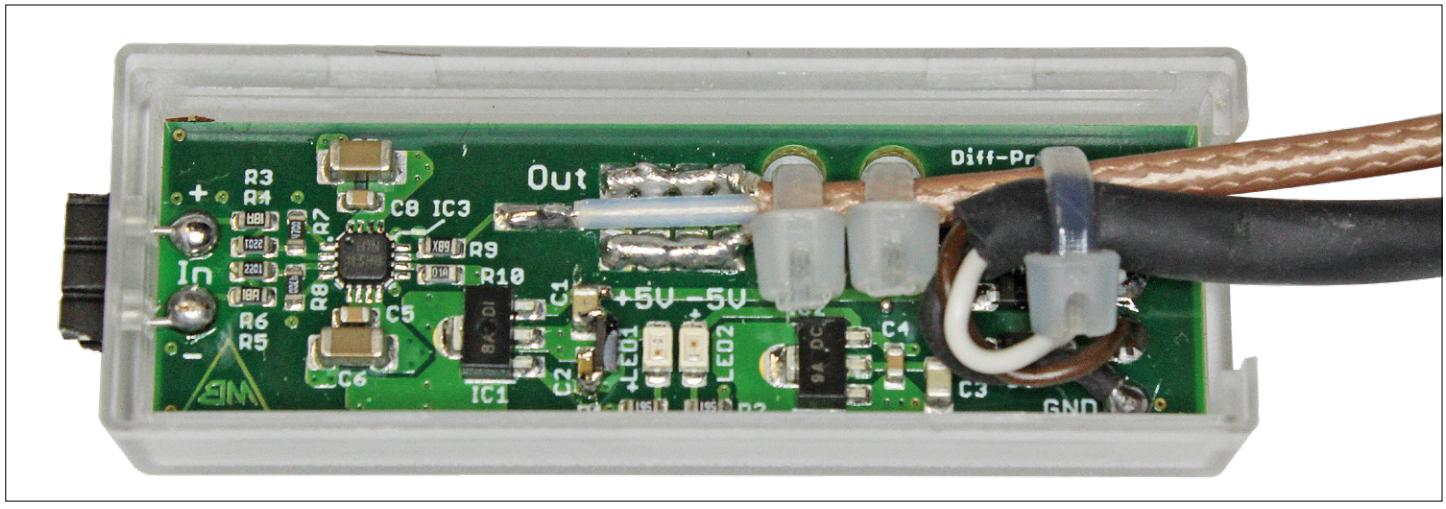


Figure 3. Sonde de mesure installée dans un boîtier transparent format clé USB.

tension de 1/10, tout comme une sonde passive à haute impédance raccordée sur 50 Ω.

L'impédance d'entrée différentielle avoisine 4,6 kΩ. Cela semble très peu en comparaison d'une sonde passive. Mais comme les signaux rapides sont à très basse impédance (100 Ω), c'est bien suffisant.

La sortie inutilisée de la puce est reliée à la masse par 100 Ω pour garantir une symétrie parfaite de charge.

La puce dispose d'une entrée V_OCM de compensation du décalage qui n'est pas utilisée ici, elle est mise à la masse et le décalage sera annulé sur l'oscilloscope.

L'entrée de coupure du courant (*Power down*) est reliée à la tension d'alimentation, la puce reste donc active tout le temps.

Sous la puce, il y a une surface de contact dite *Exposed Paddle* (EPAD) à connecter à un plan soumis à une tension comprise entre celles d'alimentation, ici à la masse. Les deux tensions de ±5 V sont produites par les régulateurs linéaires IC1 et IC2, et découplées à la masse (GND)

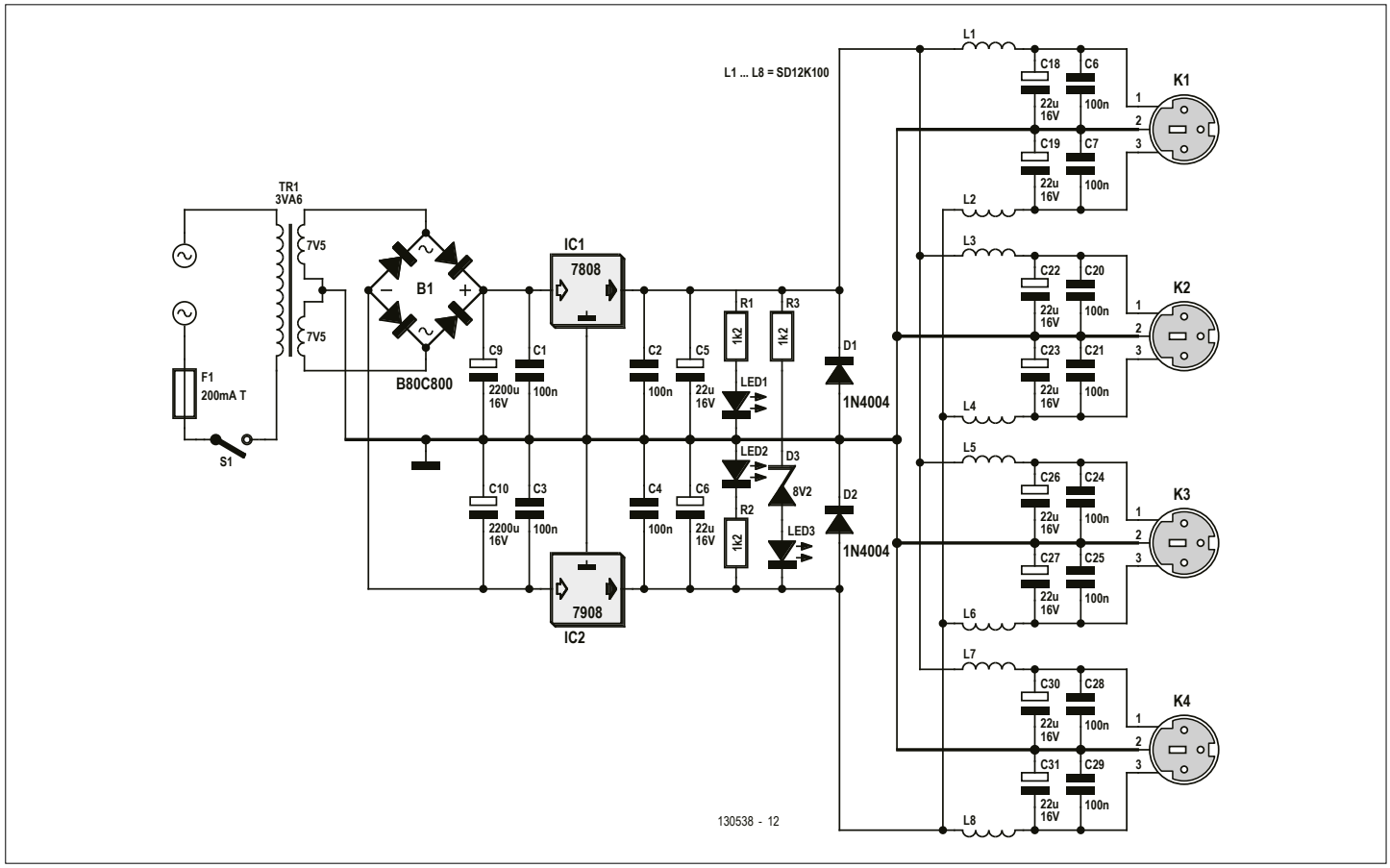


Figure 4. Schéma de l'alimentation secteur pour quatre sondes de mesure.

Une sonde différentielle professionnelle est trop chère pour un amateur ou une petite entreprise

par des condensateurs. Les diodes D1 et D3 protègent d'une inversion de polarité, tandis que D5 et D6 empêchent un verrouillage haut du régulateur de tension au cas où l'une des alimentations serait appliquée avant l'autre. Les LED D2 et D4 témoignent de la présence de ces tensions.

Autant le circuit est simple, autant son câblage est crucial pour atteindre le fonctionnement voulu.

J'ai eu recours à un circuit imprimé à quatre couches. Celle du haut assure les liaisons du signal, la suivante est totalement occupée par un plan de masse comme point de référence. Les deux autres servent à acheminer l'alimentation. J'ai évoqué l'EPAD sous la puce, qu'il faut relier à la masse. Comme ce n'est pas une mince affaire pour un amateur d'effectuer pareille soudure, je propose une carte toute montée et testée, prête à l'emploi.

La puce est dotée de deux paires de sorties. Les broches +FB et –FB, à gauche, qui encadrent les entrées sont utilisées pour fermer la boucle d'amplification avec R7 et R8. Les deux autres, à droite, servent de sorties. Les raccordements des deux tensions d'alimentation au-dessus et en dessous sont découplés par deux condensateurs aussi près que possible de la masse. Tant pour les alimentations que pour la masse, de nombreux *vias* sont prévus pour réduire l'inductance autant que faire se peut.

La mécanique

On peut insérer la carte de la sonde dans un boîtier transparent de clé USB (**fig. 3**). Le câble coaxial muni d'une fiche BNC se soude par le conducteur central à la pastille « Out », le blindage sur la surface à sa droite. Des trous permettent de fixer ce câble par un collier de serrage. Le câble d'alimentation sera soudé aux pastilles +9 V, –9 V et GND. Après avoir installé la carte dans son boîtier, on introduit dans les trous forés en face avant une barrette femelle sécable à trois éléments dont on aura enlevé la broche centrale. On incurve les autres broches pour les souder aux pastilles In+ et In-. C'est là qu'il faudra insérer les broches correspondantes sur lesquelles auront été soudés de petits fils fins à relier aux conducteurs du signal. Ceci permet d'abord d'installer cette prise et ensuite d'y insérer la sonde. Il est alors possible de faire l'échange entre plusieurs signaux. Si cette prise est abîmée, on s'en fait une nouvelle. Manier une sonde à deux entrées n'est pas très pratique.

Si vous utilisez plusieurs sondes, veillez à ce que les câbles coaxiaux aient tous la même longueur, sinon il y aura des différences dans le temps de parcours entre elles et vous risquez des erreurs de mesure. Le reste, on peut le compenser sur la plupart des oscilloscopes par le « *Skew Ajust* ».

Avec des mesures différentielles, la somme des décalages des deux signaux est atténuée, seule la différence est affichée. Mais si vous voulez mesurer le décalage, vous pouvez utiliser une sonde à haute impédance ou relier l'une des entrées à la masse. Il faut alors penser à la baisse relative de la résistance d'entrée.

L'alimentation

On peut alimenter la sonde de mesure par une alimentation de laboratoire d'au moins ±8 V. C'est très simple avec une seule sonde, mais plus il y en a, plus le câblage se complique et on finit par s'y perdre. Aussi ai-je étudié une alimentation secteur de table (**figure 4**) pour quatre sondes, en concordance avec le nombre maximal d'entrées d'un oscilloscope. Ce sont des fiches mini-DIN à trois broches que j'ai choisies pour le branchement : aucun risque d'inversion de polarité. Les quatre sorties sont largement filtrées en interne pour minimiser la diaphonie entre sondes. Le circuit se compose d'un transformateur secteur à deux secondaires, un redresseur, des électrolytiques et deux régulateurs linéaires encadrés des habituels condensateurs de découplage. Deux LED indiquent le fonctionnement de chacune des alimentations et, à l'aide d'une diode zener en série, une troisième en façade témoigne que les deux tensions sont bien là. Deux autres diodes empêchent qu'une tension inverse, issue éventuellement de la sortie, ne cause des dommages.

Les deux tensions sont encore filtrées par bobines et condensateurs avant d'atteindre les connecteurs de sortie.

Conclusion

Cette sonde différentielle active bon marché, parce que construite pour une fraction du prix des produits commerciaux comparables, présente d'excellentes caractéristiques (cf. **figures 5** et **6**). Pour cela, il faut de bons composants bien choisis, mais aussi une conception mûrement réfléchie du circuit imprimé.

Les lecteurs intéressés peuvent me contacter pour se procurer une carte montée et testée. Il est aussi possible de commander un kit comprenant le boîtier, le câble HF avec fiche BNC et le câble d'alimentation avec fiche. Pour des informations supplémentaires, écrivez à alfred_rosenkraenzer@gmx.de.

(130538 – version française : Robert Grignard)

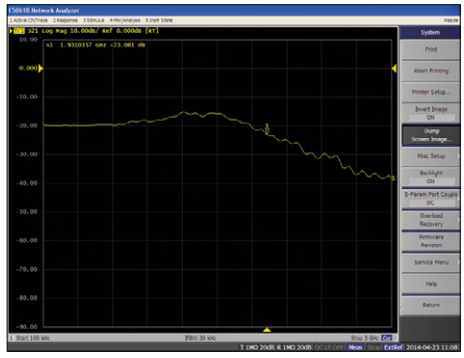


Figure 5. Comportement en fréquence jusqu'à 3 GHz de la sonde de mesure. Le point à –3 dB se situe à environ 1,9 GHz.

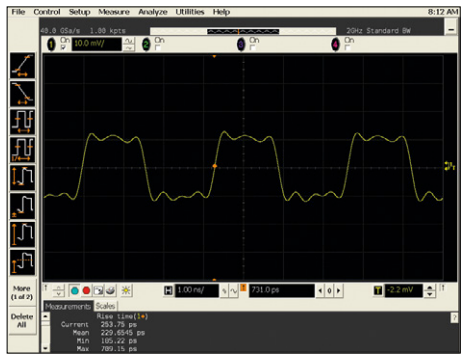
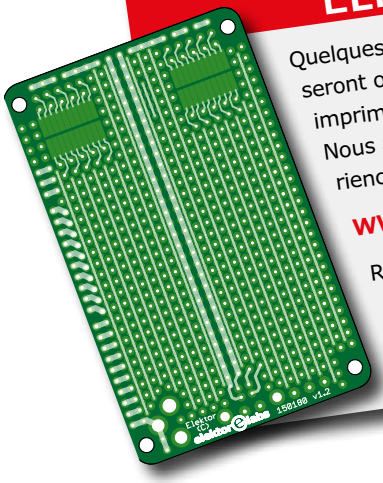


Figure 6. Mesure d'un signal d'horloge à 300 MHz avec un oscilloscope de 2 GHz.

carte ELPB-NG : le prototypage revisité la *techtonique* des plaques

Recevez une ELPB-NG gratuite !



Quelques centaines de cartes ELPB-NG seront offertes aux acheteurs de circuits imprimés et de kits Elektor à souder. Nous attendons votre retour d'expérience avec la ELPB-NG sur la page **www.elektor.com/elpb-ng.**

Rejoignez l'équipe de test, vos commentaires et idées nous aideront à améliorer les cartes d'expérimentation.

Alimentation

Disposer de bonnes connexions d'alimentation est essentiel lorsqu'on crée des prototypes. Voilà pourquoi la carte ELPB-NG est équipée de deux rails d'alimentation et différentes empreintes (prise USB-B, embase femelle pour jack d'adaptateur CC, borniers à vis pour circuit imprimé). Finies les pinces crocodiles qui se détachent ou créent des courts-circuits, il suffit de monter le connecteur adapté.

Dimensions

Avec ses 87,6 x 54,6 mm, la carte n'est ni trop grande, ni trop petite. Ce sont en gros les dimensions d'un paquet de cigarettes, parfaites pour un boîtier DIP de 40 broches. Trop petit ? Pas de problème, vous pouvez empiler autant de cartes ELPB-NG que nécessaire.

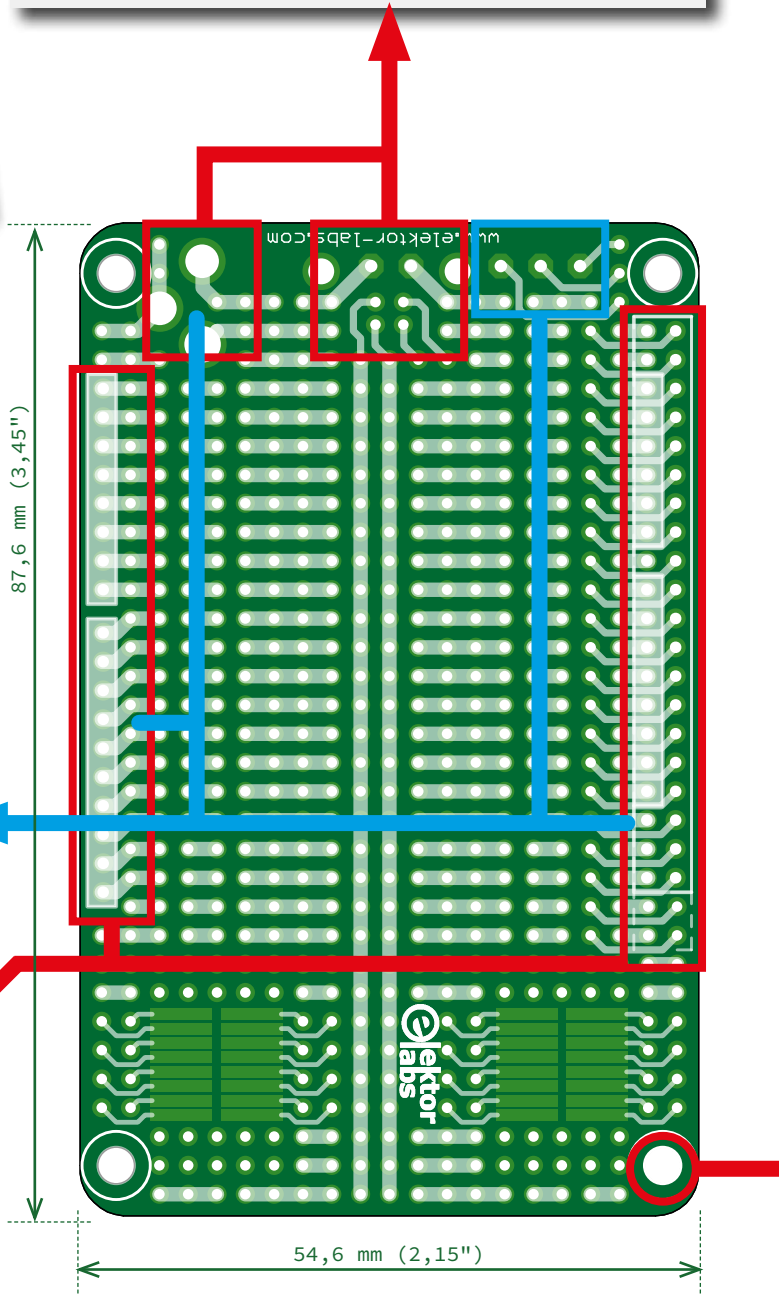
(Les cartes reproduites ici sont à l'échelle 150%)

Connexions

Arduino, Raspberry Pi et programmateurs de microcontrôleur peuvent être reliés facilement à la carte ELPB-NG grâce aux empreintes spéciales « connecteur » et à leurs broches.

Compatibilité Arduino & Raspberry Pi

C'est indéniable, de nos jours la conception de circuits s'articule souvent autour d'une carte Arduino ou Raspberry Pi. Il est donc logique que la carte ELPB-NG soit compatible avec Arduino et Raspberry Pi (version 2, modèle B).

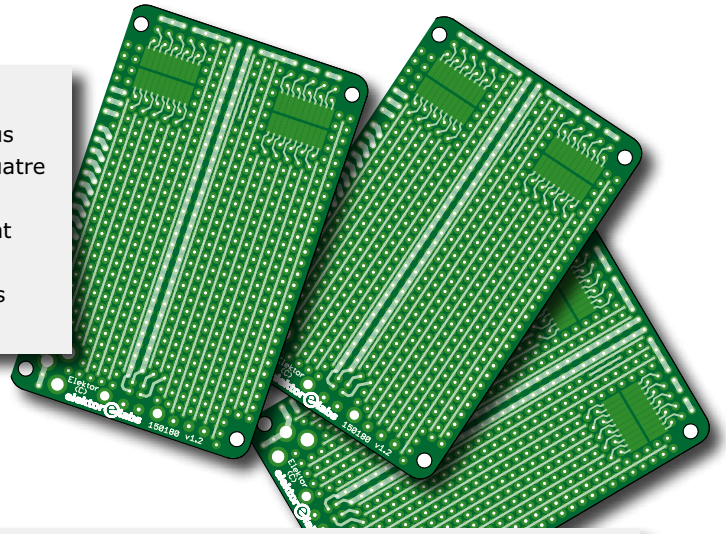


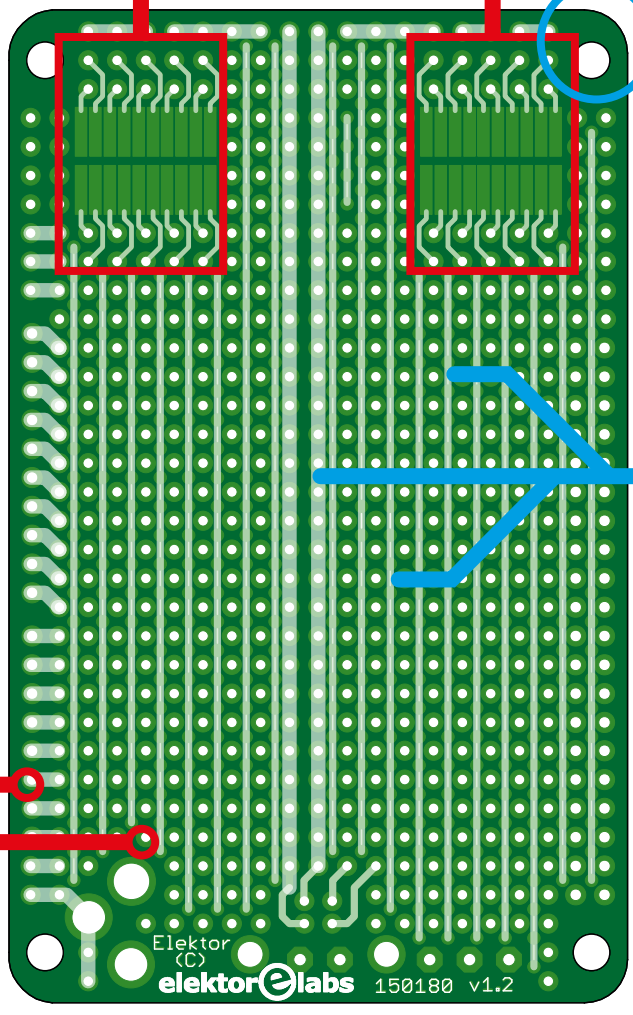
Clemens Valens, Elektor.Labs

Les cartes de prototypage (ou plaques à trous, ou plaques d'essai) sont omniprésentes et pourtant dépassées : les microcontrôleurs ont remplacé les composants TTL et les amplis op n'ont plus besoin de lignes d'alimentation symétriques. Bref l'électronique ne cesse d'évoluer alors que les plaques d'essai sont pour ainsi dire restées dans leurs trous. Elektor.labs est donc fier de vous présenter ELPB-NG™ (*Elektor. Labs Prototyping Board Next Generation*), une carte de prototypage de nouvelle génération.

CMS

Il existe une grande variété d'empreintes CMS, un problème en soi. Nous avons combiné les empreintes SOIC et SOT-23 en une seule et défini quatre zones de prototypage CMS multiboîtiers (deux par face) pour permettre la construction de circuits très compacts. Regroupées, ces zones peuvent accueillir jusqu'à 20 boîtiers SOT-23 à 3 pattes (ou 10 à 6 pattes), ou 4 SOIC-14/16, et même 2 SOIC-20. On peut aussi y monter des boîtiers à 2 bornes comme les 0603, 0825 ou 1206.





Angles arrondis

Les angles sont arrondis pour des questions de sécurité et d'ergonomie. Les trous de 3,2 mm de diamètre facilitent la fixation de la carte sur un support.

> Le prototypage entre enfin dans le 21e siècle grâce à la technologie Smart-Grid™ de la carte ELPB-NG.

Technologie Smart-Grid™ d'Elektor.Labs

Oubliés les petits morceaux de fil (émaillé) arrachés ou les pistes tracées à la soudure : grâce à la technologie Smart-Grid™d'elektor.labs (brevet déposé), la plupart des fils nécessaires sont disponibles sous la carte ! Les tracés étroits peuvent aisément être coupés à la bonne longueur ; ils forment alors avec les colonnes de l'autre face une matrice de câblage.

Trous

Une plaque d'essai sans trous ne serait pas très pratique. Chaque carte ELPB-NG a donc été minutieusement perforée 620 fois. Les 548 trous de 0,9 mm et les 58 trous de 1 mm permettent de placer des composants selon un grand nombre de configurations. Les autres trous servent aux connecteurs.

(150180-I – version française : Hervé Moreau)

sonnette musicale sans fil programmable

personnalisez votre carillon !

Victor Hugo Pachon (HUGO Tecnologia) (Colombie)

Les ariettes ou carillons de bienvenue joués par les sonnettes sans fil du commerce sont rapidement lassants. Nous sommes électroniciens, alors tant qu'à héberger un pinson en cage électronique, autant qu'il soit programmable et sache accueillir nos hôtes avec une mélodie originale.

Le son des sonnettes sans fil bon marché est si banal qu'on peut se demander si notre tolérance aux camelotes à deux yuans n'est pas devenue de l'accoutumance. Ding-dong à deux tons, aboiements et autres meuglements sonnent faux, ils sont bébêtes et tellement répandus que dans certains quartiers on confond sa sonnette avec celle du voisin. Quant à personnaliser ces bidules : mission impossible.
La sonnette proposée ici est ouverte dans le sens où le micrologiciel (PIC), le canal radio (merci le HT12 de Holtek) ainsi que la musique (commutateur et tables de notes) sont programmables.

Caractéristiques

- Mélodie créée par l'utilisateur, stockée en mémoire, max. 127 notes
- Notes/octaves/durées de note codées sur 8 bits
- Commande sans fil faible puissance à 433 MHz
- Modules radio disponibles dans le commerce
- Microcontrôleur PIC16F873A
- Codage des données par HT12 de Holtek
- Composants traversants uniquement
- Ni USB, ni MP3

Émetteur

Les sonnettes traditionnelles comportent un transformateur, un vibreur ou une bobine, ainsi qu'un long fil relié à un poussoir lumineux. Notre système est sans fil, possède un émetteur à pile à monter sur la porte, et un récepteur associé à placer à l'intérieur, habituellement dans l'entrée.
Le schéma de la **figure 1** est celui de l'émetteur. Lorsque l'utilisateur appuie sur le bouton de sonnette relié à K5, l'amplificateur à grand gain T5-T6 met la LED LED2 sous tension tandis qu'IC7, un codeur de données série HT12E de Holtek, est activé par mise au niveau bas de sa broche TE (14). Le module émetteur IC8 est dans le même temps alimenté par le transistor PNP T7. IC8 est un QAM-TX1-433, un émetteur à 433 MHz de faible puissance utilisable sans octroi de licence et dont la portée devrait être de 50 m en intérieur. QAM signifie *Quadrature Amplitude Modulation* (modulation d'amplitude en quadrature).
Le code personnel utilisé sur le canal radio est défini dans le bloc de configuration K8. Il détermine quelles broches de codage A0 à A7 sont au niveau haut (par défaut) ou bas (cavalier placé). Le mot codé défini sur l'émetteur, secret pour des raisons évidentes, doit correspondre à celui du récepteur cible. Les puces codeuses/décodeuses HT12E et HT12D sont une référence en matière de transmission RF ou IR, sont bon marché, faciles à trouver et à implanter [1] [2]. La désignation de ces puces est claire : HT pour Holtek, 12 pour 2^{12} possibilités de codage, E pour *Encoder*, et D pour *Decoder*.
L'émetteur est alimenté par une pile bouton CR2032 de 3 V (BAT1). L'antenne de l'émetteur est un fil rigide isolé de 17 cm (un quart de la longueur d'onde) relié à ANT2.
Il est recommandé de laisser courir les antennes de l'émetteur et du récepteur hors de leurs boîtiers et de les orienter verticalement.
Si vous habitez un pays où l'émission sur 433,92 MHz est interdite, remplacez les puces utilisées ici par des modules compatibles (868, 315 ou 915 MHz).

Récepteur

Le récepteur (**fig. 2**) est alimenté par un adaptateur de sortie 9 V CC et 100 mA. Un 7805 abaisse et régule la tension de 9 V à +5 V pour la circuiterie logique et le module récepteur. La tension non régulée de 9 V alimente le petit amplificateur audio construit autour de T2 et T3.
Le signal radio de l'émetteur recueilli sur ANT1 est combiné et démodulé par IC2 (module récepteur QAM-RX2) afin de reconstituer le flux de données original. La puce HT12D (IC3) scrute en permanence sa broche *DATA IN* pour retrouver

Figure 1. Schéma de l'émetteur.

parmi les paquets de données entrants le mot codé qui a été défini en K8. Si celui-ci correspond au mot contenu dans le signal reçu, la sortie *VT* est mise au niveau haut. Le PIC (IC5) est alors averti, via l'inverseur IC4.A, la porte NON-ET IC4.C et l'entrée RB7, que quelqu'un a appuyé sur le « seul et unique » bouton de sonnette. Le PIC exécute une routine chargée de produire par MLI (modulation de largeur d'impulsion) sur le port RA2 un flux de données correspondant à une musique. Le son joué suit la séquence de notes entrées ; il ne faudra donc vous en prendre qu'à vous-même si le résultat évoque le souvenir de votre premier cours de flûte. La mélodie se programme note par note à l'aide de S3, de la LED LED1 et d'un banc de huit commutateurs relié à K2 (S1, **fig. 3**). Ce compositeur relié à K2 est activé de façon logicielle via la ligne du port RA4. La mélodie est sauvegardée dans le PIC.

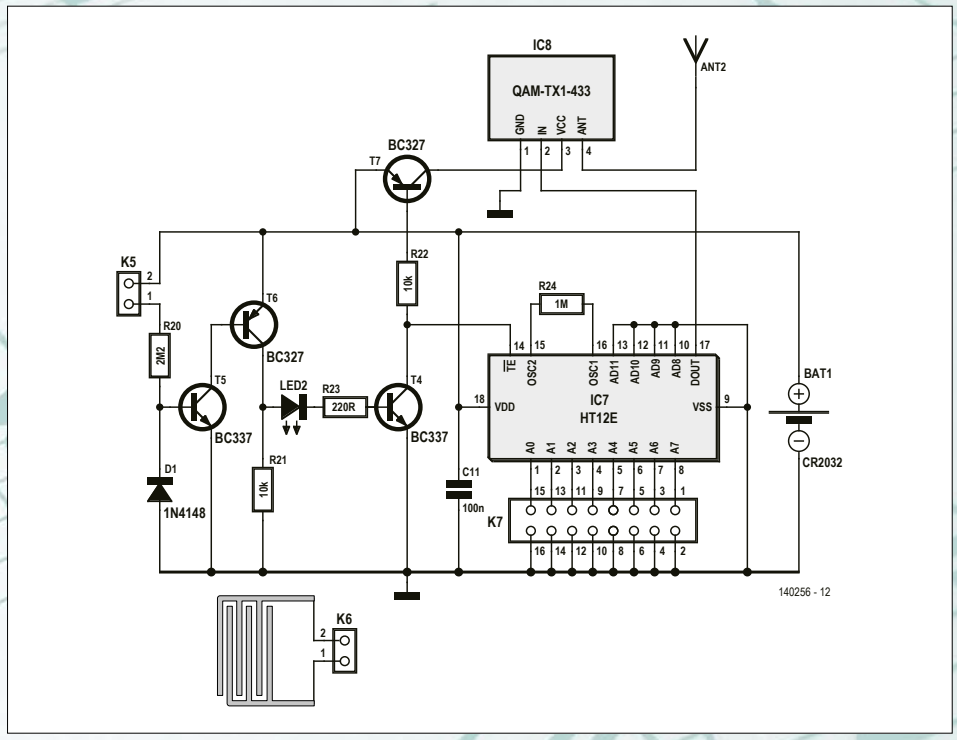
Le PIC16F873A est cadencé à 4 MHz par le quartz X1 et les condensateurs C9/C10, et automatiquement initialisé au démarrage par le réseau C6/R4.

Le poussoir S3 sert à programmer la mélodie et à la jouer « localement », c'est-à-dire sans avoir à appuyer sur le bouton de sonnette. Il est préférable qu'il ne soit pas facilement accessible.

L'interrupteur à glissière S2 permet de régler deux niveaux sonores pour le haut-parleur LS relié à K4.

Construction

Les cartes du récepteur, de l'émetteur, du compositeur et du bouton à effleurement proviennent de la même plaque de circuit imprimé (**fig. 4**). La carte d'aspect brunâtre que vous voyez sur la photo est un prototype fabriqué par le labo d'Elektor avec sa fraiseuse Colinbus. Les cartes vendues sur l'e-choppe sont vertes et recouvertes d'une sérigraphie pour l'implantation des composants. Conseil : utilisez des supports pour tous les CI en boîtier DIL.



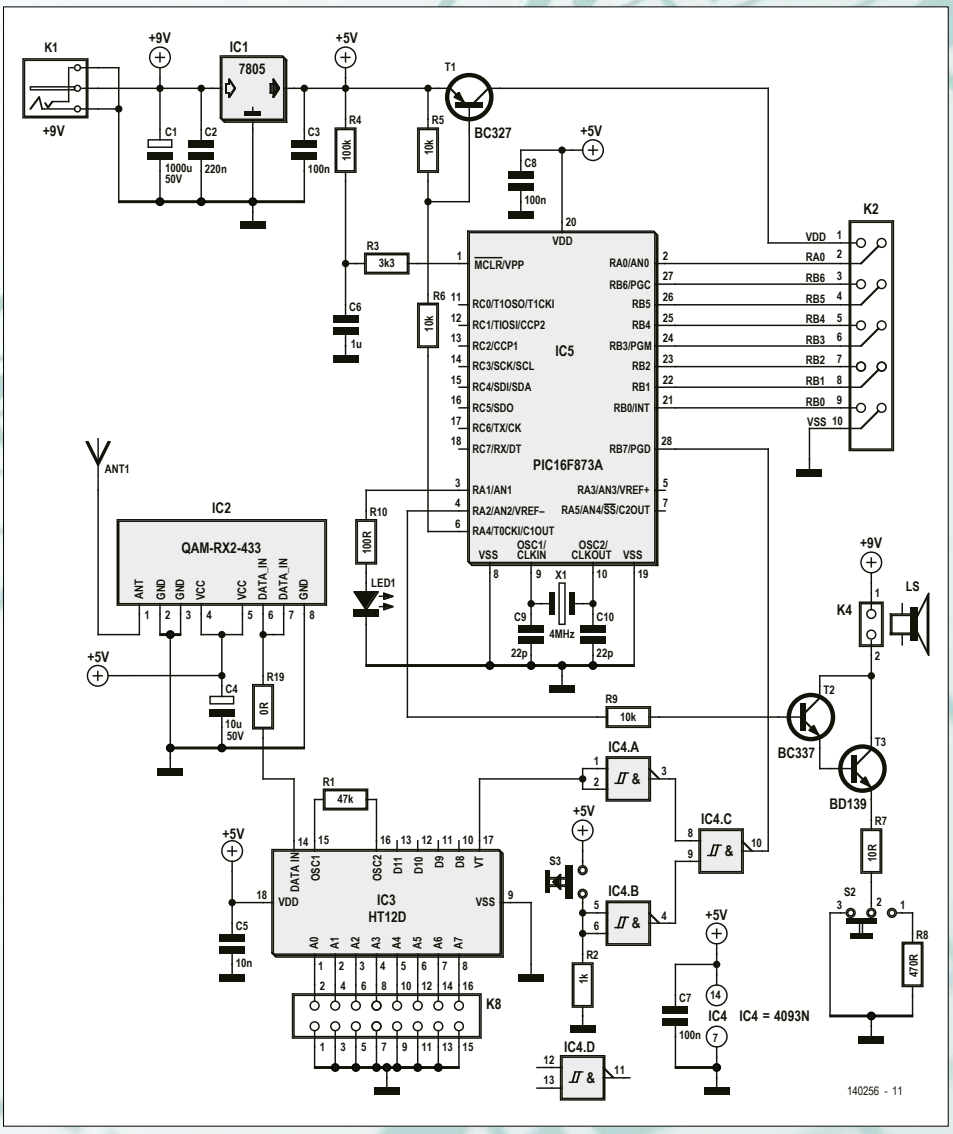


Figure 2. Schéma du récepteur.

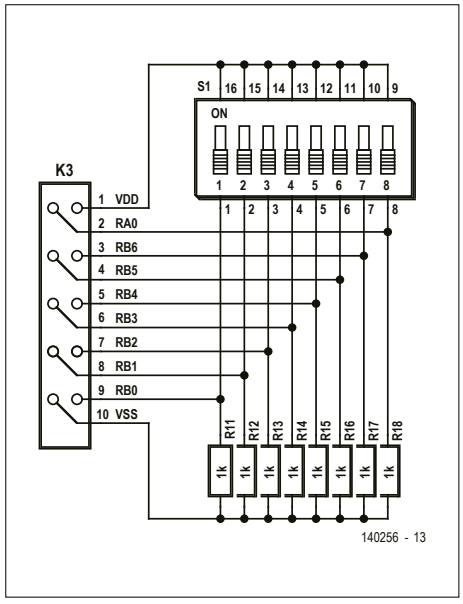


Figure 3. Ce compositeur de notes n'est peut-être pas un triomphe de la technologie moderne, mais il fait son travail correctement.

Les trois sous-cartes sont spacieuses et, à l'exception des modules radio émetteur et récepteur, ne recourent qu'à des composants traversants. Vous n'aurez donc besoin d'aucun outillage spécial pour assembler ce projet, juste de quelque habileté avec un fer à souder.
La carte émettrice peut être logée dans un boîtier Hammond XXL 1591. La petite carte du bouton à effleurement se monte à l'extérieur du boîtier en passant la LED LED2 par un trou percé juste au-dessus du bouton. Vous pourriez bien sûr préférer un bouton-poussoir classique.

Tableau 1. Programmation des notes

			bits sur S1							
			1	**2**	**3**	**4**	**5**	**6**	**7**	**8**
note	silence		0	0	0	0	×	×	×	×
	do	*C*	1	0	0	0	×	×	×	×
	do#	*C#*	0	1	0	0	×	×	×	×
	ré	*D*	1	1	0	0	×	×	×	×
	ré#	*D#*	0	0	1	0	×	×	×	×
	mi	*E*	1	0	1	0	×	×	×	×
	fa	*F*	0	1	1	0	×	×	×	×
	fa#	*F#*	1	1	1	0	×	×	×	×
	sol	*G*	0	0	0	1	×	×	×	×
	sol#	*G#*	1	0	0	1	×	×	×	×
	la	*A*	0	1	0	1	×	×	×	×
	la#	*A#*	1	1	0	1	×	×	×	×
	si	*B*	0	0	1	1	×	×	×	×
octave	inférieure		×	×	×	×	0	×	×	×
	supérieure		×	×	×	×	1	×	×	×
durée de la note	1/64		×	×	×	×	×	0	0	0
	1/32		×	×	×	×	×	1	0	0
	1/16		×	×	×	×	×	0	1	0
	1/8		×	×	×	×	×	1	1	0
	1/4		×	×	×	×	×	0	0	1
	1/2		×	×	×	×	×	1	0	1
	1		×	×	×	×	×	0	1	1

Liens

[1] Codeur Holtek HT12E : www.holtek.com.tw/english/docum/consumer/2_12e.htm
[2] Décodeur Holtek HT12D : www.holtek.com.tw/english/docum/consumer/2_12d.htm
[3] Code du projet : www.elektor-magazine.com/140256
[4] Elektor Labs : www.elektor-labs.com/project/musical-programmable-bell.13959.html

La carte réceptrice tient dans un boîtier Hammond plus grand, un 1591 XXG ABS, qui sert aussi de logement, réduit mais efficace, pour le haut-parleur.
La carte du compositeur est construite séparément et reliée au récepteur par un câble plat à 10 conducteurs, terminé par des connecteurs autodénudants (IDC).

Programmation et utilisation

Voyons comment programmer les mélodies, ritournelles et autres chefs-d'œuvre de « votre » sonnette. Les plus paresseux (ou moins inspirés) d'entre vous pourront envoyer leurs rejetons sur le net afin qu'ils récupèrent la partition d'un air quelconque dont vous n'aurez alors plus qu'à envoyer la transcription en octets dans le PIC. Niek et Jan, du labo d'Elektor, ont enregistré l'air de *Jingle Bells* dans le PIC préprogrammé de l'e-choppe (réf. : 140256-41).

Le **tableau 1** donne la relation entre les mots de 8 bits à écrire dans la mémoire du PIC et les notes, octaves et durées de note. Ces correspondances forment le cœur de la sonnette.
Une fois que vous avez composé la suite de mots de 8 bits de la mélodie de votre sonnette, voici la séquence à suivre pour sa programmation :

1. La LED1 s'allume après la mise sous tension. Appuyez sur S3 pendant que cette LED est allumée pour entrer dans le mode de programmation.
2. Configurez les commutateurs de S1 pour former le mot de 8 bits de la première note.
3. Appuyez sur S3 pour programmer le mot de 8 bits dans le PIC.
4. Configurez les commutateurs de S1 pour constituer les 8 bits de la note suivante.
5. Appuyez sur S3.
6. Retournez à l'étape 4.

Pour terminer la programmation de votre mélodie (max. 127 notes), configurez le pseudo-mot FF (tous les commutateurs sur ON = 1111 1111) et appuyez sur S3. D'après le rédacteur de notre rubrique Rétronique : « C'est comme programmer un Altair, un ELF ou le Junior Computer de 1979 ! » Le système est prêt à jouer votre mélodie. Pour l'entendre, appuyez sur le bouton de la carte émettrice ou sur le poussoir S3 de la carte réceptrice. Les membres abonnés peuvent téléchar-

ger le code source du projet depuis [3]. C'est de l'assembleur !
Les utilisateurs avancés ne manqueront pas de faire la moue devant le côté primitif des commutateurs avant de composer une mélodie avec un Cortex ARM, de la diffuser en *streaming*, de revendiquer des droits d'auteur, d'*hashtaguer* à tout va leur projet sur Twitter pour finir par programmer leur mélodie dans l'humble PIC. N'hésitez pas à partager vos meilleures compositions de 127 notes sur l'Elektor. Labs [4], page où je répondrai également à vos questions.

(140256 – version française : Hervé Moreau)

Liste des composants

Carte récepteur

Résistances

film carbone, 5 %, 0,25 W
R1 = 47 kΩ
R2 = 1 kΩ
R3 = 3,3 kΩ
R4 = 100 kΩ
R5, R6, R9 = 10 kΩ
R7 = 10 Ω
R8 = 470 Ω
R10 = 100 Ω
R19 = 0 Ω

Condensateurs

C9, C10 = 22 pF, 50 V, C0G/NP0, pas de 2,54 mm
C6 = 1 µF, 50 V, X7R, pas de 5,08 mm
C3, C7, C8 = 100 nF, 50 V, X7R, pas de 5,08 mm
C5 = 10 nF, 50V, X7R, pas de 2,54 mm
C2 = 220 nF, 50 V, X7R, pas de 5,08 mm
C4 = 10 µF, 50 V, pas de 2 mm , 5x11 mm
C1 = 1000 µF, 50 V, pas de 7,5 mm, 16x26 mm

Semi-conducteurs

LED1 = LED, rouge, 3 mm
X1 = quartz de 4 MHz, C_L=18 pF
T1 = BC327
T2 = BC337
T3 = BD139
IC1 = MC7805
IC2 = QAM-RX2-433
IC3 = HT12D
IC4 = 4093
IC5 = PIC16F873A, programmé, e-choppe 140256-41

Divers

K1 = embase jack femelle pour adaptateur CC, broche 1,95 mm, 12 V, 3 A (broche centrale = GND !)
K4 = bornier à vis à 2 voies pour CI, 5,08 mm
K2 = embase à 10 picots (2x5) avec col
K8 = barrette à 16 broches, 2 rangées, droites
S2 = interrupteur à glissière, SPDT, à monter au bord du circuit imprimé
S3 = bouton-poussoir à effleurement, 6x6 mm
boîtier Hammond 1591 ABS, 122,45 x 95,9 x 35,25 mm
ANT1 = fil, 17 cm
haut-parleur miniature, Ø 36 mm , 1 W
support CI DIP-14
support CI DIP-18
support CI DIP-28 étroit

Carte émetteur

Résistances

film carbone, 5 %, 0,25 W
R20 = 2,2 MΩ
R21, R22 = 10 kΩ
R23 = 220 Ω
R24 = 1 MΩ

Condensateurs

C11 = 100 nF, 50 V, X7R, pas de 5,08 mm

Semi-conducteurs

LED2 = LED, rouge, 3 mm
T4, T5 = BC337
T6, T7 = BC327
IC7 = HT12E
IC8 = QAM-TX1-433
D1 = 1N4148

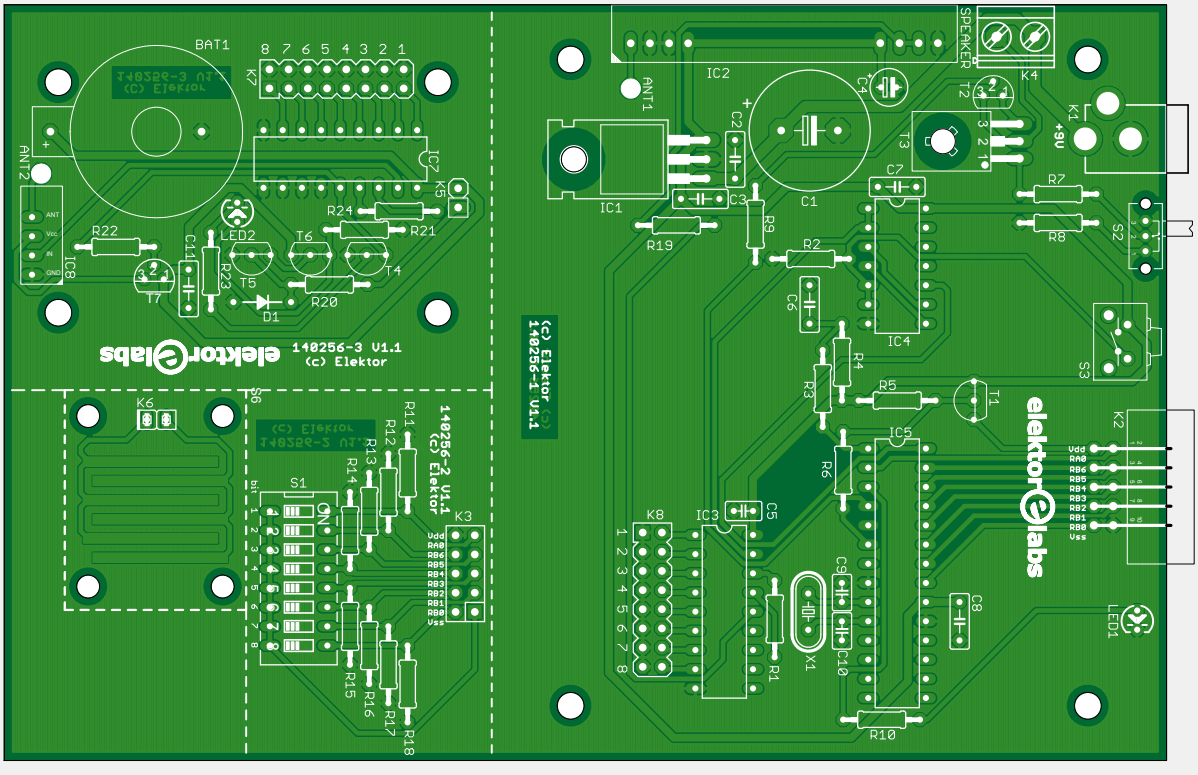
Divers

K5 = barrette à 2 broches, pas de 2,54 mm
K6 = embase à 2 contacts, pas de 2,54 mm
K8 = barrette à 16 broches, double rangée, droite
G1 = porte-pile pour CI pour la CR2032
boîtier Hammond type 1591XXLBK
BAT1 = pile bouton CR2032
S6 = carte du bouton à effleurement ou bouton « classique »
ANT2 = fil, 17 cm
support CI DIP-18

Carte compositeur

S1 = commutateur DIP à 8 voies
K3 = socle de connecteur IDC (contacts autodénudants) à 10 voies (2x5)
R11-R18 = 1 kΩ

Figure 4. Le circuit imprimé composite du projet peut être découpé à la scie à métaux pour séparer les quatre cartes : récepteur, émetteur, compositeur et bouton à effleurement.

