janvier / février 2017 | nº 463 / 464
www.elektormagazine.fr

elektor

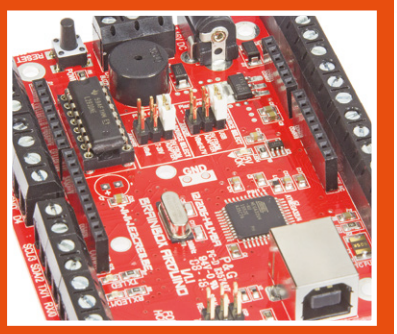
DÉCOUVRIR CRÉER PARTAGER

horloge de sable Arduino

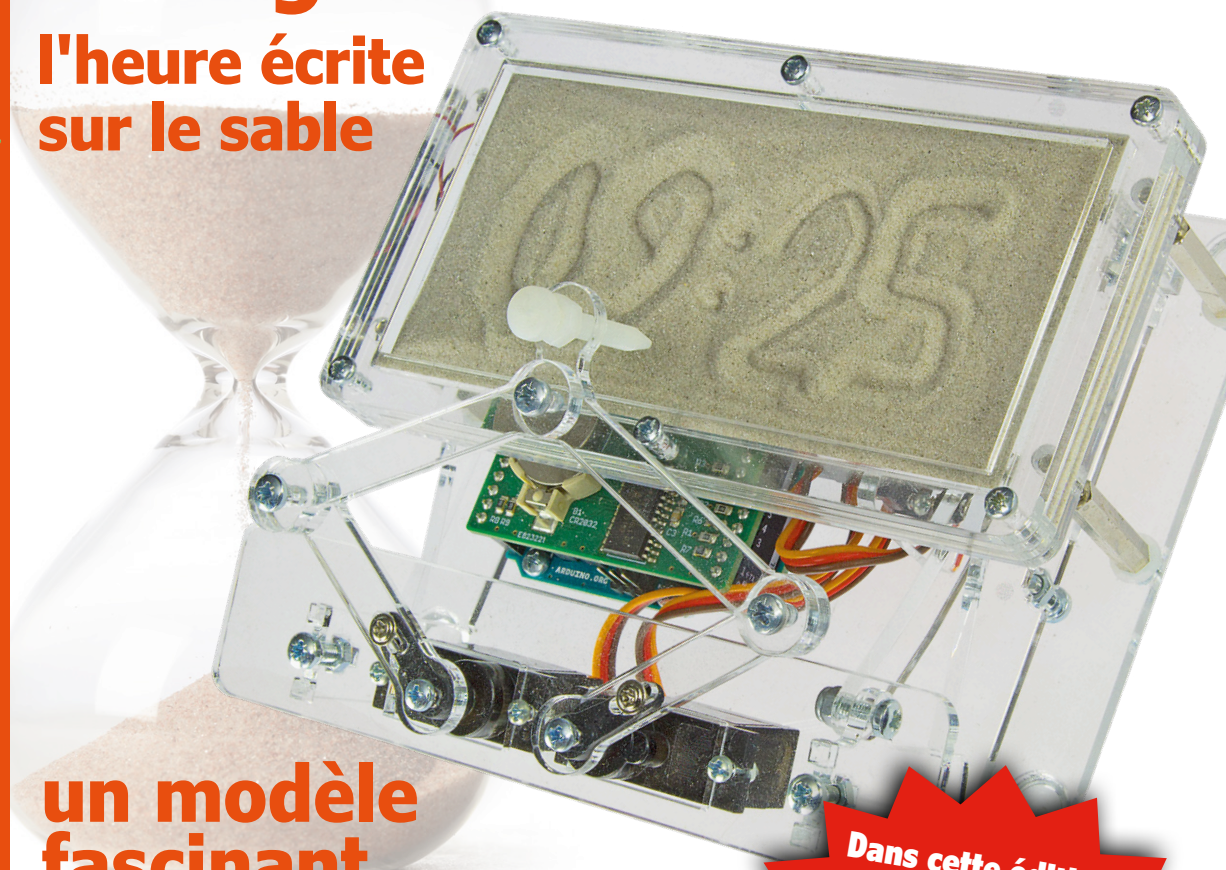
l'heure écrite sur le sable

régulateur de température
de tête d'imprimante 3D

pisteur de tension
mesures à long terme avec *shield* Arduino



BrainBox Arduino
matériel « costaud » avec bornes à vis



un modèle fascinant

Dans cette édition :
12 projets du labo
3 projets de lecteur
Arduino, CAO, DAB+, LED, LoRa, RPi, SDR, ...

sonomètre Arduino tricolore • LED en couleur • **connecter des objets avec Genuino 101** • simuler avec SystemVision® • **nouvelles lames pour le Swiss Pi** • Rétronique - du Verobox au Heavy Metal • **débogage sur Arduino Zero & M0 Pro** • circuits imprimés faits maison • ***shield* IdO pour Arduino** • moteur Mendocino • ***shield* d'affichage MAXREFDES99#** • R. Lacoste : LoRa échange débit contre portée • **diplexeur d'antenne** • gagnants de *l'electronica Fast Forward Award*

ISSN 0181-7450

(B) 7,90 € • (CH) 13,20 FS • (CAN) 9,75 CAD • DOM surface 8,20 e • DOM avion 10,20 €
TOM surface 940 XPF • TOM avion 1620 XPF • Réu/A 9,70 € • Cal/S 950 CFP • Cal/A 1560 CFP • And 7,20 €

L 19624 -463- F: 15,50 € - RD

enova STRASBOURG

LE SALON
DES **TECHNOLOGIES**
POUR LES **INNOVATIONS**
DE DEMAIN

ÉLECTRONIQUE / EMBARQUÉ / IOT / MESURE / VISION / OPTIQUE / BIG DATA

15 & 16 MARS 2017
Parc des expositions de Strasbourg

DONNEZ VIE À VOS PROJETS

AÉRONAUTIQUE | MILITAIRE | AGROALIMENTAIRE | AGRICOLE | AUTOMOBILE | TRANSPORT
SMART CITIES | SMART BUILDING | MÉDICAL | INDUSTRIE 4.0 | RECHERCHE ACADÉMIQUE

Made by GL events

www.enova-event.com

40ème année,
n° 463/464 – janvier/février 2017

ISSN 0181-7450
Dépôt légal : décembre 2016
CPPAP 1113 U 83713
Directeur de la publication : Donatus Akkermans

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

@ : service@elektor.fr

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

Fax : (+33) 01.49.19.22.37

www.elektor.fr | www.elektormagazine.fr

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

Publicité :

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

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

Cap sur le changement et l'innovation !

En 2017, le paquebot Elektor revoit sa jauge et son carénage, change d'allure et naviguera sur de nouvelles routes.
Dès janvier, le magazine prend du coffre et devient bimestriel. Désormais chaque numéro comptera au moins 132 pages. Puis, avec le numéro de mars-avril, nous passerons au format A4. Nous réorganiserons le contenu d'Elektor en rubriques moins nombreuses et plus étoffées, recentrées sur l'essentiel : vos besoins dans votre pratique de l'électronique.
Le contenu technique sera enrichi et développé. Dans la salle des machines, l'équipage s'active pour répondre à votre attente. Les filles et les gars du labo créent leurs propres montages, sélectionnent les projets que vous partagez avec la communauté d'Elektor et les accompagnent jusqu'à leur publication.
D'ailleurs, sur notre site www.elektormagazine.fr, la popularité de l'onglet Elektor Labs progresse de jour en jour, au fil des propositions, des commentaires et des questions que vous y publiez. Et des réponses, entre autres aux questions techniques sur les circuits, que vous y trouvez. Elektor Labs attire de plus en plus de visiteurs, car c'est là que sont mis à disposition les compléments aux articles : listes de composants, logiciels, schémas, fichiers Gerber, tous tenus à jour.
L'offre diversifiée et modernisée, concoctée par la rédaction, se cristallisera autour du magazine bimestriel. Sur le site, vous trouverez nouvelles, bancs d'essai, articles de fond, etc. sous diverses formes : PDF et pages internet sur elektormagazine.fr, vidéos sur elektor.tv...
En parallèle, nos collègues de la boutique en ligne d'Elektor s'affairent à remplir les rayons sur www.elektor.fr. Suggérez-nous vos idées de produits nouveaux et utiles à tout électronicien, que l'e-choppe d'Elektor devrait proposer.
L'internet a refaçonné le monde, il aurait été étonnant qu'il ne refaçonne pas Elektor. Le changement est un signe de vitalité.
Nous voici parés pour de nouveaux périples.

Bonne et heureuse année 2017 !

Mariline Thiebaut-Brodier

Notre équipe

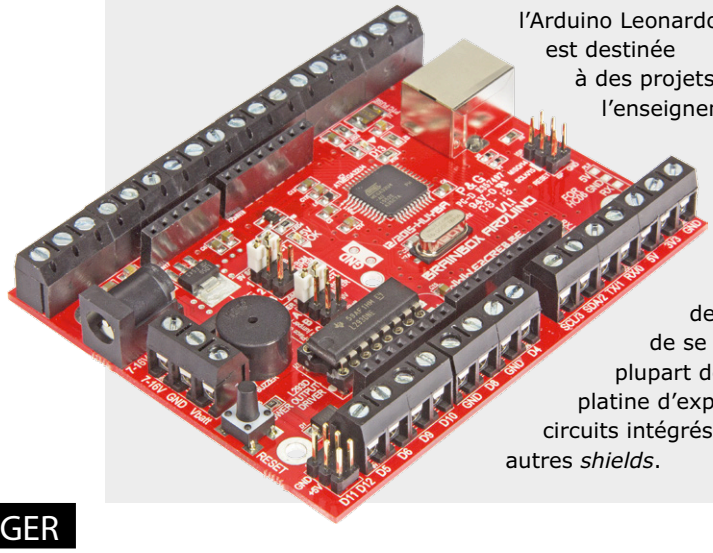
ce mois-ci

40e année – nº 463/464 janvier/février 2017

BrainBox Arduino

un Arduino « costaud » avec bornes à vis

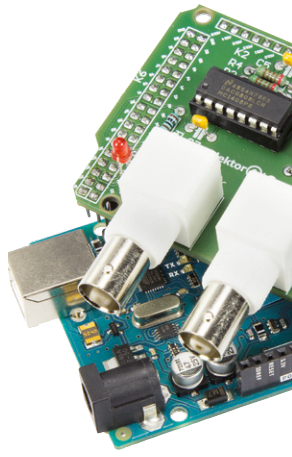
72



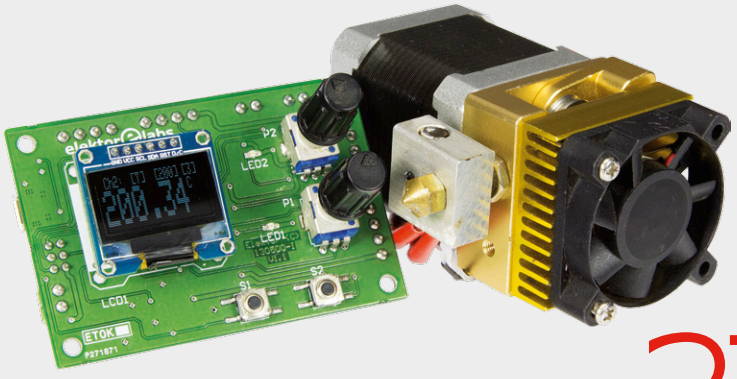
Cette version robuste de l'Arduino Leonardo est destinée à des projets autonomes et à l'enseignement. Les solides bornes à vis, les diverses options d'alimentation, le buzzer intégré et le pilote pour la commande directe de moteurs permettent de se passer, pour la plupart des applications, de platine d'expérimentation, de circuits intégrés additionnels et autres *shields*.

DÉCOUVRIR | CRÉER | PARTAGER

DÉCOUVRIR | CRÉER | PARTAGER

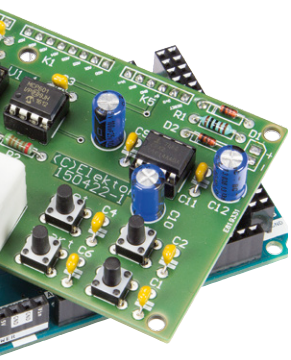


horloge de sable
un modèle fascinant
64
l'heure écrite sur le sable



régulateur de température
37
de tête d'imprimante 3D

Pour un bon résultat, chaque type de filament d'impression 3D (ABS, PLA, etc.) doit travailler à sa température d'extrusion optimale. Pour un même matériau, cette température peut dépendre de la couleur du filament. Il faut donc une régulation précise de la température de la tête d'extrusion. Et pourquoi ne pas contrôler aussi la température du lit ?



elektor magazine

DÉCOUVRIR CRÉER PARTAGER

bientôt sur ces pages

Extrait du sommaire du prochain numéro :

- Carte AVR playground
- Carte Linux Elektor
- Commande de moteur CC
- Elektor R4 & MLI
- Extensions matérielles pour Swiss Pi
- Horloge RVB
- Mesure de la charge d'accus et de batteries
- OBD avec Raspberry Pi
- Passerelle IdO

Etc.

Attention : le prochain numéro sera double (mars-avril) et aura un nouveau format.

La date limite pour la participation à l'hexadoku (chatdoku ce mois-ci) a été avancée au 1er février 2017. Ne tardez pas à jouer !

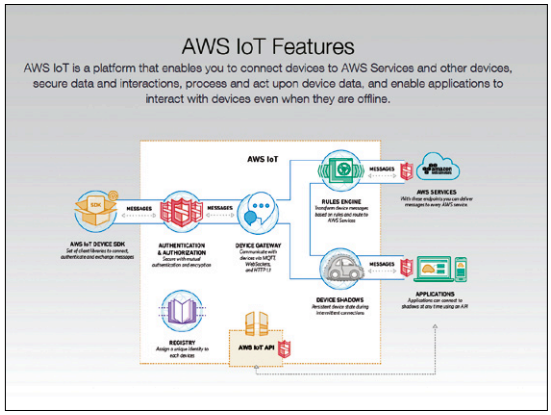
Sous réserve de modifications.

Le numéro de mars-avril paraîtra le 22 février 2017.

bienvenue dans la section DÉCOUVRIR

En route, les voyageurs !

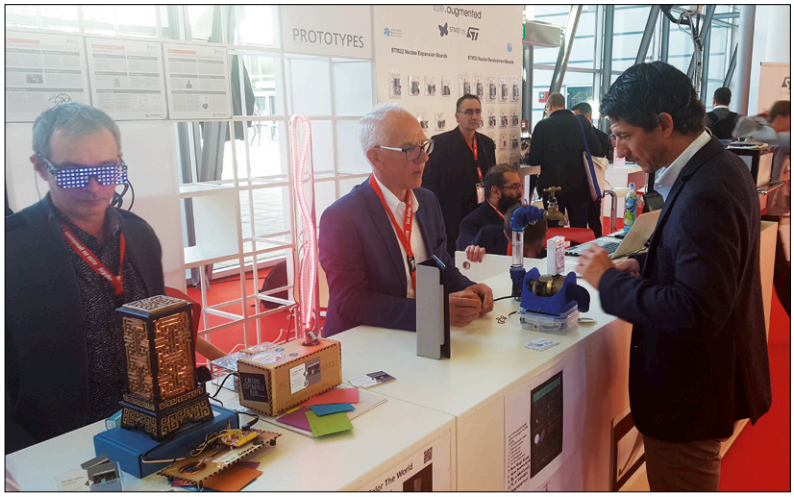
De Cologne à Munich (où avait lieu le salon *electronica*), il y a plusieurs heures de train. Heureusement, j'ai pu rester en ligne grâce au réseau WLAN de la compagnie ferroviaire ou via mon ordiphone. De toute façon, je n'ai pas eu le temps de m'ennuyer puisque j'avais emporté une littérature électronique abondante. Les nombreux articles sur l'IdO et en particulier sur sa sécurité annonçaient la couleur du salon munichois : ces sujets étaient dans toutes les bouches. Les cartes et kits des différents fabricants, présentés sur les stands, sont prêts à travailler avec l'un des plus grands fournisseurs de services dans le nuage (*cloud*) : *Amazon Web Services* (AWS) proposés (évidemment) par Amazon, l'un des géants du numérique. Après connexion (les services de base sont gratuits, mais limités), on peut télécharger les mesures de son propre capteur dans le nuage, par ex. via MQTT. Les cartes sont représentées dans le nuage par des appareils virtuels qui contiennent les données collectées. Même lorsque la carte de mesure est hors ligne, les mesures effectuées restent accessibles pour les ordiphones et autres terminaux.



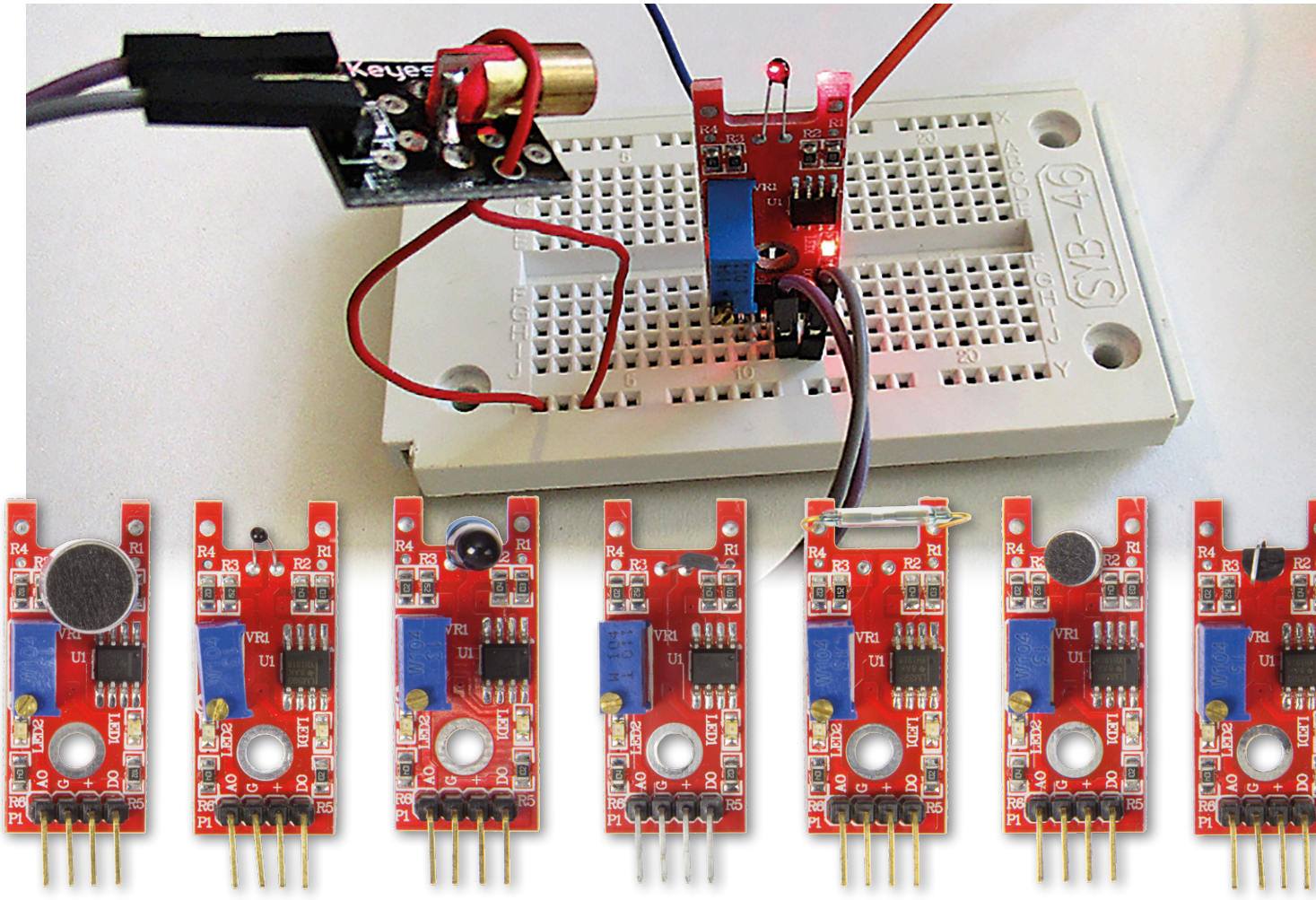
Il faut impérativement que j'examine cela de plus près, je partagerai le résultat de mes investigations sur mon blog : « Mon voyage dans le nuage ».
www.elektormagazine.fr/tags/mon-voyage-dans-le-nuage

Trop de choses à voir

Sur le stand d'Elektor, j'ai été très occupé. Bien entendu, je voulais rencontrer personnellement tous les participants au concours *Fast Forward Start-up Award* et examiner leurs prototypes (j'y suis presque arrivé, mais il y avait beaucoup, beaucoup… de projets intéressants).
Du coup, je n'ai pas eu le temps de rendre visite aux principaux fondeurs de circuits. Heureusement, notre auteur indépendant et expert en microcontrôleurs, Viacheslav Gromov, s'en est chargé, il vous présente ses trouvailles dans cette section.
Il y a quand même un stand que je voulais absolument voir, c'est celui de Microchip/Atmel. J'y ai assisté à une démonstration sur le débogueur « Power » ainsi que sur un kit (vous vous en doutez) spécifiquement conçu pour l'internet des objets (www.atmel.com/tools/at88ckecc-aws-xstk.aspx). Il embarque une puce de chiffrement d'Atmel, préconfigurée pour une authentification par (tiens donc !) les *Amazon Web Services*. Je dois admettre que je n'ai pas tout compris des explications détaillées, fournies par le spécialiste d'Atmel, mais ce n'est pas une faute que de le reconnaître. De toute façon, j'avais déjà au programme de me pencher sur les certificats (*root*), le cryptage ECC et les processus de connexion et d'authentification d'AWS… ⏮



(160244 – version française : Yves Georges)



capteurs (2)

pour Arduino et Cie

Les capteurs sont soit analogiques soit numériques. La lecture de valeurs de mesure analogiques requiert une entrée A/N tandis que pour celle des signaux numériques, il suffit d'un simple port. Cependant, certains capteurs ont deux sorties : une analogique et une numérique.

