www.elektor-magazine.fr

.magazine

mai 2013 | nº 419

elektor

libérez les microcontrôleurs

EFM: bibliothèque portable entre plateformes

I/O LEDs LCD ADC PWM SD MUX TCP/IP

● de BASIC à Python | **barostick USB** | **et l'homme créa sa puce** (4)

LCR-mètre 0,05 % 3e partie | bibliothèque de micrologiciels embarqués

réveil multifonction 1ère partie | **guidage par menu**

● **flux et reflux de soudure** ● le monde d'Elektor

ISSN 0181-7450

(B) 7,90 € • (CH) 13,20 FS • (CAN) 9,75 $ca • (And) 7,20 €
DOM surface 8,20 € • DOM avion 9,70 € • N Cal/S 950 cfp

L 19624 -419- F: 7,20 €

L'ORIGINAL DEPUIS 1994
PCB-POOL®
Beta LAYOUT

Le nouveau standard

Finition ENIG au lieu de l'étamage chimique sur votre carte

De l'or !

Sans supplément !

La meilleure finition : nickel-or chimique

Multicouche

Jusqu'à 50%

de réduction de prix

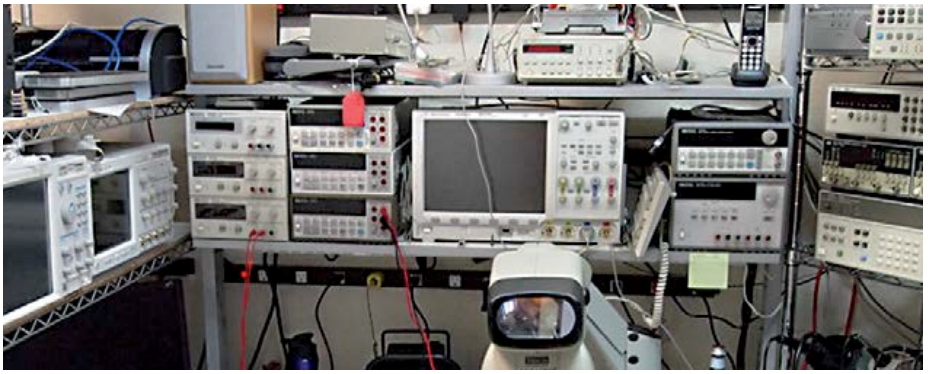
pour les cartes multicouche 2-5

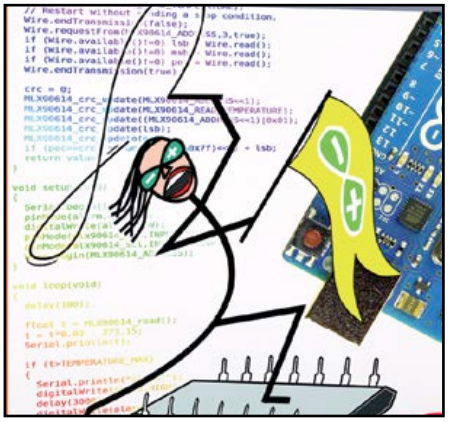
www.pcb-pool.com

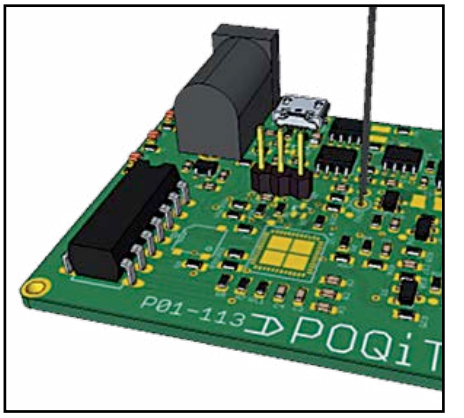
Beta LAYOUT
create : electronics

sommaire

elektor magazine



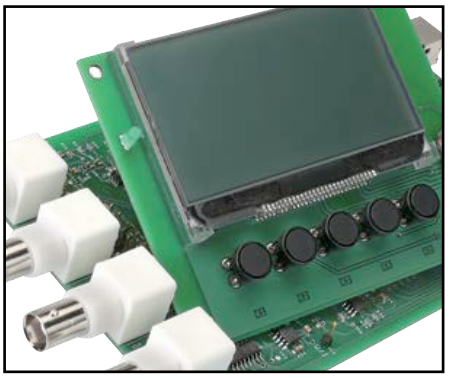


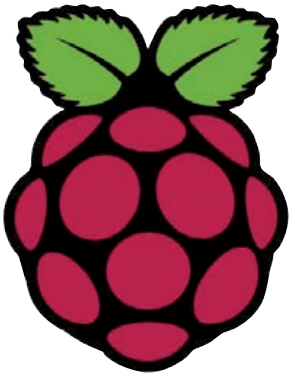


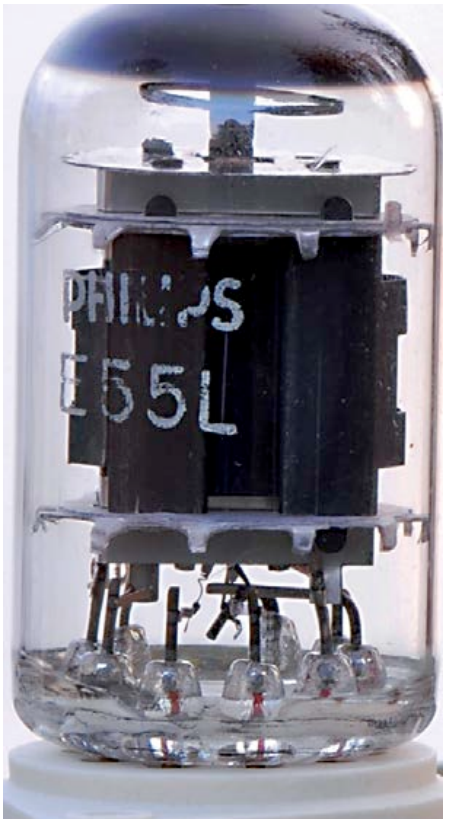
communauté

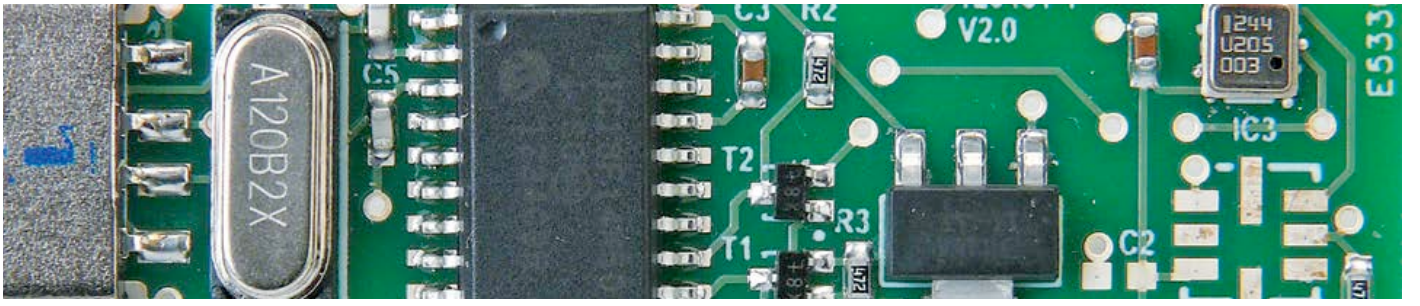
industrie

projects









36ème année, n°419 mai 2013
ISSN 0181-7450
Dépôt légal : avril 2013
CPPAP 1113 U 83713

ELEKTOR / PUBLITRONIC SARL
c/o Regus Roissy CDG
1, rue de la Haye
BP 12910
FR - 95731 Roissy CDG Cedex

Tél. : (+33) 01.49.19.26.19
du mardi au jeudi de 8h30 à 12h30
Fax : (+33) 01.49.19.22.37
www.elektor.fr

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

Elektor est édité par Elektor International Media B.V.
Siège social : Allee 1 - 6141 AV Limbricht, Pays-Bas

Imprimé aux Pays-Bas
par Senefelder Misset – Doetinchem

Distribué en France par M.L.P.
et en Belgique par A.M.P.

Votre labo

Conformément à la mission que nous nous étions assignée il y a quelques années, Elektor est devenu la plateforme internationale pour les électroniciens passionnés. Avec la montée en puissance de **www.elektor-labs.com**, on peut affirmer que *notre* labo est en train de devenir *votre* labo. Non seulement *vos* contributions sur ce site sont de plus en plus nombreuses, mais leur qualité et leur degré de finition ne cessent d'augmenter. Cette double progression est d'autant plus souhaitable que pour répondre aux besoins du plus grand nombre de lecteurs, le contenu de nos publications doit correspondre à *vos* attentes.
C'est un plaisir pour nous de coopérer avec vous et l'on entend dire que ce plaisir est réciproque. Elektor est plus qu'une publication farouchement indépendante, c'est aussi une communauté de membres payants et d'auteurs payés. Autour du chaudron où mijotent les projets, nos lecteurs partagent leur expertise.
Nous n'ignorons pas, cependant, que pour bien des auteurs francophones potentiels, la langue anglaise en usage sur **www.elektor-labs.com** est perçue comme un obstacle. Nous admettons volontiers qu'il est intimidant de se retrouver face aux catégories *Proposal*, *In Progress*, et *Finished.*
À tous ceux qui ressentent cette inhibition, je recommande de lire ce que Clemens Valens écrit page 14 de ce numéro. Suivez ses conseils, laissez flotter les rubans. Et si ça coince, Clemens vous aidera. Il comprend et parle parfaitement le français, il est même charmant (surtout avec les timides).
Pour diverses raisons, une grande partie des propositions qu'Elektor reçoit ne finissent pas dans la revue sous forme d'article. Pour certaines d'entre elles, c'est regrettable. Autrefois, nous ne pouvions que le déplorer. Désormais, ces contributions-là vivent aussi leur propre vie sur **www.elektor-labs.com**

Denis Meyer

Notre équipe

France
Denis Meyer
+31 46 4389435
d.meyer@elektor.fr

United Kingdom
Wisse Hettinga
+31 (0)46 4389428
w.hettinga@elektor.com

USA
Hugo Vanhaecke
+1 860-875-2199
h.vanhaecke@elektor.com

Germany
Ferdinand te Walvaart
+49 241 88 909-17
f.tewalvaart@elektor.de

Netherlands
Harry Baggen
+31 46 4389429
h.baggen@elektor.nl

Spain
Eduardo Corral
+34 91 101 93 95
e.corral@elektor.es

Italy
Maurizio del Corso
+39 2.66504755
m.delcorso@inware.it

Sweden
Wisse Hettinga
+31 46 4389428
w.hettinga@elektor.com

Brazil
João Martins
+55 11 4195 0363
joao.martins@editorialbolina.com

Portugal
João Martins
+351 21413-1600
joao.martins@editorialbolina.com

India
Sunil D. Malekar
+91 9833168815
ts@elektor.in

Russia
Nataliya Melnikova
+7 (965) 395 33 36
Elektor.Russia@gmail.com

Turkey
Zeynep Köksal
+90 532 277 48 26
zkoksal@beti.com.tr

South Africa
Johan Dijk
+31 6 1589 4245
j.dijk@elektor.com

China
Cees Baay
+86 21 6445 2811
CeesBaay@gmail.com

Notre réseau

elektor · VOICE COIL · CIRCUIT CELLAR · hifi.nl MUZIEK - FILM - LIFESTYLE · audioXPRESS · techthefuture

vous connecte à

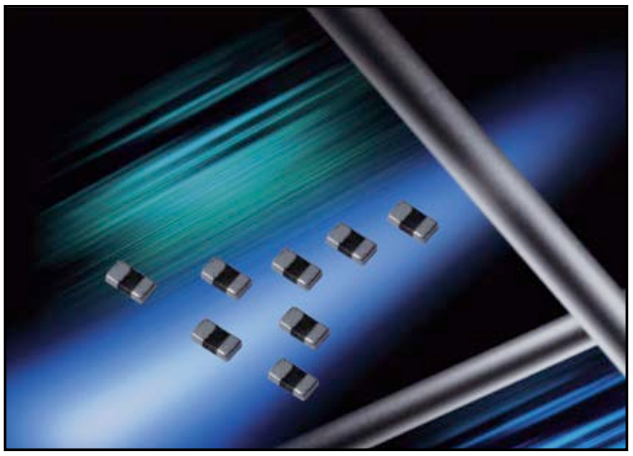
Nos annonceurs

Pour placer votre annonce dans le prochain numéro d'Elektor
veuillez contacter Mme Ilham Mohammedi par téléphone au (+31) 6.41.42.25.25
ou par courrier électronique : i.mohammedi@elektor.fr

Vos correspondants

Nous sommes à votre service pour toute question relative à votre commande ou votre abonnement par téléphone au (+33) 01.49.19.26.19 lundi, mardi et jeudi de 8h30 à 12h30 ou par courriel : service@elektor.fr

Une varistance automobile offre une protection bidirectionnelle contre les ESD et une ultra basse capacité



AVX Corporation, spécialiste des composants passifs avancés et de l'interconnexion, propose une série de varistances automobiles sub-picofarad. Utilisant une technologie fiable à l'oxyde de zinc, les varistances certifiées AEC-Q200 Série AG offrent aux circuits automobiles sensibles à la capacité une protection bidirectionnelle contre les ESD et une ultra basse capacité sub-picofarad. Avec une valeur de capacité de 0,8 pF, un boîtier 0402 compact et de bas profil et un excellent temps de réponse aux décharges électrostatiques, elles protègent contre les surtensions les circuits sensibles tels que les systèmes d'information automobiles modernes à large bande passante, les commandes et écrans tactiles, et les applications RF.

« Les varistances automobiles sub-picofarad de la série AG sont destinées aux nouvelles technologies de traitement de l'information à large bande passante, qui exigent d'ultra basses valeurs de capacité pour minimiser la distorsion de signal » explique Jiri Machanicek, directeur de marketing technique chez AVX.

Les nouvelles varistances automobiles sub-picofarad de la série AG d'AVX affichent une haute fiabilité, de faibles pertes d'insertion et une bonne tenue aux décharges répétées, et elles sont conformes RoHS et certifiées AEC-Q200. Offrant jusqu'à 16 V de tension de travail, 0,8 pF de capacité avec une tolérance de ± 20 %, et des terminaisons 100 % étain à barrière de nickel, la série AG est idéale pour des applications d'antennes, de circuits RF, d'interfaces optiques, HDMI, Firewire, Thunderbolt, de bus de communication haut débit, GPS, de caméra, de capteurs tactiles, entre autres.

Amplificateurs Analog Devices

Les amplificateurs d'erreurs isolées d'Analog Devices surclassent les optocoupleurs et les régulateurs sans les applications d'alimentation.

Analog Devices annonce deux nouveaux amplificateurs d'erreurs isolés grâce auxquels les concepteurs d'alimentations disposent d'une alternative monocircuit aux techniques d'isolation habituelles basées sur des optocoupleurs et des régulateurs parallèle. Conçus pour les alimentations à contre-réaction linéaire utilisant des contrôleurs côté primaire, les amplificateurs d'erreurs isolés ADuM3190 et ADuM4190 se caractérisent par une bande passante de 400 kHz avec une précision initiale typique de 0,5 % à 25°C, et une précision totale de 1 % dans la plage de température comprise entre -40 et +125°C. Les fabricants d'alimentations alternatif/continu et continu/continu, notamment des modèles conformes au standard DOSA (*Distributed-power Open Standards Alliance*), bénéficient ainsi d'améliorations significatives en termes de vitesse et de plage de température, ainsi que d'une réduction de 5 % de la réponse transitoire.

Conçus dans la technologie d'isolation numérique *i*Coupler® d'ADI, les amplificateurs ADuM3190 et ADuM4190 éliminent le taux de transfert de courant (*CTR - Current-Transfer Ratio*) des optocoupleurs qui se dégrade au cours du cycle de vie des composants.

Caractéristiques :

- Bande passante : 400 kHz
- Précision initiale : 0,5 % à 25°C
- Précision totale de 1 % dans la plage de température comprise entre –40˚C et +125˚C
- Fonctionnement basse consommation : <7 mA
- Tension d'isolation :
  - ADuM3190 : 2.5 kVeff (rms)
  - ADuM4190 : 5 kVeff (rms) (renforcé)

www.analog.com/ADuM3190 **www.analog.com/ADuM4190**

PIC24 Lite : nouvelles fonctions analogiques intégrées

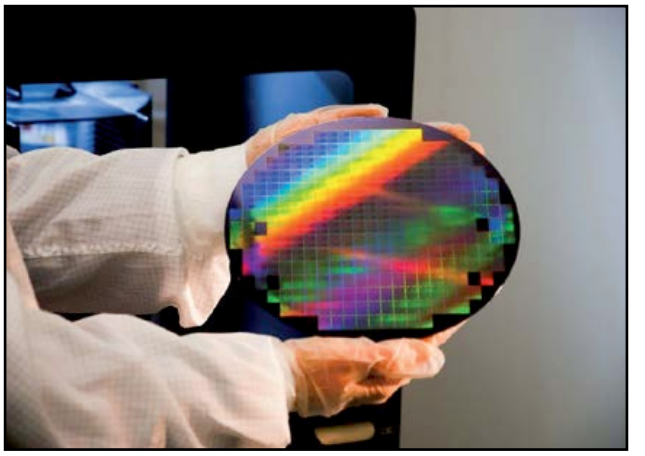
La famille économique PIC24 KM de microcontrôleurs Microchip à 16 bits offre jusqu'à 16 Ko de mémoire Flash, 2 Ko de RAM et 512 octets d'EEPROM, ainsi qu'un niveau élevé d'intégration des fonctions analogiques, idéal pour les applications économiques du marché automobile, du grand public, du secteur médical et industriel.

Parmi ces fonctions : CAN à 12 bits à détection de seuil, CNA à 8 bits pour des boucles de commande analogiques et des références de comparateurs précises, ainsi que des amplificateurs opérationnels (AOP) pour amplifier des signaux de capteurs. Les PIC24 KM sont les premiers à intégrer les nouveaux modules périphériques MCCP (Capture/Comparaison/PWM à multiples sorties) et SCCP (Capture/Comparaison/PWM à sortie unique), dotés de temporisateurs et de la commande PWM avancée, permettant de créer des applications de commande de moteur, d'alimentation et d'éclairage. Les modules MCCP et SCCP associent des fonctions de temporisateurs, de capture d'entrée, de comparaison de sortie et de PWM, le tout avec une seule et unique base de temps, pour une flexibilité optimale. Ces modules, capables de fonctionner de manière asynchrone, sont compatibles avec des temporisateurs à 16/32 bits ainsi qu'avec des horloges très rapides pour une meilleure résolution. Ils permettent également le fonctionnement automatique en veille.

Il s'agit par ailleurs de la première famille de PIC24 dotée de 2 cellules logiques configurables (CLC) pour une meilleure interconnexion des périphériques intégrés. Grâce au module CLC, il est possible de créer des fonctions logiques intégrées personnalisées en temps réel. Le module est compatible avec l'outil de configuration CLC, qui permet de générer les équations logiques graphiquement au lieu de le faire en assembleur ou en C, ce qui représente un gain de temps pour les programmeurs.

La famille KM permet de créer des applications sous 3 V et 5 V. Avec leur très faible consommation et l'intégration analogique avancée, les produits KM conviennent pour des applications économiques, du type débitmètres, détecteurs de fumée, moteurs pas à pas ou sans balais, gradation de LED, chargement de batteries, capteurs de conditions environnementales et appareils médicaux jetables.

LED sur substrat en silicium de 200 mm

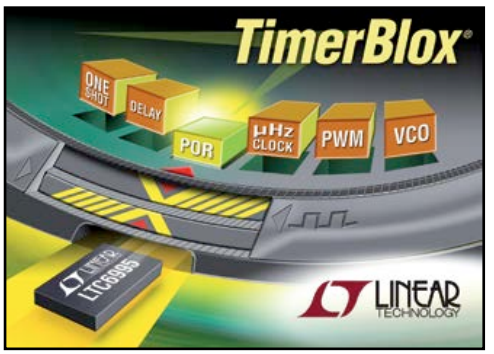


S'appuyant sur une technologie de rupture, à base de microfils GaN sur silicium, *Aledia* (Grenoble) annonce la fabrication de ses premières LED sur des plaques de silicium de 200 mm. Avec ces puces 3D, le coût de production est divisé par quatre, comparé aux LED planaires 2D conventionnelles.

Sur le marché en pleine croissance des LED (13,7 milliards d'euros en 2012), pour continuer à développer de nouvelles applications, notamment dans le domaine de l'éclairage, les LED doivent être disponibles à un prix nettement moins élevé qu'actuellement. Avec ses LED fabriquées sur des plaques de silicium de grande taille et ses coûts de production très faibles, la technologie microfils d'*Aledia* constitue une véritable rupture en termes de coût de production. « … Nous avons développé notre procédé de fabrication de microfils et l'avons transféré sur des plaques en silicium de grande taille soit 200 mm de diamètre par rapport au standard de l'industrie des LED encore de 50 à 100 mm », commente Giorgio Anania, co-fondateur d'*Aledia*. « Désormais, nous optimisons les performances de ces produits et préparons leur lancement sur le marché. »

Plus de 6 ans de R&D ont été nécessaires au laboratoire LETI du CEA, à Grenoble, pour concevoir la technologie 3D de microfils GaN sur silicium. Cette innovation majeure en cours de commercialisation a été pilotée notamment via les équipes de développement d'*Aledia*. Ce dernier a aujourd'hui une licence mondiale exclusive sur les brevets du CEA existants et à venir, portant sur cette technologie et ses applications dans le domaine de l'éclairage.

LTC6995 : minuteur simple et efficace



Le LTC6995 est une horloge basse fréquence, précise et simple. Il peut être configuré uniquement pour les applications de réinitialisation à la mise en marche et de minuterie de chien de garde, de longue durée. Comme les autres membres de la famille *TimerBlox*, il combine un oscillateur programmable avec des circuits de précision et logiques. Une très large bande de fréquences, fixées par une résistance, permet d'obtenir une horloge de 1 ms à 9,5 h. À la mise en marche ou sur un signal de réinitialisation, le LTC6995 démarrera un cycle complet d'horloge. La possibilité d'une réinitialisation et la période programmable sont destinées aux longues durées.

La fonction de réinitialisation écourte la durée de l'impulsion de sortie, vide les diviseurs internes et maintient la sortie en un état haut ou bas. La polarité du signal de réinitialisation d'entrée et le signal de sortie peuvent être configurés pour un fonctionnement actif à l'état haut ou actif à l'état bas. Deux versions du LTC6995 sont disponibles, avec une fonction de réinitialisation inversée.

Les composants *TimerBlox* sont des circuits intégrés et peuvent fonctionner sous de fortes accélérations, vibrations et températures extrêmes. Aucun condensateur de durée, ni quartz, ni microcontrôleur, ni programmation n'est requis. Ces composants intégrés assurent plus de précision et de stabilité et consomment moins que les oscillateurs standard à base de résistances et de condensateurs. La possibilité de supporter un courant de 20 mA, entrant ou sortant, permet de piloter directement des photocoupleurs pour assurer une isolation électrique. Spécifiés entièrement sur la gamme de températures de –55 °C à 125 °C, les composants *TimerBlox* conviennent aux applications de l'automobile et de l'industrie, où de nombreux oscillateurs et microcontrôleurs ne peuvent pas fonctionner. LTC6995 est simple et efficace. »

nouveau livre de Clemens Valens, le chef du labo d'Elektor

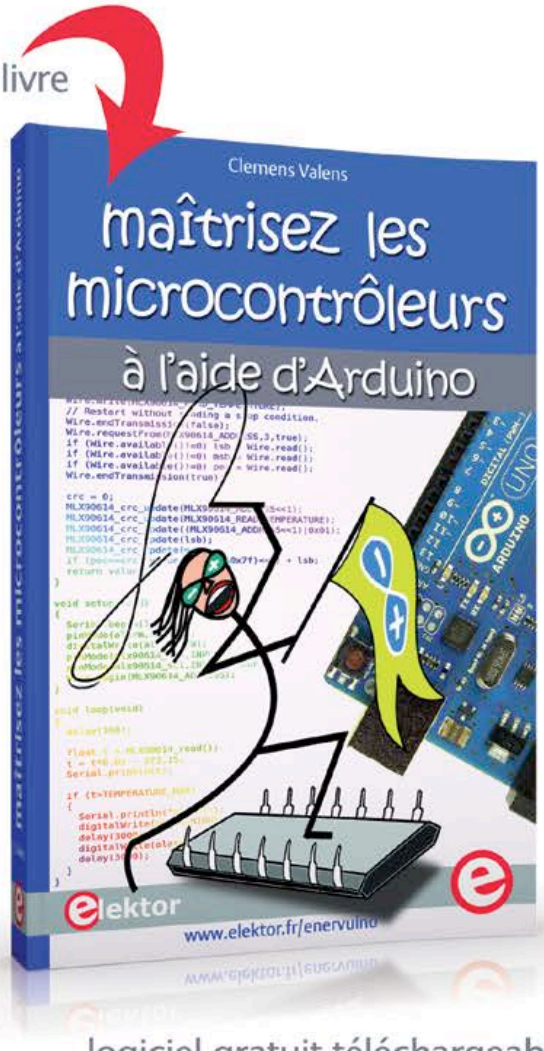
« Enfin une méthode à base de montages Arduino qui marche pour se débarrasser définitivement de ses amis et de sa famille...
Et pour se retrouver enfin seul et libre de passer tout son temps à apprendre la programmation des microcontrôleurs. »

Ce livre n'est pas

banal
pas superficiel
pas réducteur ni même simplificateur
pas barbant, pas soporifique
pas du genre « 30 applications rigolotes »
pas lâche, pas infantilisant
pas appuyé sur du matériel introuvable et/ou hors de prix

mais il est

original, par le fond et la forme
consistant, profond, complet
plaisant, souvent drôle, parfois hilarant
ou déconcertant, selon le tempérament de chacun,
généreux : il ne se contente pas de vous donner des envies,
mais donne aussi les moyens de les satisfaire,
courageux : il aborde les sujets laissés savamment
dans l'ombre par les autres, p. ex. les interruptions,
formateur, encourageant, exigeant et stimulant,
enraciné sur la plateforme de prototypage rapide Arduino,
la plus largement diffusée dans le monde



maîtrisez les microcontrôleurs à l'aide d'Arduino
352 pages – 10 chapitres – 13 réalisations inédites
sommaire et extraits : www.elektor.fr/arduino

logiciel gratuit téléchargeable

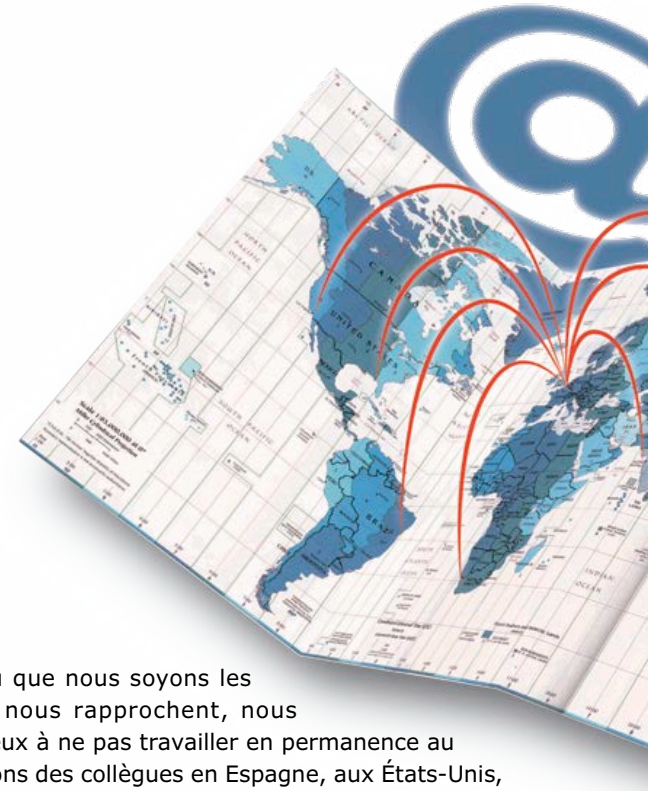
changez de loisirs devenez dresseur de puces !

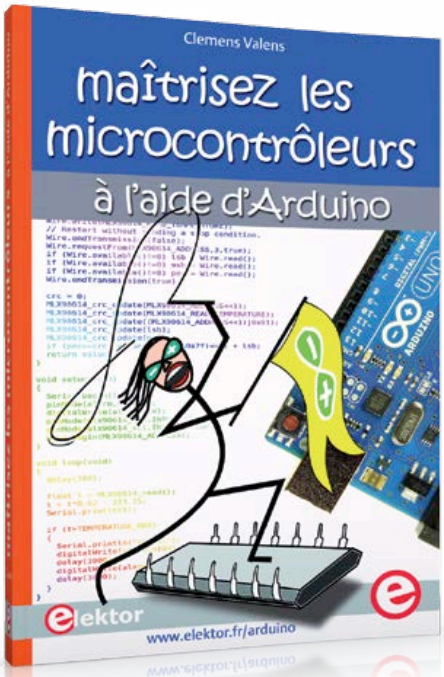
elektor

À la fin du livre, le lecteur qui n'aura rien sauté sera capable de mettre en œuvre n'importe quel microcontrôleur. Clemens Valens

le monde d'Elektor

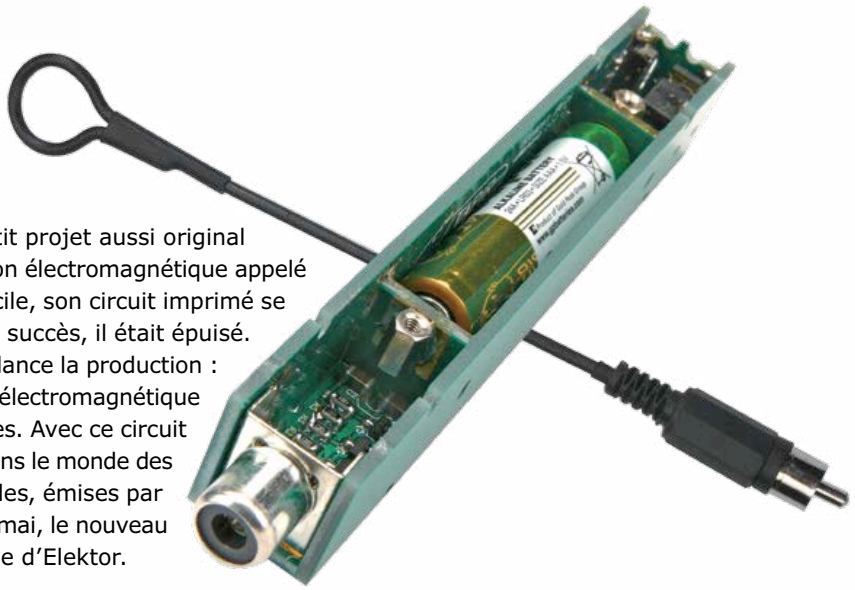
Wisse Hettinga (Elektor)





Le fouet

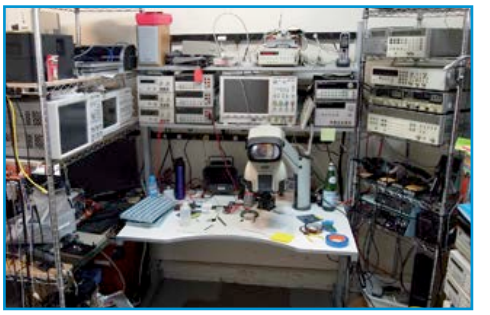
Elektor s'internationalise, où que nous soyons les moyens de communication nous rapprochent, nous sommes donc de plus nombreux à ne pas travailler en permanence au siège, aux Pays-Bas. Nous avons des collègues en Espagne, aux États-Unis, en Italie, en Inde et… en France, où réside par exemple Clemens Valens, directeur technique du laboratoire ! Nous avons déjà présenté ici son livre paru récemment et ne sommes nullement surpris par son beau succès, lié à l'originalité de l'approche des microcontrôleurs, par le versant Arduino, en pente douce, avec une bonne dose d'humour. Oui, l'humour, ingrédient énergisant, est présent jusque sur la couverture : le personnage au fouet qui terrasse le circuit intégré a fait jaser nos collègues, surpris d'ailleurs qu'on puisse se faire plaisir en écrivant à ses heures perdues un livre technique à la fois sérieux et drôle. Un *fouet*, sur un livre d'électronique ? Tant d'audace les intrigue, alors du coup ils vont le faire traduire en allemand et en anglais. Comme le texte est truffé d'astuces, ils vont s'amuser.



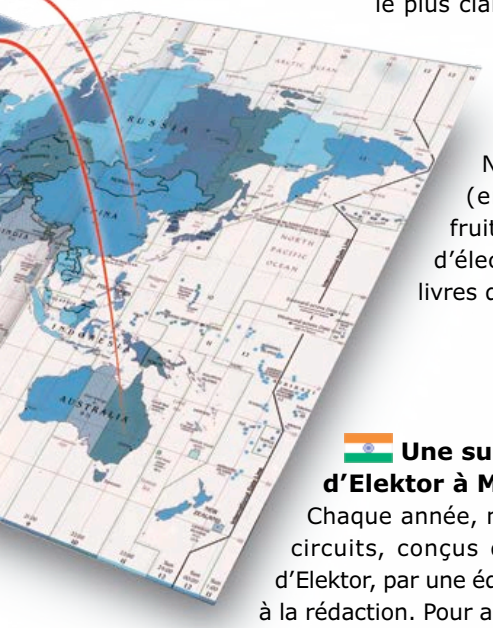
Le retour du TAPIR

L'an dernier, Elektor proposait un petit projet aussi original qu'attrayant : un détecteur de pollution électromagnétique appelé TAPIR. Conçu pour une réalisation facile, son circuit imprimé se transforme en boîtier. Victime de son succès, il était épuisé. Or notre partenaire Eurocircuits en relance la production : 1000 pièces. Renifler le rayonnement électromagnétique en le transformant en signaux audibles. Avec ce circuit et une simple oreillette, vous entrez dans le monde des ondes invisibles, mais non moins réelles, émises par votre téléphone ou votre tablette. Fin mai, le nouveau TAPIR sera disponible dans l'e-choppe d'Elektor.

Antre fécond



Greniers et garages bourrés d'électronique, voilà les lieux où s'invente la technique. C'est là que beaucoup d'entre nous passent le plus clair de leur temps, à concevoir de nouvelles applications. Bien sûr, les laboratoires de haute technologie ont aussi leur rôle à jouer, avec leurs budgets colossaux, mais ils n'ont pas le monopole de la fécondité. Notre confrère, le magazine Circuit Cellar (en anglais, *la cave à circuits*) publie les fruits récoltés dans leurs celliers par de nombreux passionnés d'électronique. Découvrez l'antre de Vincent Himpe, auteur des livres de la série LabWorX publiés par Elektor.



Une succursale du labo d'Elektor à Mumbai (Inde)

Chaque année, nous publions au moins 200 circuits, conçus dans notre labo, au siège d'Elektor, par une équipe de techniciens associés à la rédaction. Pour augmenter notre capacité de production, nous ouvrons une succursale à Bombay, en plein cœur du plus grand marché indien de l'électronique. Ce développement, soigneusement préparé depuis des mois, tombe à pic pour nous permettre de mener à bien notre mission à une cadence toujours plus soutenue : imaginer, tester et publier. Lamington Road est *la* rue de l'électronique de Bombay ; la densité de ses boutiques de composants laisse rêveur. Lorsqu'un concepteur a besoin d'un composant, il n'a que trois pas à faire pour le trouver ! Notre nouvelle équipe, de gauche à droite : Krishna Chandran, Sunil Malekar, Clemens Valens, Nandini Singh et Shreenivas GM Shree.

Des idées étincelantes

Elektor vient de s'engager dans un nouveau partenariat avec *RSComponents*, autour de *DesignSpark*. Il y aura d'une part le magazine du même nom, une nouvelle version de la publication BE adressée par *RSComponents* à ses clients. Il couvrira les derniers développements sur les nouvelles plates-formes matérielles et présentera une nouvelle série de projets d'Elektor. D'autre part, l'activité *DesignSpark* sera axée sur le lancement d'une section spéciale de conception et de réalisation du site DesignSpark.com. Les circuits imprimés de ces projets seront conçus spécifiquement avec le logiciel gratuit de CAO *DesignSpark PCB*, disponible sur www.designspark.com. Dans Elektor et sur notre site www.elektor-labs.com, des pages spéciales seront consacrées à cette nouvelle initiative.

(130051)

« sorry for my English »

Clemens Valens
(Elektor .Labs)

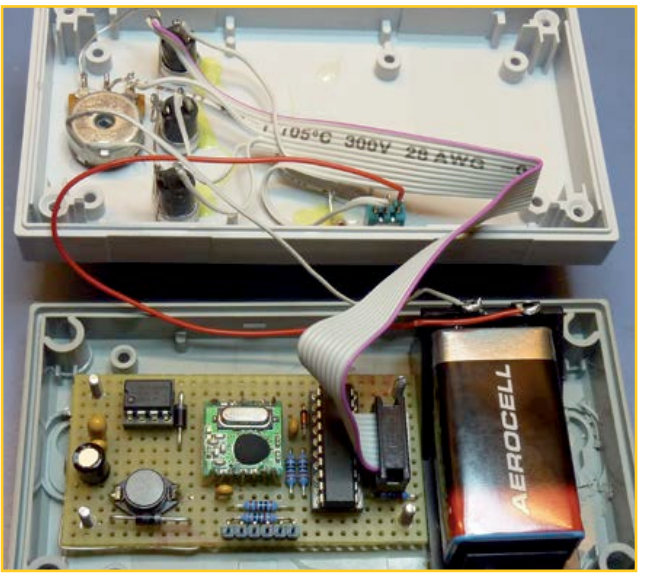
« *sorry for my English* » est une remarque fréquente sur les forums dans les discussions entre internautes, surtout quand quelqu'un désespère de trouver une information vitale pour sortir d'une panade quelconque. À moins que ce soit *sorry about my english* ou d'autres variantes. Depuis que je l'ai trouvée aussi sur le site Elektor**.**Labs, je me dis qu'il faut en bannir cette excuse inepte.

Cher co-LABO-rateur, laisse-moi donc te dire qu'en tant que modé-LABO-rateur de .labs je n'en ai rien à [♪♪♪] de ton angliche, ce sont tes [♪♪♪] idées qui m'intéressent, c'est mon [♪♪♪] job qui en dépend (*sorry for my English* ;-).
Tant fé pa pour ton nanglé, kontinu de publié des zidé kikif et tant fé pa pour la grand-mère et la ponctutation.

EDITOR'S CHOICE

électronicien courageux évite guerre de trains

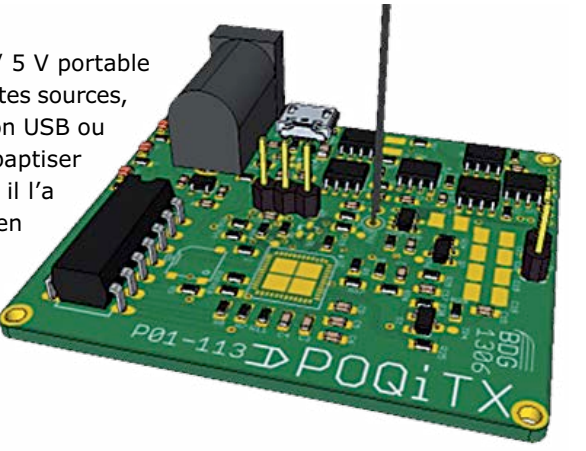
Quand deux conducteurs de trains ont, sans le vouloir, pris le contrôle du train l'un de l'autre et que la moutarde leur est montée au nez, l'ingénieur en électronique ***waltro*** a décidé qu'il était temps d'agir avant que ça tourne mal. Une nuit, alors que les belligérants pionçaient, notre valeureux ingénieur s'est introduit dans chaque train pour y remplacer le système de commande d'origine par un système multicanal, plus intelligent, mis au point par lui-même. Le lendemain, la paix était revenue. Bravo, ***waltro*** !
www.elektor-labs.com/node/3020



bon projet coulé par un mauvais titre

L'OP* ***BifrostDevGrp*** mettait la dernière main à son alim régulée 3,3 V / 5 V portable sur piles, un joli petit circuit, ramassé, futé, fonctionnant à partir de différentes sources, rechargeant sa batterie li-ion de 950 mAh sur le secteur, sur une connexion USB ou carrément sans fil, quand le désastre s'est produit. L'OP* a décidé de rebaptiser son projet d'un joli titre puis, sans se rendre compte des conséquences, il l'a posté sur Elektor**.**Labs. Ah, s'il avait au moins pris la peine de le décrire en quelques lignes captivantes, la casse aurait été limitée et le projet sauvé de l'oubli. Il ne l'a pas fait, malheureusement. Heureusement que je suis là, moi, pour repêcher de telles propositions vouées aux grands fonds et pour tenter de les remettre à flots.

www.elektor-labs.com/node/2969



la quête du Graal à tubes

La passion est le moteur de l'action, pour le pire et le meilleur. L'OP* Ken, passionné d'audio, est en quête du meilleur amplificateur à tubes jamais conçu. Il a entrepris de réunir une partie des composants et du matériel, mais il est loin d'avoir arrêté tous les détails. Il demande de l'aide. L'audio vous branche ? Ça vous tenterait de participer à cette quête de l'ultime ? Cela vous concerne, forcément, sinon vous ne liriez pas Elektor. Votre avis, vos connaissances, votre expérience, partagez-les ! Vos lumières sur les transformateurs et les condensateurs et sur tout ce qui détermine la qualité du signal audio, faites-les briller sur la page de ce projet !

www.elektor-labs.com/node/2949

pense-bête pour poubelles de recyclage

Je n'avais jamais accordé d'attention au calendrier du recyclage de ma ville jusqu'au jour récent où il a changé. Tout s'est détraqué. Soit je sortais la mauvaise poubelle, soit ce n'était pas le bon jour. Et c'est là que j'ai pensé à la proposition de ***MarkDonner*** sur **.**Labs. Un lien, publié par ***amigaman***, a attiré mon attention sur un groupe allemand (*ils s'y connaissent en recyclage !*) d'entraide pour *oublieurs de poubelles*. Ouf ! Je n'étais donc pas le seul.