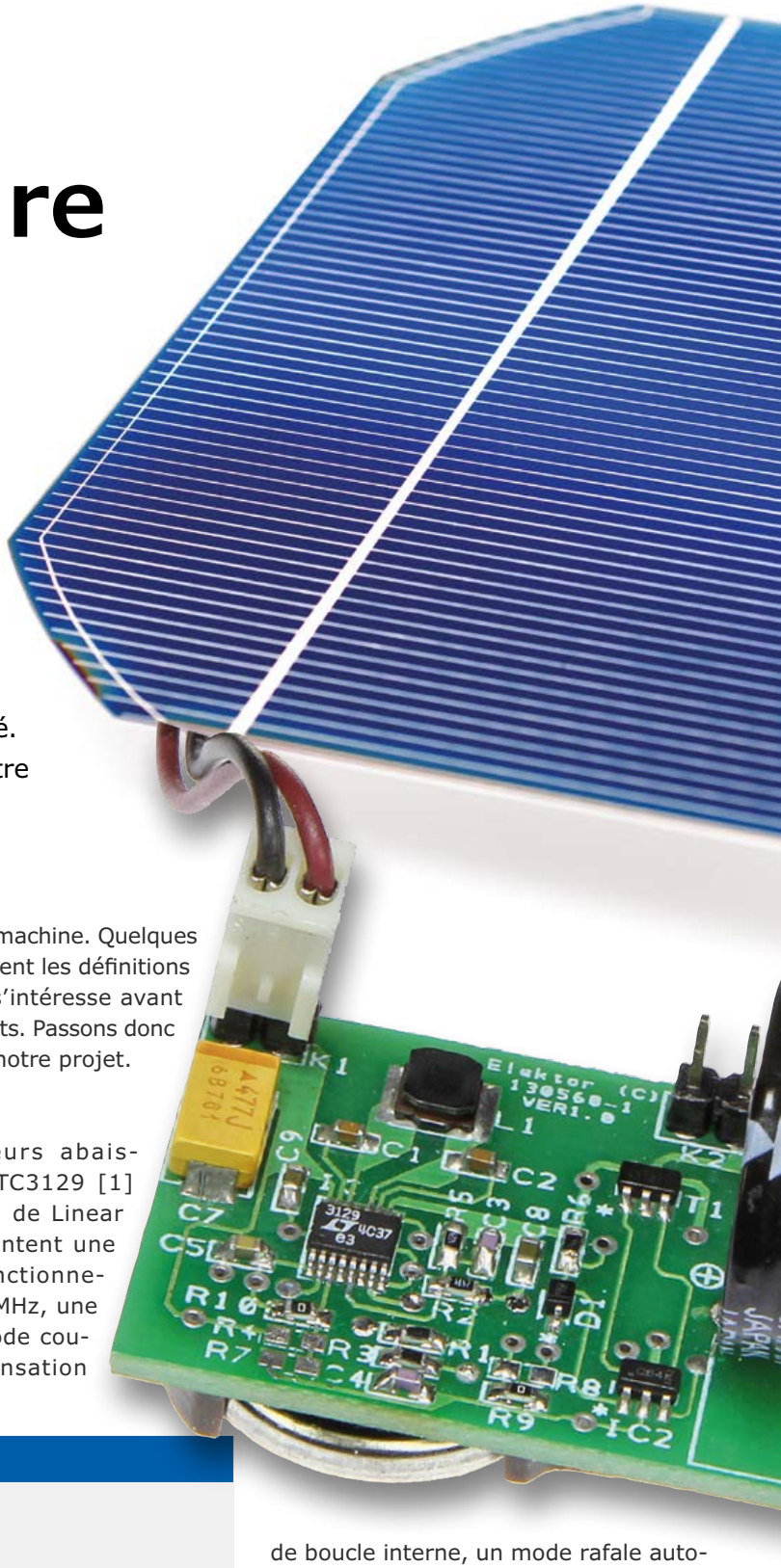


convertisseur de tension de panneau solaire

pour éclairage intérieur et IdO

Sunil Malekar (Elektor Inde)

Ce circuit sans microcontrôleur permet d'alimenter de petits dispositifs de l'IdO à partir d'un panneau solaire et d'un éclairage intérieur. La puissance d'entrée très faible, à peine 7,5 µW, peut donc provenir d'un mini-panneau solaire bon marché. La carte est équipée d'une pile de secours et a pu être rendue très compacte grâce au LTC3129.



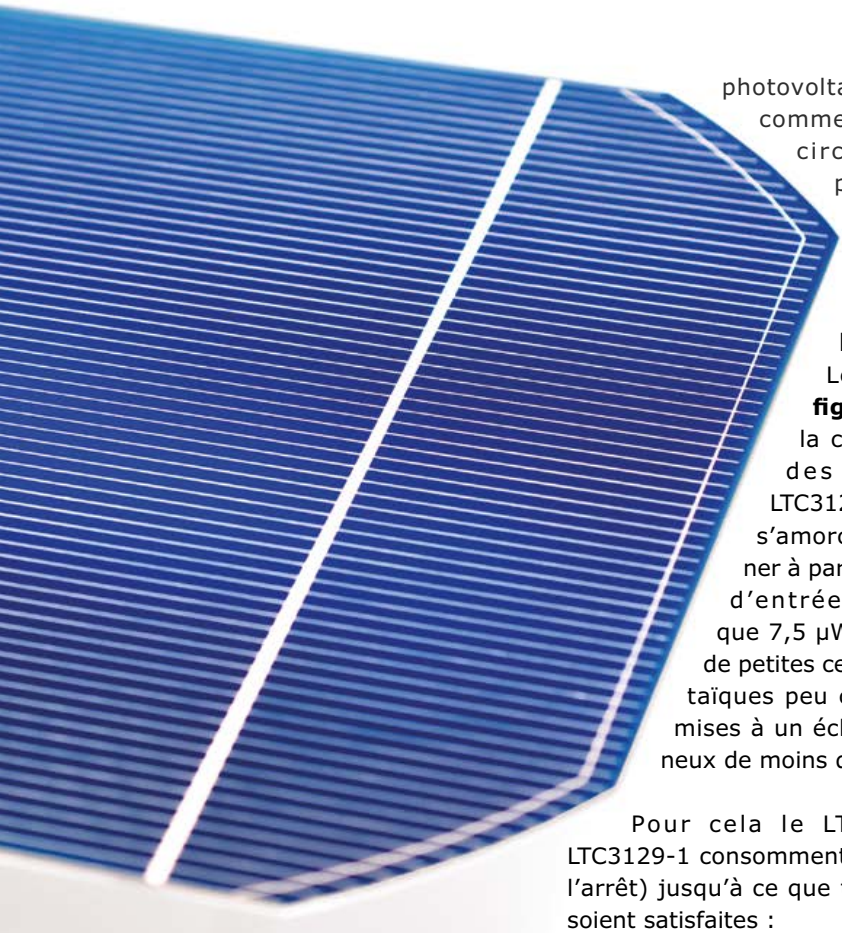
L'Internet des Objets (IdO) peut être défini comme un réseau d'objets, dont l'électronique, les programmes, capteurs et moyens de connexion embarqués fournissent des services par échange de données entre les dispositifs connectés. On peut aussi voir l'IdO comme un continuum mondial interconnecté de dispositifs, qui serait né du développement d'une technologie bon marché ne nécessitant pas de licence : la RFID. D'autres encore voient l'IdO comme la possibilité d'attacher un identifiant unique à tout être physique ou vivant, et de transférer des données via un réseau sans interaction interhumaine ou homme-machine. Quelques éloquentes que soient les définitions de l'IdO, Elektor s'intéresse avant tout aux composants. Passons donc en revue ceux de notre projet.

Les LTC3129

Les convertisseurs abaisseur/élévateur LTC3129 [1] et LTC3129-1 [2] de Linear Technology présentent une fréquence de fonctionnement fixe de 1,2 MHz, une commande en mode courant, une compensation de boucle interne, un mode rafale automatique, un mode MLI à faible bruit, une broche *RUN* avec un seuil précis de déclenchement de la fonction UVLO, un signal *Power Good*, et une fonction *MPPC* (*Maximum Power Point Control*, contrôle du point de puissance maximal) qui optimise le transfert d'énergie provenant de sources non idéales comme les cellules

Caractéristiques

- Stockage de l'énergie de l'éclairage intérieur
- Panneau solaire type : 5 V et 42 µA
- Puissance d'entrée utilisable à partir de 7,5 µW
- Sortie type de 3,2 V
- Convertisseur LTC3129 ou LTC3129-1 ; carte configurable pour les deux circuits intégrés
- Nombreuses tensions de sortie, simples à configurer
- Pile de secours optionnelle
- Condensateur de grande capacité optionnel
- Carte assemblée avec LTC3129-1, 3,3 V et mode MPPC



photovoltaïques. Voyons comment utiliser ces circuits intégrés pour rendre l'IdO écolo.

Alimenter (aussi) l'IdO

Le circuit de la **figure 1** exploite la capacité unique des LTC3129 et LTC3129-1 de pouvoir s'amorcer et fonctionner à partir d'une source d'entrée aussi faible que 7,5 µW, donc à partir de petites cellules photovoltaïques peu coûteuses soumises à un éclairement lumineux de moins de 200 lux.

Pour cela le LTC3129 et le LTC3129-1 consomment 2 µA (moins à l'arrêt) jusqu'à ce que trois conditions soient satisfaites :

- la tension de la broche *RUN* excède 1,22 V (valeur type) ;
- la tension de la broche *VIN* excède 1,9 V (valeur type) ;
- V_{CC} (produite en interne à partir de V_{IN}, mais la source peut être externe) excède 2,25 V (valeur type).

Le circuit intégré reste dans un état de veille « douce » en ne consommant que 2 µA tant que ces trois conditions ne sont pas remplies. Cet état permet à une petite source d'entrée de charger le condensateur de stockage jusqu'à ce que la tension soit assez élevée pour satisfaire les trois conditions. Le LTC commence alors à commuter et V_{OUT} est régulée si le condensateur a stocké suffisamment d'énergie. L'alimentation de secours est construite autour de BAT1 (il manque un T, désolé). Une pile CR2032 alimente le circuit lorsque l'énergie lumineuse est insuffisante, le LTC3129 est alors utilisé pour programmer V_{OUT} sur 3,2 V, une valeur mieux en accord avec la tension de la pile. Une tension d'entrée de 5 V délivrée par un panneau solaire relié à K1 donne une

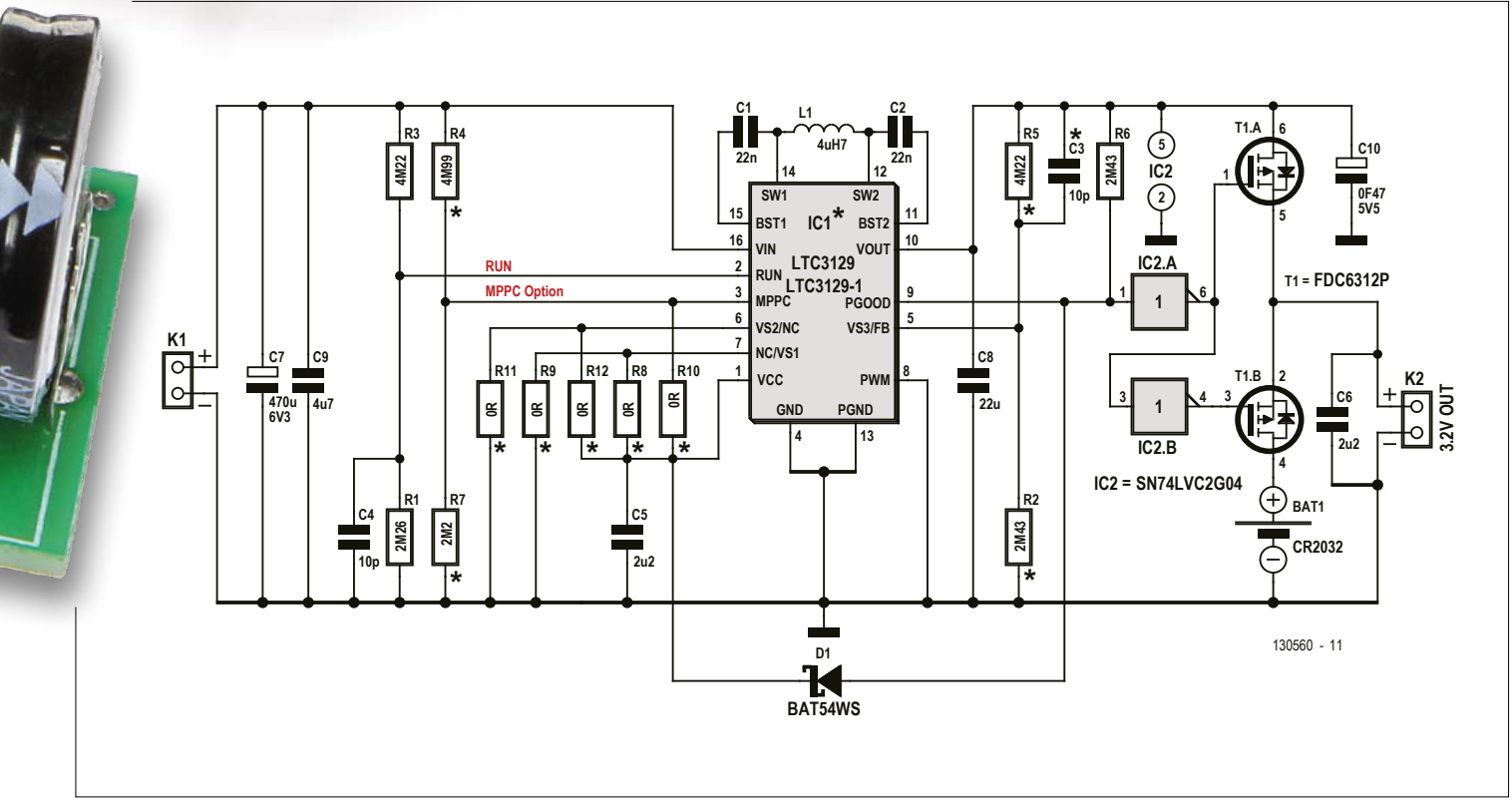


Figure 1. Le cœur du convertisseur peut être au choix un LTC3129 ou un LTC3129-1. Certains composants doivent être configurés en fonction du LTC utilisé. Le circuit dissipe à peine un microwatt de l'énergie lumineuse récoltée.

Liste des composants

Résistances
R1 = 2,26 MΩ, 1 %, 0,063 W, 0603 *
R2, R6 = 2,43 MΩ, 1 %, 0603 (R2*)
R3, R5 = 4,22 MΩ, 1 %, 0603 (R5*)
R4 = 4,99 MΩ, 1 %, 100 mW, *
R7 = 2,2 MΩ, 1 %, 0603 *
R8 à R12 = 0 Ω, 0603 *

Condensateurs
C1, C2 = 22 nF, 25 V, 0603
C3, C4 = 10 pF, 25 V, 0603
C5, C6 = 2,2 µF, 25 V, 0603
C7 = 470 µF, 6,3 V, boîtier D
C8 = 22 µF, 10 V, 0603
C9 = 4,7 µF, 25 V, 0603
C10 = 0,47 F, 5,5 V, super-condensateur, radial

Semi-conducteurs
IC1 = LTC3129EMSE#PBF ou LTC3129-1 *
IC2 = SN74LVC2G04DBVR
T1 = FDC6312P (Newark/Farnell # 1700713)
D1 = BAT54WS-E3-08

Inductance
L1 = 4,7 µH, 20 %, CMS

Divers
BAT1 = pile bouton au lithium, CR2032, 3 V, 20 mm, avec support encartable
K1,K2 = barrette à 2 broches
panneau solaire, AM-1815CA, 58,1 x 48,6 mm (Panasonic)
circuit imprimé, réf. e-choppe 130560-1 v. 1.0
carte assemblée, réf. e-choppe 130560-91. Version : LTC3129-1, sortie 3,3 V, MPPC, pile non fournie

* composant, valeur et/ou utilisation en fonction de la configuration de l'utilisateur ; voir texte

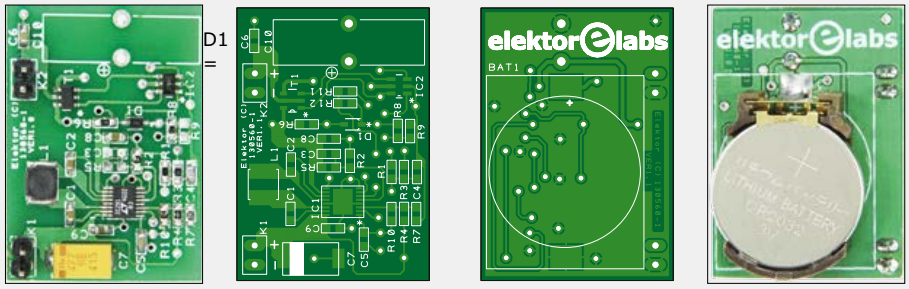


Figure 2. Le circuit imprimé très compact de ce convertisseur d'énergie solaire recourt pour l'essentiel à des composants CMS. La carte est disponible nue (réf. 130560-1) ou assemblée (réf. 130560-91, cf. texte et liste des composants) (photo du prototype ici).

sortie de 3,2 V en K2. La tension de sortie peut être adaptée aux exigences du projet en configurant le diviseur de rétroaction (voir plus bas) dans le cas du LTC3129, ou en paramétrant les trois broches programmables dans le cas du LTC3129-1. Le circuit stabilise la tension de sortie et peut aussi fonctionner sans pile bouton.

Puisqu'il est prévu pour une utilisation en intérieur, le circuit comprend un condensateur C10 chargé de stocker l'énergie de la lumière ambiante. Lorsque ce condensateur de grande capacité se charge, le circuit ne délivre pas la tension de sortie nominale. T1, un double MOSFET FDC6312P de Fairchild Semiconductor à canal P de 1,8 V et spécifié PowerTrench® [3], commute les sorties du convertisseur (V_{OUT}) et de la pile. La sélection de la sortie se fait avec les deux portes inverseuses d'IC2 (74LVC2G04) et sous le contrôle du signal *PGOOD* délivré par le CI convertisseur lorsque les tensions d'entrée et de sortie sont à leurs valeurs nominales. Le signal *PGOOD* n'est pas produit si la tension d'entrée est trop basse, ce qui force la pile de secours à délivrer la tension de sortie.
Les broches *RUN* du LTC3129 et du LTC3129-1 ont des configurations et exigences différentes. La broche *RUN* est une entrée vers le comparateur *RUN*. Sa tension doit être supérieure à 1,1 V pour activer le régulateur V_{CC}, supérieure à 1,22 V pour activer le convertisseur. Relier cette broche à un pont diviseur placé entre V_{IN} et la masse permet de programmer pour V_{IN} un seuil d'amorçage supérieur au seuil (type) de 1,8 V. Dans notre cas le seuil est calculé avec :

$$V_{IN} = 1{,}22 \times [1 + (R3 / R1)]$$

Il faut une résistance de grande valeur puisque l'intensité d'entrée se mesure en µA. Notre relation devient, avec 4,22 MΩ pour R3 et 3,5 V pour V_{IN} :

$$1{,}22 \times [1 + (4{,}22 \times 10^6 / R1)] = 3{,}5$$

Soit ici R1 = 2,26 MΩ.
L'encadré **Configuration** fournit d'autres calculs et valeurs de configuration pour le LTC3129 et le LTC3129-1.

Carte pourvue du LTC3129

Lorsqu'un panneau solaire de 5 V et 42 µA est relié en K1, le condensateur C7 se charge. Comme ce courant de charge est faible, il faut 20 à 30 s pour que la tension atteigne 5 V. Lorsque la tension d'entrée atteint 3,5 V, la tension aux bornes de la broche *RUN* vaut 1,22 V grâce au pont diviseur, ce qui active la sortie du convertisseur. Au même moment le signal *PGOOD* est produit et IC2.A force le double transistor T1 à délivrer la tension de sortie du convertisseur IC1 au connecteur K2. La réaction de V_{OUT} est envoyée à la broche *Feedback* d'IC1 via le diviseur R5/R2 qui détermine la tension de sortie disponible sur K2 (entre 3 et 3,2 V). Le condensateur de couplage attaché au diviseur de réaction R5/R2 atténue les ondulations du mode rafale sur la tension de sortie.

Lorsqu'une pile de 3 V est connectée et qu'aucun panneau n'est relié ou que la luminosité est très faible, IC2 active T1.B pour transmettre le courant de sortie à la charge reliée à K2.
Lorsque le circuit intégré convertisseur ne délivre pas de signal *PGOOD*, la pile peut servir de source de courant en attendant que le panneau solaire soit suffisamment éclairé.
Le super-condensateur C10 se charge en présence d'un panneau, mais il peut mettre entre 8 et 12 h avant d'atteindre la valeur nominale de sortie de 3 à 3,2 V puisque l'intensité du courant ne se mesure qu'en µA. Il n'en reste pas moins que ce processus stocke effectivement une énergie utilisable.

Carte pourvue du LTC3129-1

Les deux circuits intégrés sont dotés des mêmes fonctions, seule diffère leur configuration dans le circuit. Avec le **LTC3129,** c'est un **pont diviseur** relié à la broche *Feedback* qui définit la valeur de la tension de sortie, tandis qu'avec le **LTC3129-1** ce sont les **broches VS1, VS2 et VS3** qui servent à définir cette tension.
Une sortie de 3,3 V s'obtient ainsi : VS1 (broche 7) reliée à la broche *VCC* via la

résistance de 0 Ω R8, VS2 (broche 6) reliée directement à la masse, et VS3 (broche 5) également reliée à la masse mais via une résistance R2 de 0 Ω.
La broche *MPPC* est reliée à la broche *VCC* via la résistance de 0 Ω R10 car sa fonction nécessite une source d'entrée supérieure à 10 mA. Pour une configuration et une source d'entrée différentes, cette fonction peut être utilisée et définie via le pont diviseur R4/R7, comme expliqué ci-dessus. Les autres fonctions, comme l'alimentation de secours et le rôle du super-condensateur C10, sont semblables à celles de la carte équipée du LTC3129.

Assemblage et essai

Si vous avez accès à un matériel de fabrication (CMS) professionnel, vous pouvez fabriquer le circuit imprimé (**fig. 2**) en soudant le LTC3129 ou bien le LTC3129-1 accompagnés de leurs composants. La liste des composants est là pour ceux qui souhaitent assembler eux-mêmes le circuit, mais nous nous permettons de recommander la carte assemblée, disponible dans l'e-choppe (réf. **130560-91**) : carte équipée du LTC3129-1, avec mode MPPC, min. 10 mA en entrée et sortie de 3,3 V (pile non fournie).
Sur le schéma, les composants marqués d'une étoile * sont optionnels et/ou dépendent du circuit intégré utilisé. Les valeurs des résistances des ponts diviseurs associés aux broches *Feedback*, *RUN* et *MPPC* dépendront de vos choix de configuration ou des tensions d'entrée/sortie souhaitées.

Pour ce qui est de la version LTC3129-1, la sortie peut être configurée jusqu'à 5 V en changeant simplement la configuration résistive des broches VS1 et VS2. Il faut monter soit R11, soit R12, mais pas les deux.
Utiliser ou non la pile de secours et le super-condensateur dépend du projet ; la carte peut fonctionner sans eux.

L'économie protège du besoin

La carte s'alimente en reliant à K1 un panneau solaire de tension de sortie maximale 5 V, la charge se reliant à K2. Un panneau comme le AM-1815 de Sanyo peut alimenter la carte à partir d'un éclairage intérieur, et si le super-condensateur C10 est utilisé, une charge utile s'accumulera.
Un courant de charge de p. ex. 35 µA donnera au bout de 24 h une charge de presque 700 µAh, soit 7 mA de courant durant 6 min, ou 42 mA durant 1 min.

La carte est donc adaptée aux applications dont les exigences de consommation sont situées dans cette plage de valeurs. Elle devrait donc convenir à la plupart des objets de l'IdO qui sont réveillés, répondent brièvement, puis replongent dans leur cybersommeil. Dites-nous ce que vous comptez alimenter avec cette carte !

(130560 – version française : Hervé Moreau)

Liens

[1] LTC3129 : www.linear.com/docs/42736
[2] LTC3129-1 : www.linear.com/docs/42735
[3] FDC6312P : www.fairchildsemi.com/pf/FD/FDC6312P.html

Configuration

Fonction MPPC des LPC3129 et LPC3129-1

Relier la broche *MPPC* d'un LTC3129 à un pont diviseur placé entre V_{IN} et la masse active la fonction MPPC (*Maximum Power Point Control*). Si la charge sur V_{OUT} excède la capacité de la source d'alimentation, la MPPC réduit le courant de l'inductance pour réguler V_{IN} sur une valeur donnée par :

$V_{IN} = 1{,}175 \times [1 + (R4 / R7)]$

Soit, avec R4 = 4,99 MΩ :

$1{,}175 \times [1 + (4{,}99 \times 10^6 / R7)] = 3{,}2$ V
$[1 + (4{,}99 \times 10^6 / R7)] = 2{,}9787$ V
R7 = 2,13 MΩ ≈ 2,2 MΩ

Une valeur adéquate de la tension de régulation V_{IN} assure un transfert maximal de la puissance produite par la source. Notez que la broche est sensible au bruit, il faut donc éviter les pistes de circuit imprimé trop longues et les capacités parasites.
La source délivre ici moins de 10 mA et la MPPC n'est donc pas utilisée. Sa broche est reliée à la broche VCC via la résistance R10 de 0 Ω. Pour activer la MPPC, ôtez R10 et configurez le pont diviseur R4/R7.

Signal de réaction sur LTC3129

La broche *Feedback* sert d'entrée de réaction vers l'amplificateur d'erreur. Elle est reliée à un pont diviseur placé entre V_{OUT} et la masse. Pour obtenir une tension de sortie de 3,2 V, nous calculons, en prenant R5 = 4,22 MΩ :

$V_{OUT} = 1{,}175 \times [1 + (R5 / R2)]$
R2 = 2,44 MΩ ≈ 2,43 MΩ

Configuration de la tension de sortie sur LTC3129-1

La tension de sortie se définit en reliant les broches de sélection VS1, VS2 et VS3 soit à V_{CC}, soit à la broche GND. Ces broches ne doivent être ni flottantes ni négatives. Le tableau ci-dessous montre la correspondance entre leur configuration et la tension de sortie.

VS3	VS2	VS1	V_{sortie}
0	0	0	2,5 V
0	0	V_{CC}	3,3 V
0	V_{CC}	0	4,1 V
0	V_{CC}	V_{CC}	5 V

e-BoB BL600

Bluetooth Low Energy

(5e partie)

- Port SPI & convertisseur numérique-analogique
- Application Android

Jennifer Aubinais (Paris) elektor@aubinais.net

Après le port I²C du BL600 examiné le mois dernier, nous nous intéressons ici à son interface sérielle. Pour cela, nous utiliserons un convertisseur numérique-analogique équipé lui-même d'un port SPI. Ce sera l'occasion de réaliser votre propre application Bluetooth.

Dans l'arsenal impressionnant du BL600, nous examinons ici son interface sérielle dite SPI. Pour cela, nous utiliserons un convertisseur numérique-analogique (CNA) à 16 bits équipé lui-même d'un port SPI. Ce sera l'occasion de réaliser votre propre application Bluetooth. L'objectif n'est évidemment pas un cours de développement sous Android, mais tous les éléments sont donnés dans l'article pour que désormais votre téléphone et le BL600 fassent bon ménage. Si la demande est suffisante, nous présenterons peut-être un jour ici une application Android consacrée entièrement à notre composant SPI.

NB : Les outils et les commandes utilisés dans cet article ont été présentés dans les articles précédents de cette série. [1].

Port SPI et CNA

Le CNA LTC1655L de *Linear Technology* [2] peut être alimenté sous 3,3 V et, si nous utilisons sa référence de tension interne, il fonctionne sans autre composant (**fig. 1**). Dans ce cas, sa tension de sortie est limitée entre 0 à 2,5 V. La valeur que nous souhaitons donner à cette tension de sortie sera envoyée sur le port série du CNA par le port SPI de notre e-Bob BL600 (broche 10, MOSI = *Master Output, Slave Input* et broche 12, CLK) (**fig. 2**). Le CNA dispose aussi d'une sortie série que nous avons connectée à l'entrée du port SPI de l'e-BoB (broche 11, MISO = *Master Input, Slave Output*), ce qui nous permettra de vérifier le fonctionnement de notre montage. Avec un voltmètre relié à la sortie du CNA (broche 7), nous mesurerons sa tension de sortie et ajusterons la commande de façon à obtenir une tension de 1 V par exemple.

Notre module BL600 et le CNA sont maintenant connectés. Intéressons-nous au programme en SmartBASIC du BL600. Voici ce qui se passe dans le programme *LTC1655L.sb* (**listage 1**) téléchargeable sur le site d'Elektor [4] :

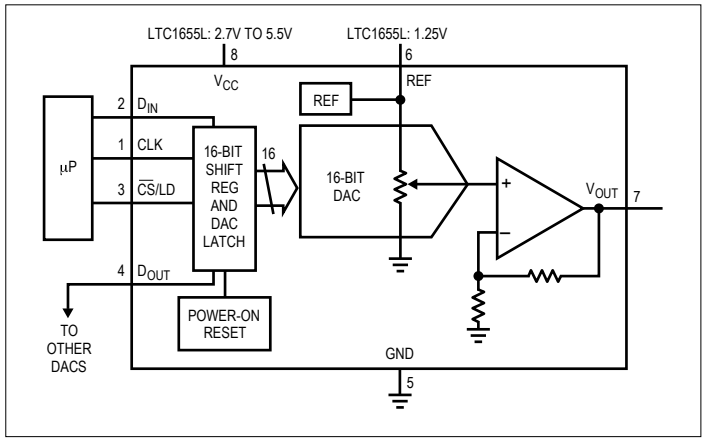


Figure 1. Circuit interne du convertisseur analogique numérique LTC1655L utilisé pour illustrer la communication par l'interface SPI.

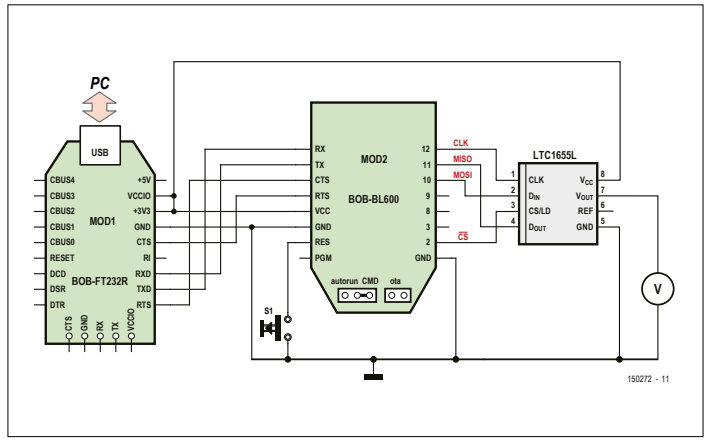


Figure 2. Schéma du BL600 avec le LTC1655L. La mise en œuvre de l'e-BoB FT232 a été décrite dans les épisodes précédents.

a) Open SPI

La première étape est d'ouvrir le port SPI et de traiter son code retour par une simple condition *if*.

```
Rc = SpiOpen(0,1000000,0,handle)
```

- **nMode** = 0 : CPOL (*Clock Polarity*) à 0 et CPHA (*Clock Phase*) à 0
- **nClockHz** = 1000000 : fréquence d'horloge
- **nCfgFlags** = 0 : doit être mis à 0.
- **nHandle** = si le code de retour est 0, alors cette variable peut ensuite être utilisée pour lire, écrire et fermer l'interface SPI.
  rc = le code retour doit être zéro pour pouvoir continuer. Si ce n'est pas le cas, nous convertissons sa valeur en une chaîne de caractères par la fonction *SPRINT* avec l'option *INTEGER.H* et nous l'affichons. Puis nous arrêtons le programme (*STOP*).

b) Écrire et lire des octets SPI

L'objectif est d'obtenir 1 V à la sortie du CNA. Nous lirons aussi les données que nous avons envoyées, ce sera un bon exercice.

- **Calcul pour une tension de sortie de 1 V**
  La plage de tension de sortie de 0 à 2,5 V du CNA est couverte avec des mots de seize bits (2^{16}) de 0 à 65.536. Sa résolution est donc de 2,5/65536 ≈ 38 µV. La valeur à envoyer pour obtenir 1 V sera donc (1/2,5) x 65536 ≈ 26.214 (= 6666 en hexadécimal, noté 0x6666).

- **Écriture des deux octets**
  Comme la fonction d'envoi de données par le port SPI n'accepte que des chaînes de caractères ASCII, il faudra convertir les valeurs numériques à envoyer. Or, le CNA attend un mot de seize bits, de sorte que nous aurons à convertir la valeur numérique 26.214 en deux valeurs hexadécimales qui formeront la chaîne de caractères 0x66-0x66, soit *ff* en ASCII. En envoyant cette chaîne, nous obtenons une tension d'1 V à la sortie du CNA.

```
Rc = SpiReadWrite(stWrite$,stRead$)
```

- **stWrite$** = chaîne de caractères des données à écrire
- **stRead$** = chaîne de caractères des données lues de la même longueur que la chaîne de caractères **stWrite$.**
- **rc =** Le code de retour doit être à 0.

- **Lecture du port SPI**
  Le port SPI est synchrone (*full duplex*), on y écrit et on y lit en même temps. Notre fonction d'écriture de données sur le port SPI est également une fonction de lecture. Et puisque la sortie série du CNA est bouclée sur l'entrée MISO de l'e-BoB et que le CNA renvoie les données reçues, nous recevons donc en écho les données envoyées lors de la précédente opération d'écriture à notre CNA.

Liste des composants :

Semi-conducteurs :
IC1 = LTC1655L

Divers :
S1 = poussoir
MOD1 = e-Bob FT232 assemblé 110553-91 [7]
MOD2 = e-Bob BL600 assemblé 140270-91 [7]
ou circuit imprimé 140270-1 [7]

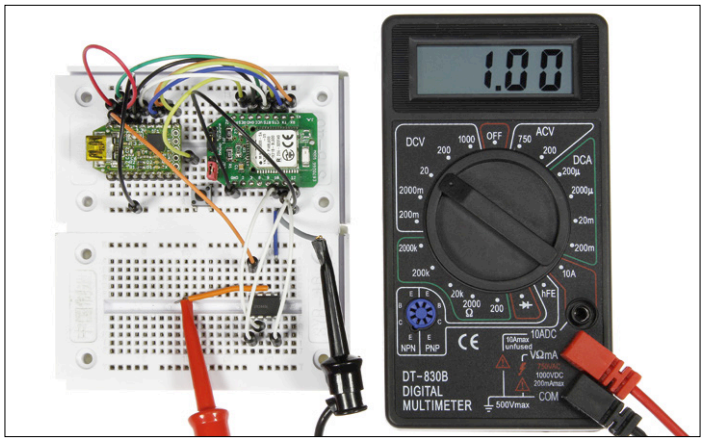


Figure 3. Photo de l'assemblage expérimental sur une plaque d'essais.

Listage 1

```
//--------------------------------------------------
DIM rc
DIM handle
DIM stWrite$, stRead$
//--------------------------------------------------
rc = GpioSetFunc(2,2,0)    // pin 2 at low
//--------------------------------------------------
rc=SpiOpen(0,1000000,0,handle)
if rc!= 0 then
print "\nFailed to open SPI with error code
";integer.h' rc
else
print "\nSPI open success"
endif
//--------------------------------------------------
GpioWrite(2,0) // pin 2 at low
stWrite$ = "\66\66" : stRead$=""
rc = SpiReadWrite(stWrite$, stRead$)
if rc!= 0 then
print "\nFailed to ReadWrite"
else
print "\nWrite = 0x";strhexize$(stWrite$)
endif
//--------------------------------------------------
stRead$=""
rc = SpiReadWrite(stWrite$, stRead$)
if rc!= 0 then
print "\nFailed to ReadWrite"
else
  print "\nRead = 0x";strhexize$(stRead$)
endif
//--------------------------------------------------
GpioWrite(2,1) // pin 2 at high
SpiClose(handle)  //close the port
SpiClose(handle)  //no harm done doing it again
print "\nCLOSE SPI\n"
//--------------------------------------------------
```

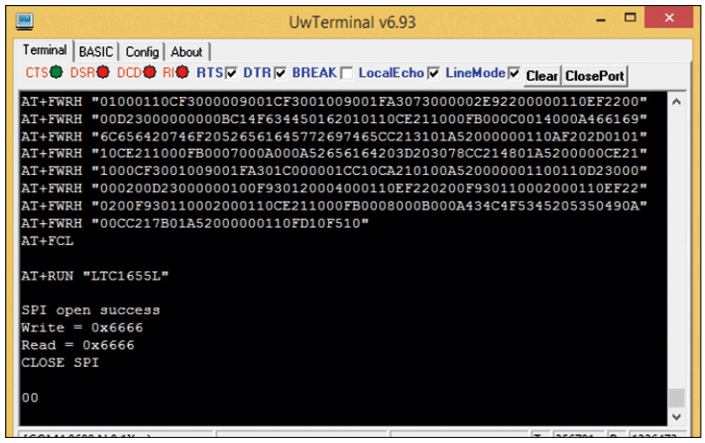
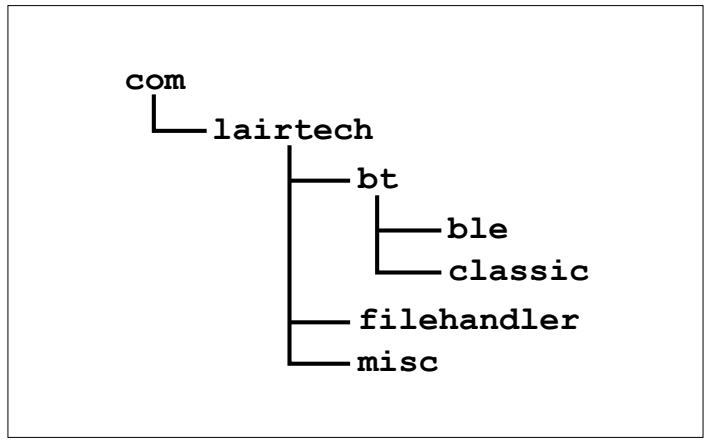


Figure 4. Si le voltmètre affiche une tension de 1 V, le programme *LTC1655L.sb* s'est déroulé avec succès.



L'arborescence de la bibliothèque

c) Close SPI

Pour finir, on referme le port SPI.

```
SpiClose(nHandle)
```

- **nHandle** = valeur créée par SPIOpen

L'application Android

Le moment est venu de se lancer dans la programmation d'une application sous Android. Sur le site du fabricant [3] du BL600, vous trouverez le code source de l'application *Toolkit* qui comprend les services suivants :

- BPM (pression artérielle)
- HRM (rythme cardiaque)
- *Proximity*
- HTM (thermomètre médical) utilisée avec le capteur de température dans le précédent article.
- Serial (UART)
- OTA (*Over The Air*)
- Batch

  Nous vous proposons en téléchargement [4] le code source simplifié à l'extrême d'une application utilisant seulement le service UART. Nous avons utilisé *Android Studio*, disponible sous Windows, MAC OS et Linux [5]

Le fabricant a créé une bibliothèque *laird_library_ver.0.18.1.1.jar* afin d'accélérer le développement d'applications Android en Bluetooth normal et Bluetooth Low Energy. Une documentation est disponible avec le téléchargement de la bibliothèque.

Il y a trois classes :

- `BluetoothAdapterWrapper Class`
  - Initialisation du Bluetooth
  - Vérifie si le Bluetooth est présent
  - Détecte les périphériques Bluetooth
- `BluetoothAdapterWrapperCallback Interface`
  - Détecte, arrête les dispositifs Bluetooth
  - Traitement des Call-Back
- `BleDeviceBase Abstract Class`
  - Accès aux méthodes Bluetooth pour initialiser la connexion au périphérique. Par exemple : `connect, isConnected ….`

Dans notre programme, vous trouverez l'arborescence de la bibliothèque reproduite ci-dessus.

Le programme

L'ensemble du programme est téléchargeable sur le site d'Elektor [4]. La première étape est de créer son projet. Nous avons choisi comme *Package name* : `com.ja.serial` (où ja sont les initiales de l'auteure)

L'agencement de l'écran : Pour commencer, restons simples : deux boutons *btnScan* et *btnSend*, une zone d'affichage des données reçues avec le scrolling *scrollViewVspOut* et *valueVspOutTv* et la zone texte de saisie *valueVspInputEt*. C'est tout (**fig. 5).**

Exemple du *activity_main.xml* de la zone d'affichage des données reçues avec le scrolling (**listage 2**).

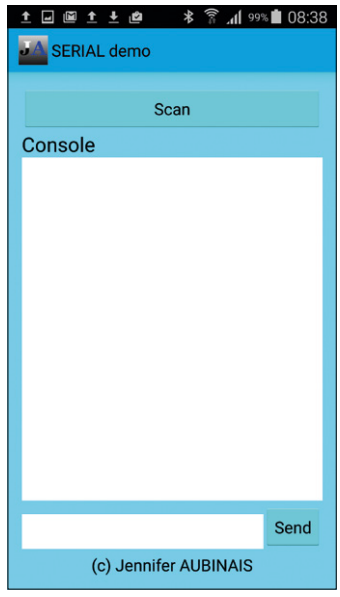


Figure 5. Au plus simple : l'écran de notre application Serial.

Figure 6. Affichage de la class Dialog mDialogFoundDevices avec deux périphériques.

Les permissions : Le fichier *AndroidManifest.xml* autorise notre programme à accéder au Bluetooth. Si vous l'oubliez, il en résultera dans l'application une erreur difficile à identifier.

```
    <uses-permission android:name="android.permission.
BLUETOOTH" />
    <uses-permission android:name="android.permission.
BLUETOOTH_ADMIN" />
```

La déclaration dans le programme : Nous allons importer différentes *class* sans aucun changement des sources originales dans les répertoires suivants :

-
- com.ja.bt.ble.vsp
  - FifoAndVspManager.java
  - FifoAndVspUiBindingCallback.java
  - FileAndFifoAndVspManager.java
  - VirtualSerialPortDevice.java
  - VirtualSerialPortDeviceCallback.java
- com.ja.serial.bases
  - BaseActivityUiCallback.java
- com.ja.serial.serialdevice
  - SerialManager.java
  - SerialManagerUiCallback.java
- com.ja.serial
  - ListFoundDevicesHandler.java

L'importation des classes de la bibliothèque du fabricant est décrite dans le **listage 3.**

Etant donné qu'il y a une implémentation des deux classes *SerialManagerUiCallback* et *BluetoothAdapterWrapperCallback,* vous trouverez toutes leurs classes spécifiées dans notre programme. Celles-ci sont reconnaissables par la présence des commentaires en début d'énumération (**listage 4**).