Burkhard Kainka

En regardant de plus près notre jeu de capteurs (disponible chez Elektor [1]), on s'aperçoit que sept d'entre eux reposent sur la même carte. Cette dernière est dotée d'une sortie analogique AO et d'une sortie numérique DO. Pour convertir un signal analogique en signal numérique, on se sert d'un comparateur. Chaque carte est donc munie d'un comparateur, en l'occurrence un double : le LM393.

Capteurs avec comparateur

Le schéma de la carte (fig. 1) est simple : le capteur est inclus dans un diviseur de tension réglable avec un potentiomètre de précision à 25 tours. Un deuxième diviseur de tension composé de deux résistances de 100 kΩ fournit au comparateur une tension de référence de 2,5 V. Le capteur de température (*Digital Temp*) peut donc être réglé de telle sorte que sa tension soit exactement égale à 2,5 V pour une température donnée. Lorsque la température augmente sur le capteur CTN, le comparateur active la sortie numérique DO, tandis qu'il coupe la tension lorsque la température baisse. Le deuxième comparateur en aval sert uniquement à allumer la LED d'état qui permet de tester les sept capteurs, sans logiciel.

Comme le circuit n'a pas de condensateur de dérivation (voir **encadré**), son comportement risque d'être influencé par les moindres variations ou impulsions parasites de la ligne d'ali-

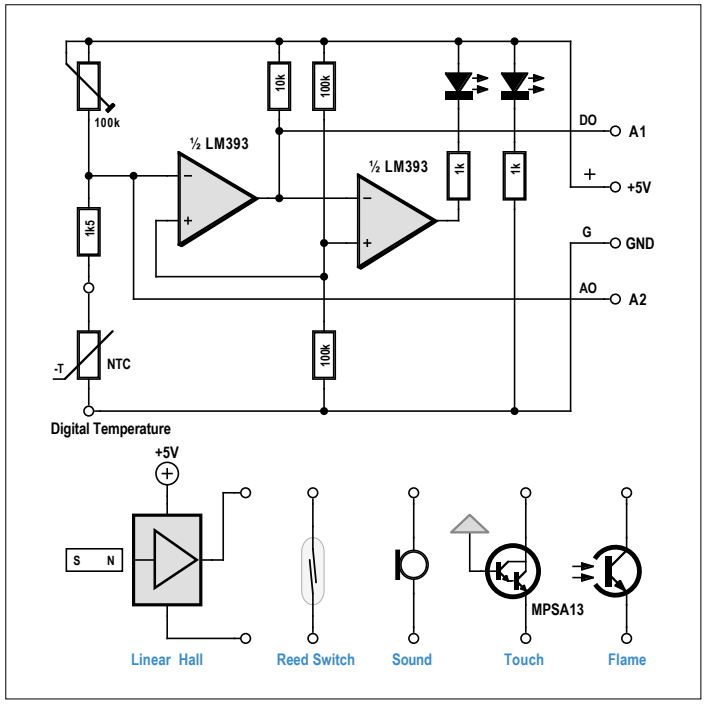


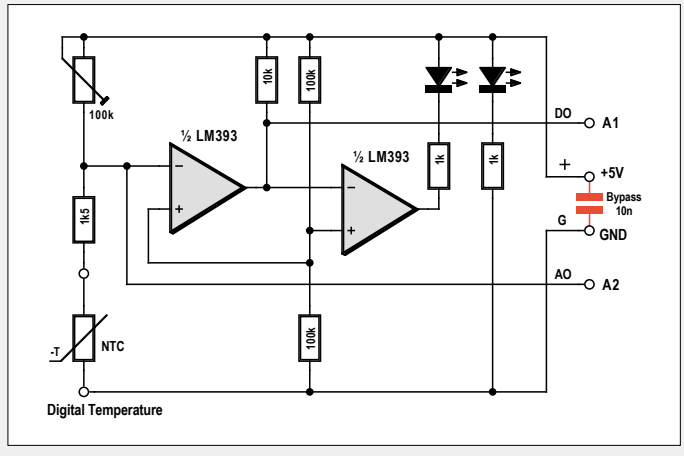
Figure 1. Capteurs avec comparateur.

mentation. Ces perturbations potentielles dépendent aussi de la longueur de la ligne et d'autres contingences. En effet, le capteur de température présente une petite plage dans laquelle la sortie oscille lorsque la température augmente, ce qui peut provenir des capacités parasites de la carte. Si on imagine qu'il y a un petit condensateur entre l'entrée et la sortie du deuxième comparateur, on voit un oscillateur. L'oscilloscope permet de le détecter facilement. La LED d'état de la carte du capteur le montre également. Lorsque la température augmente lentement, la luminosité de la LED est tout d'abord moyenne (oscillations) avant d'être maximale (état stable). Pensez à ce point, si le logiciel fait des caprices.

Les autres capteurs avec le même circuit réagissent pareillement, sauf que généralement les valeurs de mesure ne changent pas aussi lentement. Le phototransistor du détecteur de flamme (*Flame*) est comparable au capteur CTN. Son boîtier de couleur foncée laisse passer les longueurs d'onde les plus longues, ce qui permet de détecter les flammes. Le capteur à effet Hall (*Linear Hall*) se distingue par le fait que sa troisième broche nécessite une tension d'alimentation. Ce qui peut surprendre c'est que l'interrupteur *Reed* (à lames souples) est connecté comme un capteur analogique. Cela permet néanmoins d'avoir deux sorties en opposition de phase. À l'approche d'un aimant, une sortie est activée tandis que l'autre est désactivée.
Le capteur de son (*Sound*) et le capteur tactile (*Touch*) se comportent différemment, ils fournissent en principe un signal rectangulaire avec des changements rapides. Les deux capteurs de son utilisent un grand microphone à électret et un petit. Si le potentiomètre est correctement ajusté, les demi-ondes des signaux acoustiques forts apparaissent sous forme de rectangles en sortie. Il faut en tenir compte lors de l'évaluation.

Condensateurs de dérivation

Beaucoup de montages sont dotés de condensateurs entre la tension d'alimentation (V_{CC}) et la masse (GND). Ils permettent d'améliorer la stabilité d'un circuit et de parer aux interférences radio.
Lorsqu'un circuit électronique est relié à sa source d'alimentation via un long câble, non seulement la résistance du fil interne, mais aussi l'inductance du câble ont un impact sur le circuit. En fonction de l'épaisseur du câble et de l'écart entre les conducteurs, un câble double d'une longueur de 1 m peut créer une inductance d'environ 0,5 µH. Il en résulte une résistance inductive de 3 Ω à 1 MHz ou de 30 Ω à 10 MHz. Si le circuit est soumis à des variations d'intensité, c'est comme si on avait un courant alternatif sur la ligne d'alimentation et donc des chutes de tension. Les problèmes typiques de ce type de chute de tension sont une sensibilité aux interférences radio et une vulnérabilité des éléments passifs du circuit. De plus, le long câble peut se transformer en antenne qui rayonne des signaux HF qui peuvent dépasser les valeurs limites autorisées. Inversement, les impulsions parasites peuvent entraîner de brèves variations de la tension de service, ce qui peut entraver le bon fonctionnement d'un circuit.
Les moteurs à courant continu soumettent eux aussi leur source d'alimentation à des charges alternées ce qui peut provoquer des interférences radio. C'est la raison pour laquelle, on met un condensateur directement à leurs bornes, c'est le condensateur de déparasitage. Les moteurs à courant continu des modèles réduits télécommandés en sont toujours pourvus pour éviter de



perturber leur propre émetteur. Les microcontrôleurs comme l'Arduino sont eux aussi dotés d'un condensateur entre GND et VCC, sinon ils ne pourraient pas passer les tests de compatibilité électromagnétique (CEM). Dans ce cas, on parle de condensateur de dérivation, parce que le courant HF est dérivé au travers de ce condensateur, généralement de 100 nF. Autrefois, on appelait ce type de condensateur un condensateur fixe ou monobloc.
Nos capteurs ne sont pas particulièrement sensibles à de telles perturbations. En revanche, si vous utilisez des câbles de longueur supérieure à 30 cm, mettez un condensateur de dérivation de 100 nF, directement sur le capteur, entre les bornes de la tension d'alimentation (voir figure).

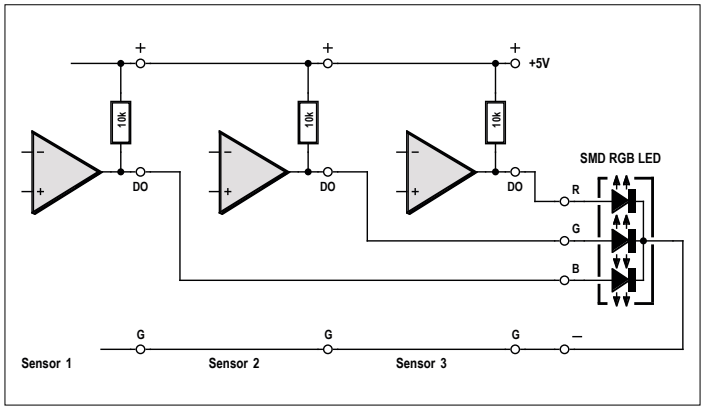


Figure 2. Raccordement direct de la LED RVB.

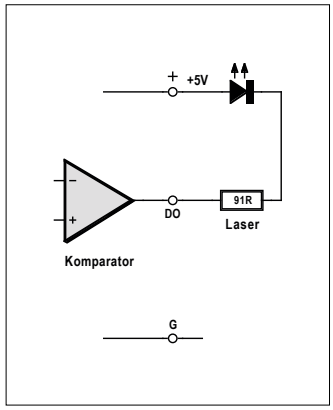


Figure 3. Le laser à la sortie du comparateur.

Lorsqu'on le touche, le capteur tactile fournit généralement un signal rectangulaire à 50 Hz (60 Hz aux États-Unis).
Le circuit du comparateur présente un point de commutation précis, sans hystérésis. Lorsque la température varie lentement par ex., il est possible qu'il y ait une plage dans laquelle la sortie ne cesse de passer d'un état à l'autre. Soit on en tient compte dans son programme et on procède à une évaluation de la sortie numérique à des intervalles de temps suffisamment grands ; soit on utilise le signal de sortie analogique AO directement à partir du diviseur de tension et on effectue soi-même l'évaluation (voir plus bas). Par rapport au capteur CTN déjà présenté, ici il est possible de régler la température souhaitée depuis l'extérieur, avec le potentiomètre.

Nota : la sortie numérique de ces cartes peut commander directement un actionneur, sans passer par un µC. La sortie DO convient pour l'entrée d'une carte à relais. Et voilà notre régulateur de température est terminé, mais sans hystérésis, ce qui peut être gênant pour certaines applications. Si vous souhaitez connecter une des LED directement au comparateur, rappelez-vous qu'il présente une sortie à collecteur ouvert. À l'état haut, seule la résistance de rappel vers le haut de 10 kΩ fournit du courant, la LED ne brille donc pas beaucoup. Il est possible néanmoins d'envisager des applications dans lesquelles on a trois capteurs différents qui commandent directement les trois couleurs d'une LED RVB (**fig. 2**). La luminosité ne sera pas forte, mais quand même bien visible. Le violet (mélange) p. ex. signalerait quelque chose de « brûlant et bruyant ».

Pour commander directement le laser, il convient de le brancher entre DO et le +5 V (**fig. 3**). Certes, le fonctionnement du bouton marche/arrêt est alors inversé, mais on dispose d'un courant plus intense : 20 mA. Ce montage permet de découvrir un phénomène intéressant : la rétroaction opto-thermique. Pointez le laser vers le capteur CTN (voir photo en tête de l'article). Ensuite, à l'aide du potentiomètre, réglez la température de commutation au point de commutation. Le laser commence à clignoter dès qu'on a trouvé le bon point d'enclenchement. À l'état actif, le laser réchauffe légèrement le capteur, la sortie numérique est alors activée ce qui permet de désactiver le laser puisque la connexion est inversée. Ceci permet au capteur de refroidir un peu jusqu'à ce que le comparateur bascule, et ainsi de suite.

Trigger de Schmitt logiciel

Dans la zone de transition, un comparateur peut présenter des états de sortie instables. Le *trigger* de Schmitt logiciel permet d'y remédier en fournissant des points de commutation légèrement décalés pour l'activation et la désactivation. On peut p. ex. mettre en marche à 25 °C et à l'arrêt à 20 °C (cf. article précédent) avec le capteur CTN (*Analog Temp*). Tous les capteurs avec comparateur possèdent une sortie analogique, ce qui permet d'effectuer une comparaison par logiciel et de créer l'hystérésis appropriée. Comme tous les capteurs peuvent être réglés avec le potentiomètre à un point de commutation situé à 2,5 V, on aura recours au même logiciel pour piloter les différents capteurs.
Dans le logiciel, les broches B.2 (LED2 du *shield* d'extension

Listage 1. Un comparateur avec hystérésis (Komparator.bas).

```
Do
  D = Getadc(2)
  If D > 514 Then Portb.2 = 0
  If D > 514 Then Portb.5 = 1
  If D < 510 Then Portb.2 = 1
  If D < 510 Then Portb.5 = 0
...
  Waitms 500
Loop
```

Listage 2. Le comparateur Arduino.

```
//Comparator AD1
...
void loop() {
  value = analogRead(sensorPin);
  if (value > 514) {
    digitalWrite (output1, 1);
    digitalWrite (output2, 0);
  }
  if (value < 510) {
    digitalWrite (output1, 0);
    digitalWrite (output2, 1);
  }
  Serial.println(value);
  lcd.setCursor(0, 0);
  lcd.print(value);
  lcd.print ("   ");
  delay(50);
}
```

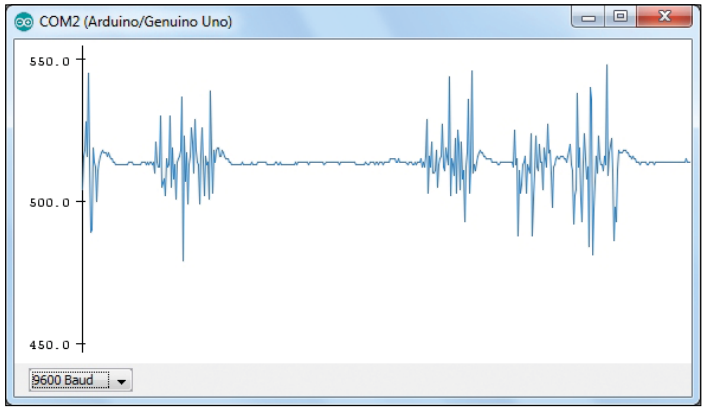


Figure 4. Signaux du microphone.

Elektor, voir partie 1 [2] et B.5 (LED de la carte Arduino) sont configurées en sortie. Nous avons une bonne raison de les piloter en opposition de phase. Soit on monte la LED bicolore sur les deux sorties, soit on attaque entre ces deux sorties des actionneurs qui comportent une résistance série, dans ce cas il y a au moins deux résistances internes en série. La LED RVB montée en surface et la LED infrarouge font partie de ces actionneurs. La LED multicolore présentée plus loin est elle aussi un actionneur, en revanche elle est dotée d'une diode de protection contre la tension inverse qui n'apprécie pas trop la tension inverse. Dans ce cas, il vaut mieux se servir d'une vraie résistance série.

L'exemple en BASCOM dans le **listage 1** permet d'évaluer les données brutes du CA/N (comme toujours, tous les exemples de code sont téléchargeables sur le site d'Elektor [3]). Au milieu de la plage de mesure, on obtient une valeur de mesure de 512. Dans l'exemple, les points de commutation se situent entre 509 et 515. Avec env. 5 mV par pas du CA/N, on obtient ainsi une hystérésis de 30 mV. Le programme commande également la LED2 du shield d'extension. Cela permet de comparer la commutation avec celle de la carte du capteur. Le logiciel permet d'obtenir deux points de commutation légèrement décalés alors que le comparateur du capteur offre un point bien précis. Le programme complet permet d'afficher la tension analogique du capteur sur l'écran du *shield* et dans un terminal, c'est pratique pour régler correctement le potentiomètre.

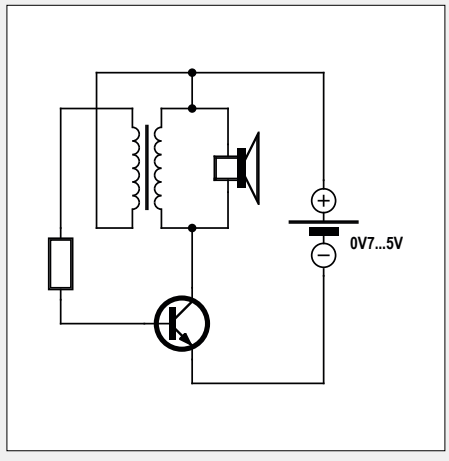
La version Arduino du programme (**listage 2**) n'est pas très différente de l'exemple en BASIC. Les signaux analogiques seront envoyés au traceur série sans aucun effort supplémentaire. La **figure 4** montre les signaux du capteur à microphone. Ce programme peut servir pour les sept capteurs qui se trouvent sur les cartes à comparateur rouges.

Interrogation du capteur tactile

Le capteur tactile ne fournit pas de signal à variation lente, mais en général un signal rectangulaire à 50 ou 60 Hz. Si on devait s'en servir pour commander un relais, le résultat serait plutôt médiocre, bruyant et inélégant. Néanmoins il est possible d'améliorer le signal par logiciel (**listages 3** et **4**). La sortie numérique est alors raccordée à AD1. Certes, il s'agit d'une entrée analogique, mais on peut s'en servir comme port d'entrée numérique. Le résultat de l'interrogation est 1 ou 0, il est directement copié sur un port de sortie. Dans notre cas, nous avons choisi B2 (broche 10 sur l'Arduino) pour pouvoir commander en même temps la LED2 du *shield*. On commande

Oscillateurs

Comme dirait Murphy : fabriquer un oscillateur, c'est parié qu'il n'oscillera pas et fabriquer un amplificateur, c'est parié qu'il oscillera à tous les coups. Disons qu'il y a du vrai dans cette loi. Il vaut donc mieux y regarder de plus près chaque fois qu'on se sert d'un oscillateur dans un circuit. En principe, un oscillateur se compose d'un amplificateur et d'une rétroaction de la sortie vers l'entrée. De plus, le signal envoyé en retour doit avoir la bonne phase. Si la tension augmente en entrée, la tension en sortie devrait également augmenter, après amplification. Il suffit maintenant qu'une partie du signal de sortie soit ramenée à l'entrée par l'intermédiaire d'un condensateur pour avoir un oscillateur.



Il est possible de créer un tel amplificateur avec deux transistors, où chacun introduit un déphasage de 180°. On peut aussi prendre un amplificateur opérationnel ou un amplificateur de haut-parleur intégré.

Un simple étage amplificateur avec un transistor en émetteur commun permet d'effectuer un déphasage de 180°. Une augmentation de la tension d'entrée provoque une diminution de la tension de sortie. Nous avons un oscillateur avec un seul transistor. Il faut simplement s'assurer que la phase est inversée de manière convenable. C'est possible soit avec plusieurs condensateurs et résistances (oscillateur déphaseur), soit avec un transformateur pour la rétroaction (oscillateur Meissner). Si l'oscillateur n'oscille pas comme l'avait prédit la loi de Murphy, il suffit d'inverser l'un des deux bobinages pour que la phase soit correcte. Le petit vibreur devrait avoir cette configuration (figure). Le « haut-parleur » a deux bobinages qui servent en même temps de transformateur.

Si on construit un amplificateur avec beaucoup d'étages et un gain élevé, il est effectivement difficile de parer aux auto-oscillations. Parfois les signaux se faufilent de la sortie vers l'entrée en passant par l'alimentation, on peut empêcher cela avec un condensateur de dérivation plus gros. Parfois il y a une petite capacité entre les lignes de la sortie et celles de l'entrée. Dans les cas extrêmes, on se sert d'une tôle de blindage pour remédier au problème.

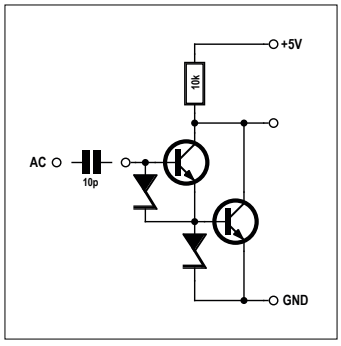


Figure 5. Capteur tactile doté d'un transistor Darlington.

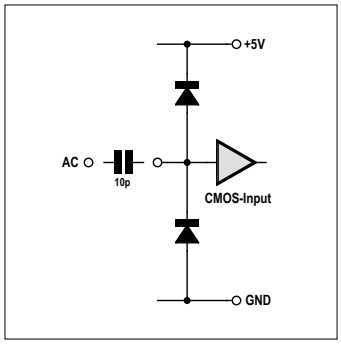


Figure 6. Port d'entrée transformé en capteur tactile.