Voilà que nous avons nous aussi notre groupe d'entraide. Vous cherchez de l'aide, vous aussi ? Alors rejoignez-nous pour participer à la création d'un pense-bête électronique pour poubelles de recyclage.

www.elektor-labs.com/node/2946

(130090-I)

www.elektor-labs.com

** Nous appelons OP (Original Poster) l'auteur d'une proposition ou d'une discussion en ligne et rappelons aux OP désireux de voir leur projet publié dans le magazine Elektor de lire (régulièrement) le courriel à l'adresse avec laquelle ils sont inscrits sur Elektor.Labs. C'est notre seul moyen de communiquer avec eux.*

le capteur capacitif du pauvre

Vous pouvez vous toucher pour faire plus simple !

Après le clavier tactile à reconnaissance de motif alias *Pattern Lock*, très perfectionné, présenté le mois dernier, voici ce qui est certainement le plus simple capteur capacitif pour microcontrôleurs et FPGA jamais conçu. Pas cher, très pratique et suffisamment robuste pour bon nombre d'applications.

Radovan Stojanovic
(Monténégro)

Coauteurs :
Nedjeljko Lekic et Zoran Mijanovic

Avec un capteur monobroche, l'électrode est habituellement reliée à une patte d'E/S bidirectionnelle P_n d'un microcontrôleur ou d'un FPGA à travers une résistance R (**fig. 1**). Le sens de la patte, entrée ou sortie, se règle avec un registre et son état est lu dans un autre registre. Ici, C_p correspond à la capacité d'une patte, soit environ 4 ou 5 pF pour un microcontrôleur ou un FPGA. C_b correspond à la capacité du condensateur formé entre votre corps et le sol — elle peut atteindre 400 pF, mais vaut typiquement entre 100 et 150 pF. C_{x1} et C_{x0} correspondent respectivement à la capacité de l'électrode lorsqu'elle est ou n'est pas en contact avec votre doigt. Habituellement, $C_b >> C_{x1}$, et $C_x \approx C_{x0}+C_{x1}$. En prenant C_{xo}=6,5 pF et D=10 mm, le graphique de la **figure 2** représente la décroissance de C_{x1} de 7 pF à 1,5 pF lorsque d_{x1} varie de 0,1 mm à 1 mm.

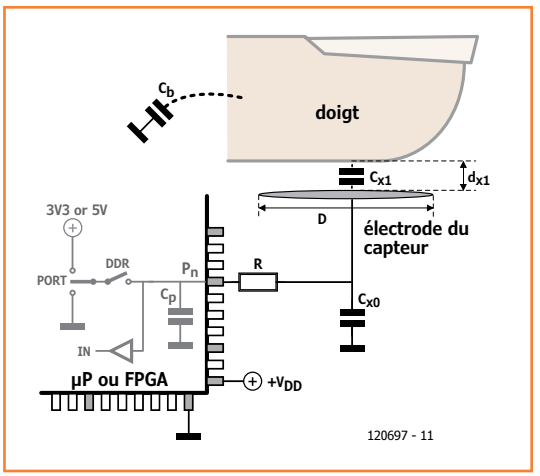


Figure 1.
Ajout d'un capteur capacitif à un microcontrôleur ou un FPGA.

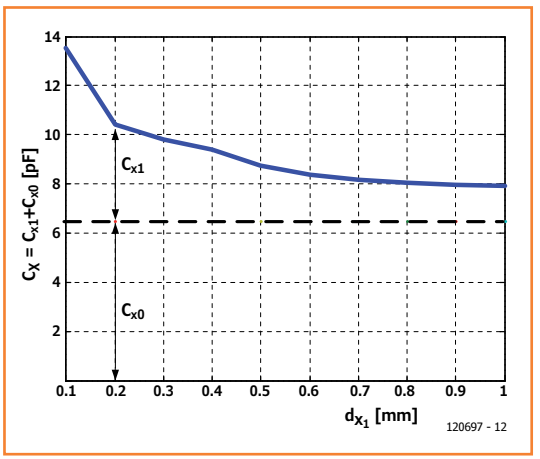


Figure 2.
Capacité équivalente d'un doigt et de l'électrode en fonction de la distance d_{x1}, pour D = 10 mm.

Principe du capteur capacitif

Le processus de mesure peut être comparé au transfert d'une certaine quantité d'eau d'un bol à un autre (**fig. 3**) ; le niveau d'eau correspond à la tension V, la taille du bol à la capacité C, et enfin la quantité d'eau à la charge Q.
La phase CHARGE démarre lorsque l'on impose un 1 logique (3,3 V ou 5 V) sur la patte P_n durant t_c=0,75-2 µs, c.-à-d. suffisamment longtemps pour charger C_p et C_x jusqu'à la tension correspondant à un niveau haut ; par exemple $V_1=V_2$=3,3 V. Lors de la phase DÉCHARGE, avec t_d entre 200 ns et 750 ns, C_p est entièrement déchargé, et V_2 décroit à une vitesse qui dépend de la constante de temps RC_x, jusqu'à $V_{2(td)}$; la charge restante est alors $QC_{x(td)}$. Lors de la phase ÉQUILIBRAGE qui dure t_s, la patte Pn passe en mode à haute impédance (Z) et la charge $QC_{x(td)}$ se répartit proportionnellement entre C_p et C_x. À la fin de cette phase $V_1 \approx V_2$, $i \approx 0$, et la phase MESURE peut commencer. La tension V_1 sera alors inférieure ou supérieure au seuil de basculement V_{TR} de la patte selon que votre doigt est ou non en contact avec l'électrode du capteur. Numériquement parlant, cela correspond à zéro ou un pour le bit correspondant du registre d'entrée.
Déterminer la valeur optimale pour R n'est pas facile. D'après nos essais, R=68 kΩ est la valeur

optimale pour V_{DD}=3,3 V, pour V_{DD}=5 V c'est 47 kΩ. Petit bonus : R protège la patte du FPGA des décharges électrostatiques jusqu'à environ 1000 V.

Le logiciel

La mesure est pilotée par des routines logicielles. Nous présenterons ici le code pour un seul capteur. Côté microcontrôleur, le pseudo-code du **listing 1** est prévu pour le compilateur WinAVR GNU GCC. La puce choisie est un ATMega16 de la famille de micros RISC à 8-bit d'*Atmel*.
Côté FPGA, notre composant (code VHDL du **listing 2**) possède trois ports : horloge (CLK), entrée du capteur (S) et état du capteur (SS). En gros, il s'agit d'un compteur et d'un comparateur. Les constantes génériques CHARGE_TIME_NS, DISCHARGE_TIME_NS et START_READ_NS correspondent respectivement aux durées t_c, t_d et t_s+t_m et PERIOD_NS à la durée d'un cycle de mesure.

Résultats expérimentaux

L'approche présentée ici a été confrontée à la pratique sur un ATMega16 d'*Atmel* et sur un FPGA EP1C6Q240C8 de la série *Cyclone* d'*Altera*. Après avoir écrit, compilé et débogué le code, nous avons programmé et configuré le micro et le FPGA. L'électrode de 10 mm de diamètre a été isolée avec une couche de 0,1 mm de papier. Pour nos mesures (**fig. 4a**), les sondes de l'oscilloscope ont été reliées aux deux pattes de la résistance R, introduisant une capacité supplémentaire C_{op} de 15 pF en parallèle de C_x et C_p. Nous avons utilisé un oscilloscope *Agilent Tech* DSO3102A et une sonde N2862A. Comme la capacité C_{x1} vaut 7 pF, l'ajout des sondes a diminué significativement la sensibilité du capteur. Pour amoindrir l'effet des capacités

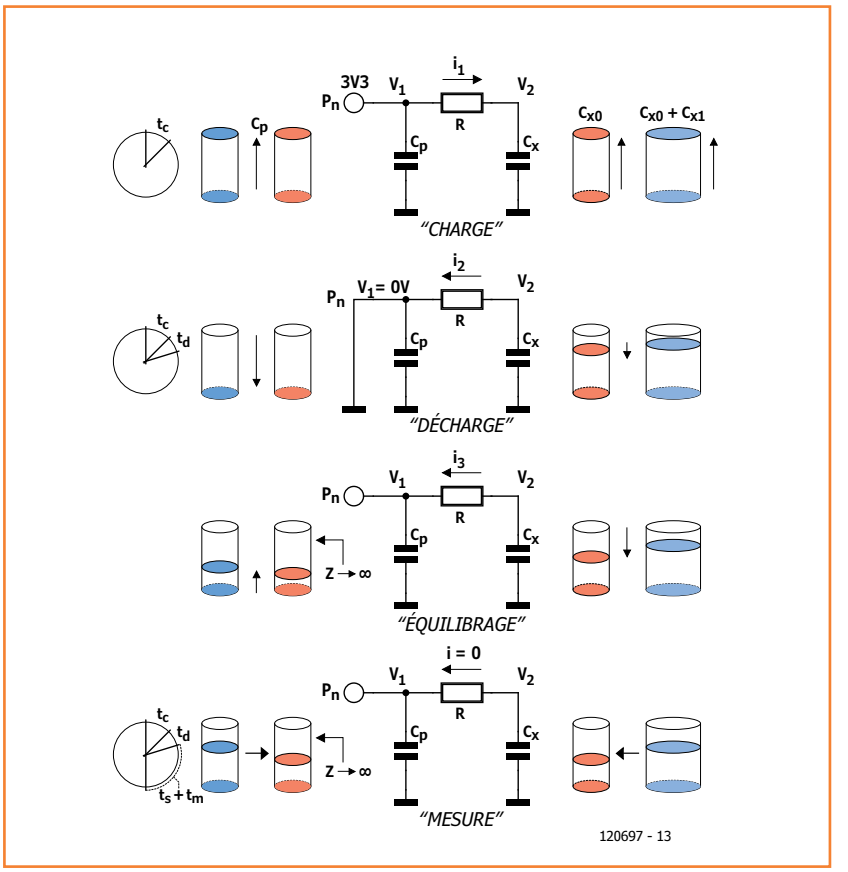


Figure 3.
Les quatre phases de fonctionnement de notre capteur capacitif.

Listing 1: Pseudo code de gestion de notre capteur dans un microcontrôleur

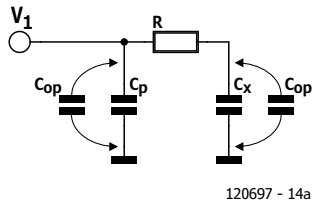
```
// MC CAP PAD PSEUDOCODE
// phase DÉCHARGE
// Initialement PORT=1
PORTC &= ~TOUCH;  // PORT=0
short_delay();  // délai de 750ns

// phase ÉQUILIBRAGE
DDRC &= ~TOUCH; // port en entrée
short_delay();  // délai de 750ns

//phase MESURE
if(PINC & TOUCH) PORTD |= LED; // si la patte est au niveau haut, on allume la LED
else PORTD &= ~LED; // sinon on l'éteint

//phase CHARGE
PORTC |= TOUCH; // port=1
  DDRC |= TOUCH;  // port en sortie
```

parasites introduites par les sondes, les durées des phases de charge, décharge et d'équilibrage ont été augmentées respectivement aux valeurs suivantes : t_c=24 µs, t_d=2.5 µs et t_s+t_m=15 µs. Les oscillogrammes (**fig. 4b**) correspondent aux circuits équivalents de la figure 3. La présence d'un doigt sur le capteur est détectée de manière fiable. Sans les sondes, la différence entre les deux oscillogrammes devrait être plus prononcée. Un microcontrôleur est capable de gérer plusieurs capteurs, p. ex. 8 ou 16 reliés à un ou plusieurs port(s) d'E/S. L'algorithme restera quasiment le même étant donné que les différents capteurs sont lus en parallèle en une seule fois simplement en lisant l'état du port d'E/S. Mais, quand il s'agit de traitement parallèle, un FPGA est une meilleure solution. Un seul capteur n'utilise typiquement que 60 EL (éléments logiques) d'un FPGA. Pour un FPGA *Altera Cyclone* EP1C6Q240C8, cela ne correspond qu'à un petit 1 % du nombre d'EL disponibles (5980). Des ensembles de 2, 4, 8, 16, 32 ou 64 capteurs utilisent respectivement 140, 151, 168, 213, 297 ou 462 EL. Vous remarquerez que le composant ne prend que très peu de place ; un ensemble de 64 capteurs n'utilise que 7,7 % des ressources d'un EP1C6Q240C8 et 72 des 240 pattes disponibles. En pratique, vous pourrez embarquer une interface à *N* entrées dans un petit FPGA pour peu que vous disposiez du nombre de pattes nécessaire.

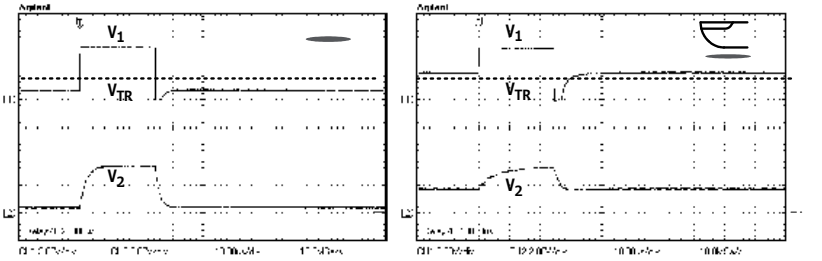
Le capteur possède, jusqu'à un certain point, une bonne immunité au bruit. Étant donné que le traitement est réalisé en parallèle, la période d'échantillonnage n'augmente pas avec le nombre de capteurs et pourra être ajustée de 5 µs à 20 µs en fonction de votre application. L'impédance d'entrée de la patte P_n sera faible durant les

Listing 2 : Le code VHDL de notre capteur capacitif.

```
-- FPGA CAP SENSOR, VHDL CODE
LIBRARY ieee;
USE ieee.std_logic_1164.all;
USE ieee.numeric_std.all;

ENTITY FPGA_tri_state IS
generic (CHARGE_TIME_NS       : integer := 144405;
         DISCHARGE_TIME_NS    : integer := 4545;
         START_READ_NS        : integer := 10000;
         PERIOD_NS            : integer := 400000
         );
    PORT(
            CLK    : in std_logic;
            S  : inout std_logic;
            SS : out std_logic
        );
END FPGA_tri_state;

ARCHITECTURE arh OF FPGA_tri_state IS

constant clock_cycle  : integer := 21;
constant one_time: integer := CHARGE_TIME_NS / clock_cycle;
constant zero_time    : integer := (CHARGE_TIME_NS + DISCHARGE_TIME_NS) / clock_cycle;
constant read_time    : integer := (START_READ_NS + CHARGE_TIME_NS + DISCHARGE_TIME_NS) / clock_cycle;
constant max_time     : integer := PERIOD_NS / clock_cycle;

SIGNAL  b  : STD_LOGIC; -- bascule D stockant la valeur de retour.
shared variable counter : integer range 0 to PERIOD_NS/clock_cycle :=0;
signal dir      : std_logic;

BEGIN
    PROCESS(CLK)
```



a



b

phases de charge et décharge et ne sera sensible au bruit que lors des phases d'équilibrage et de mesure (à cause de la haute impédance). Cela permet au capteur de bien résister aux bruits de basse fréquence, tel que ceux engendrés par le secteur à 50 Hz ou 60 Hz. Par exemple, un bruit dû au secteur qui possèderait une amplitude induite de 100 V n'ajouterait qu'une tension de V_{ACmax}= 22 mV lors de la phase de mesure. On peut donc considérer que la tension du secteur n'affecte pas la mesure. L'influence du bruit rayonné (haute fréquence) peut être réduite à l'aide de filtres logiciels implantés en C ou VHDL, même si en pratique le bruit environnemental ne varie que peu en fonction du temps.

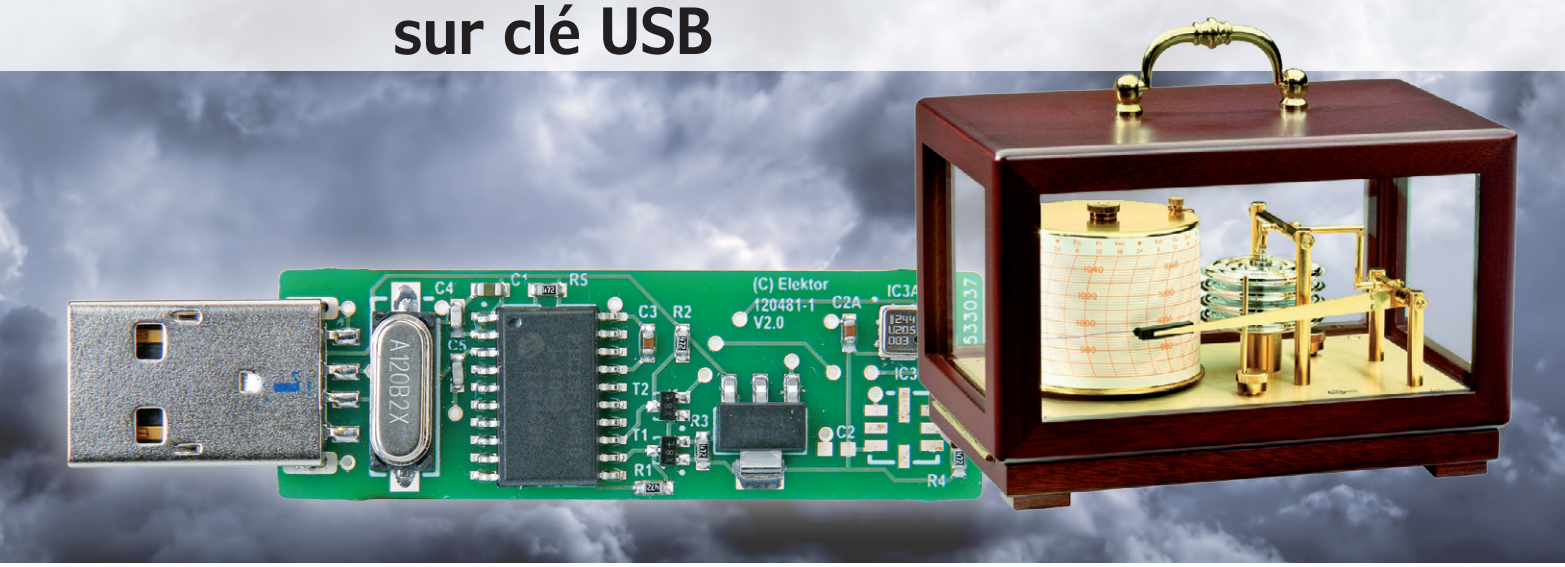
(120697 – version française : Kévin PETIT)

Figure 4. Oscillogrammes lorsque le capteur est à vide (en haut) et chargé avec un doigt (en bas), avec D = 10 mm et d_{x1} = 0.1 mm (isolateur en papier).

```
    BEGIN
        IF rising_edge(CLK) THEN  -- Création des bascules
            if counter < max_time then
                counter := counter + 1;
                if counter < zero_time then
                    dir <= '1';
                else
                    dir <= '0';
                end if;
            else
                dir <= '0';
                counter := 0;
            end if;
        END IF;

        SS <= b;

        IF( dir = '0') THEN
            if counter = zero_time then
                S <= 'Z';
            else
                if counter = read_time then
                    b <= S;
                end if;
            end if;
        ELSE
            if counter < one_time then
                S <= '1';
            elsif counter >= one_time and counter < zero_time then
                S <= '0';
            end if;
        END IF;
    END PROCESS;
END arh;
```

barostick

baromètre et thermomètre sur clé USB



Ruud van Steenis
(Pays-Bas)

Clés USB, clés du succès ? Elles sont partout, elles sont notre album d'images, de vidéos, de musiques préférées, d'articles, de fichiers et même de températures. Et la pression atmosphérique, y aviez-vous pensé ?
C'est fait : sur un baromètre sans mercure, avec un capteur *Bosch*, hectopascals et degrés Celsius rejoignent *Windows* pour se faire tirer le portrait.

Quelle que soit sa forme, la clé (dite aussi *stick*) USB est légère, pratique, plus facile à caser qu'une souris, mais dotée d'une mémoire d'éléphant. Elle passe sans gêne du PC au portable ou à la tablette et ne réclame que rarement un programme pilote. Depuis le temps, elle s'est acoquinée avec les instruments de mesure. C'est ainsi qu'un fabricant chinois fournit des clés pour la prise de température et d'humidité relative, des données qui intéressent beaucoup ceux qui veillent à leur confort. En revanche, rares sont les clés qui mesurent la pression atmosphérique (ou son dérivé, l'altitude[1]). Avec un capteur *Bosch*, il est simple de construire soi-même un baromètre USB doublé d'un thermomètre. On approche déjà de la station météorologique !
Le capteur utilisé dans le circuit proposé, dont le diagramme fonctionnel est à la **figure 1**, est un BMP085 [1], il est étalonné d'usine. Les valeurs d'étalonnage sont inscrites dans le BMP085, il faut donc les lire et les prendre en compte avant de prétendre à une mesure valable.
Il y aura donc encore des calculs à effectuer ? Non, parce que c'est une dll [2] qui s'en charge pour vous. De quoi vous simplifier la vie lors de l'élaboration maison du logiciel de votre baromètre à thermomètre intégré. Toutes les routines nécessaires sont bien sûr dans l'application correspondante pour la mesure et l'enregistrement de la pression comme de la température.

Le circuit

La communication avec le capteur se déroule selon le protocole I²C. C'est ici qu'il faut être attentif au fait que la tension sur les lignes I²C ne peut en aucun cas dépasser 3,3 V. Or, chacun sait que la norme USB prévoit une alimentation sous 5 V. Il faudra donc passer par un adaptateur de niveau.

On peut en revanche alimenter directement le petit processeur PIC avec le 5 V du port USB. J'ai choisi le PIC18F14K50 en CMS et boîtier SOIC. À côté d'un quartz CMS à 12 MHz et de ses condensateurs acolytes, il y a quelques condensateurs de découplage de l'alimentation et bien peu d'autres composants, comme vous le montre le schéma (**fig. 2**).
La tension d'alimentation de 3,3 V du BMP085 est fournie par un stabilisateur à faibles pertes du type AP1117. L'épineuse question du couplage bidirectionnel entre le PIC, qui travaille sous 5 V et le BMP085 qui ne peut supporter que 3,3 V a trouvé sa réponse, illustrée à la **figure 3**.

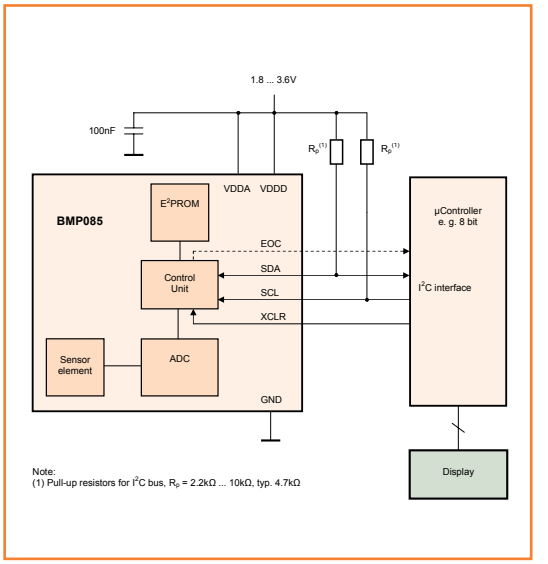


Figure 1.
Outre le capteur proprement dit, le BMP085 contient un convertisseur A/N, une EEPROM avec les données d'étalonnage et une interface I²C.

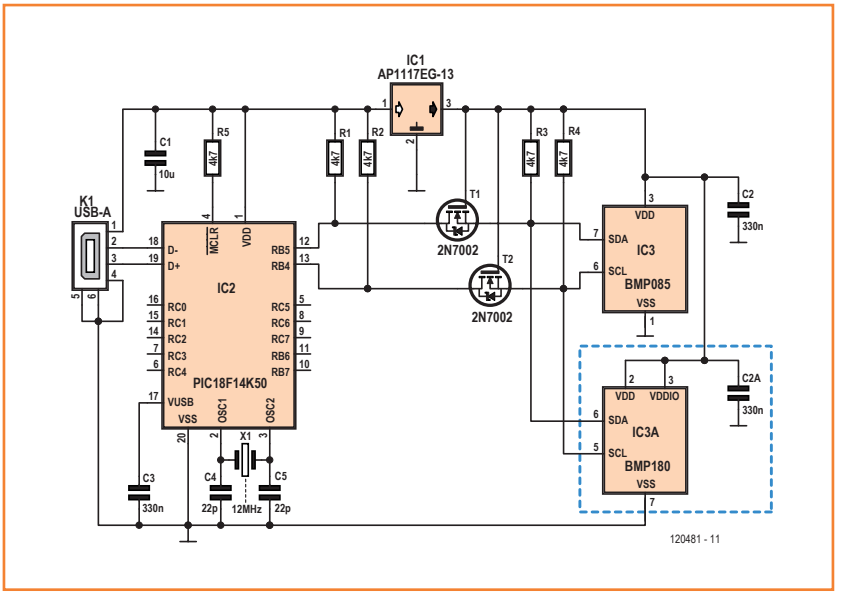


Figure 2.
Le schéma de l'instrument de mesure de la pression atmosphérique et de la température ne contient guère plus que le capteur Bosch et un microcontrôleur PIC.

Examinons les différents cas possibles.

- **1re situation :** côté 3,3 V, l'une des lignes n'est pas au niveau bas. La grille et la source du FET inséré dans ladite ligne sont soumises à du 3,3 V de part et d'autre, si bien que la tension V_{gs} est en dessous du seuil de conduction, le FET est donc bloqué. Le côté 5 V est haut en raison de la présence de la résistance de polarisation.

- **2e situation :** côté 3,3 V, une ligne est mise au niveau bas. La source du FET inclus passe aussi au niveau bas alors que la grille reste à 3,3 V. La tension V_{gs} devient supérieure au seuil du FET qui se met à conduire. La ligne de bus à 5 V est alors attirée également au niveau bas par l'intermédiaire du FET en conduction.

- **3e situation :** côté 5 V, la ligne est mise au niveau bas. Via la diode drain-substrat, le côté 3,3 V est d'abord mis au niveau

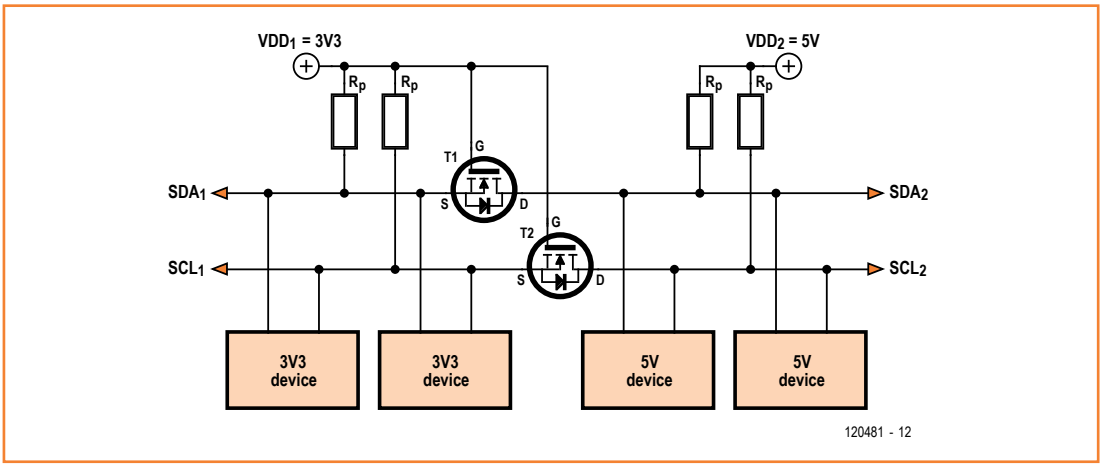


Figure 3.
L'adaptation de niveau entre le 5 V et le 3,3 V repose sur deux FET et quelques résistances de rappel au niveau haut.

Liste des composants

(toutes résistances et tous condensateurs au format 0603)

Résistances :
R1 à R5 = 4,7 kΩ

Condensateurs :
C1 = 10 µF/10 V X5R
C2, C3 = 300 nF
C4, C5 = 22 pF

Semi-conducteurs :
T1, T2 = 2N7002 (SOT232)
IC1 = AP1117E33G-13 (SOT-223)
IC2 = PIC18F14K50-I/SO (programmé, EPS 120481-41)
IC3 = capteur de pression atmosphérique BMP085 ou BMP180 (Bosch Sensortec)

Divers :
X1 = quartz 12 MHz (HC49US)
connecteur USB-A montage en surface (Lumberg 2410 07)
boîtier de clé USB (USB1SW, Conrad 531275-89)
circuit imprimé 120481-1
module assemblé et testé 120481-91 [3]
téléchargement gratuit du code source et du code hexadécimal, du logiciel pour *Windows* et de la DLL avec son code source [3]

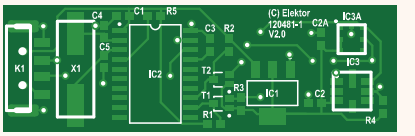


Figure 4.
Le circuit imprimé est prévu pour les deux types compatibles de capteurs *Bosch*.

bas. Ensuite le FET se met à conduire et du même coup, la tension sur la ligne est la même des deux côtés.

Dans le schéma de la figure 1, on voit qu'il y a deux capteurs, IC3 (BMP085) et IC3A (BMP180). La raison en est que le BMP085, bien qu'encore largement disponible chez les revendeurs, *Bosch* l'a déclaré obsolète. Son remplaçant direct est le BMP180, fonctionnellement compatible avec le 085, il est vrai, mais sous un autre boîtier et avec d'autres dispositions de raccordement. Le circuit imprimé a donc été dessiné de manière à convenir aux deux types. N'en montez toutefois qu'un seul, il suffira à la tâche ! Les modules actuellement disponibles auprès d'Elektor sont équipés du BMP085.

Figure 5.
La platine est disponible complète, assemblée et prête à l'emploi avec le boîtier approprié [3].

La construction du circuit

Le petit circuit imprimé conçu pour ce circuit, tout en étant compatible avec les deux types de capteurs, s'inscrit exactement, vous l'aurez deviné, dans les dimensions du boîtier d'une clé USB standard (**fig. 4**).
Tous les composants, y compris le connecteur USB, se soudent sur le côté cuivre. Ce sont des CMS, mais la plupart d'entre eux, on peut les souder manuellement sans difficulté, pourvu qu'on ait un minimum d'expérience. Le seul qui présente des difficultés, c'est le capteur, que ce soit le BMP085 ou le 180. C'est pourquoi il est préférable de commencer avec lui. La meilleure manière consiste à étamer au préalable les connexions utilisées, tant du capteur que du circuit imprimé. Ensuite, mettre exactement le BMP085 en position et chauffer les pistes de cuivre l'une après l'autre, éventuellement avec un léger apport de soudure. Avec le BMP180, il faut de la pâte à souder et un vrai four à refusion, parce que ses bornes sont cachées sous le boîtier. Les autres CMS se laissent souder avec un fer normal à panne fine.
J'ai réalisé sans difficulté différents prototypes avec ma méthode de soudage à refusion. Il est toujours bon de vérifier à l'ohmmètre si toutes les liaisons ont effectivement réussi.
Il convient de programmer le PIC au préalable. Le

code source pour le PIC en PICBasic et le fichier hexadécimal à programmer dans le PIC sont disponibles gratuitement au téléchargement sur la page du projet [3]. Un adaptateur SOIC spécial n'est pas indispensable pour programmer le PIC. J'ai utilisé un petit circuit imprimé d'adaptation de DIL vers CMS [4] à pistes dorées. Pour la durée de la programmation, je l'ai appuyé fermement sur la platine d'adaptation. Je n'ai pas eu de souci en pratique. Mieux vaut vérifier après l'opération si le contenu de l'EEPROM de programme dans le PIC est réellement celui du fichier hexadécimal. Elektor fournit d'ailleurs le PIC programmé pour ceux qui le désirent.
Quant à ceux qui ont envie d'un Barostick mais ne souhaitent pas le construire eux-mêmes, ils peuvent toujours commander à l'e-choppe [3] un module tout fait et testé, avec un boîtier à clipser par-dessus.

Le logiciel

Le projet est accompagné d'un programme sous *Windows* très détaillé qui traduit en graphiques les résultats de mesure de pression et de température. De plus, on peut enregistrer ces mesures quotidiennement et en faire varier l'intervalle (cf. **figure 5**).
On peut télécharger le logiciel pour *Windows* par le lien [3]. Celui qui veut le programmer lui-même disposera d'une dll (PSensor.dll). L'explication de son utilisation se trouve dans l'application DllDemo.exe en Delphi. Le code source de la dll est aussi disponible.

Chacun peut choisir selon ses goûts la langue, les couleurs, l'épaisseur des lignes, etc. Il est aussi possible d'envoyer par internet les images des cadrans de mesure de la pression et de la température à intervalles choisis vers un serveur FTP, par exemple vers un site de météorologie. Pour les amateurs d'interactivité, une fonction a été ajoutée qui permet de transmettre les résultats de mesure vers le site de météorologie *Weather Underground* [5].

Avant de brancher le Barostick pour la première fois sur le PC, il est prudent de mesurer la consommation sous 5 V qu'il soutirera de la prise USB. Si le courant est voisin de 10 mA, vous pouvez y aller. Peu après l'insertion de la clé, *Windows* vous avertira qu'il a détecté un nouveau matériel. Comme le Barostick se comporte comme un appareil HID, il n'est pas nécessaire de charger un programme pilote spécial et *Windows* affichera après quelques instants qu'il a trouvé un capteur de pression. Puis il vous indiquera ensuite que le nouveau matériel est configuré. Vous pourrez alors utiliser le logiciel d'application et le régler comme bon vous semble.

Il me reste à vous souhaiter de profiter pleinement… de nombreuses périodes de haute pression !

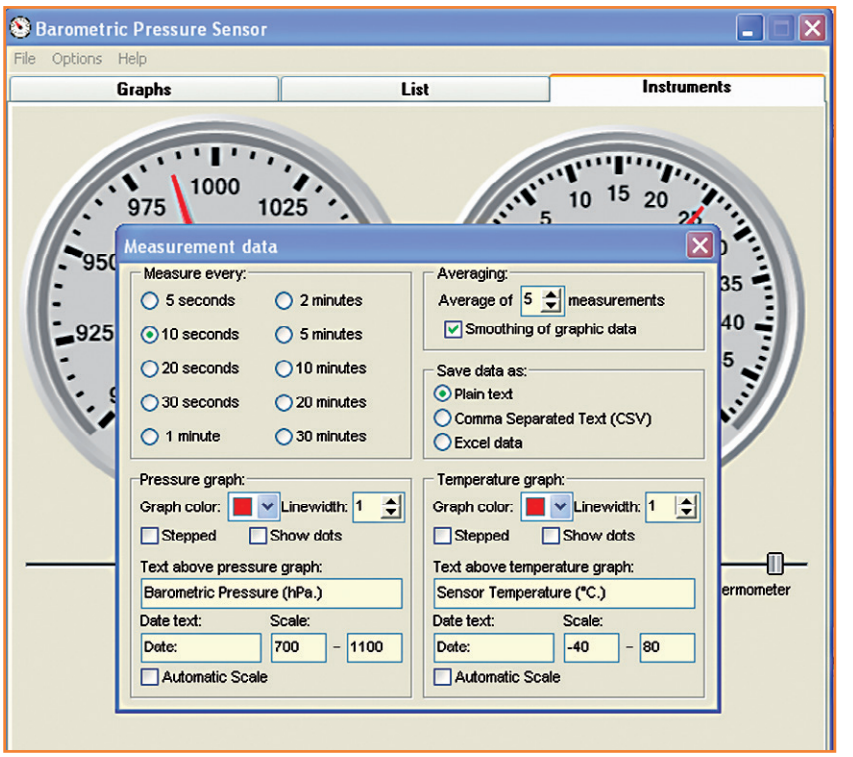
(120481 – version française Robert Grignard)

[1] L'altitude peut se dériver de la pression, une source d'inspiration pour le tourisme, l'alpinisme, l'ULM, une fusée expérimentale…

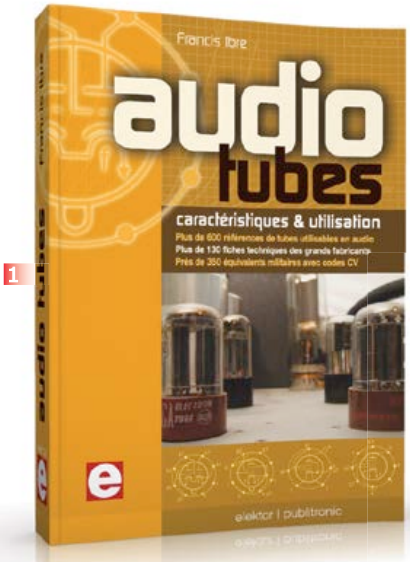
Liens

[1] www.datasheetarchive.com/barometric BMP085-datasheet.html
[2] www.property-protection.nl/Temper
[3] www.elektor.fr/120481
[4] www.voti.nl/common/pcb-12.jpg
[5] www.wunderground.com/

Figure 6.
Le programme pour *Windows* qui accompagne le projet offre une large gamme de possibilités de réglages.

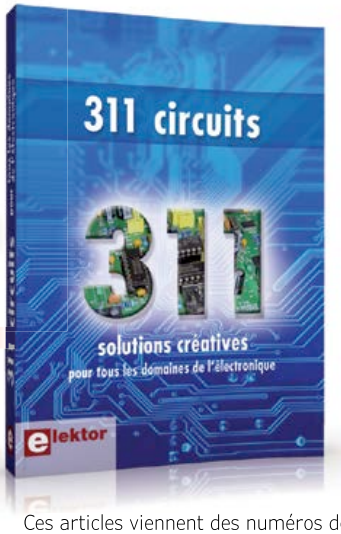


Elektor e-CHOPPE



1

3



2

4

Caractéristiques & utilisation

1 Audio tubes

- Plus de 600 tubes utilisables en audio (préampli, puissance, redresseur) dans 9 tableaux descriptifs (type, équivalence...)
- Plus de 130 fiches techniques des grands fabricants
- Près de 350 équivalents militaires avec codes CV
- Recommandations pour interpréter les fiches techniques et en tirer pleinement parti (test, mesure, remplacement des tubes)
- Exemples concrets de calculs, de schémas et d'illustrations

576 pages - ISBN 978-2-86661-174-3 – 60,00 €

Solutions créatives
pour tous les domaines de l'électronique

2 311 circuits

Cet ouvrage est un trésor : il réunit 311 schémas d'électronique analogique, logique ou numérique, des programmes, des liens vers des sites internet, des tableaux de caractéristiques de composants et des dessins de circuit imprimé. Il est le onzième volume de la collection « 3xx circuits ». Ses deux tables des matières alphabétique et thématique vous permettent de trouver rapidement et facilement parmi les 311 articles proposés ceux qui répondront à vos besoins. Ces articles viennent des numéros doubles récents de la revue Elektor. C'est une source d'inspiration inépuisable, et à partir de laquelle chacun élaborera ses propres variantes qu'il combinera ensuite à sa guise avec d'autres circuits. Tous les domaines familiers et usuels de l'électronique sont abordés.

448 pages • ISBN 978-2-86661-184-2 • 36,00 €

Kit complet avec circuit imprimé, afficheur et microcontrôleur

3 Détecteur de rayonnement

Il ne faut guère plus qu'une photodiode PIN et un amplificateur de capteur adapté pour mesurer un rayonnement radioactif. Nous présentons ici un préamplificateur optimisé couplé à un compteur particulier : un microcontrôleur qui gère la durée de la mesure et affiche la fréquence des impulsions en « counts per minute ».

Réf : 110538-71 – 39,95 €

Une carte compacte et bon marché qui vous initiera tout en douceur !