La class Dialog apparaît lors de la recherche (*scan*) des périphériques. Puis, à partir de la liste, l'application se connecte à notre e-BoB BL600. Attention, l'application fonctionne seulement avec les périphériques ayant le service UART.

Les boutons : Nous retrouvons le *setOnClickListener* des deux boutons de notre application : *btnScan* et *btnSend*. Voici comment ça se passe pour *btnSend :* le texte à envoyer est lu dans la zone de texte ; s'il n'est pas nul, il est recopié dans la zone de texte de réception puis envoyé par la fonction *startDataTransfer* (**listage 5**).

(Astuce) Lancement automatique de la connexion : Pour établir une connexion automatique, rappelons-nous comment nous le faisons manuellement : nous lançons la recherche par le bouton *btnScan* puis nous choisissons notre périphérique dans la boite de dialogue *Dialog*. Pour notre connexion automatique, nous laissons donc le programme principal exécuter le code de la fonction *setOnClickListener* puis, au lieu d'afficher la liste des périphériques de dialogue *Dialog*, nous lancerons la connexion au module renseigné dans le code *name*.

(Astuce) Réception et envoi des données : Pour saisir les données à envoyer et afficher les données reçues, nous avons deux zones de texte : *valueVspInputEt* et *valueVspOutTv*. Rien n'empêche votre programme d'envoyer les données par une autre action. Par exemple, vous pourriez envoyer 0 ou 1 selon la position d'un bouton *Switch*. Vous pouvez traiter les caractères reçus dans votre programme comme nous avons fait dans l'application Thermomètre décrite dans le numéro de janvier/février 2015 [6]. Là, nous avions effectué un calcul complexe qui ne pouvait pas être réalisé par le BL600 avant d'afficher le résultat en gros caractères.

Le convertisseur N/A en Bluetooth

Vous savez désormais comment créer une application Bluetooth pour l'e-Bob BL600. Il suffit de prendre comme base dans le paquet du firmware téléchargeable sur le site du fabricant [3] le programme *upass.vsp.sb*, renommé *$autorun$.LTC1655L.*

Listage 2

```
<LinearLayout
     android:orientation="vertical"
     android:layout_width="match_parent"
     android:layout_height="match_parent"
     android:layout_below="@+id/textView"
     android:layout_alignParentStart="true"
     android:layout_above="@+id/btnSend">

     <ScrollView
          android:layout_width="match_parent"
          android:layout_height="match_parent"
          android:id="@+id/scrollViewVspOut"
          android:background="#ffffffff">

          <TextView
               android:layout_width="wrap_content"
               android:layout_height="wrap_content"
               android:textSize="20sp"
               android:id="@+id/valueVspOutTv" />
     </ScrollView>
 </LinearLayout>
```

Listage 3

```
import com.lairdtech.bt.BluetoothAdapterWrapperCallback;
import com.lairdtech.bt.BluetoothAdapterWrapper;
import com.lairdtech.bt.ble.BleDeviceBase;
import com.lairdtech.misc.DebugWrapper;
```

Listage 4

```
/*
 * ************************************
 * SerialManagerUiCallback
 * ************************************
 */
/*
 ************************************
 * Bluetooth adapter callbacks
 * ************************************
 */
```

listage 5

```
mBtnSend.setOnClickListener(new OnClickListener() {
    @Override
    public void onClick(View v) {
        switch(v.getId())
        {
            case R.id.btnSend:
            {
                String data = mValueVspInputEt.getText().toString();
                if(data != null){
                    mBtnSend.setEnabled(false);
                    mValueVspInputEt.setEnabled(false);
                    if(mValueVspOutTv.getText().length() <= 0){
                        mValueVspOutTv.append(">");
                    } else{
                        mValueVspOutTv.append("\n\n>");
                    }

                    mSerialManager.startDataTransfer(data + "\r");

                    InputMethodManager inputManager = (InputMethodManager)
                            getSystemService(Context.INPUT_METHOD_SERVICE);

                    inputManager.hideSoftInputFromWindow(getCurrentFocus().getWindowToken(),
                            InputMethodManager.HIDE_NOT_ALWAYS);

                    if(isPrefClearTextAfterSending == true){
                        mValueVspInputEt.setText("");
                    } else{
                        // do not clear the text from the editText
                    }
                }
                break;
            }
        }
    }
});
```

uart.sb [4], puis de créer le handler *HandlerLoop* (renommé *MyHandlerLoop*) pour récupérer les données reçues en Bluetooth (**listage 6**).

Voici la liste des différentes actions à effectuer par le *handler* afin de traiter la chaîne de caractères reçue par Bluetooth :

- recevoir par Bluetooth le texte correspondant à la tension souhaitée
- transformer le texte reçu en valeur à envoyer au CNA
- envoyer la valeur sur le port SPI
- recevoir une valeur par le port SPI
- transformer la valeur reçue en une chaîne de caractères
- envoyer la chaîne par Bluetooth.

En rouge : vous avez déjà vu ce code. Nous stockons dans la variable *text$* les caractères reçus en Bluetooth. Si la chaîne de caractères contient le code de retour 0x0D (fin de la chaîne), nous rentrons dans la condition *IF*.
En vert : D'abord on récupère la valeur de la tension voulue à la sortie du CNA. Pour cela, nous convertissons la chaîne de caractères en variable numérique par la fonction *StrValDec* puis nous calculons la valeur pour le CNA. Attention, si la valeur reçue donne un résultat supérieur à 65535 (2^{16}-1) alors nous imposerons 65535, une valeur sur 16 bits. Pour respecter le format de la fonction *SpiReadWrite*, la valeur est convertie en deux caractères par la fonction *StrSetChr* faisant ainsi une chaîne de caractères *stWrite$.*

Listage 6

```
OnEvent  EVVSPRX            call MyHandlerLoop //EVVSPRX is thrown when VSP is open and data has arrived
OnEvent  EVUARTRX           call MyHandlerLoop //EVUARTRX  = data has arrived at the UART interface
OnEvent  EVVSPTXEMPTY       call MyHandlerLoop
OnEvent  EVUARTTXEMPTY      call MyHandlerLoop
```

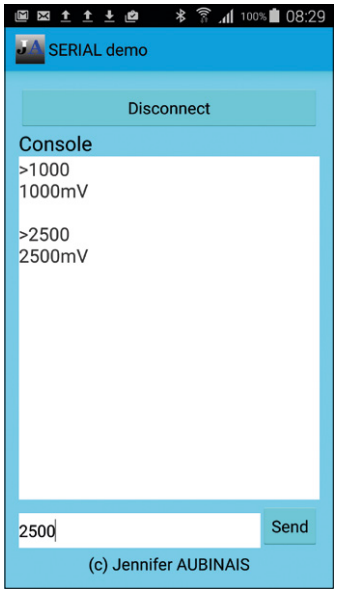


Figure 7. Utilisation de notre application Android pour commander la tension en sortie du CNA LTC1655L.

En bleu : Nous ne revenons pas sur cette partie d'utilisation du port SPI. La fonction *SpiReadWrite* est exécutée deux fois afin de récupérer la valeur du CAN écrite dans le premier *SpiReadWrite*. La valeur écrite ne peut pas être lue sur le port de sortie du CAN, il faudra alors écrire à nouveau pour lire la valeur précédemment chargée. C'est un inconvénient du protocole SPI plutôt que du composant lui-même.

En orange : Maintenant, la valeur lue du CNA est une chaîne de caractères. Il faut extraire les deux valeurs numériques (valeur haute et valeur basse) par la fonction *StrGetChr*. Puis nous calculons la valeur de la tension en millivolts par un calcul inverse. La conversion de la valeur numérique en une chaîne de caractères se fait par la fonction SPRINT. Pour finir, nous transmettons au Bluetooth la valeur de la tension par la fonction *BleVspWrite*. ⏮ (**fig. 7**)

(150272)

Liens

[1] La série des articles publiés sur le BL600 :
1. La communication sans fil sur un plateau www.elektormagazine.fr/140270
2. Éditer, compiler, transférer un programme avec le module BLE, www.elektormagazine.fr/150014
3. programmer en smartBASIC le module BLE www.elektormagazine.fr/150129
4. Le port I2C et son capteur de température www.elektormagazine.fr/1501230

[2] www.linear.com/product/LTC1655

[3] https://laird-ews-support.desk.com/?b_id=1945

[4] Elektor de juillet/août 2015 n° 445 www.elektormagazine.fr/150272

[5] https://developer.android.com/sdk/index.html

[6] Thermomètre sans fil Elektor janvier/février 2015 n° 439/440 www.elektormagazine.fr/140190

[7] e-BoB BL600 www.elektor.fr/bl600-e-bob-140270-91
e-BoB FT232 www.elektor.fr/ft232r-usb-serial-bridge-bob-110553-91

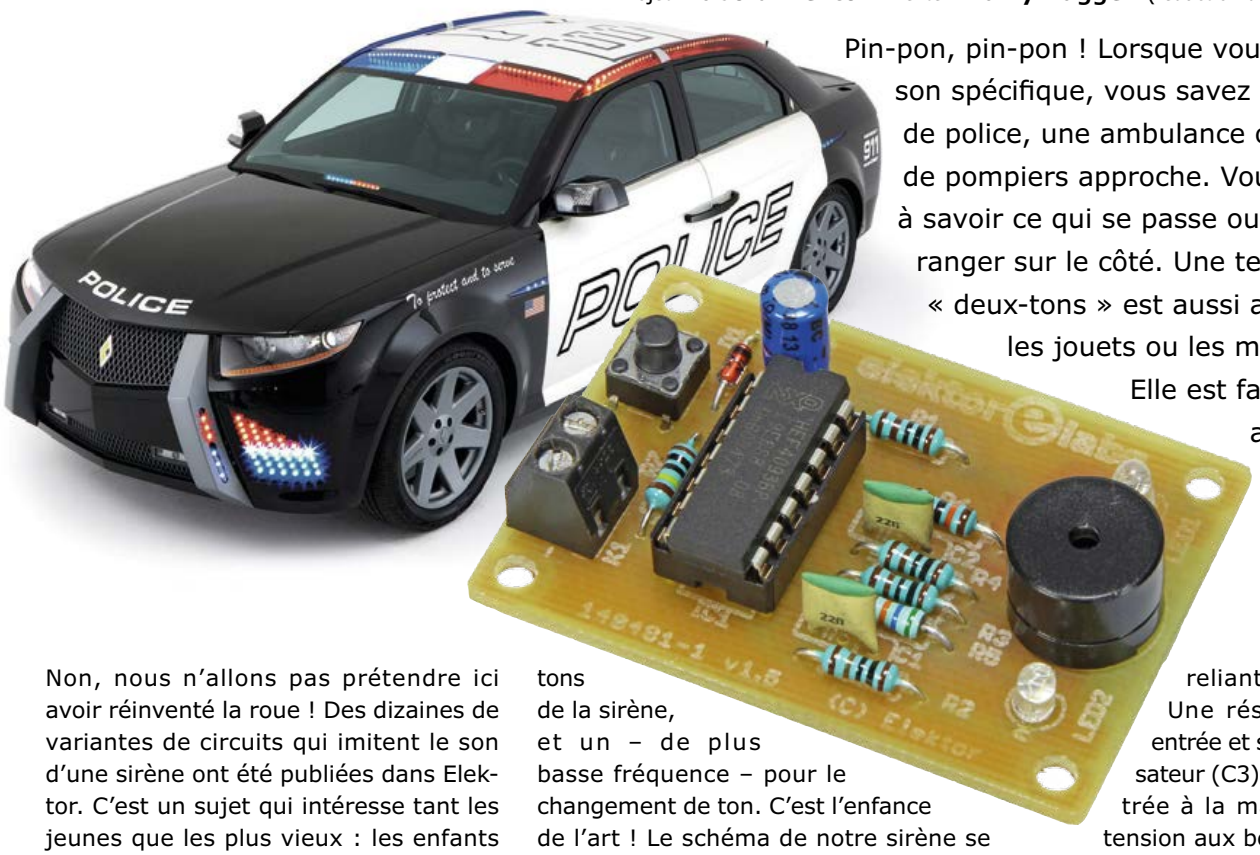
Listage 7

```
function MyHandlerLoop()
  DIM n, rc, tempo$, tx$, text$
  DIM value, valtemp, pos, return$
  DIM handle
  DIM stWrite$, stRead$
  // Wait return from received data
  tx$ = "0D"
  return$ = StrDehexize$(tx$)
  tempo$ = ""
  n = BleVSpRead(tempo$,20)
  text$ = text$ + tempo$
  pos = STRPOS(text$,return$,0)
  IF ( pos >= 0 ) THEN
    //------------------
    // convert Voltage value for LTC1655
    //------------------
    value = StrValDec(text$)
    value = value * 65535
    value = value / 2500
    IF value > 65535 THEN : value = 65535 : ENDIF
    // Init string for SPI write
    stWrite$ = "00"
    valtemp = value / 256
    value = value - (valtemp * 256)
    // write chr value
    rc = StrSetChr(stWrite$,valtemp,0)
    rc = StrSetChr(stWrite$,value,1)
    //------------------
    // SPI
    //------------------
    rc=SpiOpen(0,1000000,0,handle)
    // CS (pin select) at low
    GpioWrite(2,0)
    stRead$=""
    rc = SpiReadWrite(stWrite$, stRead$)
    stRead$=""
    rc = SpiReadWrite(stWrite$, stRead$)
    // CS (pin select) at high
    GpioWrite(2,1)
    // close the SPI port twice
    SpiClose(handle)
    SpiClose(handle)
    //------------------
    // convert SPI value to Voltage
    //------------------
    // read chr value
    valtemp = StrGetChr(stRead$,0)
    value = StrGetChr(stRead$,1)
    value = (valtemp * 256) + value
    value = value * 2500
    value = value / 65535
    // convert value to string
    SPRINT #tx$,value
    tx$ = "\n" + tx$ + "mV"
    // send to Bluetooth
    n = BleVSpWrite(tx$)
    text$ = ""
  ENDIF
ENDFUNC 1
```

sirène de police avec un seul circuit intégré

simple et amusant pour les débutants

Projet : **labo d'Elektor** Texte : **Harry Baggen** (rédaction des Pays-Bas)



Pin-pon, pin-pon ! Lorsque vous entendez ce son spécifique, vous savez qu'une voiture de police, une ambulance ou un camion de pompiers approche. Vous cherchez à savoir ce qui se passe ou bien à vous ranger sur le côté. Une telle sirène « deux-tons » est aussi amusante pour les jouets ou les modèles réduits. Elle est facile à construire avec un seul circuit intégré.

Non, nous n'allons pas prétendre ici avoir réinventé la roue ! Des dizaines de variantes de circuits qui imitent le son d'une sirène ont été publiées dans Elektor. C'est un sujet qui intéresse tant les jeunes que les plus vieux : les enfants jouent à la voiture de police avec tandis que les adultes s'en servent pour agrémenter leur modèle réduit ferroviaire.
Vous pouvez bien sûr consulter les archives d'Elektor pour y trouver un tel modèle de sirène, mais nous vous facilitons le travail avec une nouvelle version et un circuit imprimé. Le labo d'Elektor est souvent plongé dans des projets complexes, et un circuit simple comme celui-ci c'est rafraîchissant.

Et que ça oscille !

De quoi avons-nous besoin pour imiter le son d'une sirène ? Si nous analysons le son d'une vraie sirène, il semble que nous ayons alternativement un son à une certaine fréquence et un autre à une fréquence un peu plus élevée (ou un peu plus basse) ; le changement de ton s'effectue toutes les demi-secondes. Il en découle que nous avons besoin de trois oscillateurs : deux pour la production des tons de la sirène, et un – de plus basse fréquence – pour le changement de ton. C'est l'enfance de l'art ! Le schéma de notre sirène se trouve en **figure 1**, et bien entendu sans microprocesseur, ni composants exotiques. Nous avons choisi comme base de notre circuit un circuit logique 4093 (*Quad 2-Input NAND Schmitt Triggers*). En français : quatre portes NON-ET à deux entrées équipées d'une bascule de Schmitt. Ce dernier point signifie que chaque entrée possède deux seuils de basculement, et réagit donc à un certain niveau de tension. Lorsque le niveau dépasse le seuil supérieur, la sortie bascule (dans ce cas au niveau bas), et ce n'est que lorsque la tension d'entrée redescend en dessous du seuil inférieur que la sortie redevient haute. La bascule de Schmitt a été développée en 1934 et doit son nom à son inventeur, Otto Schmitt.
Une seule résistance et un condensateur suffisent pour réaliser un oscillateur avec une porte NON-ET équipée d'une bascule de Schmitt. Prenons comme exemple la porte IC1.A, configurée en inverseur en reliant les deux entrées. Une résistance (R7) relie entrée et sortie, et un condensateur (C3) est placé entre l'entrée à la masse. Au départ la tension aux bornes du condensateur est nulle, l'entrée de la porte est au niveau bas, et la sortie au niveau haut. C3 se charge progressivement via R7, jusqu'à ce que le seuil de basculement de l'entrée soit atteint. La sortie bascule alors à l'état bas, et C3 se décharge via R7. Et le cycle recommence dès que le seuil inférieur de l'entrée est franchi...
Nous avons un oscillateur ! La **figure 2** montre les signaux à l'entrée et à la sortie de la porte ; le cycle de charge et décharge du condensateur nous donne une tension à la forme triangulaire à l'entrée.
La fréquence de l'oscillateur est déterminée par la résistance et le condensateur. Plus la valeur de la résistance ou du condensateur est élevée, plus basse sera la fréquence d'oscillation. La feuille de caractéristiques du circuit intégré fournit une formule pour le calcul de la fréquence, mais laissons cela de côté au profit de la simplicité.

Nous avons donc reproduit trois circuits RC autour de IC1.A, C et D. La porte IC1.A délivre la fréquence la plus basse (1 Hz), qui servira au basculement entre les deux tons. Les deux autres oscillateurs délivrent une fréquence de 1,2 kHz (IC1.C) et 620 Hz (IC1.D). Le déclenchement est effectué via une des deux entrées des portes NON-ET. IC1.A fonctionne en continu, IC1.D uniquement lorsque la sortie de IC1.A est à l'état haut. Pour déclencher l'autre oscillateur (IC1.C) lorsque la sortie de IC1.A est à l'état bas, nous avons utilisé la porte restante du circuit (IC1.B) comme un inverseur à la sortie de IC1.A. Et pour la visualisation du fonctionnement, les sorties d'IC1.A et B sont équipées d'une LED bleue, reliée à la masse via une résistance ; les deux LED s'illuminent alternativement.

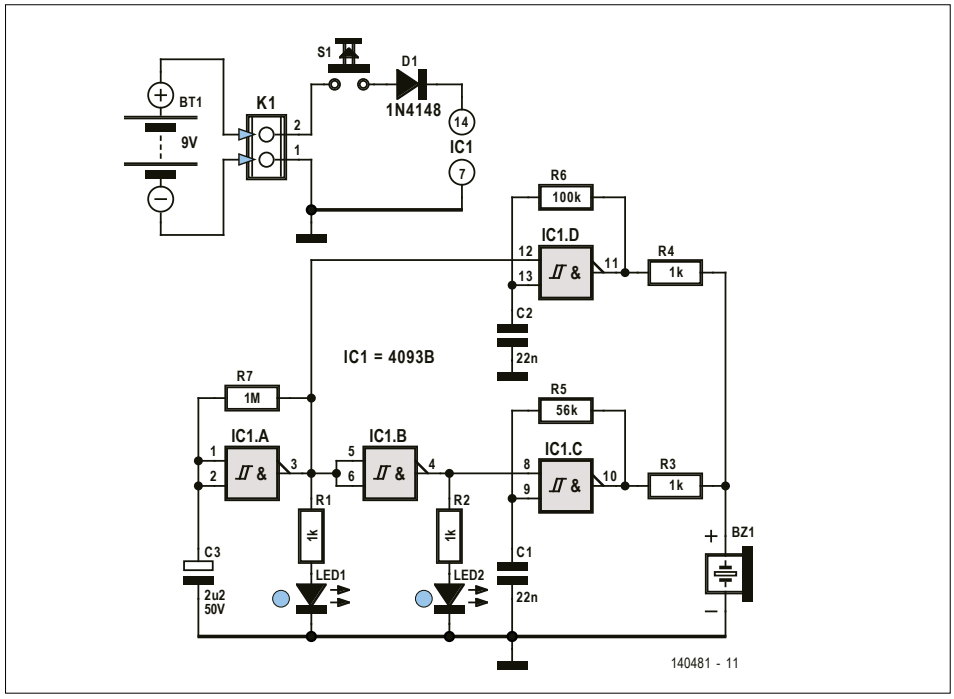


Figure 1. Schéma : trois oscillateurs et un inverseur.

Il nous fallait encore rendre les deux tons audibles, ce qui fut fait à l'aide d'un *buzzer* piezo. Celui-ci est relié à la sortie des deux oscillateurs via une résistance de 1 kΩ. Le niveau sonore est relativement modeste, ce qui est impératif si vous ne voulez pas « disjoncter » lorsque vos enfants s'amuseront avec le circuit. L'alimentation est des plus simples, une pile de 9 V suffit. Comme la consommation est faible (quelques milliampères), la durée de vie sera relativement longue. En outre, nous avons encore ajouté un bouton (S1) qu'il faut maintenir enfoncé pour déclencher la sirène. On peut bien sûr s'en passer ou le *shunter*. Enfin, la diode D1 sert de sécurité contre une inversion accidentelle de polarité de la pile : vous ne ferez pas tout sauter si vous vous trompez lors du raccordement de celle-ci.

Souder

Pour faciliter aux débutants la réalisation de ce projet au nombre de composants réduit, nous avons créé un circuit imprimé (**fig. 3**) que l'on peut aussi bien remplacer par un morceau de plaquette d'expérimentation. Nous n'allons pas nous attarder sur le montage, de peur d'entrer en discussion avec les lecteurs plus expérimentés, qui auraient sans doute préféré une électronique plus « intelligente ». Si vous savez souder, vous pouvez monter ce circuit.

Pour plus de sécurité, utilisez un support pour le circuit intégré et veillez bien au respect de la polarité de la diode, des LED et des condensateurs. N'hésitez pas à impliquer vos enfants dans la construction, vous pourrez leur donner trucs et astuces pour le soudage. Amusez-vous bien ! ⏮

(140481 – version française : Jean-Louis Mehren)

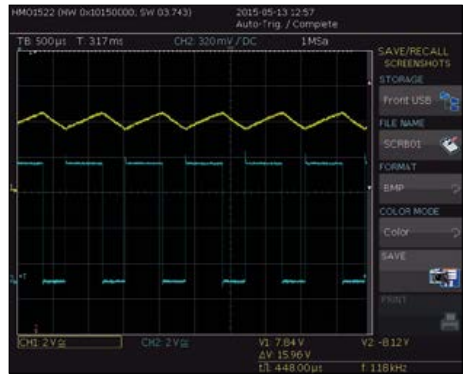


Figure 2. Pour mieux comprendre, cette copie d'écran montre les signaux recueillis sur les broches 9 (jaune) et 10 (bleu) du circuit intégré.

Liste des composants

Résistances :
(5%, 250 mW)
R1 à R4 = 1 kΩ
R5 = 56 kΩ
R6 = 100 kΩ
R7 = 1 MΩ

Condensateurs :
C1,C2 = 22 nF, 100 V, X7R , pas de 2,54 mm
C3 = 2,2 µF, 50 V, pas de 2 mm

Semi-conducteurs :
D1 = 1N4148
LED1, LED2 = LED bleue, 3 mm
IC1 = 4093B

Divers :
Support DIL à 14 broches pour le CI (facultatif)
K1 = bornier à 2 pôles, pas de 3,5 mm
BUZ1 = buzzer piezo passif, diam. de 12 mm
S1 = bouton-poussoir pour circuit imprimé, 6x6 mm

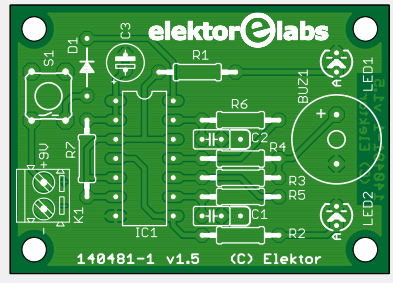


Figure 3. Le circuit imprimé peut être qualifié de *super luxe* ; le circuit est presque aussi vite monté sur une plaquette d'expérimentation.

ampli audio T-board

à haut-parleur intégré

Ton Giesberts (Elektor Labs)

Pour la mise au point de circuits analogiques sur une platine d'expérimentation, un petit amplificateur audio et un haut-parleur sont souvent utiles, afin de rendre les signaux audibles.
Nous avons donc réalisé une carte *T-board*, qui reprend sur un seul circuit imprimé un amplificateur de classe D, un convertisseur CC/CC, et un haut-parleur.

Caractéristiques techniques

• Courant de repos :	7,5 mA (4 V)
	2,3 mA (20 V)
• Sensibilité d'entrée (P1 max.) :	100 mV (1 kHz / 300 mW / 8 Ω)
	85 mV (1 kHz / 400 mW / 4 Ω)
• Bande passante :	min. 20 à 20 kHz (entre LS+ et LS-)
• Distorsion harmonique :	< 1% (pour 300 mW / 8 Ω, 400 mW / 4 Ω)
• Tension d'alimentation :	4 à 20 V (8 Ω)
	4,5 à 20 V (4 Ω)
• Consommation max.	sur 8 Ω : 125 mA (4 V), 30 mA (20 V)
	sur 4 Ω : 225 mA (4 V), 45 mA (20 V)

Jusqu'à présent nous n'avions proposé que des cartes *T-board* numériques : *T-board* 8, 14, 28 et *Wireless* (sans fil). Pourquoi pas une version analogique ? Beaucoup de circuits analogiques aussi sont mis au point et testés sur une platine d'expérimentation (*breadboard* en anglais), nous nous sommes donc demandé quel circuit serait utile aux expérimentations avec ce type de montage. On se retrouve tout naturellement dans le domaine de l'audio : pourquoi pas une carte *T-board* avec un petit ampli, et si possible un haut-parleur, qui permettrait d'entendre les signaux audio sans avoir à réaliser un circuit séparé et à y connecter un haut-parleur ? Il faudrait bien sûr une alimentation pour cet amplificateur – un adaptateur secteur ou des piles – mais en principe ce n'est pas un problème. Ainsi naquit l'idée de réaliser un circuit imprimé avec un amplificateur et un haut-parleur.

Nous pensions utiliser un circuit intégré analogique, tel le LM386, mais l'obligation de monter un condensateur électrochimique en sortie compromettait la taille de la carte *T-board*, que nous voulions petite. Pour éviter cet écueil et ne pas avoir recours à une alimentation symétrique, il ne reste guère que la configuration en pont, où le haut-parleur est branché entre les sorties de deux amplificateurs. La question suivante est le choix d'amplificateurs analogiques ou numériques. Ces derniers ont l'avantage d'un rendement plus élevé (pas besoin de radiateur !) et délivrent plus de puissance pour une tension d'alimentation basse, le choix est donc vite fait. Reste la question de la puissance : 150 ou 200 mW sont suffisants pour un petit haut-parleur, et permettent même d'alimenter un plus gros haut-parleur externe si nécessaire. Pour le circuit intégré, le choix s'est porté sur un SSM2305 d'*Analog Devices* (IC2, **fig. 1**). La tension d'alimentation de ce circuit intégré est comprise entre 2,5 et 5 V ; avec 2,5 V, il délivre encore une puissance d'un peu plus de 300 mW sur 8 Ω, sans distorsion appréciable. C'est plus que suffisant pour nos besoins. Le SSM2305 contient un amplificateur de classe D, avec entrées symétriques et sorties *push-pull*. D'après la feuille de caractéristiques, la structure du circuit permet de s'affranchir d'un filtre LC en sortie. Le nombre de composants externes est donc réduit au minimum.

Nous avons opté pour un haut-parleur sur le circuit imprimé. Le circuit sera donc un peu plus grand (la carte *T-board* est bien

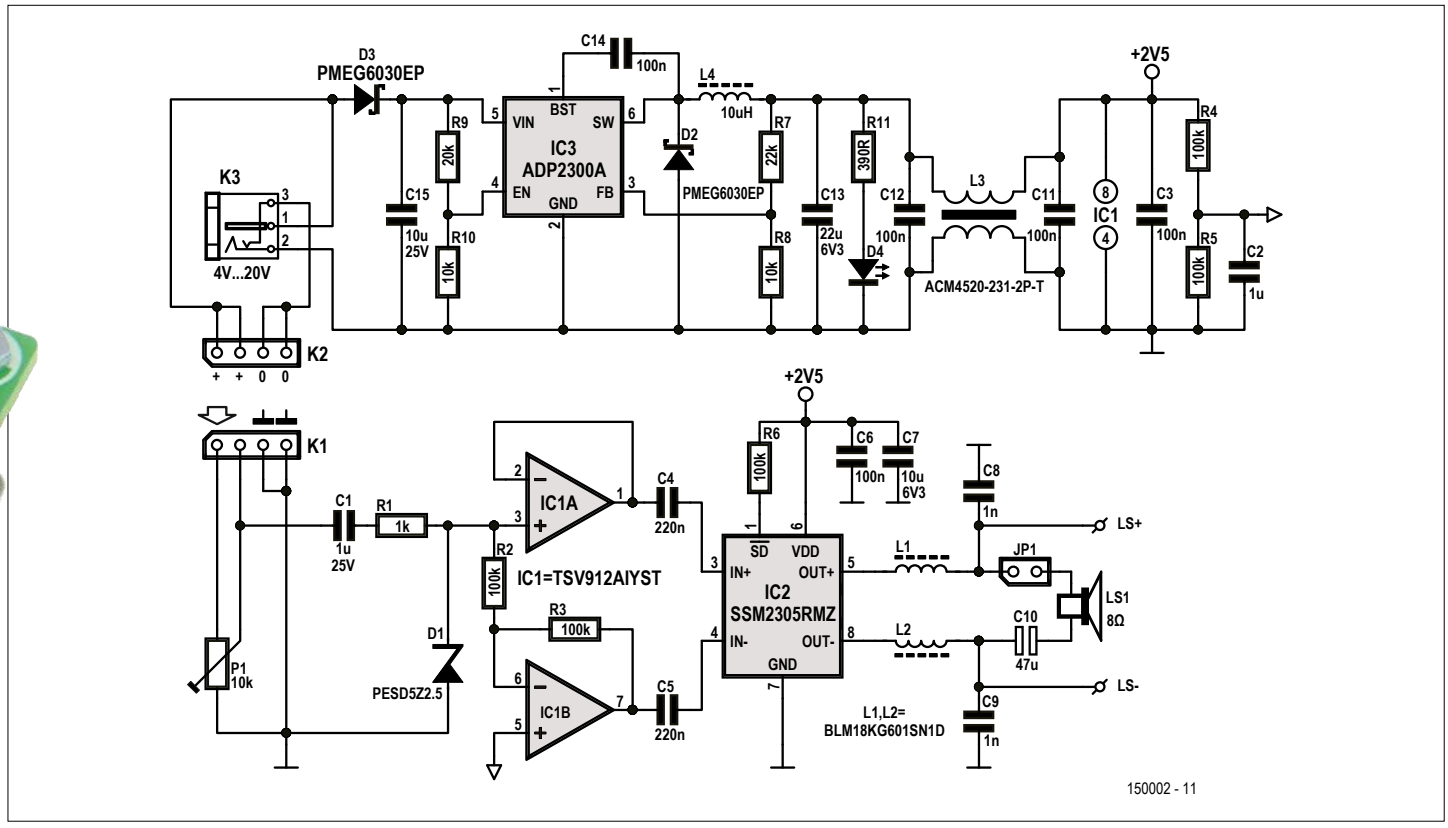


Figure 1. Le circuit est construit autour d'un amplificateur de classe D (circuit intégré) et d'un convertisseur CC/CC.

chargée...), mais son confort d'utilisation y gagne. Quelques picots à souder permettent de raccorder un éventuel haut-parleur externe.

L'amplificateur

Le signal provenant du circuit à l'étude sur la platine d'expérimentation entre par le connecteur K1. Si nous utilisons la broche 1 de K1, le volume peut être réglé avec P1 ; si nous utilisons la broche 2 de ce même connecteur, le signal est transmis à l'amplificateur sans atténuation. La sensibilité est relativement bonne : 100 mV pour une charge de 8 Ω. L'impédance d'entrée varie entre 9 et 10 kΩ, en fonction de la position de P1. Le signal est ensuite dirigé vers les entrées du double amplificateur opérationnel IC1 via C1 et R1. La diode de protection contre les décharges électrostatiques D1 protège ces entrées contre les tensions trop élevées.

Le SSM2305 possède des entrées symétriques ; il faut donc transformer le signal d'entrée asymétrique en un signal symétrique. C'est le rôle du double amplificateur opérationnel IC1 : IC1A est configuré en tampon et attaqué sur son entrée non-inverseuse IN+, tandis qu'IC1B est configuré en inverseur et attaqué par son entrée IN-. La bande passante de TSV912AIYST (IC1) est de 8 MHz, sa tension d'entrée peut monter jusqu'à la tension d'alimentation comprise elle-même entre 2,5 et 5,5 V. Les valeurs de C4 et C5 permettent de passer le 20 Hz, un haut-parleur externe pourra donc reproduire tout le spectre audio.

La broche de mise en veille du SSM2305 est inutilisée et reliée à la tension d'alimentation via R6.

De petits filtres LC (L1-C8 et L2-C9) sur les sorties de l'amplificateur, faits de perles de ferrites et de condensateurs de faible valeur, réduisent le rayonnement haute fréquence – surtout avec un haut-parleur externe. Ce ne sont pas des filtres passe-bas pour le signal audio, comme on en trouve en général sur les amplificateurs de classe D ; la valeur et les dimensions des inductances et condensateurs auraient été bien plus grandes.

Le haut-parleur qui équipe le circuit est très petit, on ne peut donc pas trop lui en demander, surtout pour la reproduction des fréquences les plus graves (la courbe de niveau d'un tel haut-parleur commence à chuter dès 500 Hz). En outre, C10 entre la sortie de l'amplificateur et le haut-parleur limite l'amplitude des fréquences graves et donc l'excursion du cône du haut-parleur. Le haut-parleur du circuit imprimé peut être mis hors service à l'aide du cavalier JP1.

L'alimentation

L'alimentation à découpage offre d'une part un haut rendement, d'autre part une plage plus étendue pour la tension d'entrée. Le but est de pouvoir utiliser la plupart des adaptateurs secteur, l'entrée doit donc pouvoir accepter au moins 12 V. Le choix s'est à nouveau porté sur un circuit intégré d'*Analog Devices*, le ADP2300A (IC3), un régulateur abaisseur. Il n'est disponible qu'en boîtier UJ-6 (6 broches, *thin small outline transistor package*, pour transistors à profil très bas et faible écartement des broches), mais il délivre un courant de sortie de 1,2 A. Le nombre de composants externes est réduit, et nous suivons les recommandations du constructeur. Grâce à une fréquence de découpage élevée (700 kHz), l'inductance L4 peut rester de faible valeur (10 µH) ; elle est dimensionnée pour une ondulation de 30% avec une tension d'entrée de 5 V et un courant de sortie maximal. Cette ondulation sera

plus grande pour des tensions d'entrée plus élevées. La tension de sortie maximale de l'amplificateur (sans écrêtage) est d'environ 3,6 V_{CC} sur 4 Ω, ce qui nous donne un courant crête de ±450 mA à fournir par l'alimentation.
L'entrée d'activation (*enable*) d'IC3 permet de fixer la tension de démarrage du régulateur. Avec les valeurs choisies pour R9 et R10, la conversion CC/CC commencera avec quelque 3,8 ou 3,9 V en entrée (en comptant la tension de polarisation inverse de la diode de protection D3). La tension d'entrée peut atteindre 20 V. L'atténuateur R7/R8 détermine la tension de sortie du régulateur, ici 2,5 V. La LED D4 sert d'indicateur de tension. La bobine d'arrêt (ou self de choc) en mode commun L4 atténue de manière effective les signaux parasites HF. Enfin, le diviseur R4/R5 offre à IC1B une tension de référence égale à la moitié de la tension d'alimentation.

Pour le raccordement d'un adaptateur secteur ou d'un support d'accus ou de piles (il en faut trois au minimum), il y a deux possibilités : via l'embase femelle d'alimentation K3 qui se trouve sur le dessus du circuit imprimé (broche du milieu = plus de l'alimentation), ou via les broches de K2 qui se connecteront à la platine d'expérimentation. Dans ce dernier cas, il faudra tenir compte de l'augmentation du courant qui transite par la platine d'expérimentation. La préférence devrait aller à une alimentation séparée. Lorsqu'un connecteur d'alimentation est inséré dans K3, un contact intégré à cette embase interrompt la liaison de masse de K2. Ceci évite un problème de boucle de masse lorsque les deux circuits (celui de la platine d'expérimentation et l'amplificateur) sont alimentés par le même adaptateur secteur (ce qui est fortement déconseillé, rappelons-le) : le raccord des masses court-circuiterait en fait une des deux bobines de L3.

La construction

Le circuit imprimé de la carte *T-board* est à double face (**fig. 2**). Les composants traversants se trouvent sur le dessus, sauf K1 et K2 installés sur le dessous

Liste des composants

Résistances :
(0,1 W 1%, CMS 0603, sauf mention contraire)
R1 = 1 kΩ
R2 à R6 = 100 kΩ
R7 = 22 kΩ
R8, R10 = 10 kΩ
R9 = 20 kΩ
R11 = 390 Ω
P1 = pot. ajust. 10 kΩ, 0,5 W 10% (*Vishay Sfernice* T73YU103KT20)

Condensateurs :
C1, C2 = 1 µF / 25 V, 10%, X7R, CMS 0805
C3, C6, C11, C12, C14 = 100 nF / 16 V, 10%, X7R, CMS 0603
C4,C5 = 220 nF / 10 V, 10%, X7R, CMS 0603
C7 = 10 µF / 6,3 V, 20%, X5R, CMS 0603
C8,C9 = 1 nF / 50 V, 5%, C0G / NP0, CMS 0603
C10 = 47 µF / 100 V, 20%, bipolaire, ø 13 mm, pas de 5 mm, I, 240 mA
C13 = 22 µF / 6,3 V, 10%, X5R, CMS 1206
C15 = 10 µF / 25 V, 10%, X5R, CMS 1206

Inductances :
L1, L2 = 1,3 A / 0,15 Ω, 600 Ω à 100 MHz, CMS 0603 (*Murata* BLM18KG601SN1D)
L3 = 3 A / 2x50 mΩ, 230 Ω à 100 MHz, bobine d'arrêt de mode commun, CMS (ACM4520-231-2P-T)
L4 = 10 µH / 2,1 A, blindée, CMS (*Laird* TYS5040100M-10)

Semi-conducteurs :
D1 = PESD5Z2.5, CMS SOD-523
D2, D3 = PMEG6030EP, CMS SOD-128
D4 = LED verte faible consommation, 3 mm, avec fils de connexion
IC1 = TSV912AIYST, CMS MSOP-8
IC2 = SSM2305RMZ-REEL7, CMS RM-8
IC3 = ADP2300AUJZ-R7, CMS UJ-6

Divers :
K1, K2 = embase à 4 broches fines, au pas de 2,54 mm (*Harwin* D01-9923246, *Farnell* 1022218)
K3 = fiche femelle d'alimentation 12 V / 3 A pour circuit imprimé, broche centrale ø 1,95 mm (*Lumberg* NEB 21 R)
PC1, PC2 = picot à souder rond, ø 1,3 mm
JP1 = embase à 2 broches avec cavalier, pas de 2,54 mm
LS1 = haut-parleur pour circuit imprimé, ø 31,9 mm, hauteur 15,1 mm (*RS-online* 7243119)
Autres possibilités pour LS1 : *Multicomp* MCABS-201-RC (*Farnell* 2361100) ou *Visaton* 2823, K 23 PC - 8 Ω (*Farnell* 2357167)
Circuit imprimé 150002-1 ou module complet monté 150002-9

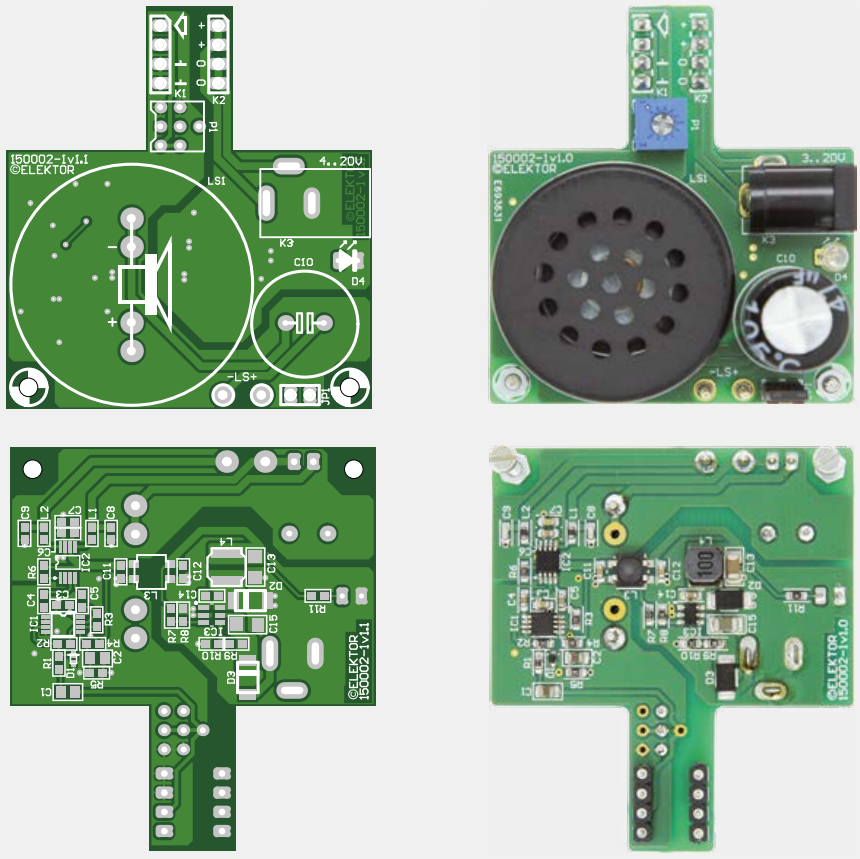


Figure 2. Le circuit imprimé à double face reçoit des éléments sur ses deux côtés : les composants avec fils sur le dessus, les composants montés en surface (CMS) et deux connecteurs sur le dessous.

(et qui se connecteront plus tard à la platine d'expérimentation). Les composants montés en surface (CMS) sont aussi sur le dessous. Les CMS choisis ont des dimensions telles qu'on peut les manipuler et souder à la main.
Le condensateur bipolaire entre la sortie de l'amplificateur et le haut-parleur peut sembler gros pour sa valeur, mais il a l'avantage d'être bon marché et peut supporter un courant alternatif de 240 mA (valeur qui doit être supérieure au courant maximal dans le haut-parleur, < 200 mA). Attention, ceci est valable pour un modèle 100 V ; n'en choisissez donc pas un avec une tension maximale plus faible.
Le circuit imprimé est conçu pour trois modèles de potentiomètre de chez *Vishay Sfernice* (divers trous sont prévus), mais les modèles d'autres fabricants devraient pouvoir être implantés sans problème.
Pour ceux que le soudage de CMS rebute, Elektor propose le module complet monté (avec haut-parleur), sous la référence 150002-91.

La **figure 3** montre clairement comment les CMS sont montés sur le dessous de la carte *T-board*. Il faut aussi fixer deux boulons M2 d'une longueur de 15 mm avec deux écrous dans les trous prévus à cet effet sur le circuit imprimé ; ils serviront à soutenir la carte lorsqu'elle sera connectée à la platine d'expérimentation. La **figure 4** montre la carte *T-board* en action, connectée à l'extrémité d'une plaque d'expérimentation, ce qui laisse beaucoup de place pour le circuit à étudier.

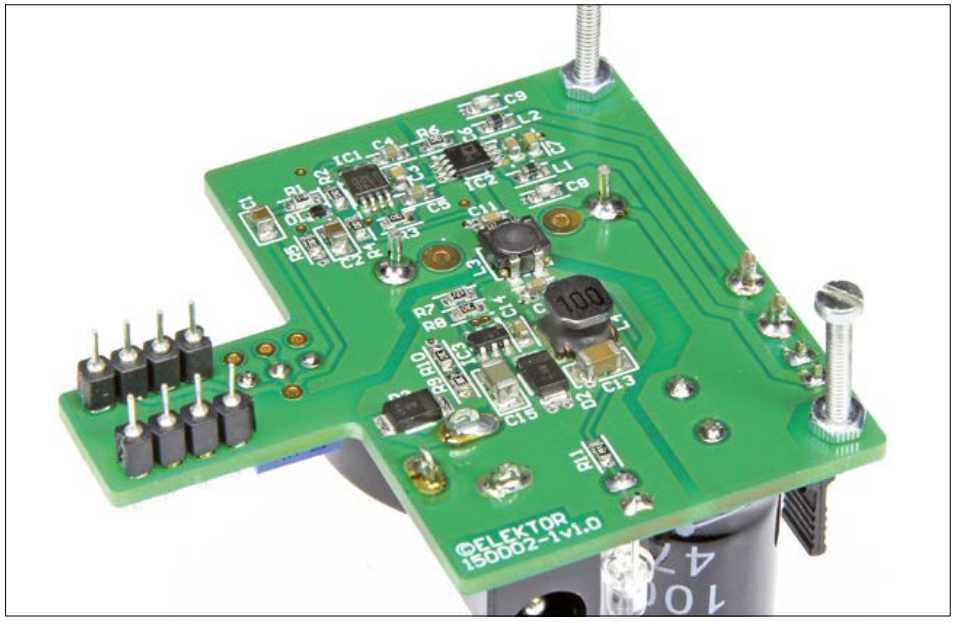


Figure 3. Sur cette photo, on voit bien les composants montés en surface (CMS). Deux boulons M2 d'une longueur de 15 mm soutiennent la carte *T-board* connectée à la platine d'expérimentation.