également B5 parce que c'est là que se trouve la LED interne de l'Arduino. Il est possible d'y raccorder une autre LED ou le laser de manière externe.
Le programme se compose d'une simple boucle qui comporte une pause de 21 ms. La sortie clignote dès qu'on touche le capteur tactile. En effet, au toucher, un signal parasite de 50 Hz est appliqué sur la base du transistor Darlington. C'est la raison pour laquelle la sortie numérique délivre un signal rectangulaire à 50 Hz, soit des impulsions de 20 ms. Si ce signal est échantillonné à une fréquence légèrement différente, on obtient une fréquence beaucoup plus faible. Une période de 21 ms correspond à une fréquence d'échantillonnage d'environ 48 Hz. La différence entre 50 et 48 Hz, soit 2 Hz, apparaît en opposition de phase sur les sorties B2 et B5.

Il n'était pas évident que le transistor puisse être piloté de cette manière. En principe, on fixe un courant de base et on détermine le point de fonctionnement de cette façon. Avec de petits signaux, cela ne fonctionnerait pas sans courant de base. Du point de vue électrique, la personne qui touche le capteur est considérée comme une plaque de condensateur, la deuxième plaque est constituée de tous les fils qui se trouvent dans l'environnement. Le courant de base charge négativement le condensateur jusqu'à ce que le transistor soit bloqué. Le montage ne fonctionne que parce que le transistor bipolaire comporte une diode Zener située entre la base et l'émetteur, pour la plage comprise entre 7 et 10 V (voir **fig. 5**). La tension de ronflement, au repos, d'une personne dépasse généralement les 10 V_{cc}. Il en résulte un courant alternatif dont la demi-onde positive permet de commander le transistor.
Il devrait y avoir un signal rectangulaire approximatif sur le collecteur du transistor Darlington. Raccordons la sortie analogique AO à l'entrée numérique A1. Et voilà, la sortie clignote chaque fois qu'on touche la base. La différence est que cette fois-ci la sortie est au repos parce que le comparateur de la carte du capteur inverse le signal.

Néanmoins on peut se passer de la carte du capteur : pour cela, il suffit de brancher un fil isolé sur l'entrée A1 (**fig. 6**). Si on touche l'isolant à l'extérieur, le système clignote quand même, parce que l'entrée du contrôleur est à très haute impédance. Le fil, l'isolant et le doigt forment un petit condensateur de couplage de quelques picofarads. On applique ainsi une tension alternative à l'entrée, limitée à la plage de tension d'entrée grâce aux diodes de protection internes. Toutefois l'entrée ouverte présente un inconvénient par rapport au vrai capteur tactile. En effet, le repos n'est pas clair, ce qui rend l'évaluation difficile.

Préparation du signal de commutation

Jusqu'à présent, nous avons obtenu un clignotement en sortie lorsqu'on touche le capteur tactile. On en attend plus de ce capteur, comme une mise en marche au toucher. Il faut donc modifier le programme pour que la mise en marche devienne

Listage 3. Entrée et sortie de port en BASCOM.

```
Dim D As Boolean
Config Portb = Output

Do
  Portb.2 = Pinc.1
  D = Pinc.1 Xor 1
  Portb.5 = D
  Waitms 21
Loop
```

Listage 4. Entrée et sortie de port en langage C Arduino.

```
//Touch1  A2 > 10, 13
#include <LiquidCrystal.h>
int input = A2;
int output1 = 10;
int output2 = 13;

void setup() {
  pinMode(output1, OUTPUT);
  pinMode(output2, OUTPUT);
}
void loop() {
  digitalWrite (output1, digitalRead(input));
  digitalWrite (output2, 1-digitalRead(input));
  delay(21);
}
```

Listage 5. Capteur tactile et interrupteur à effleurement (Touch2.bas).

```
Dim T As Word

Config Portb = Output

Do
  If Pinc.1 = 1 Then T = 50000
  'If Pinc.1 = 0 Then T = 50000
  If T > 0 Then T = T - 1
  If T > 0 Then Portb.2 = 1 Else Portb.2 = 0
  If T > 0 Then Portb.5 = 0 Else Portb.5 = 1
  Waitus 10
Loop
```

prioritaire et que la mise à l'arrêt soit légèrement différée. Dans le programme *Touch* (**listages 5** et **6**), la temporisation est réduite à 10 µs, ainsi le port est interrogé plus souvent. Dès qu'un état haut est détecté, le programme lance un compteur T qui démarre à 50.000 et active la sortie. Le compteur compte alors lentement à rebours. La sortie est désactivée au bout de 500 ms env., s'il n'y a pas d'autre impulsion. Le compteur est remis à l'état haut dès que d'autres impulsions surgissent, ce qui permet de prolonger la durée en conséquence. Le montage fonctionne maintenant comme on le souhaite. La sortie est activée dès qu'on touche le capteur et elle est désactivée dès qu'on relâche le capteur. Cette fonction correspond à une bascule monostable réarmable.
Maintenant, le capteur tactile fonctionne avec son transistor Darlington comme on le désire. Le signal rectangulaire sur la sortie numérique est devenu un signal de commutation univoque. Cela fonctionne également dans un environnement sans ronflement du secteur. Il suffit de brèves impulsions dues à des charges statiques pour activer la sortie.
C'est dans un but bien précis que nous avons choisi une temporisation de la boucle beaucoup plus courte (10 µs) que celle requise pour un signal à 50 Hz. De cette façon, le programme convient également aux deux capteurs de son de notre kit. Selon la fréquence acoustique, il peut y avoir en sortie jusqu'à 10 kHz, voire plus. Si le potentiomètre est correctement réglé, on obtient ainsi un interrupteur à claquement de mains qui réagit également aux sifflets et aux cris. Un simple toucher du microphone permet de déclencher automatiquement l'interrupteur. En revanche, la sensibilité dépend en grande partie de la précision du réglage du potentiomètre. Il convient de se rappeler qu'en général ce type de microphone délivre moins de 1 mV. La tension du capteur doit donc être ramenée à quelques millivolts du point de commutation, de manière à ce que les signaux acoustiques puissent conduire le comparateur à des changements de niveau.

Listage 6. Interrupteur à effleurement en langage C Arduino.

```
void loop() {
  if (digitalRead(input) == 1) timeout = 50000;
  //if (digitalRead(input) == 0) timeout = 50000;
  if (timeout > 0) timeout = timeout -1;
  if (timeout > 0) {
    digitalWrite (output1 , 1);
    digitalWrite (output2 , 0);
  }
  else {
    digitalWrite (output1 , 0);
    digitalWrite (output2 , 1);
  }
  delayMicroseconds(10);
}
```

Listage 7. Sortie série supplémentaire.

```
void loop() {
  if (digitalRead(input) == 1) timeout = 50;
  if (timeout > 0) timeout = timeout -1;
  if (timeout > 0) {
    digitalWrite (output1 , 1);
    digitalWrite (output2 , 0);
  }
  else {
    digitalWrite (output1 , 0);
    digitalWrite (output2 , 1);
  }
  Serial.
    println(analogRead(sensorPin)+100*digitalRead
    (output1));
  delay (20);
}
```

Capteur de chocs

Certains capteurs numériques comme le *tap module* conviennent bien à ce type d'application. Le module comporte un ressort qui touche un contact en cas de vibration. Il en résulte de très brèves impulsions. En revanche, le module est doté d'une résistance de rappel vers le haut, de sorte qu'au repos la tension de sortie est à +5 V tandis que les impulsions ont 0 V. Dans ce cas, il convient d'inverser l'interrogation de l'entrée (c'est signalé dans le logiciel dans une ligne de commentaire). Autre solution : il est possible d'intervertir le +5 V et GND, et de laisser le logiciel tel quel, car la polarité n'influe ni sur l'interrupteur ni sur la résistance.
La même chose a été testée avec le capteur de chocs qui lui aussi se sert d'un contact interne qui se ferme un court instant en cas de vibrations. Le détecteur de chocs est plus sensible que le module *tap*. Ce programme permet également d'évaluer l'interrupteur à bille (*Ball Switch*). Une petite bille roule vers le bas lorsque l'inclinaison est correcte, ce qui permet de fermer deux contacts. Si l'on remue rapidement le capteur, on entend la boule bouger et le programme active en même temps la sortie.

En fait, cette méthode permet d'interroger tous les capteurs de commutation y compris le bouton-poussoir (*Button*), le commutateur Reed (*Mini Switch*) et l'interrupteur Reed (*Reed Switch*) de la carte à comparateur. Ce type de temporisation permet en effet d'éliminer les rebonds de l'interrupteur. Presque tous les contacts mécaniques (sauf les interrupteurs au mercure, de toute manière interdits à cause du mercure) rebondissent une ou plusieurs fois à la fermeture, si bien qu'au début plusieurs impulsions sont détectées. Cette méthode permet de convertir ces multiples impulsions en une seule impulsion longue. En choisissant d'autres durées, cette méthode permet également de transformer le montage en minuterie par ex. pour éclairer une cage d'escalier. La lumière peut alors être déclenchée de différentes façons : taper des mains, toucher, frapper, approcher un aimant. On peut aussi de cette manière interroger la sortie numérique du capteur de température. Ceci permet de régler le problème des multiples transitions entre état haut et état bas.
Une simple modification (**listage 7**) permet d'expliquer le fonctionnement d'un capteur-comparateur également avec un moniteur série. Pour cela le signal analogique est acquis sur AD2 tandis que le signal de sortie numérique du comparateur

continue de déclencher le processus de commutation proprement dit. Pour pouvoir représenter en même temps le signal de sortie avec un seul canal, la sortie est augmentée de 100 à l'état haut. On voit que le signal analogique initial ressemble à une marche d'escalier. La **figure 7** montre le résultat obtenu avec le capteur de son. On voit bien qu'un signal acoustique doit dépasser un certain niveau pour que la sortie commute. À la fin de tous les signaux, la temporisation qui a été programmée s'écoule et la sortie est désactivée.
La **figure 8** montre le fonctionnement du capteur de température. Le déroulement du programme a été ralenti de 100 ms. Durant la période de mesure, nous avons touché deux fois le capteur, ce qui l'a réchauffé. On voit bien l'inversion effectuée par le comparateur : une baisse de tension sur le capteur permet d'activer la sortie du comparateur. On reconnaît bien également une courte phase d'oscillations autour du point de commutation. Le logiciel permet de supprimer efficacement ces oscillations.

Buzzer et autres actionneurs

La carte à relais peut de nouveau être reliée à la sortie B2. Il nous reste d'autres actionneurs qui ne demandent qu'à être utilisés, comme le vibreur actif (*Buzzer*) encore revêtu d'une feuille de protection. Comme il consomme 25 mA sous 5 V, il peut être relié directement à un port. En revanche, sa polarité n'est pas évidente, elle dépend du soudage du vibreur sur la carte. Sur notre carte, les bornes moins et signal étaient interverties, la broche S se trouvait sur GND. Un test en laboratoire a montré que le vibreur fonctionne à partir de 0,7 V. La fréquence est modifiée lorsque l'évent est fermé. Sans même ouvrir le boîtier, l'expert sait déjà ce qu'il va y trouver : un oscillateur auto-oscillant avec un transistor au silicium bipolaire dont la tension de seuil habituelle est comprise entre 0,5 et 0,7 V, ce qui impose une tension de service avec la bonne polarité. Attention, le vibreur actif ne doit pas être confondu avec le vibreur passif qui ressemble plutôt à un petit haut-parleur de 16 Ω.

Il existe un autre actionneur intéressant, la LED tricolore (*Color Flash*) avec contrôleur intégré. Cette LED automatique produit des mélanges de couleurs (rouge, vert et bleu). C'est là qu'intervient une nouvelle fois la résistance série, car les LED multicolores sont conçues pour une tension de 3 V. La carte est dotée d'une résistance de 10 kΩ, mais montée en parallèle (**fig. 9**). Une résistance de 100 Ω en série avec le 5 V serait idéale. Une fois de plus, le compromis présenté dans l'article précédent fera l'affaire : on connecte la LED entre deux broches de port, le courant est alors limité par les résistances internes du port. La broche 12 (B4) sert de pôle opposé.

Le contrôleur interne de la LED multicolore comporte comme la plupart des circuits intégrés une diode de protection inverse sur les bornes d'alimentation et réagit très mal à une inversion de polarité de la tension de service. Attention, une inversion de polarité sans résistance série peut entraîner la destruction du montage. Nota : GND se trouve au milieu des trois bornes. En passant, pour vous amuser, vous pouvez raccorder le mini haut-parleur passif en série. Ceci vous permettra non seulement de visualiser les commutations, mais aussi de les entendre.

(160173 – version française : Pascal Duchesnes)

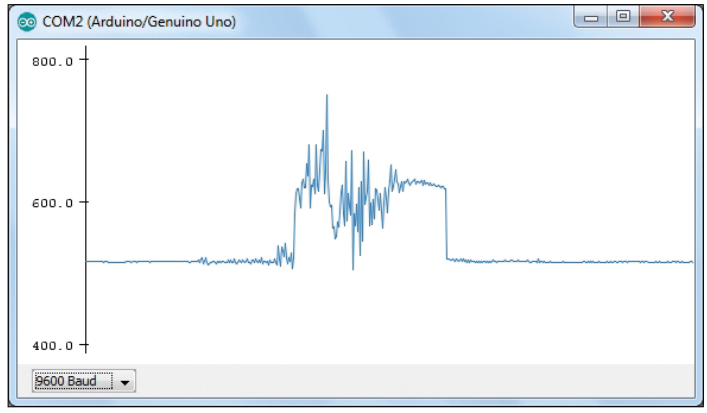


Figure 7. Commutations avec le capteur de son.

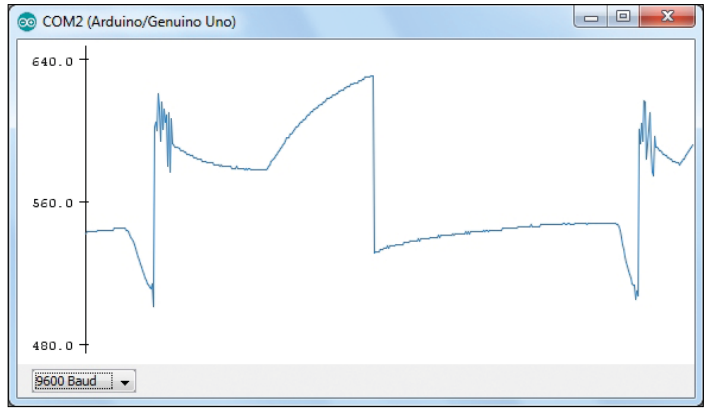


Figure 8. Le capteur de température en action.

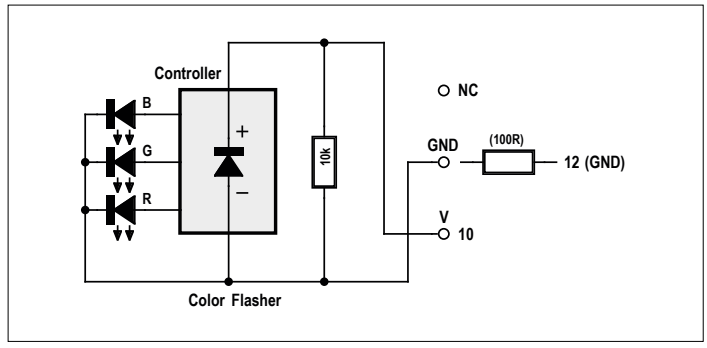


Figure 9. LED multicolore.

Liens

[1] www.elektor.fr/arduino-sensor-kit
[2] www.elektormagazine.fr/160152
[3] www.elektormagazine.fr/160173

LoRa™ échange débit contre portée

Robert Lacoste (Chaville)

Dans mon précédent article [1], je vous ai expliqué que seuls trois paramètres déterminent la sensibilité d'un récepteur radio. Pour mémoire, ce sont respectivement la qualité de l'électronique, les performances de la modulation utilisée, et surtout le débit de la liaison. Ici je passe de la théorie à la pratique : la solution LoRa développée par Semtech Semiconductors. Pour ceux que cela intéresse, sachez d'ores et déjà que je continuerai dans le même esprit dans le prochain article, avec LoRaWAN, un protocole standardisé utilisant évidemment LoRa…

LPWA ?

Démarrons par quelques généralités. Si vous ne vivez pas dans une grotte, vous savez sûrement que l'on nous promet des dizaines de milliards d'objets connectés dans les prochaines années. Imaginons que vous deviez développer un tel objet connecté, bien sûr sans fil. Comment le relier au grand Internet ? Pour faire simple, trois grandes approches existent (**fig. 1**).

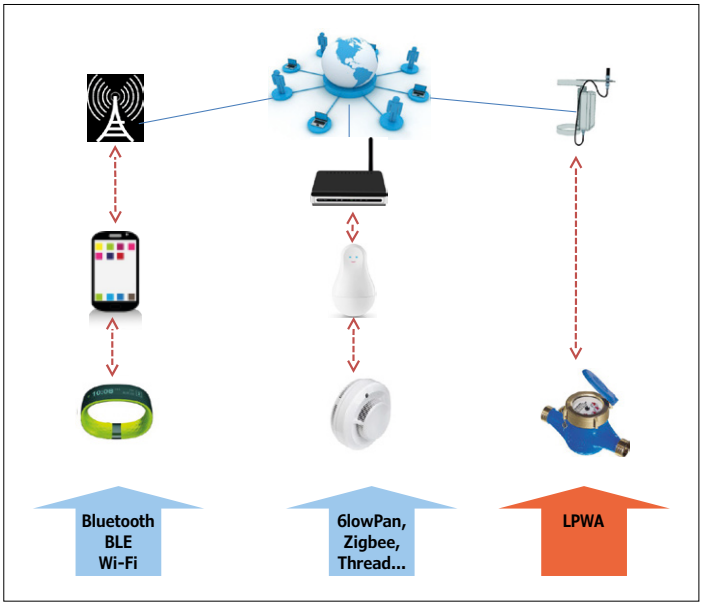


Figure 1. Les trois grandes familles de réseaux pour raccorder un dispositif connecté à l'internet.

La première solution, probablement la plus utilisée aujourd'hui, consiste à utiliser une passerelle de communication qu'on a tous dans notre poche : l'ordiphone. Votre objet connecté peut en effet se connecter par radio à celui-ci, typiquement via Bluetooth, Bluetooth Low Energy ou Wi-Fi, et l'ordiphone assure la connexion à l'internet. Les avantages sont la facilité, la faible consommation et le très bas coût, mais il faut un ordiphone.

La seconde possibilité consiste à remplacer l'ordiphone par une passerelle fixe, installée à votre domicile. Votre « box internet » peut faire l'affaire si votre objet est doté d'une interface Wi-Fi, sinon une passerelle spéciale sera nécessaire. C'est le domaine des protocoles sans fil de type « domotique » comme Zigbee, Zwave, 6lowPan ou Thread. C'est à très bas coût et sans ordiphone, mais installer des passerelles est toujours pénible. De plus, pour des projets un peu ambitieux, le nombre de passerelles nécessaires croît très vite, car ces solutions ont des portées assez faibles.

La troisième voie consiste à éviter toute passerelle locale, en adoptant une technologie à très longue portée. En effet, couvrir une grande distance permet de faire communiquer votre dispositif directement avec une passerelle qui rayonne sur tout un pâté de maisons, voire une ville entière. Traditionnellement cette solution nécessitait d'utiliser un réseau cellulaire (2G ou 3G par ex.), d'où des coûts et des consommations non négligeables. Ces dernières années, plusieurs nouvelles solutions ont toutefois changé la donne. Baptisées du sobriquet de LPWAN (*Low Power Wide Area Network*, réseau étendu à faible consommation), ces réseaux atteignent des portées qui se chiffrent en kilomètres, tout en étant peu énergivores.

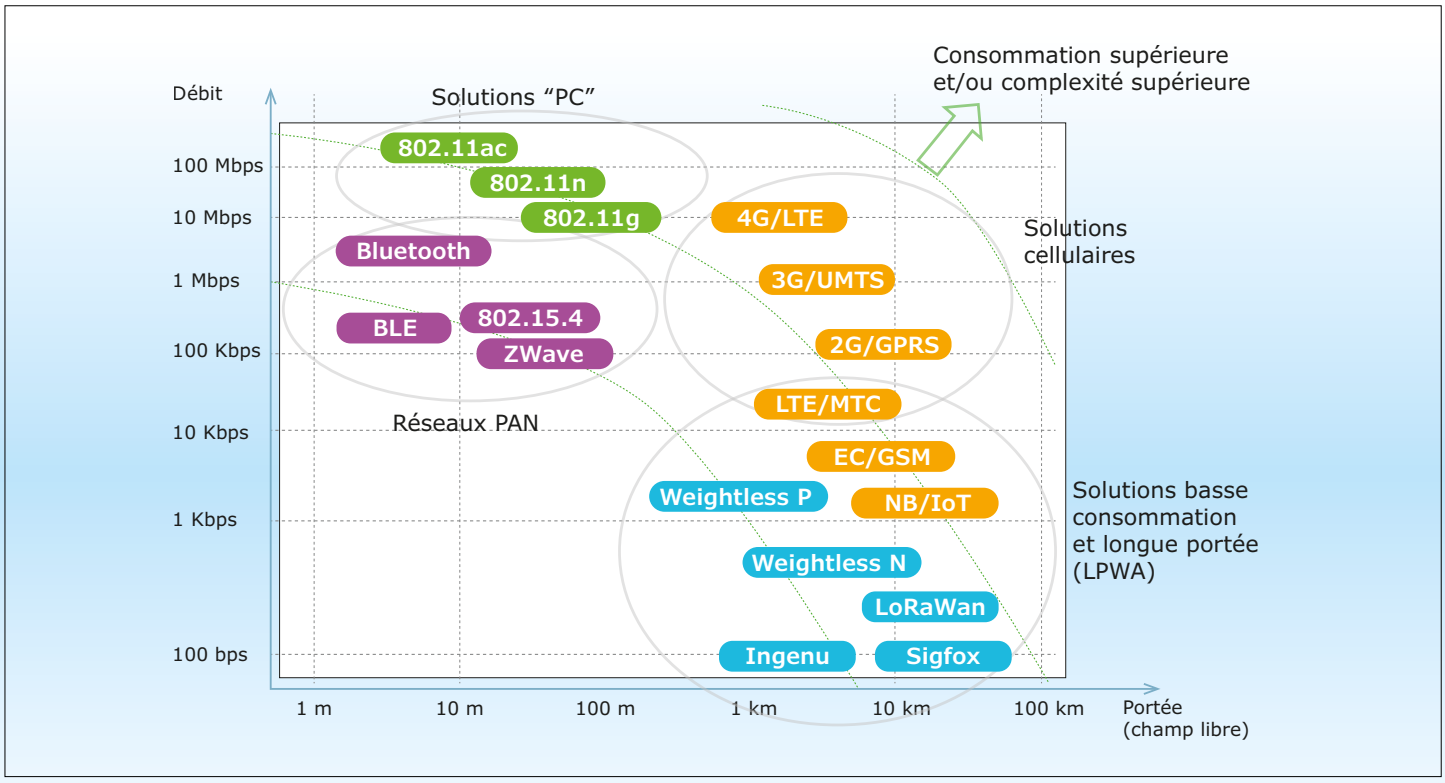


Figure 2. Un panorama global des technologies radio. Les solutions LPWA troquent débit contre portée.