4 Embarquez Linux

Linux est partout, même dans certaines machines à café. Souvent, l'électronicien tenté d'adopter ce système d'exploitation est arrêté par sa complexité et par le prix des cartes de développement. Voici Linux pour les électroniciens, sous la forme d'une carte compacte et bon marché qui vous initiera tout en douceur !

Carte Linux Elektor (montée et testée)
Réf. : 120026-91 • 64,95 €

Incontournables microcontrôleurs, d'accord. Insurmontables microcontrôleurs, non !

5 12 applications pratiques pour maîtriser le PICBASIC PB-3B

Si avant d'utiliser un tel composant il faut apprendre l'assembleur ou le langage C, l'amateur ou l'électronicien débutant risquent de décrocher bien avant le stade des premières satisfactions, celui à partir duquel tout devient possible. Grâce à la simplicité des microcontrôleurs PICBASIC programmables en langage BASIC, l'électronique numérique programmable est désormais à la portée de tous. Ces douze applications pratiques du microcontrôleur PICBASIC PB-3B couvrent des domaines variés : la domotique, la protection des biens, la mesure, l'automatisation et l'électronique de puissance.

280 pages – ISBN 978-2-86661-166-8 – 43,50 €

Avec CD-ROM

6 Controller Area Network Projects

Ce livre (en anglais) s'adresse à tous ceux qui souhaitent en savoir (beaucoup) plus sur le bus CAN et ses applications : étudiants, ingénieurs en exercice, ou amateurs motivés. Il donne les principes de base des réseaux CAN et montre comment élaborer des projets avec des microcontrôleurs sur le bus CAN. Pour en profiter pleinement, il faut quelques connaissances de base en électronique. La compréhension du langage de programmation C est utile dans les derniers chapitres du livre ; connaître au moins un microcontrôleur de la série PIC sera un avantage, notamment quand vous passerez à la réalisation de projets à base de microcontrôleurs qui utilisent le bus CAN. **CD-ROM inclus** Ce livre comporte un CD-ROM gratuit qui contient une version de démonstration spéciale du compilateur mikroC.

ISBN 978-1-907920-04-2 • Langue : Anglais • 259 pages • 34,50 €

théorie & pratique – émission & réception

7 Les Antennes

Cette bible des antennes devient l'ouvrage de référence pour les radioamateurs, les techniciens et les ingénieurs. La première partie traite de la propagation des ondes dans l'espace et sur les lignes ainsi que des caractéristiques fondamentales des antennes (gain, rayonnement, courant, tension…). Cette étude théorique est suivie de réalisations pratiques, entre autres les antennes filaires, les antennes à gain, et les antennes à large bande et multibandes. La dernière partie est consacrée aux ultimes réglages : adaptation des impédances, appareils de mesure, conseils de sécurité (poussée du vent, résistance des matériaux, pylônes et haubans, foudre…).

470 pages – ISBN 978-2-86661-165-1 – 49,50 €

plus de 90 articles d'Elektor en français en bonus

8 DVD-ROM Wireless Toolbox

Sur ce DVD-ROM, vous trouverez des documents techniques et des outils pour libérer vos propres systèmes électroniques de leurs fils. Selon la distance à couvrir, le choix est vaste : quelques cm avec la communication en champ proche (NFC) et l'identification par radiofréquences (RFID), des dizaines de mètres avec Bluetooth, Wi-Fi et ZigBee, des milliers de km avec la réception de données par GPS. Le plus difficile est de s'y retrouver dans la jungle des normes. D'où l'utilité de ce DVD, avec sa collection de normes, de protocoles standard et propriétaires (p. ex. MiWi de Microchip, SimpliciTI de Texas Instruments), sa revue des plages de fréquences utilisées (ISM p. ex.) et bien davantage.

ISBN 978-90-5381-268-6 – 32,50 €

Tous les articles d'ELEKTOR de l'année 2012

en français, anglais, allemand, espagnol et néerlandais

9 DVD Elektor 2012

Ce DVD-ROM réunit tous les articles d'ELEKTOR, le mensuel d'électronique et de micro-informatique appliquées, parus au cours de l'année 2012. Il contient non seulement le texte des articles ainsi que les schémas, mais aussi tous les dessins des circuits imprimés, sous forme de fichiers à haute résolution.

ISBN 978-90-5381-273-0 • 27,50 €

Informations complémentaires et gamme complète sur :

www.elektor.fr/e-choppe

Elektor/Publitronic SARL
1, rue de la Haye – BP 12910 – 95731 Roissy CDG Cedex
Tél. : +33(0)1.49.19.26.19 – Fax : +33(0)1.49.19.22.37
@ : ventes@elektor.fr

de BASIC à Python (1)
récit d'une découverte

Jean-Claude Feltes
(Luxembourg)

Dans les annnées 1980, pour une raison que j'ai oubliée, *QuickBasic* était installé sur mon premier PC. C'est donc avec ce langage que j'ai commencé à me frotter à la programmation. Plus tard est arrivé le célèbre *Visual Basic*, et lorsque je me suis tourné vers Linux j'ai dû lui chercher un remplaçant. J'ai alors opté pour Python, un langage clair et précis, idéal pour l'électronicien.

Peut-être vous demandez-vous pourquoi s'intéresser ici à ce langage de programmation. Eh bien, Python permet par exemple la programmation de bas niveau des petits ordinateurs comme le *Raspberry Pi*. Python et électronique forment en outre un couple étroitement lié. C'est un collègue qui m'a initié à Python. Malgré les grandes différences avec *Visual Basic*, ce langage et son concept m'ont tout de suite séduit. Un code écrit en Python dégage une impression de clarté et de concision - ni accolades ni points-virgules, ces séparateurs qui m'ont toujours gêné en Pascal, C et Java. Les différences ne s'arrêtent pas là bien sûr.

BASIC vs Python

Python est un langage interprété et n'est donc pas associé à un compilateur. Ce point important présente aussi bien des avantages que des inconvénients. Un programme en Python est ainsi exécuté moins rapidement qu'un programme compilé, toutefois la performance des bibliothèques Python permet de ne pas trop se soucier de cette question. Autre inconvénient, l'interpréteur ainsi que d'éventuels modules doivent être installés sur le système cible.

Côté avantages, un programme Python peut être rapidement modifié, par exemple lorsqu'on souhaite accéder à une interface différente. Pour un changement mineur, disons la redéfinition d'une variable, inutile de s'embarrasser avec une interface utilisateur complexe, avec Python il est simple d'aller droit au code. Un code interprété permet même de définir des fonctions lors de l'exécution, par exemple pour programmer un traceur de courbe.

Python est un langage fortement orienté objet. Cet atout est toutefois moindre quand il s'agit d'écrire des programmes simples. La littérature et les programmes d'exemple usent abondamment de la programmation orientée objet, aussi est-il préférable de comprendre ses concepts si l'on ne veut pas décrocher rapidement.

Une chose à laquelle il faut s'habituer (mais qui est si pratique) est le minimalisme du langage : les blocs de code ne sont pas délimités par des accolades ou des mots-clés comme *begin* et *end*, mais par indentation des lignes. L'encadré **C, BASIC et Python** montre que cette méthode donne un code plus court.

Autres différences : contrairement à BASIC, Python est sensible à la casse ; Python peut manier des variables complexes, ce que les électroniciens apprécieront ; il existe des interpréteurs pour Windows, Linux et OS X.

Installation

Après avoir installé l'interpréteur sur votre SE, il pourra être judicieux de télécharger également

C, BASIC et Python

C

```
#include <math.h>
#include <stdio.h>

int main(int argc, char *argv[])
{
 printf ("Hello World\n");
 int i;
 int x;
 for ( i=0; i<11;i++)
   {
    if(i%2==0) {
      x = pow(i,2);
    printf("%d ^2 = %d \n",i,x);}
    else {
      x = pow(i,3);
     printf("%d ^3 = %d \n",i,x);}
   }
   return 0;
}
```

Quick/Visual Basic

```
Print “Hello world!”
   For x = 1 To 10
      If x Mod 2 = 0 Then
        Print x; “^2 = “; x^2
      Else
        Print x; “^3 =”; x^3
      End If
   Next x
```

En VB, Print doit être créée avec Debug. Print, et le code doit p. ex. être placé dans une procédure Form_Load().

Python

```
print "Hello world!"
for x in range(0,11):
    if x % 2 == 0:
        print  x, "^2 = ", x**2
    else:
        print  x, "^3 = ", x**3
```

quelques modules supplémentaires. La question qui se pose avant cela est : Python 2.x ou 3 ? Il manque encore hélas des bibliothèques importantes à la version 3. La version 2.7 a ma préférence, puisque les améliorations de la version 3 y ont été rétro-portées. Le **tableau 1** répertorie quelques modules d'intérêt ainsi que leurs adresses de téléchargement.
La version Windows comprend un installateur.

Pour les autres systèmes d'exploitation, procédez comme suit :

1. Décompactez l'archive téléchargée dans un répertoire temporaire.
2. En ligne de commande, entrez :
   python setup.py install

Le script Python copiera les fichiers nécessaires dans un sous-répertoire connu de l'interpréteur.

Tableau 1 : Modules et adresses de téléchargement

Python 2.7 Interpréteur	www.python.org/download/ *Python est déjà installé sur les systèmes Linux*
Numpy Calcul numérique et scientifique	http://pypi.python.org/pypi/numpy
Matplotlib Tracé de courbes	http://sourceforge.net/projects/matplotlib/files/matplotlib/matplotlib-1.1.0/
PySerial Accès à l'interface série	http://sourceforge.net/projects/pyserial/files/
PyParallel Accès à l'interface parallèle	http://sourceforge.net/projects/pyserial/files/
PyUSB Module USB	http://sourceforge.net/projects/pyusb/
WxPython Création d'interfaces graphiques	www.wxpython.org/download.php
Geany Éditeur avec coloration syntaxique	www.geany.org/

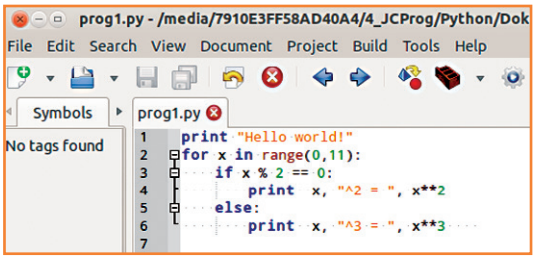


Figure 1.
L'éditeur gratuit Geany.

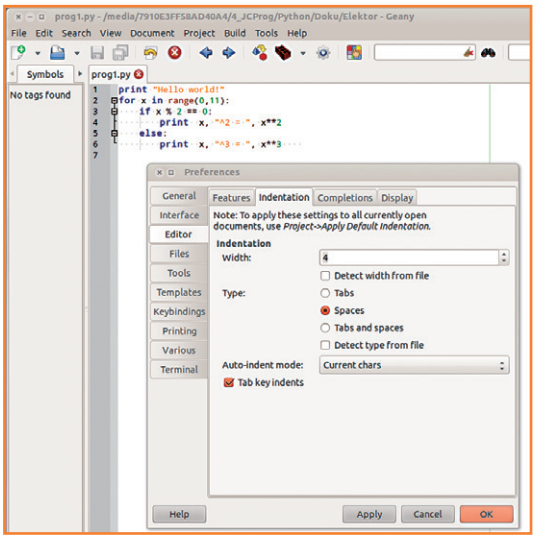


Figure 2.
En Python « correct », une indentation vaut quatre espaces.

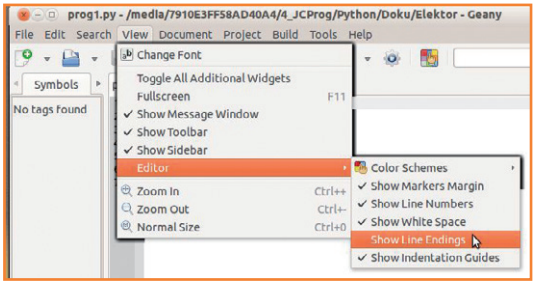


Figure 3.
D'autres paramétrages utiles sous Geany.

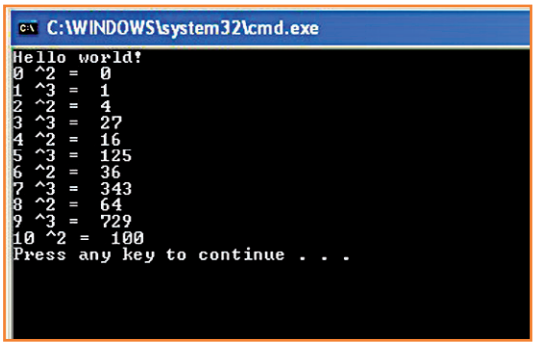


Figure 4.
Sortie du programme de test sous Windows.

Le chemin de ce répertoire dépend du système d'exploitation (*/usr/lib/python2.7* sous Linux, *\Python27\Lib* sous Windows XP).

EDI ou pas ?

Au début, un EDI comme celui de VB m'a manqué. Et puis je me suis rapidement fait à l'absence d'environnement, un bon éditeur de texte étant suffisant pour l'écriture de programmes simples. Mon préféré est Geany, dont il existe des versions pour Linux et Windows. La coloration syntaxique est automatique lorsqu'on ouvre ou sauvegarde un fichier d'extension *.py*. Bien pratiques aussi sont la fonction de repliage de code et la possibilité d'exécuter un script directement depuis l'éditeur.
Si vous êtes sous OS X, vous pouvez utiliser l'environnement de développement Xcode pour écrire du code Python. L'imposant paquet Xcode (1,65 Go) est gratuit et peut être téléchargé depuis l'App Store d'Apple. Vous verrez qu'un double-clic sur un fichier d'extension *.py* lance l'éditeur Xcode.

Un premier programme

N'importe quel éditeur de texte convient pour écrire un programme en Python. La **figure 1** montre un code ouvert dans *Geany*. En Python, l'indentation du code a un sens bien précis pour l'interpréteur ; l'utilisation d'espaces ou de tabulations est licite, mais pas leur mélange. La documentation officielle recommande quatre espaces pour l'indentation. La **figure 2** montre comment configurer *Geany* pour obtenir cette indentation. L'affichage de la marge des marqueurs, des numéros de ligne et des guides d'indentation (**fig. 3**) fournissent également une aide appréciable.
La **figure 4** montre le résultat sous *Windows* de l'exécution du code de test de la figure 1. Si vous ne lancez pas ce programme depuis *Geany*, exécutez-le depuis la ligne de commande avec : *python test.py*. Double-cliquer sur un fichier d'extension *.py* lancera éventuellement un script GUI (à interface graphique), mais s'il s'agit d'un programme texte comme *test.py*, la fenêtre se ferme dès que la fin du programme est atteinte, et il est impossible de voir sa sortie.
Sous Linux, faire précéder le code proprement dit des deux lignes suivantes est considéré comme une bonne pratique :

```
#!/usr/bin/env python
# -*- coding: utf-8 -*-
```

La première ligne appelle le programme *env* et lui demande où se trouve l'interpréteur *python* ; la seconde déclare le jeu de caractères utilisé, ici le codage UTF-8. L'encadré **Caractéristiques Python** recense d'autres caractéristiques du langage auxquelles les programmeurs BASIC devront s'habituer.

La console Python

Après avoir invoqué Python depuis un terminal Linux ou OS X, ou depuis l'invite de commandes Windows (raccourci Windows+R, puis tapez *python*), apparaît une invite « >>> ». Ces trois chevrons indiquent que l'interpréteur est prêt à répondre à toute commande. Rien de mieux pour découvrir les bases de Python. Testons p. ex. la fonction *upper()* :

```
>>> s = «hello»
>>> s.upper()
'HELLO'
```

Importer un module depuis la console est également possible :

```
import time
```

Pour obtenir de l'aide sur un module, appelez la fonction *help* : *help(module)* ; la commande *dir(module)* renvoie quant à elle les objets et méthodes associés à un module.

Erreurs de débutant

Les erreurs de syntaxe seront sans doute inévitables si vous avez l'habitude de programmer avec d'autres langages que Python. La plus classique est l'erreur d'indentation, que l'interpréteur signale par *Unexpected indent*. Ce message peut avoir pour origine un espace de trop ou manquant, ou encore un mélange de tabulations et d'espaces. Redisons-le ici, l'indentation standard est de quatre espaces.

Le débutant devra aussi rester attentif au type de ses variables, en particulier *float* et *integer* : 3.0/5.0 = 0.6, mais 3/5 = 0 !

Le formalisme Python impose de terminer les sous-programmes et méthodes par une parenthèse : on écrira ainsi *s.close()* pour fermer l'interface série ; cette instruction serait refusée par l'interpréteur si elle était écrite *s.close*.

Périphériques

L'électronicien a besoin d'un langage qui puisse accéder aux périphériques. Python dispose à cet égard de modules permettant la commande d'in-

Listage 1 : ScanSerial.py

```
import serial
def scan_serial():
    """ Scans for available serial ports """
portnames = []
# Windows
    for i in range(256):
        try:
            name = "COM"+str(i)
            s = serial.Serial(name)
            s.close()
            portnames.append(name)
        except:
            pass
# Linux
    for i in range(256):
        try:
            name = "/dev/ttyS"+str(i)
            s = serial.Serial(name)
            s.close()
            portnames.append(name)
        except:
            pass
# Linux USB
    for i in range(256):
        try:
            name = "/dev/ttyUSB"+str(i)
            s = serial.Serial(name)
            s.close()
            portnames.append(name)
        except:
            pass
    return portnames
#---------------------------------------------------------
# main
portnames = scan_serial()
for p in portnames:
    print p
```

Listage 2 : ReadSerial.py

```
"""Read and print serial data from COM1 (9600baud)"""
import serial
# init serial port COM1 / ttyS0
sCOM1 =serial.Serial(0)
sCOM1.setBaudrate(9600)
if sCOM1.isOpen()==False:
    sCOM1.open()
# read lines of data until user presses <Ctrl-C>
while(1):
    line = sCOM1.readline()
    print line
sCOM1.close()
```

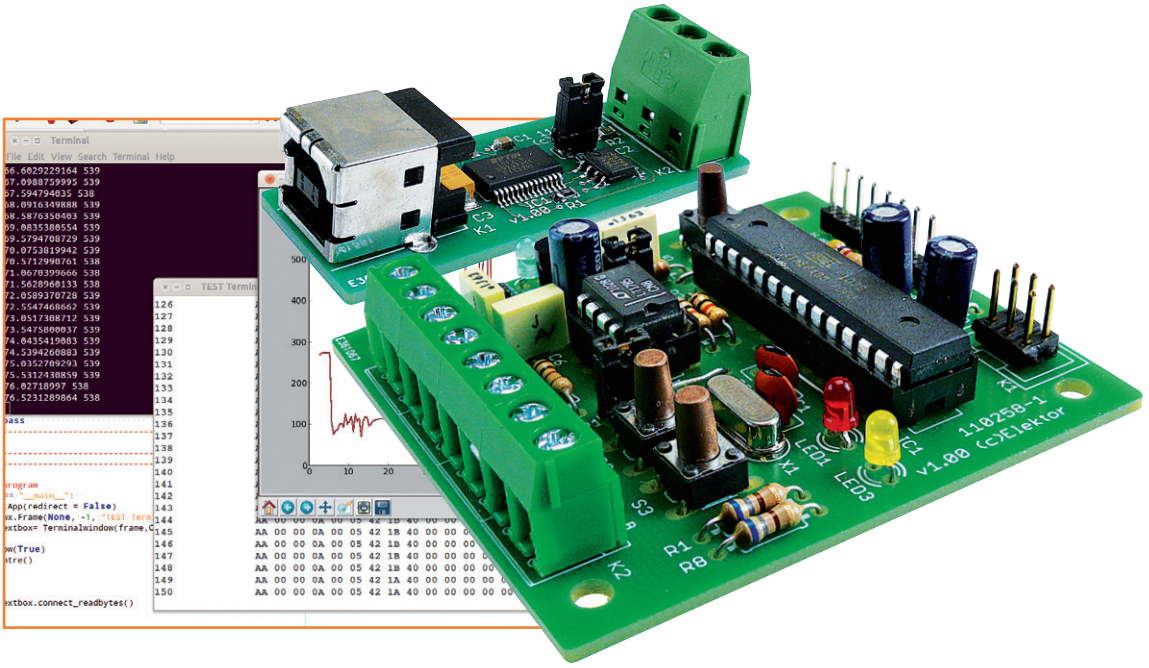


Figure 5.
Python peut également traiter des messages du bus Elektor contenant des valeurs de mesure.

terfaces PC : pyUSB, pySerial, pyParallel et pyI2C. Prenons l'exemple du module pySerial pour l'interface série : pour ouvrir un port, on crée une instance d'objet *Serial* auquel on affecte le port souhaité (p. ex. *mon_port = serial.Serial(0)* pour ouvrir le port 0). L'accès aux ports série se fait par numéro (0) ou par nom (COM1 sous Windows, ou /dev/ttyS0 sous Linux).

Pour trouver les ports disponibles, le plus simple est de tester leur instanciation (voir **listage 1**). Seuls les ports existants auront été instanciés avec succès. Cette méthode capture aussi les ports USB virtuels.

Comme vous le constaterez avec le **listage 2**, recevoir des données via l'interface série, p. ex. en provenance d'un microcontrôleur, est plutôt simple. Le port est ouvert après affectation du port COM1 à l'objet *sCOM1* et spécification du débit (bits de stop et de parité prennent ici leurs valeurs par défaut). Lors d'essais, il peut arriver que le programme se ferme avec un port encore ouvert. L'ouverture du port est alors impossible lorsque le programme est relancé. Il est donc préférable de n'ouvrir que les ports qui n'ont pas déjà été ouverts.

Le programme s'exécute dans une boucle sans fin plutôt déplaisante. La combinaison de touches Ctrl-C l'arrête, mais il serait plus élégant d'obtenir le même arrêt avec une touche. C'est faisable, mais en pratique la solution dépend du système d'exploitation. La fonction *raw_input()* ne devrait pas être utilisée, puisqu'elle attend obstinément la touche *Entrée*. Pour la gestion d'évènements comme l'appui sur une touche, des solutions simples existent du côté des bibliothèques de création d'interfaces graphiques, par exemple wxPython.

L'extension de code du **listage 3** crée un journal (*log file*) du port COM1. Les sous-programmes utilisés rendent le code plus lisible. Ils sont définis avec le mot-clé *def* et précèdent le programme principal. La fonction *show_log()* teste dans le bloc *try/except* l'existence du fichier *test.log*. S'il est présent, son contenu est lu et affiché. Appréciez la méthode *file.read*, qui permet de lire tout le contenu en une seule instruction. Le port série

Caractéristiques de Python

Les variables sont implicites, c.-à-d. déclarées par affectation : x = 5.0. Il n'existe pas de boucle du type *for/next*. Il est toutefois possible d'effectuer une itération sur le sous-objet d'un objet, p. ex. sur les caractères d'une chaîne ou les lignes d'un fichier. Comme équivalent de *for/next* on peut employer *for i in range*. Exemple :

```
for i in range (0,5):
    print i
```

itère sur l'ensemble des parties de l'objet *range(0,5)*, c.-à-d. sur les éléments de la liste [0,1,2,3,4]. Notez que la valeur de fin (ici 5) ne fait pas partie de la liste !

Les tableaux (listes) peuvent contenir des objets de types différents :

```
x = [0, 3.14, «Monty», «Python»]
```

Les fonctions peuvent retourner des multiplets :

```
(x, y, z) = myfunction(v)
```

Il n'existe pas de procédures, que des fonctions, comme en C. Une fonction est définie à l'aide du mot-clé *def* et peut retourner plusieurs valeurs ou objets.

est ensuite ouvert, chaque ligne reçue est affichée puis écrite dans le fichier.

Bilan & perspectives

Vous vous sentez soudain mordu de Python ? Bien, cette introduction devrait vous permettre d'installer l'interpréteur, des bibliothèques, et de rapidement écrire quelques programmes simples. Les listages peuvent être téléchargés depuis [1]. Le deuxième volet de cette série intéressera l'électronicien avec des sujets subtils comme les diagrammes et la synthèse de Fourier, ainsi qu'une belle interface utilisateur. Le troisième volet sera pratique. Nous utiliserons une petite carte à microcontrôleur et le bus Elektor pour envoyer des valeurs de mesure à un PC (**fig. 5**).

(110483 – version française : Hervé Moreau)

Liens & références

[1] Listages etc. : www.elektor.fr/110483

[2] Page de l'auteur (en allemand) : http://staff.ltam.lu/feljc/home.html

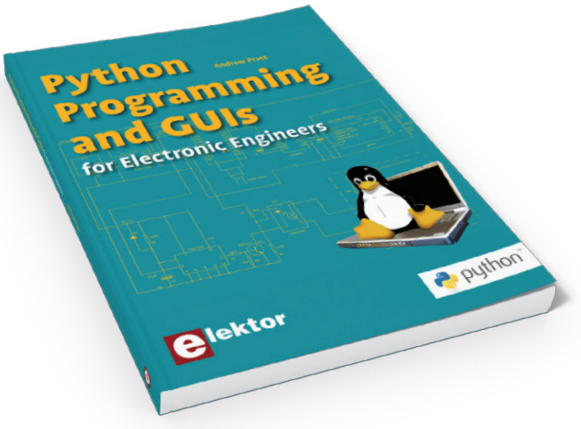
[3] Documentation Python : https://pypi.python.org/pypi/RPi.GPIO

[4] Tutoriels Python : www.awaretek.com/tutorials.html

[5] Références : Michael Weigend : Python gepackt

[6] Introduction : J.M. Hughes: Real World Instrumentation with Python

[7] Python pour l'électronicien : Andrew Pratt : Python Programming and GUIs for Electronic Engineers www.elektor.fr/python-programming



Listage 3 : ReadSerial2.py

```
"""Read and print serial data from COM1 (9600baud)"""
import serial

def show_log():
    """ show result of last logging"""
    try:
        file  = open("test.log", "r")
        text = file.read()
        file.close()
        print "CONTENTS OF LAST LOG FILE:"
        print text
        print "END OF LOG FILE"
    except:
        print "NO LOG FILE FOUND"
        pass

def open_port_log(port):
    """ open serial port and LOG file"""
    # init serial port COM1
    sCOM =serial.Serial(port-1)
    sCOM.setBaudrate(9600)
    if sCOM.isOpen()==False:
        sCOM.open()
    file = open("test.log", "w")
    return sCOM, file

def receive(sCOM, file ):
    """ read lines of data until user presses <Ctrl-C>"""
    while(1):
        line = sCOM.readline()
        print line
        file.write(line)

port=1
show_log()
ok = raw_input("NEW LOG (y/n) ?")
if ok== "y":
    s, f=open_port_log(port)
    receive(s, f)

# file and port closing done by interpreter at <Ctrl-C>
```

À propos de l'auteur

Jean-Claude Feltes enseigne l'électronique au Lycée Technique des Arts et Métiers de Luxembourg, une école publique qui forme des artisans, des artistes, ainsi que des techniciens supérieurs. Durant son temps libre, Jean-Claude se passionne aussi pour l'électronique et l'informatique (cf. [2]).

bibliothèque de micrologiciels embarqués EFM

accélérez l'aboutissement de vos propres projets !

Jens Nickel (Elektor)

Une nouvelle bibliothèque pour le *vite fait, bien fait* ? Pas seulement ! Avec elle, vous rendrez la programmation de vos projets et modules indépendante du matériel et donc portable sur d'autres plateformes. Restez libre du choix de la marque du contrôleur et accédez au Linux embarqué ! Avec le renfort de diverses fonctions pour CAN, MLI, écran, TCP/IP, cartes SD et plus encore, cette bibliothèque s'adresse aussi aux débutants en C.

Caractéristiques de la bibliothèque de micrologiciel embarqué EFL

- adaptable en principe à tout contrôleur pour lequel un compilateur C ANSI existe
- on recommande au moins 32 Ko de flash et 2 Ko de RAM
- API bien documenté
- accepte actuellement les contrôleurs ATmega328 et Xmega256A3 ; ATmega32 en préparation
- accepte actuellement les cartes de nœud expérimental ElektorBus, Xmega-WebServer [9], Extension Linux [8] ; d'autres en préparation !
- adaptation facile à d'autres cartes
- bibliothèques supérieures actuellement disponibles : LEDButton, ADCMux, StepperMotor, ElektorBus, Display, module WizNet-TCP/IP, carte SD (données brutes)
- source ouverte licence LGPL

En novembre dernier, nous avons présenté une bibliothèque C pour le noeud d'ElektorBus [1]. L'un de ses atouts réside dans la séparation entre partie liée au matériel et module de code responsable du protocole. C'est ce qui permet d'adapter vite la bibliothèque à d'autres types de contrôleur et de cartes – il suffit d'échanger la couche matérielle – et la bibliothèque propre à l'ElektorBus reste inchangée.

Dans une application de bus, on ne se contente pas de coder des infos et de les communiquer par le bus RS485. Il y a lieu de commander des LED et des relais, de lire des valeurs analogiques de capteurs notamment et d'actionner des moteurs. Devant cette complexité, il est bon de pouvoir, sans trop de peine, établir des relations avec

d'autres puces, voire avec un module TCP/IP ou une carte SD.

Programmer en communauté

Tout en développant, des idées surgissent qui, de fil en aiguille, amènent davantage de flexibilité et de puissance, largement au-delà du projet d'ElektorBus. J'ai ainsi eu la chance d'obtenir le concours d'un de nos lecteurs, Michael Busser, un membre de l'ElektorBus-Google-Group, qui a contribué au développement d'un module de code plus évolué avec des fonctions et des idées novatrices.

Pour pouvoir, à nos moments perdus, travailler ensemble au code, nous avons adhéré à la célèbre plateforme de projets SourceForge [2] dans une initiative du nom d'Embedded Firmware Library (EFL). SourceForge est réservé aux projets à source ouverte, ce qui cadre exactement avec l'esprit d'Elektor. Comme licence, nous avons choisi LGPL [3]. La caractéristique majeure de la plateforme de projets est un système intégré de gestion de version [4], compatible SVN, utilisable sous Windows avec TortoiseSVN [5] p.ex. D'un clic de souris, vous y postez (*Commit*) vos modifications ou vous téléchargez la version la plus récente du code (*Update*).

Avec le temps, nous avons pu réunir une collection considérable de fonctions, nous avons aussi découvert et mis à profit un système automatique de documentation. Quand on veut léguer des commentaires dans le code sous une notation définie, on peut documenter automatiquement une quantité impressionnante de fonctions au moyen du programme à source ouverte Doxygen [6], basé sur HTML et exploitable dans un navigateur (**fig. 1**). C'est réellement fantastique ! Pour vous en faire une idée, téléchargez la documentation en Doxygen d'EFL sur le site du projet [7].

Un système modulaire

La bibliothèque de micrologiciel embarqué est de conception modulaire, chaque module d'une couche d'abstraction (*Layer*) est interchangeable. Il y a trois couches matérielles, la plus basse accueille le code du contrôleur, la suivante, celui de la carte et par-dessus, la couche d'extension (cf. **fig. 2**), une structure qui répond aux exigences d'un projet réel. On peut échanger différents contrôleurs de brochage compatible sur la même carte, mais aussi utiliser un contrôleur sur différentes cartes. Un bel exemple en est l'ATmega328, que l'on rencontre sur des circuits imprimés d'Elektor, mais aussi sur la carte Arduino Uno.

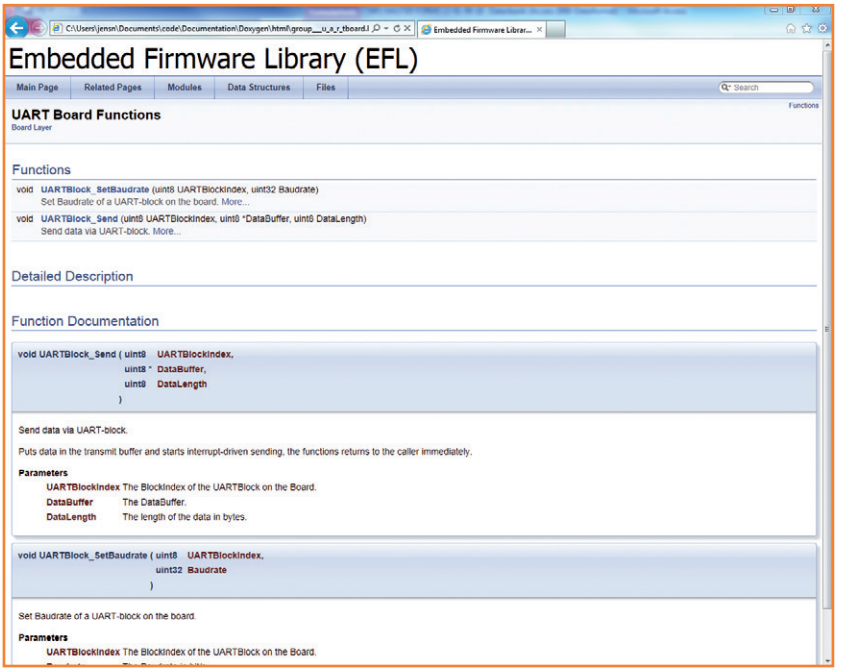


Figure 1.
L'outil Doxygen assure une documentation concise des fonctions.

Quand on spécifie un brochage déterminé pour un connecteur d'extension, on peut le combiner avec différentes cartes de contrôleur et des modules d'extension. Vous en trouverez un exemple avec la carte d'extension Linux dans ce numéro [8]. Cela ne s'applique pas seulement à la carte Elektor Linux, mais aussi au Xmegaweb Server que nous

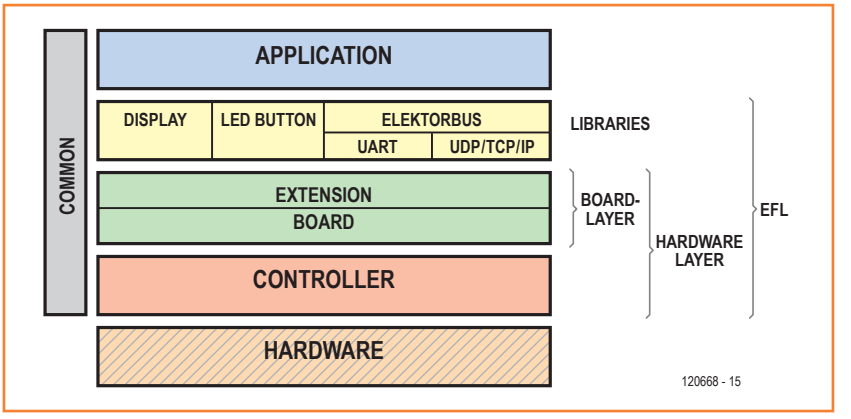


Figure 2.
Grâce à une couche matérielle nettement séparée, le projet EFL s'adapte facilement à d'autres plateformes. Très utile quand il faut continuer le développement du matériel lors du prototypage.

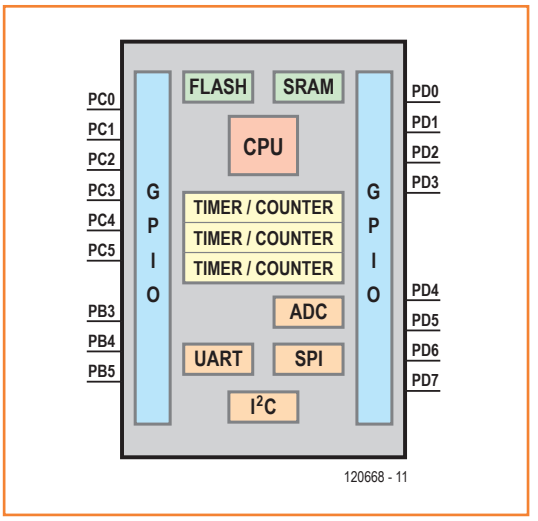


Figure 3.
Diagramme fonctionnel très simplifié d'un contrôleur comme l'ATmega328.

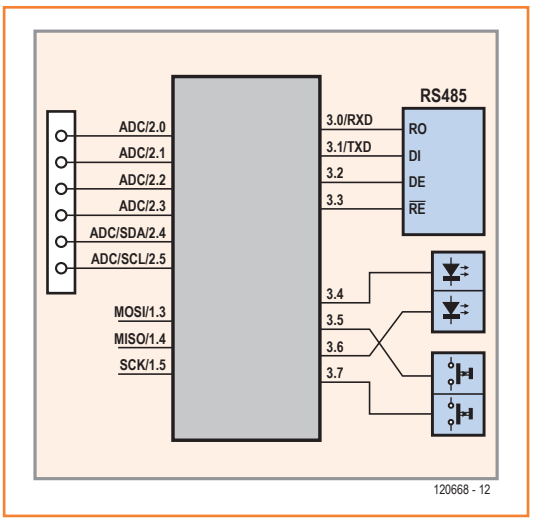


Figure 4.
Pour le développeur du code de la carte, le contrôleur est une boîte noire, il ne doit en connaître que les connexions *(Portpin)*.

vous présenterons en juin (grande photo). Une carte mère ou de base, avec divers périphériques et sur laquelle viennent se connecter différentes platines à processeur, c'est une autre application d'EFL.

Pour le contrôleur, la carte du contrôleur et celle d'extension, il y a trois modules de code différents, constitués d'un fichier d'en-tête Headerfile .h et d'un fichier Codefile .c. Il faut donc chaque fois inclure dans son projet les fichiers correspondants : ControllerEFL.h/.c, BoardEFL.h/.c et ExtensionEFL.h/c. Les noms de fichier sont identiques pour toutes les cartes et pour tous les contrôleurs pris en charge par EFL. Afin d'éviter les conflits de noms, les fichiers sont stockés dans [11] une base de code téléchargeable en sous-dossiers du nom de la carte ou du contrôleur concerné. Des infos complémentaires sont disponibles dans un document annexe sur EFL via [7].

Le principe de la boîte noire

Dans le cas idéal, les trois modules de code pour le matériel peuvent provenir de la plume de trois concepteurs différents. Il est clair, dans ce cas, que celui qui développe le code contrôleur ne peut pas savoir d'avance sur quelle carte son contrôleur viendra s'implanter, pas plus que le développeur de la carte, quelles extensions y seront ajoutées.

C'est aussi vrai dans l'autre sens, du haut en bas, le développeur n'a pas à se préoccuper des détails du code utilisé dans la couche inférieure. Le concepteur du code de la carte ne doit connaître que le schéma de la carte du contrôleur, avec les notations des broches de port et de leurs fonctions. Il ne doit rien savoir des registres ni des détails internes du contrôleur (**fig. 3**). Pour lui, le contrôleur est une *Black Box* (**fig. 4**).

Idem pour le concepteur de la carte d'extension. Il ne veut que le brochage du connecteur d'extension dans lequel sa carte devra s'embrocher et du coup assurer le contact avec le contrôleur. Mais du câblage de la carte contrôleur, donc à quelle broche de connecteur correspond celle du port du contrôleur, il n'en a que faire (**fig. 5**).

Les bibliothèques d'en haut

Au-dessus de la couche matérielle suivent des modules qui offrent de plus hautes fonctions de bibliothèque (*Libraries*, en langage EFL). Par exemple, le code de base contient déjà un module de commande d'un afficheur de texte compatible avec le HD44780, un pilote pour un module TCP/IP de WizNet, une commande de moteur pas à pas ainsi qu'un module de lecture et d'écriture de données brutes sur carte SD. Le principe de la boîte noire reste en vigueur. Pour modifier ou étendre la bibliothèque des affichages, vous pouvez considérer l'écran, ou mieux le contrôleur d'affichage, comme un bloc dans le diagramme du circuit imprimé (**fig. 6**). Nul besoin de savoir quel fil de l'écran ira à telle broche de port. Pour le concepteur de la bibliothèque d'affichage peu importe que l'écran soit attaqué en mode à 4 bits ou par bus SPI. Il peut consacrer toute son attention à l'assemblage des commandes pour le HD44780 avec les données.

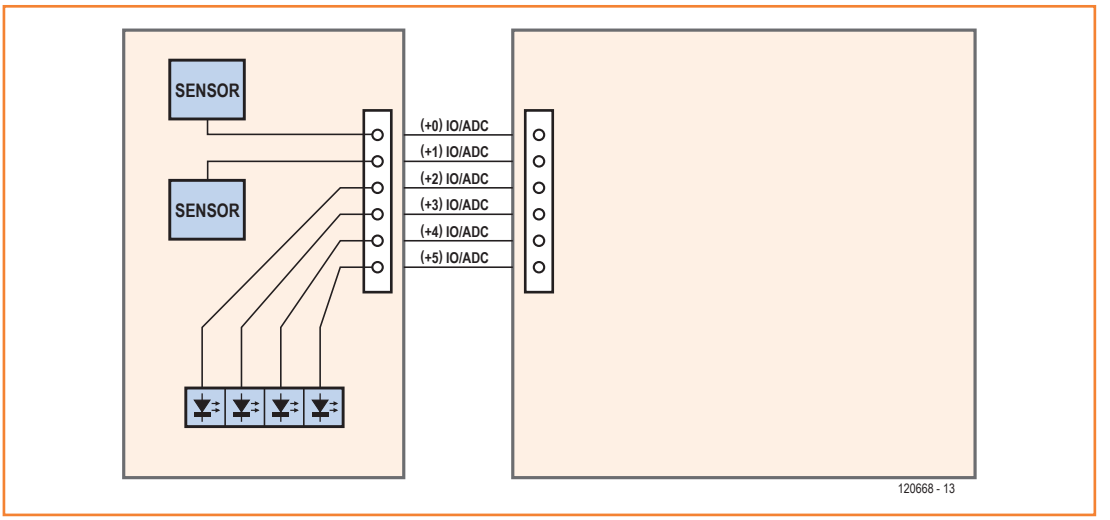


Figure 5.
Le développeur du code de la carte d'extension ne doit s'intéresser qu'aux raccordements du connecteur d'extension et à leur utilisation possible.