Le rendement

Pour ce circuit, nous recherchions le rendement le plus élevé possible, tant du côté de l'amplificateur que de la partie alimentation. Une série de mesures permet de déterminer le rendement total. Les données recueillies lors de ces mesures se trouvent sur le site du labo [1], et nous ne discuterons ici que des résultats les plus importants.

Le rendement est mesuré sur une charge purement résistive d'une part, sur une charge simulant un haut-parleur (inductance de 100 µH en série avec la charge résistive) d'autre part. Le rendement est un peu relevé par le comportement inductif du haut-parleur, comportement qui améliore aussi le rejet des résidus des fréquences de découpage (fréquence moyenne de 280 kHz pour le SSM2305). Avec une charge de 8 Ω et une tension d'alimentation de 20 V, le rendement le plus bas mesuré est de 52,5% sur charge résistive et 57,1% sur charge inductive. Comme prévu, le rendement est plus élevé avec une tension d'alimentation plus basse : avec 5 V au lieu de 20 V, le rendement passe à 58,4% sur charge résistive et 65,7% sur charge inductive. Le rendement baisse d'environ 10% avec une charge de 4 Ω. Les chutes de tension plus fortes sur les divers composants dans la chaîne du signal – résultant de courants plus intenses – ne sont pas négligeables avec une tension d'alimentation relativement basse. ⏮

(150002 – version française : Jean-Louis Mehren)

Lien

[1] www.elektor-labs.com/node/4436

Figure 4. Le circuit est placé à l'extrémité de la platine d'expérimentation, de manière à conserver un maximum d'espace libre sur celle-ci.

analy

module d'ent

Une fois encore, notre site de projets elektor-labs.com s'est révélé une bonne source d'inspiration. L'initiateur (OP) du projet, qui se fait appeler Midi-Rakete, a remis sur le métier le projet d'analyseur de canal MIDI MKII [1] qu'il avait publié il y a 20 ans dans le magazine Elektor (mai 1995, *Analyseur MIDI*, pp. 50). Il permet d'afficher sur 16 LED une communication sur les canaux MIDI (*Musical Instrument Digital Interface*) 0 à 15 ou 1 à 16, c'est comme on veut. Avec lui, pourtant, on ne voit pas quels messages MIDI circulent sur le câble, comme d'un contrôleur de clavier vers un synthétiseur.

J'ai découvert avec la production de musique électronique, qui ressemble fort à de la programmation, un passe-temps passionnant pour mes vieux jours. Sûrement pas pour monter sur scène et, même si je bricole rarement de nouveaux appareils, un petit outil qui déchiffre les messages MIDI et peut les montrer à l'écran, je trouvais cela très utile. Comme par hasard, mon collègue Clemens Valens avait déjà développé un petit module MIDI pour son synthétiseur J2B [2]. Il permet d'envoyer et de recevoir des signaux MIDI avec un microcontrôleur et une interface sérielle (TX/RX).

MIDI concrètement

Pas besoin d'intelligence sur une telle carte. Avec la synchronisation, les signaux MIDI ne sont rien d'autre que des octets que l'on empile sur une interface sérielle avec un débit binaire de 31 250 bauds [3]. Et de fait, ici, aucune définition de niveaux de tension haut ou bas pour représenter les bits de données, de départ et d'arrêt. On parle plutôt d'une boucle de courant à 5 mA établie entre la sortie MIDI d'un instrument et l'entrée MIDI d'un autre appareil : la circulation du courant représente un 0 logique et l'arrêt, un 1 logique. On comprend mieux avec la **figure 1**. Une entrée, comme une sortie MIDI, a deux contacts et c'est une paire de fils, comme pour le téléphone, qui relie les appareils l'un à l'autre (*daisy chain*). Sur l'un des contacts de sortie, il y a en permanence une résistance de 220 Ω vers le +5 V. Pour produire un 0 logique, l'autre contact de sortie est mis à la masse à travers 220 Ω. Il en résulte un petit courant qui circule par les contacts de l'entrée MIDI de l'autre appareil. Pour un 1 logique, on met le point TX au +5 V et plus aucun courant ne passe. Du coup, tout est magnifiquement compatible avec les niveaux TTL d'un UART.

seur MIDI

rée/sortie pour Arduino et Cie

Jens Nickel (Elektor)

Le célèbre duo Arduino et *shield* d'extension s'est composé une suite musicale avec un module d'analyse d'entrée et sortie MIDI qui, par l'entremise du connecteur ECC, s'adapte à d'autres cartes à µC. Notre micrologiciel didactique décode les messages MIDI et les fait apparaître. Les modules logiciels utilisés offrent un album de variations sur le thème.

Du côté entrée MIDI, on trouve un photocoupleur, composé d'une LED et d'un phototransistor. Avec du courant dans le câble MIDI, la sortie RX est attirée à la masse, tandis qu'à l'état de repos (1 logique), elle remonte au +5 V. Autant dire que nous pouvons brancher un microcontrôleur directement à la ligne RX.
Question connectique, sur les appareils MIDI, l'usage est de relier les contacts cités d'entrée et sortie aux broches 4 et 5 d'une prise DIN à 5 voies. Une entrée MIDI et une sortie MIDI sont pareilles, vues de l'extérieur, mais évidemment, en raison du caractère asymétrique du câblage, on utilise deux prises quand on veut installer une entrée et une sortie sur le même appareil.

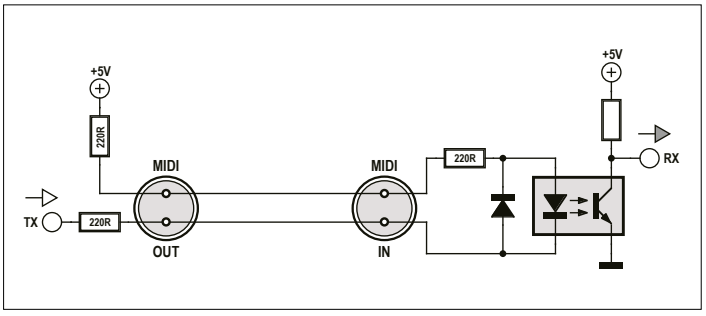


Figure 1. Les signaux MIDI se propagent par boucle de courant : s'il circule, c'est 0, pas de courant, c'est le 1 logique. On peut raccorder directement un microcontrôleur sous 5 V aux points RX et TX.

Le matériel

Les liaisons dont nous venons de parler facilitent la conception d'un circuit pour ajouter une entrée/sortie MIDI à une carte à microcontrôleur existante. Rien d'étonnant dès lors que je me sois fait fort de brancher un module MIDI à une carte à microcontrôleur par un *Embedded Communication Connector* (ECC). En réalité, il suffisait de doter le circuit imprimé de Clemens d'un ECC, puis de brancher le module MIDI par exemple à l'attelage confirmé d'une carte Arduino Uno et de son *shield* d'extension Elektor [4].

Le résultat est à la **figure 2**. À gauche, l'entrée MIDI avec le photocoupleur 6N137. Sa sortie est reliée à la broche 6 de l'ECC (K2). On y trouve le signal RX à envoyer au microcontrôleur qui y est raccordé. Par la broche 5 de l'ECC entre le signal TX du contrôleur pour le module MIDI, avec lequel on peut piloter directement la sortie MIDI sur K3, la prise DIN. Par la

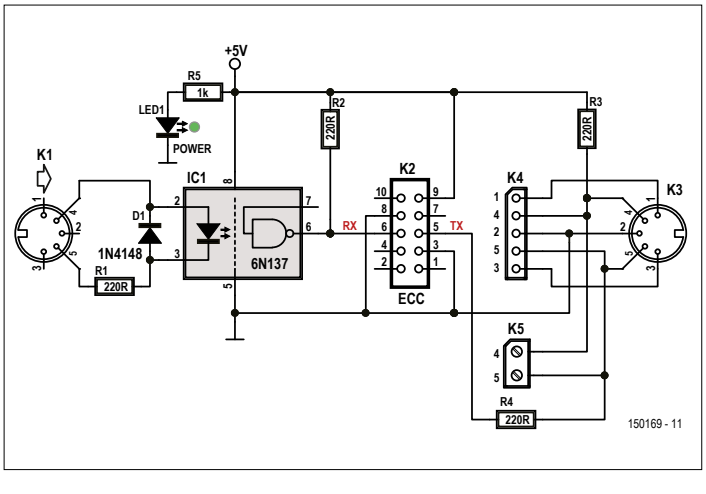


Figure 2. Le schéma du module MIDI In/Out.

Liste des composants

Résistances :
R1, R2, R3, R4 = 220 Ω
R5 = 1 kΩ

Semi-conducteurs :
D1 = 1N4148
LED1 = LED verte 3 mm
IC1 = 6N137, DIP8 (+ support)

Divers :
K1, K3 = prise DIN encartable horizontale (p.ex. Hirschmann MAB 5 SH)
K2 = embase femelle à col 2x5 voies, au pas de 2,54 mm
K4 = embase femelle à 1x5 voies, au pas de 2,54 mm
K5 = borne à 2 vis encartable, au pas de 5,08 mm
circuit imprimé réf. 150169-1 v1.0

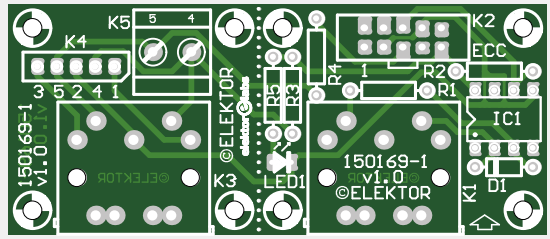


Figure 3. Le circuit imprimé est sécable en deux parties.

broche 9 de l'ECC, on alimente sous 5 V le circuit d'une carte à contrôleur et la LED baptisée Power s'allume si ça marche. Ton Giesberts du laboratoire Elektor a fabriqué un circuit imprimé (**fig. 3**) que l'on peut scier en deux en suivant les perforations [5]. La partie de droite peut alors servir d'entrée MIDI simple. Celle de gauche sera une sortie MIDI avec les signaux à prélever sur K5 ou une carte d'adaptation à prise DIN avec K4 comme entrée. Sur le prototype du laboratoire Elektor visible à la **figure 4**, K4 est encore constitué d'une embase à picots, mais une barrette femelle est préférable pour éviter un court-circuit accidentel.

Vous pouvez vous procurer ce circuit imprimé dans l'échoppe [6] et l'équiper de ses composants sans difficulté.

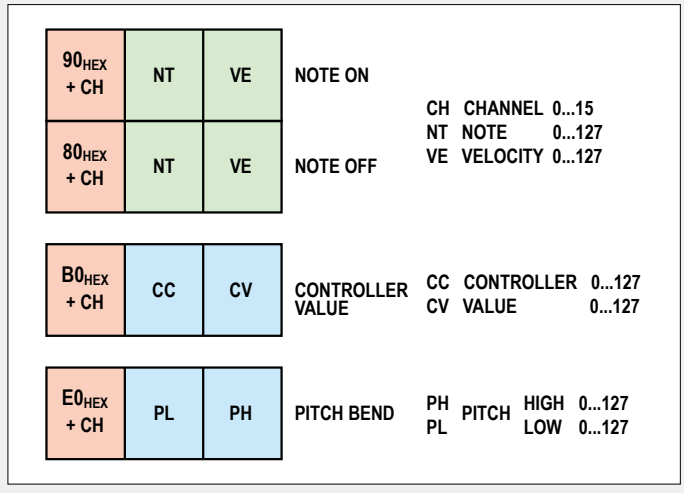
Le logiciel

Le module MIDI, une fois assemblé, est relié par câble plat à dix conducteurs au connecteur ECC du *schield* d'extension, lui-même branché sur un Arduino Uno. L'ATmega328P de l'Arduino Uno est alors en mesure de recevoir les octets MIDI par son UART, mais il lui faut aussi du logiciel [6] pour en faire un analyseur MIDI. Heureusement, le protocole MIDI n'est pas trop compliqué (voyez l'encadré). Les messages MIDI essentiels se composent presque toujours de trois octets dont le premier est aisément reconnaissable.
On peut résumer ainsi toute la technique du logiciel :

- initialisation du UART sur le débit binaire de 31 250 bauds ;

Le protocole MIDI

Contrairement à des préjugés répandus, le protocole MIDI n'a rien de compliqué. La plupart des messages MIDI qui circulent, par ex. d'un contrôleur de clavier à un synthétiseur matériel ou logiciel, se composent de trois octets. Le premier est la concaténation d'une commande, sur les quatre premiers bits, et d'un numéro de canal entre 0 et 15, sur les quatre derniers. Comme le premier bit de cet octet est toujours un 1 et que les deux octets suivants ne représentent que des valeurs entre 0 et 127, donc leur premier bit est toujours un 0, on y reconnaît vite un message dans le flux de données. D'ailleurs, le même mécanisme très simple est utilisé également sur l'ElektorBus.
Le logiciel pour ce projet [6] reconnaît les quatre principales commandes MIDI. Deux d'entre elles, 90_H et 80_H, marquent le début et la fin d'une note jouée. Le deuxième octet code chaque fois une hauteur de note en demi-tons (0 à 127, la suite des douze demi-tons depuis le DO de l'octave 0 jusqu'au SOL de l'octave 10 ; LA 440 Hz = 69). Le troisième octet (0 à 127) concerne l'enveloppe de volume ou l'ADSR (**a**ttaque, **d**éclin, maintien (***s**ustain*) et **r**elâchement) de la touche sur le clavier.
La commande $B0_H$ permet de donner à ce qu'on appelle un Contrôleur Continu (CC = 0 à 127) une nouvelle valeur (0 à 127). Les paramètres de mise en forme sonore sur un synthétiseur sont toujours dotés d'un numéro de contrôleur déterminé si l'on veut les modifier par MIDI en cours de route. On réserve par ex. CC = 1 à la célèbre molette dont pratiquement tout clavier de contrôleur est équipé. Davantage d'informations à ce sujet sous [8].
La commande $E0_H$ transmet la position de la molette de vibrato. Dans ce cas, la précision est plus grande, puisque la valeur peut varier dans une plage de 0 à 16 383 (14 bits). Les sept bits de poids fort sont transmis dans le 3e octet, les autres dans le 2e octet.



- introduction commandée par interruption des octets reçus dans une mémoire circulaire ;
- prélèvements cycliques dans la mémoire circulaire pour détecter un nouveau message à l'aide du premier octet typique, décodage de cet octet ainsi que des deux suivants ; agencement des éléments décodés dans une structure particulière ; appel d'une fonction convenue d'avance pour signaler qu'un nouveau message MIDI a été décodé ;
- affichage à l'écran des éléments du nouveau message reçu. Les précédents restent affichés ; il n'y a que quand un élément se transforme (par exemple la valeur d'un contrôleur MIDI) que la fenêtre respective est rafraîchie. Vous pouvez ainsi mieux suivre le cheminement d'un réglage.

Idéalement, des modules logiciels autonomes sont responsables de chacune des tâches, ce qui facilite la mise à niveau, la réutilisation et la portabilité des logiciels. La dépendance, même temporaire, entre les modules doit rester minimale, nous allons y revenir. J'ai bien sûr utilisé pour notre logiciel de mise en train la bibliothèque EFL (*Embedded Firmware Library*) [7], on y trouve déjà la mémoire circulaire toute faite, avec une bibliothèque d'affichage que l'on peut inclure de nouveau dans les fichiers de la carte et du contrôleur pour le *shield* d'extension, l'Arduino Uno et l'ATmega328P. Il ne me restait plus qu'à me préoccuper d'une petite bibliothèque MIDI qui traite du 3e point. À vrai dire, je n'étais pas au bout de mes peines, vous allez voir.

Décodage MIDI

Les messages MIDI sont décodés dans une petite bibliothèque du nom de MidiEFL.h/.c. Elle est déjà, tout comme d'autres bibliothèques de protocoles EFL proposées (par ex. pour l'ElektorBus), organisée de manière à travailler de concert avec différents canaux physiques de communication. On peut donc décoder ainsi des octets MIDI reçus selon le protocole TCP/IP. Tout ce qui compte maintenant, c'est de les faire atterrir dans la mémoire circulaire ; l'initialisation avec la commande suivante permet à la bibliothèque de savoir où trouver ces octets

```
MIDI_LibrarySetup(UARTInterface_Send, 0,
   UARTInterface_GetRingbuffer(0), MIDIIn_Process);
```

Les paramètres déterminent que c'est l'interface UART #0 qui doit être utilisée (la seule disponible sur l'Arduino Uno), pour envoyer et recevoir des octets. On le communique au tampon circulaire et comme premier paramètre, une fonction est appelée lors de l'envoi de messages MIDI. Alors la bibliothèque MIDI joint déjà une fonction pour pouvoir coder spécialement des messages MIDI et ensuite les envoyer.

Le dernier paramètre est un pointeur sur la fonction *Callback* qu'il faut implémenter dans le programme principal. Elle sera ensuite appelée par la bibliothèque MIDI quand un nouveau message aura été reçu et décodé.

Dans la boucle principale du programme, il faut régulièrement appeler la fonction `MidiProtocol_Engine()`. Elle achève, à l'intérieur de la bibliothèque, les travaux indiqués dans le point (3).

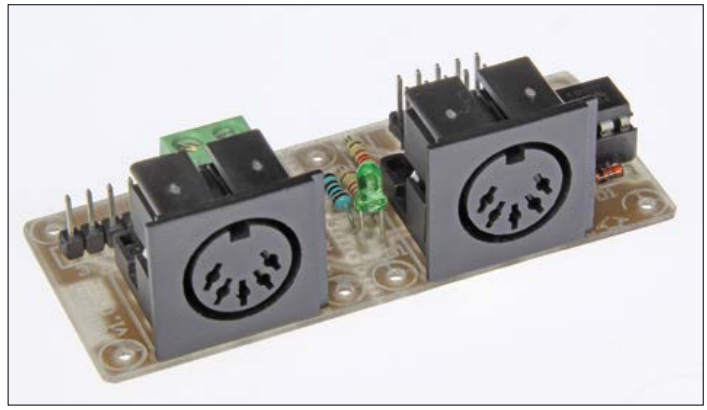


Figure 4. Le prototype du laboratoire Elektor avec ses deux prises DIN habituelles en MIDI.

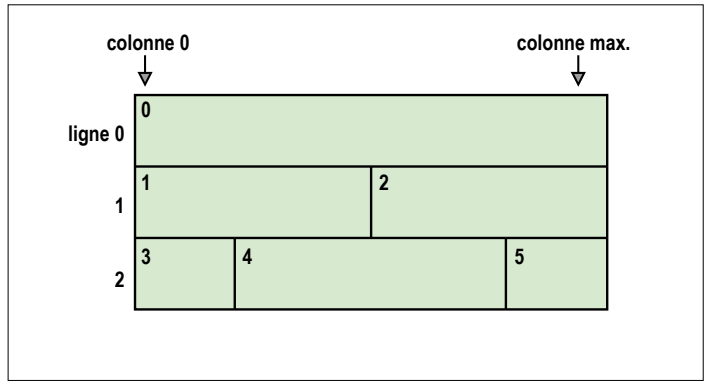


Figure 5. La bibliothèque Display d'EFL a été étendue pour permettre de configurer des champs alphanumériques. Ici, un exemple d'affichage de six (0 à 5) champs sur trois lignes.

On arrive à MIDI

Dès qu'un message est décodé, il est transféré dans la structure `ReceivedMidiData` de type `MidiData`. La fonction `Midi_GetReceivedMidiData()` affecte un pointeur vers cette structure au programme principal qui a alors accès aux éléments décodés.
Dans la fonction *Callback* déjà citée, on a aisément accès aux éléments du dernier message MIDI reçu pour les présenter à l'écran. Exemple de code possible :

```
MidiData* ReceivedMidiData =
   MIDI_GetReceivedMidiData();
Display_WriteNumber(0, 0, ReceivedMidiData->Channel);
```

On peut donc montrer dans la première ligne de l'afficheur le numéro de canal.
Jusqu'à présent, la bibliothèque d'affichage EFL ne disposait que d'instructions pour écrire du texte et des chiffres dans une ligne déterminée et en commençant toujours au début. Mais sur le LCD du *shield*, nous ne disposons que deux lignes sur lesquelles il faudrait inscrire quatre éléments. J'ai donc ajouté à la bibliothèque EFL un affichage qui présente, au lieu de deux lignes, jusqu'à huit champs alphanumériques. Les champs y sont repérés par des numéros entre 0 et 7 (**fig. 5**) ; j'appelle

ces numéros des positions. Une position peut également désigner une LED précise dans un bloc ou un bouton-poussoir dans un clavier.

Les instructions pour commuter une LED (ON = 1 ou OFF = 0) dans un bloc de LED et l'inscription de textes ON ou OFF dans un champ déterminé de l'afficheur ont maintenant l'air de ceci :

```
SwitchLED              (LEDBlockNummer, Position,  ON);
Display_WritePosition (DisplayNummer,  Position,
   "ON");
```

On peut fixer les coordonnées et la longueur du champ en chiffres dans le programme principal avec la fonction `Display_SetPosition(uint8 DisplayPosition, uint8 row, uint8 column, uint8 columnmax)`. Pour simplifier, lors de l'initialisation de la bibliothèque d'affichage, on crée deux champs égaux par ligne complète. Dès lors, on dispose de quatre champs de même longueur pour y inscrire les quatre éléments reçus du message, comme à la **figure 6**.
Pour des raisons de disponibilité de mémoire, il n'est pas encore possible de panacher la configuration de l'affichage ; un champ ne peut pas non plus déborder sur l'autre ligne. Mais si nécessaire, libre à vous de modifier ces fonctions.

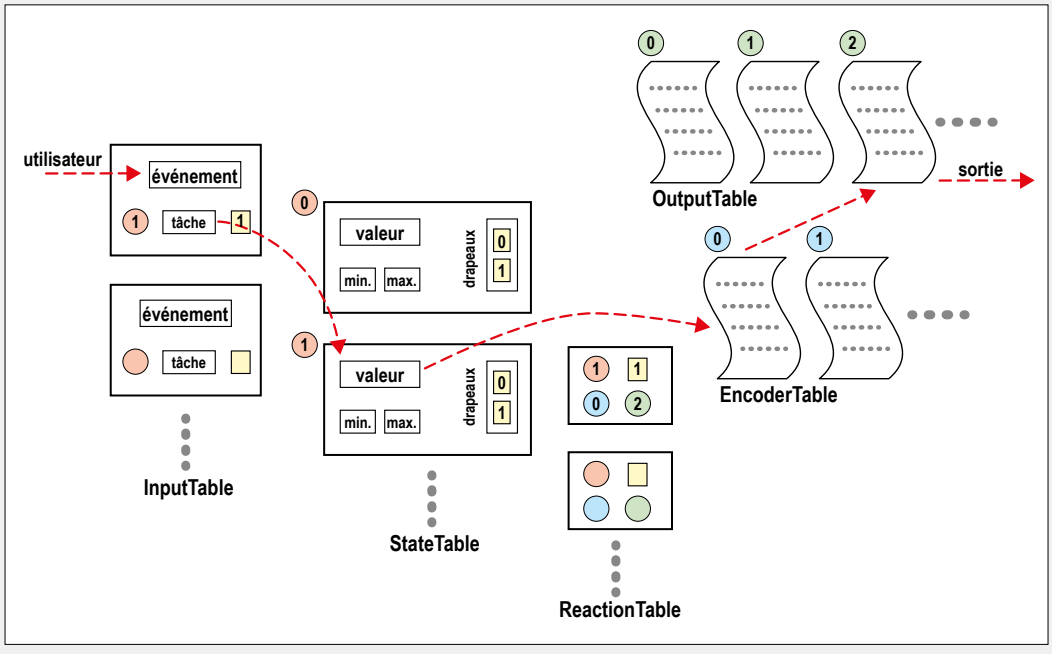
Souplesse d'édition

En faisant effectuer ainsi l'écriture sur l'afficheur directement par la fonction, nous avons en réalité instauré un couplage étroit entre les modules et, à terme, cela peut se révéler indésirable. Il est plus raisonnable d'éviter d'actualiser l'affichage tout de suite. D'autres tâches pourraient, à un stade ultérieur du logiciel, avoir la priorité. Pensons par ex. à l'écriture d'octets d'entrée MIDI auxquels on adjoint un horodatage. On pourrait aussi, un beau jour, vouloir y ajouter un mini synthétiseur qui permettrait d'entendre la note reçue.

La solution consiste à placer une entrée dans une table spéciale (*StateTable*) pour chacun des différents éléments MIDI

La bibliothèque EFL InOut

Pour montrer les avantages de la bibliothèque, un petit exemple vaut mieux qu'un long discours. Imaginons un régulateur pour appareils alimentés sur secteur pour lequel il y a quatre variables essentielles, les valeurs d'état *U_Soll* (consigne), *U_Mess* (mesure), *I_Mess* (mesure) et *I_Max*. Des entrées, par boutons-poussoirs, codeur rotatif ou potentiomètres, modifient *U_Soll* et *I_Max* et les valeurs *U_Soll*, *U_Mess*, etc. doivent aussi être reproduites en sortie, par exemple sur un écran. Notre logiciel doit pouvoir tourner sur différentes plateformes avec le moins possible de modifications. Sur une carte, on règle *U_Soll* avec un potentiomètre, sur une autre avec deux boutons (*Up*, *Down*). La portabilité ne sera pas possible sans aucune modification, mais pour avoir moins de travail, nous allons isoler toutes les parties du programme qui dépendent du matériel dans la fonction `ApplicationSetup`.



Un autre objectif est le découplage (temporel et logiciel) des entrées, des changements des valeurs d'état et des sorties. Si une valeur de mesure change 1 000 fois par seconde, cela n'aurait pas de sens d'actualiser chaque fois l'écran.
Notre troisième vœu serait d'être débarrassés du pénible travail de programmation.

Pour les entrées de l'utilisateur qui modifient des valeurs numériques, il faut toujours vérifier que les limites haute et basse ne sont pas franchies. Le nouveau Framework nous soulagera du fastidieux codage de toutes les comparaisons par *if*.

La nouvelle bibliothèque EFL *Common InOutEFL.h/.c* remplit les nombreuses tables nécessaires au début du programme. Les fonctions `State_Add(...)`, `Output_Add(...)` etc. créent une nouvelle entrée dans chaque table et renvoient les indices de ces entrées que l'on pourra utiliser comme paramètres pour des entrées dans d'autres tables. On fait déjà la même chose pour se passer du câblage des cartes. Maintenant, on fait l'impasse sur les liaisons entre entrées, modifications des valeurs numériques et sorties.

que l'on veut afficher. À côté de la valeur actuelle de l'élément, la table contient aussi un drapeau qui indique si la valeur a changé depuis la dernière actualisation de l'affichage. Dans la fonction appelée après décodage d'un message MIDI, on écrit la nouvelle valeur de l'élément MIDI dans la *StateTable*. Si la valeur ne correspond plus à celle précédemment stockée, on lève le drapeau STATE_UPDATED.

Il faut ensuite veiller cycliquement à actualiser l'affichage sur les valeurs modifiées, mais quand et à quel rythme, nous pouvons en décider indépendamment de la communication MIDI. L'actualisation de l'affichage en réaction au changement de la valeur a lieu dans le programme principal sur appel de la fonction `Reaction_Process()` qui est implémentée dans le nouveau module EFL *InOutEFL.c*. Dans ce but, la fonction scrute dans quelle entrée de la table le drapeau est levé. Ensuite, on appelle la fonction d'édition adéquate. Toutes ces fonctions sont également mises en dépôt au préalable dans une autre table (*OutputTable*). Une troisième table (*ReactionTable*) relie

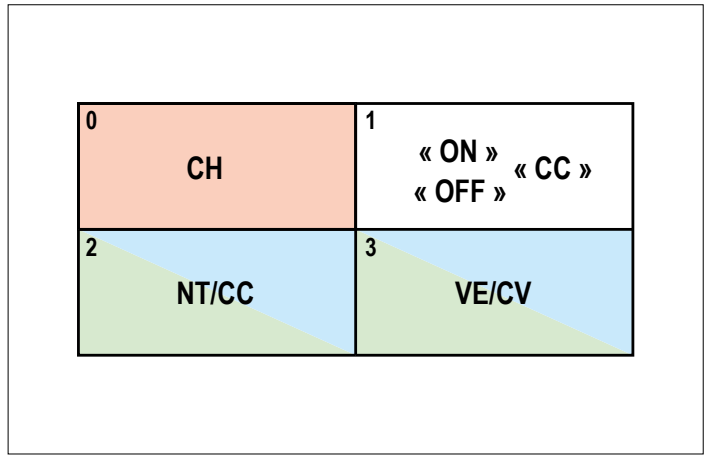


Figure 6. Les différents éléments d'un message MIDI représentés dans quatre champs de l'afficheur. CH = canal, NT = hauteur de note en texte, VE = enveloppe de volume, CC = numéro du contrôleur, CV = valeur du contrôleur.

La table centrale est la **StateTable** : chaque entrée contient la valeur actuelle (int16) de la variable d'application, les valeurs minimale et maximale possibles et aussi jusqu'à huit drapeaux. L'**InputTable** relie les événements d'entrée (par ex. l'exécution d'une tâche déterminée au moyen du numéro de bloc du bouton et de sa position) à un indice de la *StateTable* ; à cela s'ajoute la longueur de l'étape, puis la valeur de la variable lorsque l'événement se produit, ainsi que le drapeau qu'il faut lever dans ce cas.

On peut, dans l'*InputTable*, par exemple décider que lors d'une action sur un certain bouton, *U_Soll* (la tension voulue en millivolts) monte de 100. Dès qu'on augmente une valeur s'ensuit une vérification pour savoir si *U_Soll* est toujours dans les limites permises. S'il y a dépassement, la valeur est ramenée soit à son maximum, soit à son minimum, selon la configuration établie, ce qui permet un réglage cyclique de la valeur. Quand la valeur a été modifiée, le drapeau correspondant est levé, par exemple STATE_UPDATED.
L'**OutputTable** est la contrepartie de l'*InputTable* où sont enregistrées les fonctions d'édition (présentes dans la bibliothèque Display, par ex.) ensemble avec le numéro de bloc et la position de l'élément d'édition. On peut importer ici des fonctions entières dont la signature est la suivante :

```
Funktionsname(uint8 Blocknummer, uint8 Position, int16
   NumerischerWert)
```

ou

```
Funktionsname(uint8 Blocknummer, uint8 Position, char*
   TextString)
```

L'**EncoderTable** stocke les fonctions qui traduisent les valeurs numériques en chaînes de texte. On peut les implémenter aussi dans différents modules EFL, mais elles conservent la même signature :

```
char* Funktionsname(int16 NumerischerWert)
```

Une entrée dans la **ReactionTable** relie un indice de la *StateTable* (donc une variable d'application), un drapeau, une fonction d'édition et éventuellement un codeur (comme indice de la table en question). Pour activer les sorties, il faut appeler régulièrement la fonction `Reaction_Process()`. Elle parcourt la *ReactionTable* et vérifie si le drapeau convenu qui doit déclencher la réaction est levé sur la variable d'application concernée dans la *StateTable*. Si oui, on appelle la fonction de sortie, soit pour lui donner la valeur actuelle de la variable d'application, soit le résultat de la fonction de codage. On abaisse alors le drapeau dans la *StateTable*.

Pour rester dans l'exemple, nous pourrions définir une entrée dans l'*OutputTable* pour l'écriture d'un texte à la position 0 de notre afficheur #0. En outre, nous allons même implémenter une fonction de codage qui transforme la valeur en millivolts en texte au format x,yyy V. Nous écrivons cette fonction comme nouvelle entrée dans l'*EncoderTable*.

Si nous avions poussé sur le bouton pour remonter *U_Soll* de 100 mV, lors de l'appel suivant de `Reaction_Process()`, d'abord la fonction de codage aurait été appelée, puis le texte aurait été présenté sur l'afficheur. Passer au point décimal anglo-saxon est facile à faire : il suffit de rédiger une autre fonction de codage légèrement modifiée, de l'inscrire aussi dans l'*EncoderTable* et modifier l'indice correspondant. On peut le faire pendant que le programme tourne, pour permettre par exemple à l'utilisateur de changer de langue.
Nous vous en dirons plus sur la bibliothèque InOutEFL dans la prochaine édition.

les deux autres ; toute variation d'une valeur se voit donc toujours affecter une fonction d'édition. En supplément, on peut encore introduire un index sur une entrée dans une quatrième table (*EncoderTable*). Tout cela, on peut le clarifier au moyen d'un exemple : le traitement d'une valeur de note reçue par le canal MIDI.

Au lancement du programme, nous plaçons l'entrée suivante dans la table :

```
S_MidiIn_Note   = State_Add(0, 0, 127,
   STATE_MINMAXMODE_OVERFLOW);
O_Write_Pos2    = Output_Add(Display_WritePosition,
   0, 2);
E_Midi_Note     = Encoder_Add(Midi_NoteEncode);

Reaction_AddOutput(S_MidiIn_Note, STATE_UPDATED,
   O_Write_Pos2, E_Midi_Note);
```

Après réception d'une nouvelle valeur de note (valeurs permises entre 0 et 127), il faut l'inscrire dans *StateTable* et lever le drapeau STATE_UPDATED ; pour ce faire, nous implémentons dans la fonction *Callback* :

```
MidiData* ReceivedMidiData =
   Midi_GetReceivedMidiData();
State_Update(S_MidiIn_Note, ReceivedMidiData->Note);
```

Dans la boucle principale, on appelle régulièrement `Reaction_Process()`. La fonction surveille la levée du drapeau. Alors, on appelle d'abord la fonction de codage mise en dépôt `Midi_NoteEncode` (implémentée dans la bibliothèque MidiEFL), avec la nouvelle valeur de note (entre 0 et 127) comme paramètre. Le résultat est une chaîne de caractères telle que *C#4*. Cette chaîne devient alors paramètre de la fonction d'édition convenue `Display_WritePosition(...)`, laquelle écrit le texte à la 2e position de l'afficheur #0.
L'adaptation de notre logiciel se fait par de simples modifications d'entrées dans la table, même de façon dynamique pendant l'exécution du programme. Pour définir une autre langue de l'utilisateur, il n'y a qu'à changer les entrées dans le codeur.

L'indépendance vis-à-vis du matériel

Par ailleurs, nous avons remonté d'un cran l'indépendance par rapport au matériel et la modularité d'EFL. Il nous suffit, au début du programme, dans la fonction `ApplicationSetup`, d'initialiser la table correspondante et tout se déroule alors automatiquement et découplé du reste. Nous pouvons ainsi porter le micrologiciel sur une autre carte, sur laquelle l'afficheur ne porte pas le numéro #0, mais s'appelle Display #1 et y afficher notre élément MIDI dans un autre numéro du champ. Peut-être voudriez-vous ne plus représenter l'élément MIDI décodé sur un afficheur, mais le transmettre par un (autre) UART, ou bien visualiser la vélocité de l'attaque par la luminosité d'une LED ? Tout cela ne nécessite que de légères adaptations dans le code d'initialisation, le reste du programme reste inchangé. Dans un prochain article, nous élargirons encore le concept aux entrées de l'utilisateur. L'encadré *La bibliothèque EFL InOut* expose déjà de quoi il s'agit.
Restez à l'écoute ! ⏮

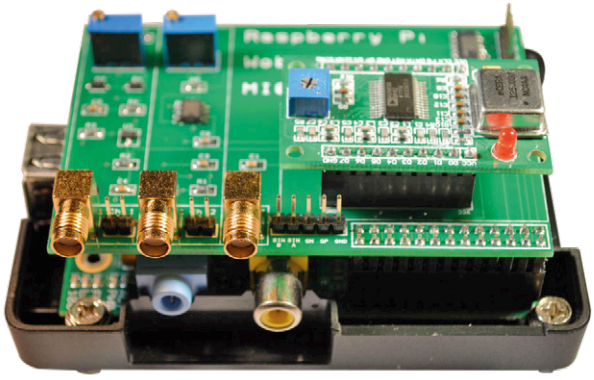
(150169 – version française : Robert Grignard)

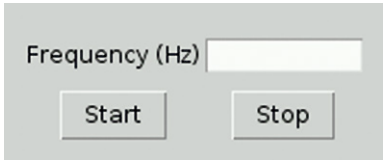
Liens

[1] www.elektor-labs.com/project/MIDI-channel-analyzer-mkii-140065-i.13380.html
[2] www.elektormagazine.fr/140182
[3] http://fr.wikipedia.org/wiki/Musical_Instrument_Digital_Interface
[4] www.elektormagazine.fr/140009
[5] www.elektor-labs.com/project/150169-MIDI-analyzer-light.14481.html
[6] www.elektormagazine.fr/150169
[7] www.elektormagazine.fr/120668
[8] www.midi.org/techspecs/midimessages.php

vobulateur à Raspberry Pi

avec module DDS et amplificateur logarithmique





Tom Herbison, MI0IOU (Royaume-Uni) Twitter : @TomHerbison

Le Raspberry Pi a beau être petit, la richesse de sa connectique permet de réaliser toutes sortes de choses plus ou moins baroques, par exemple le vobulateur présenté ici. Initié sur elektor-labs.com, ce projet de lecteur a été développé en suivant les trois règles d'action d'Elektor : Découvrir, Créer, Partager.

Un vobulateur s'utilise conjointement avec un oscilloscope pour mesurer la réponse en fréquence d'un circuit (RF) (**fig. 1**). Un générateur de fonctions de type « dents de scie » ou « rampe » relié à un oscillateur commandé en tension (VCO) produit en sortie un balayage de fréquences couvrant une bande déterminée. L'attaque des déviations X et Y de l'oscilloscope par les signaux du circuit à tester et du vobulateur permet de visualiser la réponse du circuit à tester — habituellement un filtre ou un amplificateur. On utilise le vobulateur pour aligner les étages de fréquence intermédiaire des récepteurs superhétérodynes, mais également pour mesurer les réponses en fréquence de filtres RF ou d'autres circuits.

Ce qu'il y a de formidable avec un vobulateur, c'est qu'il fournit immédiatement la courbe de réponse d'un amplificateur ou d'un filtre. Autrement dit, il permet de visualiser « en direct », pendant l'ajustement d'un circuit, des caractéristiques comme les points où l'atténuation vaut 3 dB, la raideur de la courbe, et l'ondulation intrabande. « Vobuler » des coupe-bandes et des passe-bandes à étages RF multiples avec retour visuel instantané de la réponse est amusant. C'est comme tracer une courbe par approximations successives, et vous pouvez jouer durant des heures à tenter de retrouver la courbe d'un manuel.

Fonctionnement

Les vobulateurs étaient autrefois des instruments coûteux, complexes, et entièrement analogiques (avec même des tubes à vide). Aujourd'hui leur fonction peut être reproduite par un mini-ordinateur comme le RPi. Le matériel est plus simple et, luxe suprême, les programmes peuvent être modifiés et optimisés.
Mon instrument implante les fonctions d'un vobulateur traditionnel au moyen d'un Raspberry Pi, d'un module DDS (synthèse numérique directe) et d'un module CAN (convertisseur analogique-numérique). Le programme qui pilote l'interface d'entrées/sorties à usage général (GPIO) du Pi envoie des instructions au DDS pour qu'il produise un balayage de fréquences, et au CAN pour qu'il mesure la réponse du circuit à tester. L'interface graphique utilisateur permet de choisir les paramètres du balayage de fréquences et affiche aussi les résultats.

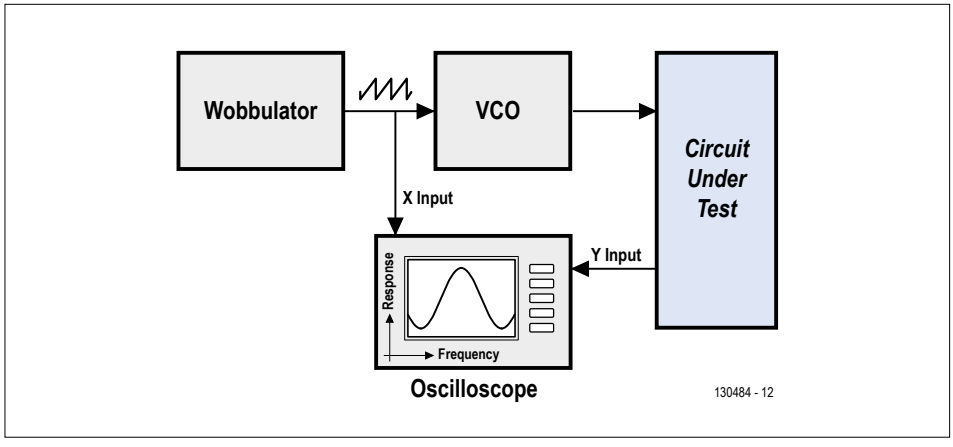


Figure 1. Diagramme fonctionnel simplifié d'un vobulateur, ou générateur de balayage.

Circuit

Le schéma de la **figure 2** montre la dernière version des extensions matérielles du RPi qui composent le vobulateur.
Au départ, j'avais juste utilisé un amplificateur/tampon d'entrée suivi d'un redresseur pour afficher la réponse linéaire du circuit à tester. Ensuite le projet a évolué et j'ai ajouté un amplificateur logarithmique pour avoir une lecture en dBm sur l'axe Y. J'ai tout de même conservé l'amplificateur LIN, entouré ici du FET U1 et des diodes D1/D2 chargées de la rectification RF.

L'amplificateur LOG repose essentiellement sur l'AD8307ARZ, un circuit intégré qui connut un succès immédiat auprès des radioamateurs lorsqu'il fut lancé il y a quelques années.

La sélection entre les signaux de sortie LOG et LIN est traitée par le MCP3424 (IC2), un convertisseur A/N sigma-delta à résolution max. de 18 bits avec entrée différentielle à 4 canaux, utilisé ici en esclave I²C. Ses broches SDA et SCL scrutent les instructions I²C ainsi que les données en provenance du port GPIO du Pi, ici le maître I²C. La ligne SDA transmet aussi au Pi sous forme numérique le signal « Y » mesuré (LOG ou LIN). Le Pi effectue les calculs idoines puis affiche le résultat sur un écran HDMI, comme le faisait autrefois votre oscilloscope à tube cathodique. Les lignes CH3 et CH4 du 3424 ne sont pas exploitées ici, mais tout de même regroupées sur un connecteur pour vos extensions éventuelles.

J'ai assemblé les premières versions de mon vobulateur sur une carte d'extension d'Adafruit qui s'enfiche directement sur le Pi. La carte loge la plupart des composants du vobulateur ainsi que le module DDS et son connecteur à 20 broches. Ce DDS repose sur la puce AD9850, un synthétiseur de fréquence disponible sur eBay et que je distribue aussi. Le Pi y envoie les instructions qu'il faut pour produire un signal balayant la gamme de fréquences paramétrée. Disponible sur le connecteur RF OUT, ce signal alimente le circuit à tester, dont la propre sortie va à l'entrée LOG ou LIN, selon l'application.

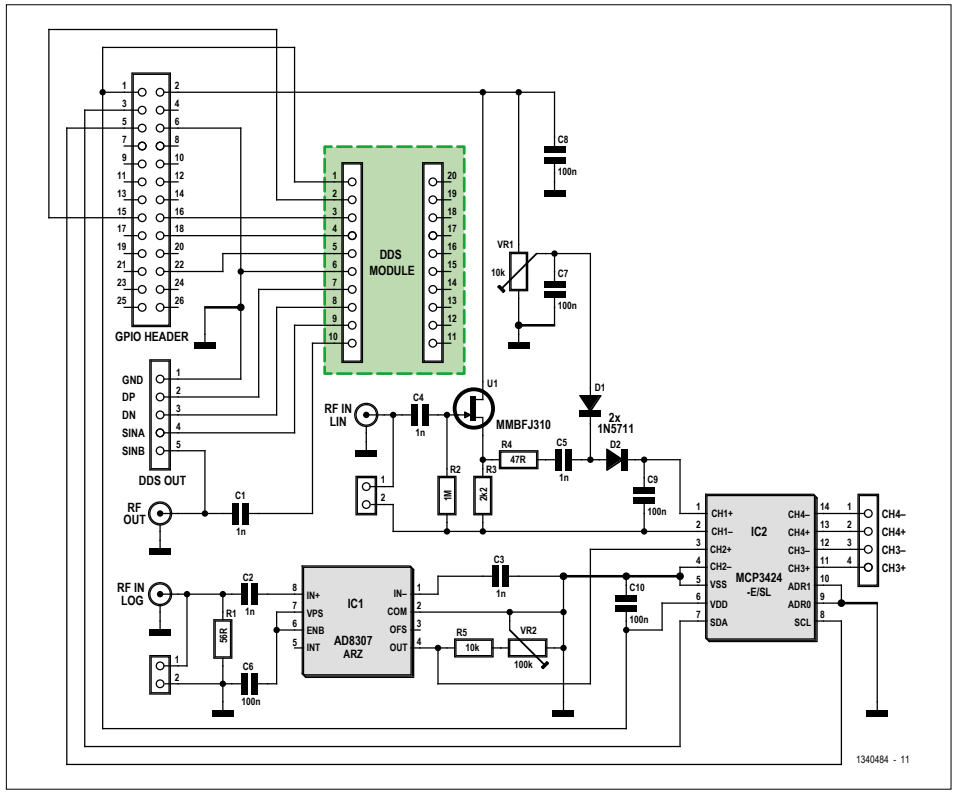


Figure 2. Schéma du vobulateur commandé par Raspberry Pi. Le DDS est un module du commerce.