Réduire le débit !

L'idée générale du LPWA est simple. Comme l'illustre la **figure 2**, les technologies radio classiques (Wi-Fi, Bluetooth, etc.) privilégient des débits très élevés et ont donc une portée réduite. Le réseau LPWA répond à d'autres exigences : il doit porter beaucoup plus loin, mais ne pas consommer plus, ce qui a pour conséquence une réduction drastique du débit de la liaison. Le réseau étendu LoRa est une solution parmi d'autres : Sigfox, Ingenu, Weightless, etc. Notez que l'on observe cette tendance chez les opérateurs de téléphonie mobile également : même s'ils augmentent de manière faramineuse les débits en 4G pour vous permettre de regarder des vidéos en HD sur votre ordiphone, ils déploient aussi des réseaux de type NB-IoT (*Narrow Band Internet of Things*) qui privilégient justement des débits nettement plus faibles pour le monde de l'Internet des Objets.

Est-il si simple de réduire considérablement le débit d'une liaison radio ? La réponse est non. Imaginez que vous avez une transmission d'un débit de 10 kbps qui utilise une modulation classique de type FSK (*Frequency Shift Keying*, modulation par déplacement de fréquence). Cette modulation augmente ou baisse la fréquence de l'émetteur pour transmettre respectivement un 0 ou un 1, typiquement sur une plage de l'ordre de ±10 kHz ici. Si vous souhaitez réduire le débit de la liaison, disons à 100 bps, il faut, pour que la modulation conserve les mêmes caractéristiques, réduire aussi ce déplacement de fréquence à ±100 Hz. C'est là que le bât blesse. Cela implique que le récepteur soit très stable en fréquence. Par ex. si la fréquence de la porteuse est de 868 MHz, ±100 Hz représente ±0,1 ppm (partie par million), bien plus faible que la dérive de n'importe quel quartz classique. Cette technique, baptisée à bande ultra-étroite, existe, c'est celle utilisée par Sigfox, mais elle nécessite de sacrées ruses du côté du récepteur.

La solution LoRa

LoRa est une solution alternative à ce problème, elle permet de réduire énormément le débit d'une liaison sans pour autant nécessiter de quartz très précis. Pour la petite histoire, LoRa a été inventée et développée en 2009 par une jeune pousse française, Cycleo, rachetée en 2013 par le fabricant de semi-conducteurs Semtech. Avec LoRa, la plage de modulation est toujours large (typiquement de 125 à 500 kHz), mais les bits à transmettre sont encodés avec une technique qui réduit le débit sans modifier la largeur de modulation. Cette modulation met bien sûr également à profit des techniques avancées (correction d'erreur, etc.) pour améliorer autant que possible la sensibilité du récepteur.

Pour les lecteurs inquiets, je me permets de proposer une relecture de mon précédent article : à débit binaire égal, la sensibilité d'un récepteur est la même que l'on utilise une modulation à bande ultra-étroite ou très large (pour une même complexité du récepteur), donc cette approche a tout son sens…

Une solution à large bande comme LoRa offre plusieurs avantages : la possibilité d'utiliser un quartz ordinaire, mais aussi une meilleure insensibilité aux parasites (en général à bande étroite) et une très grande flexibilité. Évidemment il y a des inconvénients. Tout d'abord le récepteur nécessite des traitements numériques complexes. Heureusement il y a des circuits intégrés spécialisés pour cela (j'y reviens plus bas). L'autre

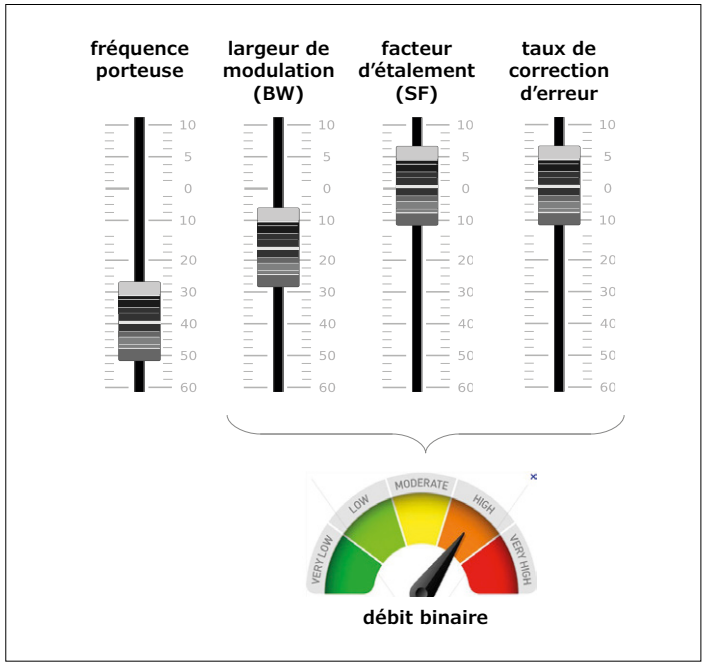


Figure 3. LoRa permet de régler indépendamment quatre paramètres qui ont tous un impact sur la portée et le débit.

inconvénient potentiel est un risque de saturation plus rapide du canal radio, mais pour l'instant on n'en est pas encore là.

CSS, vous avez dit CSS ?

Rentrons dans les détails. LoRa est ce qu'on appelle une couche physique. C'est-à-dire que c'est juste une méthode d'encodage de la suite de bits qui constitue le message, pour l'envoi dans les airs, et avant de procéder au traitement inverse du côté du récepteur. Lora ne se charge pas des fonctions nécessaires pour l'application, par ex. le codage de l'adresse de l'émetteur et des destinataires, les acquittements éventuels…, c'est le rôle d'un protocole de plus haut niveau (LoRaWAN en est un). Comme je l'ai annoncé, l'une des forces de LoRa est sa très grande flexibilité. Tout d'abord la fréquence de la porteuse peut

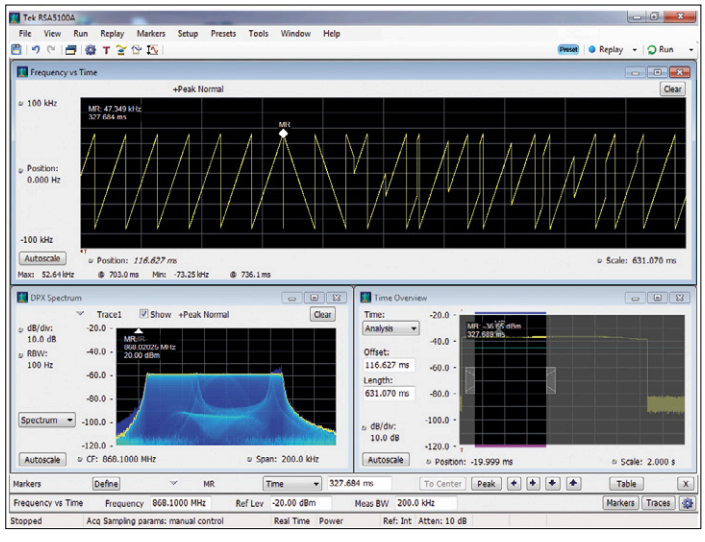


Figure 4. Une trame LoRa visualisée avec un analyseur de spectre en temps réel, ici un Tektronix RSA5106.

être choisie dans la plage de 137 MHz à 1020 MHz, avec les composants actuellement disponibles. Cependant, en Europe, la plupart des applications utilisent la bande 868 MHz. L'utilisateur peut également régler librement trois paramètres qui influencent tous le débit binaire (**fig. 3**). Tout d'abord la **largeur de modulation**, notée BW (*Band Width*), peut être sélectionnée entre 7,8 kHz et 500 kHz. Une bande plus large donne bien sûr un débit plus rapide. Ensuite un paramètre étrange nommé **facteur d'étalement** (SF, *Spreading Factor*) peut être réglé entre 6 et 12. Augmenter ce nombre de 1 réduit le débit de moitié. Enfin il est possible de sélectionner un **code de correction d'erreur** plus ou moins efficace ; il ajoute de 0 à 4 bits de correction d'erreur tous les 4 bits utiles transmis.
C'est bien beau tout cela, mais comment ça marche ? Il y a là un problème. Les détails de la modulation LoRa ne sont pas officiellement divulgués par Semtech, qui nous dit juste que LoRa utilise une modulation particulière appelée CSS (*Chirp Spread Spectrum*) [2]. Si vous regardez avec un analyseur de spectre classique l'allure d'un signal LoRa, vous serez déçu : on observe une sorte de plateau qui occupe une largeur de bande BW autour de la fréquence de la porteuse choisie, rien de plus. Plus précisément, à un instant donné le signal est une porteuse pure, mais sa fréquence évolue dans le temps un peu comme en FM. Pour y comprendre quelque chose, il faut utiliser un analyseur de spectre sophistiqué qui permet de mesurer et d'afficher la fréquence du signal au cours du temps. Fort heureusement mon labo dispose d'un joli analyseur de spectre en temps réel Tektronix RSA5106 avec ce genre de fonction. Regardez la **figure 4** pour voir le résultat. Dans l'analyse de fréquence en fonction du temps, on voit que la fréquence est en permanence modifiée, avec une vitesse de balayage constante. Par contre il y a des « sauts » dans cette variation, ce sont eux qui encodent les bits transmis. Maintenant vous avez compris pourquoi on parle d'*étalement de spectre à modulation de fréquence* (CSS) : si l'on revient à des fréquences audio et qu'on applique à un haut-parleur ce signal avec de telles rampes en fréquence, on entend un son ressemblant à *chiiiiiiirp-chiiiiirp*… Les Français appellent plutôt ça un signal wobulé (*woooaab-woooaab* au lieu de *chiiiirp-chirrrrp*...), mais c'est un peu la même chose !

Bien que les détails de la modulation LoRa ne soient pas officiellement divulgués, nombre de personnes ont creusé le sujet. La **figure 5** montre les grandes lignes de l'encodage utilisé avec les paramètres suivants pour l'exemple : BW = 125 kHz, SF = 12, code correcteur 5/4. Avec BW = 125 kHz, la largeur du balayage en fréquence est bien sûr de 125 kHz, c'est l'amplitude sur l'axe des ordonnées du graphe *fréquence=f(t)*. Ensuite on remarque que chaque trame radio commence par un préambule constitué de rampes complètes, suivi de quelques rampes « inversées » (la fréquence réduit au cours du temps au lieu d'augmenter). Ces rampes inversées sont le signal de synchronisation qui indique au récepteur le début du message effectif.
Comment sont codés les bits à transmettre ? Tout d'abord, les bits du message sont groupés par paquets, qu'on appelle des symboles. Comme on a sélectionné SF = 12, ces bits sont regroupés 12 par 12. Un symbole est donc un nombre de 0 à 4095 ($2^{12} - 1$), et va être transmis en une seule fois. Sur le graphe, on remarque qu'il y a un saut exactement

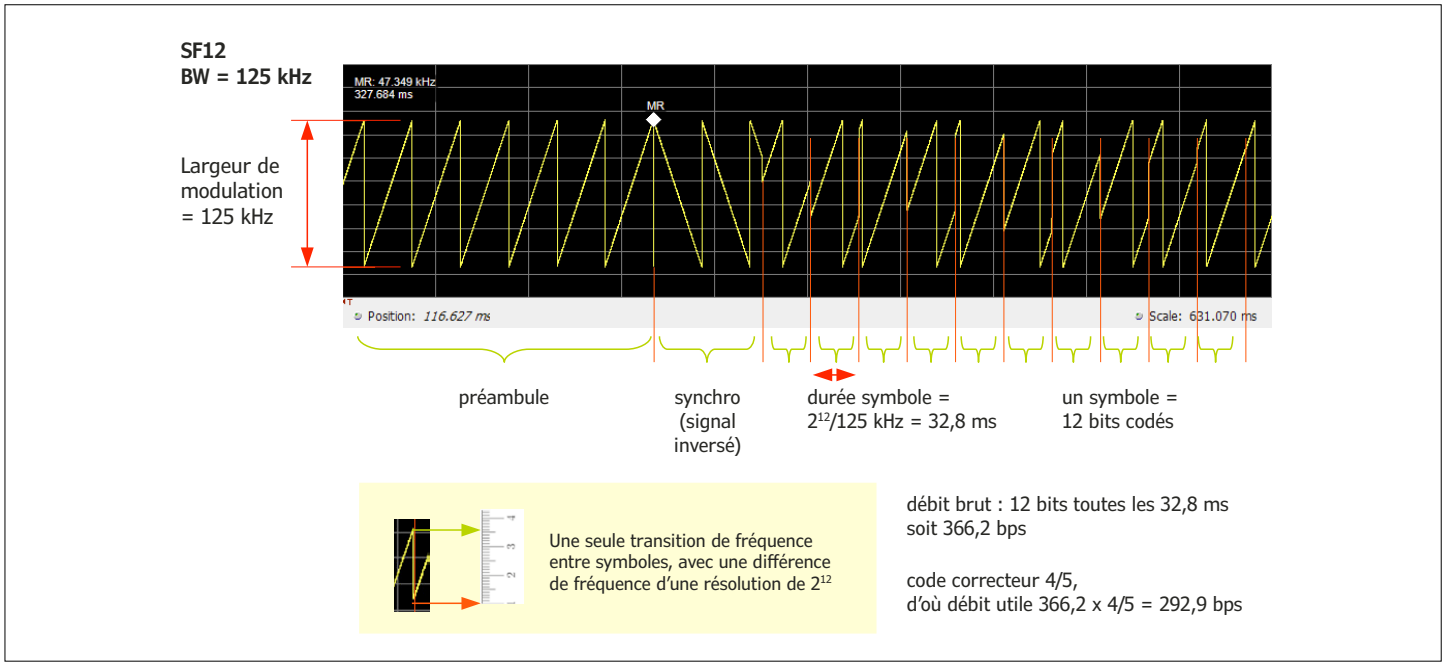


Figure 5. LoRa décrypté !

toutes les 32,8 ms dans la séquence de balayage de fréquence. Cela correspond à la durée en LoRa d'un symbole avec BW = 125 kHz et SF = 12 ; cette durée est calculée avec la formule $2^{12}/125.000$ Hz = 0,0328 s. Dans ce mode, LoRa transmet un symbole, soit 12 bits d'un coup, toutes les 32,8 ms, soit 366,2 bps si vous faites le calcul. Ajoutez-y un bit de correction d'erreur tous les 4 bits et vous obtiendrez un débit utile de 292,9 bps, ce qui est bien le débit de LoRa dans cette configuration.

Vous vous demandez peut-être comment un symbole de 12 bits est codé en un saut de fréquence ? C'est simple, du moins sur le papier : il suffit de décomposer la plage de variation de fréquence de 125 kHz en 2^{12} = 4096, soit des pas de 125.000/4096 = 30,52 Hz. Si le symbole de 12 bits vaut 1, LoRa fait un saut de 30,52 Hz ; pour un symbole égal à 2, le saut est d'une fréquence deux fois supérieure, et ainsi de suite jusqu'à la valeur 4095. Bon, évidemment c'est plus complexe que cela, car la loi d'encodage est plus subtile, mais vous avez compris l'idée. Sachez qu'une rétroanalyse complète de LoRa est maintenant disponible sur la toile, voir [3]

Outil logiciel...

Le calcul du débit binaire effectif d'une liaison LoRa en fonction des différents paramètres n'est donc pas très simple. Heureusement Semtech met à disposition une petite application Windows qui se charge des calculs [4]. Regardez la **figure 6** où j'ai sélectionné les mêmes paramètres que dans mon exemple ci-dessus (BW = 125 kHz, SF = 12, codage 5/4). L'outil indique que le débit binaire effectif sera de 292 bps, comme prévu. Ouf ! L'application calcule également la durée d'émission d'une trame avec le nombre voulu d'octets utiles, en ajoutant les préambules et autres octets d'encapsulage. Les ordres de grandeur sont intéressants : pour huit octets utiles avec ces paramètres, la durée d'émission sera de 925 ms.

Ceci permet de toucher du doigt une limitation de toutes les solutions à bas débit comme LoRa : en Europe, la réglementation impose des limites au temps d'émission dans les bandes « libres » pour laisser de la place aux copains. En particulier, dans la bande 868-868,6 MHz utilisée principalement par LoRa, cette limite est égale à 1 % du temps, calculée sur une heure glissante. 1 % de 3600 s fait 36 s, il n'est donc possible de n'envoyer au maximum qu'une petite quarantaine de messages de 8 octets par heure. Oubliez donc les rêves de transfert de gros fichiers de données, les solutions LPWA visent l'envoi de petits messages peu fréquents. Sinon la seule solution consiste à augmenter le débit, mais là on perd en portée…
Pour finir, l'application Windows fournit également une estimation de la sensibilité du récepteur avec les paramètres sélec-

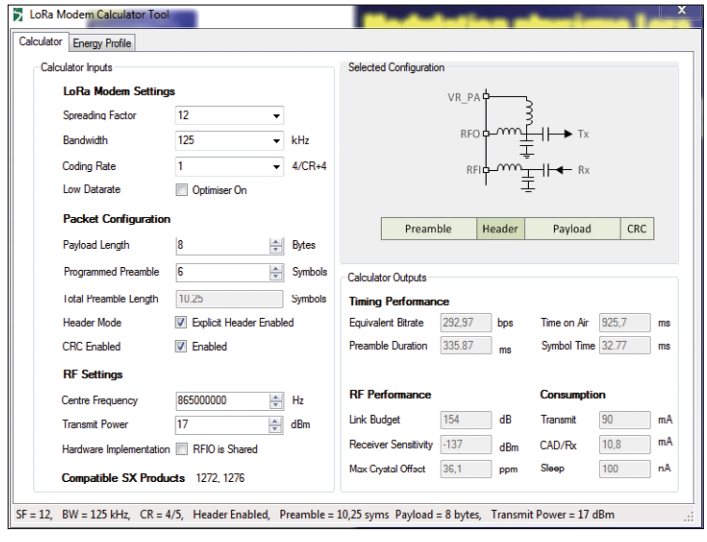


Figure 6. Copie d'écran du logiciel *LoRa Modem Calculator* fourni par Semtech, une aide appréciable...

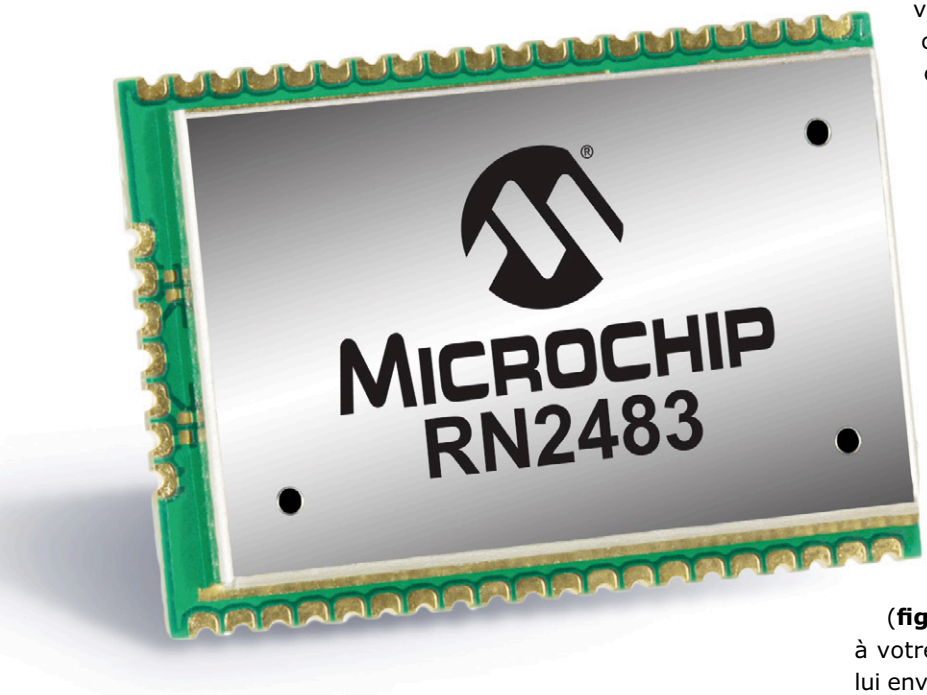


Figure 7. Un exemple de module LoRa.

tionnés. Regardez de nouveau la figure 6, elle est ici de l'ordre de −137 dBm. Avec un émetteur d'une puissance de 25 mW (soit 14 dBm), cela représente un bilan de liaison de 151 dB. Ce chiffre ne vous impressionne pas ? Et si je vous dis que cela correspond théoriquement à une portée de plus de 900 km en champ libre. Toujours pas ? Sur le terrain, il y a bien entendu des pertes, des obstacles et des réflexions, mais des portées de plusieurs dizaines de kilomètres sont tout à fait envisageables avec des antennes bien choisies et pas trop mal placées…

Et le matériel ?