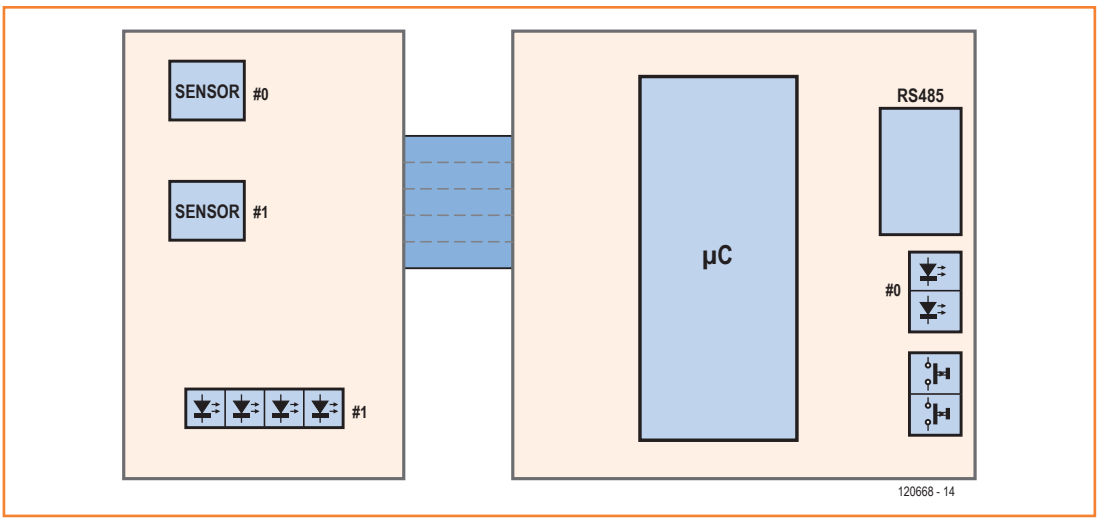


Figure 6.
Un diagramme fonctionnel suffit au développeur d'application, il n'a pas besoin de connaître le câblage sur la carte. Les adresses des deux blocs de LED sont les numéros de bloc #0 et #1.

Le code des applications

Le développeur d'application accède aisément à tous les blocs de la carte, il lui suffit de savoir que ces unités fonctionnelles sont disponibles. Il se peut aussi qu'on rencontre un bloc de LED sur une carte contrôleur ainsi que sur une carte d'extension, p.ex. un bloc de deux et un autre de quatre LED (cf. fig. 6). Le développeur peut alors commuter la 1re LED dans le bloc 0 et la 3e dans le bloc 1, sans avoir à se tracasser de savoir sur quelle carte sont installées ces LED.
Les LED, boutons, relais et autres unités commandées en numérique, on peut aussi les traiter avec les mêmes fonctions de la bibliothèque si elles ne sont pas directement reliées à une ligne de port, mais bien à une expansion de port. Les poussoirs branchés sur une entrée analogique, comme sur la carte d'extension Linux, peuvent être considérés comme des boutons numériques (grande photo).

Ensemble de fonctions uniformes

On ne peut décrire ici que de façon rudimentaire comment tout cela fonctionne en interne (voyez les encadrés). Dans le document annexe [7], on trouve beaucoup d'autres infos sur le matériel. Elles sont surtout utiles si vous avez à composer votre propre couche matérielle pour une carte qui n'est pas encore documentée.
L'idée conductrice de toute la bibliothèque est chaque fois de composer un ensemble uniforme de fonctions (API) que la couche inférieure met à disposition des couches supérieures. L'API du contrôleur contient toujours les mêmes fonctions, à condition d'être pris en charge par le contrôleur. Il s'agit ici de mettre en place

Tableau : Fonctions du contrôleur
(état en février 2013, sans fonctions d'initialisation ni de paramètres, mises à jour sur [11])

Fonction	Xmega256A3	ATmega328
interrupts on/off	X	X
Wait… (retard)	X	X
définir la broche	X	X
lire la broche	X	X
lire la valeur CAN	X	X
lancer la conversion CAN	X	X
définir valeur CAN	X	-
TX données UART (interrupt acrivé)	X	X
RX tampon en anneau	X	X
SPI-Master (envoyer / recevoir octet)	X	*
I2C-Master		
écrire / lire octet (combiné)	X	*
démarrer/arrêter timer	X	X
compte sur une broche	X	*
activé par timer :		
- incrémentation	X	X
- appel de fonction	X	X
- basculer une broche	X	X
- basculer avec variable vitesse	X	X
- mesure de fréquence	X	*
sortie MLI	X	*
logiciel MLI	*	*

X = présent * = en projet - = non disponible

d'abord des fonctions en préparation pour les unités fonctionnelles (*Units*) des contrôleurs comme CAN, SPI, UART, temporisateur, compteur et autres. Ensuite viennent les fonctions pour définir ou lire les paramètres, le débit binaire d'une unité UART, p.ex., ainsi que les fonctions de commande proprement dite, telles que lecture d'une valeur sur une broche de CAN, envoi de quelque chose sur un UART ou une interface SPI, etc. Le **tableau** vous en dira plus.
À propos de la carte (et son module d'extension), on trouve de fonctions de bas niveau pour des unités séparées comme afficheurs, moteurs pas à pas, etc. Ces fonctions sont nécessaires du fait que les différents câblages et adressages (p.ex. 4 bits ou SPI) doivent être encapsulés. Les bibliothèques supérieures ont alors accès à ces fonctions de bas niveau et, à leur tour, remettent à disposition le code d'application proprement dit dans un ensemble de fonctions.

Un exemple : l'affichage

Examinons, à titre d'exemple, la commande d'un affichage de texte. Le code d'application doit écrire dans la 1re ligne de l'écran un mot de salutation et appeler les fonctions suivantes de la bibliothèque de présentation :

```
Display_LibrarySetup();
Display_WriteString(0, 0, «Welcome!»);
```

Le premier 0 indique le numéro de l'afficheur (comme en C, toutes les numérotations commencent à zéro en EFL). Pour exécuter l'ordre d'écriture, la bibliothèque d'affichage appelle plusieurs fois la fonction

```
void Display_SendByte(uint8
DisplayBlockIndex, uint8 ByteToSend,
uint8 DATABYTE_COMMANDBYTE)
```

au niveau matériel, qui chaque fois transmet une donnée ou un octet d'instruction au contrôleur d'affichage.

Dans le cas de la carte Xmega-Webserver, la liaison à l'écran se fait par SPI, la fonction est alors écrite pour que l'appel

```
uint8 SPIMaster_TransceiveByte(int8
Handle, uint8 Databyte)
```

soit relayé au niveau du matériel. Les entrées supplémentaires CS et RS servent à l'écran.

Le téléchargement du logiciel

Dans le téléchargement du logiciel relatif à cet article [7] se trouve un exemple d'application. Il s'agit d'un projet similaire à celui de [1]. Le nœud expérimental [10] de l'ElektorBus est couplé à l'aide d'une carte d'extension porteuse de deux capteurs et de quatre LED supplémentaires (**fig. 7**). Ces composants ont été installés sur un demi-circuit imprimé ELEX, mais un fichier LochMaster est également joint au projet.
Si vous ouvrez le logiciel avec Studio 6 d'Atmel, vous verrez que les fichiers de code sont logés dans des sous-dossiers (virtuels) spécifiques, à savoir Matériel, Commun et Bibliothèques (voir **fig. 8**). De plus, on reconnaît dans le fichier principal (ExperimentalNodeEFL.c) la base fondamentale d'une application EFL. On peut y voir entre autres quelles fonctions il faut appeler au démarrage du programme.

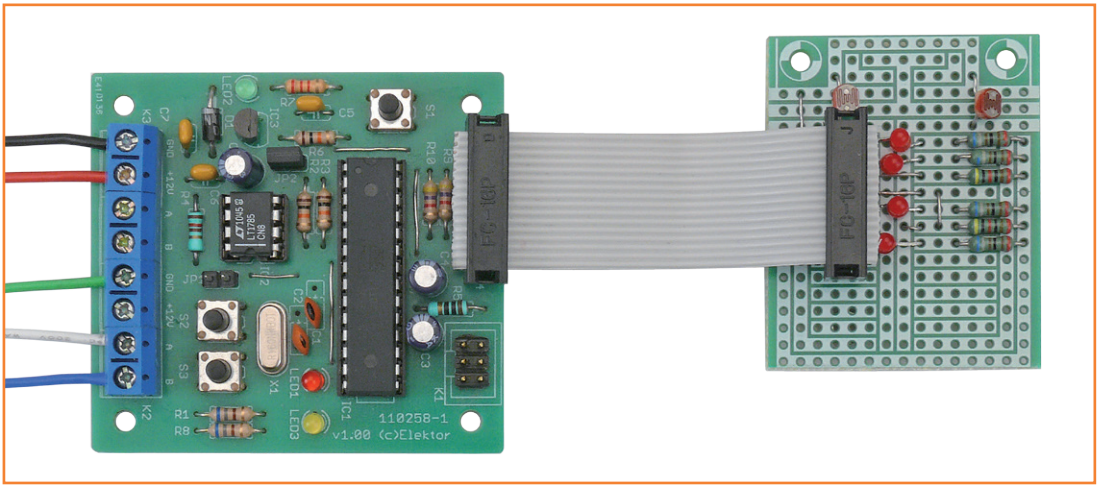


Figure 7.
Le nœud expérimental ElektorBus avec carte d'extension (deux capteurs et quatre LED). La couche d'abstraction de la carte s'adapte rapidement à une autre, comme la prochaine carte de moteur pas à pas.

Dans la fonction ApplicationSetup(), on prépare d'abord les données de configuration pour le bus. Plus besoin d'un fichier spécial de configuration ElektorBus.h / .c). Ensuite viennent les fonctions d'initialisation pour la bibliothèque du module, dont les paramètres ne doivent normalement pas changer. Avec `SwitchLED(1, 0, ON);` on allume la 1re LED sur la carte d'extension comme témoin du démarrage du programme.
Appuyer sur le premier bouton du nœud expérimental fait basculer la LED rouge et la 3e LED sur la carte d'extension. Sur demande

Un coup d'œil sous le capot de l'EFL

Broches de port

Les fonctions d'E/S numériques (p.ex. fixer un niveau sur une broche) en code contrôleur sont appelées avec un paramètre « Portpin », un entier non signé de deux octets (uint16) :

```
void IO_SetPinLevel(uint16 Portpin, uint8 PinLevel)
```

Le premier octet de Portpin indique le port (le groupe de broches) ; sur les contrôleurs AVR, au port A est attribué le numéro 0, au port B le numéro 1, etc.
C'est sur cette valeur que la fonction calcule le registre *ad hoc*. Le deuxième octet désigne la broche parmi celles du port ; sur un contrôleur à 8 bits la valeur va de 0 à 7. On peut ainsi les indiquer sur le schéma par 0.1 0.2 … 1.0 1.1 etc. comme à la fig. 4.

Fonctions, unités et données opaques
(Features, Units & Handles)

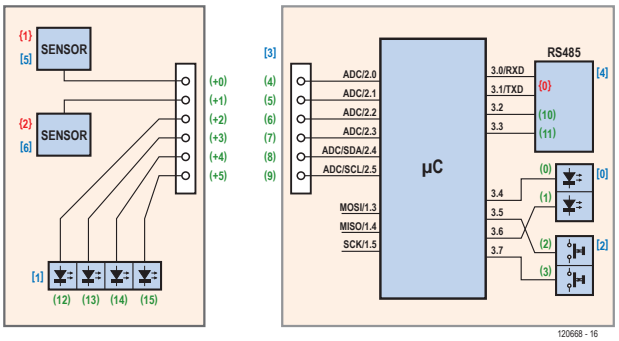
Les différentes fonctions qu'un contrôleur propose (CAN, UART, SPI, temporisateur, compteur, etc.), EFL les appelle ***Features***. D'une pareille fonction, comme SPI, il peut y en avoir plusieurs unités, ainsi le Xmega256A3 en dispose de 3. Sur chacun de ces ***Units***, on compte un certain nombre de broches de port (cf. cahier de caractéristiques du contrôleur utilisé). Mais le code de la carte peut ignorer la numérotation des unités et des registres qui en font partie. Lors de l'initialisation (*setup*) de l'unité par une fonction de la couche matérielle du contrôleur, une des broches du port est transmise comme paramètre, p.ex. avec un SPI, :

```
int8 SPIMaster_Setup(uint16 Portpin)
```

La fonction *Setup* dans le code du contrôleur repère alors, d'après la broche de port, le numéro de l'unité de l'interface SPI sollicitée, avec lequel le contrôleur peut calculer le registre matériel SPI correspondant. Pour les accès ultérieurs, le numéro de l'unité et celui de la broche de port sont stockés dans une table du nom de *Map*. Dans la suite, la fonction SPIMaster_Setup rendra l'indice de cette entrée comme ***Handle*** (0…).
C'est au moyen de cette **donnée opaque** que le code de la carte pourra s'adresser à l'interface SPI voulue pour envoyer un octet et en recevoir :

```
uint8 SPIMaster_TransceiveByte(int8 Handle, uint8 Databyte)
```

Et tout marche pareillement avec les autres *Features*..



Broches de carte et blocs

Chaque broche de port du contrôleur est reliée à des unités fonctionnelles sur la carte, p.ex. des rangées de LED ou de boutons, un pilote RS485. Les broches de port de ces différents « blocs » sont introduites successivement dans une table par Board_Init dans le code de la carte. Les indices de ces tables s'appellent *Boardpin*.
Pour le bloc de LED de la figure 4, on attribuerait p.ex. Boardpin (0) et (1), pour le bloc de boutons, Boardpin (2) et (3) et pour les broches du connecteur pour carte d'extension, Boardpin (4) à (9) :

Boardpin	Portpin
(0)	3.4
(1)	3.6
(2)	3.5
(3)	3.7
(4)	2.0
(5)	2.1
(6)	2.2
…	
(9)	2.5

La fonction Extension_Init dans le module de code d'extension réserve pour ses propres LED (figure 5) d'autres broches de carte, p.ex. de (12) à (15). Les broches de port correspondantes 2.2 à 2.5 sont ainsi atteintes indirectement en passant par les entrées de Boardpin des connecteurs d'extension (6) à (9) :

```
…
(12)  2.2      // = (6)
(13)  2.3      // = (7)
(14)  2.4      // = (8)
(15)  2.5      // = (9)
```

Dans la table de Boardpin se trouvent déjà deux blocs pour LED : (0), (1) et (12) à (15). Ces nombres, plus précisément le premier Boardpin et le numéro de LED dans le bloc, on les stocke pour chaque bloc dans une table séparée du nom de « Block », avec le type de bloc :

BlockIndex	BlockType	First	Count
[0]	BLOCKTYPE_LED	0	2
[1]	BLOCKTYPE_LED	12	4
[2]	BLOCKTYPE_BUTTON	2	2
…			

Si une bibliothèque de LED veut atteindre une LED en particulier dans un bloc donné, elle appelle la fonction

```
void SwitchDigitalOutput(uint8 BlockIndex, uint8 Position, uint8 ON_OFF)
```

de la couche de la carte.
La fonction `SwitchDigitalOutput(…)` repère le Boardpin adéquat au moyen du paramètre BlockIndex et de la position de la LED dans le bloc, puis enfin le Portpin.
Dans la table de bloc, on mémorise également si la LED à allumer a besoin d'un niveau haut ou bas, ce qui peut varier d'une carte à l'autre.

Ressources

L'énorme souplesse d'EFL a aussi un prix : outre la mémoire flash pour le code du programme, la bibliothèque couvre aussi de la RAM, entre quelques octets et de nombreuses centaines. Et la performance, la rapidité d'exécution, s'en ressent. Elle baisse principalement avec des fonctions simples comme de rapides basculements de certaines broches. Raison qui a présidé à l'exécution de fonctions spéciales dont le déroulement est dominé par le temps ; on évite ainsi de devoir commuter la broche en haut puis en bas à coup d'appels répétitifs de fonctions. Mais même si l'on voulait le faire pour obtenir p.ex. une gradation sur plusieurs LED, la performance resterait acceptable.

du superviseur du bus, les nœuds transmettent au PC les valeurs des deux capteurs. La 2e LED de la carte d'extension clignote à ce rythme ; le PC peut éteindre la 4e LED.
Le fichier ExtensionEFL.c contient aussi le code d'initialisation pour une carte à relais, cette partie de code a été compilée, on peut la sélectionner avec une directive #define dans ExtensionEFL.h.
Le téléchargement pour ce projet contient aussi, comme indiqué, un document plus détaillé sur EFL et la documentation générée avec Doxygen. Il y a là la totalité de la base de code, y compris d'autres bibliothèques, comme Display ou le module WizNet-TCP/IP, en code source.

Perspective

Cet article se veut uniquement une brève introduction. À l'intention des débutants en C et de ceux qui désirent en savoir plus sur EFL, il y a en prévision un article détaillé. En outre, EFL reviendra aussi dans de futurs projets d'Elektor. Il est délicat d'en fixer le calendrier de publication, mais une prochaine édition vous proposera une carte d'interface entre un moteur pas à pas et l'ElektorBus et auparavant il est prévu de vous présenter la carte Xmega-Webserver.
Peut-être y a-t-il certains de nos lecteurs qui, en plus d'utiliser EFL, souhaitent aussi y apporter leurs idées et leur contribution personnelle. Ils trouveront des renseignements dans le document additionnel sur [7]. Si vous avez des suggestions pour des changements ou des extensions, tels que l'API du contrôleur, n'hésitez pas à me contacter : redaktion@elektor.de. Bien sûr, il nous serait agréable aussi d'avoir de l'aide pour l'adaptation à d'autres contrôleurs, cartes et blocs périphériques.
Le site d'Elektor.Labs [11] publie régulièrement des mises à jour.

(120668 – version française : Robert Grignard)

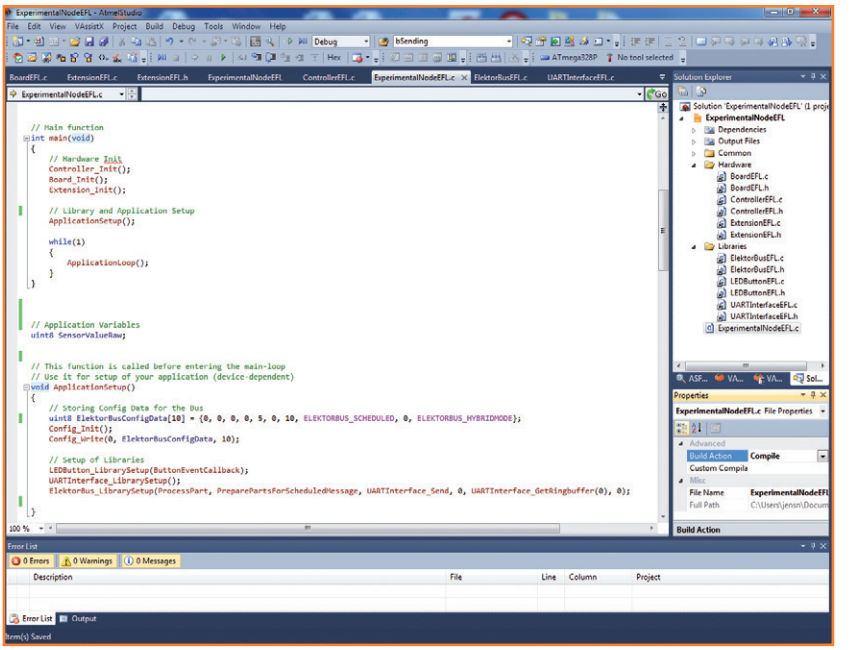


Figure 8.
Copie d'écran d'un exemple de projet en Studio 6 d'Atmel. Les fichiers de code du projet sont classés (virtuellement) dans des sous-dossiers définis, ce qui simplifie la supervision.

Liens

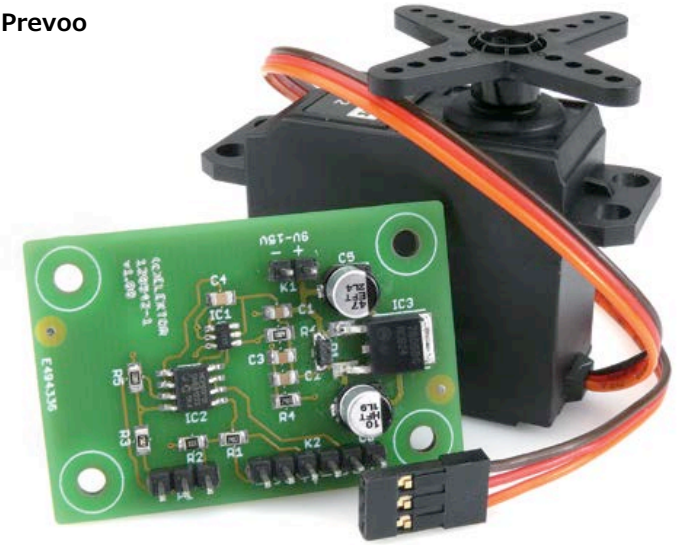
[1] www.elektor-magazine.fr/120582
[2] http://sourceforge.net/projects/embeddedlib/
[3] www.gnu.org/licenses/lgpl.html
[4] http://subversion.apache.org
[5] http://tortoisesvn.net
[6] www.doxygen.org
[7] www.elektor-magazine.fr/120668
[8] www.elektor-magazine.fr/120596
[9] www.elektor-labs.com/xmegawebserver
[10] www.elektor-magazine.fr/110258
[11] www.elektor-labs.com/EFL

Par RS4885, TCP/IP et radio : l'ElektorBus et son protocole

L'indépendance par rapport au matériel de la bibliothèque pour l'ElektorBus s'est encore améliorée. Le module de code ElektorBusEFL.h/ .c n'est désormais plus responsable que du protocole de l'ElektorBus. Il est alors possible d'émettre et de recevoir des messages non plus uniquement sur RS485, mais aussi par RS232, TTL/UART ainsi que par TCP/IP, on l'a déjà testé avec la carte Xmega-Webserver qui arrive et un module TCP/IP WIZ820io de WizNet. L'utilisation de modules radio à 433 MHz est aussi prometteuse.
De même, on peut écrire sa propre bibliothèque de protocole, elle aussi indépendante du canal de transmission, de la carte et du contrôleur. Il en résulte un immense terrain de jeu pour l'expérimentation et les applications.

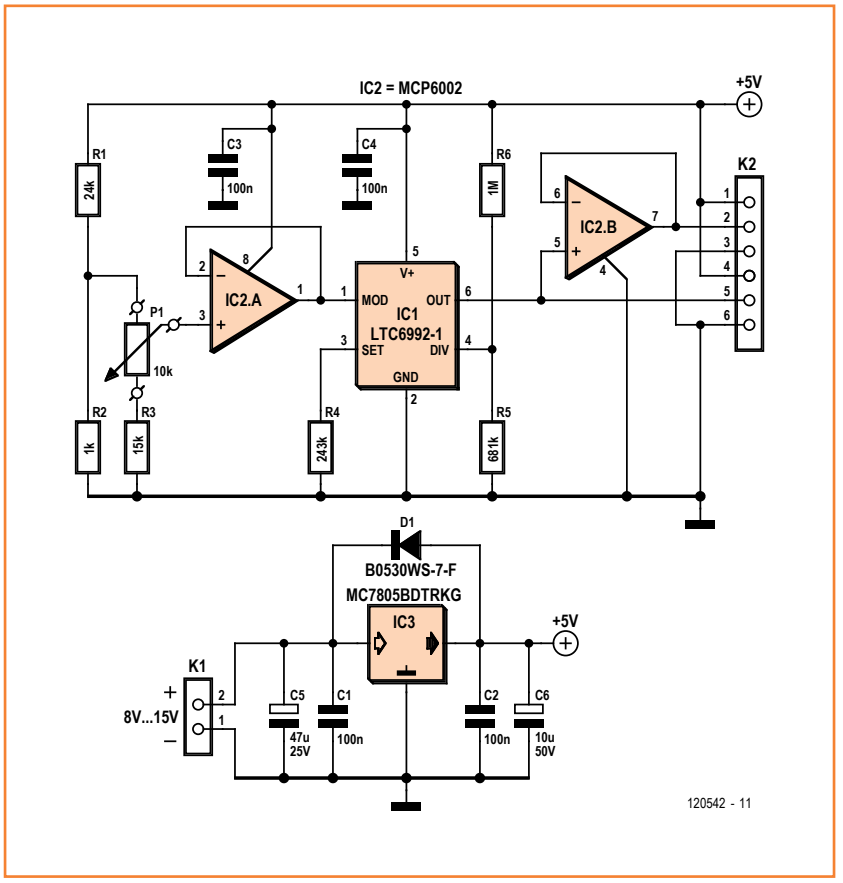
testeur de servos
exercice de soudage de CMS

Cederique Prevoo
(Pays-Bas)



Si vous utilisez souvent des servos, ce montage vous plaira pour des vérifications rapides : vous branchez le servo, vous tournez le potard dans un sens et dans l'autre et vous voyez s'il répond bien.
C'est tout, mais c'est déjà pas mal.

La commande de servos se fait avec la modulation de la largeur d'impulsion d'un signal. MLI en français ou *Pulse Width Modulation* en chouinegomme. La durée de la période est habituellement de 20 ms. On ne s'en fait pas pour l'exactitude, tant que la fréquence est stable. C'est la durée de (la période haute de) l'impulsion qui détermine la position adoptée par le servo. Les durées mini et maxi de l'impulsion sont de 1 et 2 ms (il y a des exceptions), et entre les deux, à 1,5 ms, le servo est à mi-course.



Ce circuit produit un tel signal MLI dont on fait varier la largeur avec le potard. Contrairement à la majorité des testeurs de ce genre, celui ne fait pas appel à un temporisateur 555, mais à un circuit intégré spécialisé de Linear Technology, le LTC6992, que son fabricant présente comme *Voltage-Controlled Pulse Width Modulator*. C'est donc une tension de commande analogique qui en fait varier la largeur d'impulsion, tandis que la fréquence est fixée par quelques résistances. Cette puce sort d'un nid que le fabricant LT appelle *TimerBlox*. Il fournit aussi un programme qui permet de dimensionner les composants qui déterminent la fréquence et la plage de modulation.

Ici la plage de réglage pour l'impulsion de commande est située entre 0,5 et 2,5 ms, de sorte que tous les types de servos pourront être testés d'un bout à l'autre de leur plage. La tension de commande requise par le LTC6992 varie de 0,12 à 0,2 V. Pour éviter les interférences, P1 et le réseau atténuateur R1/R2/R3 ne sont pas connectés directement à l'entrée MOD du LTC6992, mais à travers un tampon. C'est un

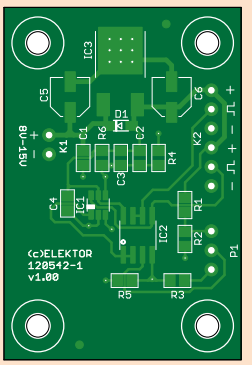
MCP6002 de Microchip qui a été retenu, un amplificateur opérationnel avec entrée et sortie rail à rail, de sorte que la tension relativement faible du potard est reproduite fidèlement. Le signal de sortie et les bornes d'alimentation pour le servomoteur à tester sont disponibles sur K2. Comme il reste un ampli op dans le MCP6002, nous en avons fait un tampon pour un deuxième signal de sortie sur K2. Hé oui, comme ça vous pouvez tester deux servos en même temps (comme le laisse supposer le titre) ou encore utiliser cette sortie pour examiner le signal à l'oscillo. Attention, l'ordre des connexions sur K2 ne correspond pas forcément à celui des connecteurs standard des servos que vous testerez. Il faut donc prévoir un convertisseur, facile à confectionner.

La régulation de tension est assurée par un 7805. La tension d'entrée sera comprise entre 7 et 15 V. Si elle est élevée, ne laissez pas le servo sous tension trop longtemps, car le régulateur 7805 risque de chauffer au point de se mettre en rideau, faute de radiateur efficace.
Comme annoncé au début de l'article, ce projet est un bon exercice de soudage de CMS. Amusez-vous !

(120542)

Liste des composants

Résistances (CMS 0805) :
R1 = 24 kΩ, 5%
R2 = 1 kΩ, 5%
R3 = 15 kΩ
R4 = 243 kΩ
R5 = 681 kΩ
R6 = 1 MΩ
P1 = pot. 10 kΩ lin.



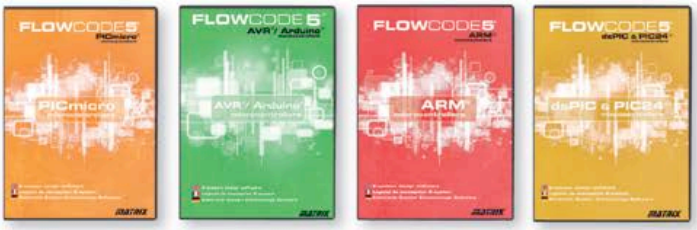
Condensateurs :
C1,C2,C3,C4 = 100 nF
C5 = 47 µF/25 V
(p. ex Panasonic EEEFT1E470AR)
C6 = 10 µF/50 V
(p. ex. Panasonic EEEFT1H100AR)

Semi-conducteurs :
D1 = diode Schottky 30 V/0,5 A, SOD323
(p. ex. Diodes Inc. B0530WS-7-F)
IC1 = LTC6992CS6-1, 6SOT-23 (Linear Technology)
IC2 = MCP6002-I/SN, SOIC8 (Microchip)
IC3 = MC7805BDTG, DPAK (On Semiconductor)

Divers :
K1 = embase à 2 points, pas de 2,54 mm
K2 = embase à 6 points, pas de 2,54 mm
P1 = embase à 2 points, pas de 2,54 mm
circuit imprimé 120542-1 (www.elektor.fr/120542)

Flowcode 5 pour concev

FLOWCODE5

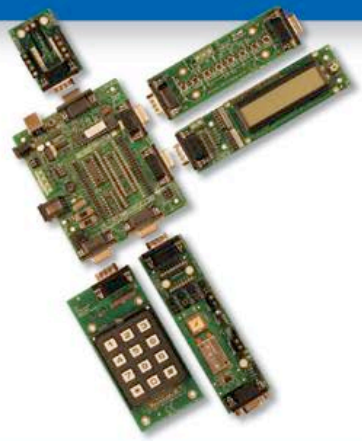


Flowcode 5 est l'un des langages de programmation graphique pour microcontrôleurs (PIC, AVR, ARM et dsPIC/PIC24) les plus avancés au monde. Son avantage principal est de permettre la création de systèmes électroniques et robotiques complexes même si l'on manque encore d'expérience.

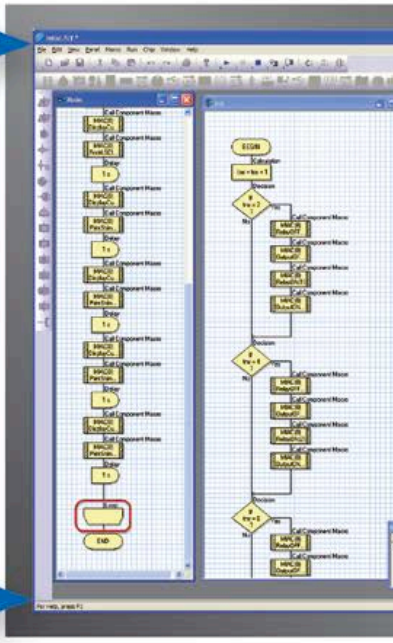
Flowcode est utilisé :

- dans l'enseignement, pour initier les étudiants à la programmation
- dans l'industrie, autant pour le prototypage rapide que pour des projets d'envergure

… en électronique



Les E-blocks sont des circuits électroniques compacts, correspondant chacun à une fonction autonome comme on les trouve dans les circuits embarqués. Il en existe une quarantaine, dont la complexité va croissant, depuis le simple afficheur à LED jusqu'au circuits de programmation, aux modules Bluetooth ou TCP/IP.
Les E-blocks peuvent être assemblés aisément pour élaborer des systèmes propices à l'apprentissage par l'expérimentation. Ils conviennent aussi pour le prototypage rapide de systèmes complexes. L'ensemble est complété efficacement par une gamme étendue et sans cesse renouvelée de logiciels puissants, et de capteurs variés.



… pour la commande industrielle

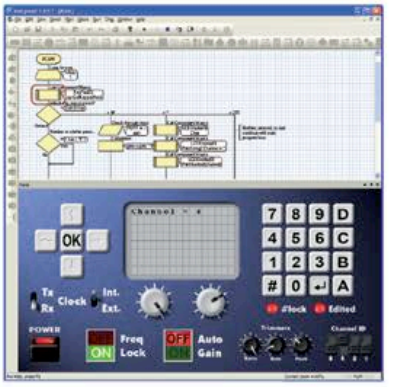
MIAC (**M**atrix **I**ndustrial **A**utomotive **C**ontroller) est une unité de commande industrielle pour circuits électroniques variés avec pour champs d'application privilégiés la capture, la mesure, la surveillance et l'automatisation.
Le MIAC lui-même est construit autour d'un puissant microcontrôleur PIC de la série 18 qui se connecte directement au port USB et se programme en Flowcode, en C ou en assembleur. Flowcode est fourni avec le MIAC, lequel est équipé d'origine du bus CAN, qui facilite la connexion en réseau de plusieurs MIAC.

Outil de débogage FlowKit

L'outil FlowKit offre la fonction ICD (In Circuit Debug) pour une gamme étendue d'applications Flowcode dans des projets PIC et AVR :

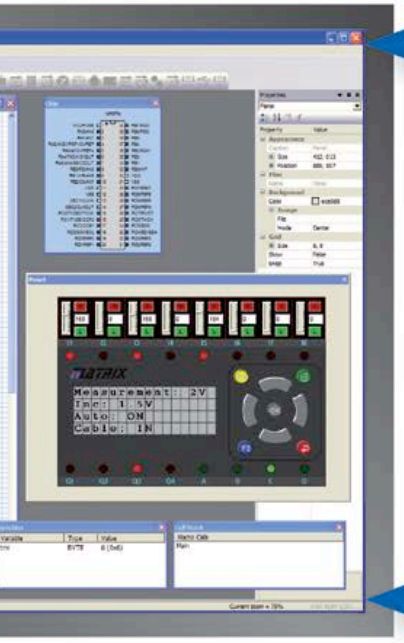
- marche, arrêt, pause et pas-à-pas pour programmes en Flowcode en temps réel
- suivi des variables de votre programme
- modification des variables
- débogage en circuit du robot mobile Formula Flowcode Buggy, ECIO et de projets autour de MIAC

oir et pour se former…



NOUVEAU dans Flowcode 5 :

- Nouvelle présentation personnalisable du code C
- Simulation améliorée
- Fonction de rechercher et de remplacement
- Nouveaux types et nouvelles fonctions des variables, des constantes et des variables de port
- Documentation automatique du projet
- Codage facilité par le nouvel explorateur de projet
- Mise en place de signets de code pour la navigation dans le programme
- La refonte complète du système d'interruption offre aux développeurs l'accès direct à plus de fonctions intégrées
- Amélioration de la signalisation des erreurs de compilation
- Désactivation de fonctions des icônes
- Amélioration des annotations
- Amélioration des liens vers les supports média



… en robotique

Formula Flowcode Buggy est le nom d'un petit robot mobile remarquable par son rapport performances/prix. Ce véhicule fournit aussi bien un support adéquat pour l'apprentissage de la robotique, qu'une plate-forme idéale pour des compétitions de robotique. Loin d'être un jouet, ce robot programmable par l'USB, est doté d'une détection de ligne, de capteurs de proximité, de 8 LED incorporées, d'un capteur sonore, d'un haut-parleur et du connecteur d'extension E-blocks. Ce véhicule se prête à de nombreux exercices de robotique depuis la simple détection de ligne jusqu'à l'analyse de labyrinthe. Le connecteur d'extension E-blocks autorise l'adjonction d'afficheurs, ou de modules Bluetooth, ZigBee ou d'un GPS.

… pour les projets USB

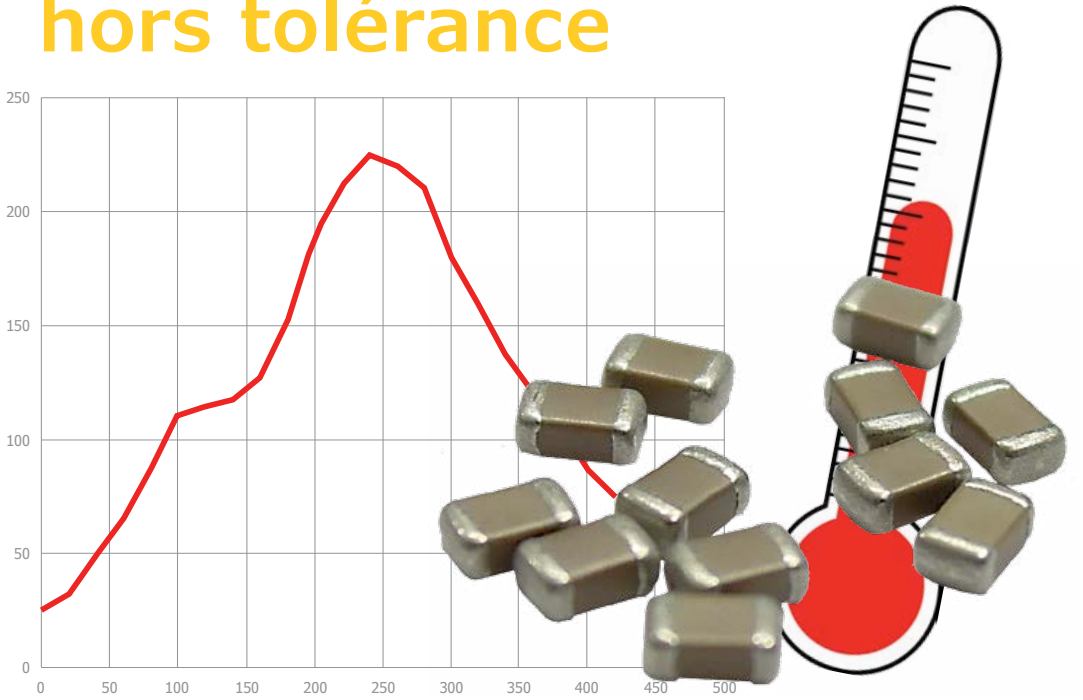
Les ECIO sont de puissants modules à microcontrôleurs programmables par l'interface USB, avec une empreinte au choix de 28 ou 40 broches au standard DIL (0,6 pouce). Construits sur des microcontrôleurs des séries PIC18 et des ARM7, les modules ECIO conviennent bien à l'étudiant et à l'autodidacte. Ils sont programmables en Flowcode, en C ou en assembleur. De nouvelles routines USB sous Flowcode favorisent le développement rapide de projets autour de l'USB, avec y compris les fonctions USB HID, USB esclave, et USB bus sériel (PIC seulement). Vous pouvez incorporer ECIO à vos propres circuits de façon à les doter de la fonction de reprogrammation.

Retrouvez les E-blocks et leur documentation sur :

www.elektor.fr/eblocks

condensateur céramique hors tolérance

Thijs Beckers
(Elektor)



En assemblant un prototype du LCR-mètre 0,05 %, dont la publication s'achève dans ce numéro, mon collègue Jan Visser s'est vu confronté à une propriété souvent négligée des condensateurs à isolant de céramique. C'est arrivé avec des condensateurs de 150 pF. La commande livrée et les composants déballés, Jan monte *aussitôt* les condensateurs en toute confiance sur le circuit imprimé avec sa station de soudage de CMS. Tout va bien… jusqu'à un certain point de la procédure semi-automatique d'étalonnage du LCR-mètre où Jan reste bloqué en raison de tolérances excessives des condensateurs. S'il les avait mesurés avant de les implanter, Jan aurait vu qu'avec 157 pF au lieu des 150 prévus, ils ne feraient pas l'affaire. Or avant l'implantation, il n'aurait pas mesuré cette valeur-là ! L'inspection des autres condensateurs du même lot révèle en effet que leur capacité réelle dépasse largement la valeur marquée : entre 168 et 183 pF, au lieu des 150 pF ±5 % ! Jan mesure alors quelques condensateurs *avant* et *après* passage par un cycle de chauffe dans notre four pour CMS. Avant, leur valeur oscillait entre 170 et 180 pF, après, elle retombe à peu près à la valeur correcte. Un condensateur de 178 pF p. ex. est ramené à 156 pF. Un exemplaire de 173 pF est passé à 152 pF, simplement après passage au four de soudage. Les électroniciens chevronnés connaissent l'existence des protocoles de chauffage recommandés par les fabricants pour que leurs composants prennent les caractéristiques spécifiées. Si vous êtes nouveau dans le métier ou si vous n'avez jamais eu ce genre de soucis ou si vous n'avez pas accès à une ligne de production professionnelle mais soudez les composants à la main comme l'a fait Jan, vous saurez maintenant qu'il n'est pas inutile de vérifier l'effet de la température sur des composants aux tolérances critiques, surtout des condensateurs à isolant de céramique. Désormais vous leur accorderez l'attention dont ils ont besoin.