Programme

Le projet, démarré en 2013, s'est enrichi depuis de nombreuses nouvelles fonctions, dont les échelles logarithmiques X et Y, et une option de balayage continue. J'ai écrit le programme de commande en Python. La dernière version du code source est disponible sur GitHub [2] et sur le groupe Yahoo *Raspberry Pi Wobbulator* [3]. J'ai par ailleurs écrit un guide d'installation détaillé à l'attention de ceux qui débutent avec Linux et le Raspberry Pi. Vous le trouverez dans le billet du mois d'avril 2014 de mon blog [4].

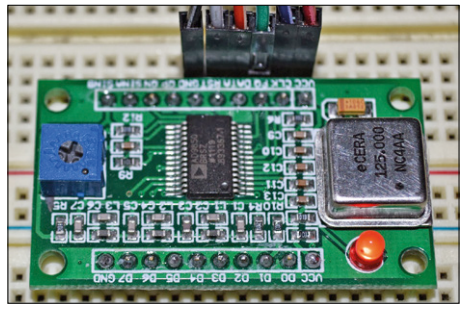


Figure 3. La carte du DDS à AD9850 est vendue sur eBay par des vendeurs quasi résidents.

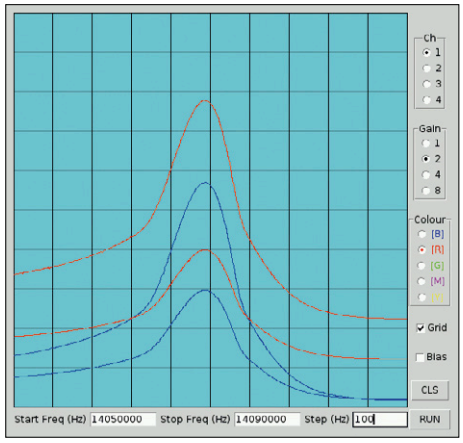


Figure 4a. Effet de l'option « *Bias* » sur un filtre pour la bande de 20 m (14 MHz).

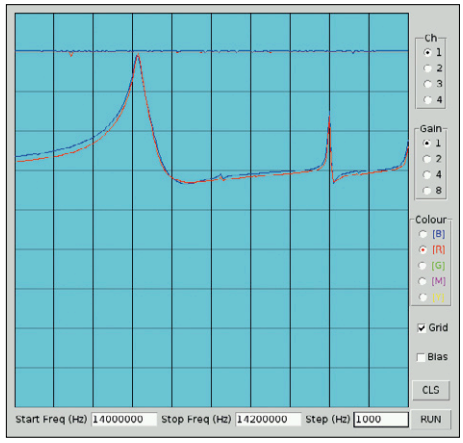


Figure 4b. Réponse en fréquence d'un quartz.

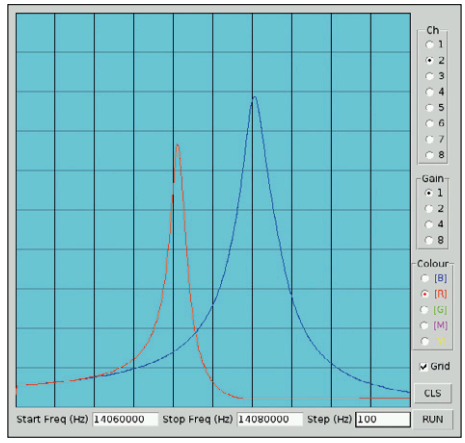


Figure 4c. Fréquence de résonance d'un quartz avec condensateur en parallèle (courbe bleue) et sans (courbe rouge).

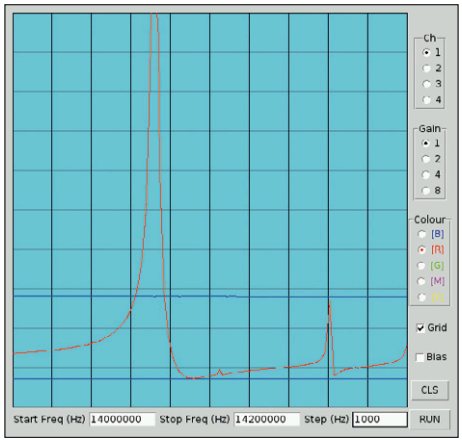


Figure 4d. Réponse d'un filtre RF mesuré sur CH1 (LIN).

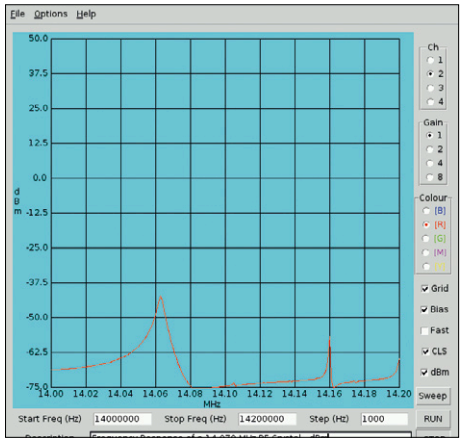


Figure 4e. Filtre 14 MHz analysé selon l'axe logarithmique Y (dBm).

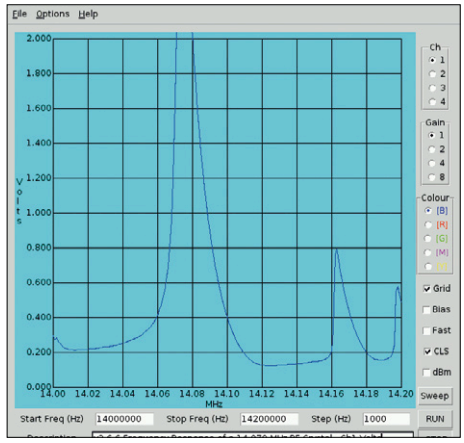


Figure 4f. Filtre 14 MHz analysé selon l'axe linéaire X.

Conclusion

Pour vérifier le bon fonctionnement de mon vobulateur, j'ai « balayé » un quartz et des filtres pour la bande radio de 20 m (14 MHz). Les copies d'écran (**fig. 4**) montrent différentes courbes de réponse. Le labo d'Elektor n'a ni construit ni testé mon vobulateur, mais c'est bien sur le site elektor labs [5] (où il a reçu 5 étoiles sur 5) que mon projet a grandi et abouti. Peut-être ce parcours, de mon propre blog jusqu'à cet article en passant par la vitrine et l'interactivité d'elektor-labs.com, vous donnera-t-il envie de lancer votre projet sur le même chemin.

J'ai conçu une carte pour le vobulateur (**fig. 5**) et je propose des kits d'assemblage avec le module DDS en option. Vous pouvez me contacter via mon blog [4] ou Twitter @TomHerbison.

(130484 – version française : Hervé Moreau)

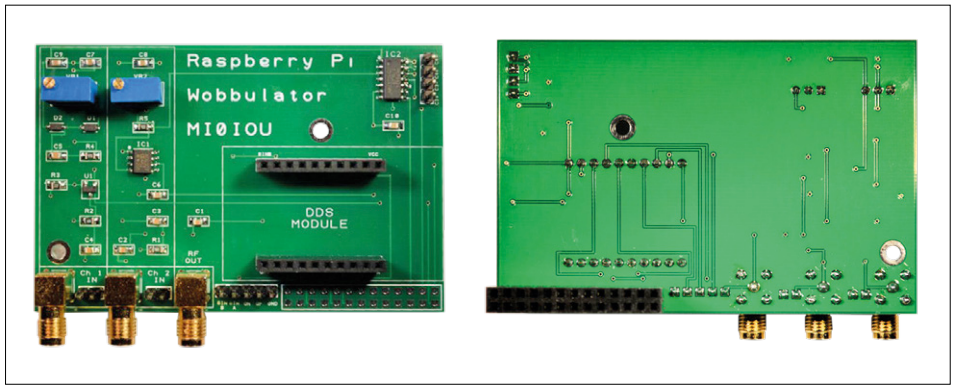


Figure 5. Des cartes pour le vobulateur à RPi sont disponibles auprès de l'auteur.

Liens

[1] MCP3424 : www.microchip.com/wwwproducts/Devices.aspx?product=MCP3424
[2] Code source (Python 3) du vobulateur RPi : https://github.com/mi0iou/
[3] Groupe Yahoo : http://groups.yahoo.com/neo/groups/rpiwobbulator/info
[4] Blog de l'auteur : www.asliceofraspberrypi.co.uk
[5] www.elektor-labs.com/project/raspberry-pi-wobbulator-130484-i.13612.html

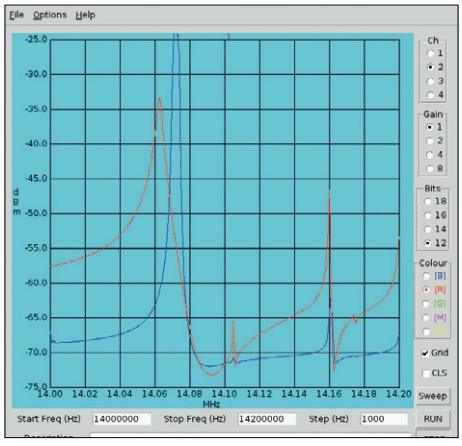


Figure 4g. Courbe des canaux 1 (bleu) et 2 (rouge) d'un quartz de 14 MHz.

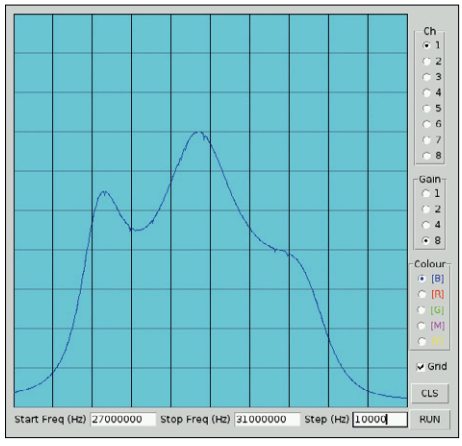


Figure 4h. Caractéristique d'un filtre passe-bande pour 29 MHz (10 m).

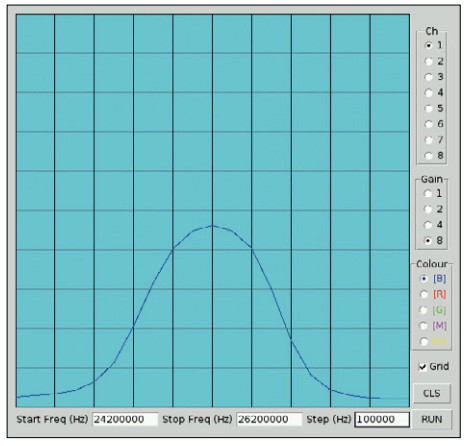


Figure 4hi. Caractéristique d'un filtre passe-bande pour 24,9 MHz (12 m).

appli d'entrée/sortie

une appli Windows pour piloter du matériel externe

Veikko Krypczyk (Allemagne)

Dans l'article « appli pour commander des circuits électroniques » de ce numéro, nous avons acquis les bases de la programmation d'applis pour la dernière version de Microsoft Windows. Nous allons utiliser une appli de ce type pour piloter des LED et des relais ainsi qu'afficher l'état d'entrées sur un PC (ou une tablette).

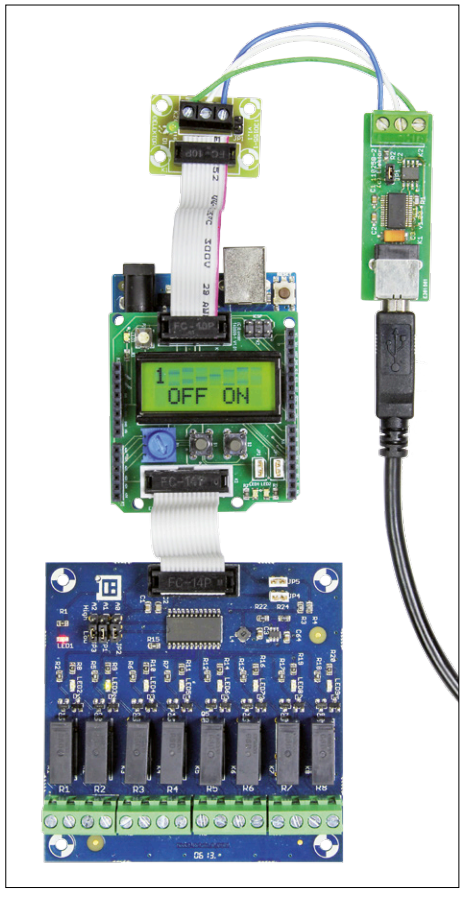


Figure 1. Grande flexibilité grâce au matériel modulaire.

Les applis Windows Store sont d'un *design* moderne, leurs fonctions sont réduites à l'essentiel et leur utilisation est agréable. Dans la section DÉCOUVRIR, nous avons présenté une introduction à la programmation d'applis Store pour le système d'exploitation Windows 8.1.

La communication avec du matériel externe est particulièrement intéressante pour les électroniciens. Il y a de nombreuses possibilités d'utilisation. Le mode d'utilisation (tactile) et l'interface utilisateur (moderne, minimaliste) sont tout à fait adaptés à la commande. Les applis peuvent tourner sur un ordinateur de bureau classique, un ordinateur portable ou une tablette. Il est possible de connecter un circuit électronique (à base de microcontrôleur) au port USB, afin de permettre une communication dans les deux sens.

Matériel

Pour notre petit projet, nous aurons recours à une carte déjà utilisée plusieurs fois dans Elektor : la carte Arduino Uno. Notre labo a créé un *shield* d'extension qui permet d'y connecter d'autres matériels (par ex. une carte à relais) [2a]. Pour le test, ce *shield* accueille deux LED, un écran LCD, deux boutons et un potentiomètre. Le *shield* d'extension dispose d'un module RS485 et d'un convertisseur RS485/USB pour la connexion à l'ordinateur. La transmission sans interférence du RS485 permet de tirer de longs câbles. Il est intéressant de pouvoir connecter d'autres matériels au *shield* d'extension (par ex. une carte à relais). Cela se déroule sans souci grâce au connecteur d'extension intégré à 14 broches. Il est également appelé connecteur Gnublin, car il permet de connecter les modules Gnublin de Benedikt Sauter et son équipe. Une carte à huit relais fait également partie de ces modules. Les modules matériels et leurs câbles sont reproduits à la **figure 1**, l'installation est décrite au point [2b].

Commandes simples

Le point fort réside dans le micrologiciel, présenté en [2b] et téléchargeable en [1]. Il suffit d'envoyer de simples commandes ASCII à la carte Arduino Uno (voir [2c] pour le protocole). Grâce à la commande « L 1 0 + <CR> », on peut allumer la LED sur le *shield*. Si on connecte la platine à relais, il est possible d'exciter l'un des huit relais avec la commande « R 0 X + <CR> » (remplacer X par un chiffre

de 0 à 7). Si on remplace le « + » par un « – », on peut éteindre la LED ou mettre au repos un relais.
Il est possible de contrôler l'état du bouton sur le *shield* avec la commande « B 0 0 ? <CR> » ou « B 0 1 ? <CR> ». La commande « A 0 0 # <CR> » permet de déterminer la position du potentiomètre (0 à 1023 en chiffres hexadécimaux).

Port COM virtuel

Afin d'interagir avec le matériel grâce à notre appli, nous devons être en mesure d'envoyer et de recevoir des données via un port COM virtuel (VCP). Les données transmises via VCP circulent en réalité sur un port USB, un pilote VCP *émule* une interface série classique pour Windows et les logiciels s'y référant (par ex. un programme de terminal ou notre application).
En effet, les ordinateurs modernes ne comportent plus de véritables ports série (ports COM) depuis longtemps. Le port COM permet pourtant une communication simple grâce à l'API du système d'exploitation. On utilisera un pilote de port COM virtuel du fabricant de puces FDTI [3]. Ainsi, le programme verra un port série COM et le transfert de données s'effectuera par ce biais.

Vous pourrez vérifier que tout fonctionne avec un programme de terminal par exemple. Il faut sélectionner le port COM créé par le module FTDI sur le convertisseur RS485/USB et paramétrer la vitesse de transfert à 9600 bauds.

Problématique

En principe, nous pouvons aisément transmettre des caractères par une interface série (virtuelle) grâce au framework .NET. Il faut pour cela se servir de la classe *SerialPort.* Cette classe encapsule les appels de l'API Windows pour contrôler un port série. Les propriétés de la classe permettent de régler la vitesse de transmission, le nom du port et d'autres paramètres. Des méthodes simples (*Write, Read*) permettent d'écrire et de lire des données. Cet usage ne pose aucun problème pour les applications Windows de bureau. Les applis Store ne peuvent pas accéder directement à cette classe ; elles sont isolées de l'environnement système et des autres applications, et tournent dans un *sandbox* du système d'exploitation. Le principe du bac à sable (*sandbox*) élève le niveau de sécurité, mais a également ses inconvénients. L'accès direct à de nombreux capteurs, fonctions du système et en particulier au matériel externe est restreint ou impossible. Si un tel accès est autorisé, il faut l'annoncer dans le manifeste de l'appli lors de son développement. L'utilisateur doit accepter ces autorisations d'accès étendues lors de l'installation de l'appli. Certaines fonctions du système sont exclues pour un accès direct, c'est-à-dire que les API Win32 correspondantes ne sont pas disponibles pour les applis Store. Il existe des alternatives (**tableau 1**), mais pas dans le cas de l'interface série.

Il y a toutefois des solutions : Windows 8.1 a été amélioré par Microsoft et l'accès au port USB (générique) est autorisé. La bibliothèque système *Winusb.sys* doit être installée sur l'ordinateur cible. Il est ensuite possible d'utiliser les classes de l'espace de noms *Windows.Devices.Usb* depuis l'appli [5]. Le point [6] décrit comment s'adresser à la puce FDTI par cette voie. En principe, cette procédure est envisageable pour notre tâche. Comme la plupart de nos lecteurs connaissent mieux le port COM virtuel, nous avons cherché une autre solution qui nous permet de passer outre les restrictions du bac à sable.

Maîtriser la communication

La solution consiste à ajouter dans le dossier de projet un *Brokered Windows Runtime Component* [7]. Les étapes suivantes expliquent comment activer l'accès au matériel par un port COM (via USB) [8] :

1. Il faut d'abord préparer une appli comme mentionné dans l'article « appli pour commander des circuits électroniques » dans la section Découvrir.

2. Ajoutez ensuite deux autres projets via l'explorateur de solutions sur la base du modèle « *Brokered WinRT Component Project Templates* ». Si ce modèle n'est pas encore disponible sur l'ordinateur de développement, il faut recourir à la recherche en ligne pour le trouver (**fig. 2**). Deux projets sont ajoutés : un projet C#, que nous appelons « *SerialPortSampleAppBroked* ». Ce projet permettra la communication grâce à la classe `SerialPort`. Les interfaces nécessaires à l'appli sont créées ici. Le projet ne peut toutefois pas être directement intégré à l'appli. Le compilateur ne le permet pas. Il faut donc passer par un proxy pour obtenir l'accès désiré. Le proxy est enregistré en tant que troisième

Tableau 1. Autres bibliothèques pour des accès proches du système à partir des applis Store [4]

Fonction du système sous Win32 (application de bureau)	API alternative pour applis Windows Store
Bluetooth	Windows.Networking.Proximity
Device enumeration (Function Discovery, PnP-X, WSD)	Windows.Devices.Enumeration
FAX	pas d'accès possible
Location API	Windows.Devices.Geolocation
Print	Windows.Graphics.Printing
Sensors	Windows.Devices.Sensors
Serial and parallel ports	pas d'accès possible
SMS	Windows.Devices.Sms
UPnP	Windows.Devices.Enumeration.Pnp
Windows Portable Devices	Windows.Devices.Portable
WSD	Windows.Devices.Enumeration

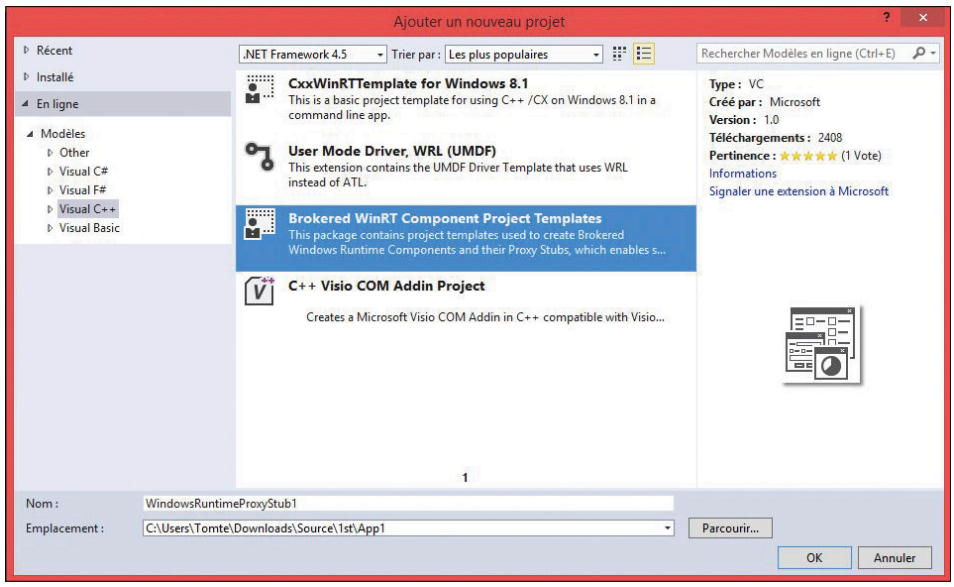


Figure 2. Ajout d'un projet (modèle : *Brokered Windows Runtime Component*).

projet (C++) dans le dossier de projet, nous avons choisi de le baptiser « *SerialPortSampleProxy* ». Il n'est pas nécessaire d'ajouter du code dans le projet proxy, il sert uniquement de pont entre l'appli *Store* et les composants Windows.

3. Tous les projets sont à présent disponibles et les références respectives sont créées. Il faut encore ajouter dans le projet proxy *SerialPortSampleProxy* une référence au projet *SerialPortSampleAppBroked* par le menu Explorateur de solutions|Références.

Il faut également connecter l'appli « *SerialPortSampleApp* » au projet proxy (**fig. 3**).
L'architecture globale est représentée dans la **figure 4**. Même si la procédure semble complexe, elle nous laisse indifférents en tant que développeur d'applications. La communication est assurée sans problème par la classe `SerialPort`. Il faut encore mentionner un inconvénient : le recours au *Brokered Windows Runtime Component* est critiqué par Microsoft pour des raisons de sécurité. Il n'est donc pas possible de distribuer ces applis dans le *Store,* mais elles peuvent être installées par *sideloading*, ou grâce à un compte développeur, directement sur l'appareil cible. On ne doit passer par le *sideloading* que si l'appli n'est pas destinée au grand public [9]. Cette restriction ne devrait pas poser problème pour des projets d'électronique.

Exemple d'appli

Le concept décrit a été intégré dans une petite appli. Ainsi, plusieurs sorties peuvent être activées ou désactivées à l'aide de boutons. L'état des entrées du matériel est affiché (**fig. 5**). Selon le but prévu, on peut adapter l'UI. Des icônes simples et une commande tactile intuitive sont la norme. Le code XAML définissant l'UI se trouve dans le **listage 1**.

La classe SP créée est représentée dans le **listage 2**. Elle doit être intégrée dans le projet *SerialPortSampleAppBroked,* car elle contient les accès à la classe `SerialPort`, déjà mentionnée. Les tâches sont exécu-

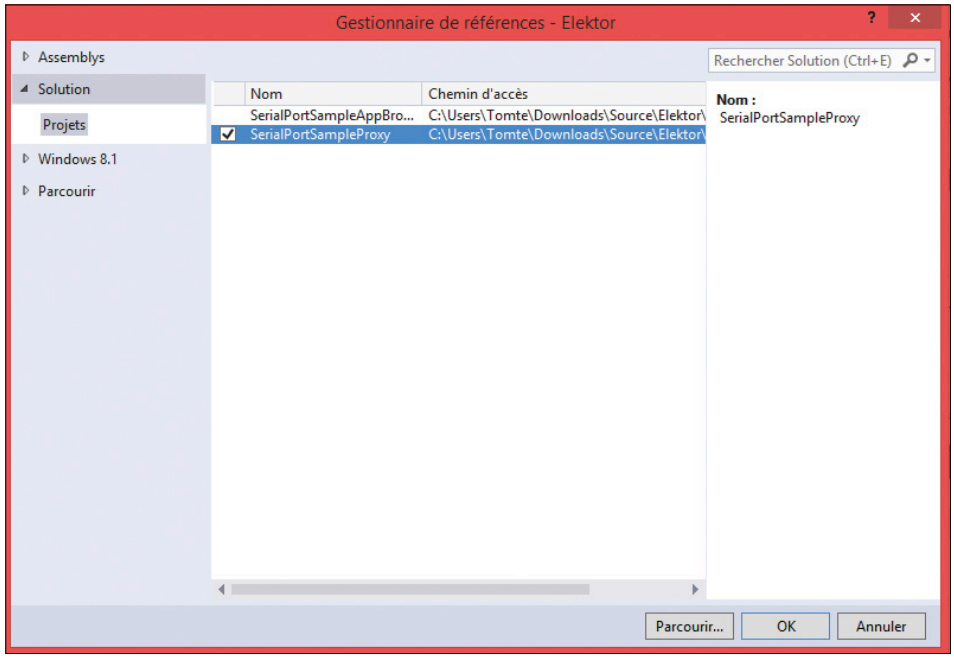


Figure 3. Référence de l'appli vers le projet proxy.

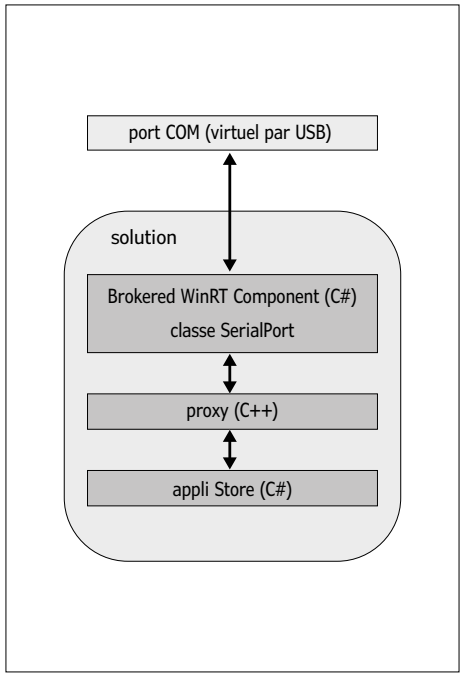


Figure 4. Architecture globale du logiciel pour accéder au port COM d'une appli.

tées à l'aide de la commande d'écriture `Write` ; la consultation de l'interface série passe par la commande `Read`.
Seul le constructeur de la classe `SP` est montré dans le listage. Le port correct et les paramètres initiaux (par ex. taux de transfert) sont déterminés au sein du constructeur. De même, il est possible de transmettre un paramètre au constructeur sous forme de notre chaîne d'instructions.

Il est ensuite aisé d'accéder au port depuis l'appli, par ex. en envoyant la commande « L 1 1 + <CR> » de la manière suivante :

Figure 5. Exemple d'appli : allumer des LED et faire commuter des relais.

Listage 1. Code XAML pour l'UI de l'exemple d'appli (extrait)

```
<Page
    x:Class="SerialPortSampleApp.MainPage"
    xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
    xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
    xmlns:local="using:SerialPortSampleApp"
    xmlns:d="http://schemas.microsoft.com/expression/blend/2008"
    xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006"
    mc:Ignorable="d">
    <Grid Background="{ThemeResource ApplicationPageBackgroundThemeBrush}">
        <Grid.RowDefinitions>
            <RowDefinition Height="150"/>
            <RowDefinition/>
            <RowDefinition/>
            <RowDefinition/>
        </Grid.RowDefinitions>
        <Image Source="Assets/SplashScreen.png" HorizontalAlignment="Left" Margin="50,20"/>
        <StackPanel Orientation="Horizontal"  Grid.Row="1" Margin="20">
            <ToggleButton Margin="10"  Width="200" Height="100"  Content="LED 1" FontSize="24"/>
            <ToggleButton Margin="10"  Width="200"  Height="100" Content="LED 2" FontSize="24"/>
        </StackPanel>
        <StackPanel Orientation="Horizontal"  Grid.Row="2" Margin="20">
            <ToggleButton Margin="10"  Width="200" Height="100" Content="Relay 1" FontSize="24"/>
            <ToggleButton Margin="10"  Width="200" Height="100" Content="Relay 2" FontSize="24"/>
            <ToggleButton Margin="10"  Width="200" Height="100" Content="Relay 3" FontSize="24"/>
            <ToggleButton Margin="10"  Width="200" Height="100" Content="Relay 4" FontSize="24"/>
            <ToggleButton Margin="10"  Width="200" Height="100" Content="Relay 5" FontSize="24"/>
            <ToggleButton Margin="10"  Width="200" Height="100" Content="Relay 6" FontSize="24"/>
            <ToggleButton Margin="10"  Width="200" Height="100" Content="Relay 7" FontSize="24"/>
            <ToggleButton Margin="10"  Width="200" Height="100" Content="Relay 8" FontSize="24"/>
        </StackPanel>
        <StackPanel Orientation="Horizontal"  Grid.Row="3" Margin="20">
            <ToggleButton Margin="10"  Width="200" Height="100" Content="Button 1" FontSize="24"
IsHitTestVisible="False"/>
            <ToggleButton Margin="10"  Width="200" Height="100" Content="Button 2" FontSize="24"
IsHitTestVisible="False" IsChecked="True"/>
        </StackPanel>
    </Grid>
</Page>
```

```
SerialPortSampleAppBroked.SP client
= new SerialPortSampleAppBroked.
SP("L 1 1 +\r\n");
```

D'autres méthodes de la classe SP permettent de consulter les entrées en lisant l'état de l'interface série. S'il faut réagir immédiatement à la modification d'un niveau d'entrée, les entrées doivent être consultées régulièrement, ce qui peut être contrôlé par une minuterie (classe `DispatcherTimer` [10]).

Conclusion

Les applis fascinent par leur facilité d'utilisation et la concentration sur une tâche. Les applis Windows Store tournent sur tablettes et ordinateurs portables, qui disposent le plus souvent de ports USB. Ces applis sont donc une alternative aux logiciels classiques pour ordinateurs de bureau pour piloter du matériel externe. La programmation n'est pas de la sorcellerie. La logique de programmation est basée sur le C# et l'UI est déclarée au moyen de code XML. Une interface utilisateur réussie relève de la gageure, le minimalisme est à privilégier. La communication avec du matériel externe n'est pas sans problème, mais reste possible. ⏮

(150276 – version française : Thierry Destinobles)

Listage 2. Code C# pour la communication via le port COM

```
public sealed class SP
    {
        private SerialPort _serialPort;
        public SP(string command)
        {
            string portName = "";
            string[] ports = SerialPort.GetPortNames();
            if (ports.Count() > 0)
            {
                int portInt = 0;
                foreach (var port in ports)
                {
                    var portNumber = port.Substring(port.Length - 1);
                    if (portInt < Int32.Parse(portNumber))
                    {
                        portInt = Int32.Parse(portNumber);
                    }
                }
                if (portInt != 0)
                {
                    portName = "COM" + portInt;
                    _serialPort = new SerialPort(
                        portName,
                        9600,
                        Parity.None,
                        8,
                        StopBits.One);
                    _serialPort.ReadTimeout = 200;
                    if (_serialPort != null && _serialPort.IsOpen)
                        _serialPort.Close();
                    _serialPort.Open();
                    _serialPort.Write(command);
                    _serialPort.Close();
                }
            }
        }
    }
```

Liens

[1] www.elektormagazine.fr/150276
[2a] Elektor, juillet/août 2014, www.elektormagazine.fr/140009
[2b] Elektor décembre 2014, www.elektormagazine.fr/140328
[2c] Elektor, juillet/août 2013, www.elektormagazine.fr/130154
[3] www.ftdichip.com/Drivers/VCP.htm
[4] https://msdn.microsoft.com/en-us/library/windows/apps/hh464945.aspx
[5] https://msdn.microsoft.com/en-us/library/windows/hardware/dn303342(v=vs.85).aspx
[6] www.ftdichip.com/Support/Documents/InstallGuides/AN_271%20D2xx%20WinRT%20Guide.pdf
[7] https://msdn.microsoft.com/en-us/library/windows/apps/dn630195.aspx
[8] http://ioi.solutions/serial-port-access-metro-style-app/
[9] https://technet.microsoft.com/fr-fr/windows/jj874388.aspx
[10] www.wpf-tutorial.com/misc/dispatchertimer

construisez vous-même un

poste de soudage de CMS avec Platino

Martin Kumm, Sunil Malekar et **Clemens Valens**

Le poste de soudage Platino est un outil compact et abordable construit autour de la carte à microcontrôleur Platino d'Elektor. Le fer à souder est un modèle de la série RT de Weller équipé, outre l'élément de chauffe, d'un capteur de température. Exactement ce qu'il faut pour se fabriquer un régulateur de température précis et rapide, particulièrement adapté à la soudure des CMS.

Il n'y a aucune raison de se priver des composants modernes, mais quand ce sont des CMS, on hésite, parfois on croit l'obstacle insurmontable. Or, avec un matériel de soudage adéquat et du doigté, ça marche. Avec une pointe fine sur un fer ordinaire, on s'aperçoit que bien peu de chaleur arrive effectivement à l'extrémité de la panne, car le régulateur est trop lent pour compenser la chute de température.

Sur les postes de soudage modernes pour CMS, le capteur de température est intégré à la pointe de chauffe : la réponse est immédiate, mais ces postes coûtent cher. En revanche, le coût est moindre quand le seul matériel consommable à acheter est la tête du fer, c'est-à-dire le fer proprement dit sans le poste de régulation.

Caractéristiques techniques

- Tension de chauffage : 12 à 18 V CC, 5 A
- Platino Elektor avec microcontrôleur ATmega328P
- LCD à 2 lignes de 16 caractères
- Tête de soudage Weller série RT avec jack stéréo 3,5 mm
- Prise pour jack stéréo 3,5 mm
- Réglage de température entre 90 °C et 450 °C
- Mise à température en MLI par Platino
- Affichage des températures de consigne et actuelle sur LCD
- Réglage de la température par codeur rotatif

À la pointe

La firme Weller propose toute une armada de têtes de fer à souder. Dans leur assortiment, les pointes RT sont particulièrement intéressantes pour se doter d'un poste de soudage. Elles se distinguent par des caractéristiques spéciales : le raccordement s'effectue par un jack stéréo de 3,5 mm, facile à insérer dans la prise correspondante (**fig. 1**). Cette liaison conduit le signal du capteur de tempé-

Figure 1. Pratique, compacte et disponible dans de nombreuses exécutions : la tête de soudage série RT de Weller.

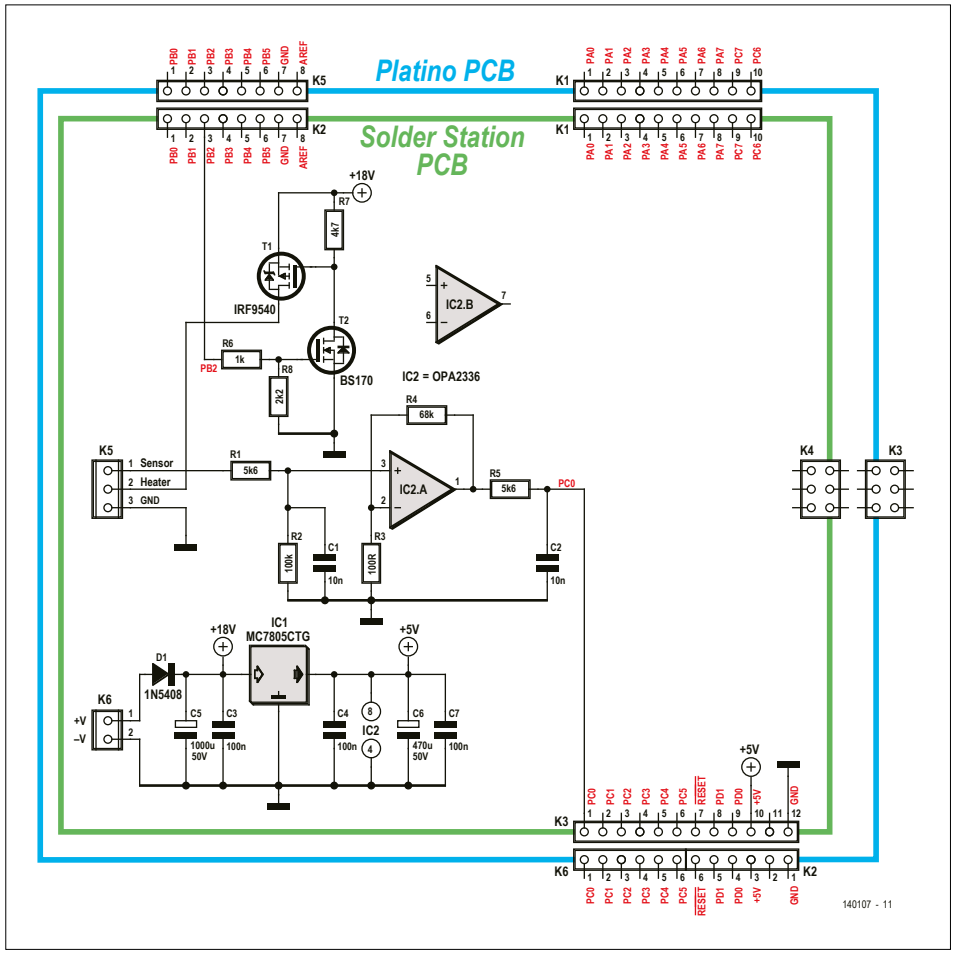


Figure 2. Le circuit du poste de soudage à installer sur une carte d'extension pour Platino.

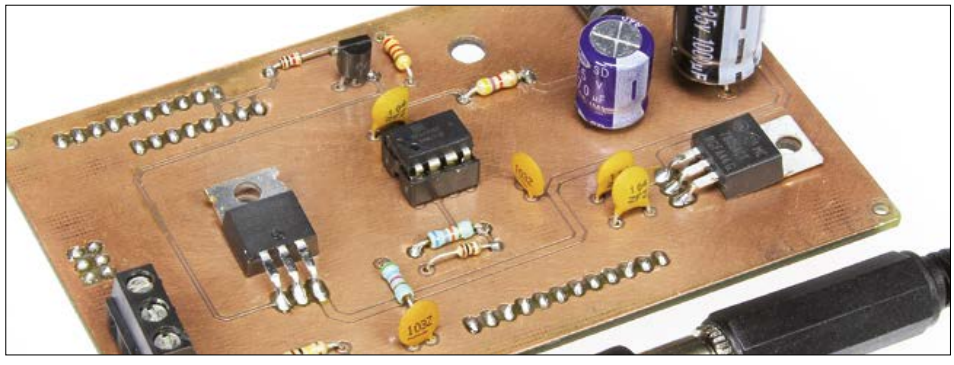


Figure 3. Le prototype de carte d'extension réalisé par le laboratoire. Le circuit imprimé a été fraisé !

rature ainsi que le courant commuté pour la résistance interne de chauffe. La tête est aussi dotée d'un manchon ergonomique pour une prise en main pratique. Sa taille réduite permet de bien voir les petites pastilles à souder.
Il y a sur le site du fabricant [1] plus de 30 modèles différents, avec tous le même connecteur et une puissance maximale de 40 W, mais des pannes droites, courbes, rondes, biseautées, ou en forme de lame, de longueur, d'épaisseur et de diamètre variés à l'infini.
On peut alimenter l'élément chauffant sur un bloc secteur, régulé ou non, de maximum 18 V, mais d'au moins 3 A. Une tension élevée raccourcit le temps de chauffe et de changement de régime. Une alimentation pour portable ou la tension à bord d'une voiture ferait l'affaire. Sous 18 V, il ne faut que 4 s pour atteindre 330 °C. Malgré l'extrême finesse de son fût cylindrique (sur lequel est montée la panne), ce fer convient parfaitement pour souder aussi de la tôle de blindage ou même des radiateurs. Associé au poste de régulation, son usage ne se restreint donc pas aux CMS, mais en fait un outil polyvalent.

Carte enfichable

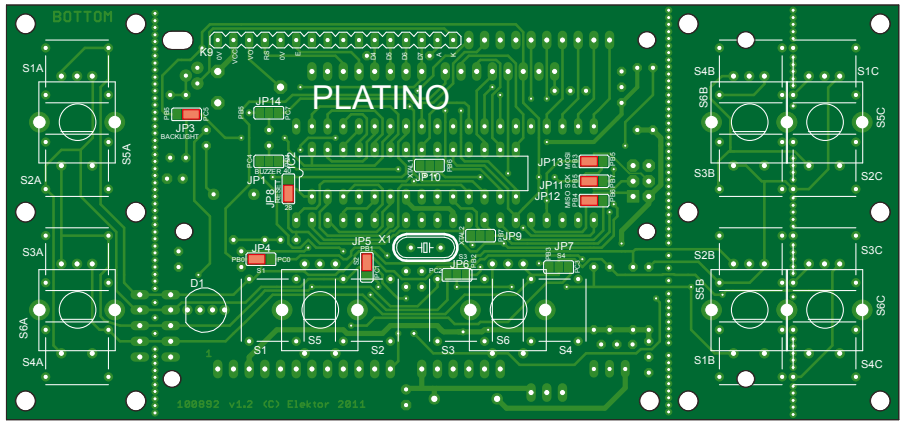
Fondamentalement, un poste de soudage remplit la fonction de régulateur de température : mesurer la température actuelle de la pointe et la comparer à celle de consigne pour doser la puissance de chauffe en conséquence. La température de soudage est ici réglable entre 90 et 450 °C.

La **figure 2** montre le circuit composé d'un amplificateur à faible bruit pour la mesure de la température et d'un commutateur de puissance MOSFET qui fournit le courant de chauffe au fer. Le circuit imprimé fraisé du prototype, monté au laboratoire (**fig. 3**), ne présente rien d'extraordinaire.

Pour la mesure de la température, on utilise le capteur embarqué dans le fer dont le coefficient thermique est de 16 µV/K. Il en résulte une tension de 7 mV environ à la température maximale de 450 °C. Bien trop peu pour l'appliquer à l'entrée analogique du contrôleur AVR qui attend un maximum de 5 V. On passe donc par l'amplificateur opérationnel OPA2336 (IC2.A) qui se caractérise par une faible tension de décalage (typ. ±60 µV) et

Tableau 1. Pose des cavaliers sur Platino. Les autres restent non connectés.

cavalier	position
JP3	PC5
JP4	PB0
JP5	PB1
JP8	28
JP11	PB5
JP12	PB4
JP13	PB3



Liste des composants Platino

Résistances :
Standard : 5%, 0,25 W
R3 = 47 Ω
R4, R5, R6, R7, R10, R12 = 10 kΩ
R11 = 4,7 kΩ
P1 = pot. ajust. 10 kΩ horizontal

Condensateurs :
C5, C6 = 100 nF, au pas de 5 mm

Inductance :
L1 = 10 µH

Semi-conducteurs :
IC1 = ATmega328P-PU, programmé, e-choppe 140107-41
T1 = BC547C

Divers :
S5A = codeur rotatif avec bouton
K2, K6 = embase à 1x6 picots au pas de 2,5 mm
K3 = embase à 2x3 picots au pas de 2,5 mm
K5 = embase à 1x8 picots au pas de 2,5 mm
K9 = embase à 1x16 picots au pas de 2,5 mm
LCD1 = LCD, alphanumérique à 2x16 caractères** (e-choppe 120061-74)
support étroit DIP28
*voir [3] pour la description de Platino.
** type cf. ELPP: https://github.com/ElektorLabs/PreferredParts

une alimentation asymétrique. Le gain, déterminé par R4 et R3, vaut 680. La résistance R1 limite le courant quand lors de la mise à température, la pleine tension de chauffage est appliquée à la résistance de chauffe. En plus, les réseaux R1/C1 et R5/C2 constituent des filtres passe-bas pour affaiblir les hautes fréquences, de sorte que le signal de mesure ainsi conditionné atteigne finalement par K3-1 l'entrée PC0 du convertisseur A/N de Platino.