Comment utiliser LoRa dans votre prochain projet ? Tout d'abord vous pouvez soit n'utiliser que la couche physique LoRa et interconnecter vos dispositifs en point à point, soit avoir recours à LoRaWAN comme je vous l'expliquerai dans mon prochain article. Ensuite sur le plan matériel, vous avez deux options : une puce ou un module. Pour les puces, le choix est réduit, car le seul fabricant de circuits intégrés radio qui supportent LoRa est Semtech. La situation changera courant 2017 grâce à des accords de licence prévus avec STM et Microchip, donc surveillez la presse… Pour le moment, furetez sur semtech.com et téléchargez la documentation du SX1276 [5], c'est le composant le plus complet de la gamme LoRa ; il couvre toute la gamme de fréquences de 137 MHz à 1040 MHz. À noter qu'une version un peu moins chère est suffisante pour le 868 MHz : le SX1272. Vous verrez que les composants externes nécessaires sont peu nombreux : un quartz, quelques composants passifs, un connecteur d'antenne et bien sûr un microcontrôleur pour le piloter (en général un petit 32 bits de type Cortex M0). Pour le logiciel, vous trouverez de quoi démarrer en *open source* en cherchant « SX1276 » sur le site www.github.com.

L'autre solution, plus simple, consiste à mettre en œuvre un module prêt à l'emploi qui contient un circuit intégré qui supporte LoRa, un microcontrôleur préprogrammé et tous les composants auxiliaires. De très nombreux modules de ce genre sont maintenant disponibles, comme le RN2483 de Microchip (**figure 7**), il coûte moins de 15 €. Il suffit de le raccorder à votre microcontrôleur principal via une liaison UART et de lui envoyer quelques commandes ASCII pour être connecté en LoRa. Ah, il faudra aussi une antenne… Pour les aficionados du format Arduino, vous trouverez aussi un *shield* supportant un SX1276 et la tripaille nécessaire, compatible avec l'environnement de développement Mbed d'ARM (SX1276MB1xAS [7]). Elektor propose également dans son e-choppe un HAT pour RPi [8]. Aucune raison donc pour attendre !

Pour conclure

Les réseaux LPWA, et LoRa en particulier, permettent comme vous l'avez compris d'établir des liaisons par radio sur de longues distances et à faible coût, avec comme principale contrainte un débit de données très lent, ce qui limite donc le flot de messages. Mais cela répond quand même à de très nombreuses applications !

Dans le prochain article, je vous présenterai LoRaWAN, le protocole qui s'appuie sur LoRa et qui a été développé par l'alliance éponyme. Le fait que cette Alliance LoRa regroupe déjà plus de 200 *petites* sociétés dont Cisco, IBM, ARM, Orange, Bouygues Telecom, KPN, Proximus, ST Microelectronics, Microchip et quelques autres est probablement un bon signe pour l'avenir.

(160236)

Cet article a été publié dans la revue Circuit Cellar (n°313, août 2016).

Liens et références

[1] *Hors Circuits : bruit et sensibilité des récepteurs*, Elektor, 12/2016 : www.elektormagazine.fr/160307

[2] AN1200.22, *LoRa Modulation Basics* : www.semtech.com/images/datasheet/an1200.22.pdf

[3] *Reversing Lora*, Mark Knight / Bastille networks : https://archive.org/details/ReversingLora

[4] Outil de calcul, *LoRa Calculator: fast evaluation of link budget, time on air and energy consumption* : www.semtech.com/wireless-rf/rf-transceivers/sx1272/

[5] Émetteur-récepteur LoRa SX1276 : www.semtech.com/wireless-rf/rf-transceivers/sx1276

[6] Module sans fil LoRa RN2483 : www.microchip.com/RN2483

[7] Shield LoRa SX1276MB1xAS : developer.mbed.org/components/SX1276MB1xAS/

[8] HAT LoRa pour RPi : www.elektor.fr/dragino-lora-gps-hat-for-raspberry-pi

programmes de CAO gratuits

outils de création de circuits imprimés offerts (ou presque)

Il n'y a guère d'électronicien qui se satisfasse de laisser tourner un circuit sur une carte de prototypage. C'est particulièrement difficile lorsque le circuit est complexe ou comporte des composants CMS, les plaques d'essai sont alors d'une utilité limitée. Ce qu'il faut, c'est un « véritable circuit imprimé » conçu sur ordinateur. Le marché propose quelques outils gratuits, dont voici une revue – non exhaustive.

Harry Baggen, Thijs Beckers et **Thomas Scherer**

Certes, il y a l'esthétique, mais c'est surtout parce que la robustesse et la fiabilité d'un circuit électronique dépendent fortement de sa réalisation concrète qu'un circuit imprimé est indispensable pour les plus petites séries et même pour une seule pièce. Mais se former à l'utilisation d'un logiciel de conception de circuit imprimé coûte du temps et de l'énergie. De plus, les fichiers des schémas et des circuits imprimés sont souvent dans un format spécifique au logiciel et guère exportables (à l'exception des fichiers de production comme Excellon, etc.). De plus, les différents logiciels proposent des fonctions très diverses. Le dessin du schéma et celui du circuit imprimé sont sans doute standard et incontournables. Quelques logiciels se distinguent par la simulation intégrée ou une représentation en 3D du circuit imprimé, c'est utile pour choisir un coffret. Les points significatifs sont les fonctions du logiciel et la richesse de la bibliothèque de composants, car, si l'on peut toujours en ajouter, c'est souvent long et fastidieux. Enfin, pourront marquer des points les logiciels qui associent les composants à un distributeur.
Voici donc une sélection de logiciels pour créer des circuits imprimés, qui devrait faciliter votre choix.

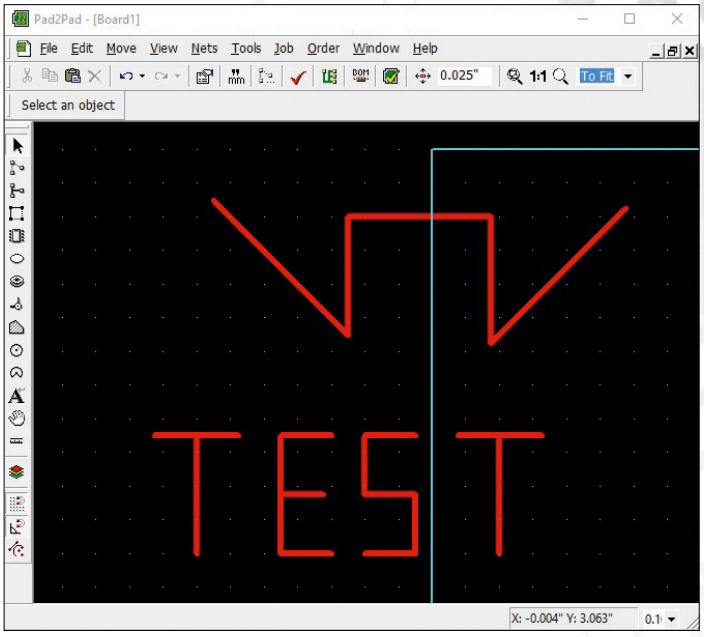


Figure 1. L'interface rustique du programme de routage Pad2pad.

Pad2Pad (v. 1.9.111)

Pad2Pad est un fabricant de circuits imprimés qui s'est spécialisé dans le marché du CI sur mesure sur l'internet. Avec son logiciel gratuit Pad2Pad, on peut dessiner un circuit imprimé très facilement (mais, hélas, pas un schéma) et envoyer les fichiers de production au fabricant.
Après le démarrage du programme, il faut saisir les caractéristiques du circuit imprimé, comme le nombre de couches, ainsi que le nombre d'exemplaires prévu. La conception du CI commence par l'implantation des composants, pour lesquels le logiciel dispose d'une bibliothèque généreuse, qu'on peut soi-même enrichir d'un composant manquant. La commutation entre le système métrique et le système anglo-saxon (dimensions en pouces) ne s'effectue malheureusement pas en un seul endroit, il faut la refaire dans pratiquement chaque fenêtre.
Il y a une appréciable quantité de modèles ou de raccourcis. On peut par ex. numéroter correctement des connexions en une seule opération ou transférer le routage d'une carte connue (par ex. un *shield* Arduino) sur son circuit imprimé.
Malheureusement, le programme s'est planté lors des essais, ce qui a provoqué l'envoi d'un rapport d'erreur à l'éditeur. L'importation de fichiers dxf, créés avec Eagle (v. 6.4), s'effectua non sans erreurs. Des optimisations s'avèrent donc encore indispensables.
Après l'enregistrement du logiciel, on reçoit quotidiennement un courriel avec un lien vers un tutoriel vidéo, ce que nous avons trouvé bien utile. La documentation complète est en ligne, y compris le tutoriel.
L'éditeur améliore constamment le logiciel et envoie régulièrement des mises à jour qui apportent de nouvelles fonctions et corrigent les bogues. Pad2Pad tourne sous Windows, à partir de la version XP.

gEDA

gEDA, comme son nom l'indique, est une collection d'outils pour l'***E**lectronic **D**esign **A**utomation* (en français, conception

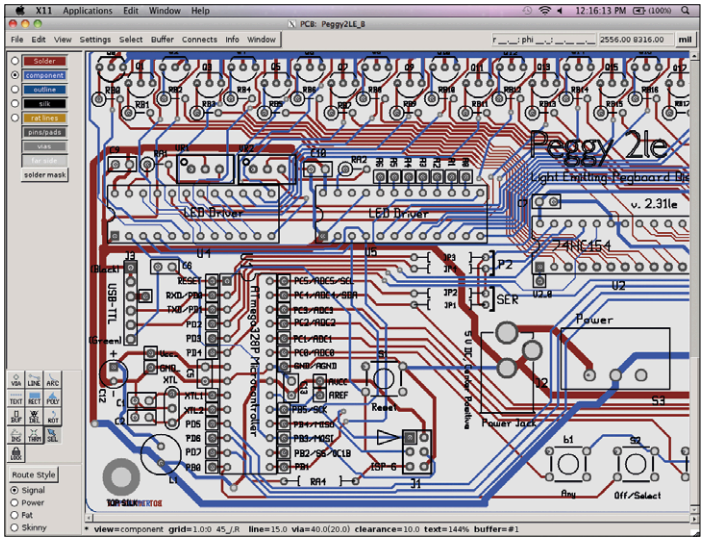


Figure 2. gEDA sous X11 sur Mac.

de systèmes électroniques assistée par ordinateur), diffusée sous licence GPL et tournant sous Linux (SUSE et Debian) ainsi que sous OS X. Voilà qui semble prometteur à première vue, mais les versions du logiciel que nous avons testées (*gschem* 1.8.2 et *PCB* 20140316) présentent encore un bon nombre de petits bogues. On en arrive rapidement à devoir recourir aux menus d'aide, à la documentation, aux FAQ, au Wiki. Mais une fois qu'on s'y est mis et qu'on a mémorisé les commandes au clavier, il est facile de se servir de l'éditeur de schémas. Nota : sur le Mac, la suite n'est pas intégrée à l'interface graphique, mais tourne sous le système X-Windows X11, ce qui ne lui donne pas belle allure.

C'est alors que ça se complique : pour créer un circuit imprimé à partir du schéma, il faut associer manuellement les composants avec leur boîtier, ce qu'il est possible de faire avec l'éditeur de schémas au moyen de la fonction d'édition d'attributs. C'est plus simple si l'on connaît par cœur la référence du boîtier, car on ne dispose d'aucune aide visuelle.

C'est alors seulement qu'on peut importer le schéma dans l'éditeur de circuit imprimé. Les composants commencent par atterrir sur un tas unique. On les déplace ensuite vers leur emplacement, de sorte à obtenir le CI souhaité. Si, plus tard,

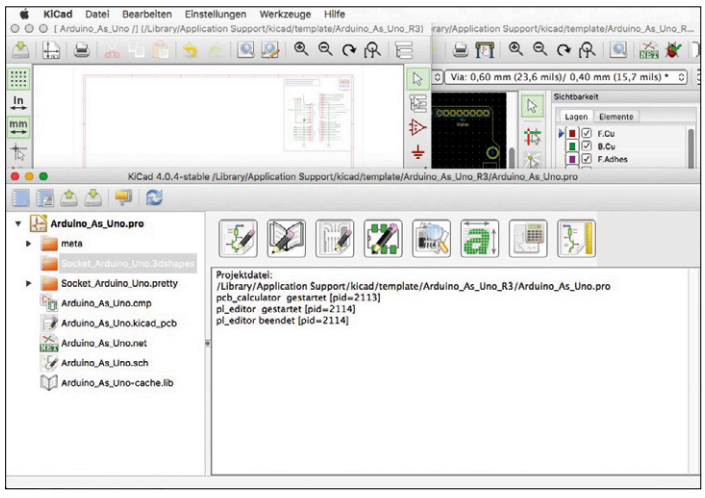


Figure 3. Tout sous un même toit : le logiciel ouvert KiCad.

on modifie dans *gschem* le boîtier d'un composant, cette modification est transmise à l'éditeur de circuit imprimé au moyen de la commande *gsch2pcb project.* On entre cette commande dans la fenêtre de terminal en remplaçant *project* par le vrai nom du projet.

KiCad

KiCad, autre suite d'outils de conception assistée par ordinateur, est un logiciel ouvert, arrivé à maturité dans sa version 4.0.4. On reconnaît le soutien de la Communauté à ce projet, non seulement à sa maintenance régulière, mais aussi parce qu'il en existe des distributions achevées, non seulement pour Windows (et peut-être aussi pour OS X), mais aussi pour pas mal de distributions Linux : Ubuntu, Debian, Mint, Arch, Fedora, open SUSE, Snappy et Gentoo. Celui qui parie sur KiCad est raisonnablement assuré de ne pas se retrouver l'année prochaine avec des fichiers illisibles ou non éditables à cause d'un logiciel non maintenu à jour.

La suite se compose d'un gestionnaire de projet *kicad*, d'un éditeur de schémas *eeschema*, d'un éditeur de circuits *pcbnew* et des outils *pcb_calculator*, *pl_editor*, *bitmap2component* et *gerbview*. Il s'agit là d'applications indépendantes dont l'homogénéité des données est assurée par le gestionnaire de projet. KiCad n'est pas seulement un logiciel mature et riche (il occupe environ 1 Go sur le disque), mais il offre aussi la possibilité de visualiser le circuit fini en 3D, car ses bibliothèques incluent les images en 3D des composants. Sur le Mac, les applications (/Programs/) et les bibliothèques ainsi que les modèles (/library/KiCad/...) sont stockés à des endroits différents, ce qui n'est pas incorrect, mais peut s'avérer malcommode.

En tant que « professionnel » parmi les logiciels ouverts de CAO, KiCad n'est sans doute pas à la pointe de l'esthétique et n'est pas aussi intégré que d'autres solutions, mais il a beaucoup à offrir, par exemple la vue en 3D déjà mentionnée. Du fait de ses nombreuses possibilités, son usage n'est guère intuitif et sa courbe d'apprentissage est pentue. Mais la documentation est disponible en plusieurs langues, ce qui facilite la tâche aux non-anglophones.

Fritzing

Fritzing est un logiciel ouvert, disponible sous Windows, OS X et Linux, en versions 32 et 64 bits, sans limitations. Sa grande particularité : c'est un logiciel intégré pour créer des circuits imprimés, avec affichage du schéma, du routage du circuit et – c'est là le clou – de la platine d'expérimentation ! On peut donc, après le dessin du schéma et avec l'aide du logiciel, commencer par tester et déboguer le circuit sur une platine d'expérimentation avant de procéder à l'élaboration d'un véritable circuit imprimé. C'est une fonction formidable, en particulier pour le monde des faiseurs.

Comme c'est l'habitude dans le monde des logiciels libres, Fritzing est téléchargeable dans une version < 1, c'est-à-dire une version bêta. La version courante, 0.9.3b, a l'air tout à fait convenable, et bénéficie d'un soutien de la Communauté. Il y a un mode d'emploi en ligne ainsi que des exemples. En particulier la représentation d'un circuit sur platine d'expérimentation est très réussie. Cerise sur le gâteau : même le programme du microcontrôleur peut être géré et écrit sous Fritzing, qui dispose de son propre éditeur de code, c'est idéal pour les faiseurs et les projets Arduino. De plus, presque toutes

les langues européennes sont supportées, en plus du chinois et du japonais.
Quand on a optimisé le circuit, qu'on l'a testé sur la platine d'expérimentation et qu'on a réalisé le routage, on peut commander directement un circuit imprimé à partir du logiciel, car derrière Fritzing, il y a Fritzing Fab, un service de fabrication de circuits imprimés. Mais à partir de 0,70 € le cm² de circuit imprimé, c'est plutôt cher : un simple *shield* Arduino revient déjà à la coquette somme de 29 €. Mais rien n'oblige à utiliser ce service. Le logiciel peut produire le circuit imprimé sous forme d'image ou de fichier PDF, SVG ou Gerber étendu et, outre la liste des composants, il peut exporter des *netlistes* XML et Spice.

CometCAD

Tout ce qui gratuit n'est pas forcément bon. CometCAD offre une combinaison simple d'éditeur de schémas et de logiciel de routage. Certes, le logiciel maîtrise les annotations (du schéma vers le CI) et peut produire des circuits imprimés utilisables, mais en pratique, il en va autrement.
Rien que par son aspect visuel, le logiciel est plutôt modeste et du côté de la maintenance ça ne semble pas aller très fort : la version courante 1.09 remonte à décembre 2015 et il manque une version pour Windows 10. Les autres systèmes d'exploitation ne sont pas supportés. La version gratuite L1 est fortement bridée : le nombre de feuilles est limité à deux par schéma avec 50 symboles au maximum (soit effectivement moins de 50 composants). Il est dommage qu'un circuit imprimé ne puisse pas excéder une taille de 102 × 102 mm et accueillir plus de 250 pastilles. La bibliothèque ne contient que 2.000 composants environ. En somme, les autres logiciels ont bien plus à offrir. Cette version gratuite n'incite pas à acheter la version L2 à 67 $ ou même la version L3 à 134 $, qui autorisent davantage de composants et des circuits de plus grande taille.

Osmond PCB

Osmond PCB appartient à la même catégorie que CometCAD. Il y a une version gratuite limitée et une version payante. Les limitations ne s'expriment pas ici d'une manière aussi flagrante : seul le nombre de pastilles est limité à 700 si l'on veut exporter le circuit imprimé sous forme de fichier (Excellon ,etc.) ou l'imprimer. Il y a deux grandes différences avec CometCAD : OsmondPCB est « Mac-Only », seulement disponible sous OS X, et il y a régulièrement des corrections de bogues. La version courante 1.1.33 est datée d'août 2016.
Voilà pour les bonnes nouvelles. Pour l'aspect esthétique, le logiciel en est resté au début du développement, à l'OS X 10.5. Il n'y a pas que l'interface utilisateur qui soit rustique : la bibliothèque de composants, organe central de la création d'un CI, est proposée avec ni plus ni moins de 130 composants.
Qu'un langage de script soit supporté ou qu'on puisse exporter tous les formats de fichiers ne change rien à l'affaire. En concurrence avec les logiciels ouverts, les petits éditeurs ont la vie dure et l'utilisateur doit se demander s'il a vraiment envie de s'en remettre à un logiciel de niche.

EasyEDA

Si vous n'aimez pas trop les logiciels de CAO volumineux des grands éditeurs qui ont pignon sur rue, ou si vous pouvez vous contenter d'un circuit imprimé réalisé à la va-vite, jetez un œil

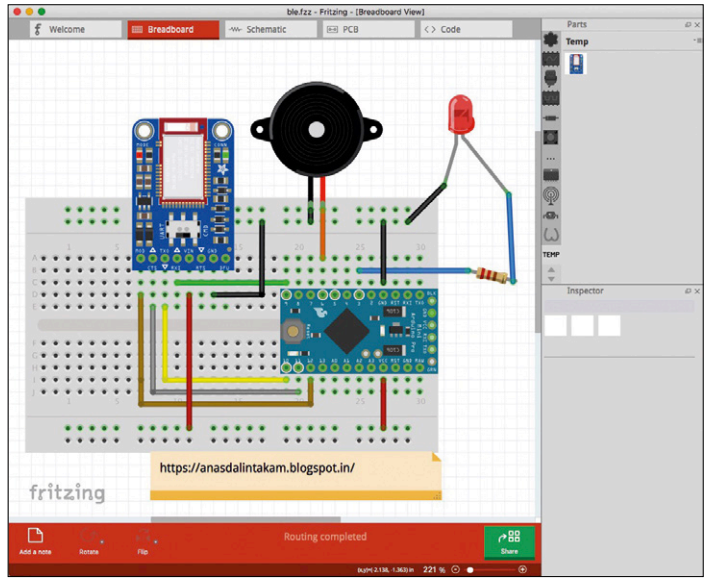


Figure 4. Coloré et pétillant pour les faiseurs : Fritzing.