Le diélectrique joue ici un rôle important, mais en fait nous ignorons lequel. Si vous pensez disposer d'informations de nature à nous aider à le comprendre, envoyez-les nous par courriel à editor@elektor.com avec comme sujet : ceramiz. En échange de votre contribution, si elle est substantielle, nous vous offrirons un kit mbed LPC11U24 offert par NXP et je publierai vos éclaircissements dans cette rubrique.

(130118 – version française : Robert Grignard)

Lien

[1] http://mbed.org/handbook/mbed-NXP-LPC11U24

flux et reflux de soudure

Thijs Beckers
(Elektor)

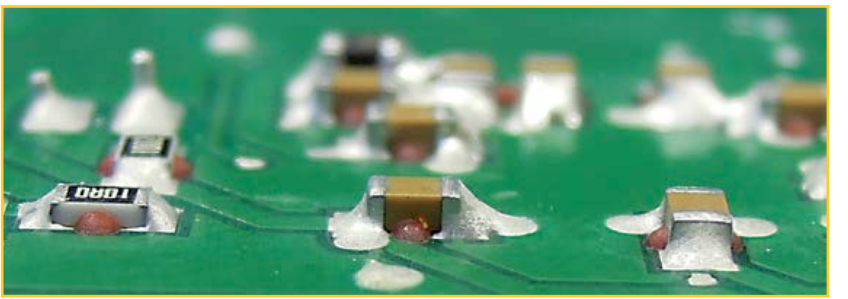
La vague est *la* technique par excellence pour souder rapidement et en masse des cartes électroniques. Elle consiste à faire surfer la platine sur une vague de soudure en fusion. Les pastilles et tous les composants de la face exposée sont ainsi brasés à grande vitesse. Si vous n'avez jamais vu ça, l'internet pullule de photos et de courtes vidéos qui montrent comment ça marche. Cette méthode, appliquée dans l'industrie depuis des années, a atteint les sommets de ses capacités. Et parfois elle les dépasse.

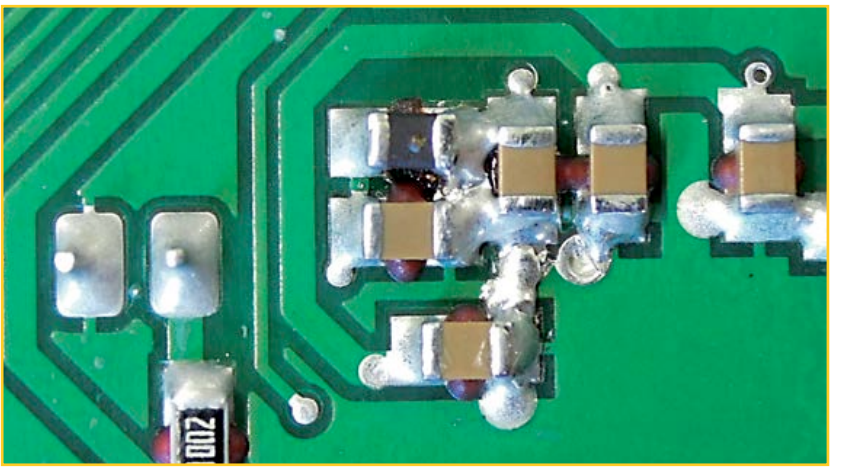
Regardez bien la première photo. Remarquez-vous que la soudure au bout du condensateur CMS est plus pentue à gauche que de l'autre côté ? Ce n'est pas uniquement à cause de la traversée (via) proche et donc de l'absence de masque à cet endroit.
C'est surtout que, lors du passage de la carte sur la vague de soudure, le sens contraire du mouvement de la platine et de celui de la soudure fait qu'il en reste moins à l'avant du composant qu'à l'arrière. Comme la queue d'une comète indique la direction du soleil, la « queue » de la soudure révèle la direction de la vague. Le même effet est visible sur la résistance CMS tout à gauche. On voit que l'onde de soudure est passée de la gauche à la droite de la carte.
C'est bon à savoir, parce que cela implique des restrictions dans l'implantation des composants lors de la conception du circuit imprimé. Observez bien la seconde photo de la même carte. La direction de la vague, de gauche à droite, est la même que sur la première ; vous en voyez déjà les conséquences ?
À gauche de la self noire d'en haut et du condensateur beige plus bas, les pastilles sont reliées entre elles par le tracé prévu. À droite, il y a un court-circuit manifeste entre les deux pastilles et les composants voisins, et le défaut est d'autant plus grave que, cette fois, l'alimentation est carrément mise à la masse.

Cela arrive en dépit de la présence, entre les pastilles, d'un masque à présent recouvert d'un grumeau de soudure. Ce masque n'a visiblement pas empêché le court-circuit. On a atteint là les limites du procédé de soudage à la vague.
Tout se serait bien passé si le condensateur n'avait pas été placé au voisinage direct du circuit de découplage LC. Celui-ci, en endiguant la vague, retient de la soudure entre les trois composants. La solution serait soit d'espacer l'implantation soit de choisir une autre méthode de fabrication. Un concepteur de circuits imprimés avisé doit donc tenir compte de la méthode de soudure envisagée. Prévoir ce genre d'écueil lui permet d'éviter de pénibles avaries.

Quand on fait de l'électronique, il y a tellement de détails dont il faut tenir compte ! Si vous vous demandez ce que sont les renflements rouges de part et d'autre des composants, il s'agit de colle pour les maintenir en place pendant le soudage. En fait, les photos montrent la face inférieure des cartes ; au cours du processus de soudage, les CMS sont donc tête en bas ; s'ils n'étaient pas collés au circuit imprimé, ils tomberaient et disparaîtraient avec la vague de soudure. Nous n'avons pas à nous préoccuper de ce collage, il est assuré automatiquement lors de l'implantation des composants en vue du soudage à la vague.

(130027 – version française : Robert Grignard)





LCR-mètre 0,05 %

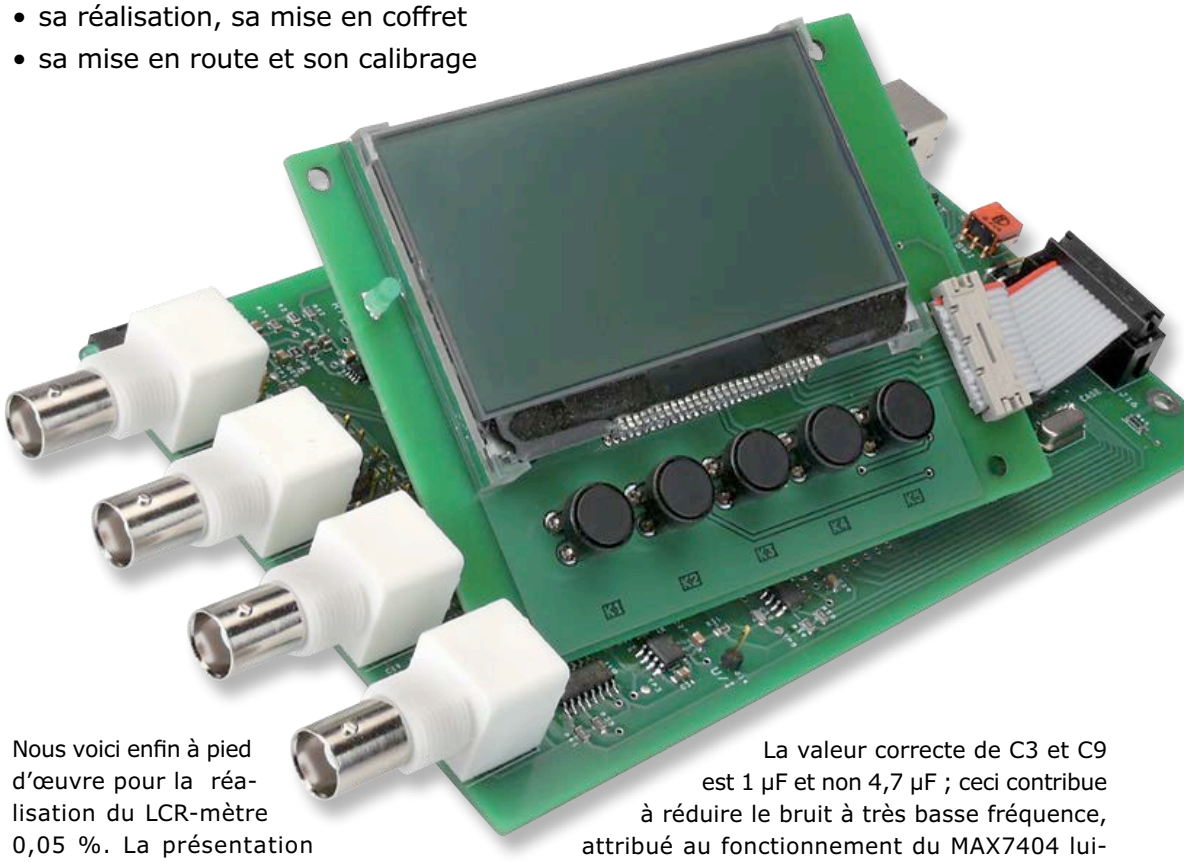
Le luxe de la précision accessible à tous

3e partie

Dans ce troisième article, vous découvrirez :

- quelques rectifications du schéma
- les circuits imprimés et les listes de composants de l'appareil,
- sa réalisation, sa mise en coffret
- sa mise en route et son calibrage

Jean-Jacques Aubry,
Ollioules



Nous voici enfin à pied d'œuvre pour la réalisation du LCR-mètre 0,05 %. La présentation des deux cartes occupe une grande place dans cet article, mais avant de passer enfin à la pratique, il nous faut revenir sur le schéma.

Divergences et précision

Nous avons en effet relevé quelques différences entre le schéma publié dans le numéro de mars 2013 et la liste des composants définitive de cet article. Comme souvent, quand il y a un doute, c'est la liste des composants qui fait autorité, mais comme celle-ci n'avait pas encore été publiée, personne ne pouvait s'en rendre compte ! Sur la **figure 3**, p. 50 du premier article [1], il manquait la valeur du quartz Y1 : 24 MHz.

La valeur correcte de C3 et C9 est 1 µF et non 4,7 µF ; ceci contribue à réduire le bruit à très basse fréquence, attribué au fonctionnement du MAX7404 lui-même et mis en évidence lors de la mesure de condensateurs de forte capacité (1000 µF et plus) : à l'oscilloscope, en visualisant le signal d'entrée des ADC, la valeur moyenne du signal fluctue d'autant plus à TBF que ces condensateurs sont de valeur élevée.

Pour U2, c'est un MAX7404**C**SA normal qu'il faut (ici, la plage de température étendue d'un MAX7404ESA est inutile). Pour U10, il faut un LM40**4**0 (et non le LM4050 que j'avais sous la main et qui fait aussi l'affaire). L'amplificateur utilisé pour U6 est un OPA3**65** et non un OPA354 dont la tension de décalage est trop forte. Et enfin pour U19, c'est un FT23**2**RL et non FT235RL

Caractéristiques

Le LCR-mètre 0,05 % est un pont de mesure automatique d'impédance. Précision de mesure maximale et facilité de construction ont été les deux fils conducteurs de sa conception. Il mesure la résistance, la capacité et l'inductance pour des composants dont l'impédance va de 1 mΩ à 1000 MΩ. Les mesures peuvent être faites à 3 fréquences : 100 ou 120 Hz (secteur 50 ou 60 Hz), 1 kHz et 10 kHz.

Deux configurations sont possibles :

- Appareil de base, sans afficheur ni clavier. Ne fonctionne *que* connecté par USB à un ordinateur sur lequel tourne le programme utilisateur. Ce programme, développé à partir des bibliothèques *Qt*, a été testé sous *Windows XP*, *Windows 7, Linux* (*Ubuntu* 11.04) et MacOSX (*Snow Leopard*, *Lion* et *Mountain Lion*).
- Appareil de base + extension *afficheur & clavier*. Fonctionne *aussi* de manière autonome (sans PC) avec alimentation externe de 5 V sur le câble USB (qui peut aussi être relié à un ordinateur pour l'alimentation).

(banale faute de frappe).
Enfin, nous avons constaté que pour faciliter un des réglages, il faut implanter une résistance de 10 Ω pour R17, indiquée à tort non câblée (NC) dans le schéma. Les autres composants marqués NC ne devront être implantés que si la procédure de calibrage l'exige, nous y reviendrons.
Avant de mettre à chauffer votre fer, nous vous invitons à reporter soigneusement ces petites corrections sur votre schéma du premier article. Une version XL corrigée de ce schéma sera mise en ligne et pourra être téléchargée [3] avec le logiciel et les autres documents, au nombre desquels figurent les circuits imprimés.

Assemblage

Le dessin des deux circuits imprimés qui forment le LCR-mètre est donné par les **figures 1** à **3.** Rappelons que **vous n'avez besoin de la carte d'extension que pour la version autonome de l'appareil**, autrement la carte principale suffit. Si vous êtes équipé et que vous jouissez d'une solide expérience dans le soudage manuel des CMS, vous pouvez souder les composants vous-même sur les cartes nues. La satisfaction de la réussite n'en sera que plus grande. Cependant, même s'il n'y a pas de difficulté particulière, c'est un travail de spécialiste qu'il ne faut entreprendre qu'en connaissance de cause.
Respectez scrupuleusement la liste des composants, pas pour l'ordre des composants bien sûr, mais pour leur valeur et leur tolérance. En dehors des 4 résistances de précision, toutes ont une tolérance standard de 1%. R50 est bien une résistance de 0 Ω qui permet de relier en un point bien précis, et un seul, les masses numériques et analogiques ; un pont de soudure fait aussi l'affaire ! La tolérance des différents condensateurs est précisée dans la liste des composants. Bien respecter les types de diélectriques spécifiés, particulièrement pour les condensateurs NPO.
À côté du quartz, vous remarquerez sur la carte un point marqué XTAL CASE. Il faut le relier par un petit morceau de fil au boîtier du quartz dont le blindage métallique n'opère que s'il est relié à la masse. Pour ne pas détériorer le quartz, ne vous attardez surtout pas avec le fer !
Attention : Si vous soudez vous-même les composants, le microcontrôleur sera vierge. Il faudra y charger le programme d'amorçage ; pour cela vous devrez utiliser l'USB DEBUG ADAPTER de *Silicon Laboratories* [3].

Si vous optez pour la version autonome, faites attention aussi à la référence complète de composants tels que le connecteur HE10 du câble en nappe soudé directement sur la carte d'affichage (**fig. 4**). La hauteur d'un modèle conventionnel empêcherait de monter le module dans le boîtier recommandé. L'afficheur graphique est soudé par une rangée de broches fines flanquée de deux broches de taille normale, mais cela ne suffit pas à l'immobiliser ; pour éviter qu'il ne se soulève accidentellement, nous l'avons fixé du côté opposé aux broches à l'aide de quelques gouttes de colle thermofusible.

Pour éviter tout malentendu sur le choix des composants, nous mettons à votre disposition sur notre site [3] la nomenclature (BOM *bill of material*) qui complète la liste des composants reproduite ici.

Pour l'ordre d'implantation, c'est la taille des composants et l'écart de leurs broches qui priment. N'oubliez pas les composants montés sous le circuit principal.
Les composants marqués NC ne seront éventuellement implantés que plus tard, pour le calibrage. Si, au contraire, vous n'êtes pas un adepte du soudage des CMS ou si vous n'avez pas le temps, il suffit de commander les modules prêts à l'emploi auprès d'*elektorPCBservice*, notre service de production professionnel : vous recevrez par la poste des modules qu'il suffira de mettre en coffret avant de les calibrer.
En effet, comme beaucoup d'appareils électroniques, un LCR-Mètre est sensible au rayonnement électromagnétique, notamment celui du secteur électrique à 50 ou 60 Hz. Un boîtier métallique s'impose, p. ex. *Hammond* 1455L1601, idéal pour une carte de 160 x 100 mm plus éventuellement l'extension. Des plans mécaniques cotés sont disponibles sur notre site [3].

Préparatifs

Le perçage pour les embases BNC et la LED D6 sur une des petites faces ne pose aucun problème. Pour le connecteur USB et l'interrupteur SW1 sur la face opposée, il faut un outil à section carrée ou… une bonne lime trempée dans l'huile de coude ; en fait, pour faire du travail soigné, il en faudra au moins deux, une lime plate et une queue de rat et peut-être un tiers-point.

Pour la mise à la masse de la carte principale et du boîtier, il faudra aussi percer un trou (Ø 3,2 mm) dans le fond du coffret en aluminium, à l'aplomb du trou du circuit imprimé situé à côté de J16 ;

Liste des composants circuit principal

Résistances :

(CMS 0805 1 %, sauf mention contraire)
R1, R16, R17, R28 = 10Ω
R2, R34 = 820 kΩ
R3, R5, R11, R13 = 8,2 kΩ
R4, R10, R47, R55, R56, R71, R72, R78, R81, R82, R94, R95 = 10 kΩ
R6, R58, R59 = 1,8 kΩ
R7, R100 = 5 kΩ aj. (Vishay-Sfernice TS53YJ502MR10)
R8, R60, R62 = 16 kΩ
R9, R23, R25, R35, R38, R39, R41, R52, R68, R69, R70, R87, R88, R96 = 56 Ω
R12, R14 = 5,6 kΩ
R15, R97 = NC (non câblé)
R18 = 10kΩ 0,05 % 10 ppm (Panasonic ERA6ARW103V)
R19 = 1 Ω
R20 = 1 kΩ 0,05 % 10 ppm (Panasonic ERA6ARW102V)
R21 = 100 Ω 0,05% 5 ppm (Vishay-Dale TNPU0805100RAZEN00)
R22 = 100 kΩ 0,05 % 10 ppm (Panasonic ERA6ARW104V)
R24, R26, R27, R29, R33, R36, R37, R40, R44, R57, R64 = 100 Ω
R30, R61, R76, R77, R80, R101 = 20 kΩ
R31 = 750 Ω
R32, R42, R49, R66, R93 = 100 kΩ
R43, R84, R85, R86, R89 = 2,2 Ω
R45, R73 = 39 kΩ
R46, R90, R91 = 680 Ω
R48, R51, R74 = 4,7 kΩ
R50 = 0 Ω
R53, R65 = 470 Ω
R54 = 1 kΩ
R63, R98 = 1,6k Ω
R67 = 62 Ω
R75 = 5 kΩ aj. (Bourns 3266W-1-502LF)
R79 = 4,3 kΩ
R81 = 7,5 kΩ
R83 = 30 kΩ
R92 = 430 Ω
R99 = 2 kΩ

Condensateurs :

(CMS, taille 0805 sauf mention contraire)
C1,C2,C4,C10,C11,C12,C20,C26,C27, C28,C29,C31,C33,C34,C35,C36, C37,C40,C41,C43,C44,C45,C46, C47,C48,C49,C50,C51,C53,C54, C61,C63,C65,C78,C80,C86,C87,C88 = 100 nF 10 % X7R 25 V
C3, C9, C62, C64, C90 = 1 µF 10 % 25 V X7R
C5, C13, C18 = taille 1206 15 nF 5 % 50 V NPO
C6, C7, C14, C15, C16, C56 = taille 1206 47 nF 5 % 50 V NPO
C8, C73, C89 = 1 nF 5 % 50 V NPO
C17, C21, C23, C55, C60 = NC (non câblé)
C19, C25 = 150 pF 5 % 50 V NPO
C22, C52, C58, C59 = 1,5 nF 5 % 50 V NPO
C24, C32, C69, C72, C74, C75, C81, C85 = 2,2 µF 10 % 16 V X7R
C30, C57 = 4,7 nF 5 % 50 V NPO
C38, C39, C83, C84 = 33 pF 5 % NPO
C42, C70 = tantale boîtier A 33 µF 6,3 V (Vishay Sprague 293D336X96R3A2TE3)
C66, C71, C82 = 10 nF 10 % 50 V X7R
C67, C77, C79 = 4,7 µF 10 % 10 V X5R
C68, C76 = tantale boîtier D 470 µF 6,3 V (Kemet T495D477K006ATE100)

Inductances :

L1 = inductance double CMS, 20 µH (Bourns PM3602-20-RC)
L2, L3 = perle de ferrite CMS 0805 (Murata BLM21PG221SN1D)

Semi-conducteurs :

(CMS sauf mention contraire)
D1, D2, D3, D4, D5 = BAV199 (SOT23)
D6 = LED verte 3 mm, à fils, montage horizontal (Dialight 551-0207F)
D7 = MBR0520 (SOD123)
D8, D9 = LED rouge 0805 (ex. Kinbright KP-2012SURC)
D10 = BAT54A (SOT23)
U1 = OPA725AIDBV (SOT23-5) (TI)
U2 = MAX7404CSA (SOIC-8) (Maxim)
U3 = 74HCT4052D (SOIC-16)
U4, U20 = TLC2274AID (SOIC-14 (TI)
U5 = INA128U (SOIC-8) (TI)
U6, U11, U14 = OPA365DBV (SOT23-5) (TI)
U7 = PGA103U (SOIC-8) (TI)
U8 = 74HCT4053D (SOIC-16)
U9 = C8051F061-GQ (TQFP-64) (Silicon Laboratories)

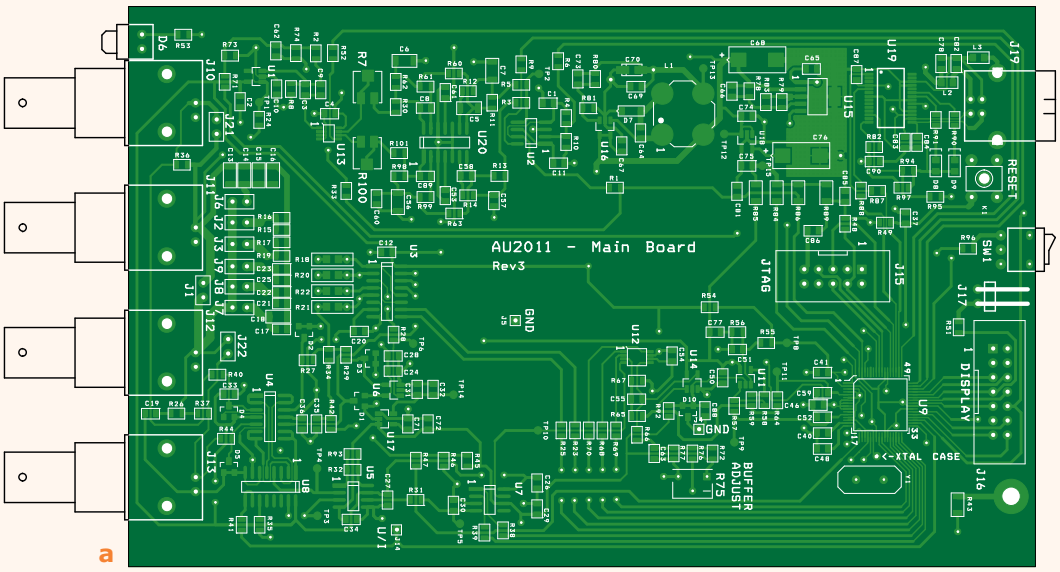
utiliser un ensemble vis M3 (12 mm) + écrou et rondelle + entretoise de 5,5 mm.
Pour loger l'extension afficheur & clavier, le couvercle en aluminium extrudé devra être usiné et muni d'un marquage (sérigraphie ou gravure) pour les fonctions des boutons (**fig. 5**). Ensuite, utiliser 4 vis M3 (tête fraisée, 12 mm) + écrous et rondelles, et 4 entretoises de 7 mm.

Note : pour la fixation de l'extension, à défaut d'entretoises, il est possible d'utiliser d'autres écrous (et rondelles) :
- 4 pour fixer les 4 vis sur le couvercle,
- 4 pour régler l'écartement à 7 mm,
- enfin 4 autres pour fixer l'extension.

Deux remarques pour finir sur les cavaliers : Normalement la mise à jour du *firmware* est faite par le programme principal sur l'ordinateur. Aussitôt après, le circuit doit fonctionner, mais si pour une raison quelconque le *firmware* ne répondait pas (par ex. en cas de bogue d'un nouveau *firmware* qu'on vient de charger), il faut alors agir au niveau du *bootloader* et lui dire que l'on veut faire une mise à jour *inconditionnelle*. C'est le cavalier en J17, accessible en démontant le panneau arrière, qui donne cette information.
La procédure de mise à jour est décrite dans la documentation téléchargeable [3].

Pour assurer une bonne prise de la sonde de mesure sur les broches des points de mesure J4, J5, J14, il n'est peut-être pas inutile de les couder.

Tous les autres cavaliers sont décrits dans la documentation en ligne.



a

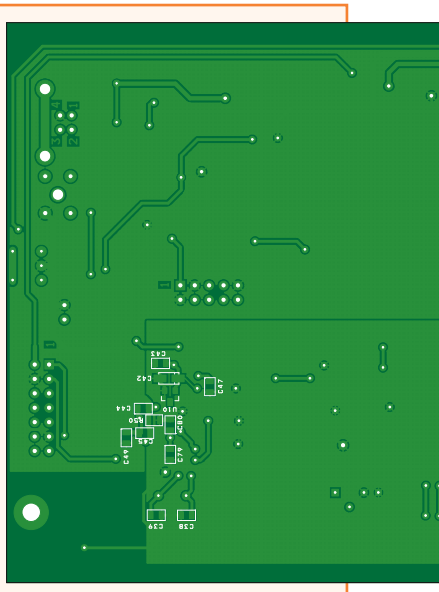

b

Figure 1. Le circuit imprimé double face principal du LCR-mètre, côté composants (a). Ne pas oublier la petite douzaine de composants à monter sous la carte (b). L'absence de sérigraphie sous le circuit imprimé réel est délibérée ; ce n'est ni un oubli ni une anomalie.

U10 = LM4040D25IDBZ (SOT23) (TI)
U12 = DAC8811CDGK (MSOP-8) (TI)
U13 = SN74LVC2G53DCT (SM-8) (TI)
U15 = REG102GA-A (SOT223-5) (TI)
U16 = LT1611CS5 (SOT23-5) (Linear Technologies)
U17 = TPS72325DBV (SOT23-5) (TI)
U18 = TLV70030DDC (SOT23-5) (TI)
U19 = FT232RL (SSOP-28) (FTDI)

Divers :
Y1 = quartz 24 MHz fondamentale (ex. Euroquartz 24.000 MHz HC49/4H/30/50/40+85/18pF/ATF)
SW1 = inverseur (RS Components 4US1R1020M6RNS, code 734-6934)
J1, J2, J3, J6, J7, J8, J17, J21, J22 = barrette 2 pts mâle, pas de 2,54 mm
J4, J5, J14 = barrette 1 pt mâle, à recourber (cf texte)
J10, J11, J12, J13 = embase BNC horizontale isolée (ex. TE Connectivity 1-1337543-0)
J15 = HE10-10 contacts (ex. Multicomp MC9A12-1034)
J16 = HE10-14 contacts (ex. Multicomp MC9A12-1434)
J17 = barrette coudée 2 pts mâle, pas de 2,54 mm
J19 = embase USB type B horizontale (ex. TE Connectivity 292304-1)
K1 = bouton-poussoir (Omron série B3F-10xx)
circuit imprimé nu 110758-1
ou module assemblé 110758-91
boîtier (Hammond 1455L1601 + usinage + marquage)
visserie

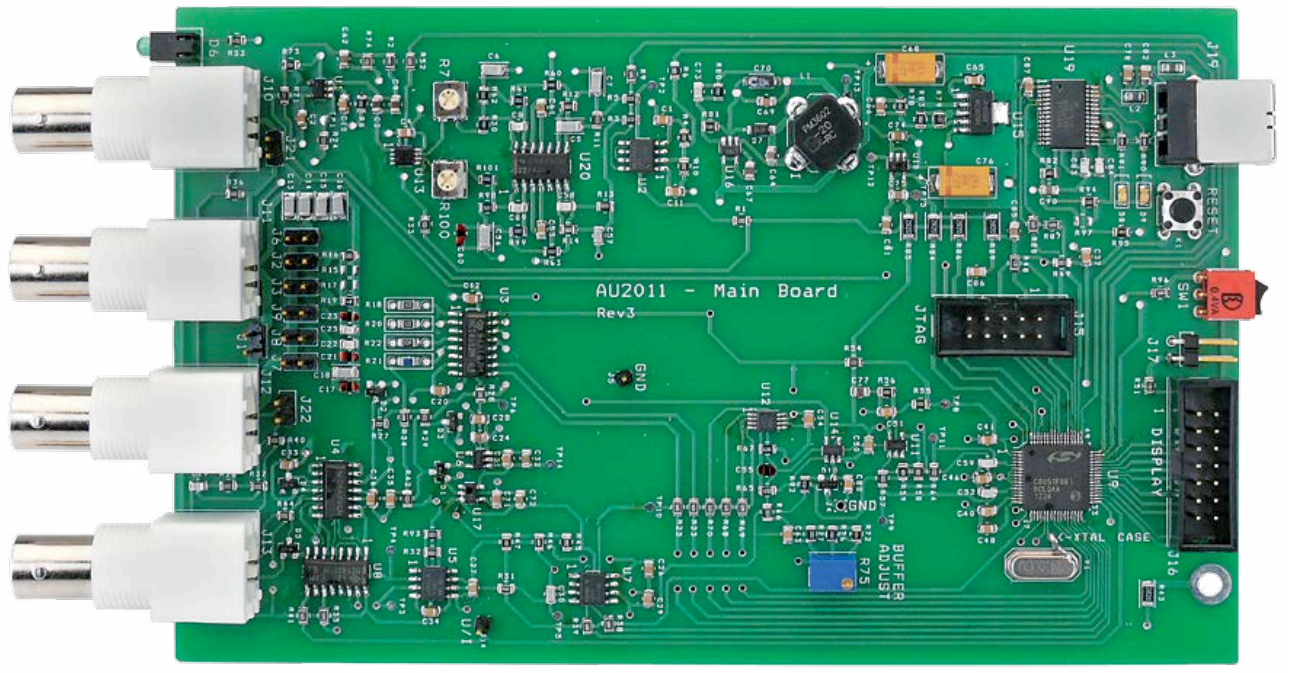


Figure 2.
Photo du prototype.

Réglage semi-automatique

Contrairement à ce que pourrait laisser craindre la précision remarquable de cet appareil, sa procédure de réglage ne requiert ni compétences ni appareillage particulier. Hormis un multimètre pour vérifier la tension d'alimentation, aucun appareil de mesure n'est requis ! C'est le logiciel qui prend en charge le calibrage semi-automatique. L'utilisateur n'a à effectuer lui-même que quelques réglages, pour lesquels il est guidé pas

Liste des composants afficheur-clavier

Résistances :
(CMS 0805 1%)
R1, R2, R3, R4, R10, R11 = 4,7 kΩ
R5, R6, R7, R9 = 56Ω
R8 = 1kΩ

Condensateurs :
(CMS, taille 0805)
C1, C2, C3, C4, C5, C6, C7, C8, C9, C10 = 1uF 10 % 25V X7R

Semi-conducteurs :
D1, D2, D3, D4 = BAT54A (SOT23)
D5 = LED verte 3 mm à fils (ex. Kingbright L-424GDT)
Q1, Q2 = MOSFET N (SOT23) (ex. Fairchild FDV303N)

Divers :
U1 = afficheur graphique Displaytech 64128M-FC-BW-3
J1 = transition HE10 14 voies **épaisseur 5,4 mm** (Harting 09 18 114 9622)
K1, K2, K3, K4, K5 = bouton-poussoir (Multicomp TS0B22)
connecteur nappe 14 voies (ex. Multicomp MC6FD014-30P1)
nappe 14 voies 1m (10 cm utilisés) (3M 3365-14)
circuit imprimé nu 110758-2
ou module assemblé 110758-92

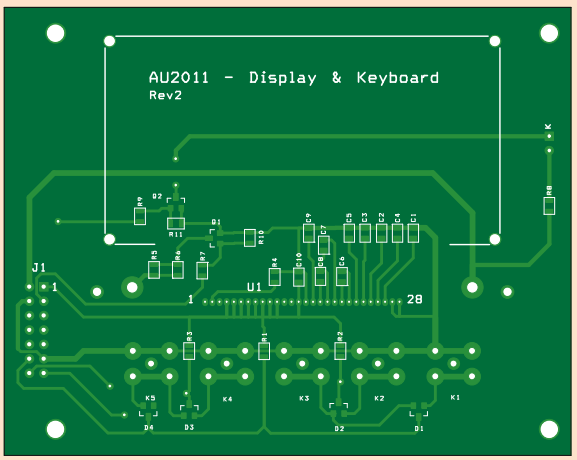


Figure 3. Le dessin du circuit imprimé double face de l'extension d'affichage (optionnelle).

à pas par le programme qui effectue toutes les mesures. La procédure est soigneusement décrite dans un document téléchargeable [3], y compris l'implantation éventuelle des composants NC et la configuration des cavaliers. Il suffit d'un petit tournevis, d'un peu de jugeote et de patience, car il faut lire la documentation d'un bout à l'autre et en suivre scrupuleusement toutes les indications, sans brûler les étapes.

Nous avons vu dans le premier article l'importance des cordons de mesure et leur incidence sur la précision. Selon l'usage que l'on fera du LCR-mètre, on utilisera donc des pinces Kelvin (**fig. 7a**) et/ou un boîtier de mesure à 4 connecteurs BNC comme la référence TH26001A, dite *4 terminal test fixture* de TONGHUI (**fig. 7b**). On les trouve facilement en effectuant une recherche sur l'internet (mots clé *LCR test clip*, *Kelvin clip*, *TH26001A*).

Une fois que vous aurez votre carte assemblée par vos soins, vous pouvez l'alimenter avec une alimentation USB et vérifier les tensions d'alimentation. Le µC étant vierge, tous ses ports sont en haute impédance et il ne fait rien ! Donc aucun signal à mesurer.
Vous pouvez raccorder la carte brièvement à un PC et devriez voir les diodes D8 et D9 s'allumer : c'est le FT232R (U19) qui dialogue avec le PC. Une fois le programme d'amorçage (*bootloader*) chargé dans le microcontrôleur par le programme *FlashUtil* au moyen du module USB

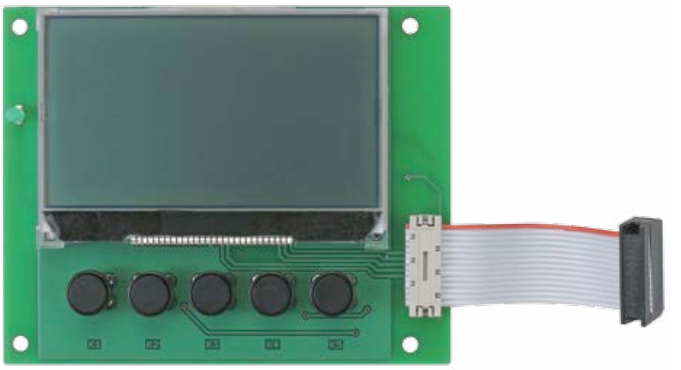


Figure 4a. Attention au modèle du connecteur HE10 de l'afficheur.

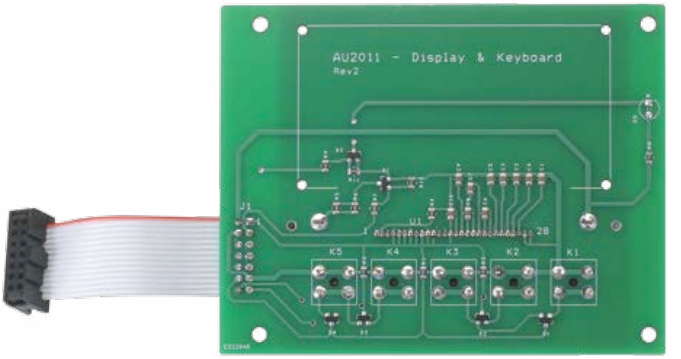
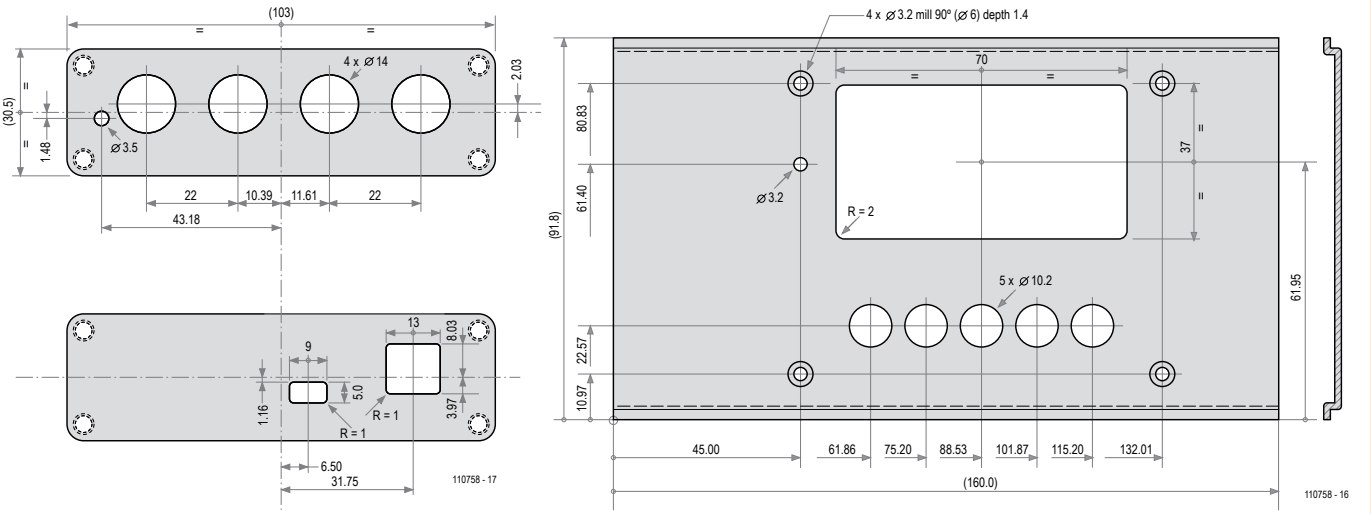


Figure 4b. L'extension vue de dessous.

Figure 5. Plans mécaniques cotés.



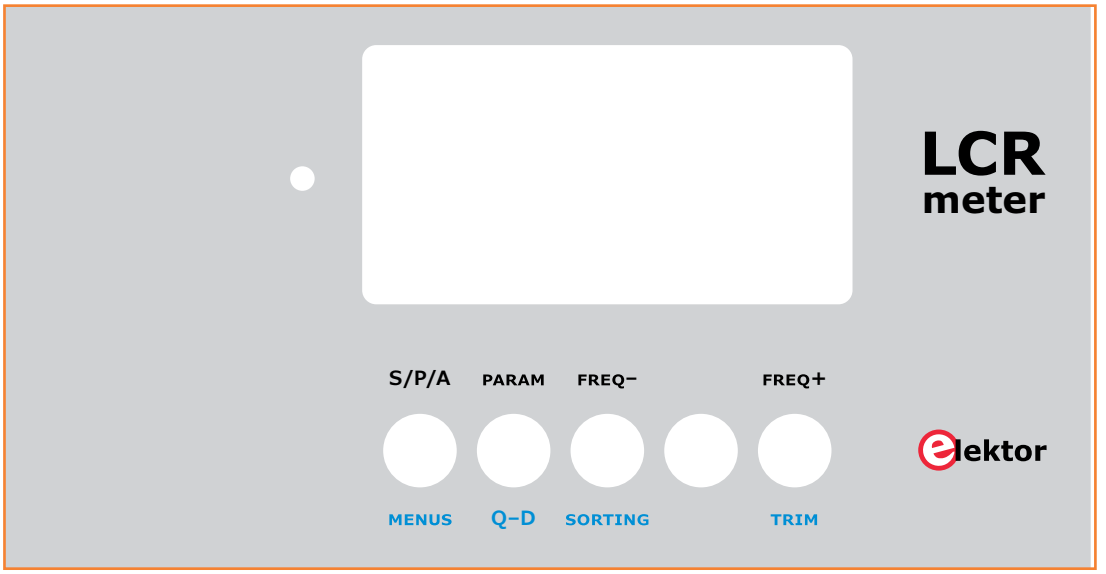


Figure 6.
Proposition de face avant, à graver ou à sérigraphier.