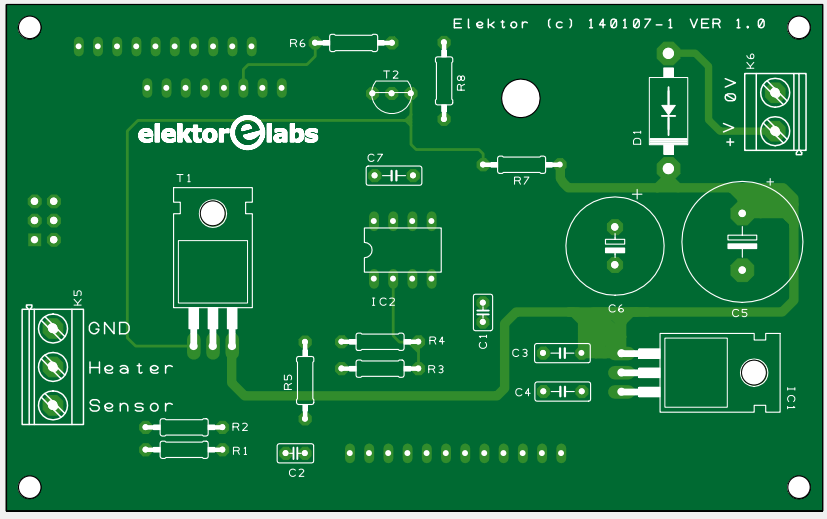
Liste des composants du poste de soudage

Résistances :
Standard : 5 %, 0,25 W
R1, R5 = 5,6 kΩ
R2 = 100 kΩ
R3 = 100 Ω
R4 = 68 kΩ
R6 = 1 kΩ
R7 = 4,7 kΩ
R8 = 2,2 kΩ

Condensateurs :
C1, C2 = 10 nF, au pas de 2,5 mm
C3, C4, C7 = 100 nF, au pas de 5 mm
C5 = 1 000 µF, 50 V, au pas de 7,5 mm
C6 = 470 µF, 50 V, au pas de 5 mm

Semi-conducteurs :
D1 = 1N5408
T1 = IRF9540
T2 = BS170
IC1 = 7805
IC2 = OPA2336

Divers :
K2 = embase à 1x8 picots au pas de 2,5 mm
K3 =embase à 1x12 picots au pas de 2,5 mm
K1, K4 = voir ci-dessus
K5 = borne encartable à 3 vis au pas de 5 mm



K6 = borne encartable à 2 vis au pas de 5 mm
support DIP 8 pour IC2

tête de soudage Weller série RT [1]
prise prolongateur pour jack stéréo 3,5 mm

Boîtier Bopla 2616 (Farnell 1217479)

La réponse de Platino arrive par sa sortie PB2, c'est le signal de commutation qui passe par K2-3 pour atteindre les pilotes à MOSFET. Un petit d'abord, le BS170 à canal N T2 qui commande le MOSFET de puissance IRF9540, capable de commuter à l'aise de forts courants, ici les 2,5 à 3 A pour l'élément de chauffe du fer, quand la tension d'entrée sur la carte enfichable est de 18 V sur K6. On pourrait aussi bien travailler sous 12 ou 15 V. Pour simplifier le câblage, il y a un régulateur de 5 V dans le circuit pour alimenter l'amplificateur opérationnel ; il servira aussi du 5 V à Platino. Le connecteur K5 est réservé au fer à souder (masse, résistance de chauffe et capteur).

Platino et son logiciel

Connaissez-vous Platino ? Non ? Alors l'encadré (page suivante) est fait pour vous. Lisez-le d'abord et pour mieux vous y retrouver, sachez qu'il y a aussi un article [3] qui a été consacré à cette carte à microcontrôleur.

Le signal du capteur, préalablement conditionné, est appliqué à la ligne PC0 (broche 23) de l'ADC0 du contrôleur Platino pour être numérisé.

Le courant de chauffage du fer lui est prodigué en MLI (modulation en largeur d'impulsion) pour doser la puissance. Le signal est rythmé par le temporisateur Timer1. Il est disponible sur la sortie OC1B qui est reliée à PB2 (broche 16). Le codeur rotatif de Platino permet de régler la température voulue du fer, affichée alors sur le LCD. L'exécution du logiciel se passe d'horloge de précision, le contrôleur peut tourner sur son oscillateur interne à 8 MHz.

Le logiciel pour ce projet est assez simple, il a été rédigé en C avec le Studio4 d'Atmel, il est disponible comme projet pour l'ATmega328.

Après l'initialisation du matériel et les salamalecs d'usage sur l'afficheur, on entre de plain-pied dans la boucle principale. C'est là que s'effectuent les mesures de température de la panne et les comparaisons avec la valeur de consigne, mémorisée en EEPROM. Si la température est trop basse, on augmente le rapport cyclique du signal MLI ; on le diminue si elle est trop haute. L'algorithme repose sur la régulation proportionnelle et différentielle (PD, système PID sans l'intégration), c'est-à-dire que l'amplitude de la variation de la puissance de chauffe est proportionnelle à la différence entre la température actuelle et la température de consigne. Cette méthode permet d'atteindre une précision du résultat de l'ordre de 3 à 4 °C.

L'élément de chauffe

Dans cette logique, tant que la différence de température est d'au moins 5 °C, le système fonctionne en tout ou rien, pleine puissance ou arrêt, selon le signe de la différence, pour accélérer le processus de régulation. Ensuite, il passe en MLI. Puis, au voisinage de la consigne, le rapport cyclique de la MLI varie par pas de 1/7e (MLI – MLI/7).
Pour produire la MLI, Timer1 est configuré en mode rapide à 9 bits au moyen du registre B de comparaison. Le diviseur préalable (*prescaler*) est positionné sur 256. Ce réglage conduit à une fréquence inférieure de MLI avec une bonne stabilité et plus de précision lors de l'étalonnage. On en calcule la fréquence comme ceci :

fréquence MLI = clock/(N*prescaler) =
8 MHz / (512*256) = 62 Hz

avec N = 512 pour le mode rapide à 9 bits en MLI (2^9 = 512)

Encore un point important. Il faut absolument arrêter le signal MLI pendant toute mesure de température. La proximité immédiate du capteur avec la panne rendrait sa tension, déjà très faible, indétectable. Même si l'on ne supprimait la MLI qu'au début de la mesure (OCR1B = 0), les pics parasites en provenance de la broche de chauffage produiraient des fluctuations de la valeur du CAN. C'est pourquoi il faut arrêter la production de MLI déjà 7 ms avant de lire le CAN.

Tableau 2. Réglage des fusibles pour l'ATmega328P

fusible	valeur
Low	0xe2
High	0xd9
Extended	0xff

Le capteur

Pour une température de capteur de 50 °C, la valeur du CAN est de 67 et pour 450 °C, de 1020. Entre 50 et 450 °C, on calcule le facteur multiplicateur comme suit :

coefficient =
(450 – 50) / (1020 – 67) = 0,419

Si la valeur donnée par le CAN est de 0, la température réelle avoisine 30 °C. On doit donc ajouter un décalage de 30, d'où la formule :

température = temp_CAN * 0,42 + 30

Le logiciel reconnaît la rotation du codeur au changement de niveau produit sur les broches PB0 et PB1 du contrôleur, la nouvelle valeur est mémorisée en EEPROM en appuyant sur le bouton.
Pour épargner tant l'énergie que la panne du fer, le logiciel la ramène à plus basse température après 15 min. Comme le programme ne sait pas si vous êtes encore occupé à souder, vérifiez la valeur affichée et appuyez sur le bouton. Le logiciel réactivera alors le fer à souder.

Construction, essais et premières armes

Avant tout, il faut munir la carte Platino des cavaliers (repérés en couleur) prévus dans le **tableau 1**. Le dessin vous y aidera. Installez d'abord sur la carte les composants et le LCD, le codeur rotatif et le support pour le contrôleur ATmega328P. Remarquez que K9 (pour le LCD) et le codeur à la position S5A se montent au dos de la carte.
Aucun souci pour le montage de la carte enfichable. Les connecteurs K2 et K3 vont sur la face des soudures. Pour que tout tombe à pic, branchez-les d'abord sur Platino, puis soudez. Il ne vous manque que le contrôleur programmé. Quoi, vous ne l'avez pas ? Vous pouvez le comman-

Liens

[1] www.weller.de/de/Weller--Produkte.html?cat_id=ID151
[2] www.elektormagazine.fr/140107
[3] www.elektormagazine.fr/100892
[4] www.martin-kumm.de/smd_solder_station

der auprès d'Elektor (140107-41) ou le programmer vous-même avec le fichier de données hexadécimal [3]. N'oubliez pas de régler les fusibles en suivant les instructions du **tableau 2**. Et n'oubliez pas que le contrôleur roule sur son oscillateur interne à 8 MHz.

Avec le contrôleur convenablement préparé, vous pouvez assembler les trois cartes : Platino, la carte enfichable et celle du LCD (**figure 4**). La tête n'est pas livrée avec un câble. Le câble est souple, avec des conducteurs d'un diamètre d'environ 1 mm. Il n'y a pas de blindage. Puisque vous savez comment souder et assembler un tel projet, on a tout lieu de croire que votre poste de soudage va fonctionner du premier coup. Rien ne vous empêche plus d'enfermer l'électronique dans le joli boîtier Bopla (**fig. 5**). Les circuits imprimés y trouvent place et il en reste encore suffisamment pour mettre un puissant bloc d'alimentation secteur. Reliez un câble muni d'une prise pour jack de 3,5 mm au connecteur K5 et branchez la tête de soudage. Raccordez l'alimentation à K6. Au moins 50 W, sinon ce n'est pas la panne qui chauffe rapidement mais le bloc secteur qui chauffe lentement. Réglez maintenant une température de consigne et observez sur l'afficheur l'évolution rapide de la température mesurée. Bon travail !

(140107 – version française : Robert Grignard)

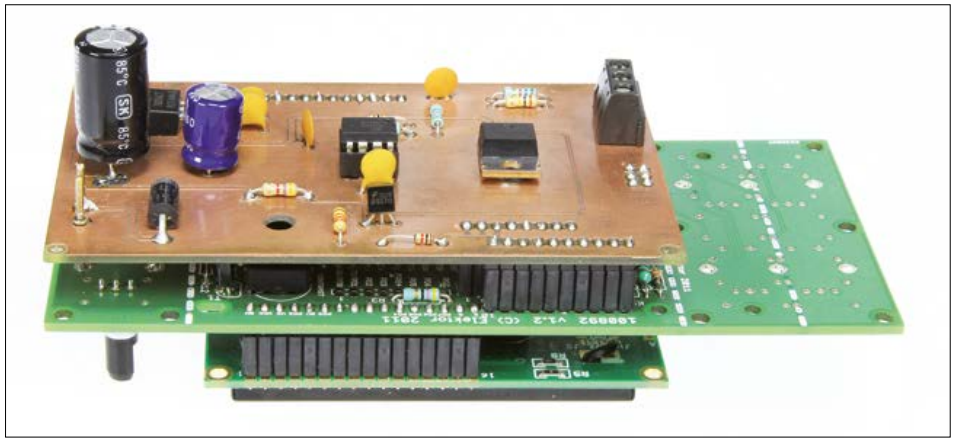


Figure 4. Le triplan composé de Platino, du poste de soudage et du LCD.

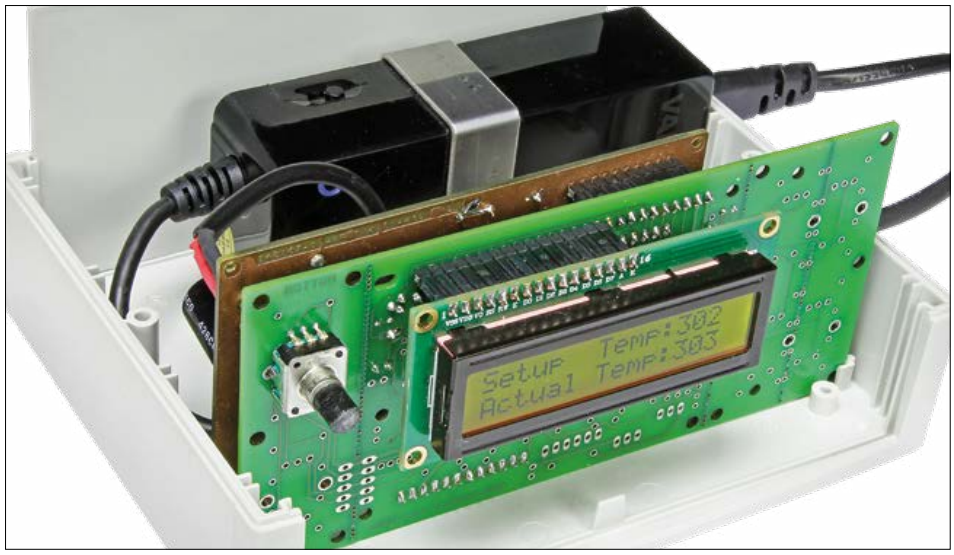


Figure 5. Le circuit complet mis en boîte avec l'alimentation secteur.

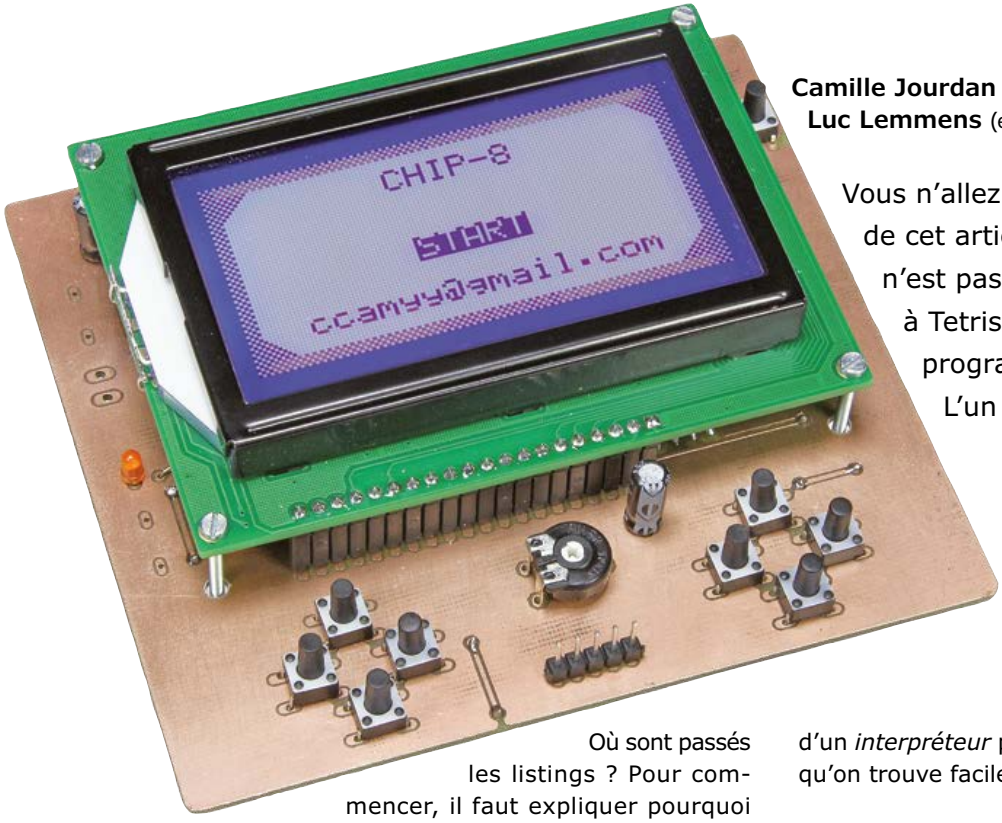
Platino

Platino [3] est une puissante carte équipée d'un contrôleur à 8 bits AVR d'Atmel, sous un boîtier à 28 broches, conçue en 2011 par Elektor. En version de base, elle porte un ATmega328P, mais on peut aussi bien l'équiper d'un ATmega168, ATMega324 ou du très puissant ATmega1284. On règle son fonctionnement au moyen d'un codeur rotatif et les résultats s'affichent sur un LCD à 2 lignes de 16 caractères.
Le nom de Platino lui vient de son modèle, la star italienne Arduino. D'ailleurs, ce poste de soudage a d'abord été construit sur la base d'Arduino [4]. Il y a bien sûr des différences entre les deux, mais surtout une grande similitude : l'une et l'autre cartes disposent de connecteurs compatibles sur lesquels on peut brancher une carte d'application ou d'extension, comme un *shield*.
Cette souplesse leur permet de remplir des tâches très variées, comme le testeur de transistors Platino (mars 2015), l'alimentation de table régulée par Platino (avril 2014) ou le générateur de fonctions avec Platino (janvier 2015). On peut se procurer dans l'e-choppe d'Elektor les circuits imprimés et les contrôleurs déjà programmés.

édition	projet	platine	contrôleur
10/2011	Platino	100892-1	-
5/2015	Robot aérateur de cave	140432-1	-
10/2012	P & A dans les nuages	120306-1	-
4/2014	Alimentation de table	130406-1	-
1/2015	Générateur de fonctions	130407-1	130407-41
3/2015	Testeur de transistors	130544-1	130544-41
7/2015	Poste de soudage	140107-1	140107-41

SAME : émulateur de jeux vidéo Chip-8

Single Arcade Machine Emulator : une machine virtuelle sur un PSoC



Camille Jourdan Gay (Japon) ccamyy@gmail.com
Luc Lemmens (elektor.labs)

Vous n'allez pas me croire, mais le propos de cet article, malgré les apparences, n'est pas de vous faire jouer, ni à Pong ni à Tetris, mais bien de vous encourager à programmer.
L'un n'exclut pas l'autre, d'ailleurs.

Où sont passés les listings ? Pour commencer, il faut expliquer pourquoi cet article ne reproduit presque pas une ligne de code. Cette absence, apparemment contradictoire avec le sujet, est délibérée : plutôt que de l'aborder par un examen forcément superficiel dans un article de quelques pages, toute la description du logiciel a été réunie dans un document téléchargeable, très complet et clairement structuré. Le code source est évidemment intégralement disponible.

Chip-8 en 2015 ?

La plateforme virtuelle Chip-8 a bientôt 40 ans [1] ! Elle avait permis, dans les années 70 et 80, de créer des jeux vidéo sans se soucier de leur contexte matériel. C'est l'idée, toujours actuelle, des *machines virtuelles*, comme Java ou autres : le logiciel imite ou simule, ou *émule* comme on dit, les fonctions d'un matériel qui n'existe pas. Ainsi, les jeux vidéo Chip-8, écrits pour une plateforme virtuelle, peuvent-ils tourner sur une plateforme réelle avec laquelle ils n'ont pas la moindre affinité. En y programmant une machine virtuelle, on transformera n'importe quel système embarqué en console de jeux vidéo : un téléphone tactile, un ordinateur, un réfrigérateur ou une machine à laver, peu importe pourvu que leur microcontrôleur soit à même d'émuler une machine virtuelle (simple). Il suffit d'un µC, d'un afficheur, de quelques boutons, et d'un *interpréteur* pour exécuter le code des jeux vidéo Chip-8 qu'on trouve facilement sur la toile.

Idéal pour des étudiants, l'exercice est hautement pédagogique et très instructif pour quiconque veut acquérir de l'expérience en programmation d'*architecture système* : registres, pile, espace mémoire, compteur ordinal… Cela permet de bien comprendre et de façon ludique le fonctionnement d'un ordinateur de base.

Le cœur de la machine virtuelle Chip-8 ressemble à celui d'un vrai µC, avec son processeur, ses registres, sa pile, sa mémoire et ses deux temporisateurs. Il est capable d'exécuter 35 codes opératoires (ou instructions) à 16 bits (appelés *opcodes*) avec lesquels sont programmés les jeux vidéo, à l'exclusion de toute autre ressource. Tout se passe entre quelques boutons pour la saisie, et un afficheur pour les yeux et éventuellement un buzzer pour les oreilles.
L'interpréteur est un programme qui, sur un système embarqué, exécute le code des jeux vidéo Chip-8 récupérés tout faits (p. ex. `Tetris.ch8`). Il est écrit dans le langage du µC embarqué. Il traduit les 35 codes opératoires *virtuels* en un code exécutable par le µC réel. Dans le code d'un jeu, le code opératoire `00E0` p. ex. commande l'effacement de l'écran en une seule ligne, mais dans le code du système embarqué, ce seront plusieurs lignes dont le nombre et la nature exacte dépendront du matériel utilisé.

La partie de la mémoire du système embarqué utilisée par l'interpréteur est divisée en deux: il a son propre espace de travail pour ses propres variables, pour l'interfaçage avec le matériel etc., et un espace pour les variables virtuelles de l'architecture Chip-8, p. ex. le compteur ordinal (*program counter*) ou le pointeur de pile (*stack pointer*). Le code du jeu lui-même est stocké dans un tableau (*array*) d'octets que l'interpréteur lit pour en traduire chacun des codes opératoires en une fonction équivalente exécutable par le µC auquel il s'adresse. En C, pour traiter ces codes opératoires, on utilise tout bonnement une structure d'aiguillage en cascade de type *switch/case*. Par exemple comme ceci :

```
while(1)
{
  opcode = gameMemory[programCounter];
  switch(opcode)
  {
    case 0001: ExecuteOpcode0001();
      break;
    case 0002: ExecuteOpcode0002();
      break;
    //...
  }
  programCounter++;
}
```

Certains des 35 codes virtuels sont faciles (ADD ou SUB p. ex.), d'autres donnent du fil à retordre, notamment ceux qui permettent de dessiner sur l'afficheur ou celui qui produit un nombre aléatoire, une fonction très utile dans les jeux. Il sera impossible de rentrer ici dans les détails, mais rien de cet interpréteur ne vous sera caché ; tout est décrit soigneusement dans un document téléchargeable de 31 pages, dont nous ne saurions trop recommander la lecture [2]. Vous y trouverez le détail de la programmation de chaque code. Il nous a semblé vain et frustrant de ne montrer ici que certains aspects qui resteraient incompréhensibles, faute de vue d'ensemble. Cet article décrit en revanche la réalisation d'un circuit embarqué réel, qui permet de jouer à des jeux vidéo Chip -8. Ce principe est applicable à n'importe quel autre circuit à µC doté des entrées sorties adéquates. Quand, grâce à la documentation téléchargeable, vous aurez vu comment j'ai configuré mon système (temporisateurs, entrées, etc.) et programmé l'interpréteur, il se pourrait que mon approche et mes ruses vous soient utiles et vous inspirent. Même si vous prenez une autre route pour arriver au même résultat, vous rencontrerez sans doute les mêmes difficultés que moi, comme p. ex. le manque de mémoire ou le trop petit nombre d'entrées disponibles.

Virtualité matérialisée

Assez parlé de virtuel, abordons le versant matériel. Voici ce qu'il nous faut :

- un afficheur matriciel de 64 *32 points ou plus
- 8 boutons (en fait, 3 ou 4 suffisent pour la plupart des jeux)
- un buzzer (sauf si vous préférez les jeux silencieux)
- un microcontrôleur capable d'émuler la machine virtuelle Chip-8 dont voici les spécifications :
  - 35 codes à 16 bits (*opcodes*)
  - 4 Ko de mémoire
  - pile de 16 bits à 16 niveaux, utilisée pour l'adresse de retour
  - 16 registres à 8 bits d'usage général et 3 registres spécifiques (pointeur de mémoire, compteur ordinal, pointeur de pile).
  - 2 décompteurs à 8 bits à 60 Hz : *Delay Timer* et *Sound Timer*.

Ce n'est pas la mer à boire. J'ai même prévu la compatibilité de mon système avec une version plus récente de Chip-8, dite *Super Chip* (SCHIP), qui admet un affichage sur 128 *64 points et connaît quelques codes et registres supplémentaires.

Si vous vous attendiez à un schéma compliqué, vous serez déçu par la **figure 1**. En dehors des boutons, de l'afficheur à cristaux liquides et du régulateur de tension (IC3)), il n'y a que deux circuits intégrés : un PSoC 29466, qui a le statut de microcontrôleur (IC1), et une EEPROM I²C. Ah, j'oubliais Bz1. Au départ, j'avais imaginé ce circuit pour un seul jeu, résidant

Ceci n'est pas une console de jeux

Ceci n'est pas une console de jeux, il ne faut pas en attendre un confort aux standards de 2015. Il est normal que la balle de Pong, par exemple, soit aussi difficile à distinguer sur un écran à cristaux liquides que si elle vous avait été envoyée sur un vrai court de tennis par Rafael Nadal.
L'EEPPROM est divisée en blocs de 4 Ko, ce qui correspond à la mémoire vive de Chip-8, avec un jeu par bloc. La taille maximum d'un jeu Chip-8 est de 3584 octets, ce qui laisse 512 octets pour l'en-tête, dont seuls 64 octets sont utilisés : 2 pour une pseudo somme de vérification, 1 pour la vitesse du jeu, 16 pour le clavier, et le reste pour le titre du jeu. Soit 8 jeux dans 32 Ko d'EEPROM. La taille moyenne d'un jeu Chip-8 est de 256 octets (1 Ko pour S-Chip) de sorte que l'EEPROM pourrait en contenir bien plus.
Après l'initialisation, le PSoC lit les en-têtes des jeux présents dans l'EEPROM et en affiche les titres pour que le joueur fasse son choix. Puis il charge le jeu choisi et applique la configuration du clavier et le paramètre de vitesse, également sauvegardés dans l'en-tête du jeu.

Espace indéfini

L'espace de travail a une taille indéfinie qui dépend du matériel et des choix du programmeur. Par exemple, le code opératoire en cours d'exécution n'a pas d'espace mémoire défini par l'architecture Chip-8; c'est une combinaison du compteur ordinal (*program counter*) et de l'espace mémoire virtuel.
Le programmeur peut soit choisir d'en faire une macro

```
#define OPCODE ((READ_PROGRAM_MEMORY(PC) << 8) |
READ_PROGRAM_MEMORY(PC+1))
```

ou d'utiliser une variable dans l'espace de travail

```
short OPCODE = ((READ_PROGRAM_MEMORY(PC) << 8) |
READ_PROGRAM_MEMORY(PC+1))
```

Le premier cas limitera l'utilisation de la mémoire du µC, le second cas l'utilisation du CPU en cas d'appel fréquent.

dans la mémoire du microcontrôleur. Chemin faisant, je me suis tellement amusé que l'idée a germé de loger huit jeux dans une EEPROM séparée. Il a suffi d'ajouter un menu de sélection, pour que le joueur puisse faire son choix parmi les jeux disponibles. C'est d'ailleurs comme ça qu'est apparu le bouton de remise à zéro S9, inutile dans la version *monogame*.

Un PSoC (*Programmable System on Chip*) associe sur la même puce un microcontrôleur et divers accessoires numériques (compteurs, convertisseurs…) ou analogiques (amplificateurs, filtres, …) qu'il est possible de combiner librement à l'aide du logiciel. Mon choix du PSoC 29466 est fondé d'une part sur son prix (30 € pour l'indispensable programmateur, et 5 € pour le PSoC lui-même) et d'autre part sur la présence de plusieurs sous-ensembles numériques qui permettent de disposer de deux temporisateurs, d'un générateur de signal MLI (pour le buzzer) et d'un générateur aléatoire intégré. Pour les boutons S1 à S8, il n'y a même pas besoin de résistances de polarisation sur les entrées correspondantes du PsoC, celles-ci sont internes et configurables.

Pour réaliser les deux temporisateurs à 8 bits déjà évoqués, je fais appel à plusieurs modules du PSoC. Leur configuration est possible soit en écrivant laborieusement des valeurs données dans des registres prévus pour cela, soit en passant par un outil spécifique de configuration, beaucoup plus confortable : le *PSoC Designer*, doté pour cela d'une interface graphique avec laquelle il suffit d'agencer les modules comme bon nous semble avant d'établir les connexions souhaitées entre modules et entre ports au moyen d'opérateurs logiques. Ce configurateur permet aussi, par un simple clic, de définir des variables globales, comme la fréquence du microprocesseur, la configuration des ports, etc. *PSoC Designer* crée automatiquement des fonctions d'interaction directe du microcontrôleur et des modules (p. ex. `Timer1_Start()`, `Timer1_Stop()`, etc.) Les ports reliés aux boutons se voient configurés en entrées munies de résistances interne de polarisation au niveau haut. Les autres ports sont configurés en sorties. La **figure 2** illustre la configuration des modules. Le nom des temporisateurs `DELAY_TMR` et `SND_TMR` ne laisse aucun doute sur leur fonction ; le second s'auto-décrémente et fait sonner le buzzer tant que sa valeur est différente de 0. Quant à `SND_PWM`, c'est le générateur de signal MLI de 4 kHz rendu audible par Bz1. C'est PRS8 qui délivre les nombres aléatoires pour le code opératoire RND de Chip -8.
Voici quelques particularités du PsoC.

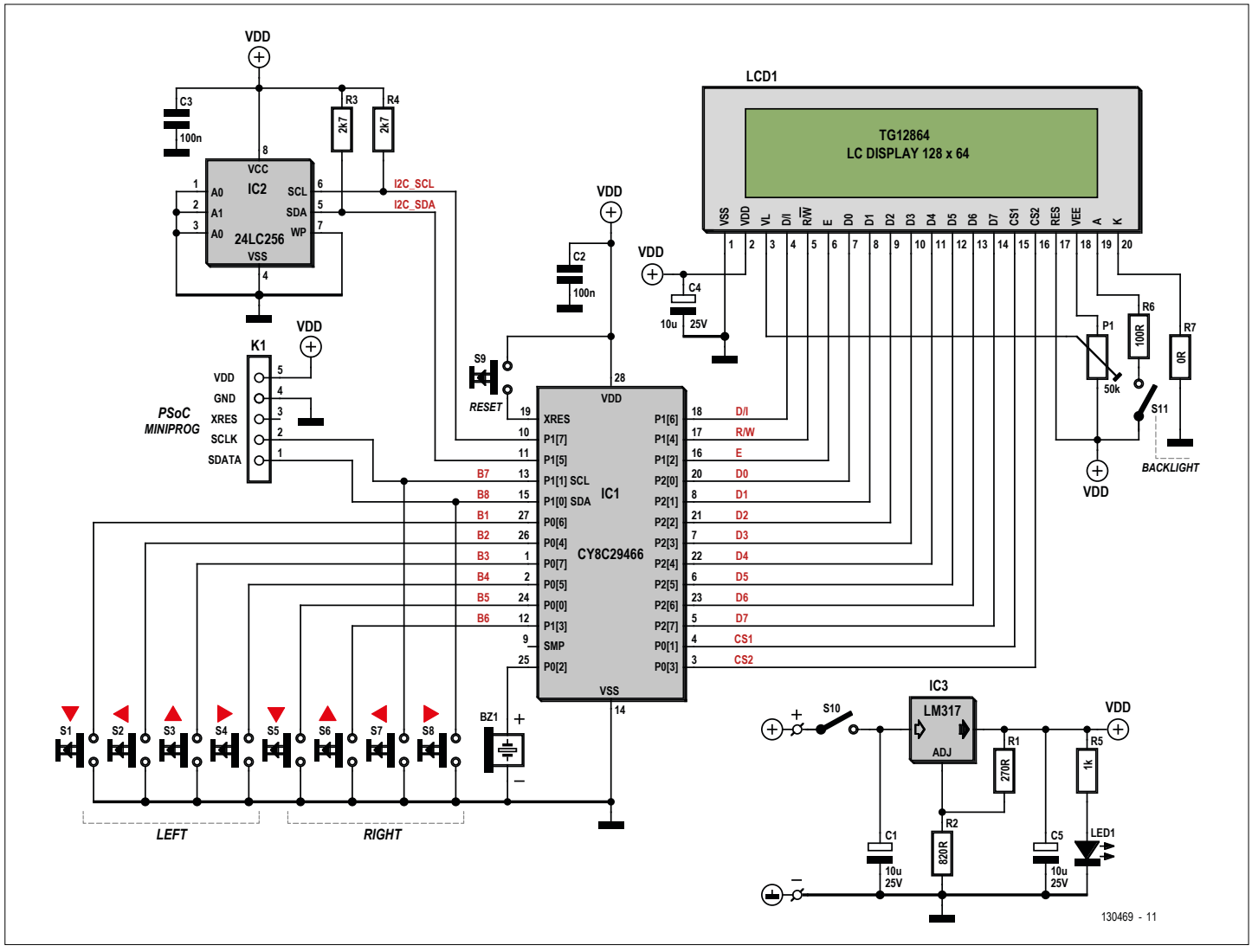


Figure 1. La simplicité du schéma de l'émulateur est trompeuse : dans le PsoC IC1 se cache un arsenal programmable qui fait tourner la machine virtuelle Chip-8

L'exercice est ludique et pédagogique pour acquérir de l'expérience en programmation d'*architecture système* : registres, pile, espace mémoire, compteur ordinal...

Boutons : L'état de S1 à S8 est lu par une fonction qui retourne une variable dont chaque bit correspond à l'un des boutons. C'est simple, mais quand vous examinerez le logiciel et ma documentation, vous verrez cependant qu'il a fallu ruser, à cause des résistances internes dont il faut que la polarisation haute soit établie avant la lecture du port. Sinon l'entrée reste au niveau bas même une fois que le bouton a été relâché.

Affichage : Le choix est vaste et les techniques de commande des afficheurs divergent. Ici il est fait appel à une commande par une puce KS0108 qui utilise le *bit-banging* parallèle. La broche DI distingue les commandes des données, la ligne RW la lecture de l'écriture, les deux broches CS1/CS2 la partie droite de la partie gauche de l'écran (128*64 = 2 x 64*64) et enfin le signal EN pour la synchronisation. Les 8 broches du port 2 sont utilisées pour les données lues ou écrites. Dans ma documentation [2], on trouvera une macro appropriée pour passer le port de données du PSoC en entrées (HIGHZ) ou en sorties (STRONG).

Temporisateurs : Le PSoC 29466 offre trois oscillateurs (VC1, VC2, VC3) en cascade à partir de l'horloge du système (24 MHz). Ici les diviseurs sont configurés de sorte que VC1 = 1,5 MHz, VC2 = 93 kHz et VC3 = 360 Hz. Cette dernière fréquence, la plus basse que nous puissions tirer du PsoC, sera divisée par 6 pour obtenir la temporisation à 60 Hz propre à Chip -8. La documentation explique comment faire avec une temporisation virtuelle (*virtual timer*). Vous y trouverez également des explications sur les subtiles combinaisons logiques, dans le PsoC, de la sortie du compteur et du module PWM.

L'interpréteur

La grande affaire de l'interpréteur est de simuler le processeur de l'architecture Chip-8 dans la mémoire du microcontrôleur. Pour les registres déjà mentionnés, seuls quelques octets sont requis, mais il ne s'agit pas de s'emmêler les pinceaux. À la mémoire de jeu Chip-8, il faut allouer 4 Ko. Mais comment faire avec un PSoC qui n'a que 2 Ko de mémoire vive ? Là encore, on trouvera dans la documentation une astuce de programmation vraiment digne d'intérêt, mais dont la finesse dépasse le cadre de cet article. Ne vous privez pas de l'étudier à tête reposée, vous ne serez pas déçu.
Sans entrer davantage dans les détails ici, signalons qu'il est aussi fait usage de l'une des caractéristiques remarquables et, partant, un des avantages des PSoC sur les autres circuits programmables, qui est la possibilité d'une reconfiguration dynamique du PSoC par l'intermédiaire de commandes I²C comme s'il s'agissait d'une EEPROM. C'est cela qui nous permet de charger un nouveau jeu dans le PsoC sans avoir à le reprogrammer. Quelqu'un qui remplacerait le PSoC par un µC sans reprogrammation dynamique pourrait émuler une espace de 4 Ko dans 2 Ko de RAM, mais charger un jeu dynamiquement dans la ROM depuis une EEPROM, ça non !

Nous en sommes arrivés au stade où l'interpréteur, après avoir préparé la mémoire, peut commencer à simuler les cycles du code Chip -8. Nous avons déjà vu qu'il s'agissait pour l'essentiel d'une boucle while(1) qui égrène les codes opératoires du jeu avec un *switch/case*, les exécute et incrémente généralement le compteur ordinal (sauf p. ex. en cas de saut). Les autres tâches de notre interpréteur sont la scrutation des boutons, la détection d'erreurs éventuelles (p. ex. par débordement de la pile ou par défaut d'accès à la mémoire) et d'autres besognes comme p. ex. introduire un temps d'attente pour éviter que certains jeux s'exécutent trop vite. Il ne vous échappera pas à ce sujet que comme nous sommes en présence de codes opératoires de 16 bits dans un tableau de 8 bits, l'incrémentation se fait par pas de 2.

Somme toute, le squelette de l'interpréteur n'a rien d'imposant :

```
while(1)
  {
    //Clear the flags
    OS_FLAGS = 0;
    ERROR = ERROR_NO_ERROR;
    input_read(&buttons);
    //fetch the opcode (16 bits)
    OPCODE = ((READ_PROGRAM_MEMORY(PC) << 8) |
READ_PROGRAM_MEMORY(PC+1));

    switch((OPCODE >> 12)& 0xF)//we decode the opcode
by groups of 4 bits
      {
      /*OPCODE DECODING AND EXECUTION*/
      }
```

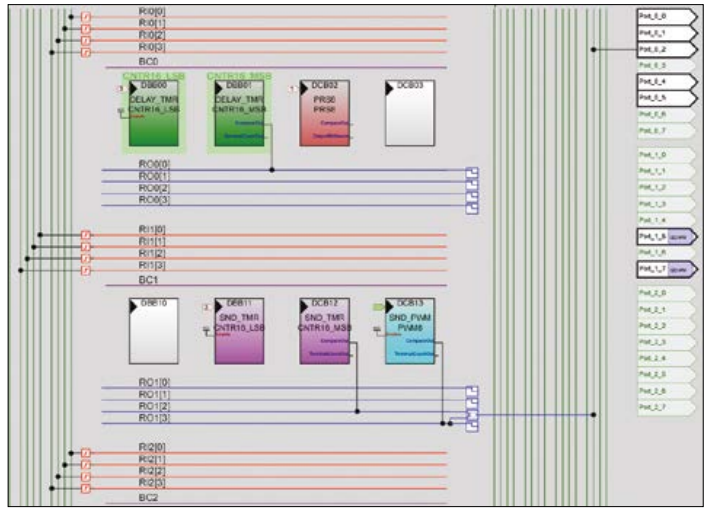


Figure 2. Configuration des modules du PSoC.

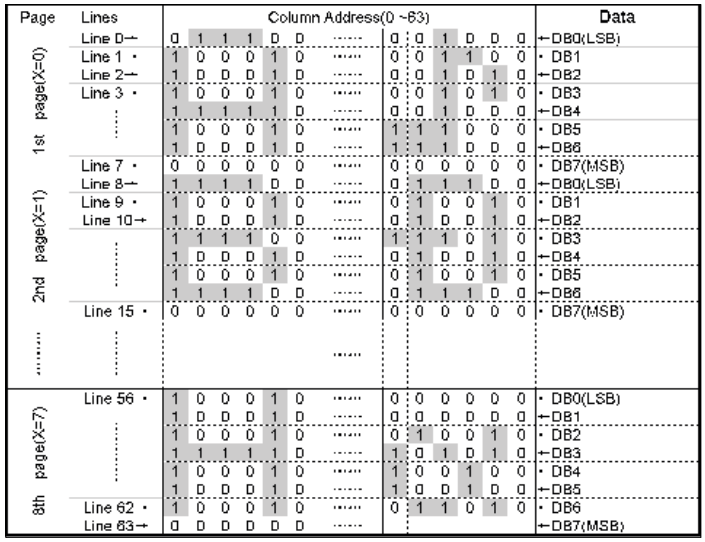


Figure 3. La configuration des pixels pour l'afficheur n'a pas grand-chose à voir avec leur organisation dans l'architecture Chip-8

Figure 4. Le circuit imprimé dessiné en double face est facile à réaliser en simple face.

```
    if(ERROR)
    {
    /*ERROR HANDLING*/
    }

    if(!((OS_FLAGS >> PC_NOT_INCREMENT_FLAG)&1))
    {
      PC += 2; // Increment the PC except if told not
to
    }
  }
```

Nous voici arrivés aux 35 codes opératoires, parmi lesquels nous distinguerons :

- les opérateurs arithmétiques comme ADD et SUB que la plupart des µC connaissent
- les opérateurs de branchement comme JMP et CALL, apparemment simples, mais qui, parce qu'ils affectent l'architecture virtuelle, doivent être maniés avec précaution
- les opérateurs qui affectent la mémoire, qui ne posent aucun problème aux µC riches en mémoire, mais qui imposent le recours à des astuces si l'espace mémoire du µC est plus exigu que celui de l'architecture Chip-8
- les opérateurs qui affectent le matériel et dépendent directement de la configuration matérielle

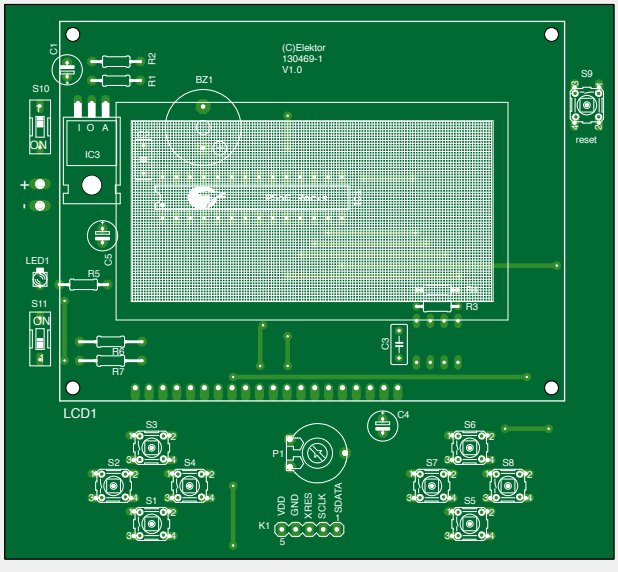
Liste des composants

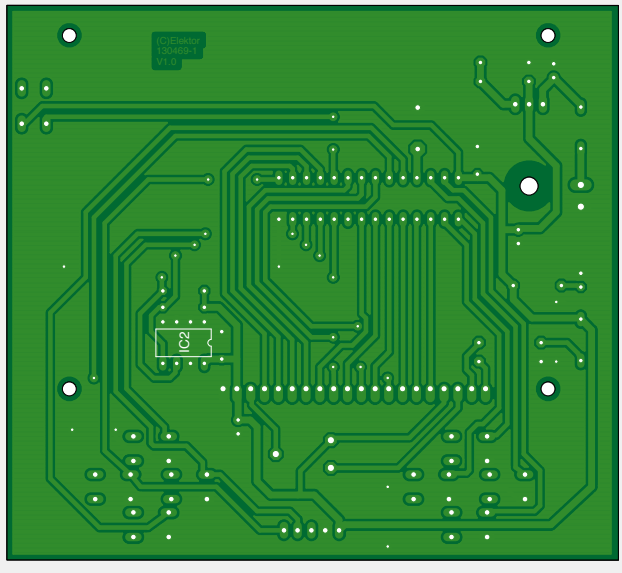
Résistances :
R1 = 270 Ω 250 mW 5 %
R2 = 820 Ω 250 mW 5 %
R3,R4 = 2k7 250 mW 5 %
R5 = 1 k 250 mW 5 %
R6 = 100 Ω 250 mW 5 %
R7 = fil (cf texte)
P1 = 47 k aj.

Condensateurs :
C1,C4,C5 = 10 µF 50 V radial
C1,C2,C3,C4,C5,C6,C7 = 100 nF 50 V 20 %

Semi-conducteurs :
LED1 = LED, rouge, 3 mm
IC1 = PSoC CY8C29466 EPS 130469-41
IC2 = EEPROM 24LC256
IC3 = LM317

Divers :
S1 à S9 = poussoir encartable 24 V, 50 mA, 6x6 mm
S10, S11 = interrupteur unipolaire
support DIL à 8 broches (pour IC2 – cf texte)
LCD1 = afficheur à cristaux liquides 128 x 64 points (p. ex. *Vatronix* TG12864B-03 ou *Midas* MC128064A6W -- cf texte)
Bz1 = buzzer 5 V





Tout cela est décrit par le menu dans la documentation en ligne, de même que les codes opératoires spécifiques aux temporisateurs, aux boutons et aux nombres aléatoires. Nous conclurons ici sur une catégorie particulière, les codes opératoires graphiques de Chip -8. On a vu que le format d'afficheur standard est une matrice de 64x32. Comme nous disposons d'un écran de 128x64 points, un pixel Chip-8 correspond à 4 pixels réels. L'origine (0,0) se trouve dans le coin en haut à gauche. Les codes graphiques sont de loin les plus difficiles à transposer, entre autres parce que la logique de notre afficheur est entièrement différente de celle de l'architecture Chip-8 : celle-ci écrit en «lignes» de 8 pixels tandis que l'afficheur LCD écrit en «colonnes» de 8 pixels. Chip-8 dessine les pixels à partir de coordonnées quelconques en groupes de largeur fixe (8 bits) et de hauteur variable.
L'afficheur de 128x64 est divisé en deux écrans de 64x64, divisés à leur tour en 8 pages, elles-mêmes divisées en 64 colonnes (**fig. 3**). Nous sommes donc contraints d'écrire nos pixels par colonnes de 8 dans l'une des 2*8*64 cases, ce qui implique une hauteur fixe pour ces groupes de pixels. Nous ne pourrons utiliser que des ordonnées fixes (lignes 0, 8, 16, 24, 32, 40, 48 et 56) et chaque opération d'écriture affectera aussi certains des pixels voisins, ce qui demande une certaine gymnastique. Le document en ligne vous explique comment on s'y prend et donne d'autres explications précieuses sur la gestion de l'affichage.

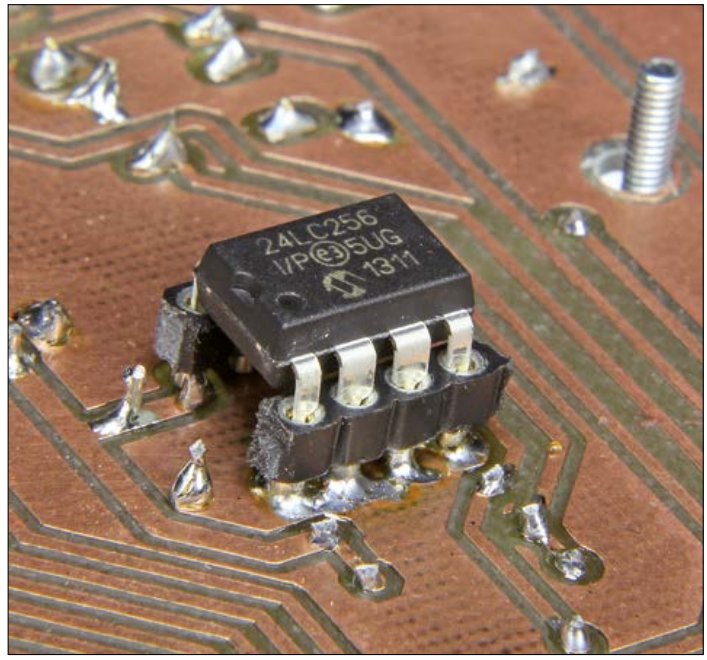


Figure 5. IC2 est obligatoirement monté sur un support à confectionner soi-même à partir de barrettes, pour pouvoir en souder les broches sur les pistes sous le circuit imprimé.