à EasyEDA. Il s'agit là d'une solution complète en ligne, qui fonctionne avec presque tous les navigateurs et relève de l'informatique en nuage. Non seulement elle comprend un éditeur de schémas et une fonction de routage, mais elle permet aussi de lancer des sessions de simulation entre les phases d'élabo-

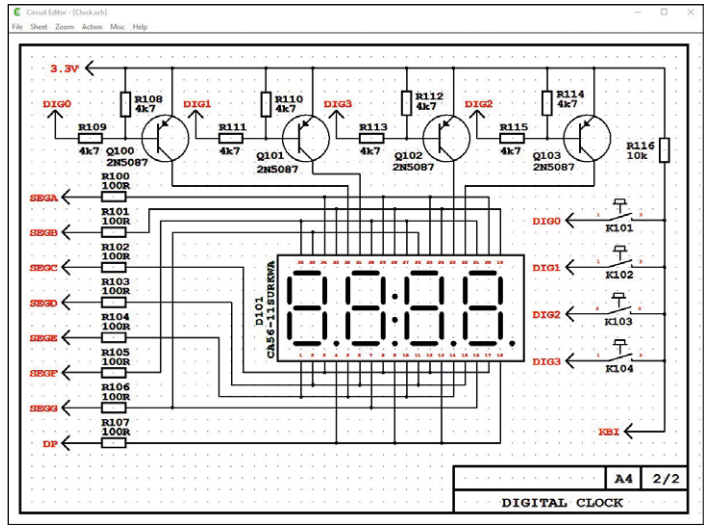


Figure 5. Ça marche mais c'est fruste et limité : CometCad.

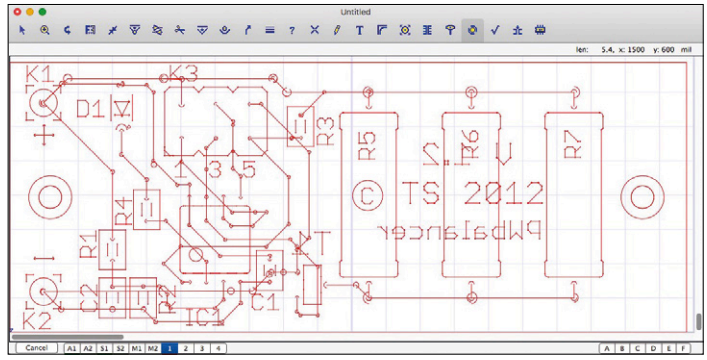
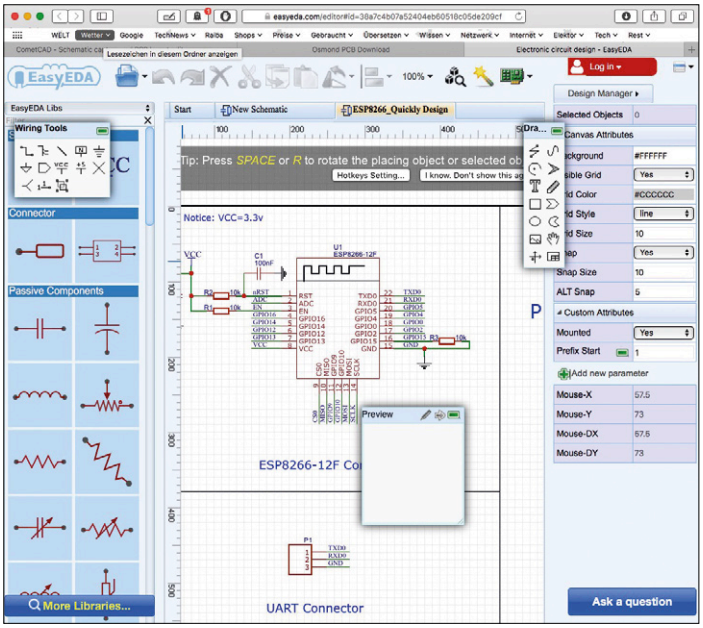


Figure 6. *Mac only* mais tout aussi fruste et limité : Osmond PCB.



ration du schéma et du circuit imprimé, du fait de la présence d'un émulateur en mode mixte. On peut se demander si la platine d'expérimentation n'est pas devenue obsolète chez les électroniciens modernes, puisqu'ils passent directement de la simulation au circuit final.
Une solution en ligne, qui tourne dans un navigateur, présente un certain nombre d'avantages : on n'a rien à installer, le logiciel est totalement indépendant du système d'exploitation et toujours à jour. De même, le partage des schémas et des circuits imprimés est simplifié. Bien entendu, il est toujours possible de sauvegarder ses projets sur son propre PC. Nous ne voyons guère d'inconvénients sérieux par rapport aux logiciels hors-ligne « normaux ». EasyEDA est facile à utiliser, puissant et moderne. Même sa bibliothèque de composants offre un bon choix pour les connecteurs et les transistors. Ça se restreint un peu pour les circuits intégrés. C'est compensé

Figure 7. Solution disponible en ligne (logée dans le nuage), EasyEDA est indépendante de la plateforme et offre une simulation intégrée.

Alternatives

À côté des logiciels ouverts et des versions gratuites (et partiellement bridées) des suites de CAO commerciales, il y a encore une troisième catégorie d'outils de CAO, ceux pris en charge par les grands distributeurs. On peut ainsi citer EAGLE de Farnell/Element14, DesignSpark de RS Components et MultiSIM BLUE de Mouser Electronics. Elektor a déjà publié des articles d'information sur tous ces logiciels. On trouvera ci-dessous une revue sommaire des fonctions de chacun d'eux.

EAGLE

EAGLE est probablement le logiciel de routage de circuit le plus connu chez les amateurs. C'est sans doute lié au fait qu'il en existe des versions gratuites qui n'ont pratiquement pour seule limitation que la taille des circuits réalisables. Il est très populaire chez les étudiants. Il se compose d'un module d'édition de schémas et d'un module de routage des circuits imprimés. Ils communiquent l'un avec l'autre par *annotation* directe avant-arrière.
Il en existe deux versions gratuites : *Educational* et *Express*. *Educational* est réservée à l'enseignement et ses limitations sont minimales : il y a un autorouteur et un schéma peut comporter jusqu'à 99 feuilles. Un circuit imprimé peut comporter

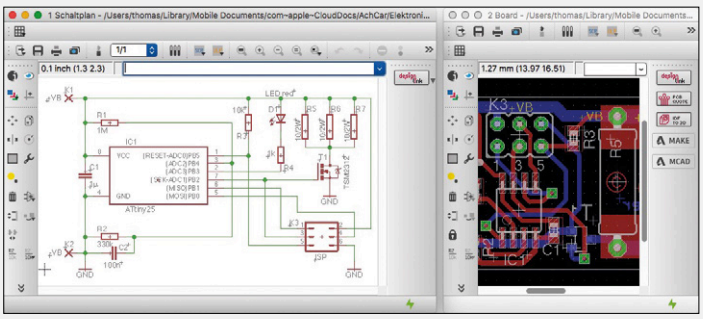


Figure 8. Tous les électroniciens reconnaissent EAGLE à ses schémas rouges et verts.

jusqu'à six couches, sa taille maximale est le format Europe (100 × 160 mm). Une seule condition : avoir une adresse courriel *.edu* valide. La version *Express* est disponible pour tous. Elle dispose aussi d'un autorouteur, mais restreint le nombre de feuilles du schéma à deux et la taille du circuit à 100 × 80 mm, la moitié de la précédente, ce qui est donc la principale limitation. EAGLE est disponible sous Windows, Linux et même OS X en versions 32 et 64 bits.
Un panneau de contrôle permet d'accéder aux différents modules : l'éditeur de schémas, le module de routage ainsi que les bibliothèques de composants. EAGLE est largement suffisant pour la plupart des applications. Chez Elektor, on peut se procurer quelques bons livres sur EAGLE qui permettent d'acquérir rapidement et facilement une prise en main complète. EAGLE a été racheté récemment par Autodesk. Le nouveau propriétaire a déjà annoncé que les versions gratuites seraient maintenues.

MultiSIM Blue

Il s'agit ici d'une version limitée de la suite de CAO professionnelle et chère MultiSIM de National Instruments. Mouser s'est associé à NI pour mettre à disposition une version gratuite. Cette version a bien entendu quelques limitations, mais elle comprend une bibliothèque de plus de 100.000 composants du catalogue de Mouser. C'est encore loin d'atteindre toute l'offre de Mouser, mais la bibliothèque est complétée petit à petit. Il est d'ailleurs possible de réunir les composants d'un circuit sur une liste et d'obtenir le prix de ces composants chez Mouser. C'est bien pratique, mais si un composant est introuvable chez Mouser, on a un problème.
Les limitations sont malheureusement sévères : au maximum six composants « maison », au total maximum 65 composants et un seul circuit par platine. Par contre la taille du circuit imprimé n'est pas limitée ;-).
La suite MultiSIM se compose des logiciels MultiSIM pour le schéma et Ultiboard pour le routage. Sa particularité par rapport aux solutions déjà vues est la possibilité – comme son nom

par le fait qu'on peut importer des fichiers de schémas et de circuits imprimés du logiciel professionnel Altium Designer ainsi que d'EAGLE. On peut aussi importer des *netlistes* de LTspice, ou encore des bibliothèques de KiCad. Derrière EasyData se dissimule une société chinoise qui propose la fabrication de circuits imprimés qu'on peut commander directement depuis le logiciel. Le coût d'un circuit imprimé de 100 × 100 mm à deux couches (17 €) paraît tout à fait raisonnable. Cette solution en ligne mérite d'être retenue.

Quoi d'autre ?

Loin d'être exhaustive, la sélection ainsi mise en lumière donne un bon aperçu de ce qu'on peut trouver comme versions gratuites allégées ou bien comme solutions complètes ouvertes, sans limitations. Sur le site Wikipedia anglais, vous trouverez un tableau comparatif des logiciels de conception assistée par ordinateur, qui inclut aussi les produits spéciaux et de niche. Vous pouvez consulter ce tableau si la liste des liens ci-dessous ne vous suffit pas.

(160176 – version française : Helmut Müller)

Liens

Pad2Pad : www.pad2pad.com/General/Software.html

gEDA : www.geda-project.org

KiCad : http://kicad-pcb.org

Fritzing : http://fritzing.org/home

CometCAD : www.cometcad.com

Osmond PCB : www.osmondpcb.com

EasyEDA : https://easyeda.com/editor

EAGLE : https://cadsoft.io

MultiSIM Blue : www.mouser.de/multisimblue

DesignSpark : www.rs-online.com/designspark

Vue d'ensemble des logiciels de CAO :
https://en.wikipedia.org/wiki/Comparison_of_EDA_software

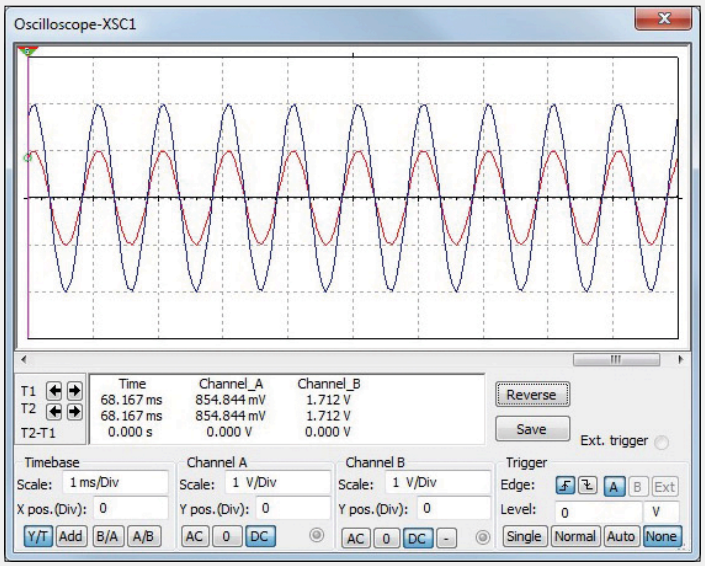


Figure 9. Simulation d'un amplificateur sous MultiSIM Blue, la solution professionnelle de National Instruments, allégée pour le distributeur Mouser.

le donne à penser – de simuler le circuit. Le fonctionnement et l'utilisation des deux programmes ne souffrent guère de critiques, mais il s'agit tout de même de deux programmes distincts. Il y a bien un bouton pour l'*annotation avant* dans MultiSIM, mais dans la version Blue, on ne sait pourquoi, il est désactivé. Le logiciel est disponible sous Windows à partir de la version XP.

DesignSpark

DesignSpark de RS Components est une version spécialement adaptée pour RS d'*Easy-PC PCB* de l'éditeur anglais *Number One Systems*. Au cours des dernières années s'y sont ajoutées deux variantes : *DesignSpark Electrical* pour le schéma des installations électriques et *DesignSpark Mechanical* comme logiciel de CAO en mécanique classique. C'est pourquoi le logiciel originel a été rebaptisé *DesignSpark PCB.*

DesignSpark PCB est moderne et fonctionne sans problème. Il inclut un contrôle des règles de conception (*Design Rule Check*) et l'*annotation avant/arrière* (*Forward/Backward Annotation*) entre le schéma et le circuit imprimé. Il dispose d'une bibliothèque conséquente de 80.000 composants et peut produire automatiquement une liste de composants. On peut vérifier la disponibilité ainsi que le prix des composants via RS.

DesignSpark peut importer des fichiers EAGLE et créer une *netliste* Spice avec laquelle on peut effectuer des simulations dans LTSpice ou TINA. À la différence de presque tous les autres concurrents, ce logiciel peut afficher une vue en 3D du circuit imprimé et de ses composants. Autant que nous le sachions, le logiciel n'est pas limité, mais la version complète n'est disponible que pour Windows 7, 8 et 10.

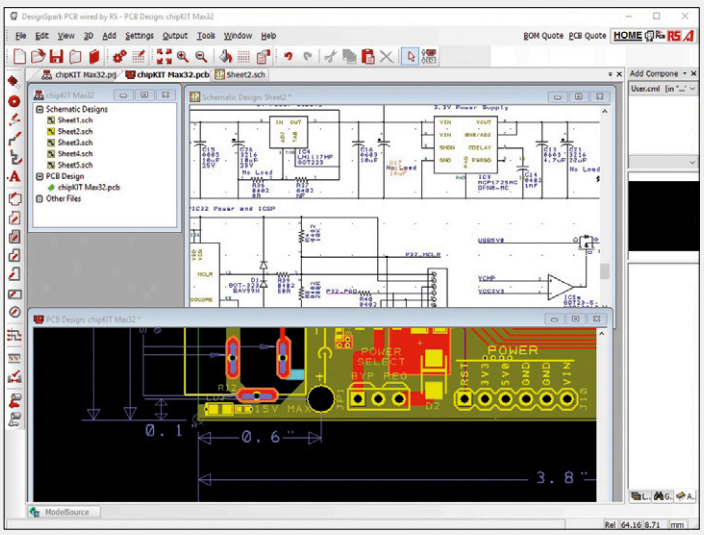


Figure 10. DesignSpark propose une représentation en 3D du circuit terminé.

LED en couleur

hier, aujourd'hui, demain

Thomas Scherer (Allemagne)

Depuis cent ans, l'homme est capable de produire de la lumière par action d'un courant électrique sur de la matière inorganique. Ces petits voyants lumineux sans filament ni remplissage gazeux ont gagné lentement en qualité et en nombre de couleurs. Ces dernières années, leur rendement a tellement augmenté que cette technologie est en train de prendre la place des tubes luminescents. Le dernier développement : les LED pourraient même remplacer les afficheurs à cristaux liquides.

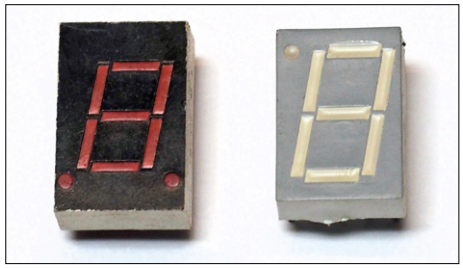


Figure 3. De ma collection personnelle d'antiquités : afficheurs à 7 segments rouges avec des caractères de 10 mm de haut.

Figure 4. Les premières calculettes pour le marché de masse étaient équipées d'afficheurs à 7 segments rouges lilliputiens (source : Erhaka, Wikimedia Commons).

Figure 5. Collection de LED. Rangée supérieure de gauche à droite : LED rouge, jaune et vertes (quatre modèles) des années 1970, comparées à une LED bleue moderne, si lumineuse qu'elle sature le capteur de l'appareil-photo. Rangée inférieure : LED vertes modernes de 10 et 20 mm de diamètre, RVB, blanche et infrarouge.

Après l'invention du détecteur à galène par Ferdinand Braun en 1874, il fallut attendre 33 ans avant que Henry Joseph Round découvre qu'on pouvait observer au point de contact de la pointe métallique sur le cristal une lueur jaune, rouge ou même bleue, en fonction de la tension appliquée. Round fut donc le véritable inventeur de la LED et, en l'honneur de sa découverte, l'électroluminescence des semi-conducteurs est aussi appelée effet Round. Ce fut le début.

Une LEDvolution lente

La première vraie LED fut fabriquée par RCA et produisait une lumière invisible, car infrarouge. C'était en 1955 et son concepteur s'appelait Rubin Braunstein. Le mérite d'avoir lancé la première production en série en 1962 revient à Texas Instruments. La LED infrarouge de TI avait quand même déjà un rendement de 1,1 % ! La première LED rouge visible s'alluma aussi en 1962, chez General Electric. Après une pause étonnamment longue de dix ans, ce fut le tour de la première LED jaune, également chez GE. Les LED rouges s'améliorèrent rapidement dans les années 1970. Avec leur rendement de 5,5 %, on les utilisa partout comme voyants témoins. Les modèles en boîtier plastique de 5 ou 3 mm de diamètre ne consommaient qu'une puissance de 40 mW et présentaient beaucoup d'avantages : ils ne craignaient pas les chocs comme les ampoules à filament de tungstène et duraient nettement plus longtemps. Et les néons, avec leur tension d'alimentation de 70 à 100 V, appartenaient plutôt à l'époque tout juste révolue des tubes. Les LED, avec leur tension de service de 1,2 à 4 V (en fonction de leur couleur, voir le tableau **tension & couleur**) étaient manifestement plus « compatibles avec les transistors ».

Dans les années 1970, il y avait aussi des afficheurs à technologie LED sous la forme d'éléments rouges à sept segments (**fig. 3**). Si vous vous souvenez des premières calculettes, elles avaient des afficheurs rouges minuscules (**fig. 4**) qui en général consommaient plus d'énergie que la puce de calcul. Ma TI-59 de 1977 – l'une des premières calculatrices programmables – était puissante, mais elle avait un de ces afficheurs lilliputiens. Malgré cela, l'écolier que j'étais a dû mettre de l'argent de côté pendant une longue période avant de pouvoir se l'acheter ;-).

Dans les années 1970, avec les LED rouges, jaunes et vertes, il y avait donc déjà trois couleurs disponibles. Dans la rangée supérieure de la **figure 5**, on voit (à l'exception de la LED bleue à droite) les LED originales « antiques » des années 70. Elles n'étaient souvent pas très lumineuses, les tons rouges et jaunes étaient changeants et le vert n'était pas nécessairement vert. La **figure 6** montre les différences effectives de luminosité

Tension et couleur

Les LED inorganiques sont à base de semi-conducteurs dopés généralement avec les éléments suivants : aluminium, arsenic, gallium, indium, phosphore et azote. Depuis peu, il y a des recherches sur les LED au carbone (notamment au diamant), au silicium et au zinc. La composition, la concentration et la structure n'influent pas seulement sur le rendement, mais surtout sur la couleur de la lumière émise. Car l'énergie des photons émis dépend directement de la largeur de la bande dite interdite, à savoir la différence énergétique entre la bande de conduction et la bande de valence du semi-conducteur utilisé. Plus la bande interdite est large, plus l'énergie du photon est élevée et plus la longueur d'onde de la lumière émise est petite. La bande interdite a une grande influence sur la tension de seuil qu'il faut franchir pour a) qu'un courant circule et b) que les électrons passant de la bande de conduction à la bande de valence émettent un photon. La **figure 1** illustre cette relation. Il n'est donc pas étonnant que la tension de fonctionnement d'une LED bleue soit à peu près le double de celle d'une LED rouge, la fréquence de la lumière étant proportionnelle à l'énergie du photon.

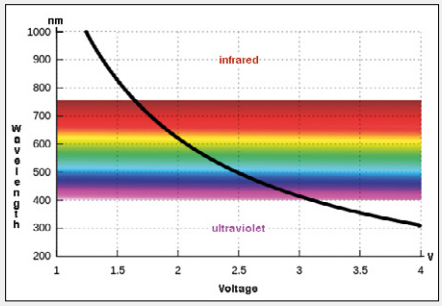


Figure 1. Rapport entre la tension de service (bande interdite) et la couleur des LED. La tension et la longueur d'onde sont inversement proportionnelles.

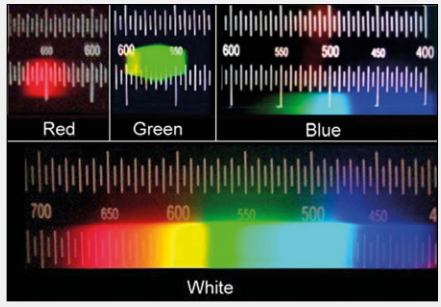


Figure 2. Spectres de LED de différentes couleurs. Plus la longueur d'onde est courte, plus le spectre est large. En dessous, à titre de comparaison, le spectre du soleil (source : Wikimedia Commons).