DEBUG ADAPTER (**fig. 8**) connecté à J15, le µC est fonctionnel. Cette opération et toutes celles qui suivent sont décrites en détail dans la documentation en ligne.
C'est le programme AU2011 installé sur le PC qui doit ensuite établir la liaison logicielle en ouvrant le port ; ça marche ? Alors, vous êtes sur la bonne voie ! Il reste à charger le *firmware* depuis le PC par le menu *Outils/Mise à jour du programme* du programme AU2011.
Après ça, au redémarrage du LCR-mètre, vous devriez voir les réglages (semi-) automatiques des *offsets* s'effectuer normalement comme le décrit le manuel de *Mise en route* [3]. Si oui, il y a

Figure 7a.
On trouve ce jeu de pinces Kelvin avec cordons à 4 BNC pour une dizaine d'euros.

Figure 7b.
Cette borne de mesure montée directement sur le boîtier du LCR-mètre se trouve à moins de 50 €.

95 % de probabilité que tout soit correct. Ensuite vous lancerez les mesures en cliquant sur le menu *démarrer* sur le PC. Si les paramètres d'un circuit ouvert avant TRIM s'affichent (**fig. 9**), alors la probabilité du succès est proche de 100% !

(130093)

Liens & références

[1] LCR-mètre 0,05 % 1re partie
www.elektor-magazine.fr/110758

[2] LCR-mètre 0,05 % 2e partie
www.elektor-magazine.fr/130022

[3] LCR-mètre 0,05 % 3e partie
www.elektor-magazine.fr/130093

Documentation en ligne

- logiciel (bootloader, firmware et programme principal)
- dessins des circuits imprimés 1 & 2
- schéma d'implantation dessus 1 & 2
- schéma d'implantation dessous 1
- BOM complète
- version XL du schéma
- plan mécanique (versions autonome et avec extension)
- façade (version autonome seulement)
- documents Mise en route & Mode d'emploi

Figure 8.
Pour charger le programme dans un microcontrôleur vierge, il faut cet accessoire.

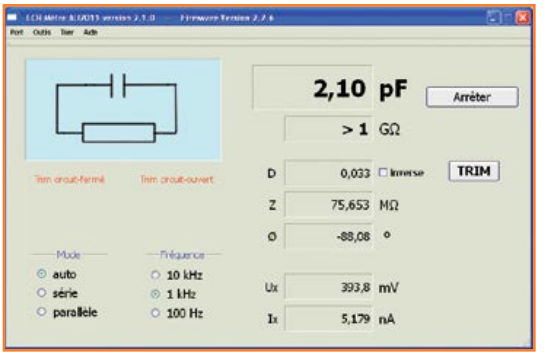


Figure 9.
L'apparition de cette fenêtre avec les paramètres d'un circuit ouvert avant TRIM confirme le bon fonctionnement du circuit.

guidage par menu

programmer le menu d'un afficheur

Benedikt Sauter [1]

Même sur un afficheur à petit écran, un guidage par menu bien programmé est synonyme d'accès rapide à de nombreux paramètres et réglages. La navigation dans un menu est d'autant plus aisée que les touches sont situées près de l'afficheur. C'est le cas de notre carte d'extension Linux, ne la privons donc pas du luxe d'un menu.

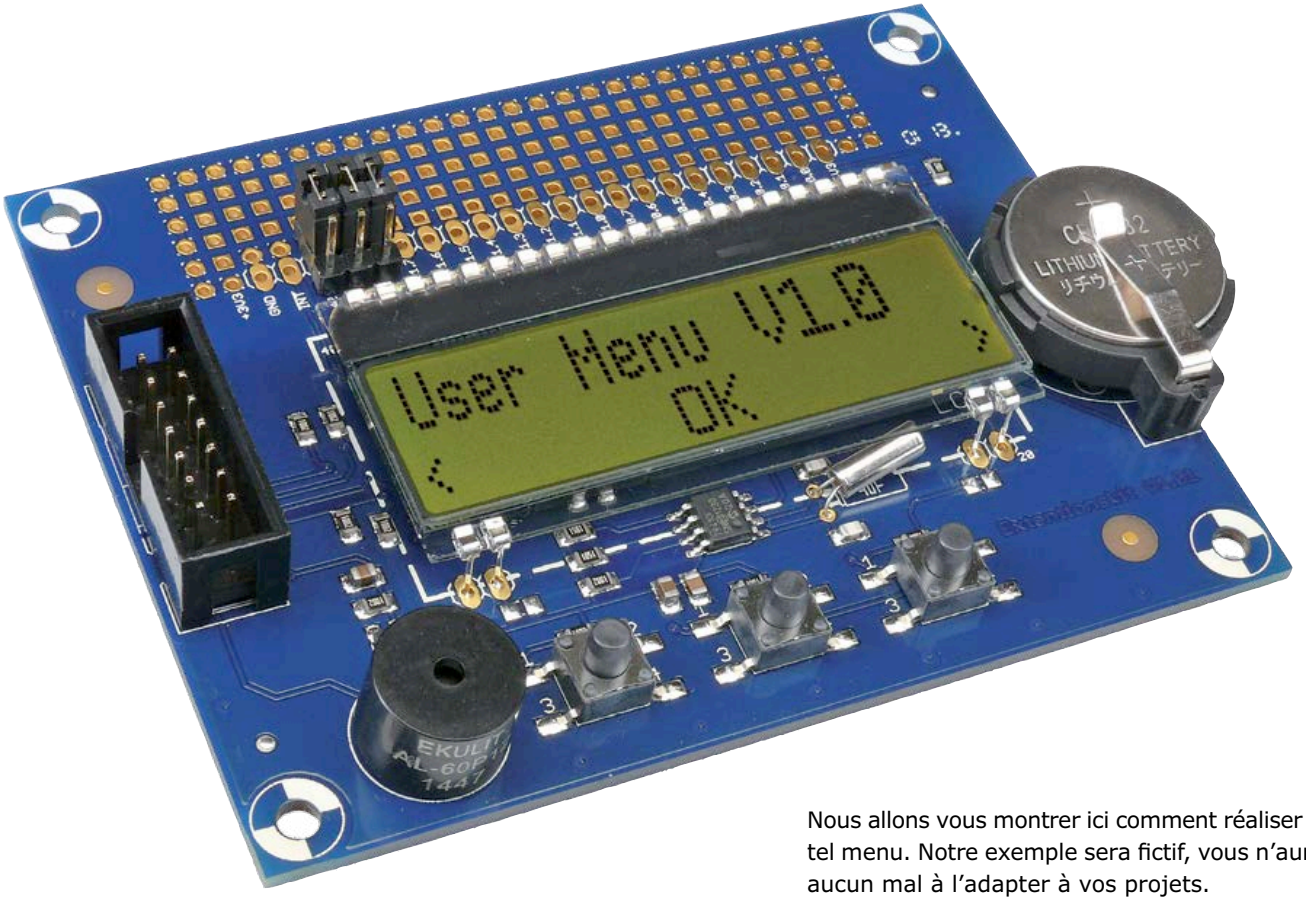


Figure 1.
La carte d'extension Linux et ses trois poussoirs judicieusement placés sous l'afficheur.

À l'heure où les fabricants de téléphones tactiles et de tablettes offrent des résolutions d'écran toujours plus hautes, de nombreux appareils, comme des imprimantes, des serveurs de stockage en réseau ou encore des télécopieurs, sont encore équipés d'un simple afficheur de texte. Sur ces appareils, quelques touches et un menu dépouillé suffisent à l'utilisateur pour effectuer un réglage ou consulter l'état de la machine.

Nous allons vous montrer ici comment réaliser un tel menu. Notre exemple sera fictif, vous n'aurez aucun mal à l'adapter à vos projets.

Structure du menu

La carte d'extension Linux [2] sera un support matériel idéal pour notre exemple, puisqu'elle est équipée de trois boutons-poussoirs placés directement sous son afficheur de 2 lignes et 16 caractères (**fig. 1**).

Commençons par dresser la liste des éléments de menu qui serviront au réglage ou à la visualisation de différents paramètres :

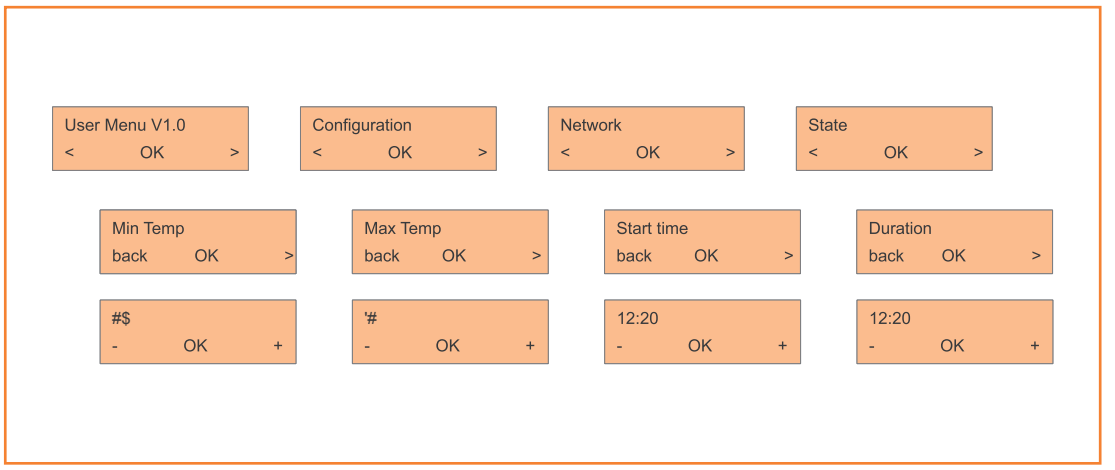


Figure 2.
Exemple d'une structure de menu.

- Configuration
  - Température minimum
  - Température maximum
  - Heure de début quotidienne
  - Durée maximale
- Configuration du réseau
  - Adresse IP : dynamique ou statique
  - Adresse IP
  - Adresse réseau
  - Passerelle
  - Serveur DNS
- État
  - Température actuelle
  - Taux de charge du processeur
  - Occupation de la mémoire
  - Durée d'exploitation de la machine

Les éléments du menu *Configuration* donnent accès à des réglages importants. Les valeurs de seuil minimum et maximum permettront par exemple de déclencher un processus de commutation quelconque. La configuration d'une liaison réseau nécessite de même la modification de quelques paramètres. On décidera ainsi d'une configuration dynamique ou statique de l'adresse IP. Dans le cas d'une configuration statique, il faudra en outre entrer les paramètres IP. Un programme bien pensé n'autorisera bien sûr que la saisie d'adresses IP au format correct.

Réalisation du menu

Pour le guidage par menu, on peut toujours chercher l'inspiration du côté des appareils qui nous entourent : les exemples, plus ou moins réussis, ne manquent pas. La **figure 2** montre le schéma du système de navigation que nous allons mettre en œuvre. Ce n'est qu'une ébauche, les éléments décrits ci-dessus ne sont pas tous présents, mais vous n'aurez aucune peine à imaginer le reste de ce guidage par menu.
Une fois fait le choix des éléments auxquels l'utilisateur aura accès, l'étape suivante consiste à créer la logique qui commande le menu, y compris ses sous-menus et ses éléments d'action. Comme éléments d'action nous pourrions choisir :

- un affichage statique
- une sélection entre différentes options avec confirmation d'une option
- le réglage de paramètres numériques (nombres)
- le réglage de paramètres alphanumériques
- un affichage dynamique (barre de progression, heure, etc.)

Affichage statique et sélection d'options (p. ex. configuration statique ou dynamique d'une adresse IP) sont simples à réaliser. Le réglage de paramètres et l'affichage actualisé d'une valeur demandent plus de réflexion.

Programmer à l'emporte-pièce serait la meilleure façon d'aboutir à un code labyrinthique. Notre code doit au contraire être suffisamment flexible et extensible pour que l'ajout ultérieur d'un menu puisse se faire de façon rapide et intuitive, sans avoir à plonger et replonger dans les méandres du code.

Solution possible

On peut procéder en deux étapes pour mettre en œuvre un menu. D'abord, recourir à une bibliothèque pour la création du menu lui-même. Le **listage 1** montre le pseudo-code correspondant.

Listage 1. Pseudo-Code pour créer un menu

```
configuration   = MainMenu("Configuration")
network    = MainMenu("Network")
state    = MainMenu("State")

//Menu Configuration
mintemp  = SubMenu(configuration,"Min Temp")
maxtemp  = SubMenu(configuration,"Max Temp")
start  = SubMenu(configuration,"Start Time")
duration   = SubMenu(configuration,"Duration")

//Menu Network
ipchoose   = SubMenu(network,"IP-Address: Dyn/Static")
ip  = SubMenu(network,"IP-Address")
netaddress = SubMenu(network,"Network")
gateway= SubMenu(network,"Gateway")
dns   = SubMenu(network, "DNS-Server")

//Menu State
temperature  = SubMenu(state, "Temperature")
cpu   = SubMenu(state,"CPU")
ram   = SubMenu(state,"RAM")
uptime = SubMenu(state,"Uptime")

...

RegisterMenu(mintemp,MinTemperature)
```

Listing 2. Pseudo-Code pour le réglage d'un paramètre

```
Function MinTemperature()
{
  mintemp = ConfigRead("mintemp")
  switch(button)
  {
    case "button_left":
      mintemp--
    case "button_ok":
      Back()
    case "button_right":
      mintemp++
  }
  ConfigWrite("mintemp",mintemp)
  Display("%s",mintemp)
}
```

Les fonctions MainMenu et SubMenu permettent de créer une structure arborescente qu'il est facile d'étendre : lorsqu'un nouveau nœud est créé, le programme retourne une référence vers ce nœud. La création de sous-nœuds se fait alors par référence vers cette valeur de retour. La fonction RegisterMenu permet ensuite de lier les nœuds à des fonctions qui seront appelées lorsque l'utilisateur sélectionnera les sous-menus correspondants.

Vient ensuite l'étape d'implantation de ces fonctions. Lorsque l'utilisateur sélectionne un sous-menu, par exemple le réglage de la température minimum, le programme doit afficher un menu par l'intermédiaire duquel pourront être ajustées des valeurs numériques. Le **listage 2** montre un exemple de pseudo-code.

La fonction du listage 2, de même que toutes celles servant à régler un paramètre, doit posséder un accès en lecture et en écriture à un fichier de configuration ou à une base de données (ConfigRead et ConfigWrite). L'état des touches est évalué dans une structure de contrôle *switch*. La fonction Display affiche la valeur actualisée de la température.

Autre possibilité

Une autre voie, plus élégante, serait de se servir du système de fichiers (Linux) pour créer le menu. Pour cela nous placerions la structure du menu dans une arborescence de répertoires, dans chacun desquels serait placé un mini-programme gérant les sous-menus. La sortie du programme serait alors simplement dirigée telle quelle vers l'afficheur.

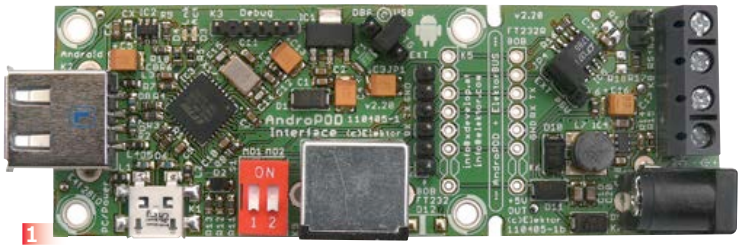
(130044 – version française : Hervé Moreau)

Liens

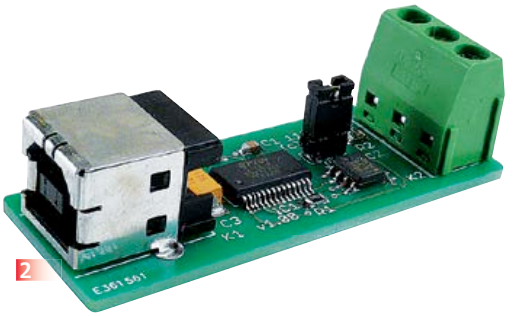
[1] sauter@embedded-projects.net

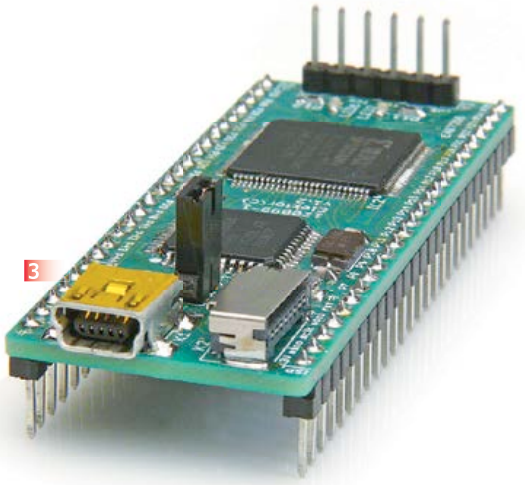
[2] www.elektor.fr/120596

[3] www.elektor.fr/130044



1


2


3

4

1 Andropod

Commandez vos montages avec un smartphone ou une tablette Android

Écran tactile à haute définition, méga puissance de calcul, connexion réseau et fonctions de téléphonie sans fil : les téléphones tactiles et les tablettes Android seraient des centrales de commande presque idéales pour nos projets d'électronique s'il était plus facile de s'y connecter. Voici AndroPOD, votre interface sérielle TTL et RS485. Décollage vertical garanti ! L'offre de matériel Android (par différents fabricants) et de logiciel est énorme et le code source du système d'exploitation est libre. Le puissant kit de fonctions logicielles ou framework d'Android offre accès à presque toutes les fonctions matérielles et permet de programmer des applications élégantes et conviviales.

Réf 110405-91

2 Convertisseur USB/RS485

Le convertisseur USB/RS485 établit la connexion entre un port USB de votre ordinateur (portable) et un bus RS485 à deux fils. Un bornier à vis assure une connexion facile et fiable du bus RS485. Grâce au populaire circuit intégré FT232L et ses pilotes de FTDI, le convertisseur fonctionne non seulement avec toutes les versions actuelles de Windows, mais aussi avec Windows CE, Windows Mobile, Linux et Mac OS X. Le module a été décrit dans une série d'articles d'Elektor qui traitent entre autres du protocole d'application, de mesure, de commande et de régulation, des platines expérimentales, etc.

Réf 110258-91

3 Et l'homme créa sa puce

Sans aucun doute l'un des composants les plus universels, mais aussi des plus complexes dans l'électronique actuelle, c'est le **FPGA**, dédale de portes et d'autres briques logiques et numériques, pour composer, sur une seule puce, les circuits intégrés dont vous rêviez et dont vous avez besoin.

Le laboratoire Elektor a conçu une nouvelle carte d'expérimentation pour FPGA, pour faciliter la vie des électroniciens désireux de se familiariser avec cette famille de logique programmable aussi excitante qu'elle est exigeante. Cette carte servira aussi de support à une série d'articles qui expliqueront comment s'y prendre, et proposeront par exemple la réalisation d'un projet ISE pour programmer cette carte de FPGA, ou montreront comment hiérarchiser le projet à l'aide de composants développés soi-même : un compteur-décompteur avec affichage sur deux chiffres à 7 segments….

Réf 120099-91

4 Éclairage annulaire à LED

L'éclairage annulaire est une source de lumière utilisée avec les ***webcams***, microscopes et appareils photo. Il se monte autour de l'objectif pour fournir un éclairage uniforme quand le sujet est très près. Les sources annulaires sont utilisées pour le portrait ou la photo rapprochée, ou pour éclairer des objets sous une loupe binoculaire. Assemblez vous-même un éclairage annulaire puissant à partir d'un kit de composants.

Astuce : Avant de confirmer votre commande sur le site d'ElektorPCBService, vous devriez considérer la possibilité d'y joindre l'excellent livre (en anglais) ***Mastering Surface Mount Technology***, le deuxième de la série ***LabWorX*** d'Elektor. Ce livre vous emmène dans un cours accéléré sur les techniques, trucs et savoir-faire en vue d'introduire les composants montés en surface (CMS) dans vos méthodes de travail. Visitez www.elektor.fr/labworx pour plus d'information.

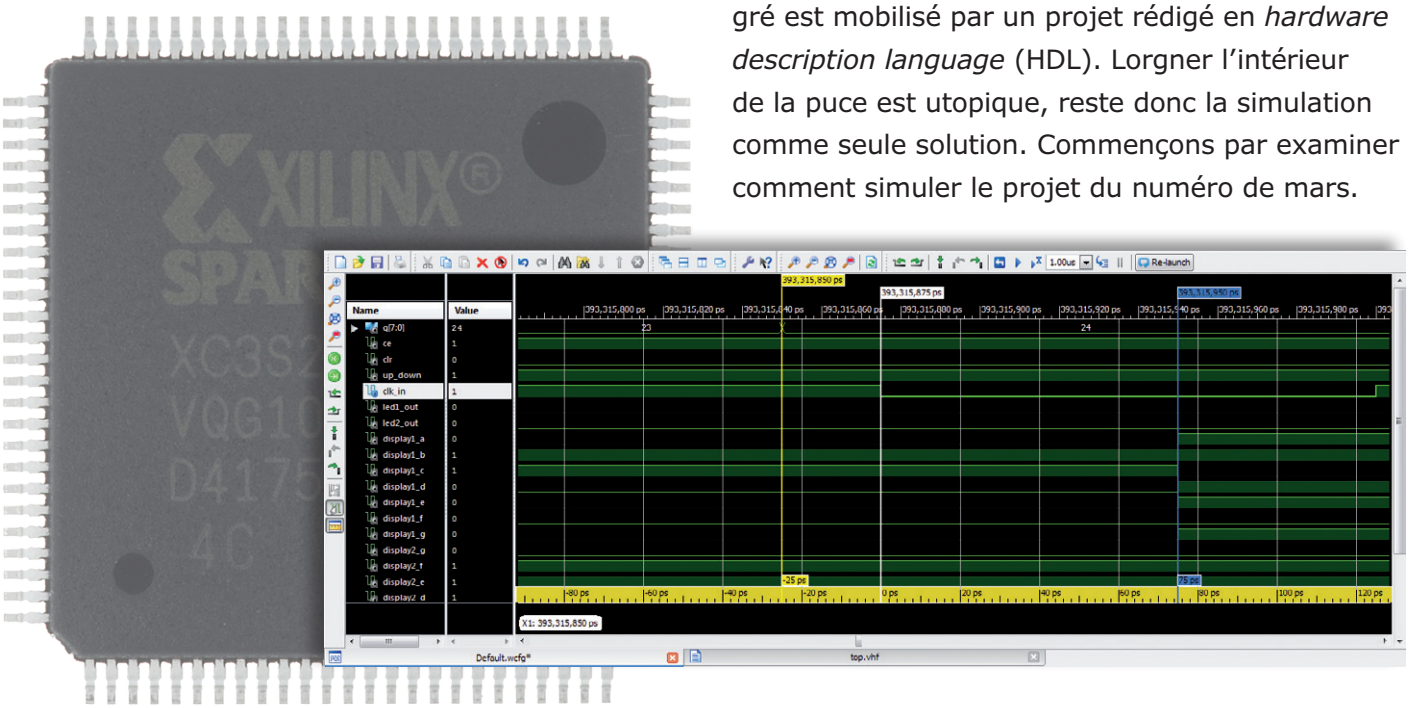
Réf 129003-71

et l'homme créa sa puce (4)

250 000 portes pour faire semblant

Clemens Valens
(Elektor.Labs)

La simulation tient une place importante dans la programmation d'une application sur FPGA. Tant qu'on ne joue qu'avec quelques portes, la réussite est aisée à atteindre. Mais c'est une autre paire de manches quand presque tout le circuit intégré est mobilisé par un projet rédigé en *hardware description language* (HDL). Lorgner l'intérieur de la puce est utopique, reste donc la simulation comme seule solution. Commençons par examiner comment simuler le projet du numéro de mars.



L'erreur est humaine, même en créant sa puce, mais l'homme n'est pas seul en cause. Il faut aussi compter avec l'architecture du FPGA et les imperfections des outils de développement. Et quand vous devez travailler avec des horloges à grande vitesse, des détails de ce genre prennent de l'importance, quand ils ne font pas tout rater. L'allongement des distances et des temps de parcours dans la puce peut induire des parasites. Les programmeurs de FPGA ont recours à quantité de simulations pour dépister les dysfonctionnements. Ils observent aussi si le projet n'est pas occupé à faire des choses qu'il ne devrait pas.
Il existe des simulations pour les différents stades de développement. Au commencement, il y a la simulation fonctionnelle pour vérifier que le travail est organisé de manière correcte. Mais cela ne suffit pas, le projet devra encore passer d'autres épreuves. Chaque étape risque d'introduire de nouvelles erreurs et il faut refaire des simulations pour garder l'ensemble sur les rails. Si la simulation finale, après la dernière étape du développement, est une réussite, la probabilité est grande que le projet fonctionne, mais ce n'est pas encore une garantie, parce qu'une simulation n'est jamais qu'un modèle mathématique qui imite le comportement de l'objet. La réalité peut encore réserver des surprises.

L'ouverture du simulateur

Attaquons la simulation du compteur-décompteur BCD à deux chiffres, le projet de la fois dernière. Je l'avais appelé *part3* (cf. [3]), je le recopie d'abord sous le nom de (qui l'eût cru ?) *part4*. Vérifiez que le nouveau projet est ouvert dans ISE avant d'aller plus loin.

Le simulateur ISE se trouve sous l'onglet Design, au-dessus, où l'on voit *View*. Dans le précédent épisode, c'était l'*Implementation View* qui était de mise ; aujourd'hui, ce sera la *Simulation View* qui sera active. Au moment où vous lancez la fonction d'un clic sur *Simulation*, vous voyez que le contenu de la fenêtre *Processes* se modifie. À remarquer aussi que, juste au-dessus de la fenêtre *Hierarchy*, un nouveau bloc est apparu avec la mention *Behavioral*. Comme nous sommes partis d'un projet non encore compilé, il n'y a qu'une seule ligne dans la fenêtre *Processes* : *Design Utilities*.

Sélectionnez *top* dans la hiérarchie de manière à rendre visible le processus *ISim Simulator* dans la fenêtre correspondante. Développez cette ligne d'un clic sur le + puis choisissez *Simulate Behavioral Model* et cliquez sur le bouton à l'extrême gauche, juste sous la flèche verte. Bien sûr, on peut aussi choisir *Run* dans le menu déroulant qui s'installe d'un clic droit sur la ligne correspondante ou d'un double clic sur cette même ligne. Alors, ISE se met en marche et après un certain temps, espère-t-on, on voit dans la fenêtre de *Console* le message « *completed successfully* ». Si tout va bien, une fenêtre ISim s'ouvre aussi, à côté d'ISE (mais pas dedans), qui représente un écran d'analyseur de spectre (**figure 1**). Dans la console, sous l'écran ISim, on peut lire qu'on est en mode *Lite* du fait qu'on n'a pas encore (provisoirement) de vraie licence. On y découvre aussi que la résolution temporelle de la simulation est d'une picoseconde (1 ps = 10^{-12} s). Au-dessus, à droite de l'écran, on voit que la simulation totale couvre une période de 1 µs.
Commençons par une expérience !

La simulation manuelle

Cliquer sur le bouton de redémarrage (il est bleu avec une flèche blanche cassée vers la gauche) ou par le menu *Simulation*. Un clic droit sur le signal ***ce*** dans la liste des signaux (il y en a deux, mais peu importe lequel, ça marche aussi en cliquant sur la reproduction graphique du signal). Il faut veiller à ne sélectionner qu'un seul signal à la fois. Dans le menu déroulant qui s'affiche, choisir *Force Constant*, car il s'agit ici d'un signal plus ou moins constant, certainement en comparaison d'une horloge. ISim ouvre alors une boîte de dialogue pour y remplir quelques paramètres. Il faut mettre « 1 » dans le champ *Force to value* et « 1ms » dans *Cancel after Time Offset*. Laisser inchangés les autres champs et cliquer sur *OK*. Alors, le signal ***ce*** passera au niveau haut dès le lancement de la simulation et y restera pendant une milliseconde. Recommencer la procédure pour le signal `up_down` et aussi pour `clr`, mais ici avec une *Force Value* de « 0 ».
Pour le signal `clk_in`, utilisons une procédure similaire, mais en choisissant *Force Clock* au lieu de *Force Constant*. La boîte de dialogue *Define Clock* qui apparaît ressemble fort à celle de *Force Selected Signal*, mais il s'agit d'autre chose. Rem-

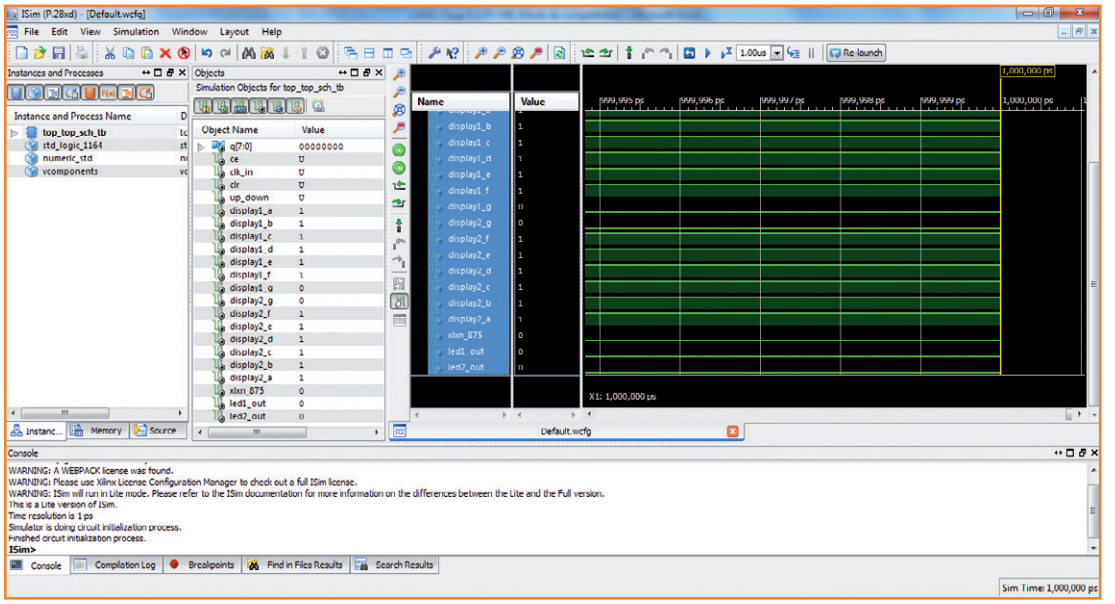


Figure 1.
Sim dans toute sa splendeur. Rien d'intéressant à voir dans la simulation, puisqu'il n'y a pas encore de stimuli définis.

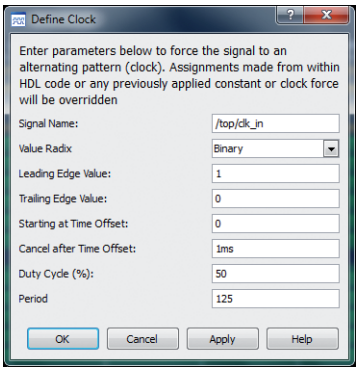


Figure 2.
La fenêtre *Define Clock*. Les périodes sans unités sont en picosecondes.

plissez-y les valeurs de la **figure 2** et cliquez sur *OK*. Elles font en sorte que le signal démarre sur un flanc montant au temps 0, reste actif pendant 1 ms avec un rapport cyclique de 50 % et une période de 125. Mais 125 quoi ? 125 ps, parce que l'unité de temps par défaut s'exprime ici en picosecondes. Quand on ne spécifie pas d'unité, ISim utilise l'unité par défaut. La fréquence d'horloge du FPGA de 8 MHz sur notre carte correspond à une période de 125 ns. Indiquez donc « 125 ns » et pas 125.

Tous les signaux d'entrée étant définis, lançons la simulation. Quand on le fait d'un clic sur le bouton *Run All* (ou via le menu *Simulation*), elle continue jusqu'à ce qu'il n'y ait plus d'événement ou jusqu'à la découverte d'un point d'arrêt. Comme nous n'avons défini aucun *breakpoint*, la simulation ne s'arrêtera pas de sitôt faute de stimuli, puisqu'une horloge produit sans cesse des événements, mais bien après 1 ms. Pour interrompre la simulation, on peut toujours cliquer sur le bouton *Break*, celui qui porte le symbole de pause. Mais voilà, après un arrêt forcé, ISim bloque et ouvre un fichier source VHDL, le plus souvent top.vhd, mais parfois le banc de test, alors la vue de la forme d'onde reste en arrière-plan, on la soupçonne grâce à l'onglet wcfg. Cliquer dessus la ramène à l'image.

Il est possible d'exécuter la simulation par étapes au moyen du bouton *Run*, le petit triangle bleu avec un sablier, à côté de l'autre, le bouton *Run All*. Mais au préalable, il faut indiquer la durée de la simulation dans la case voisine. Tapez p.ex. « 10ms » pour pouvoir observer une période de 10 ms. Chaque clic sur ce bouton fait avancer la simulation de 10 ms.

Quand la simulation s'est arrêtée d'elle-même, donc sans intervention extérieure, on peut observer les signaux comme à la **figure 3**. Il est alors loisible de zoomer en avant ou en arrière et aussi de faire défiler les signaux. On peut également en modifier l'ordre pour les grouper selon ses désirs en les tirant vers le haut ou vers le bas. Si vous ne voyez pas de changement, vérifiez la couleur des signaux d'entrée. S'ils sont orange au lieu de vert, c'est qu'ils ne sont pas définis, il y a donc une erreur quelque part.

Avec un bon facteur de zoom, vous pouvez voir les changements de niveau du signal d'horloge autant que des autres. Pour les bus, c'est à dire des signaux à plusieurs bits tels que q[7:0], la valeur courante est donnée en notation binaire. C'est le réglage par défaut, mais vous pouvez le modifier par l'option *Radix* (base de numération) dans le menu déroulant où se trouve *Force Constant*. Vous pouvez ainsi voir facilement le fonctionnement d'un compteur et déterminer dans quel sens il va. À gauche des formes d'ondes, vous pourrez lire la valeur de tous les signaux à l'instant correspondant à la position du curseur jaune.

Au banc d'essai

La simulation d'un programme selon la méthode manuelle ci-dessus est amusante, mais pas très utile. Même quand un circuit fonctionne bien, il y a toujours une chose ou l'autre à vérifier, mais si vous effectuez des modifications à répétition, il faut chaque fois redéfinir les signaux dans ces boîtes de dialogue malcommodes, vous en aurez vite assez. Si vous dactylographiez vite et bien, vous pouvez plutôt choisir de saisir les commandes en console. Regardez donc dans la console après avoir fermé par exemple une fenêtre *Force Constant*. Mais pour ça, il faut d'abord savoir comment faire. Voyez à ce sujet le document plugin_ism.pdf, disponible sur le site de Xilinx. Une meilleure méthode, plus puissante encore et recommandée par Xilinx justement, consiste à automatiser la simulation au moyen d'un banc d'essai.

Pareil *test bench* (ou *test fixture*) n'est rien d'autre qu'un fichier à ajouter au projet ISE pendant que la fenêtre de *Simulation* est active. On le fait de la manière habituelle par *New Source* d'un clic droit dans la fenêtre *Hierarchy*. Le *New Source Wizard* se présente, on choisit le VHDL Test Bench auquel on donne un nom, on vérifie que la case *Add to project* est cochée et l'on clique sur *Next*. Il faut encore attribuer le banc d'essai à un fichier source. Puisque c'est l'ensemble du

Banc d'essai en Verilog

Tout comme il existe de nombreux langages de programmation pour les ordinateurs, il y en plusieurs pour la programmation des FPGA. J'ai utilisé le VHDL dans cet article parce que ISE semble lui donner la préférence, mais ce n'aurait pas été plus mal avec Verilog. Alors, pour ne léser personne, je recommence la construction d'un banc d'essai, mais en Verilog.
Le processus d'horloge se fait avec une instruction `always` (attention, à l'inverse de VHDL, Verilog fait bien la différence entre bas de casse et capitale) et une ligne suffit, grâce à l'opérateur d'inversion ! On règle la fréquence avec le symbole « # » qui introduit un retard, dans ce cas-ci de 62,5 ns, parce que l'unité par défaut est ici la nanoseconde, bien que la résolution reste de 1 ps.

```
always
  #62.5 CLK_IN = !CLK_IN;
```

Toutes les 62,5 ns, le niveau de `clk_in` est inversé. Ça marche à condition de donner à `clk_in` une valeur initiale. Pour faire le signal d'horloge de la même façon qu'en VHDL, on peut écrire ceci :

```
always
begin
  CLK_IN = 1;
  #62.5;
  CLK_IN = 0;
  #62.5;
end
```

Avec ce bloc `always`, le bloc `clk_in` ne doit pas être initialisé. L'initialisation des stimuli se fait dans le bloc dit `initial` qui n'est exécuté qu'une seule fois au début de la simulation. La syntaxe ressemble à celle de VHDL et ici aussi, nous commencerons par une pause de 100 ns.

```
initial
begin
  #100;
  CLK_IN <= 0;
  CE <= 1;
  CLR <= 0;
  UP_DOWN <= 1;
end
```

Au lieu de « <= », vous pouvez aussi utiliser « = ». Par précaution, j'ai quand même initialisé `clk_in`. Ça ne sert peut-être à rien, mais au moins ça ne fait pas de tort.
Les blocs `always` et `initial` peuvent ensemble remplacer le bloc généré par ISE « `ifdef auto_init … endif` » placé à la fin dans le banc d'essai Verilog.

Prêt ? En route pour une simulation !

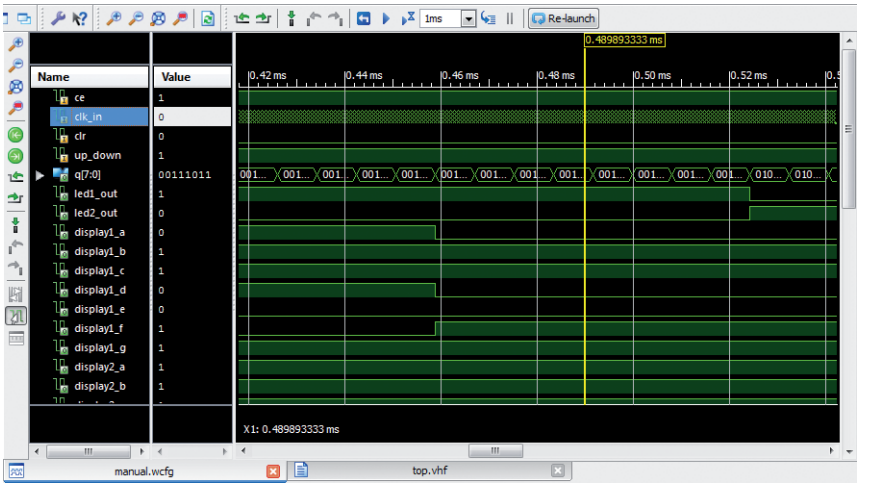


Figure 3.
La fenêtre des formes d'ondes après une simulation. Avec l'échelle utilisée, le signal `clk_in` change si souvent que ISim le hachure. D'un clic droit, vous pouvez adapter la base dans laquelle la valeur du bus est exprimée. Ici, elle est donnée en binaire.

circuit que nous allons tester, associons-le à *top*. Un clic sur Next pour que s'affiche le résumé habituel. C'est souvent sans intérêt, mais ici il permet au moins de vérifier que le fichier de test a bien obtenu l'extension VHD, preuve qu'il est devenu un fichier VHDL. Un clic sur *Finish* pour terminer la procédure.
Vérifiez bien que vous introduisez le banc d'essai dans la fenêtre de *Simulation*, pas dans celle d'*Implementation*. Si vous l'avez fait par inadvertance, corrigez la situation en mettant, dans les propriétés du fichier (*Source Properties*), *View Association* sur *Simulation*.
L'ajout d'un banc d'essai a plus de conséquences qu'il n'y paraît à première vue. C'est ainsi qu'après un certain temps, une ligne s'inscrit dans la fenêtre de la hiérarchie, entre la puce et *top*, avec le texte mélodique top_top_sch_tb. Mais *top* aussi a été modifié en *UUT – top*, dans lequel UUT signifie *Unit Under Test*. La hiérarchie indique donc clairement ce qui est sous investigation et par qui. Entre-temps, le fichier test_bench a été ouvert dans l'éditeur et comme vous le voyez, il y a plein de choses dedans.
Amusons-nous à y ajouter encore un autre banc d'essai, mais cette fois-ci en Verilog à la place de VHDL, de quoi voir la différence entre les deux langages. Tout se déroule de la même façon que ci-dessus, il n'y a qu'à prendre un *Verilog Test Fixture* comme type de source. Après l'adjonction d'un second banc d'épreuve, la fenêtre de hiérarchie prend une toute nouvelle structure. J'avais pensé qu'il était facile de mettre en œuvre plusieurs bancs d'essai dans le même projet, mais ça n'a pas marché. ISE donne manifestement la priorité au banc VHDL. On remarque que le fichier Verilog a l'extension V et son icone est différent de celui du fichier VHDL. Comme leurs extensions sont différentes, il est possible d'attribuer le même nom aux deux bancs. À quoi ça servirait est une autre histoire.

Zut, du VHDL maintenant !