Réalisation

Après tant de virtualité, nous voici arrivés aux choses concrètes. Le tracé du circuit imprimé (**fig. 4**) proposé est double face, mais au prix de quelques ponts de câblage vous pourrez facilement réaliser le vôtre en simple face, d'autant qu'il n'est fait appel qu'à des composants traversants. L'afficheur à 128 x 64 pixels avec sa puce KS0108 ne posera pas de problèmes d'approvisionnement, mais attention à son brochage. Nous avions commandé un TG12864B-02 (Vatronix), mais reçu un TG12864B**-03** dont les broches 19 et 20 d'anode et de cathode du panneau de rétroéclairage sont interverties. Gare au suffixe ! D'où la présence de la très philosophique résistance R7 de 0 Ω, en fait un prosaïque pont de câblage, pour changer la configuration de ces deux broches selon le modèle d'afficheur que vous aurez. Le cas échéant, croisez R6 et le pont de câblage (R7) de sorte que la broche 19 soit mise à la masse par R7 et la broche 20 reliée à V_{DD} par R6.

Farnell propose le MC128064A6W (*Midas*), mais qui veut trouver meilleur marché cherchera son bonheur sur Ebay.
Sans être obligatoire, le rétroéclairage offre au joueur un confort certain. Il ne consomme que 20 mA.
Si l'EEPROM est montée sous le circuit imprimé (côté soudures), c'est pour en faciliter l'accès ; si vous voulez rajouter des jeux ou en changer, il est plus facile d'extraire l'EEPROM de son support pour la mettre sur un programmateur que si elle se trouvait… sous l'afficheur. Remarquez sur la **figure 5** le type particulier du support à confectionner soi-même pour pouvoir le souder sur la face cuivrée du circuit imprimé!
Le régulateur IC3 est monté à plat sur le circuit imprimé. Si vous gravez un circuit imprimé double face comme celui des photographies, il ne faudrait pas que la surface métallique du LM317, sur laquelle il règne le potentiel V_{DD} (+5 V), entre en contact avec le cuivre de la surface, même si celui est hors potentiel. Par précaution, il vaut mieux ménager un écart suffisant et fixer le régulateur avec une vis en nylon (**fig. 6**).
La programmation du PSoC s'effectue avec une interface *PSoC MiniProg* reliée à K1, directement depuis l'environnement de programmation *PSoc Designer* (nous avons utilisé la version 5.4). Durant la programmation, le circuit entier peut être alimenté par cette interface et l'alimentation normale est inutile.

Figure 6. On distingue le montage du régulateur à quelque distance de la surface cuivrée.

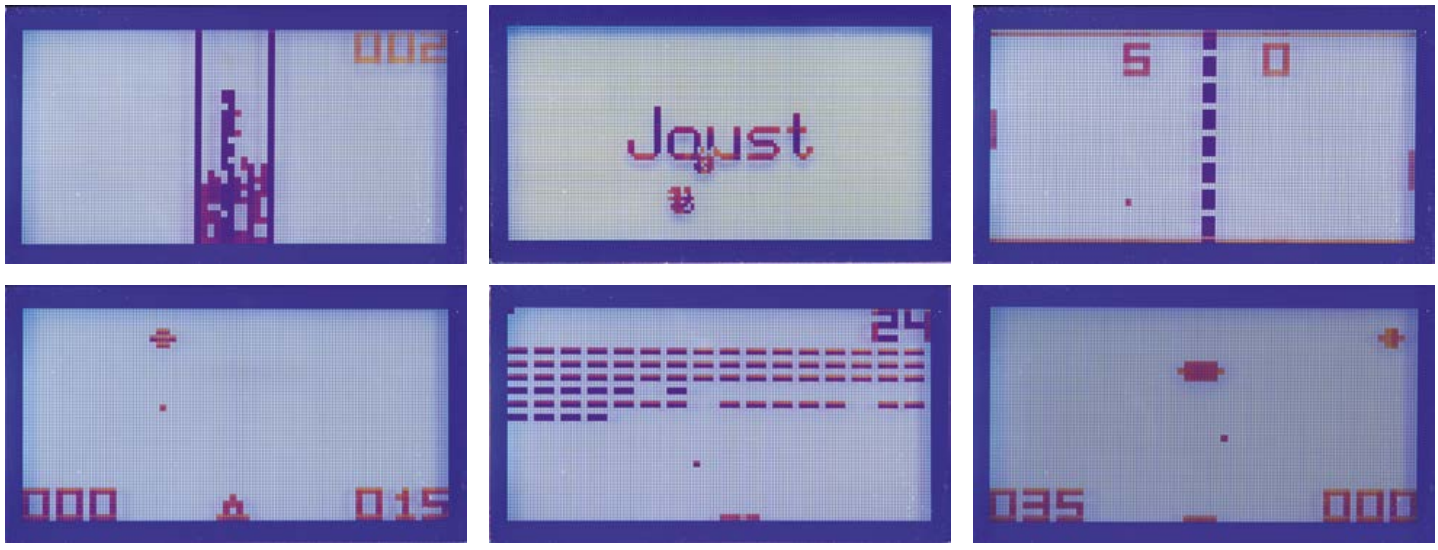


Figure 7. Menu de sélection des jeux que le PSoC trouve dans l'EEPROM.

Microprogramme (*firmware*)

Elektor fournit le code (source) pour programmer les jeux dans l'EEPROM et le code de l'émulateur pour le PsoC, mais pas de jeux ! Le PSoC programmé disponible dans l'e-choppe [8] (130469-41) ne contient que le *firmware*. Vous n'aurez aucun mal à télécharger du code de jeux Chip-8 [3]. Certains sont dans le domaine public, d'autres pas. Respectez le *copyright* éventuel ; celui-ci reste en vigueur même après des décennies. Connectez à K1 un programmateur MiniProg (*Cypress*) si vous voulez programmer vous-même le microprogramme dans la mémoire flash du PSoC.

Voici les trois manières de charger un jeu :

- insérer le jeu dans l'EEPROM avec un outil tiers (qui permet de copier un fichier vers une EEPROM)
- insérer le jeu dans l'EEPROM avec l'outil [9] que j'ai mis à disposition ; dans ce cas, la carte de l'émulateur (fig. 4) sert pour flasher l'EEPROM.
- insérer le jeu directement dans la mémoire flash du PSoC. En ce cas là, il faut modifier les valeurs par défaut des variables de l'interpréteur pour y insérer le jeu. Le code doit donc se trouver dans le tableau `PROGRAM_MEMORY` juste après les polices de caractères (`global.c`).

Pour finir, voici les trois scénarios pris en compte par l'interpréteur :

- EEPROM absente : lancement du jeu présent en mémoire du PSoC (qu'il ait été copié depuis une EEPROM précédemment connectée ou depuis le programmateur du PSoC)
- EEPROM présente, absence d'en-tête (`header`) : lancement du jeu dans l'EEPROM avec les réglages par défaut
- EEPROM présente avec en-têtes (`headers`) : lancement du menu de sélection, puis du jeu choisi avec les réglages indiqués. ⏮

(130469)

Qui suis-je ?

Après mes études d'ingénieur en génie électrique à l'INSA de Lyon (France) et une année d'échange à l'université de Tokyo, je suis au Japon en volontariat international en entreprise chez *Ichikoh Industries*, fabricant d'éclairages automobiles.

J'ai travaillé dans l'électronique, le traitement d'images et dans le développement de logiciels à Tokyo et à Paris.

Ce projet Chip-8 a été conçu et réalisé sur une initiative personnelle pendant mon temps libre, au Japon, avec du matériel d'*Akihabara*, le quartier électronique de Tokyo, un lieu de pèlerinage que je recommande aux lecteurs d'Elektor !

Je remercie Matthieu Denoual de m'avoir aidé à publier ce projet, ainsi que mon directeur de laboratoire Yoshio Mita, et Luc Lemmens bien sûr !

bienvenue dans votre e-choppe

Elektor recommande

L'été sera cool

Cet été, Elektor vous propose un festival de bonnes affaires à un rythme *tropicool* ! Peu importe la météo, sur le site d'Elektor il fera beau et agréablement frais tous les jours. Au risque de provoquer des tachycardies chez les plus émotifs d'entre nos lecteurs, voire des arrêts cardiaques, nous avons préparé une irrésistible sélection de produits électroniques, triés sur le volet pour vous. Du début à la fin de l'été, ce sera une méga-parade de nouveaux produits et d'offres qui déchirent. À l'heure de mettre sous presse, j'ignore encore certains détails, mais d'après ce que j'ai pu voir, je vous conseille de visiter régulièrement notre e-choppe www.elektor.fr.

La température augmente, mais les prix baissent chez Elektor !

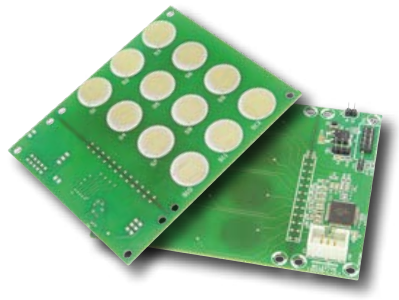
Ne manquez aucun de ces événements !

Clemens Valens, labo Elektor

www.elektor.fr/festival

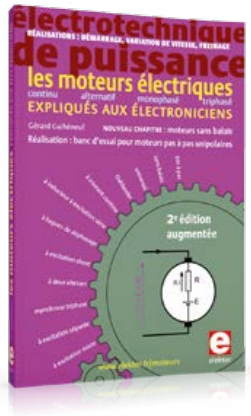
Vos favoris :

1. clavier capacitif à 12 touches
   www.elektor.fr/top1



2. Raspberry Pi Mod. B+
   www.elektor.fr/top2
3. DVD 2014
   www.elektor.fr/top3
4. Atmel SAM D20 Xplained Pro
   www.elektor.fr/top4
5. Maîtrisez les µC à l'aide d'Arduino
   www.elektor.fr/top5
6. BL600 e-BoB
   www.elektor.fr/top6
7. Arduino Uno
   www.elektor.fr/top7
8. L'électronique pour les débutants
   www.elektor.fr/top8

Les moteurs électroniques expliqués aux électroniciens



Les électroniciens découvriront ici les moteurs électriques, tandis que les électrotechniciens admettront qu'électronique n'est pas synonyme de complexité. L'information est digeste : constitution, fonctionnement, caractéristiques, domaines d'utilisation, réalisations simples et concrètes. Le lecteur mesurera ses connaissances grâce à des questionnaires d'évaluation (avec corrigés).

Prix (membres) : 39,50 €

www.elektor.com/moteurs

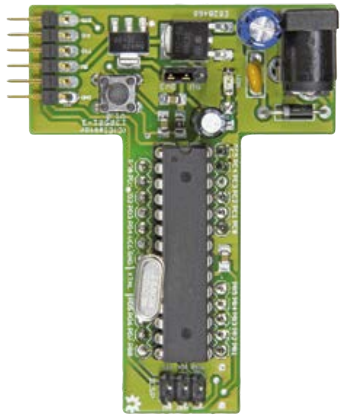
DVD Elektor 2014

Ce DVD-ROM réunit tous les articles d'ELEKTOR parus au cours de l'année 2014, en français (mais aussi en anglais, allemand et néerlandais). Il contient le texte des articles ainsi que les schémas, et tous les dessins des circuits imprimés, sous forme de fichiers à haute résolution. Une fonction de recherche dans la table des matières vous permet de trouver immédiatement l'article souhaité.

Prix (membres) : 24,75 €

www.elektor.fr/dvd-2014

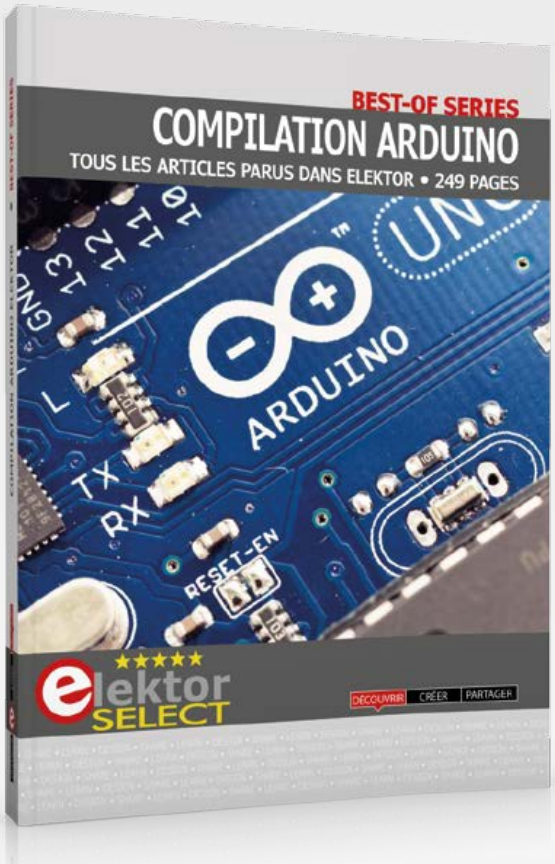
T-Board 28



Les T-Boards, plus flexibles que les modèles Arduino, sont conçus pour un prototypage simple et rapide sur plaque d'essai, tout en réduisant le nombre de fils de raccordement nécessaires. Un cavalier de sélection de tension permet au contrôleur de fonctionner en 5 V ou 3,3 V. Il en existe trois modèles, compatibles chacun avec différents contrôleurs. Ici la version à 28 broches, pour les µC ATmega328, programmable via un câble FTDI. Il permet d'utiliser votre propre quartz.

Prix (membres) : 13,46 €

www.elektor.fr/t-board-28



l'intégrale des articles d'Elektor en format PDF

249 pages bourrées d'idées, d'explications, d'astuces, de schémas, de programmes, de circuits imprimés...

Des heures de lecture instructive et stimulante !

Cette compilation réunit l'intégrale des articles parus dans Elektor entre juillet 2012 et novembre 2014.

Les liens dans la table des matières de ce document numérique (PDF) permettent de naviguer facilement vers les articles qui vous intéressent.

Prix (membres) : 9,50 €

www.elektor.fr/compilation-arduino-e-book

Compilation Arduino

Imagine ce que tu désires, souhaite ce que tu imagines, tu finiras par créer ce que tu veux.
Avec Arduino. Avec Elektor.

interface de diagnostic OBD USB KKL

kit de diagnostic Red Pitaya

Kit *Three fives* – un modèle agrandi du temporisateur 555

Le kit *Three fives* est une réplique exacte et fonctionnelle du temporisateur intégré NE555, l'une des puces les plus populaires de tous les temps. Il ressemble fidèlement à un circuit intégré (agrandi), avec un support en aluminium qui imite les broches du boîtier DIP original à huit broches. Ce kit permet surtout de reproduire, avec des transistors, le fonctionnement du célèbre temporisateur et de l'étudier pour en comprendre la structure interne et d'expérimenter.

Prix (membres) : 35,96 €

www.elektor.fr/trois-555

Personnalisez vos montages Arduino

L'objectif de ce livre est de vous emmener à pas guidés vers la maîtrise d'Arduino. Les projets sont regroupés par thème, avec des bases théoriques. Vous apprendrez à exploiter des techniques essentielles (conversion analogique-numérique, modulation de largeur d'impulsion, pilotage de différents types d'afficheurs, interface I2C, interruptions).

Prix (membres) : 31,05 €

www.elektor.fr/personnalisezArduino

Quadricoptère Crazyflie 2.0

Le Crazyflie 2.0, conçu comme kit sans soudure, est un projet à source ouverte, idéal pour les développeurs. Il tient dans la paume de votre main, mais vous ouvre des horizons infinis grâce à ses fonctions avancées. Ses capacités Bluetooth LE permettent de le piloter à partir d'appareils mobiles. Il est parfait pour l'intérieur, mais ne résistez pas à la tentation de le faire voler au-dessus de votre maison. Il suffit de monter les moteurs sur le châssis et il sera prêt à voler.

Prix (membres) : 159,95 €

www.elektor.fr/crazyflie-kit-integral

MON AVIS SUR CE PRODUIT

par Marcel Člověčko

Vous êtes nouveau dans le monde de l'électronique embarquée ? Et vous vous demandez « *c'est quoi un microcontrôleur et comment ça fonctionne ?* » Si vous avez répondu « oui » deux fois, je vous recommande ce livre. J'en ai acheté la dernière édition et cette lecture m'a beaucoup enrichi. J'avais un peu de connaissance d'Arduino (notamment les puces Atmel AVR), mais il me fallait un livre qui à la fois m'expliquerait le principe depuis le début, et me conduirait beaucoup plus loin. J'ai trouvé cet ouvrage. Son auteur, expert Elektor, donne les bases théoriques universelles qui permettront ensuite de programmer n'importe quel microcontrôleur. Le code proposé permet de manipuler différents composants électroniques : clavier matriciel, afficheurs (à LED, alphanumérique, graphique couleur), moteur, capteurs (température, pression, humidité, infrarouge, son, luminosité), codeur rotatif, ronfleur piézoélectrique, bouton-poussoir, relais.... La mise en pratique des notions abordées est simple et ludique, grâce notamment à une carte d'expérimentation polyvalente disponible séparément ; avec un Arduino, j'ai réalisé un grand nombre de montages du livre. Quelle satisfaction de découvrir ainsi de nouvelles techniques. Grâce à ce livre !

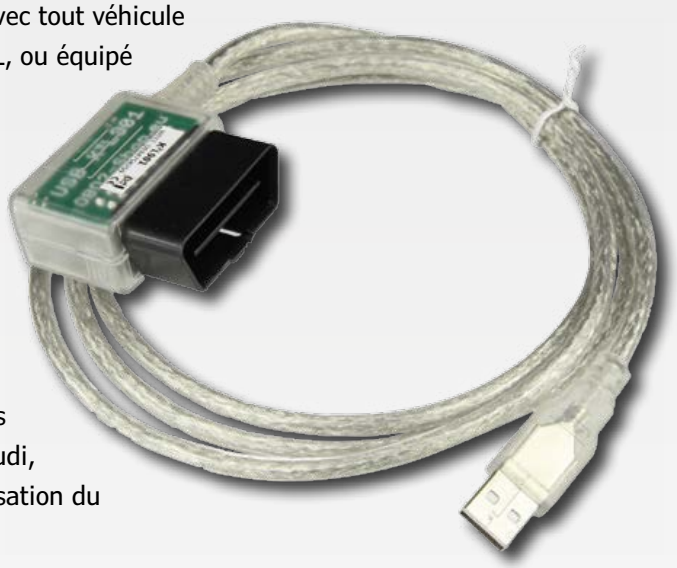
Plus d'infos sur ce livre sur :
www.elektor.fr/maitrisez-arduino

Votre témoignage sur un article vendu dans l'e-choppe d'Elektor peut vous rapporter 100 € sous la forme d'un bon d'achat de produits à valoir sur vos achats dans notre boutique en ligne.

Informations complémentaires et conditions sur www.elektor.fr/rev

Le meilleur diagnostiqueur OBD conçu pour les véhicules du groupe VAG tels que VW, Audi, Seat, Skoda.

L'interface OBD 2 VAG-COM vous permet de connecter votre PC à l'ordinateur de bord de votre voiture et de lire les codes de défauts qui apparaissent lorsque les voyants d'alerte de votre véhicule s'allument. Vous pourrez également faire une remise à zéro, modifier des paramètres, etc... Il est compatible avec tout véhicule utilisant la simple *K line* ou la *K Plus* L, ou équipé d'un connecteur de données OBD 2 à 16 broches, ou d'un connecteur de données 2x2 à 4 fils des marques Volkswagen, Audi, Seat et Skoda, produit entre 1996 et aujourd'hui. Il dispose d'une base de données de plus de 7500 codes d'erreur et peut utiliser le nouveau code à 7 chiffres PIN/SKC pour se synchroniser avec les nouvelles clefs des modèles de VW, Audi, Seat et Skoda avec fonction immobilisation du véhicule.



interface de diagnostic OBD USB KKL

Prix (membres) : 16,16 €
www.elektor.fr/OBD-USB-interface

Compilation Arduino

Imagine ce que tu désires, souhaite ce que tu imagines, tu finiras par créer ce que tu veux.
Avec Arduino. Avec Elektor.

interface de diagnostic Red Pitaya

Les microcontrôleurs PIC pour les débutants

Rémy Mallard initie les débutants à la programmation des PIC au moyen d'exemples pratiques. Il commence par les principes essentiels de programmation, puis regroupe par chapitre les informations nécessaires à la réalisation de chaque exemple. Il fait la part belle aux « petits » contrôleurs à 8 bits, qui disposent entre autres de comparateurs, d'un oscillateur interne, de convertisseurs A/N, de communication à deux fils ou série, et bien davantage.

Prix (membres) : 34,16 €

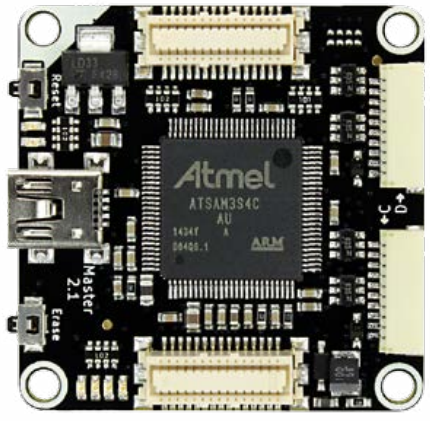
www.elektor.fr/debutpic

offre mystère

oserez-vous ?

www.elektor.fr/mystery

Tinkerforge Master Brick 2.1



La plate-forme Tinkerforge de modules à microcontrôleur empilables comme des briques permet de combiner les différents modules (appelés *Bricklets*) disponibles dans l'e-choppe d'Elektor. Le module maître comporte 4 ports de sortie et se comporte comme si chaque brique de la pile était reliée directement au PC via une connexion USB.

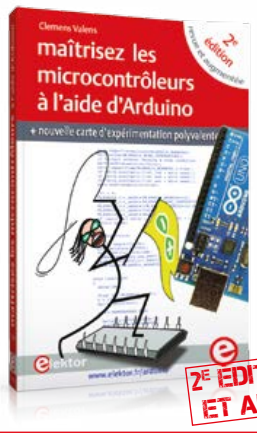
Prix (membres) : 26,96 €

www.elektor.fr/tinkerforge

ma liste d'achats pour ce numéro

- **p.47 - Arduino en esclave I2C**
  - carte ELPB-NG
- **p.48 - capteur I²C sans fil**
  - émetteur prog. 120586-42
  - récepteur prog. 120586-41
  - paire de cartes 120586
  - émetteur-récepteur 130559-91
- **p.52 - interface LCD mono-fil**
  - circuit imprimé 130397-1
- **p.62 - prototypage revisité**
  - carte ELPB-NG
- **p.68 - convertisseur de tension**
  - carte assemblée 130560-91
  - carte nue 130560-1
- **p.72 - e-BoB BL600 - 5e partie**
  - e-BoB FT232 assemblé 110553-91
  - e-BoB BL600 assemblé 140270-91
- **p.78 - sirène de police**
  - circuit imprimé 140481-1
- **p.80 - T-board audio**
  - module complet monté 150002-91
  - circuit imprimé 150002-1
- **p.84 - analyseur MIDI**
  - circuit imprimé 150169-1
- **p.99 - poste de soudage**
  - circuit imprimé du poste 140107-1
  - LCD 120061-74
- **p.126 - wattmètre - BoB ADS1115**
  - carte préassemblée 140169-91
  - circuit imprimé 140169-1
- **p.128 - monde d'Elektor**
  - livre : L'électronique pour les débutants
  - livre : Les PIC pour les débutants

Maîtrisez les microcontrôleurs à l'aide d'Arduino

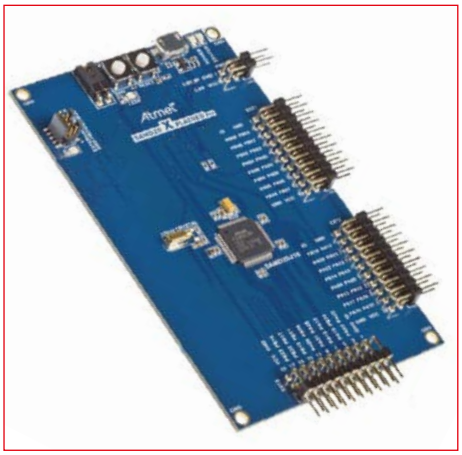


Des montages Arduino inédits, étudiés spécialement pour se débarrasser définitivement de ses amis et de sa famille, et pour se retrouver enfin seul et libre de passer tout son temps à apprendre la programmation des microcontrôleurs ! Le seul livre sérieux de micro-électronique et de micro-informatique dans lequel il soit question d'Arduino mais aussi de Blanche-Neige, de la Cucaracha (en stéréo), de Saint Augustin, de Scarlatti et de Pindare (pas Pandore).

Prix (membres) : 36,85 €

www.elektor.fr/arduino

Carte Atmel SAM D20 Xplained pro

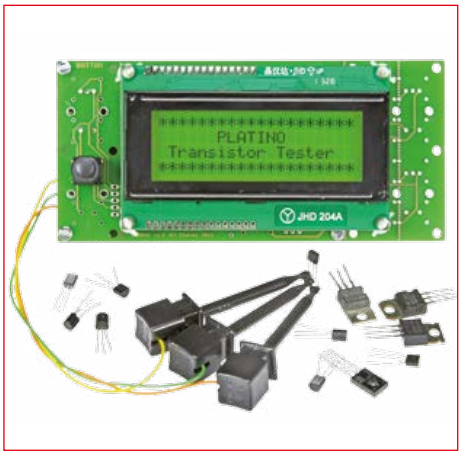


La carte Atmel SAM D20 Xplained Pro est conçue pour le prototypage avec les microcontrôleurs Cortex-SAM D20-M0+. Elle bénéficie d'un riche environnement d'outils pratiques et gratuits. Elle est utilisée notamment dans notre nouveau cours d'initiation aux µC à 32 bits dont le premier épisode a été publié dans le n° de janvier-février 2015, p. 112, et se poursuivra dans chaque numéro tout au long de l'année. Ce module est livré monté et testé. Lancez-vous !

Prix (membres) : 29,96 €

www.elektor.fr/samd20-board

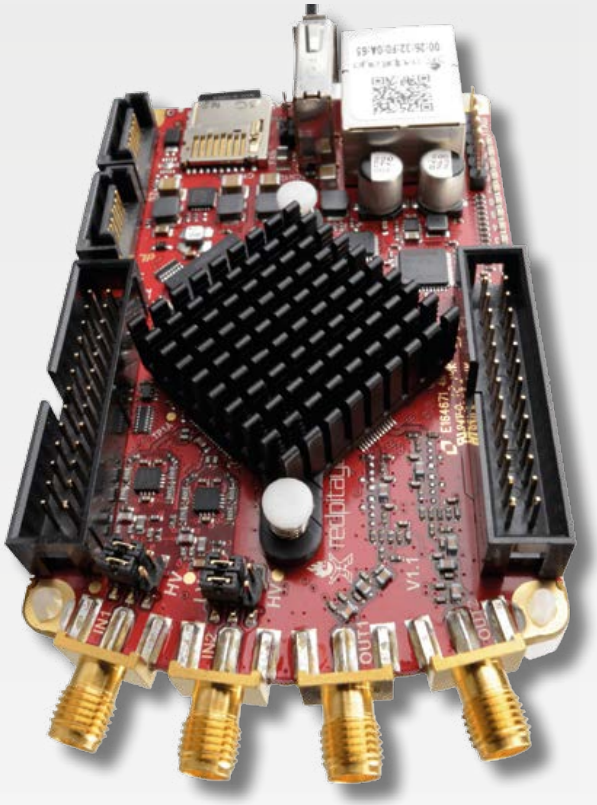
Testeur de transistors avec Platino



Ce testeur permet de trier comme bons ou mauvais les transistors sans référence ou douteux, de déterminer leur type (bipolaires PNP et NPN, MOSFET à canal N ou P) et de mesurer leur gain. L'instrument de mesure repose sur la carte Platino d'Elektor. Sont fournis : circuit imprimé, microcontrôleur programmé, circuit imprimé polyvalent pour AVR-Platino, afficheur et boîtier.

Prix (membres) : 66,95 €

www.elektor.fr/platino-transistortester



Kit Red Pitaya complet à petit prix

Red Pitaya est un outil de mesure et de commande à source ouverte, de la taille d'une carte crédit. À lui seul, il remplace avantageusement de nombreux instruments de laboratoire coûteux, tout en stimulant la créativité de ses utilisateurs. C'est un instrument de mesure de choix aussi bien pour les professionnels et les chercheurs que pour les enseignants et les étudiants.

Le kit de diagnostic Red Pitaya comprend un instrument de mesure à code source ouvert Red Pitaya V1.1, deux sondes d'oscilloscope, deux adaptateurs SMA(M) à BNC(F), un adaptateur d'alimentation micro USB et une carte Secure Digital Kingston MicroSHDC de 4 Go.

Prix (membres) : 283,50 € = Elektor casse la baraque !

www.elektor.fr/red-pitaya-kit

kit de diagnostic Red Pitaya

Compilation Arduino

Imagine ce que tu désires, souhaite ce que tu imagines, tu finiras par créer ce que tu veux.
Avec Arduino. Avec Elektor.

interface de diagnostic OBD USB KKL

bienvenue dans la section PARTAGER

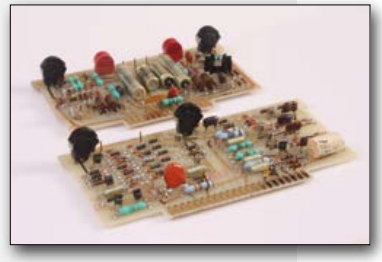
Jaime González-Arintero
jaime.glez.arintero@eimworld.com

while(1){ reinvent(yourself); }

Vous êtes un expert incollable dans un domaine particulier, ou bien un couteau suisse adapté à n'importe quelle situation ? La spécialisation est essentielle en conception, mais des connaissances générales et le goût de l'improvisation sont aussi des atouts dans d'autres domaines, l'ingénierie des services par ex. Je dis toujours que l'électronicien est le plus polyvalent des ingénieurs, car sans s'en rendre compte il acquiert des connaissances en mécanique, optique, chimie, etc. Autoformation permanente et remise en question sont indispensables, car tout va vite de nos jours, mais apprendre de nouvelles choses devient plus difficile avec l'âge. Cette inertie n'est pas propre aux humains : trop rigides, trop spécialisées, certaines grandes entreprises sont incapables de se diversifier ou de réagir à de nouvelles tendances et finissent par disparaître. Comme les humains, les entreprises doivent sans cesse se réinventer pour survivre.

Je suis tombé sur le blog de l'entrepreneur Steve Blank. Il y raconte qu'en 1956 les transmissions de l'armée américaine décidèrent d'acquérir leur premier ordinateur à des fins de recherche [po.st/hpmemo]. Comme à l'époque personne dans les transmissions ne s'y connaissait vraiment en ordinateurs, on demanda conseil à Frederick Terman, membre du comité consultatif et par ailleurs recteur de l'université de Stanford. Terman se tourna à son tour vers William Hewlett, un de ses anciens étudiants qui avec David Packard avait fondé *Hewlett-Packard*. Hewlett répondit : *Je ne m'y connais pas en ordinateur et personne dans notre société n'est compétent dans ce domaine.* Qui l'eût cru !

Au début des années 60, HP était spécialisée dans les appareils de mesure et d'essai, mais bien que l'entreprise fût prospère et que l'idée n'emballait guère Hewlett, Packard souhaitait se lancer dans l'informatique. Il aurait même tenté de racheter la jeune *Digital Equipment Corporation*. Sans succès. Ironie de l'histoire, *DEC* fut rachetée par *Compaq* en 1998, qui fut à son tour rachetée par HP en 2002. Quoi qu'il en soit, HP lança son premier mini-ordinateur en 1996, le 2116A à 16 bits, d'ailleurs plus un contrôleur d'instrument qu'un ordinateur. HP lança deux ans plus tard son premier calculateur de bureau programmable, le 9100A, cette fois-ci c'était un ordinateur de bureau. Après avoir ajouté plusieurs appareils (numériques) à sa gamme de produits, HP devint dans les années 70 le premier fournisseur de technologie au monde et même, jusqu'en 2013, le premier fabricant d'ordinateurs. Plutôt pas mal pour deux types qui disaient ne rien y connaître en informatique, non ?

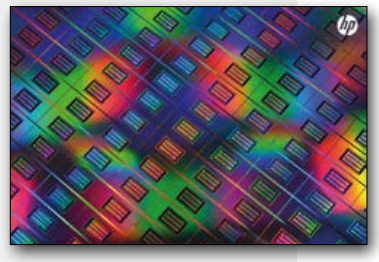


Ce n'est pas la carte-mère d'une station de travail HP actuelle, non, mais deux cartes d'un générateur de fonctions 3300A de 1969. Photo subrepticement empruntée à la rubrique Rétronique de Jan Buiting. +6 dB pour toi, Jan ! ☺

HP a su se réinventer, mais n'a jamais songé à se séparer de ce qui fit son succès, la fabrication d'instruments de mesure. HP transféra cette activité en 1999 à *Agilent*, société qui l'an dernier se scinda à son tour pour donner naissance à *Keysight Technologies*, dorénavant en charge des secteurs électronique et radio. Après quelques turbulences dues à la crise, HP semble à nouveau se réinventer. Voyez ce que nous avons aujourd'hui : les *memristances* ! ⏮

(150279 – version française : Hervé Moreau)

Le 9100B d'HP avec une imprimante. © David G. Hicks, The Museum of HP Calculators [hpmuseum.org]



Memristances d'HP sur une tranche de 300 mm. © 2015 Hewlett-Packard Development Company, L.P.

Il ne suffit pas de s'occuper ; les fourmis en font autant. La question est de savoir : à quoi occupons-nous notre temps ? – Henry David Thoreau

comment câbler ce connecteur ?

des aide-mémoire à la rescousse

Harry Baggen (rédaction des Pays-Bas)

Au fil des ans, de nombreux types de connecteurs – probablement des milliers – ont été développés afin de raccorder toutes sortes d'appareils et de circuits électroniques. Comment peut-on dès lors retrouver les caractéristiques d'un connecteur donné ? Heureusement Internet nous permet aujourd'hui de trouver assez rapidement comment les fils d'un connecteur doivent être câblés, et quels signaux sont présents sur les différentes broches.

Il est recommandé d'avoir quelques adresses de sites sous la main, ou sauvegardées dans les favoris de votre navigateur, vous pourrez y récupérer les données de la plupart des connecteurs. Cela fait gagner du temps, surtout pour les connecteurs exotiques. Un bon site pour débuter est ***The Hardware Book*** [1]. Il se définit lui-même comme « *Internet's largest free collection of connector pinouts and cable descriptions* ». Je n'ai pas compté, mais le site dispose en effet d'une collection respectable. On n'y trouve pas les spécifications les plus récentes, comme celles de l'USB 3.1 ou du HDMI 2.0, mais à part cela vous y dénicherez à peu près tout ce que vous voulez dans le domaine de l'électronique. Il y a des sections séparées pour l'audio, la vidéo, l'informatique, les claviers, les consoles de jeu, les alimentations, les modules d'extension de divers appareils, et d'autres encore. Vous voulez tout savoir sur un *Atari*, un *Amiga* ou un *Commodore* ? Vous les trouverez sur ce site. La mise en page du site n'est pas des plus esthétiques, mais ce qui compte pour nous électroniciens, c'est avant tout l'information. Au fil de mes recherches je suis tombé sur l'une ou l'autre page qui ne fonctionnait pas ou plus, mais cela reste une exception.

Un autre site très pratique, le ***Pinoutsguide*** [2]. Ici aussi une pléthore de prises et de connecteurs. C'est un site russe, et l'adresse originelle était *Pinouts.ru* ; heureusement, toute la documentation y est disponible en anglais et en russe. Nous y trouvons bien entendu de nombreux connecteurs et câbles liés à l'informatique, mais aussi pour les jeux vidéo, les alimentations secourues, les *smartphones*, les récepteurs GPS, les appareils audio et vidéo, les caméras, et les interfaces de test pour l'automobile (OBD-2). En sus du brochage, on y retrouve aussi pour de nombreux connecteurs les signaux et la norme concernée. Le site est une vraie mine d'informations pour l'utilisateur. Et si votre connecteur n'est pas encore sur le site, vous l'ajouterez au moyen d'un champ ad hoc.

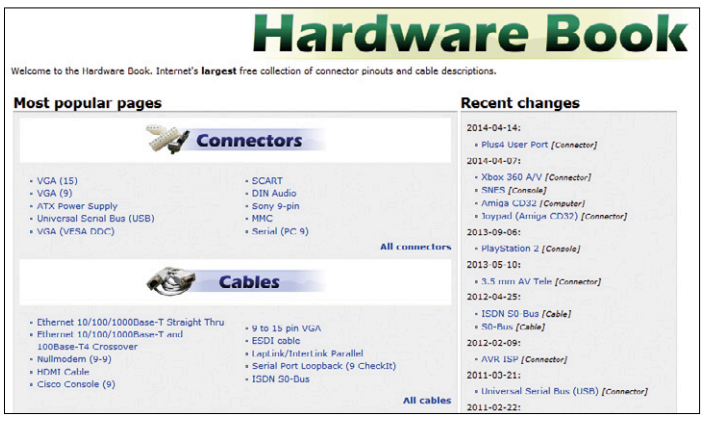
Le site ***Allpinouts*** [3] aurait pu figurer à la première place de cet article, vu son énorme catalogue de connecteurs, mais l'interface sommaire du site ne facilite pas la recherche. Il s'agit d'un projet communautaire, au contenu disponible librement (aux termes de la licence GNU de documentation libre, *GNU FDL*), et où les utilisateurs eux-mêmes assemblent l'information. Le site est en fait conçu sous la forme d'un *wiki*, ce qui a pour conséquence qu'il faut parcourir une arborescence plus ou moins longue avant d'arriver à l'objet convoité. En outre, il y a diverses publicités ; elles sont parfois mal placées et rendent de temps à autre la poursuite de la recherche tout bonnement impossible. Mais c'est peut-être le prix à payer pour trouver le connecteur qui n'est pas sur les autres sites...

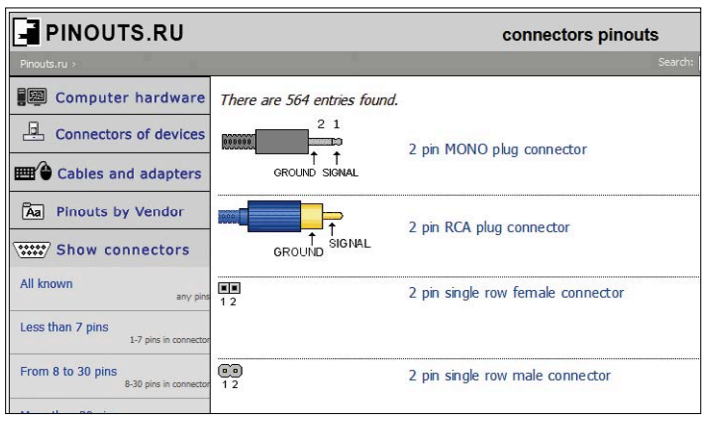
Avec ces trois liens dans vos favoris, vous êtes armé pour dénicher les caractéristiques d'un connecteur ou câble donné.

(150290 – version française : Jean-Louis Mehren)

Liens

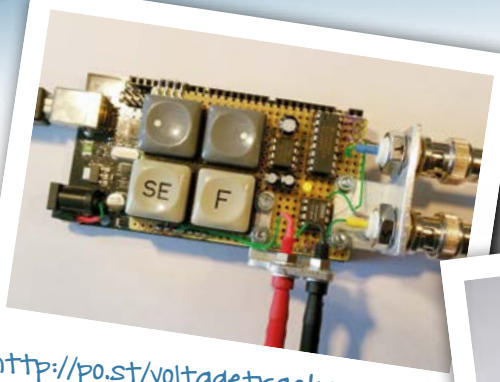
[1] www.hardwarebook.info
[2] http://pinoutsguide.com ou http://pinouts.ru
[3] www.allpinouts.org





.LABorama

S'adapter pour survivre, cela vaut aussi en électronique, la preuve sur elektor-labs.com avec ce recyclage d'un vieil adaptateur de téléphone. Parmi les autres projets mis ici en vedette, un oscilloscope capable de visualiser des signaux très lents, un barographe, une commande d'interrupteur par l'internet…



http://po.st/voltagetracker

Pas de précipitation

La plupart des signaux électriques avec lesquels nous travaillons sont si rapides qu'il est impossible de les interpréter sans l'aide d'instruments. D'autres événements sont en revanche si lents que ces mêmes instruments sont bien incapables de les capter. Basé sur Arduino Mega, cet enregistreur de tension pour oscilloscopes vous permet de surveiller l'évolution de processus électriques très lents (comme la charge ou la décharge des batteries). Le temps de suivi atteint douze heures. Détendez-vous et allez-y doucement…

Croyez-vous en la réincarnation ?

Nous oui ! En tout cas en celle des vieux boîtiers. Redonner vie à des appareils est rentable, et en plus c'est utile et amusant. Ce circuit simple tient dans le boîtier d'un vieil adaptateur de téléphone et peut servir d'éclairage nocturne pour des pièces ou autres entrées. Et il est alimenté par une prise secteur, donc pas besoin de sortir la perceuse pour le monter contre un mur.

http://po.st/nightlight

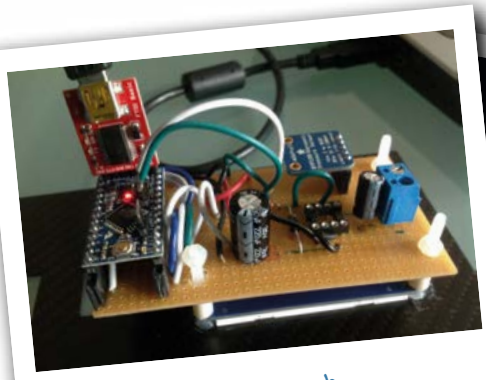
Il y a quelque chose qui cloche

Le condensateur à capacité variable est souvent utilisé dans les circuits oscillants des montages radio. Il est composé de deux armatures : l'une mobile (rotor) et l'autre fixe (stator). Un axe relié au rotor permet de faire varier sa capacité. Nos traducteurs de la rubrique Rétronique rencontrent parfois aussi le terme « beehive trimmer » dans la description de récepteurs radio. Il se caractérise par la disposition concentrique de ses deux armatures imbriquées l'une dans l'autre. L'un de nos lecteurs assidus nous a rappelé que ce composant est connu en français sous le nom de « condensateur cloche »

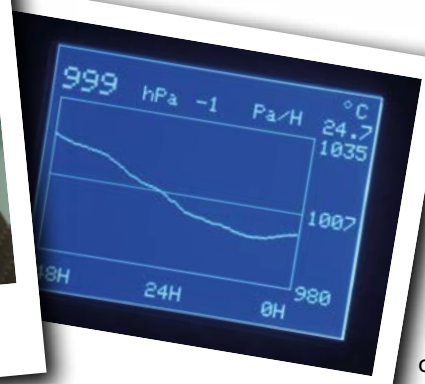
Loin des yeux, loin de l'interrupteur

Commander à distance un interrupteur depuis un navigateur web : bienvenue dans le futur ! O-k, le projet débute, mais le mettre en œuvre est simple et le développer devrait être un jeu d'enfant. Cet interrupteur distant utilise une Seeeduino Ethernet pour établir une connexion Internet, et affiche sur un écran OLED l'adresse IP correspondante. Lorsqu'on tape cette adresse dans un navigateur, une « prise » verte ou rouge s'affiche et il est possible de modifier l'état d'un relais depuis n'importe où dans le monde. Les dessins sont créés à partir d'un fichier SVG, autrement dit tracés par le navigateur et donc indépendant de la plate-forme utilisée.

http://po.st/remoteswitch



http://po.st/barograph

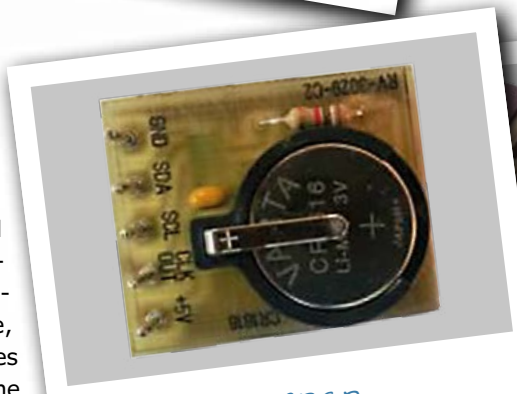


Combien de capitaines…

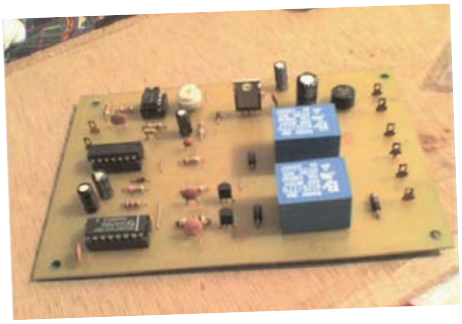
… sont partis joyeux pour des courses lointaines sans leur barographe maison ! Les marins doivent surveiller la pression atmosphérique pour éviter les épines de la rose des vents. Ce barographe à Arduino trace sur un écran TFT l'évolution de la pression sur une période de 48 h et enregistre les données sur une carte SD. Une LED et un *buzzer* s'activent lorsque la pression chute sous un certain seuil. Le capteur de pression peut aussi indiquer la température.