Le **tableau 1** donne le type de semi-conducteur utilisé pour un domaine de fréquences donné et la tension de fonctionnement correspondante. Toutefois, la tension mesurée aux bornes de la LED dépend bien entendu de la bande interdite, mais également, à côté d'autres facteurs, des résistances des matériaux et des connexions ainsi que des transitions d'un matériau à l'autre. Quand le courant augmente, la tension aux bornes de la LED augmente donc aussi. L'absence de passage brutal du blocage à la conduction du courant est due au fait que, dans la zone de transition, la probabilité pour qu'un électron franchisse la bande interdite augmente avec la tension. Par ailleurs, le spectre des LED se déplace légèrement avec la température vers les plus grandes longueurs d'onde, donc vers des couleurs plus chaudes, ce qui n'est pas étonnant, car la tension de fonctionnement baisse alors. Si pour une application particulière, on a besoin d'une couleur stable, il faut contrôler la température de la puce de la LED.

La pureté de la lumière de la LED se situe quelque part entre celle de la lampe à incandescence (colorée) et celle du laser. Autrement dit, la bande d'émission des LED n'est pas aussi étroite celle des lasers, mais elle l'est bien plus que celle des lampes à incandescence. La **figure 2** montre qu'en particulier dans le domaine des plus grandes longueurs d'onde les LED ont une bande assez étroite alors que celle des LED bleues est relativement plus large, ce qui est un avantage pour les LED blanches formées d'une LED bleue avec une couche de phosphore. Un spectre plus uniforme donne un effet plus « naturel », c'est-à-dire proche de celui du soleil. La qualité de la lumière d'une LED blanche est donnée par l'index de reproduction des couleurs Ra, avec lequel le soleil a la valeur de référence 100 et les LED blanches des valeurs de 75 à 95.

Tableau 1

Couleur	Longueur d'onde (nm)	Semi-conducteur		Tension (v)
infrarouge	>760	GaAs	(arséniure de gallium)	<1,6
		AlGaAs	(arséniure de gallium-aluminium)	
rouge	610 – 760	AlGaAs	(arséniure de gallium-aluminium)	1,6 – 1,9
		GaAsP	(phosphure d'arsenic-aluminium)	
		AlGaInP	(phosphure de gallium-aluminium-indium)	
		GaP	(phosphure de gallium)	
orange	590 – 610	GaAsP	(phosphure de gallium-arsenic)	1,8 – 2,2
		AlGaInP	(phosphure de gallium-aluminium-indium)	
		GaP	(phosphure de gallium)	
jaune	570 – 590	GaAsP	(phosphure de gallium-arsenic)	2,0 – 2,4
		AlGaInP	(phosphure de gallium-aluminium-indium)	
		GaP	(phosphure de gallium)	
vert	500 – 570	InGaN	(nitrure de gallium-indium)	2,2 – 2,7
		GaN	(nitrure de gallium)	
		GaP	(phosphure de gallium)	
		AlGaInP	(phosphure de gallium-aluminium-indium)	
		AlGaP	(phosphure de gallium-aluminium)	
bleu	450 – 500	ZnSe	(séléniure de zinc)	2,6 – 3,3
		InGaN	(nitrure d'indium-gallium)	
		SiC	(carbure de silicium)	
violet	400 – 450	InGaN	(nitrure de gallium-indium)	3,2 – 3,6
ultraviolet	230 – 400	AlN	(nitrure d'aluminium)	3,5 – 4,2
		AlGaN	(nitrure de gallium-aluminium)	
		AlGaInN	(nitrure de gallium-aluminium-indium)	

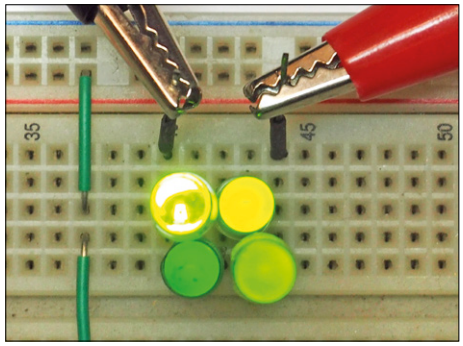


Figure 6. Quatre LED vertes différentes : chacune fournit une lumière différente. La LED en haut à droite est un modèle moderne qui sature le capteur de l'appareil-photo, les trois autres LED de 5 mm sont celles de la figure 5.

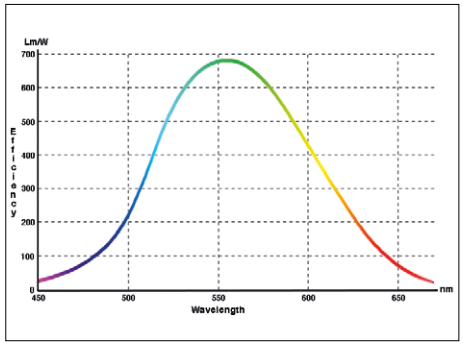


Figure 7. Rendement théorique maximal des LED en fonction de la couleur. Attention : l'unité *lumen* tient compte de la sensibilité spectrale de l'œil humain, plus élevée dans le vert.

Figure 8. Lampe bougie à LED. Dans cette lampe au culot E14, le filament a été remplacé par des rubans de LED (source : OSRAM).

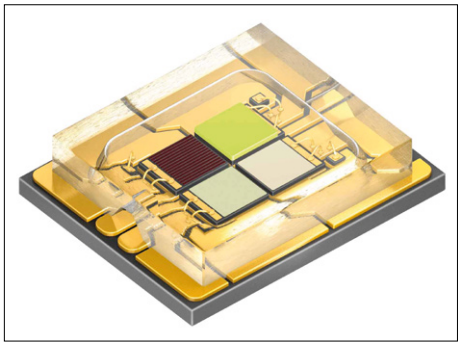


Figure 9. LED *RVB plus* : dans le boîtier CMS de l'OSTAR-LED, OSRAM a intégré, outre les couleurs primaires, une puce de LED blanche.

et de vert à cette époque. Les quatre LED sont montées en série et traversées par un courant de 5 mA. Seule la LED de gauche est un modèle récent à haut rendement, à titre de comparaison. Elle est en réalité d'un vert pur, mais si lumineuse qu'elle sature le capteur de l'appareil-photo. L'une des anciennes LED est plutôt jaune-verte que verte. Il en allait de même pour les LED rouges et jaunes. Mais les LED actuelles en boîtier de 5 mm ou CMS ont des couleurs tellement pures que c'en est un plaisir. Utilisées comme témoins, il faut bien doser leur courant, car elles sont si lumineuses qu'on est facilement ébloui.

Statut quo

Cela a duré plus longtemps avant que le bleu vienne s'ajouter à la collection : ce n'est que dans les années 1990 que Nichia mit sur le marché des LED bleues. La LED bleue fut saluée par un prix Nobel, attribué en 2014 à ses créateurs Shuji Nakamura, Samu Akasaki et Hiroshi Amano, un grand honneur pour un petit composant. Ce prix était justifié, car les LED bleues à haut rendement forment la base des LED blanches, pour lesquelles on a simplement ajouté une couche de phosphore qui convertit une partie de la lumière bleue à haute fréquence en lumière jaune, laquelle, par mélange additif de couleurs avec la lumière bleue restante, produit de la lumière blanche. Ce procédé a été développé en 1995 par Jürgen Schneider à l'institut Fraunhofer et constitue aujourd'hui la base de presque toutes les LED utilisées pour l'éclairage. La part de marché des LED blanches dépasse aujourd'hui légèrement celle des LED de couleur. Pour en savoir plus sur les LED blanches pour l'éclairage, on peut lire mon article « ainsi soient les LED ! » (Elektor, 01/2016, voir [1]).

Sur la figure 5, en haut à droite, on voit une LED bleue en boîtier plastique transparent, alimentée avec un courant de 5 mA. Elle est tellement lumineuse qu'elle sature le capteur de l'appareil-photo et prend une couleur bleue-blanche trompeuse. En bas à gauche, se trouvent deux LED vertes de calibres 10 et 20 mm. L'exemplaire le plus large est une LED multipuce avec quatre puces connectées en série. À côté, sur la droite, on voit une LED tricolore RVB à anode commune, d'où ses quatre pattes. Une commande appropriée des trois puces de couleur primaire permet d'obtenir pratiquement n'importe quelle couleur par mélange additif. La LED transparente à côté à droite est un exemplaire blanc moderne. Enfin, tout à fait à droite se trouve une LED infrarouge. La couleur de son boîtier laisse passer la lumière infrarouge sans problème.

Les faibles rendements des années 1970 augmentèrent rapidement au début de ce millénaire pour atteindre des valeurs jusqu'à 100 lm/W. Les LED blanches se retrouvèrent à un niveau qui leur permettait de concurrencer les tubes luminescents, car, à rendement comparable, elles n'avaient pas de problème de mercure et duraient plus longtemps. Entretemps, en laboratoire, les LED blanches atteignent un rendement de plus de 300 lm/W à quelques pour cent de la limite théorique (350 lm/W à 6600 K). La **figure 7** donne le rendement maximal théorique en fonction de la couleur. Elle montre un intéressant paradoxe apparent : même si elles sont basées sur des LED bleues qui ne dépassent pas 200 lm/W, les LED blanches font mieux. Ceci est dû au fait que l'unité *lumen* tient compte de la sensibilité spectrale de l'œil humain, pour lequel, à puissance égale, la lumière verte paraît bien plus lumineuse. Bien que, dans la transformation de lumière bleue en lumière jaune, une partie de l'énergie soit perdue sous forme de chaleur, la LED blanche nous paraît plus lumineuse que la puce bleue sous la couche de phosphore. Encore une chose intéressante : comme les LED ont pratiquement atteint leurs limites, du moins en laboratoire, le rendement ne pourra plus croître que de façon marginale. Le progrès technique ne pourra plus s'appliquer qu'au produit fini. La production des LED en masse a

encore un espace de progrès devant elle avant que leur rendement atteigne les valeurs fantastiques de laboratoire à un prix raisonnable. Ce processus devrait avoir lieu en moins d'une décennie, alors les semi-conducteurs électroluminescents auront mis à la disposition de l'humanité un mode d'éclairage particulièrement économique, avec une empreinte écologique limitée grâce à leur longue durée de vie.

LED de couleur

Au cours des dernières années, le rendement des LED a si fortement augmenté et leur prix a tellement réduit grâce à la production de masse que les lampes à base de LED se sont imposées à une large échelle. Il y a les lampes à LED à l'aspect parfois peu engageant, qui contiennent plusieurs puces CMS, mais également les lampes en forme de bougie, avec des LED montées en filaments (**fig. 8**), ce qui permet de conserver le style authentique des lustres, tout en faisant des économies d'énergie. Aujourd'hui il existe des lampes de toutes les formes – et de toutes les couleurs. Passons-les en revue.

Les LED infrarouges sont utilisées depuis des décennies dans les télécommandes ou de nos jours dans l'éclairage nocturne pour les caméras de surveillance, ou dans des applications militaires. On les trouve maintenant dans des lampes infrarouges qui contiennent des centaines de LED CMS et consomment plus de 100 W ; elles servent dans l'industrie d'éléments de base pour des projecteurs de quadrillages lumineux sur des sections de chaînes de fabrication ou de montage. Cela permet à des robots équipés de caméras infrarouges de voir ces lignes pour non seulement bien s'orienter dans l'espace, mais aussi évaluer avec une meilleure précision la position des pièces sur la chaîne. Les hommes qui surveillent ou coopèrent avec ces robots ne sont pas perturbés par cette lumière infrarouge.

Il n'y a pas que les LED blanches qui permettent de produire de la lumière blanche. C'est possible (mais plus cher) avec des LED RVB qui contiennent au moins trois puces de couleurs primaires dans le même boîtier. En les alimentant avec des courants différents, on peut obtenir pratiquement n'importe quel mélange de couleurs. La **figure 9** montre par ex. une OSTAR-SMD-LED d'OSRAM qui comprend non seulement des puces rouge (625 nm), verte (530 nm) et bleue (453 nm), mais aussi une blanche. Elle est utilisée pour des éclairages spéciaux comme celui d'un champ opératoire chirurgical où la fidélité des couleurs est cruciale. Malgré sa taille minuscule, cette LED fournit plus de 500 lm avec un courant nominal de 1,4 A.

Les amateurs de danse auront déjà remarqué l'invasion des LED dans les boîtes de nuit. Les projecteurs de scène ou « PAR cans » (réflecteurs paraboliques aluminés) avec filtres de couleur étaient jadis équipés de lampes à incandescence de plusieurs centaines de watts. Aujourd'hui ils sont pour beaucoup équipés de LED et entièrement électroniques (**fig. 10**). Même si les versions à LED n'ont pas encore entièrement rattrapé l'ancienne technologie, la différence s'amenuise grâce aux progrès des LED en luminosité. La durée de vie, la robustesse, le faible dégagement de chaleur, le risque limité de blessure (par explosion du projecteur) ainsi que la commande électronique sans inertie sont indiscutablement en faveur de la version LED.

Mais l'offre de LED de couleurs ne se contente pas du domaine de la musique. C'est Philips qui a détecté le premier l'intérêt du grand public pour l'éclairage d'ambiance. Avec des éclairages de table ou d'appoint, le géant néerlandais de l'électronique a mis sur le marché sous la désignation « LivingColors » toute une série de luminaires télécommandés dont la couleur est réglable dans une plage étendue. À l'intérieur se dissimulent des LED blanches et de couleur qui sont non seulement miscibles, mais peuvent clignoter ou donner une lumière atténuée. Le succès de ces produits a incité Philips à sortir la « Hue serie » (**fig. 12**), qui inclut non seulement des lampes, mais des rubans de LED et des composants d'éclairage couleur « normaux » à culots E27, le tout est commandé à distance par un « bridge » (pont). Les modèles « Hue Go » contiennent une batterie au lithium. Philips sortira sûrement d'autres produits à effets lumineux puisqu'ils sont très lucratifs. Si vous ne voulez pas trop dépenser, vous pouvez quand même trouver votre bonheur. Les magasins de bricolage s'y sont mis et proposent des rayons entiers de toutes sortes de systèmes d'éclairages à LED colorés (**fig. 13**) pour une fraction du prix des produits de marque. Encore moins cher : cherchez sur eBay ou achetez sur Alibaba directement en Chine. Avec ces deux fournisseurs, il faut non seulement

Figure 10. Projecteur de scène à effets du type CLP56RGB05PS de Cameo avec 151 LED de couleur et une puissance de 30 W sous 220 V. Ce projecteur est équipé d'une interface DMX pour la télécommande ainsi que d'un microphone intégré pour permettre des effets d'orgue lumineux.

Figure 11. *LivingColors* de Philips : un modèle à succès avec des LED de couleur et blanches dans un globe en plastique.

Figure 12. La *Hue Serie* de Philips comprend des lampes de couleur à LED (culot E27) et la télécommande *bridge*.

Figure 13. Tous les magasins de bricolage proposent aujourd'hui un large choix de rubans à LED, luminaires et lampes à effets, avec ou sans télécommande.

Figure 14. *Google Nexus 6p* : l'afficheur OLED de cet ordiphone bon marché compte 2560×1440 pixels.

s'attendre à de possibles problèmes de douane, mais aussi à ce que bon marché ne soit pas toujours synonyme de bonnes affaires (voir encadré « **Attention à la qualité** »).
Bien entendu, même les LED ultraviolettes ont leur utilité. Outre les usages spéciaux, on peut s'en servir pour insoler le vernis photosensible des circuits imprimés ou construire des lampes à lumière noire. Une autre application de ces LED est l'éclairage d'un tuyau transparent traversé par de l'eau pour freiner le développement des algues dans les aquariums et les piscines. Elles permettent également de durcir des vernis ou de tester la sensibilité aux UV de la surface de produits.

OLED

Les OLED (diodes électroluminescentes organiques) furent « inventées » bien plus tard et, du point de vue du rendement et de la stabilité à long terme, ne sont pas encore au niveau des LED à semi-conducteurs inorganiques, même si elles sont en train de rattraper leur retard. Leurs avantages sont des matériaux bon marché et une technique de fabrication qui ne nécessite pas de salles blanches. De plus, on peut leur donner une grande surface, ce qui a l'avantage, pour certains éclairages, d'adoucir les ombres. Même les supports flexibles ne posent pas de problème de principe. Il a longtemps été difficile de réaliser des produits OLED en dehors des laboratoires. Ces temps sont en train de changer.
Avec les OLED on peut par ex. réaliser facilement des afficheurs, car les techniques d'impression permettent de traiter les pixels lumineux ainsi que le câblage en une seule opération économique, produisant ainsi des afficheurs RVB lumineux et à large contraste. Avec des semi-conducteurs inorganiques, une telle entreprise serait beaucoup plus compliquée et sensiblement plus onéreuse. Les afficheurs OLED concurrencent de plus en plus les afficheurs à cristaux liquides (LCD). En principe, ils battent ces derniers sur tous les plans : le rétroéclairage (avec tous ses problèmes de répartition uniforme de la lumière) n'est pas nécessaire, puisque les OLED sont elles-mêmes lumineuses. Même la consommation électrique est plus faible, le rétroéclairage des LCD est en grande partie absorbé par la répartition de la lumière et les filtres polarisants. Avec les OLED, les variations de couleur selon l'angle de vue appar-

Attention à la qualité !

Les lampes à LED qui contiennent un grand nombre de LED individuelles pour obtenir suffisamment de lumière existent depuis un bon moment. Dès le milieu des années 1990, les constructeurs d'automobiles ont réalisé le troisième feu de freinage avec une série de LED rouges, même sur les voitures de milieu de gamme. Elles étaient déjà assez lumineuses et bon marché pour cela. L'automobiliste est content d'avoir à changer moins d'ampoules pendant la durée de vie de sa voiture. De plus un feu de freinage est un organe de sécurité qui ne doit pas tomber en panne après quelques centaines d'heures de fonctionnement. Tout cela semble logique, mais il en va autrement dans la réalité.

Car il faut prendre en compte deux phénomènes dont la combinaison rend la réalité beaucoup moins belle. Les LED intégrées aux feux de freinage ont une durée de vie « théoriquement » énorme, jusqu'à 50.000 h. Mais en voiture, ce n'est pas garanti : non seulement à cause des vibrations et de l'humidité lorsqu'il fait mauvais temps (risque de corrosion des pistes conductrices, etc.), mais aussi du fait des variations de température considérables qui entraînent des contraintes sur le boîtier, la puce et les connexions. De ce fait, une durée de vie de 50.000 h est une vue de l'esprit. Le fort échauffement de la tôle soumise aux rayons du soleil ajouté au dégagement de chaleur en cours du fonctionnement fait grimper le thermomètre. Bien que la durée de vie des LED en voiture soit encore supérieure à celle des lampes à incandescence, la différence n'est donc pas si grande. Circonstance aggravante : l'effet grand nombre de LED. Même dans les meilleures conditions, et pour de simples raisons statistiques, dans le cas de LED multiples, la durée pendant laquelle toutes les LED sont en état de fonctionner diminue. Une évaluation approximative montre que le temps de vie global est réduit d'un facteur $n^{1/2}$; on passe donc pour 10 LED, de 50.000 h par LED à 15.000 h pour l'ensemble. Si l'on prend pour une LED une durée de vie réelle de 10.000 h, il faut compter, pour l'ensemble de 10 LED sur la défaillance d'une LED au bout de 3.000 h. C'est gênant parce que, dans une voiture alimentée sous 12 V, on met deux ou trois LED en série, donc la défaillance de l'une entraîne l'extinction des autres. En outre dans les lampes d'automobile, les LED sont enfermées hermétiquement et donc impossibles à remplacer une par une. Au lieu de changer une ampoule à 2 €, il faudra mettre la main à la poche beaucoup plus profondément pour remplacer un composant beaucoup plus cher.
J'en ai fait l'amère expérience : sur une Fiat de 1996, le feu de freinage a lâché de la manière décrite au bout de trois ans. La facture pièces et main-d'œuvre était à l'époque de plus de 100 DM (env. 51 €). À celui qui dirait : « pas étonnant, c'est une Fiat ! », je réponds que la fiabilité légendaire de Toyota a aussi été mise à mal par une simple LED. Dans ce cas, ce fut le feu arrière gauche de ma Prius, constitué de nombreuses LED intégrées. Après 11 ans de service, il a fallu le changer en même temps que les disques de freins. La pièce de rechange m'a tout de même coûté 73 €. Dans les véhicules les plus récents, si des LED des phares lâchent, il faudra certainement poser plusieurs centaines d'euros sur la table. La technique moderne a un prix ;-)
Les LED en nombre deviennent un gros problème si l'on a de nombreux éclairages de ce type à la maison. L'humidité et les variations de température n'ont dans ce cas que des effets mineurs. Le problème vient alors de la qualité des composants. Les

tiennent au passé. Enfin, le contraste des afficheurs à OLED est inégalable, car lorsqu'une LED est éteinte, elle est vraiment éteinte. Le noir est bien noir.
Avec toutes ces superbes caractéristiques, on s'étonne que les afficheurs à OLED ne se soient pas encore imposés. La raison se trouve dans les tolérances extrêmes de fabrication : les nombreuses LED d'un afficheur doivent avoir une luminosité très précisément identique et qui le reste dans le temps. Il en résulte qu'il y a deux ans, les téléviseurs même de petite taille d'écran étaient hors de prix. Mais dès la fin 2016, LG, le chef de file du marché des téléviseurs OLED, proposait des appareils, disons acceptables, de déjà 55 pouces de diagonale pour moins de 1500 €. Les ordiphones à écran OLED sont apparus plus tôt ; la différence de prix avec la version LCD n'est pas considérable pour ces petits écrans, même si, comme dans le cas de l'ordiphone Google Nexus 6p (**fig. 14**, fabriqué par Huawei) de 2560×1440 pixels, on arrive tout de même à une intégration de 11 millions d'OLED. Alors pourquoi n'y a-t-il pas encore d'iPhone à écran OLED ? C'est parce que les capacités mondiales de production de l'année 2016 n'y auraient pas suffi. Mais elles augmentent constamment et, selon certains bruits de couloir, fin 2017 sortirait un iPhone 8 équipé d'un écran OLED, du moins les modèles haut de gamme, car pour plus de 100 millions d'afficheurs, la production n'est pas encore suffisante.