Le banc d'essai n'est qu'un squelette, activer les bons signaux au bon moment, c'est pour notre pomme. Pour tester le compteur, il nous faut reproduire en VHDL ou Verilog ce que nous avons fait manuellement : un signal d'horloge et quelques niveaux constants. Entamons le boulot par les niveaux constants.
À la fin du banc VHDL se trouve le fragment de code suivant (les instructions réservées du VHDL sont écrites ici en capitales, mais ce n'est pas nécessaire, le VHDL ne fait pas la différence entre les casses) :

```
tb : PROCESS
BEGIN
  WAIT;
END PROCESS;
```

C'est à peu près la routine principale du banc d'essai, on dit le *process*. Il s'appelle `tb`, il ne fait rien, il attend. Un process ressemble fort à une routine dans un programme d'ordinateur, sauf que les process s'exécutent simultanément, en parallèle, plutôt qu'à la queue leu leu. On pourrait parler d'un multitâche idéal. À moins d'être interrompu, un process est aussi exécuté sans fin, comme ici dans une boucle infinie `WAIT`.

les process tournent en parallèle : un multitâche idéal

Ajoutons-y les signaux `ce`, `clr` et `up_down` pour pouvoir leur attribuer un niveau comme ceci :

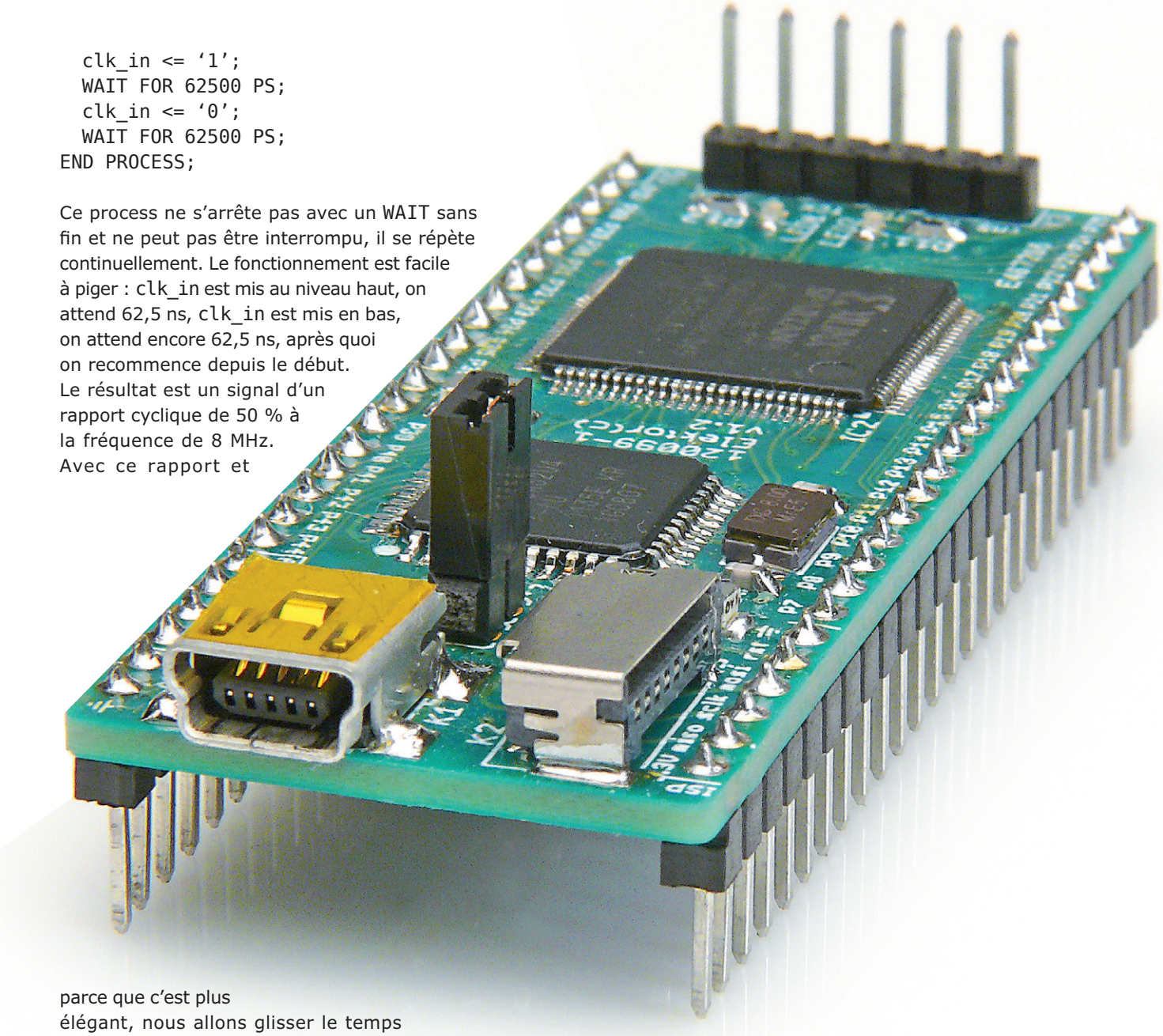
```
tb : PROCESS
BEGIN
  WAIT FOR 100 NS;
  ce <= '1';
  clr <= '0';
  up_down <= '1';
  WAIT;
END PROCESS;
```

Avant de faire attendre le process, mettons les signaux `ce` et `up_down` au niveau haut et `clr` au niveau bas. Comme ces instructions s'exécutent en même temps, pas successivement, leur ordre n'a pas d'influence. La ligne `WAIT FOR 100 NS` fait patienter le process 100 ns jusqu'à la fin du signal d'initialisation. Comme le fichier d'aide de ISim nous le conseille, suivons-le.
Générer le signal d'horloge est un peu plus compliqué, il nous faut un process séparé qui sera exécuté en même temps que `tb`.

```
clock : PROCESS
BEGIN
```

```
  clk_in <= '1';
  WAIT FOR 62500 PS;
  clk_in <= '0';
  WAIT FOR 62500 PS;
END PROCESS;
```

Ce process ne s'arrête pas avec un WAIT sans fin et ne peut pas être interrompu, il se répète continuellement. Le fonctionnement est facile à piger : clk_in est mis au niveau haut, on attend 62,5 ns, clk_in est mis en bas, on attend encore 62,5 ns, après quoi on recommence depuis le début. Le résultat est un signal d'un rapport cyclique de 50 % à la fréquence de 8 MHz. Avec ce rapport et parce que c'est plus élégant, nous allons glisser le temps d'attente dans une constante que nous ajouterons au bas de la liste SIGNAL :



```
CONSTANT clock_half_period : TIME :=
62500 PS;
```

La constante clock_half_period est du type TIME que nous pouvons utiliser avec l'instruction WAIT. Le process devient alors :

```
clock : PROCESS
BEGIN
  clk_in <= '1';
  WAIT FOR clock_half_period;
  clk_in <= '0';
  WAIT FOR clock_half_period;
END PROCESS;
```

Maintenant que le banc d'essai est prêt, nous pouvons lancer la simulation. Il faut d'abord sélectionner, dans la hiérarchie, le banc pour rendre visible le process ISim. Puis déployer la ligne pour montrer deux options : *Behavioral Check Syntax* et *Simulate Behavioral Model*. Exécuter le premier process pour voir s'il n'y a pas de faute de frappe. Si la flèche obtenue est verte, continuer. Ouvrir les propriétés de simulation d'un clic droit

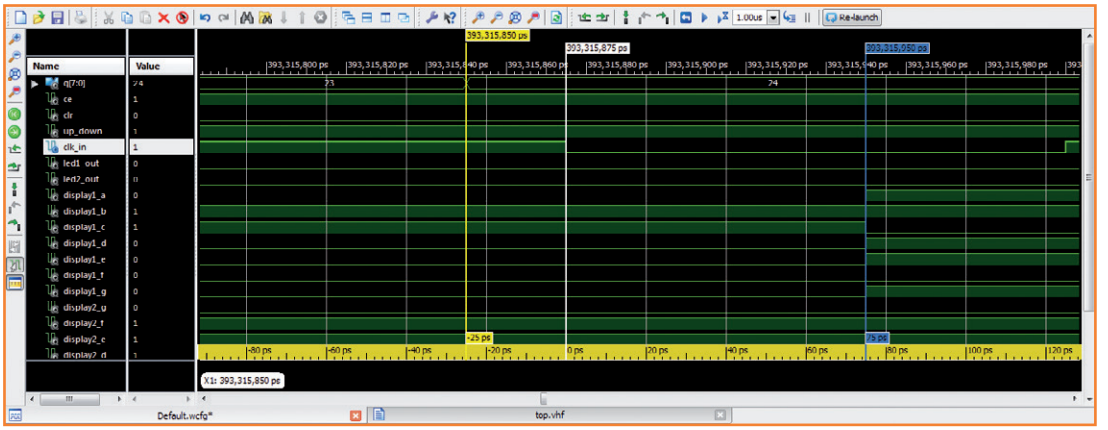


Figure 4.
Les marqueurs permettent de mesurer des intervalles de temps. Dans cette simulation, la période de l'horloge était de 250 ps au lieu de 125 ns. On repère le point zéro d'un marqueur blanc, ensuite ISim calcule la différence de temps par rapport aux autres marqueurs. On règle les unités de temps d'un clic droit sur l'axe du temps.

sur *Simulate Behavioral Model* et cocher l'option *Run for Specified Time*. La simulation va se poursuivre jusqu'à un clic sur *Break*, sans quoi, la simulation durerait le temps indiqué. Fermer la fenêtre de dialogue des propriétés et lancer le process *Simulate Behavioral Model*. La fenêtre ISim s'ouvre, mais rien ne se passe. Lancer la simulation d'un clic sur le bouton *Run All* (ou sur *Run* pour le temps spécifié dans la barre d'outils). Laisser tourner la simulation jusqu'à ce que le temps choisi apparaisse dans la ligne du temps. Cliquer ensuite sur *Break* pour arrêter la simulation. On retrouve alors les signaux comme lors de la simulation manuelle. Il peut arriver que sur l'axe du temps, le pas soit tellement petit qu'il donne l'impression que les signaux sont statiques. Le bouton *Zoom to Full View* met à l'écran la totalité de la simulation, ce qui permet d'observer les signaux lents. On peut ensuite zoomer sur les domaines les plus intéressants (**figure 4**). La séquence des signaux dans la fenêtre des formes d'ondes correspond à celle de la liste `SIGNAL`. Organisez donc cette liste à votre convenance, en prêtant toutefois attention aux virgules. Pensez aussi, chaque fois que vous voulez faire tourner un nouveau banc d'essai ou après une modification, à fermer la fenêtre ISim, non par peur des courants d'air, mais parce que sinon ISE, n'étant pas capable de le faire lui-même, vous envoie un message d'erreur.

Fini de faire semblant !

Nous voici arrivés à la fin de cet épisode. Vous êtes à présent en mesure de réaliser vous-même une simulation. De nombreuses choses n'ont pas pu être envisagées, je vous conseille donc d'approfondir cette matière complexe à l'aide d'internet ou de la littérature disponible. Simuler un objet n'est pas tellement difficile, mais le faire convenablement demande en revanche une solide connaissance de la logique programmable, des langages de programmation et des outils de développement. À vous de jouer !

(130065 – version française : Robert Grignard)

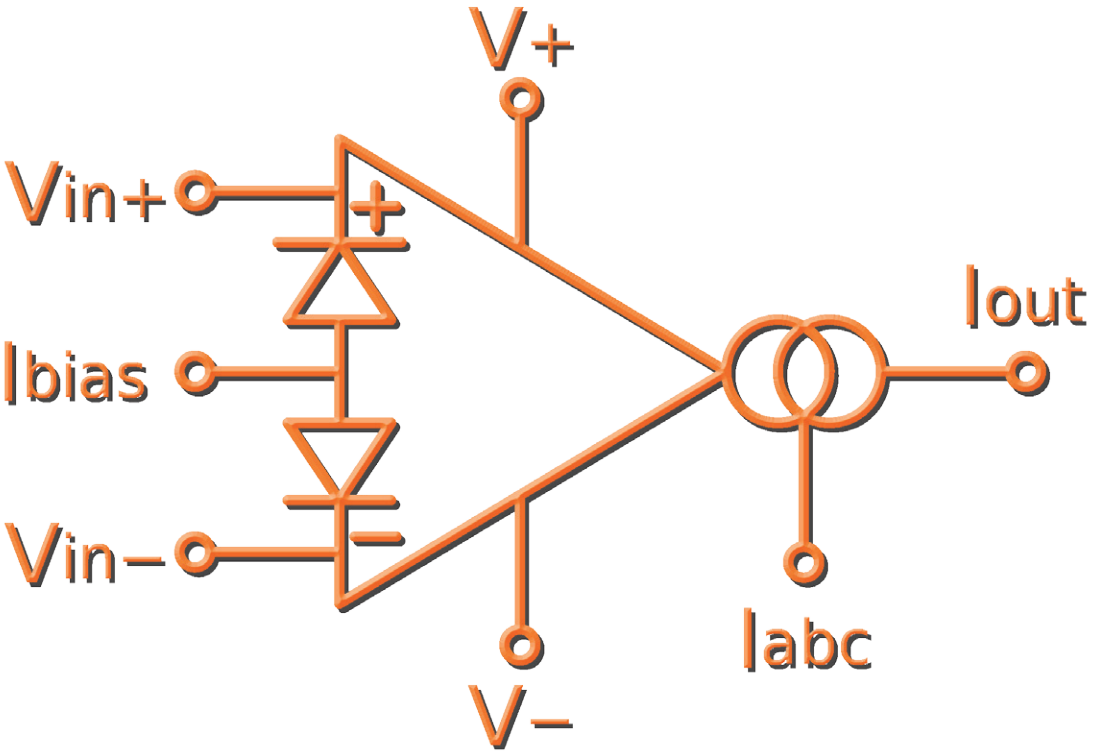
Liens

[1] 1re partie : www.elektor.fr/120099
[2] 2e partie : www.elektor.fr/120630
[3] 3e partie : www.elektor.fr/120743
[4] 4e partie : www.elektor.fr/130065

générateur d'ondes à OTA

triangles et carrés par transconductance

Si l'amplificateur opérationnel est mis à toutes les sauces, on pense rarement à adopter sa version à transconductance ou OTA. Or, elle offre des caractéristiques intéressantes dont j'ai tiré profit dans cette application d'oscillateur non sinusoïdal pour délivrer des signaux triangulaires et rectangulaires.



L'attrait majeur de l'amplificateur opérationnel à transconductance (OTA) réside dans la particularité de modifier ses paramètres en ne changeant que son courant de polarisation I_{abc} sans toucher aux autres composants. L'OTA a fait l'objet de nombreuses descriptions, dans des livres ou des articles comme dans les références [1], [2] et [3].

Mais qu'est-ce qu'un OTA ? Un convertisseur tension courant [4] dont le débit en sortie est proportionnel à la tension différentielle d'entrée. Comme illustration du fonctionnement de cet amplificateur, voici un circuit de générateur.

Comment ça marche

La **figure 1** en dévoile le schéma. Le circuit OTA1 forme avec le condensateur C un intégrateur, il le charge à courant constant I_{01}. La tension V_C varie linéairement en fonction du temps. Le circuit OTA2 constitue avec R1 et R2 un trigger de Schmitt. Le courant de sortie I_{02} développe aux bornes de R2 la tension de déclenchement V_{R2}. Quand la tension V_C atteint le seuil de déclenchement, la sortie de OTA2 bascule dans l'état opposé, ce qui change le sens du courant de sortie I_{02}, mais aussi la polarité de la tension de sortie V_2, laquelle sert de rétroaction à l'entrée de l'intégrateur. Sa sortie s'inverse alors et I_{01} fait

Libor Gajdošík
(République tchèque)

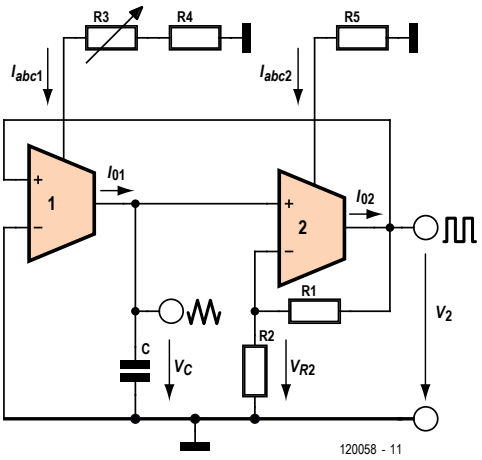
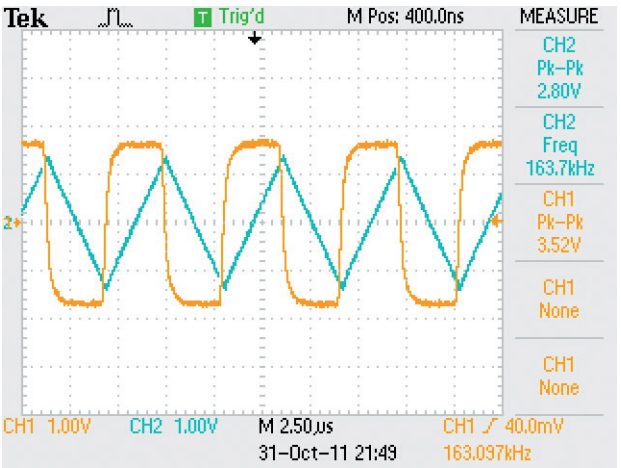


Figure 1.
Les composants nécessaires à la fabrication du générateur.

diminuer la tension sur le condensateur jusqu'à une nouvelle valeur V_{R2}. Et le circuit répète en boucle ce processus.
Le générateur, dans les conditions données, produit en sortie de l'OTA le courant de saturation dans le positif comme le négatif, parce que la tension d'entrée dépasse son niveau de régime linéaire, qui normalement se situe entre 20 et 50 mV. Si cette tension de déclenchement est beaucoup plus élevée, sa valeur réelle devient sans importance. Comme l'OTA dispose d'une entrée différentielle, les deux courants d'entrée sont faibles et voisins l'un de l'autre.

Figure 2.
Les formes d'ondes rectangulaire et triangulaire produites par le générateur à OTA.



Au cours d'une période T, le condensateur se charge à partir de t=0 et V_C=0 jusqu'au temps t=t_1, où V_C atteint la valeur maximum V_{Cm}. En équation, cela donne :

$$V_{Cm} = \frac{1}{C}\int_0^{t1} I_{01} \times \mathrm{d}t = \frac{I_{01}}{C} t_1 \qquad (1)$$

Cette valeur V_{Cm} est égale à la tension de déclenchement V_{R2} du trigger :

$$\frac{I_{01}}{C} t_1 = V_2 \frac{R_2}{R_1 + R_2} \qquad (2)$$

Comme V_2 apparaît aux bornes des résistances R_1 et R_2 à cause du courant I_{02}, on peut écrire :

$$V_2 = I_{02}\left(R_1 + R_2\right) \qquad (3)$$

En reportant (3) dans (2), on calcule le temps nécessaire t_1 :

$$t_1 = \frac{CR_2 I_{02}}{I_{01}} \qquad (4)$$

Nous supposons que les deux valeurs, négative et positive, du courant de sortie de l'OTA sont les mêmes. Puisque la forme d'onde triangulaire est constituée de parties linéaires, symétriques en horizontal et vertical, la période totale des oscillations vaut $T = 4t_1$ et la fréquence f se calcule comme suit :

$$f = \frac{1}{T} = \frac{I_{01}}{4CR_2 I_{02}} \approx \frac{I_{abc1}}{4CR_2 I_{abc2}} \qquad (5)$$

Les courants de saturation et de polarisation sont :

$$I_{01} \approx I_{abc1} \qquad \text{et} \qquad I_{02} \approx I_{abc2} \qquad (6)$$

Comme le diviseur de tension n'a aucune charge, l'amplitude de l'onde triangulaire V_{Cm} répond à l'équation :

$$V_{Cm} = V_2 \frac{R_2}{R_1 + R_2} = I_{02} R_2 \approx I_{abc2} R_2 \qquad (7)$$

Les approximations adoptées dans les dernières formules entraînent des écarts, mais leur ampleur

ne fausse pas l'estimation de fréquence ni de tension dans la conception du générateur.

Résultats pratiques

Les mesures ont été effectuées sur les composants suivants : OTA du type LM13700, résistances R_1=820 Ω et R_2=2 kΩ, condensateur C=1 nF. Les courants de polarisation proviennent de résistances branchées entre les entrées « bias » et la masse. Pour I_{abc1}, les valeurs sont R_3=1 MΩ, R_4=10 kΩ . Pour I_{abc2}, R_5=15 kΩ partout, de manière à maintenir constantes les valeurs de V_2 et V_{Cm} dans toutes les mesures. La tension d'alimentation est de ±10 V.

Le courant de sortie est évalué entre 14 µA et 0,87 mA et la fréquence va de 3,6 kHz jusqu'à 160 kHz. Nous avons mesuré un courant de sortie I_{02} de 0,57 mA. Les formes d'ondes obtenues sont à la **figure 2**.

(120058 – version française : Robert Grignard)

Références et liens

[1] T. Parveen, *Operational transconductance amplifier and analog integrated circuits*, New Delhi, India, I.K. International Publishing House, 2009.

[2] R.L. Geiger et E. Sanchez-Sinencio, «Active filter design using operational transconductance amplifiers: A tutorial», *IEEE Circuits Devices Magazine*, vol.1, pp.20 à 32, mars 1985.

[3] W.R. Grise. (1998, October). Operational Transconductance Amplifiers (OTA) for synthesis of voltage-controlled active-filter tuned oscillators. *TECHNOLOGY INTERFACE: The Electronic Journal for Engineering Technology*. Disponible en ligne sur : http://technologyinterface.nmsu.edu/fall98/electronics/grise/griseota.html

[4] http://philippe.roux.7.perso.neuf.fr/Resources/LM13600.pdf

réveil multifonction 1ère partie

instrument de torture et de délices matinaux

Michael J. Bauer
(Australie)

Pour beaucoup d'entre nous, en Australie comme à Cambridge ou à Bécon-les-Bruyères, le réveil est un outil de torture. Pourquoi nous contentons-nous de ces instruments aussi moches et mal conçus que la plupart des radios-réveils, alors que des appareils bien étudiés adouciraient nos peines matutinales ! Voici un réveil polyvalent, pour la conception duquel l'auteur n'a rien laissé au hasard.

Le point de départ de ce projet : la frustration causée par un radio-réveil d'une marque réputée, pas du genre bon marché, à l'ergonomie pour le moins douteuse. De plus, la plupart des fonctions que je souhaiterais lui font défaut. D'ailleurs, chaque fois que j'en parle à des amis ou des collègues, je me rends compte que tous les réveils et radios-réveils ne répondent pas vraiment à leurs attentes. Voici les doléances les plus courantes :

- il faut activer le réveil avant de se coucher (puisqu'il faut le désactiver le matin)
- il faut accorder correctement la radio, et régler correctement le volume, faute de quoi on n'entend rien lorsque le réveil l'allume
- le réglage de l'heure ou du réveil est fastidieux ; le défilement est soit trop rapide soit trop lent
- l'heure de réveil est la même tous les jours que ce soit en semaine, un samedi ou un dimanche alors que la plupart des gens se lèvent plus tôt en semaine que le week-end et certains travaillant à temps partiel ou en équipe ont des horaires différents chaque jour
- la durée du *snooze* (NdT : *dodo*) est fixe (habituellement 7 à 9 minutes) et ne peut pas être réglée
- il est pénible d'avoir à appuyer sur le bouton *snooze* pour faire taire le réveil. Il serait super de pouvoir régler le réveil pour qu'il sonne quelques secondes, se fasse de lui-même oublier quelques minutes, et recommence jusqu'à ce qu'on le désactive.
- le bouton *snooze* est trop petit, mal placé, ou difficile à utiliser dans le noir
- l'afficheur est trop lumineux de nuit, pas assez de jour
- le réglage de luminosité de l'afficheur est malcommode. Pourquoi la luminosité de ne s'adapterait-elle pas automatiquement à la luminosité ambiante ?

Les amateurs rivalisent avec la production de masse, non sur le prix, mais sur la qualité !

- je n'aime pas me réveiller au son de la radio, n'existerait-il pas un réveil proposant un son plus intéressant ?
- les bambins adorent jouer avec les boutons du réveil et le dérèglent souvent accidentellement
- le réveil ne fait pas le café.

Je suis sûr que la plupart de ces doléances sont valables aussi pour votre réveil, n'est-ce pas ?
J'ai essayé de trouver mon bonheur dans diverses boutiques, mais aucun des réveils proposés ne correspond à ce que je cherche. J'ai donc décidé de concevoir et assembler le réveil de mes rêves.

Un meilleur réveil

J'ai mis l'accent sur la combinaison de fonctions étendues à l'aspect pratique (facilité d'utilisation), tout en essayant de palier beaucoup de défauts des produits du commerce. Pas question de concurrencer l'industrie sur le terrain du prix. Les amateurs ne peuvent pas rivaliser avec la production de masse sur le prix, mais sur la qualité ! Il faudrait donc une interface utilisateur (panneau de commande) comprenant un afficheur, des poussoirs et un codeur rotatif pour régler le réveil lorsqu'il ne sera pas relié à un ordinateur. Il s'agit simplement de régler l'heure de réveil et de sélectionner les informations à afficher. En revanche, la programmation complète se ferait, si besoin, depuis une interface graphique sur PC grâce à un lien USB.
Le panneau de commande devra comporter un grand afficheur à 4 chiffres, affichant par défaut l'heure, ainsi que plusieurs indicateurs rétro-éclairés (p.-ex. PM, 24 h, A1–A4, S1–S4, MON, TUE, WED, etc). Les indicateurs de réveil (A1, A2, A3, A4) afficheront l'état (ON/OFF) des quatre évènements journaliers de réveil. Un autre jeu de quatre indicateurs indiquera l'état (ON/OFF) des sorties commandées par le réveil.
La luminosité de l'afficheur devra être réglable, et de préférence l'utilisateur pourra choisir entre, par exemple, l'utilisation d'un capteur de luminosité ambiante (mode automatique), un évènement de réveil (mode d'économie d'énergie), ou l'utilisation du panneau de commande.
La fonction réveil devra comporter toute une gamme d'options de configuration : choix du son de réveil, volume initial, volume maximal, croissance automatique du volume (option *ramp-up*), nombre de répétitions de l'alerte, intervalle entre les alertes, etc.
Il devra non seulement y avoir une bonne sélection de sons de réveil inclus, mais ceux-ci devront pouvoir être personnalisés par l'utilisateur. L'utilisateur devra pouvoir choisir un son différent pour chacun des évènements de réveil du programme hebdomadaire. Idéalement les sons de réveil seraient des fichiers audionumériques (codés en PCM/MP3), téléchargeables via le lien USB, permettant d'avoir un son de qualité avec une distorsion faible. Cette dernière fonction pourrait n'être qu'une option, car beaucoup d'amateurs ne seront pas prêts à accepter l'inévitable surcoût.
La fonction *snooze* pourrait avoir deux modes de fonctionnement. En mode classique, le son de réveil serait temporairement neutralisé en pressant une seule fois le bouton, mais réactivé après un temps préprogrammé (p. ex. 10 min). L'activation du réveil se répèterait jusqu'à ce qu'elle soit annulée (p. ex. en activant trois fois de suite et rapidement le bouton *snooze*) ou qu'un temporisateur préprogrammé arrive à échéance.
À l'inverse, en mode *auto-repeat*, l'activation du réveil se produirait automatiquement un certain nombre de fois, sans intervention, suivant un intervalle préprogrammé (p. ex. 10 min), avant de finir par s'annuler toute seule. Une unique action sur le bouton *snooze* permettrait, à tout moment, d'annuler les activations répétées.
Avec chacun de ces deux modes, il serait inutile de désactiver le réveil chaque matin et de le réactiver chaque soir avant d'aller se coucher !
Dans le mode de fonctionnement par défaut, le bouton du codeur rotatif permettrait de commander manuellement la luminosité de l'afficheur. Ce même bouton constituerait, dans les modes de réglage de l'heure et de configuration, un moyen pratique pour faire évoluer les données affichées.
Un moyen de confirmer, jusqu'à 48 h à l'avance, le prochain évènement de réveil en attente devra être fourni. On pourrait imaginer que le bouton *snooze/cancel*, quand aucune alarme n'est active, permette de rentrer dans un mode de vérifica-

tion où l'heure et le jour de la prochaine alarme seraient affichés. Si le bouton était maintenu appuyé pendant (disons) deux secondes, le son correspondant à la prochaine alarme en attente serait joué afin que l'utilisateur puisse vérifier le volume. En poussant un peu plus, on pourrait imaginer utiliser le bouton du codeur rotatif pour régler le volume sonore d'une part et une paire de poussoirs pour choisir le son utilisé.

Pour le plus grand plaisir des technophiles, le réveil devrait pouvoir être relié à un PC (par USB) et prendre en charge tout une gamme d'accessoires via des connecteurs d'E/S disposés à l'arrière du réveil. Ces derniers fourniraient des sorties logiques utilisables pour allumer ou éteindre des appareils à des heures programmées, et/ou à l'activation d'une alarme. Les accessoires pourraient par exemple être : une cloche ou un carillon, une lampe de chevet, une radio, une TV, une bouilloire, une machine à café, un ordinateur ou un de ses périphériques, etc. Chacune de ces voies devrait pouvoir être réglée indépendamment et différemment pour chaque jour de la semaine.

Il faudra fournir une batterie de secours afin que l'heure et la date soient conservées en cas de coupure de courant. Un avertissement sera affiché quand la tension de batterie deviendra insuffisante.

Tous les paramètres réglables par l'utilisateur devront être conservés dans une mémoire non-volatile. Ils pourront être transférés depuis ou vers un PC à l'aide d'un utilitaire sous Windows. Une fois reliée à un PC, l'horloge et la date devraient se synchroniser automatiquement avec celles du PC, ces dernières pouvant être synchronisées par internet.

En plus de la prise en charge d'une interface graphique *Windows* permettant de configurer l'horloge et programmer les alarmes, le port USB devra permettre de mettre à jour le micrologiciel. C'est essentiel pour prendre en charge les ajouts de fonctionnalités et corrections de bogues.

Technique

Le matériel comprend une carte principale et une carte d'affichage, un codeur rotatif, un haut-parleur et un bouton *snooze/cancel*. Les composants montés en surface ont été éliminés des deux cartes de base, pour le plus grand bonheur des amateurs possédant des talents de soudure limités ou peu de matériel. C'est également pour cette raison qu'un boîtier PLCC à 52 pattes a été choisi pour le microcontrôleur.

La carte principale comporte un générateur de son simple et bon marché. C'est le micro qui produit le son (attaque/décroissance, amplitude et volume global) par modulation de largeur d'impulsion (MLI). Avec l'aide de quelques astuces dans le micrologiciel, capable d'ajouter des effets de modulation de fréquence ou d'amplitude, le générateur est en mesure de produire toute une gamme de sons, de la petite musique agréable au bruit insupportable (il en faut pour tous les goûts). Une sélection de 16 sons prédéfinis est fournie avec le micrologiciel de base. D'autres sons peuvent être programmés par l'utilisateur en modifiant les paramètres *sound shape*, stockés dans l'EEPROM.

L'ensemble peut être mis dans un boîtier tout fait ou bien dans ce superbe boîtier que vous concevrez et fabriquerez pour l'occasion. Le boîtier standard est un PAC-TEC modèle CM6-225, mais des répliques chinoises à bas coût sont disponibles. Les cartes ont été conçues pour tenir dans ce boîtier (*référez-vous aux instructions d'assemblage que vous trouverez sur le site*). Placés au dessus de la carte d'affichage, les sept poussoirs de commande affleureront à la surface de la face avant, mais rien ne vous empêche de les mettre ailleurs, par exemple sur un morceau de plaque d'essai relié à la carte d'affichage. Faites jouer votre créativité !

Il est possible d'étendre le matériel à l'aide de cartes filles reliées à la carte principale avec un connecteur pour câble en nappe. Vous pourrez donc étendre les fonctions de l'horloge ou même utiliser le matériel pour tout autre chose. Je travaille déjà sur une carte capable de produire des sons PCM/MP3 de qualité. Elle comportera une puce de décodage MP3 et une mémoire flash sérielle pour stocker les clips audio.

Le micrologiciel est basé sur mon système d'exploitation ALERT (simple mais temps-réel). Rien ne vous empêche d'écrire (et partager) votre propre micrologiciel ! Vous pouvez également télécharger gratuitement ma version. Le code source est disponible sur [1] pour celles et ceux qui souhaiteraient le modifier, l'étendre ou tout simplement le lire.

Le matériel est basé sur le microcontrôleur USB AT89C5131 d'*Atmel*. Ce micro a été choisi pour sa disponibilité en boîtier PLCC, sa généreuse (32 Ko) mémoire flash et EEPROM (1 Ko), son contrôleur USB intégré, sa bonne disponibilité et son petit prix (environ 7 $ chez *Digi-Key*). Plus

Caractéristiques de l'horloge/réveil

- Aspect pratique et facilité d'utilisation
- Commande intuitive et menu de réglage
- Codeur rotatif pour une saisie aisée des heures et dates
- Accès à l'heure et au volume de la prochaine alarme avec seulement un bouton
- Plusieurs évènements d'alarme ainsi qu'un planning hebdomadaire
- Jusqu'à quatre évènements d'alarme par 24 h
- Possibilité d'heures d'alarme différentes en fonction du jour de la semaine
- Un large choix d'options dodo/annuler
- Mode classique — l'alarme sonne continuellement jusqu'à une action sur le bouton
- Mode Auto-repeat — l'alarme sonne périodiquement jusqu'à être annulée
- Intervalle de repos réglable (dans les deux modes)
- Affichage à luminosité variable, avec choix du mode de commande
- Luminosité ambiante
- Activé par alarme ou temporisateur
- Manuel — réglage de la luminosité avec le codeur
- Sons d'alarme programmables de haute qualité
- Plusieurs sons prédéfinis disponibles
- Personnalisez les sons existants ou créez les vôtres
- Possibilité d'attribuer un son différent à chaque alarme
- Lien USB avec un PC (1)
- Synchronisation de l'heure et de la date (p. ex. par internet)
- Sauvegarde et restauration des paramètres et différentes alarmes (2)
- Mise à jour du micrologiciel (nouvelles fonctions et corrections de bogues)
- Possibilité de relier des accessoires déclenchés par les alarmes
- Quatre voies indépendantes programmables sur sept jours
- Commande d'appareils secteur (lampe, radio, TV/AV, ordinateur & périphériques)(3)
- Sortie commandée par alarme pour dispositifs de réveil additionnels
- Compte à rebours auxiliaire avec sortie de commande
- Allumage automatique de périphériques PC lors de la détection du bus USB

(1) Le lien PC utilise l'API USB-CDC pour port série virtuel, également disponible à l'aide d'*HyperTerminal*.

(2) Facilité par une application graphique Windows qui reste à développer.

(3) Nécessite une carte optionnelle dotée de 4 relais statiques pour commander des appareils secteur.

important encore, le C5131 possède un chargeur de démarrage capable de programmer la mémoire flash via le port USB et on se passera de puce spéciale.
Le seul défaut du C5131, c'est son cœur 8051 qui (d'après moi) est l'un des pires jamais imaginés ! Si vous programmez en C et qu'il y a assez de place pour compenser l'inefficacité du jeu d'instruction, ce n'est pas si grave que ça. Pour être honnête, le jeu d'instructions du 8051 est plutôt efficace, mais lorsqu'il travaille sur les emplacements mémoire situés en deçà de 0xFF. Il est horriblement inefficace avec la RAM située au delà.
Ceux qui voudront bricoler le micrologiciel seront ravis d'apprendre qu'il existe un compilateur C gratuit pour le 8051 ; cherchez « SDCC 8051 compiler » sur votre moteur de recherche favori. Par-dessus tout, je pense que le 89C5131 était un très bon choix pour les amateurs qui ne sont pas équipés pour souder les CMS. Pour un produit commercial, j'aurais choisi un autre micro comme un MC9S08-JM60 de *Freescale*, un PIC18F66J50 de *Microchip*, ou même un micro avec un cœur ARM à 32 bits (leur coût est comparable à celui de beaucoup de micros à 8 bits). Tous les composants sont faciles à trouver, notamment en ligne, mais aussi à la boutique du coin (si vous avez la chance d'avoir encore une boutique du coin). Avant de commander « moins cher » à l'étranger, vérifiez l'absence de tous frais cachés.

Fonctionnement

L'alimentation

L'horloge est conçue pour être alimentée par un adaptateur 9 V continu. L'horloge elle-même consomme moins de 100 mA (avec l'afficheur réglé pour une luminosité normale). Je vous conseille toutefois d'utiliser un adaptateur régulé par découpage pour minimiser la consommation à la prise. L'horloge utilise un tel régulateur pour produire le 5 V.
Vous pourrez également alimenter l'horloge via le bus USB (V_{bus} = 5 V_{cc}). L'alimentation disponible sur les prises pour accessoires provient directement de l'alimentation externe 9 V ($+V_{EX}$) et ne sera donc pas disponible lorsque l'horloge est alimentée via l'USB. Dans ce dernier cas, les accessoires nécessitant le 9 V ne fonctionneront bien évidemment pas. La batterie de secours

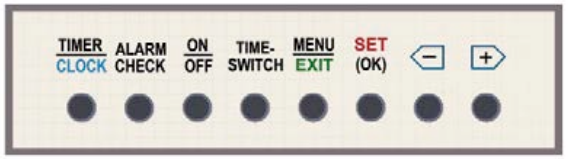


Figure 1.
Légende des boutons du dessus.

permet de conserver l'heure lorsque l'horloge n'est reliée ni à l'USB ni à l'adaptateur 9 V.

Batterie de secours

L'horloge n'a pas été conçue pour être alimentée par batterie pour un fonctionnement normal. L'unique fonction de la batterie est de permettre la conservation de la date et de l'heure lors des coupures de courant ou le temps de changer l'horloge de prise ou de la brancher au PC. Lors du fonctionnement sur batterie, seul le strict nécessaire pour conserver l'heure et la date sera alimenté.
La batterie de secours est constituée de trois cellules AA. Vous pourrez utiliser des piles 1,5 V ou bien des accumulateurs 1,2 V. Le microcontrôleur continuera de fonctionner avec une tension de batterie comprise entre 3,3 V et 5 V. Celui-ci surveille continuellement la tension de batterie lorsqu'il est alimenté par le 9 V. Un interrupteur permet de découpler la batterie quand vous souhaitez laisser l'horloge éteinte de manière prolongée.

Alimentation pour les accessoires

Les connecteurs pour accessoires *Alarm* et *Time-Switch* fournissent une alimentation continue protégée issue du 9 V. L'alimentation externe est limitée en courant par un fusible réamorçable de 0,5 A. La sortie de commande *alarm-switch* est capable de fournir jusqu'à 0,5 A à un accessoire (cloche/carillon, lampe, radio, etc).

Figure 2.
Sérigraphie de la face avant.

Téléchargement du micrologiciel

Le micrologiciel est chargé dans la mémoire flash du micro par l'USB à l'aide de l'utilitaire FLIP (*Flexible In-system Programmer*) d'Atmel. La dernière version de FLIP et sa documentation peuvent être téléchargées sur le site d'Atmel.
Pour démarrer le chargeur de démarrage du micro, maintenez appuyé le bouton ISP (*In-System Programming*), puis pressez et relâchez le bouton *reset*, relâchez ensuite le bouton ISP. L'affichage devrait être vide. Branchez le câble USB à votre PC. Démarrez l'utilitaire FLIP et suivez les instructions fournies par Atmel.

Interface de commande

Le micrologiciel intègre un protocole de communication destiné principalement à permettre l'échange de données avec un PC via l'USB. Le protocole est basé sur une interface en ligne de commande. Comme celui-ci utilise des caractères ASCII imprimables, l'interface peut être utilisée par un humain via un terminal (p. ex. *HyperTerminal*). L'interface peut être utilisée interactivement pour régler les options de l'horloge et autres paramètres. Les options de configuration avancée, qui ne sont pas accessibles via la face avant, peuvent être réglées via l'interface de commande en utilisant *HyperTerminal*, ou l'aide d'une application *Windows* qu'il faudra développer.
L'horloge se présente au PC comme un port série virtuel. Les logiciels *Windows* peuvent communiquer avec l'horloge via les appels standards de l'API COM de *Windows* (*open, read, write*, etc). Une chaîne de commande est composée d'un mnémonique de deux lettres et d'un certain nombre d'arguments fournis par l'utilisateur. Certaines commandes n'acceptent pas d'arguments. Un seul espace soit être inséré entre les arguments d'une commande en acceptant plusieurs, ainsi qu'entre la commande et son premier argument. L'interface de commande n'est pas sensible à la casse. Il n'a pas été prévu de pouvoir éditer une ligne de commande, mais vous pourrez annuler la ligne en cours avec Échap ou Ctrl-X.
La commande d'aide HE permet d'afficher un bref résumé des différentes commandes. Des commandes de débogage sont fournies pour assister les programmeurs désireux de développer leur propre micrologiciel ou d'explorer les entrailles du microcontrôleur.
Pour plus de détails sur l'interface de commande, référez-vous au guide de l'utilisateur contenu dans l'archive téléchargeable [1].