T'as pas l'heure, BoB ?

Ajouter une horloge en temps réel (RTC) à un circuit est on ne peut plus simple, le marché abonde de ce type de circuit intégré. Parmi eux, le RV-3029-C2 offre une précision exceptionnelle et une connexion I²C, mais sa petitesse le rend difficile à souder. Appelons BoB à la rescousse ! Cette petite carte de liaison comprend le circuit intégré RTC sur une face, et sur l'autre la pile CR2032 et le reste des composants. Bref elle facilite l'ajout d'une horloge très précise à un projet.



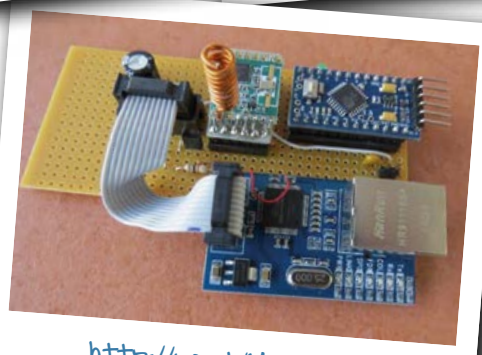
http://po.st/RTCBOB



http://po.st/clapactivatedlamp

Ma sorcière bien-éclairée

Allumer une lumière d'un claquement de doigt, déjà ça ressemble à de la sorcellerie. Mais il y a plus fort : nouveau claquement de doigt, et la luminosité augmente. Troisième claquement, la lumière s'éteint ! Et cette commande sonore pour lampe de chevet fonctionne sans microcontrôleur (ni baguette).



http://po.st/MQTTgateway

Encore une passerelle pour l'IdO

Une de plus, oui, mais ce projet n'en est pas moins intéressant ! Le module principal joue le rôle de concentrateur radio et communique avec les nœuds du réseau via le protocole MQTT. Les nœuds terminaux reposent sur le populaire module RFM69. Comme la communication se fait en duplex, chacun d'entre eux accuse réception à la passerelle de chaque message reçu. L'ensemble du projet peut être téléchargé, vous pourrez donc commander sur le champ vos objets en réseau !

(150280 – version française : Hervé Moreau)

Made in Germany

Rohde & Schwarz, Wandel & Goltermann et *Kompanie*

Jan Buiting, *Grand Manitou ès-Antiquités*

Certains lecteurs allemands m'ont gentiment reproché de ne pas suffisamment couvrir les réalisations de leur pays. Rendant à Cäsar ce qui est à César, je décris ici quelques instruments de laboratoire haut de gamme qui ont sans doute aidé de nombreux étudiants allemands à faire carrière dans l'électronique.

Les lecteurs fidèles se souviennent peut-être de l'alimentation NE-171 de Wandel & Goltermann détaillée dans le numéro de juin 2013, une alimentation HT réglable équipée d'une stabilisation électronique et de l'adorable EL156. Elle faisait partie d'un des appareils de mesure et d'essai des années 60 et 70 que la fac située près de la maison Elektor utilisait pour ses cours. De « Matériel pédagogique », ces appareils avaient un jour été étiquetés « Bons pour la casse » et j'avais pu les sauver du rebut grâce à un collaborateur d'Elektor. J'étais aux anges ce jour-là, car un grand nombre d'entre eux étaient à tubes. La plupart, rien de surprenant à ça, étaient d'origine allemande et provenaient de fabricants prestigieux : Siemens, Wandel & Goltermann, et Rohde & Schwarz. Au cas improbable où ces noms ne vous diraient rien, pensez : BMW, Audi, Mercedes Benz, voire Kraftwerk ou Bayern de Munich !

Sortez vos générateurs et prenez une feuille

Quelque part entre les années 1970 et 1985, les filtres étaient au programme des premiers cycles universitaires : Bessel, Butterworth, coupe-bande et autre facteur Q. Le cours était à la fois théorique et pratique. Si j'en juge par le matériel pédagogique que j'ai récupéré, un binôme d'étudiants devait certainement analyser et enregistrer la réponse d'un filtre *XYZ*. Par contre j'ignore si le filtre en question était attribué par l'enseignant ou conçu par les étudiants, ni qui le construisait. Chaque binôme recevait un appareillage de paillasse, au moins un générateur d'ondes sinusoïdales précis avec plage de fréquences étendue, un millivoltmètre et quelques câbles, le tout à monter par les étudiants et bien sûr calibré. Sans oublier le papier millimétré, les crayons, et éventuellement une règle à calcul. Coïncidence, en ce vendredi après-midi pluvieux où j'entassai dans le coffre de ma voiture les appareils récupérés, j'avais remarqué, esseulés dans la salle d'une université presque déserte, deux pauvres étudiants en train de fixer la réponse d'un

ESTD 2004

www.elektor.tv

Rétronique est une rubrique mensuelle sur les pages glorieuses et jaunies de l'électronique, avec occasionnellement des montages de légende décrits dans Elektor. Si vous avez des suggestions de sujets à traiter, merci de les télégraphier à redaction@elektor.fr

filtre exotique sur l'écran de leur portable ; j'avais noté qu'ils tentaient d'en retrouver le schéma dans un épais manuel, et surtout que leur professeur venait de quitter la salle avec une mine réjouie…
Les appareils gracieusement offerts à la rubrique Rétronique comptaient deux « kits pédagogiques » pour l'étude de la réponse des filtres. Ils comprenaient à l'origine au moins les instruments (plus ou moins portables) suivants :

- un générateur RC SRB de Rohde & Schwarz, 10 Hz – 1 MHz, 1 mV – 30 V, **ou** un générateur RC MG-47 de Wandel & Goltermann, 30 Hz – 300 kHz, 1 mV – 100 V ;
- un voltmètre CA ESM-1 à valeur efficace de Wandel & Goltermann, 30 Hz – 15 kHz, 3 mV – 30 V_{eff} (–50 à +30 dB).

Tous les instruments n'étaient pas rigoureusement identiques. Le MG-47 de W&G avait p. ex. un sélecteur de fréquence légèrement différent.
L'utilisation de générateurs de 300 kHz et 1 MHz semble indiquer que les techniques de filtrage enseignées allaient au-delà des fréquences audio, une bonne chose en soi.
Aucun instrument ne portait de traces de nicotine et tous paraissaient avoir été bien entretenus. Les faces avant étaient couvertes d'autocollants difficiles à retirer, robustesse allemande oblige.
La faculté avait réussi à déterrer des archives la plupart des manuels d'origine. J'avais pu les copier en échange de DVD Elektor couvrant les années 1980 à 2010.
La fonction des principaux boutons est indiquée en allemand et en anglais sur les appareils R&S et W&G, à l'exception de la séparation décimale, toujours représentée par une virgule et non par le point anglo-saxon.

Générateur R-C SRB de Rodhe & Schwarz

Le gris pâle des appareils de mesure construits par R&S dans les années 60 et 70 se repère à un kilomètre. Les instruments plus petits sont équipés d'une poignée très solide et peuvent être empilés en toute confiance puisque le haut de leur coffret est pourvu de quatre trous assurant la stabilité des pieds de l'instrument du dessus. Tous les R&S sont identifiés par trois lettres (SRB, URV, etc.) imprimées de façon visible sur un panneau réfléchissant (!) placé sous leur poignée (**fig. 1**). Avec ça les étudiants se familiarisaient très tôt au jargon des abréviations.

Figure 1. Trois lettres pour identifier l'instrument ? C'est un Rohde & Schwarz !

Figure 3. Il suffit de retirer quatre vis pour sortir le SRB hors de son boîtier. Visiblement une construction ordonnée et compacte, non ?

Un instrument petit et élégant comme le SRB (**fig. 2**) est assez léger pour que les étudiants puissent le porter à travers les couloirs de la fac sans risquer d'avoir un bras plus long que l'autre. Son *design* et son ergonomie semblent un compromis entre une utilisation en labo et sur le terrain. Il lui manque son capot protecteur à pression et son logo R&S, mais rien d'étonnant à cela, ces capots sont aujourd'hui prisés des collectionneurs.
Deux des SRB en ma possession ont un seul défaut : parfois leur cordon d'alimentation se rompt, ou bien la gaine de leur prise durcit. Tous les SRB ont parfaitement fonctionné après une petite remise à neuf.
J'ai testé la distorsion harmonique dans la plage audio avec un analyseur *Audio Precision* et trouvé –75 dB en moyenne. Sur un des SRB, la fréquence de sortie de la plage supérieure (1 MHz) a bien demandé un léger recalibrage, mais une erreur d'environ 2 % n'a rien de choquant. Ces réglages se font par l'intermédiaire de condensateurs ajustables à vernier de 30 pF (des *beehive* de Philips, appelés aussi condensateurs cloches). L'oscillateur est un circuit RC qui se règle avec un condensateur d'accord en boîtier et un potentiomètre. La stabilisation de son amplitude est lente mais efficace, la régulation étant de 0,2 dB sur la plage de fréquence.
Plusieurs des caractéristiques du SRB sont remarquables. Parmi elles un bouton de réglage de la fréquence à la fois ferme et sensible, la présence de cinq impédances

Figure 2. Le générateur RC SRB de Rohde & Schwarz et son sélecteur de fréquence si agréable à manier.

Figure 4. Le générateur MG-47 de Wandel & Goltermann est le pendant du SRB de R&S, mais sa gamme de fréquences est plus étroite et il possède plus de sorties symétriques, dissymétriques et isolées (transformateur).

de sortie (50/60/75/150/600 Ω), une tension de sortie élevée (jusqu'à 30 V_{eff}) et, plus amusant, l'inébranlable utilisation de « db » sur la face avant au lieu de « dB ». Outre l'habituelle lecture en Hz/kHz avec une échelle de 1 à 10, l'aiguille du cadran fournit également sur un plus petit cercle une indication en « lg f » que je suppose être une représentation logarithmique — très utile aux étudiants pour le tracé des filtres.
L'intérieur du SRB (**fig. 3**) révèle les classiques condensateurs électrolytiques HT ainsi que les tubes conventionnels de la série Miniwatt EL/E(C)C de Philips, ceux des radios de l'époque ; j'ai vu deux EL86, deux ECC81, et un E88CC, tous de Siemens et dûment estampillés *MADE IN GERMANY*.

L'alimentation est à semi-conducteurs et à diodes. Une rapide mesure de l'ondulation sur les lignes HT de l'instrument m'a convaincu que les « bidons » électrolytiques étaient o-k et assuraient encore leur rôle. J'ai pu éveiller en douceur tous les instruments à l'aide de mon Variac.
La fac avait remplacé l'excentrique sortie d'origine des SRB par un connecteur BNC. Comme le connecteur Dezifix, cette sortie propre à R&S et à l'Allemagne est un vrai cauchemar lorsqu'il s'agit d'en trouver des adaptateurs ou des fiches. L'adaptateur en laiton du SRB a très probablement été fabriqué sur un tour de l'atelier mécanique de la fac (qu'on ne m'a pas laissé visiter).
Je n'ai pas désossé entièrement le SRB, donc je ne peux que supposer que le sélecteur de fréquence repose sur des roulements à billes et autre mécanique de précision.
J'ai utilisé un des SRB pour faire une surprise à l'équipe de Rohde & Schwarz lors du salon *electronica* 2014 de Munich [1], et je l'ai également présenté à de jeunes étudiants [2].

Figure 5. Vue intérieure du « Wago » MG-47.

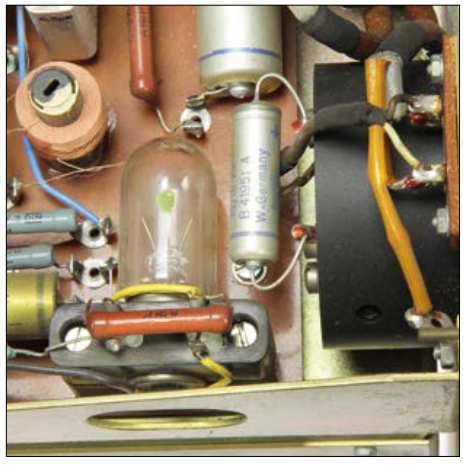


Figure 6. Le tube que vous voyez ici est décrit par W&G comme étant un *Kaltleiter* ; il sert à stabiliser l'amplitude de l'oscillateur central.

Générateur R-C MG-47 de Wandel & Goltermann

Beaucoup plus encombrant et lourd que le SRB, ce générateur (**fig. 4**) n'a sans doute pas été conçu pour les mesures de terrain. Mon modèle est un BN92-2, avec un voltmètre et une sortie de transformateur de 60 Ω. Le dessus est recouvert d'une laque gris-bleu tandis que la face avant, que j'ai nettoyée avec de la cire liquide pour auto, montre un gris clair. Pas de cadran rond ici, mais deux petites fenêtres affichant des graduations à défilement pour six plages de fréquences. Le MG-47 diffère aussi du SRB par ses sorties symétriques et un plus grand choix d'impédances. Le vumètre, un Spartan, ne paie pas de mine avec seulement deux plages de 0-11 et 0-40… quoi ? Pas une seule unité en vue, même si à l'évidence nous sommes en présence de volts (V) ou de millivolts (mV), pas de kilos (kg). Les boutons sont plus costauds que ceux du SRB mais d'un maniement très facile. L'entrée du cordon d'alimentation est judicieusement scellée par un disque en plastique, et une prise CEI a été ajoutée sur la face arrière. Ici aussi la prise de sortie a été remplacée par un connecteur BNC à rebord rond.
Sans trop de surprise, l'intérieur de l'instrument (**fig. 5** et **6**) dévoile des tubes d'époque (EL/ECC/EF à nouveau) ainsi qu'une lampe à grosse ampoule que je soupçonne être la stabilisation d'amplitude mentionnée dans la doc (*Kaltleiter*, thermistance CTP).
Deux des trois MG-47 fonctionnaient normalement. Le troisième, sans doute en raison d'une défaillance de la HT, produisait des crépitements troublants qui lui ont valu d'être déclaré *Ersatzteilträger* (pour pièces).
Les performances des appareils en état sont remarquables : même après 25 ans de placard, leurs spécifications égalent voire dépassent celles du manuel.
Le collectionneur de « Wago » (Wandel & Goltermann) le plus zélé que je connaisse est Max Koschuh, créateur d'un musée en ligne célébrant ces instruments [3].

Voltmètre audio ESM-1 de Wandel & Goltermann

Un voltmètre de précision est indispensable pour analyser et enregistrer la réponse d'un filtre, et pour ça les étudiants qui suivaient le cours « Comprenez vos filtres » disposaient d'un ESM-1 de W&G (**fig. 7**).

L'instrument ressemble au MG-47, mais sa face avant ne porte que des indications en allemand. Il s'agit aussi d'un appareil de table qui peut être empilé grâce à des dispositifs de verrouillage placés sur le haut et le bas des coffrets. Son large cadran peut être incliné de 45° et verrouillé dans cette position. Le cadre mobile de 60 µA de type 137-8101 est un Gossen qui affiche le même « db » que le générateur R&S (quelqu'un en connaît la raison ?) J'ai eu la chance de trouver deux alimentations de table de Gossen parmi les appareils récupérés. Leurs cadrans sont vraiment élégants.

L'ESM-1 en ma possession est comme neuf, mais les contacts de son interrupteur sont mauvais. Même si j'actionne délicatement le levier de l'interrupteur pour le mettre sur la position ON (*Ein*), l'aiguille du cadran se met à trembler, ce qui n'est pas bon. Je suis tout de même parvenu à stabiliser et à faire fonctionner correctement l'instrument grâce à la position *Eichen* (calibrage) du sélecteur de gamme. Comme attendu, l'aiguille a atteint la déviation maximale de 10 V après que j'ai tourné doucement le bouton de calibrage. Aucun truc à microcontrôleur ne rendra jamais la sensation que l'on éprouve devant une aiguille qui atteint une graduation.

La sortie du signal a une impédance (dissymétrique) d'environ 15 kΩ, de quoi utiliser aussi l'ESM-1 comme amplificateur de mesure, et peut-être pour piloter un traceur. Un bouton permet de choisir entre les modes « Instrument » (de mesure) et « Sortie ».

En ouvrant l'ESM je suis tombé sur quelque chose de captivant (**fig. 8** et **9**) : partout des tubes à vide C3m ! Certains disent que le tube fabriqué par Lorenz, Valvo, Siemens et Telefunken est la meilleure pentode de faible puissance jamais fabriquée. Si la construction de l'ESM-1 est classique, c'est aussi une merveille de savoir-faire. Exception faite des tubes C3m, l'ESM-1 ne recourt à aucun composant spécial, et toute son électronique s'achète encore aujourd'hui, même si la question ne se pose pas.

Figure 7. Le voltmètre audio Wandel & Goltermann ESM-1 de plage 30 Hz à 15 kHz, photographié ici avec le cadran incliné.

Und viele Andere !

Oui, beaucoup d'autres instruments récupérés ce jour-là, bien plus que je n'en pourrai décrire ici, et bien trop pour le coffre de ma voiture. Certains servaient pour les TP consacrés à l'étude des ondes centimétriques ou de la distorsion audio, même si je doute que la complexité du sujet les ait destinés à des étudiants de premier cycle. Tous ces instruments provenaient bien sûr de grandes firmes allemandes : Wandel & Goltermann, Siemens, et Rohde & Schwarz.

J'espère vous décrire prochainement les plus exotiques d'entre eux, et trouver le temps de tourner une courte vidéo sur ceux mis en vedette ici (cherchez *Retronics* sur www.elektor.tv).

(150232 – version française : Hervé Moreau)

Liens

[1] Interview Rohde & Schwarz : https://youtu.be/AwwdW-mPYRk

[2] Rétronique au salon electronica 2014 : https://youtu.be/Yrqf3YLZAWY

[3] Musée en ligne Wandel & Goltermann de Max Koschuh : www.koschuh.com/museum/wandel_goltermann/wago.htm

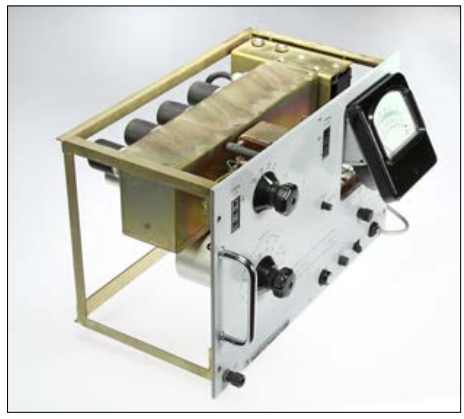


Figure 8. La conception électronique de l'ESM-1 est traditionnelle, mais avec un savoir-faire W&G qui à l'époque en faisait un instrument de premier ordre.

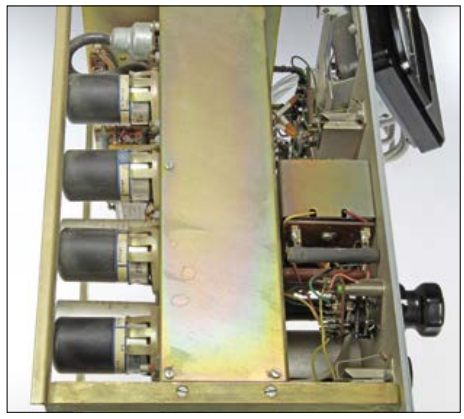


Figure 9. Une bonne surprise : des tubes à vide C3m à l'intérieur de l'ESM-1. Produit *Made in Germany* par excellence, le C3m est aujourd'hui un *must* de l'audio haut de gamme.

wattmètre avec BoB ADS1115

à la recherche du code

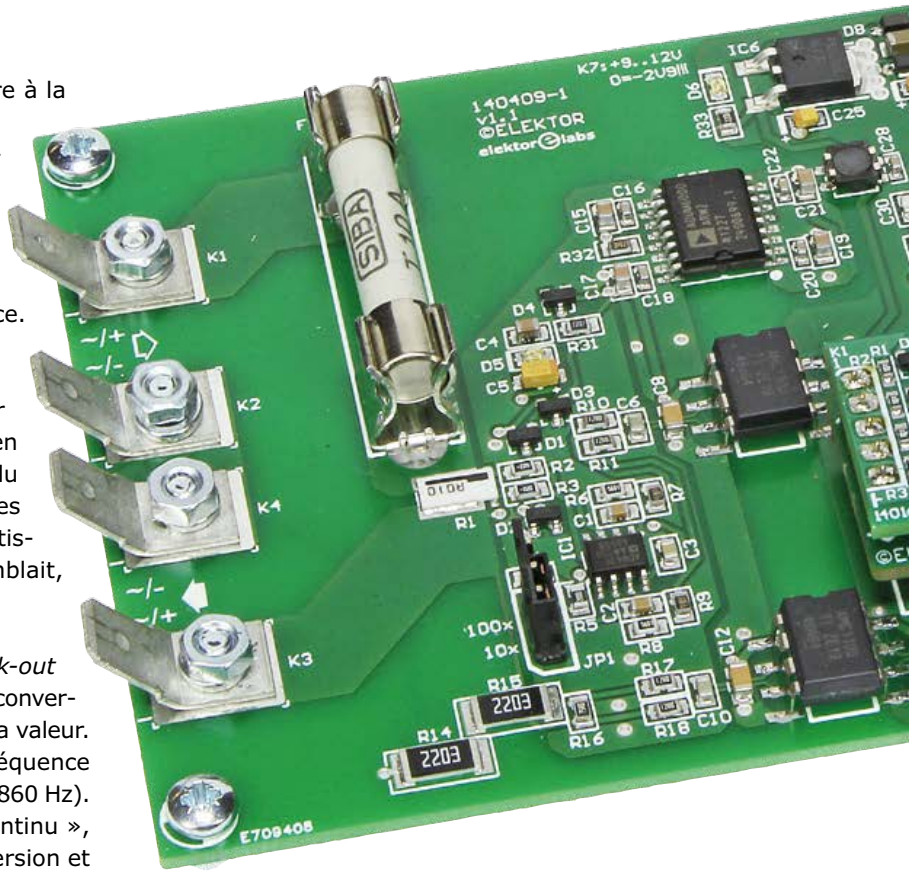
Luc Lemmens (Elektor Labs)

Pendant l'étude de la carte de liaison BoB ADS1115 (4 voies de conversion A/N à 16 bits sur I²C [1]), nous nous demandions à quoi cette carte pourrait bien servir. Ce circuit intégré de Texas Instruments est certes versatile, mais avec une fréquence d'échantillonnage maximale de 860 Hz, il n'est pas bien rapide. Comme ça suffit pour des mesures sur le réseau électrique, nous en avons fait un wattmètre présenté ici.

Il fallait ajouter à l'ADS1115 l'électronique nécessaire à la mesure – en toute sécurité ! – des tensions et courants à haute tension. C'est mon collègue Ton, spécialiste des projets analogiques de notre labo, qui s'en chargerait. La suite devait être triviale, avec un microcontrôleur qui traiterait les échantillons de tension et de courant qui nous donneront les valeurs de tension, courant, puissance et facteur de puissance.

Après construction et test du prototype du wattmètre (« Power Meter » sur le site du labo), ce fut mon tour de m'occuper de la partie numérique et du logiciel. Rien de sorcier : il suffisait d'y brancher un Arduino ! Lors du développement de la carte BoB, une bibliothèque et des exemples avaient été écrits pour raccorder le convertisseur A/N au bus I²C. C'est du moins ce qu'il me semblait, et je me mis au travail.

Dans l'exemple de code pour la carte de liaison (*break-out board*), on attendait assez longtemps pour effectuer la conversion A/N de chaque échantillon, suivie de la lecture de la valeur. En fait, trop longtemps pour pouvoir profiter de la fréquence d'échantillonnage maximale (même si elle n'est que de 860 Hz). L'ADS1115 devait aussi être configuré en « mode continu », où sa broche RDY informe l'Arduino de la fin de conversion et de la disponibilité de l'échantillon suivant. Il s'agissait donc de bien sélectionner les bits correspondants dans les registres de configuration. Toujours rien de sorcier, mais il est apparu que sans exemple concret ni note d'application, la feuille de caractéristiques de Texas Instruments n'est ni complète, ni claire. Certains fabricants font visiblement des économies sur leur documentation ! Les plus petits circuits intégrés seraient-ils devenus si complexes que plus personne n'en connait les subtilités ? Nous sommes de plus en plus souvent réduits à devoir tâtonner et expérimenter pour découvrir le fonctionnement de ces nouveaux circuits, sans vraiment savoir...

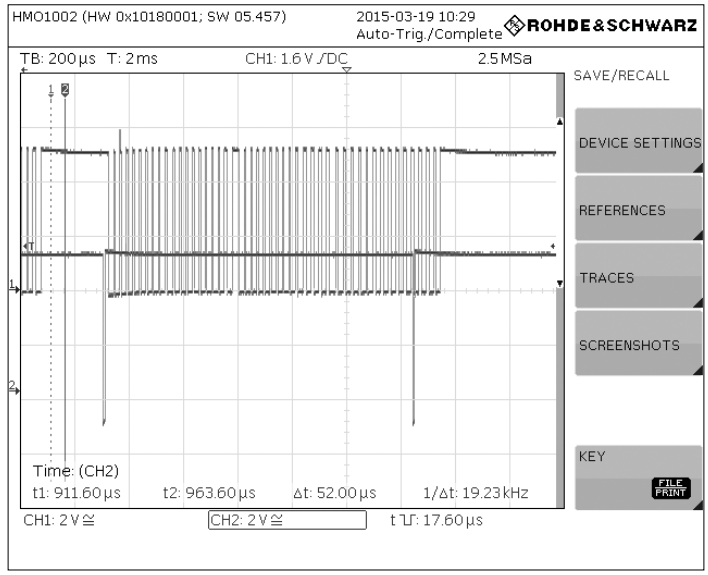


Dans l'exemple de code figuraient aussi les fonctions de gestion du bus I²C, qui n'utilisent pas la bibliothèque standard Wire de l'Arduino. C'était voulu, afin de pouvoir utiliser le programme dans d'autres environnements. Hélas ! Dans cette configuration-ci, cela ne donnait pas les résultats escomptés : le bus I²C devenait si lent qu'un échantillon n'était pas encore lu que le suivant était déjà prêt. Cela signifiait que la fréquence d'échantillonnage était divisée par deux, encore assez d'après Shannon, mais pas vraiment... L'utilisation de la bibliothèque Wire d'Arduino a résolu le problème de lenteur du bus I²C.

Une autre pierre d'achoppement était la mesure simultanée de la tension et du courant nécessaire pour déterminer la puissance d'un signal alternatif. Nous avions d'abord pensé à basculer l'entrée du convertisseur A/N entre tension et courant (et vice-versa) après chaque échantillonnage, et à soit accepter l'erreur sur la mesure de puissance qui en résulte, soit retarder un des deux échantillons pour une mesure correcte. La fréquence d'échantillonnage est alors divisée par deux, mais cela ne suffit pas car le basculement de l'entrée du convertisseur prend aussi du temps. Il faudrait donc sauter encore une période d'échantillonnage, ce qui ne nous laisserait qu'un quart des 860 échantillons par seconde.

> Pourquoi certains fabricants font-ils des économies sur la documentation de leurs circuits intégrés ?

La meilleure solution était de prendre une série de mesures de la tension, puis une autre du courant. Si tension et courant sont (raisonnablement) stables, les mesures de puissance devraient être assez fiables. Il reste alors à déduire de ces mesures la relation de phase entre les deux signaux, ce à quoi nous n'avions pas encore réfléchi. Nous aurions sans doute pu trouver une solution logicielle au problème, mais pour ne pas nous approcher des limites de puissance de calcul de l'Arduino – et risquer de tomber sur de nouveaux écueils liés à la synchronisation – nous avons ajouté un comparateur au circuit, dont la sortie bascule lors de chaque passage par zéro de la tension (du positif vers le négatif). Ce signal est utilisé pour déclencher une série de mesures de la tension et, lorsque celles-ci sont terminées, des mesures du courant synchronisées sur celles de la tension. Après ces deux séries de mesures, l'Arduino traite les valeurs recueillies et délivre les résultats sur l'afficheur LCD, après quoi recommence un cycle de mesures.



Pour une mesure en courant continu – que nous voulons bien sûr aussi effectuer – le passage par zéro n'existe plus, exit le signal de synchronisation ! En l'absence de détection de flanc, une temporisation dans le logiciel passe automatiquement de l'alternatif au continu.

Finalement, c'est assez simple, surtout après coup !

On a beau chercher à éviter de modifier le matériel, mais parfois on n'y échappe pas.

(150287 – version française : Jean-Louis Mehren)

Liens

[1] Elektor n°439/440, janvier/février 2015

[2] Page du projet BoB ADS1115 : www.elektor-labs.com/project/practical-4-channel-adc-140169.14145.html

[3] Page du projet Power Meter : www.elektor-labs.com/project/140409-power-meter.14409.html

Compilées par **Beatriz Sousa**

Lancement de la campagne d'été

Elektor lance le 21 juin sa traditionnelle campagne de promotion d'été, avec une marée de nouveaux produits, cette année sous la houlette de Ferdinand te Walvaart, directeur des ventes, de Muhammed Söküt, chargé des abonnés, et de Margriet Debeij, chargée de la clientèle et des fournisseurs. Si lors de vos pérégrinations sur l'internet vous dénichez une carte électronique sensationnelle, un outil incontournable pour l'électronicien, un kit fascinant dont vous pensez qu'il a sa place dans l'e-choppe d'Elektor, faites-le nous savoir (redaction@elektor.fr) et, si nous faisons affaire avec le fournisseur, nous vous l'offrirons (valeur du produit limitée à 200 €).

Rénovation du site du magazine Elektor

Les activités estivales d'Elektor ne se limitent pas aux campagnes promotionnelles. Nous venons de lancer une version rénovée et améliorée du site du magazine à l'adresse **www.elektormagazine.fr**. Elle regroupe sous un même toit tout le contenu Elektor. Les membres GREEN et GOLD ont accès à l'immense archive numérique regroupant les documents et magazines publiés par Elektor, les non-membres aux nouvelles et à d'autres publications en ligne. Vous trouverez rapidement les articles qui vous intéressent grâce au moteur de recherche amélioré et à ses critères de sélection : par sujet, par auteur, etc.

READ ONLY MEMORY

Le magazine Elektor et son éditeur sont fiers de leur histoire. Cette rubrique montre pourquoi.

lecteur de CD maison

Elektor a publié en janvier 1992 le premier lecteur de CD à monter soi-même. Le disque compact était déjà répandu à cette époque, mais les bons lecteurs de CD restaient chers. C'est grâce à une étroite collaboration avec Philips, un des principaux fabricants de lecteurs de CD, qu'Elektor avait pu acquérir puis revendre à prix « sortie d'usine » un grand nombre de lecteurs ainsi que les cartes de commande associées. En publiant une description claire et détaillée du premier lecteur de CD à monter soi-même, Elektor permit à de nombreux lecteurs de découvrir cette nouvelle technologie.

« Tueriez-vous plutôt » demande

Les électroniciens sont de plus en plus souvent impliqués dans le débat entre pessimisme (*La technologie déshumanise*) et optimisme (*La technologie améliore nos vies*). Nous nous interrogeons sur les effets réels, potentiels ou fantasmés, de la technologie sur la société et les individus. Jusqu'où peut-on aller dans l'utilisation de l'électronique pour améliorer la vie des gens ? Vous devez concevoir le système d'évitement d'une voiture intelligente : de l'enfant surgissant sur la chaussée à la poursuite de son chien coursant un chat, lequel choisirait votre programme : l'enfant, le chien, le chat ? Autre exemple : à supposer qu'existe un jour un moyen électronique efficace de sevrage tabagique, les assurances devront-elles le rendre obligatoire ?

INDISCRÉTIONS • Sibe Jan Koster a rejoint l'équipe des experts Elektor et entame la rédaction d'un livre représentant d'Elektor en Turquie, prépare une campagne d'été turque avec le soutien de Raoul Morreau Cirella, directeur de publication en Italie, a convaincu cet hiver 7 % d'abonnés et clients/partenaires Control, de Marc Friedheim, a été le livre le plus piraté du mois d'avril (grrrrr) • Afin de réduire les temps au point un nouveau service d'impression à la demande pour les membres et clients d'outremer

le chien ou le chat ? »
Tech the Future

Revient-il aux électroniciens et aux entreprises de répondre à ces questions ? Par où passera la ligne de démarcation entre éthique et électronique ? Sera-t-elle tracée par les techniciens ou par les politiques ?

Tessel Renzenbrink, rédactrice en chef du site Tech the Future, aimerait connaître votre point de vue et vous invite pour cela à la contacter par courriel à Tessel.Renzenbrink@eimworld.com, ou via Tweeter (@_Tessel). Après l'été, Tessel publiera sur son site les opinions reçues afin de nourrir un débat qui tôt ou tard nous concernera tous. On n'a pas fini de parler d'éthique et d'électronique !

sur les µC STM32 Cortex M4 • Cumhur Cakmak, le • Par rapport aux chiffres de l'hiver 2014, Antonio supplémentaires de nous rejoindre • Arduino in et frais de transport, Maarten Timmers Verhoeven met

PROFIL D'EXPERT

Elektor est au cœur d'un réseau de plus de 1 000 experts et d'auteurs engagés dans la publication de livres, d'articles, de DVD, de webinaires et autres événements. Coup de projecteur !

Nom : **Rémy Mallard**

Âge : **48**

Pays : **France**

Études : **Bac F2 en électronique**

Publications : **3 livres (L'électronique pour les débutants** (ISBN 9782866611866)**, Les microcontrôleurs PIC pour les débutants** (ISBN 9782866611931)**, Construisez votre émetteur FM), plusieurs nouvelles**

Enseignement : **plusieurs cours en audio et en électronique**

Qui êtes-vous, Rémy Mallard ?
J'ai 48 ans et suis père d'un garçon et de cinq filles. Je prête ma voix pour des documentaires ou autres livres audio, je suis formateur dans une école d'audiovisuel, j'offre des services pros dans le domaine audio et… plein d'autres choses encore !

Quelle est votre expérience ?
Création de logiciels pour Windows, programmation de PIC, montage et restauration de systèmes audio, formation technique en école et entreprise, animations radio, le tout depuis plusieurs années.

Qui a été votre modèle en électronique ?
Mon père, mon parrain, et tous ceux qui m'ont poussé à persévérer, même lorsque la seule chose qui sortait de mes circuits était de la fumée !

Quel progrès majeur attendre de l'électronique ?
La possibilité de fabriquer ce que nous devons acheter tout fait aujourd'hui. L'impression 3D nous y conduit, ce qu'on peut faire avec est surprenant. Demain nous pourrons fabriquer un téléphone ou une télé de la couleur ou de la forme que nous voudrons.

Sur quel sujet comptez-vous écrire à l'avenir ?
Je n'ai pas d'idée précise, mais si j'avais à écrire un nouveau livre je l'écrirais pour les débutants. Rendre simple l'explication d'un sujet technique compliqué est difficile, mais très enrichissant.

Si Elektor vous donnait 100 €, qu'achèteriez-vous, et pourquoi ?
De quoi créer un kit d'initiation, que j'offrirais à des débutants. Beaucoup hésitent à se lancer dans l'électronique parce qu'ils s'en pensent incapables, ou parce qu'ils n'ont pas d'argent. Un coup de pouce peut aider à franchir le pas.

Avez-vous un objet ou circuit électronique préféré ?
Quand j'étais petit, j'étais fasciné par les lasers He-Ne (les rayons de la mort !) Plus tard, lorsque le prix des lasers à diode a baissé, j'en ai acheté pour ajouter des fonctions à des jouets et des instruments de musique. J'en possède aujourd'hui plus de 150. ⏮

(150281 – version française : Hervé Moreau)

hexadoku

faute de grillon, mange des merles...

Un oiseau siffle dans les branches,
Et sautille, gai, plein d'espoir,
Sur les herbes, de givre blanches,
En bottes jaunes, en frac noir.
C'est un merle, chanteur crédule,
Ignorant du calendrier,
Qui rêve soleil et module
L'hymne d'elektor en juillet.

Théophile Gautier

Une grille hexadoku est composée de chiffres du système hexadécimal, de 0 à F. Remplissez le diagramme de 16 x 16 cases de telle façon que **tous** les chiffres hexadécimaux de 0 à F (0 à 9 et A à F) n'apparaissent **qu'une seule et unique fois** dans chaque rangée, colonne et carré de 4 x 4 cases (délimités par un filet gras).
Certains chiffres, déjà placés dans la grille, en définissent la situation de départ.

Pour participer, inutile de nous envoyer toute la grille, il suffit de nous donner **la série de chiffres** sur fond grisé.

Participez et gagnez ! Nous tirerons au sort **cinq bonnes réponses** internationales reçues dans les délais ; leurs auteurs recevront chacun un chèque-cadeau d'une valeur de **50 €** à valoir dans l'e-choppe d'Elektor.
À vos crayons !
Envoyez votre réponse (les chiffres sur fond grisé) avant le **1er septembre 2015** avec vos coordonnées par courrier électronique exclusivement à **hexadoku@elektor.fr**

Les gagnants

La solution de la grille du numéro de mai est **8205E**
Les cinq bons Elektor d'une valeur de 50 € vont à :
Julian Muscat (Malte), **Denis Moucharte** (Belgique), **Philippe Monnard** (Suisse), **Martin Kübel** (Allemagne) et **John Jones** (États-Unis).
Bravo à tous les participants et félicitations aux gagnants !

							E			9					1
D					F		C	0	1	6	B	8		4	
	F			B						A	8		0		
B		0			1		9			5	7	F	6		D
E		3	B		2	7	A		5	D		0			
C			6						F		3	B		5	7
	5		9	0					2						
		F	8	6			5	E			C	9		D	2
2	0		F	A			6	5			E	D	1		
						8					1	A		3	
1	6		C	D		2						E			4
			3		E	F		D	A	C		5	8		6
8		4	1	E	A			7		3			D		9
		6		2	D						A			E	
	7		D	3	6	5	B	8		F					C
A					4			9							

6	5	A	B	9	E	0	C	7	8	2	D	F	1	3	4
3	E	2	8	A	D	1	6	4	B	F	9	0	C	5	7
1	4	7	C	F	3	8	2	0	5	E	A	B	6	D	9
9	D	F	0	4	B	5	7	C	1	3	6	E	2	8	A
D	2	9	3	5	4	C	8	1	E	0	7	A	B	F	6
7	F	5	4	0	9	B	1	6	A	C	2	3	D	E	8
8	A	0	E	D	6	2	3	F	9	5	B	1	4	7	C
B	C	6	1	E	7	F	A	3	4	D	8	2	9	0	5
4	8	B	2	6	5	A	0	D	F	1	C	7	E	9	3
A	0	C	9	2	1	7	D	B	3	8	E	4	5	6	F
5	6	1	F	3	C	9	E	2	7	4	0	8	A	B	D
E	3	D	7	8	F	4	B	A	6	9	5	C	0	1	2
C	7	E	5	B	0	3	9	8	2	6	4	D	F	A	1
F	9	3	A	7	2	6	4	E	D	B	1	5	8	C	0
0	B	4	6	1	8	D	F	5	C	A	3	9	7	2	E
2	1	8	D	C	A	E	5	9	0	7	F	6	3	4	B

Tout recours est exclu, de même que le sont, de ce jeu, les personnels d'Elektor International Media et leur famille. Un seul gagnant par foyer.

DÉCOUVRIR CRÉER PARTAGER

elektor PCB Service

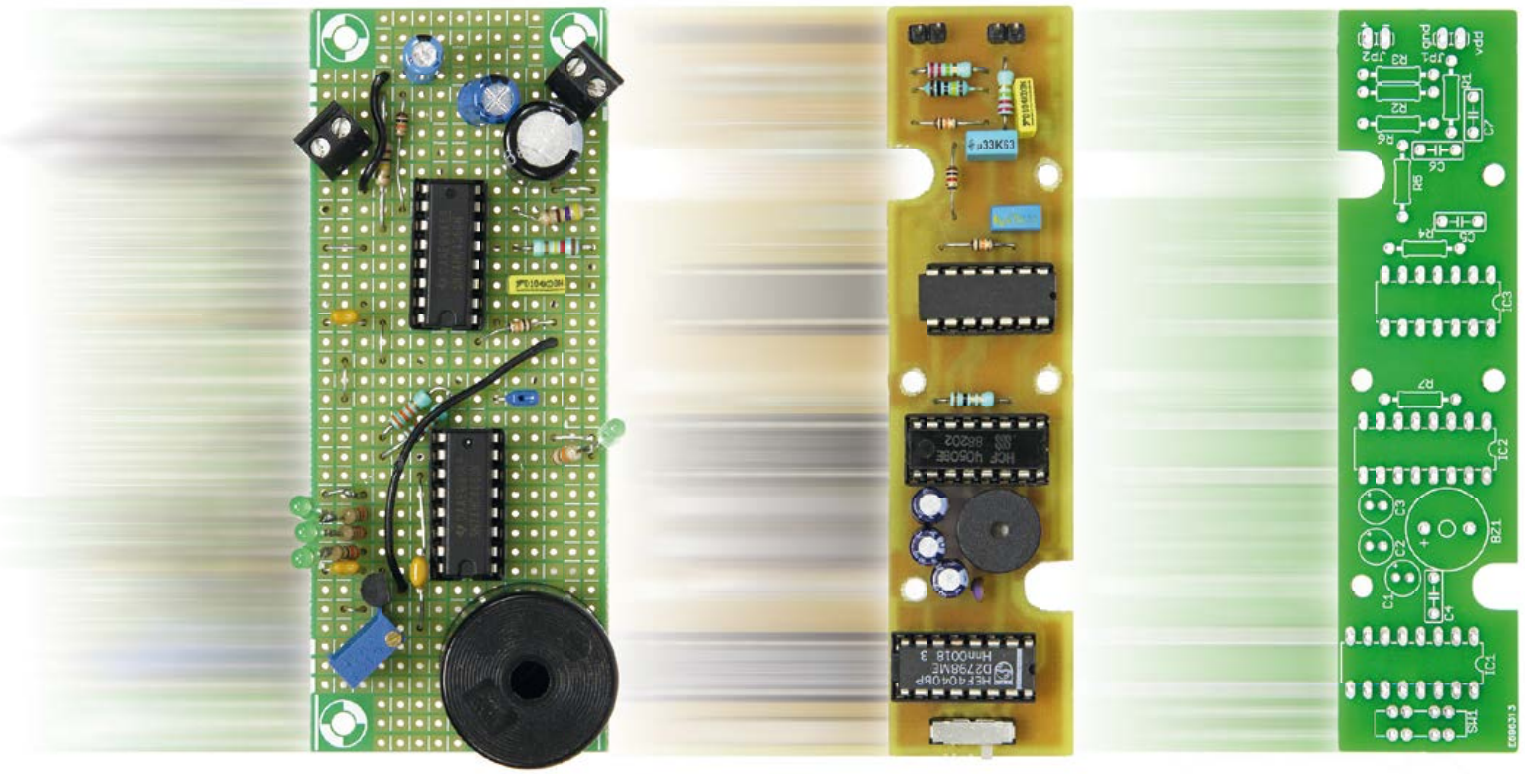
le fruit de la coopération d'elektor et d'eurocircuits

confiez-nous
la production de vos circuits imprimés,
vous avez tout à y gagner !

le prix ! *irrésistible...*

la qualité ! *imbattable...*

la fiabilité ! *inégalée...*



elektor PCB service est, à l'échelle européenne, le service le plus complet de fabrication sur mesure de circuits imprimés. Ce service commence en ligne, par le confort et l'efficacité d'outils faits sur mesure, étudiés pour vous permettre de visualiser votre commande et de l'analyser *avant* de payer.

- Pour vos débuts, vous utiliserez éventuellement le service de **prototypage sans masque de soudure** ; dans ce cas, vos circuits imprimés simple ou double face sont nus.
- Pour les utilisateurs exigeants, nous proposons les options sans compromis. Le **PCB Visualizer** montre les circuits imprimés tels qu'ils seront livrés, le **PCB checker** procède à une vérification technique de votre circuits imprimés (*design rules check*), et enfin le **PCB configurator** facilite la préparation de la commande.

Des menus bien conçus et un guidage par options accélèrent le processus de commande sans négliger aucun détail. Au moment de passer votre commande, vous savez exactement ce qui sortira de nos machines.

Qu'il soit perso ou pro, confiez votre prochain PCB à :

www.elektorPCBservice.com

Les microcontrôleurs XLP allongent la durée de vie des batteries

Faible consommation en mode veille et différentes sources du réveil

- Consommation en mode veille réduite jusqu'à 9 nA
- Réinitialisation BOR en cas de microcoupures à 45 nA
- Horloge temps réel réduite à 400 nA

Courants dynamiques faibles

- Jusqu'à 30 µA/MHz
- Efficacité énergétique à l'exécution

Vaste portefeuille de microcontrôleurs XLP

- 8 à 100 broches, 4 à 512 ko de Flash
- Un large choix de boîtiers, y compris boîtiers réduits à l'échelle d'une puce

Caractéristiques bénéficiant aux batteries

- Permet une durée de vie des batteries > 20 ans
- Fonctionne dès 1,8 V avec auto-écriture et fonctions analogiques
- Superviseurs à faible consommation pour un fonctionnement sûr (BOR, WDT)

Éventail des périphériques flexible

- USB, LCD, RTC et tactile intégrés
- Plus besoin de composants externes onéreux

microchip DIRECT
www.microchipdirect.com

MICROCHIP

www.microchip.com/xlp