Face avant

Vous pourrez admirer sur la **figure 1** la disposition standard des poussoirs sur le dessus de l'horloge. Le bouton SNOOZE/CANCEL se trouve également sur le dessus. Les boutons, l'afficheur à LED et le codeur rotatif constituent l'interface de l'utilisateur. La plupart des fonctions et paramètres de l'horloge sont accessibles via cette interface. Les fonctions les plus ésotériques ne seront réglables que via le lien USB.

Contrairement à la plupart des réveils, l'afficheur de celui-ci indique le jour de la semaine. Lorsque vous paramétrez une alarme, vous pouvez choisir le(s) jour(s) pour le(s)quel(s) elle sera active. D'autres indicateurs à LED affichent l'état des quatre évènements d'alarme A1-A4 et des quatre sorties commandées (S1–S4). La face avant (**fig. 2**) permet de commander manuellement l'état des sorties.

La plupart des poussoirs possèdent plus d'une fonction, selon le contexte. Par exemple, dans le mode de fonctionnement standard, une action sur les boutons [+] ou [−] sélectionnera un élément à afficher, par exemple les secondes, l'année, la date (mois et jour), etc. En mode de réglage, les boutons [+] et [−] serviront à incrémenter ou décrémenter une valeur numérique, ou faire défiler une liste d'éléments, par exemple les jours de la semaine.

En général, lorsque vous êtes en mode réglage, le clignotement d'un chiffre ou d'une LED indique qu'il peut être changé à l'aide d'un bouton ou du codeur rotatif. Le bouton EXIT permet de quitter le réglage en cours sans effectuer de changement.

Vous trouverez dans le **tableau 1** un résumé des différentes fonctions de l'interface utilisateur. La colonne de gauche contient les opérations qui peuvent être effectuées dans le mode standard. Vous trouverez plus de détails dans le guide de l'utilisateur.

Dans le deuxième et dernier volet que vous trouverez dans le prochain numéro d'Elektor, nous parlerons des schémas et de l'assemblage de l'horloge.

Ce sera passionnant, épatant, surprenant...

(100149 – version française : Kévin PETIT)

Liens

[1] www.elektor.fr/100149

[2] www.elektor-labs.com

Tableau 1. Fonctions des boutons

action sur les boutons ou autres commandes	effet
Appui sur le bouton TIMER/CLOCK	Le compte à rebours est affiché (temps de départ ou restant, mm.ss). Il peut à partir d'ici être réglé ou démarré.
Appui sur le bouton ALARM CHECK	Entrée dans le mode alarme. L'heure programmée pour un événement d'alarme donné (A1 à A4) est affichée. L'alarme sélectionnée (heure et jour de la semaine) peut être (re)programmée, activée ou désactivée. Les boutons [+] et [−] permettent de sélectionner le jour de la semaine pour l'alarme en cours de réglage. Le bouton ON/OFF change l'état de l'alarme. Un appui sur le bouton SNOOZE en mode alarme permet de changer le son et le volume.
Appui sur le bouton TIME-SWITCH	Entrée dans le mode Time-Switch. Une sortie programmable (S1–S4) peut être activée ou désactivée (outrepassant temporairement le planning programmé).
Maintien appuyé du bouton SET pendant plus de 3 s	Entrée dans le mode de réglage de l'heure. Le codeur permet de régler les heures, puis les minutes après un deuxième appui sur le bouton SET. Un dernier appui sur SET permet d'enregistrer les changements. Utilisez le bouton EXIT pour sortir sans enregistrer.
Maintien appuyé du bouton MENU pendant plus de 3 s	Entrée dans le mode de réglage. Plusieurs options et paramètres peuvent être réglés à partir de ce menu. Les boutons [+] et [−] permettent de faire défiler les entrées du menu.
Appui sur le bouton [+] ou [−]	L'afficheur affiche désormais les secondes ou la date (année, mois, jour). Utilisez le bouton [+] pour faire défiler les éléments : secondes (ss.cc), année (20xx), date (MM dd), puis retour à l'heure. Le bouton [−] permet le parcours inverse. Le bouton SET peut être utilisé pour régler les différentes valeurs.
Appui sur le bouton SNOOZE/CANCEL	Configuration de l'alarme : l'heure de la prochaine alarme des 48 h qui suivent est affichée. Si le bouton est maintenu pendant plus de 2 s, le son d'alarme est joué. Un appui sur le bouton SET tout en maintenant le bouton SNOOZE permet de choisir le son et régler le volume.
Manœuvre du codeur (lorsque l'heure est affichée)	Réglage de la luminosité de l'afficheur. En fonction des options choisies, le réglage peut être conservé ou changé automatiquement plus tard.

Raspberry Pi : un an, un million

Entre l'arrivée du premier lot de Raspberry Pi et aujourd'hui, pas moins d'un million de cartes ont été vendues. Sur le salon *Embedded World 2013* à Nuremberg nous avons rencontré Peter Lomas, le concepteur du matériel : excellente occasion de prendre quelques nouvelles de la fondation Raspberry Pi.

Entretien :
Clemens Valens
Transcription :
Joshua Walbey

C: *Raspberry Pi, un phénomène, une histoire étonnante.*
P: Oh que oui. On me demande souvent pourquoi nous l'avons fait tel qu'il est et ce qui fait la différence avec les autres. Une des clés de son succès, je pense, a été de communiquer dès les débuts ; nous avons présenté l'un de nos tout premiers prototypes sur un blog anglais à destination d'un des correspondants de la BBC, Rory Cellan-Jones, qui en a fait une vidéo YouTube vue 600.000 fois. Ça a donné une campagne de marketing viral très bénéfique pour le *Raspberry Pi*. Le choix du nom n'est pas étranger à son succès, et sans doute aussi le logo créé par Paul Beach ; c'est devenu une véritable icône.

C: *J'ai jeté un coup d'œil sur le site du Raspberry Pi, et cela ne m'a pas paru facile d'accès. Vous ciblez les enfants et l'enseignement, mais j'ai eu un mal fou à trouver ce qu'était le Raspberry Pi sur le site ; ce n'est pas vraiment expliqué. Il faut le savoir avant. Il y a plusieurs distributions, il faut donc connaître Linux, et on doit le programmer en Python…*
P: Hmm, c'est vrai. Pourtant, ce qui a étrangement contribué au succès, c'est l'approche active. Tu sais, pour faire quelque chose avec le Pi, il ne suffit pas de le sortir d'une boîte et d'appuyer sur *On*. Il faut aussi se creuser les méninges, chercher par soi-même. En fait, je pense que le bénéfice est là : ceux qui réussissent ont le sentiment d'avoir accompli quelque chose : « j'ai fait quelque chose » et non pas « nous avons fait quelque chose ». Quand tu fais quelque chose par toi-même, c'est gratifiant.

C: *Il y a pas mal de couches complexes à comprendre et il faut le programmer en anglais [Python est en anglais], c'est un obstacle de plus pour les gens qui ne sont pas nés avec la langue de Shakespeare. Ce qui a fait le succès d'Arduino, c'est qu'ils ont rendu la programmation très facile, sur un matériel très bon marché.*
P: Arduino est un produit génial, je n'en doute pas. Tu as raison, c'est très facile d'arriver jusqu'au *Hello World*. Mais, regarde bien, une fois que tu as branché le *Raspberry Pi* et mis une carte mémoire, tu atteins l'équivalent d'un *Hello World ;* le nôtre, *Scratch cat* [1], est juste différent. Une fois que tu as joué un peu avec *Scratch cat* (**fig. 1**), tu n'as que l'embarras du choix : continuer à jouer avec, y adjoindre une interface d'E/S pour commander une LED, ou commander *Scratch cat* avec un bouton. Les possibilités sont infinies. Mon expérience, et je pense qu'Eben [Upton] ne me contredira pas, c'est que les gamins comprennent tout de suite. Ce sont les adultes qui ont plus de mal.

C: *J'ai vu au moins trois distributions différentes. Qu'est-ce qui les différencie ? Pourquoi n'y en a-t-il pas qu'une ?*
P: En fait, les différences sont minimes. *Raspberry Pi* devait être un outil pour les étudiants de premier cycle. Tu donnes ça à la fac de Cambridge, avec un peu de chance celle de Manchester aussi, et les étudiants se lancent dans la pratique avant même d'apprendre les concepts en cours. Ils ont tout l'été pour bricoler, et revenir à la rentrée en disant « *Eh, t'as vu ce que j'ai fait !* ». C'était l'idée de départ.

C: *Vous étiez partis sur un niveau plutôt élevé… un étudiant de premier cycle, ce n'est plus tout à fait un enfant.*
P: Oui, le niveau était plutôt élevé et Scratch n'était pas à l'ordre du jour. Nous ne pensions qu'à Python – c'est d'ailleurs de là que vient le Pi du nom. Nous avons développé la communauté et l'écosystème autour du Pi afin que les différents publics désireux d'utiliser le Pi y trouvent leur compte. Aujourd'hui, tu peux aussi utiliser RISC OS. Il y a même des gens qui programment le Pi sans OS. Si nous n'avions fourni qu'une seule distribution, nous aurions fermé les perspectives. J'approuve complètement la multiplicité des distributions.

C: *Bien, mais une fois que tu as choisi une distribution, comment tu fais pour commander une LED, il faut un pilote ou un truc du genre je suppose ?*
P: Il y a une bibliothèque ; il suffit de faire un appel. Encore une fois, ce n'est pas facile. Il faut d'abord trouver les bibliothèques et les télécharger. C'est ici que des cartes comme *Pi-Face*[2] [une carte d'extension] entrent en jeu. Elle est livrée avec une bibliothèque qui vient se brancher sur *Scratch* ; il y a aussi la *Gertboard*[3] [une autre carte d'extension] livrée avec une bibliothèque de commande et quelques exemples. Par la suite, tu peux remonter jusqu'aux commandes de base de pilotage des GPIO.

C: *Donc la simplicité vient des cartes d'extension ? C'est en tout cas pour cela qu'Elektor a proposé la sienne [4].*
P: Certaines peuvent simplifier les choses en fournissant des boutons et des LED pour que tu n'aies pas à te préoccuper du matériel. Je vois un peu ça comme un oignon : tu peux commencer à la surface et déjà en faire quelque chose ; ensuite tu peux enlever les couches une à une. Plus tu es intéressé, plus tu épluches l'oignon et plus tu as le choix.

C: *Passons au matériel. Vous avez choisi un processeur Broadcom parce qu'Eben bossait chez Broadcom ?*
P: Il bosse toujours chez *Broadcom*. Je ne nie pas que ça a pesé sur la décision. Eben disait : « *Cette puce flambant neuve fait tout ce qu'on veut, pourquoi on ne l'utiliserait pas ?* » Nous avons décidé d'engager notre réputation sur le site en disant que ça coûterait 35 $ (25 $ pour la version de base). Après, plus question de revenir là-dessus ! À moins qu'on ne s'en sorte vraiment pas… les emmerdes, ça arrive. Sur le tableur, en gros, les chiffres semblaient plausibles, il ne nous restait qu'à réduire les coûts, affiner le tout, afin de respecter notre engagement. Je pense que si nous avions fait un autre choix, *Samsung* par exemple, le budget aurait explosé.

C: *Est-ce que Broadcom a aidé à rendre le projet possible ?*
P: Tous les fabricants de semi-conducteurs facilitent l'accès à leurs puces ; leur prix est aussi important. Je pense que ceux qui nous ont vraiment aidés sont ceux qui ont parié sur nous et nous ont fait un bon prix dès le début. Le problème, quand on lance un projet, c'est l'inconnue des volumes, il faut être prudent. Au début, on a donc calculé le prix pour mille cartes, puis rapidement pour vingt mille, mais, même dans nos rêves les plus fous, nous n'espérions un carnet de commandes de cent mille cartes le jour du lancement ni être si proche du million un an après. Ça nous a beaucoup aidés dans la mesure où nous avons évidemment pu obtenir un meilleur prix sur les composants. Je ne cacherai pas que cela a fait plaisir aux vendeurs de composants qui ont cru en nous dès le début, ils ont évidemment gagné de l'argent.
Nous nous sommes toujours dit qu'il fallait que notre modèle tienne la route et que la fondation, la communauté qui achète les cartes et les fournisseurs fassent leur beurre et puissent survivre. Ça aurait été un échec total si, par exemple, *Broadcom* nous avait dit : « *Bon, les gars, on vous donne les 20.000 premiers processeurs.* » On aurait cédé à la tentation de rajouter des fioritures et après avoir vendu ces 20.000 premières cartes, le prix aurait fait un bond de 12 $. Ça aurait été la fin du *Raspberry Pi*.

Figure 1.
Le chat de Scratch, un environnement de programmation multimédia pour les enfants.

C: *Et si Eben et les autres n'avaient pas bossé chez Broadcom...*
P: Est-ce qu'on aurait utilisé la même puce ? Je me suis posé la question, j'ai regardé un peu à droite à gauche et n'ai rien trouvé qui corresponde aussi bien à nos besoins que cette puce. J'étais plutôt confiant et me disais que c'était la bonne, et je pense qu'un des éléments clés a été la présence du HDMI. Du point de vue

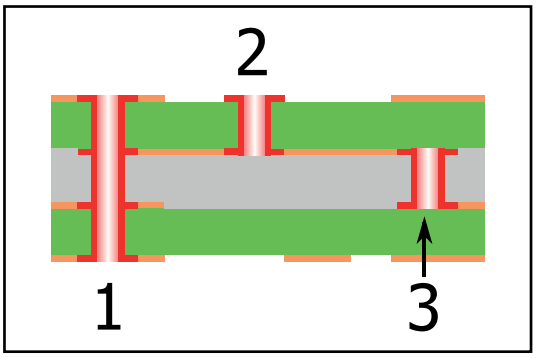


Figure 2. Trois vias différents : 1 – via standard; 2 – via aveugle ; 3 – via enterré. Imaginez une carte à six couches avec des vias aveugles qui relient les couches 1 et 2. Ces deux couches doivent être fabriquées sous forme d'une carte double face à trous métallisés séparée avant de pouvoir être assemblées avec les quatre autres couches. L'assemblage est ensuite percé et métallisé à nouveau. Au total, il faut deux étapes de perçage et métallisation alors qu'il n'en faut qu'une pour une carte sans vias aveugles ou enterrés.

technique, le vrai défi, ça a été de router tous les signaux sous le BGA, les vias aveugles et enterrés sur le PCB étaient très chers (**fig. 2**).

C: *Combien de couches y a-t-il sur la carte ?*
P: Six, rien de bien exceptionnel. La seule astuce était de n'avoir de vias aveugles que sur les couches un et deux – ce qui implique une étape de perçage supplémentaire – mais une seule étape d'assemblage des couches. Cela a augmenté le coût de la carte de deux cents, mais, comme la couche d'en dessous était un plan de masse, cela voulait dire que beaucoup des signaux venant du processeur *Broadcom* n'avaient qu'une seule couche à traverser. J'ai donc eu la place sous la puce pour router d'autres signaux et faire en sorte que cela soit possible.

C: *Broadcom ne fournit pas un guide pour ça ?*
P: Oh, si : « *utilisez des vias aveugles et enterrés ou des vias dans les pastilles* ». Notre premier prototype fonctionnait très bien, mais sa fabrication aurait coûté entre 100 $ et 110 $. On a sorti la machette et commencé à couper tout ce dont nous n'avions pas besoin. Le but : garder toutes les fonctions, la compatibilité, les performances, virer les extras.

C: *Combien d'itérations vous a-t-il fallu pour arriver à la première carte publique ?*
P: Nous avons construit des prototypes... qui ne fonctionnaient pas. Il fallait faire une modif pour que la puce *Broadcom* soit alimentée. Je pense que j'avais mal lu la feuille de caractéristiques.

C: *Comment ça ? Le problème c'est que la puce n'était pas alimentée ?*
P: Oui, la puce n'était pas alimentée, plutôt gênant comme problème ! En fait, il n'y avait que la moitié de la puce qui était alimentée. Certaines alims sont produites en interne, d'autres doivent être fournies et je m'étais un peu emmêlé les pinceaux. Je pense que nous avions une bonne étoile parce que tout ce qu'il fallait faire c'était de relier un via à un plan de cuivre situé juste à côté.

C: *Juste un pâté de soudure.*
P: *Le* pâté de soudure. Un coup de fer et les cartes prenaient vie ! Une des premières vidéos sur le site montre Eben et moi en train de gratter le vernis et de faire la soudure salvatrice.

C: *Est-ce que tu as personnellement appris des choses intéressantes avec ce projet ?*
P: J'ai appris beaucoup de choses. Je pense ce que j'ai appris de plus important, ou tout du moins ce que ce projet a confirmé, c'est que l'on n'apprend pas qu'à l'école. On peut apprendre partout. Les possibilités d'apprendre ou enseigner quelque chose sont tout autour de nous, tout le temps, dans tout ce que nous faisons. Tu sais, il y a des gens qui passent leur vie à garder leurs petits secrets et d'autres qui partagent. J'ai rencontré beaucoup de gens qui partagent, le plus simplement du monde. J'ai appris qu'il y a une multitude de moyens d'apprendre, en dehors des sentiers battus, et que ça aide les gens à accomplir ce qu'ils souhaitent.

Un coup de fer et les cartes prenaient vie !

(130101 – version française : Kévin PETIT)

Liens

[1] Scratch : http://scratch.mit.edu/

[2] Pi-Face : http://pi.cs.man.ac.uk/interface.htm

[3] Gertboard : http://elinux.org/RPi_Gertboard

[4] Carte de prototypage pour Raspberry Pi, Elektor n°417 mars 2013 www.elektor.fr/120483

nouvelle édition revue et augmentée du livre

l'électronique pour les débutants

deux kits d'initiation disponibles

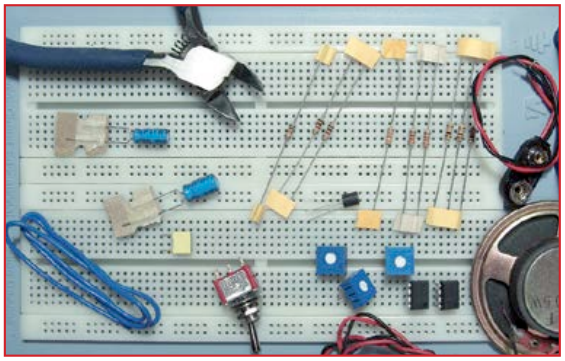
Le cadeau idéal pour partager votre passion de l'électronique avec vos enfants, petits-enfants, neveux… et autres *geeks*.

Fin pédagogue, Rémy Mallard écrit pour les débutants dans un style inédit, et répond d'abord aux questions prosaïques du néophyte : quel fer à souder acheter ? Un multimètre à 5 € peut-il suffire ? Et bien d'autres interrogations que trop de livres laissent en suspens.

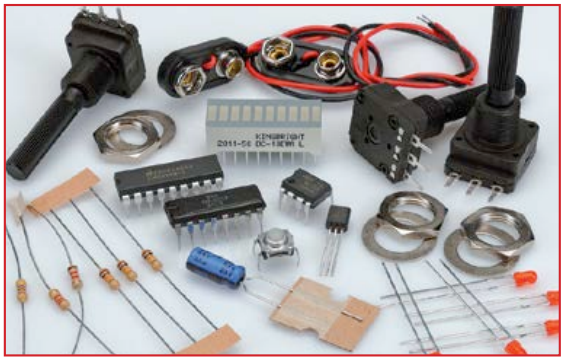
L'auteur démystifie l'électronique en n'utilisant que ce qu'il vous faut de théorie pour aborder la pratique : identifier les composants et leur rôle, les récupérer, les tester et les ranger ; lire un schéma ; choisir ses outils ; mettre en boîte ses montages…

Les deux kits disponibles séparément permettent de réaliser, sur une plaque d'expérimentation sans soudure, quelques-uns des montages simples et ludiques présentés dans le livre.

Kit n°1 : sirène | réf. 119016-71 | **24,50 €**



nouveau ! Kit n°2 : chenillard & thermomètre
réf. : 119016-72 | **24,50 €**



offre spéciale :

livre + deux kits = 81,50 €

au lieu de 91,50 €

ISBN 978-2-86661-186-6
édition revue et augmentée
352 pages - **42,50 €**

elektor

informations complémentaires et commande :
www.elektor.fr/debut

la malédiction du collectionneur

Reginald Neale
(États-Unis d'Amérique)

J'ai utilisé ce **Multimètre Fairchild 7050** à la fin des années soixante, l'un des premiers à mettre en œuvre une intégration à double pente, avec comme résultat un zéro stable par nature. Le choix d'une période d'intégration multiple de la période du secteur réduit spectaculairement le bruit à la fréquence du secteur. Photo : avec l'aimable autorisation de Marvin Collins.

Stylo-dosimètre V-742 protection civile. L'échelle graduée est ce qu'on voit en le tenant face à la lumière et en visant à travers la lentille à l'extrémité. Le V-742 est destiné à être accroché à un vêtement pour mesurer la dose radio-active cumulée. Le dosimètre est chargé sous une tension élevée qui ramène l'indicateur capillaire au zéro. Les radiations occasionnent une fuite de charge qui déplace l'indicateur vers la droite. L'appareil est toujours utilisé par la FEMA *(Federal Emergency Management Agency)* et disponible en plusieurs gammes de doses.

Ce **Convertisseur Brown** est en fait un jeu de contacts vibrants utilisés dans un amplificateur à détection synchrone, sur un enregistreur à bande, de signaux continus au niveau du millivolt.

Récepteur classique radio-amateur. Collins 2051, détenu, restauré et photographié par Marvin Collins, W6OQI.

Heathkit. Le grand nom de l'électronique à construire soi-même. J'ai réalisé cette alimentation de labo pour les tubes à la fin des années cinquante.

D'autres musées virtuels sur le ouèbe

Si vous avez apprécié cette exposition, vous serez peut-être intéressés aussi par certains de ces sites :

www.conradhoffman.com
www.oneillselectronicmuseum.com/page10g.html
www.electricstuff.co.uk/
www.tubecollector.org/list.php?L
www.unusualmuseums.org/
http://wps.com/archives/tube-datasheets/
www.ieee-virtual-museum.org/
www.seas.upenn.edu/~museum/
http://frank.yueksel.org
http://www.heathkit-museum.com/
http://semiconductormuseum.com/Museum_Index.htm
http://silicongenesis.stanford.edu

Des flopées de sites intéressants, et naturellement ça change sans arrêt. Attention : ce sont d'incroyables dévoreurs de temps. Mais vous le saviez déjà...

Compteur numérique Fluke 1941A du début des années soixante-dix, avec affichage à tubes nixie.

Les appareils et les composants reproduits ici ont été accumulés au cours d'une longue carrière dans l'électronique, pendant laquelle je me suis montré tragiquement incapable de jeter quelque chose d'élégant, original, merveilleux, curieux, tordu ou d'une qualité exceptionnelle, indépendamment de toute chance de lui trouver un jour quelque application utile. Les pages *Rétronique* de ce mois-ci sont dédiées à ceux qui souffrent des mêmes tourments.

Le **Galvanomètre haut de forme** Leeds & Northrup est familier pour qui a travaillé dans les laboratoires scientifiques d'antan. Il détecte des courants électriques infimes et utilise un faisceau lumineux en guise d'aiguille, longue et sans masse. Il fait environ 20 cm de haut. Les galvanomètres actuels sont beaucoup plus petits, plus robustes, et des dizaines ou centaines de fois plus rapides. Photo : avec l'aimable autorisation de Conrad R. Hoffman.

Des livres !

Certains livres traitent des technologies décrites dans ces pages :

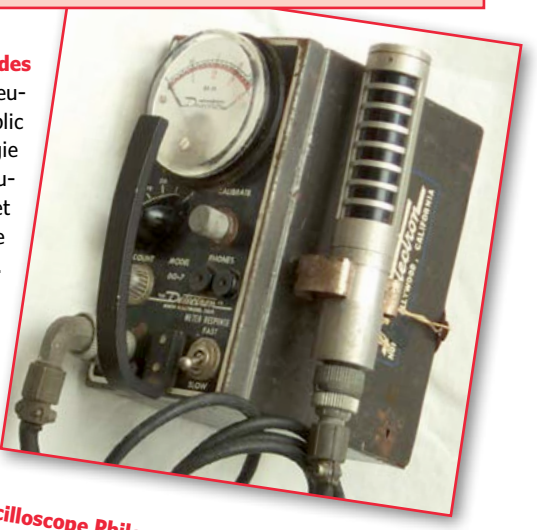
70 Years of Radio Tubes, John W. Stokes, Vestal Press 1982

Procedures in Experimental Physics, J. Strong, Prentice-Hall 1938

The Amateur Scientist, C. L. Stong, Simon & Schuster 1960

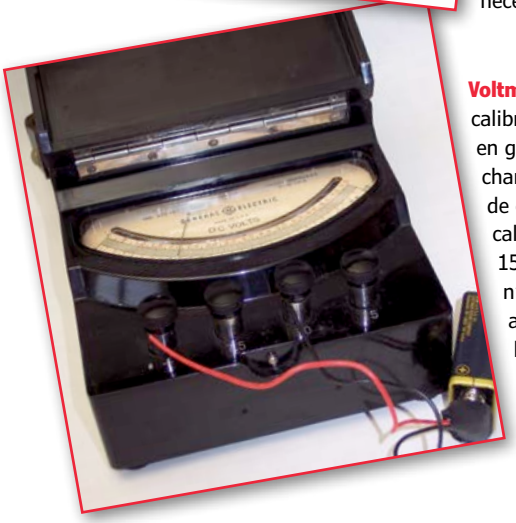
Reference Data for Radio Engineers, Federal Telephone and Radio Corp. 1946

Compteur Geiger de la fin des années quarante. Après la deuxième guerre mondiale, le public était fasciné par l'« énergie atomique ». Quelques particuliers ont acheté ces appareils et sont partis à la prospection de l'uranium.



Oscilloscope Philco utilisé dans les années cinquante pour le dépannage de téléviseurs. Ni base de temps calibrée, ni échelle de tension. Synchronisation du signal très rudimentaire. Le coffret gris-bleu martelé est caractéristique des appareils de l'époque. Photo : avec l'aimable autorisation de Marin Collins.

Grid-dip-mètre des années cinquante. Il permettait une vérification sans contact de la fréquence de résonance d'un circuit. La bobine de test enfichée sur la face supérieure de l'appareil était couplée au circuit à tester. Après avoir tourné le cadran sur le flanc gauche jusqu'à ce que le galvanomètre accuse un creux prononcé, on pouvait lire la fréquence sur ce cadran. Plusieurs bobines étaient nécessaires pour couvrir la plage de 600 kHz à 300 MHz.



Voltmètre de labo General Electric. Volmètre de laboratoire multicalibres, années cinquante. Les multimètres modernes présentent en général une résistance d'entrée fixe de 10 MΩ, avec un effet de charge négligeable pour la plupart des mesures. Ce n'est pas le cas de ces voltmètres électromécaniques. Leur résistance dépend du calibre, dans ce cas 200 Ω par volt. Sur le calibre 7,5 V, cela donne 15 kΩ, une résistance suffisamment basse pour mettre à genoux n'importe quel circuit usuel. Un autre problème des voltmètres analogiques : l'aiguille se meut dans un plan décalé de celui de l'échelle. Remarquez l'arc réfléchissant le long de l'échelle. Si vous ajustez la position de votre œil de façon à voir le reflet de l'aiguille superposé à l'aiguille elle-même, vous réduisez à son minimum l'erreur de parallaxe.

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

Hexadoku casse-tête pour électroniciens

Lâchez tout ! Les manettes, les sondes, la souris, l'écran, le clavier… Fermez votre portable. Débranchez la sonnette. Prenez un crayon bien taillé, une gomme, une boisson fraîche, et installez-vous confortablement… Desserrez votre cravate, quittez vos chaussures, c'est l'heure de votre séance de thérapie ***hexadoku***. Remplissez la grille selon les règles, et envoyez-nous votre solution. Vous gagnerez peut-être l'un des chèques-cadeaux Elektor. Et vous vous serez agréablement détendu.

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 envoyer la série de chiffres sur fond grisé.

Participez et gagnez !

Nous tirons au sort l'une des réponses internationales correctes reçues dans les délais ; son auteur recevra un chèque-cadeau d'une valeur de **100 €** à valoir sur des **circuits imprimés Eurocircuits.**
Nous offrons en outre 3 chèques-cadeaux à valoir sur des livres d'Elektor d'une valeur de **50 €** chacun. À vos crayons !

Où envoyer ?

Envoyez votre réponse (les chiffres sur fond grisé) avec vos coordonnées par courriel, télécopie ou courrier avant le **1er mai 2013** :
Elektor c/o Regus Roissy CDG – Le Dôme – 1, rue de La Haye
BP 12910 – 95731 Roissy CDG
hexadoku@elektor.fr | www.elektor.fr/hexadoku

Les gagnants

La solution de la grille du numéro de mars (417) est : **48C57**
Le gagnant des **100€** à valoir sur des circuits imprimés **Eurocircuits** est **Yves PRINTEMS** (Valbonne, France).
Les **3 chèques-cadeaux Elektor** d'une valeur de **50 €** chacun vont à :
Peter Raue (Dresden, Allemagne) | Arwin J. Vosselman (Biddinghuizen, Pays-Bas) | Torsten Clever (Wipperfuerth, Allemagne).
Bravo à tous et félicitations aux gagnants !

9	4	E			5	6			1	3			2	B	A
B		0		1		E	2	4	9		D		8		6
	1		3	0		C	8	6	A		B	4		F	
C			8	3							7	E			1
	9			2							0			D	
	7	3		9							E		B	6	
	2			4	B	5			6	9	A			7	
	5	B	C									1	E	4	
					0		4	8		2					
		5			2	3			E	7			F		
E		A	4	7	C					B	6	5	3		8
		C				B			5				0		
A		1			6					F			4		D
F	E					2			7					1	3
		D	9	E							C	A	7		
4				D	3		A	0		1	5				C

6	F	2	8	C	3	D	E	9	0	1	A	7	4	5	B
0	7	9	C	F	5	A	1	B	6	4	2	D	E	8	3
1	3	D	E	B	2	4	8	C	5	7	F	0	6	9	A
4	A	B	5	6	7	9	0	8	3	D	E	C	F	1	2
A	8	C	9	0	B	1	3	E	F	2	5	4	7	D	6
7	D	3	2	4	F	C	5	A	8	0	6	1	9	B	E
B	0	F	1	7	E	8	6	D	9	3	4	A	2	C	5
E	4	5	6	9	A	2	D	1	7	B	C	F	0	3	8
2	E	0	4	8	C	3	7	F	A	9	D	B	5	6	1
3	5	6	F	1	D	B	4	7	2	C	0	8	A	E	9
8	B	7	D	2	9	5	A	4	E	6	1	3	C	F	0
9	C	1	A	E	6	0	F	3	B	5	8	2	D	7	4
C	1	A	3	D	8	6	9	0	4	E	7	5	B	2	F
D	2	8	7	A	0	E	B	5	1	F	9	6	3	4	C
5	6	E	0	3	4	F	C	2	D	8	B	9	1	A	7
F	9	4	B	5	1	7	2	6	C	A	3	E	8	0	D

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.

elektor PCB Service
powered by Eurocircuits

PCBs
Multilayer
Prototypes
Small series

Professional Quality
Trusted Service
Secure Ordering

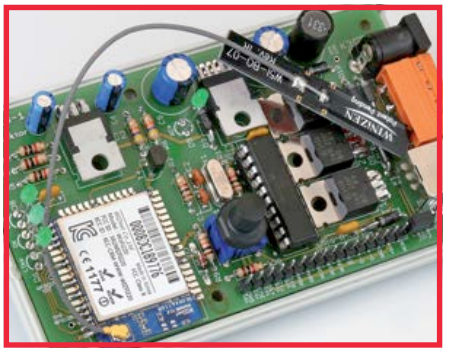
Elektor PCB Service at a glance:

- 4 Targeted pooling services and 1 non-pooling service
- Free online PCB data verification service
- Online price calculator available
- No minimum order value
- No film charges or start-up charges

Delivery from 2 working days

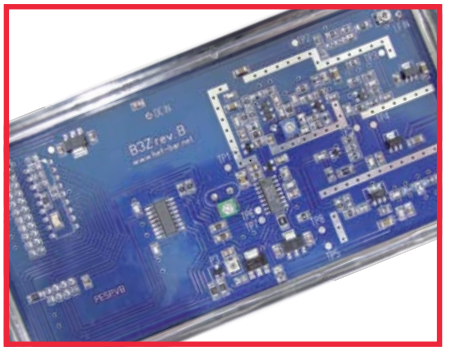
Further information and ordering at www.elektorpcbservice.com

bientôt dans Elektor



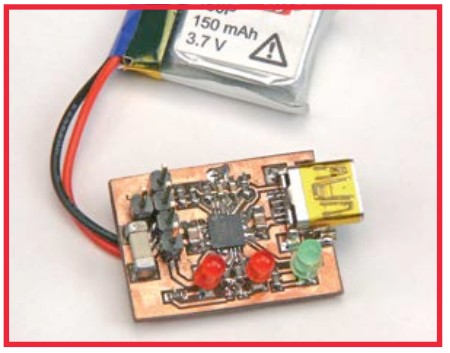
Carte de commande Wi-Fi universelle

Ce circuit de commande de LED commande une bande de LED RGB au moyen d'un réseau Wi-Fi. Intéressant, surtout avec les fonctions supplémentaires : chargeur d'amorçage, port d'extension et diodes de protection, qui permettront de l'adapter à la commande par Wi-Fi d'une porte de garage, d'un relais, d'un robot, d'un modèle réduit, etc.



Transmetteur FM 70 cm 130 mW à large bande

Cet émetteur délivre à la fréquence choisie par l'utilisateur dans la plage de 430-440 MHz de la bande des 70 cm, et sans baver, un signal FM à large bande de 130 mW. Si les dispositions de votre pays le permettent, ce TX permet d'établir des liaisons audio hertziennes de qualité, même en mode duplex.



Chargeur de batterie au lithium-ion

Réutiliser correctement des batteries au lithium-ion de récup' n'est pas facile quand ces batteries étaient rechargées dans l'appareil qu'elles alimentaient, car, dans ce cas, il n'y a pas de chargeur séparé. Il n'est pas bien difficile de construire soi-même un chargeur pour de telles cellules Li-ion avec un circuit intégré de Maxim.

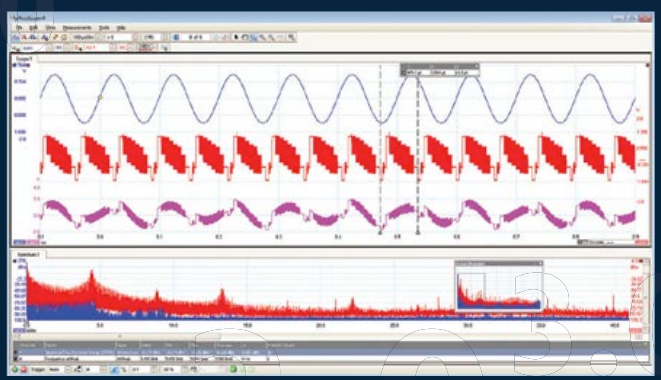
Informations préliminaires non contractuelles.
Le numéro de juin paraîtra le 22 mai.

pico® Technology

PREMIÈRE MONDIALE

USB 3.0 OSCILLOSCOPE

PicoScope	3207A	3207B
Bande passante	250 MHz	250 MHz
Taux d'échantillonnage	1 GS/s	1 GS/s
Mémoire	256 MS	512 MS
Générateur de signal	Générateur de fonctions	AWG
Prix	€1813	€1451
Alimentation	port USB	
Compatibilité	USB 2.0 & 3.0	



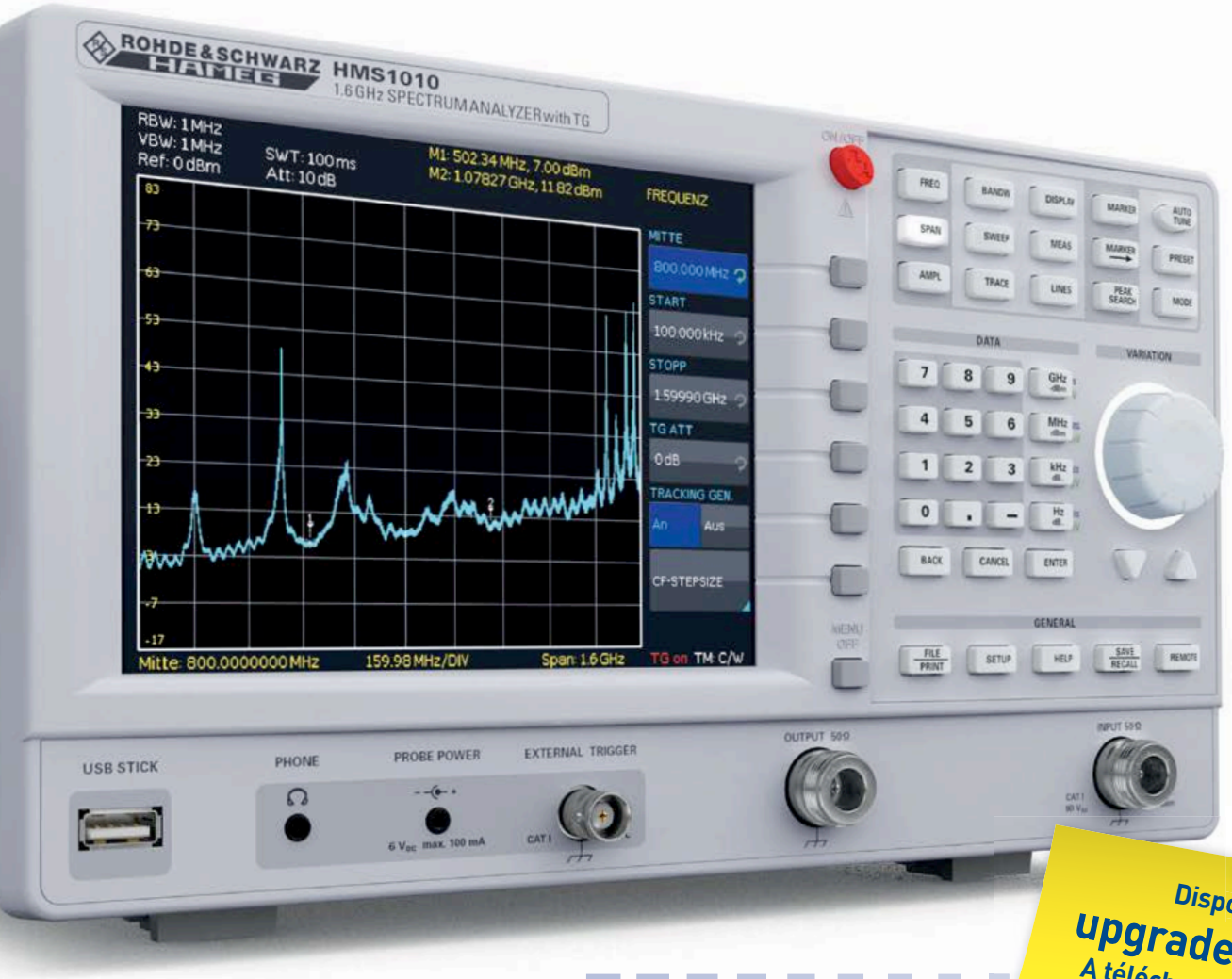
TOUS LES MODÈLES INCLUENT SONDES, LOGICIEL COMPLET, AINSI QUE 5 ANS DE GARANTIE. LE LOGICIEL COMPREND DES: MESURES, ANALYSEUR DE SPECTRE, FULL SDK, DÉCLENCHEMENTS AVANCÉS, PERSISTANCE DE COULEURS, DÉCODAGE SÉRIE (CAN, LIN, RS232, I²C, I²S, SPI), MASQUES, MATHS, LE TOUT EN STANDARD. MISES À JOUR GRATUITES.

www.usb3scope.com/PS211

ANALYSEURS DE SPECTRE

www.hameg.com | Great Value in Test & Measurement

HAMEG®
Instruments
A Rohde&Schwarz Company



NOUVEAU !

Disponible par
upgrade firmware
A télécharger gratuitement
sur www.hameg.fr

HAMEG INSTRUMENTS augmente la fréquence de sa gamme d'analyseur de spectre série 1000 de 1 GHz à 1,6 GHz.

Avec l'achat d'un analyseur de spectre HMS1000, HMS1010 ou HMS1000E, les clients HAMEG feront l'expérience d'une augmentation immédiate de 60% de fréquence en plus, sans coût additionnel.

+60%

DE FRÉQUENCE EN PLUS

HMS1010 | HMS1000 | HMS1000E

de 1 GHz à 1,6 GHz

ROHDE&SCHWARZ

Rohde & Schwarz France | 9/11 rue Jeanne Braconnier
92366 Meudon-la Forêt | Tél : 01.41.36.10.00 | Fax : 01.41.36.11.11
www.rohde-schwarz.fr | contact.rsf@rohde-schwarz.com