Le futur

Outre l'iPhone 8, les écrans à OLED vont rapidement se généraliser à tous les appareils qui ont quelque chose à afficher. Il n'est sans doute pas exagéré d'affirmer que bientôt toutes les autres techniques d'affichage auront fait leur temps. Les OLED vont également s'imposer dans les techniques d'éclairage. Quand ça arrivera, nombreux seront les fabricants de luminaires qui s'y convertiront. Mais les produits à culots standard resteront du domaine des LED inorganiques, puisque les OLED ne sont pas capables de fournir suffisamment de lumière sur une petite surface dans un avenir prévisible. À l'exception de quelques niches aux besoins particuliers, les LED inorganiques vont éliminer pratiquement toutes les autres techniques d'éclairage, non seulement parce qu'elles sont moins énergivores, mais aussi parce que les coûts d'exploitation, en particulier les coûts de maintenance, sont nettement plus bas qu'avec les techniques traditionnelles (remplacement moins fréquent des sources de lumière). L'éclairage public n'est que l'une des applications où les LED seront bientôt incontournables. Avec des LED, on peut réaliser des phares d'automobiles intelligents qui modifient le cône de lumière pour ne pas éblouir les voitures qui arrivent en face. On n'aurait plus besoin de commuter entre feux de route et feux de croisement. Le rêve... ⏮

(160246 – version française : Helmut Müller)

Lien

[1] ainsi soient les LED !, Elektor 01/2016 : www.elektormagazine.fr/150577

guirlandes lumineuses d'Extrême-Orient pour les décorations de Noël ou simplement pour le plaisir sont fantastiques. Pour beaucoup de produits, le prix est à peine croyable : la guirlande entière coûte souvent moins que l'ensemble des LED utilisées. « Mais comment font-ils ? » C'est simple : ils utilisent des composants de moindre qualité. Dès que l'on touche à des guirlandes de LED, on a droit à des pannes (partielles). Je ne fais pas exception à la règle ;-).
Les rubans de LED autocollants à bas prix, qu'on utilise pour l'éclairage indirect à basse tension de meubles ou autres, présentent le même problème. Chez moi, ce fut le cas avec deux profilés en aluminium de deux mètres avec des rubans de 60 LED collés à l'intérieur. Au bout de six mois et peut-être 600 h de fonctionnement, huit LED avaient déjà rendu l'âme. D'après la formule ci-dessus, la première panne aurait dû se produire au bout de 4.500 h environ. Après avoir dessoudé quelques LED CMS d'un ruban de réserve pour remplacer les LED en panne, deux autres moururent au bout de deux mois (**fig. 15**). Une histoire sans fin...
Même les sources « normales » d'éclairage à LED ne durent pas aussi longtemps qu'elles le devraient. Les lampes à LED n'existent pas depuis bien longtemps, et pourtant chez moi il y a déjà quatre exemplaires avec culot E27 ainsi que deux spots à embase GU10 qui sont tombés en panne. Sur les premières, c'est l'électronique de commande qui a lâché, alors que sur les seconds, les LED n'ont pas résisté à l'élévation de température. Sur deux tubes à LED achetés pour remplacer des tubes luminescents, les ingénieurs de LG ont sous-dimensionné les fusibles de protection de l'électronique de commande. Une fois équipés de fusibles plus puissants, les tubes fonctionnent encore aujourd'hui. Ma lampe de fabrication maison construite il y a plus de 10 ans (**fig. 16**) fonctionne toujours sans broncher. Il faut dire que j'ai utilisé 13 LED de bonne qualité.

Figure 15. Ruban de LED avec une panne. Je ne suis pas totalement innocent dans la mort de la LED L30 qui, trop en hauteur, était mal refroidie. Au lieu d'être coupée, elle est en court-circuit. Mais ses prédécesseures n'avaient pas de raison de décès prématuré.

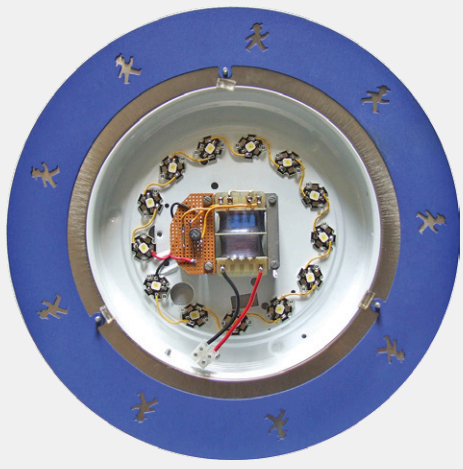


Figure 16. La première lampe à LED de la maison Scherer en 2005 était une réalisation personnelle. Et elle est toujours fidèle au poste, après bien 10.000 h de marche.

electronica Fast

Qui a gagné 75.000 € et un stand

La FFA, « electronica Fast Forward Award, the Start-up Platform powered by Elektor » selon son appellation officielle, a été une réussite à tout point de vue non seulement pour les gagnants, mais aussi pour tous les participants. Au cours du salon *electronica 2016* de Munich (Allemagne), 35 participants venus de 16 pays différents ont présenté leur projet devant un jury d'experts de STMicroelectronics, Conrad, Würth Elektronik et bien sûr Elektor. Pendant trois jours, les présentations, toutes de grande qualité, se sont succédé. Le jury a eu fort à faire pour départager les candidats des trois catégories : idée, prototype et *start-up*. Le dernier jour du salon, les gagnants de chacune de ces catégories ont concouru pour le grand prix, d'une valeur de 75.000 € à dépenser en communication et promotion (*RP*), auquel s'ajoute un stand gratuit pour *electronica 2018*.

Le concours sponsorisé par STMicroelectronics (*Platinum*), Conrad (*Gold*), Würth Elektronik (*Gold*) et Trinamic (*Bronze*), a été passionnant avec des projets qui paraissaient

Les résultats

Idées

1. Artem Kuchukov (Allemagne) – **Kewazo**, robot d'assemblage d'échafaudage
2. Jonas Galle (Belgique) – **Valcun**, imprimante 3D bon marché pour le métal
3. Michael & Andrey Shustov (Russie) – **Baristor**, une barrière résistive

Prototypes

1. Till Nauman & Lara Obst (Allemagne) – **Mowea**, appareils modulaires de production éolienne pour zones hors réseau
2. David Link & Christian Kind (Allemagne) – **nevisQ**, détection et prévention intelligente des chutes
3. Peter Wasilewsky (Pologne) – **nWatch**, plateforme de développement de microcontrôleurs vestimentaires

Start-ups

1. JF Brandon (États-Unis) – **BotFactory Squink**, imprimante 3D pour circuits imprimés
2. André Kholodov (Allemagne) – **eCozy**, thermostat intelligent connecté
3. Milan Simek (République Tchèque) – **Sewio**, système précis de suivi et de supervision d'objets

Les gagnants

1. Budget RP de 75.000 € + stand à electronica 2018 : **Mowea**, appareils modulaires de production éolienne pour zones hors réseau
2. Budget *RP* de 50.000 € : **BotFactory Squink**, imprimante 3D pour circuits imprimés
3. Budget *RP* de 25.000 € : **Kewazo**, robot d'assemblage d'échafaudage

Prix spécial « Tech for Good »

4. Budget *RP* de 5.000 € : Len Williams (Australie) – **Every Drop Counts** (chaque goutte compte), système de surveillance de consommation d'eau

Forward Award 2016

pour le salon *electronica 2018* ?

inimaginables : sécurité de l'IdO, système de veille pour les personnes âgées et handicapées, imprimantes 3D à la pointe de la technologie, éclairages RVB contrôlés par des plantes, systèmes audio, instruments de mesure de précision ou encore des robots qui construisent des échafaudages. Les participants sont venus de loin : Inde, Russie centrale, États-Unis et même Australie, mais aussi de nombreux pays voisins d'Europe ; ils étaient tous prêts à passer une semaine à Munich pour défendre leur projet. Nous avons été honorés et fiers d'avoir eu la chance de rencontrer ces gens inspirés et de découvrir leur travail.

Comme il s'agit d'un concours, tout le monde ne pouvait prétendre au Grand Prix, mais aucun n'est reparti les mains vides. Bien que certains aient gagné plus que d'autres, tous sont maintenant membres à vie de la communauté *Elektor Hero* et nous espérons entendre parler d'eux et de l'avancement de leur projet. La prochaine fois ce sera peut-être vous !

(160278 – version française : Yves Georges)

Le projet ***Mowea*** a été choisi à l'unanimité comme le grand gagnant du concours *Fast Forward Award 2016*. Till Nauman, le concepteur (second à partir de la droite), a reçu le Grand Prix des mains de Falk Senger, directeur de la foire de Munich (à droite), assisté par nos sponsors (de gauche à droite) Shawn Silberhorn (Conrad), Alexander Gerfer (Würth Elektronik) et Jacky Perdrigeat (STMicroelectronics).

JF Brandon de BotFactory est venu de Long Island City (États-Unis) pour présenter la ***Squink***, *son imprimante de bureau de circuits imprimés*. Cela en valait vraiment la peine puisqu'il est reparti avec le second prix.

Artem Kuchukov (second à partir de la gauche) était tout sourire lorsqu'il a reçu le troisième prix pour le ***Kewazo***, *un projet de robot d'assemblage d'échafaudage*.

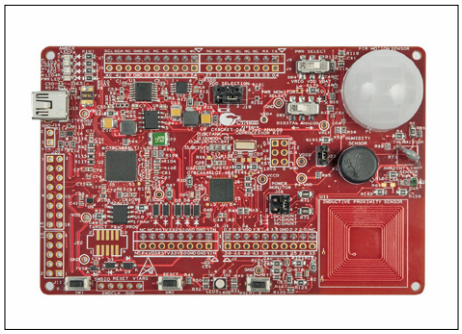
electronica 2016

tour d'horizon des nouveaux produits

Viacheslav Gromov (Allemagne) (readers@gromov.de)

Avec 2.913 exposants et env. 73.000 visiteurs venus de 88 pays différents, le salon *electronica 2016* de Munich (Allemagne) était sous le signe de diversité. Tous les domaines de l'électronique y étaient représentés grâce à une large palette d'innovations : composants passifs (en particulier la connectique), nouveaux microcontrôleurs (de sécurité), ou encore nouveaux capteurs et logiciels de reconnaissance d'images. Sans pouvoir prétendre à l'exhaustivité, nous présentons ici quelques produits qui ont retenu notre attention.

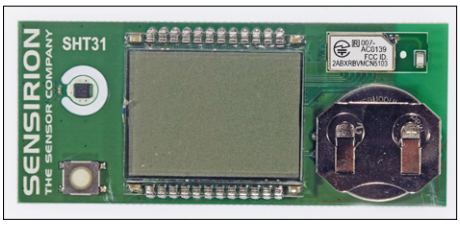
Commençons par les nouveautés de **Cypress** dans deux domaines. Ce fabricant propose une extension de la famille bien connue des µC PSoC, à savoir le coprocesseur analogique PSoC (48 MHz,



32 Ko de mémoire flash). Ce µC à 32 bits, réalisé sur un noyau ARM Cortex-M0, est conçu comme interface entre le monde analogique et le monde numérique. Il embarque des CA/N et des CN/A, des amplificateurs opérationnels, des comparateurs, des filtres analogiques, des boutons, etc. Le kit *CY8CKIT-048 PSoC Analog Coprocessor Pioneer* (autour de 49 $) permet d'exploiter toutes les fonctions disponibles et de profiter de la souplesse de la technologie PSoC : cette petite carte dispose de nombreux capteurs analogiques pour toutes sortes d'applications, de la mesure de température à la détection inductive de distance en passant par la détection de mouvement [1].

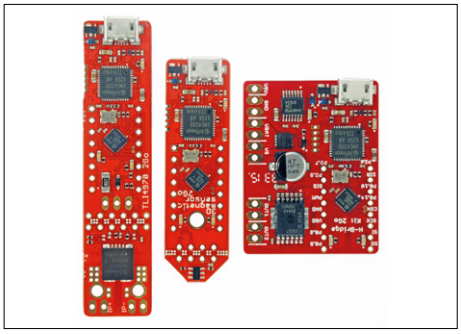
Pendant le salon, la tendance à l'USB de type C était très perceptible, Cypress a apporté sa contribution dans ce domaine. Nous avons par ex. découvert l'*EZ-PD CCG3* (48 MHz, double mémoire flash de 64 Ko, noyau ARM Cortex-M0 à 32 bits) qui peut prendre entièrement en charge une interface USB-C avec toutes les interfaces incluses ainsi que le protocole d'alimentation. Ce kit d'évaluation CY4531 EZ-PD CCG3 coûte env. 250 $. Outre ce kit, il y a aussi le nouvel analyseur de protocoles *CY4500 EZ-PD* (env. 200 $), un appareil très intéressant que l'on peut brancher entre un ordinateur et du matériel USB-C externe, pour analyser le protocole et le flux de données et les renvoyer sur l'interface micro-USB auxiliaire. Cela permet de tester aussi bien la communication programmée entre deux appareils que le câble et les connexions [2].

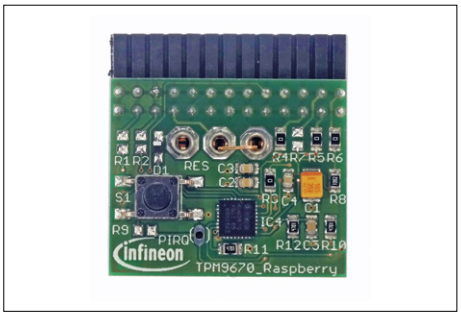
Outre des capteurs sophistiqués pour les flux gazeux, **Sensirion** présente un kit de développement de « gadgets » intel-



ligents (à partir de 30 €) qui exploite brillamment le capteur de température et d'humidité SHT31. Ce nouveau capteur de 2,5 × 2,5 mm² est relié au monde extérieur par une interface I²C. Il mesure à la cadence programmée la température et l'humidité (tolérances de 2 % / 0,3 °C seulement). Avec une mesure par seconde, ce capteur ne consomme que 2 µA. La carte qui l'accueille permet de lire les valeurs sur l'écran et sur un ordiphone via une appli BLE adéquate, ce n'est donc pas qu'un gadget, mais bien un matériel de référence pour le développement de projets. Cette carte comporte un support de pile bouton, un afficheur piloté par un µC à 8 bits MC9S08LL8CGT, un clavier et le module Bluetooth à faible consommation (BLE, en anglais ***B**luetooth **L**ow **E**nergy*) Nordic nRF51822. Les broches des capteurs et de débogage du µC et du module BLE sont facilement accessibles pour la conception de projets spécifiques. L'ensemble des informations relatives au logiciel et au matériel sont publiées sur GitHub, ce qui permet de les adapter à ses propres projets [3].

Infineon a présenté énormément de nouveautés dans nombre de domaines. Outre les innombrables *shields* Arduino qui exploitent de nouveaux circuits intégrés, la famille de cartes déjà bien connue XMC 2Go s'élargit avec la XMC1100 (M0+, 48 MHz, 64 Ko). On peut l'étendre de trois façons : avec TLI4970, un capteur de courant SPI (*Current Sensor 2Go*) ; avec TLV493D-A1B6, un capteur de champ magnétique 3D I²C (*3D Magnetic Sensor 2Go*) ; avec IFX9201, un pont en H SPI (*H-Bridge Kit 2Go*). Outre le cœur XMC, on remarque que ces cartes ont en commun l'accès externe aux sorties

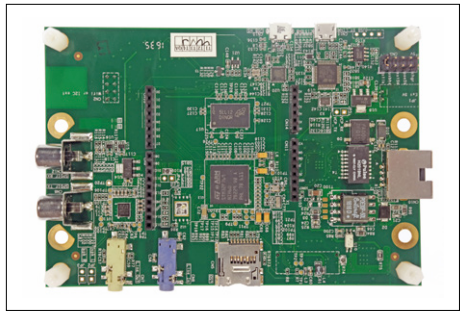




principales du µC et la présence de deux LED utilisateur. Ainsi lorsqu'on a deux cartes d'extension, on peut se servir des capteurs indépendamment de la platine principale pour d'autres applications. Pour le capteur de champ magnétique, il existe une extension avec joystick, qui le transforme en manche de commande proportionnelle. Le prix des différentes cartes tourne autour des 20 € [4].

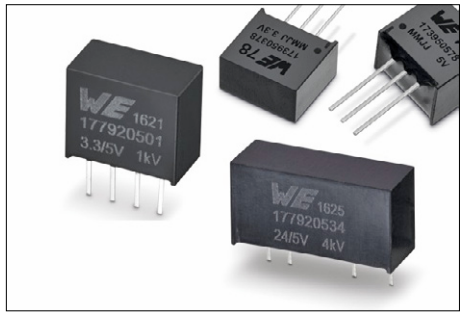
Le thème de la sécurité était naturellement bien présent à *electronica*. À cet égard, Infineon présente aussi un produit TPM (***T**rusted **P**latform **M**odule*, norme internationale de sécurité) baptisé SLB9670. Ce circuit intégré de sécurité piloté par SPI existe aussi en version I²C (SLB9645). Il améliore la sécurité d'un système (lequel peut être un simple µC ou bien un PC complet), car il peut conserver à l'abri jusqu'à huit clés de cryptage de 2048 bits. Ces clés ne quittent jamais le circuit intégré, tous les décryptages nécessaires sont effectués en interne, le processeur connecté n'a jamais connaissance de ces clés lorsqu'il tourne. Habituellement, indépendamment du type de cryptage, de telles clés sont conservées dans une carte SD ou dans une autre mémoire au détriment de la sécurité intrinsèque. Ce circuit intégré fonctionne parfaitement avec un noyau Linux ; le nano-ordinateur Raspberry Pi permet de réaliser une application type intéressante pour laquelle une carte supplémentaire a été développée (bientôt disponible). En outre, la documentation en ligne est très riche. Elle montre par ex. comment réaliser une liaison mieux sécurisée par cryptage SSL entre le RPi et un PC [5].

STMicroelectronics enrichit la famille des µC ARM Cortex-M7 à 32 bits de la gamme STM32 de plusieurs membres dont certains ne sont même pas encore en production de masse. Cependant, le fabricant a d'ores et déjà présenté des cartes développées pour ces µC très prometteurs. La carte F7 la plus récente, déjà en production, est livrée dans le kit de découverte STM32F769 (env. 48 $) avec le STM32F769NIH6 (216 MHz, Flash de 2 Mo) qui se distingue avant tout par ses capacités graphiques. Grâce à l'accélérateur Chrom-ART embarqué (DMA2D) et d'autres périphériques, ce µC est capable de jouer une vidéo DVI à une résolution de 720 p pour 30 images/s. En plus des caractéristiques d'une carte de



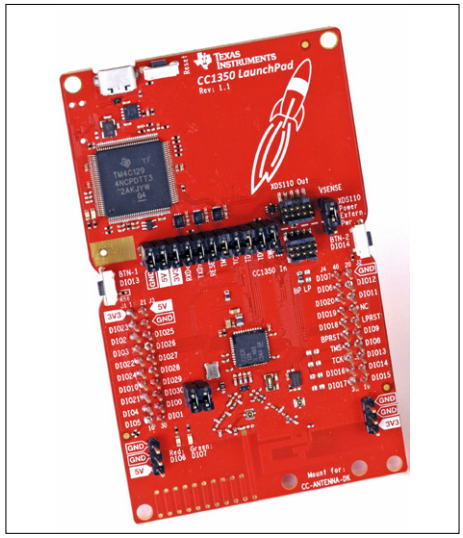
découverte ST F7 normale, cette carte se singularise par ses possibilités d'extension. Si cela s'avère nécessaire pendant la phase de développement, elle peut s'enrichir d'un écran tactile graphique TFT et d'un adaptateur DVI-HDMI ou WLAN [6].

Würth Elektronik continue de fignoler sa gamme de circuits intégrés d'alimentation. Sont apparus quelques représentants des séries MagI³C-FISM (***F**ixed **I**solated **M**odule*) et MagI³C-FDSM (***F**ixed Step **D**own Regulator **M**odule*). Tandis que le MagI³C-FDSM, un régulateur abaisseur (*step down*), adopte une tension de sortie fixe (5 V ou 3,3 V, 1 A max.) et une plage de tension d'entrée très vaste atteignant 42 V (rendement max. 93 %), le MagI³C-FISM reflète une stratégie très différente. Les circuits intégrés de cette série procurent une isolation galvanique entre l'entrée et la sortie jusqu'à une différence de potentiel de 1 kV. La tension de sortie est toujours fixée à 5 V, la tension d'entrée peut atteindre 24 V, la puissance maximale de sortie est de 1 W (rendement max. 80 %). Les particularités communes



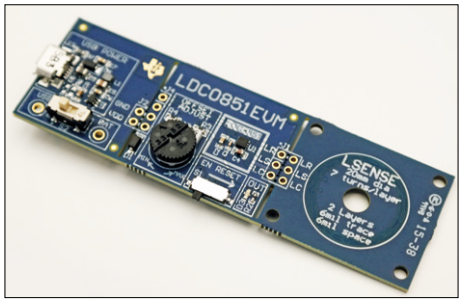
à ces deux circuits sont la simplicité d'utilisation, la sécurité et les éléments externes quasi inexistants [7][8].

Texas Instruments n'est pas en reste avec sa nouvelle gamme de produits. Le nouveau µC *Simple Link CC1350* a fait sensation. Outre son noyau ARM Cortex-M3 à 32 bits qui peut être cadencé à 48 MHz, ce µC dispose de mémoire flash (jusqu'à 128 Ko) et de deux périphériques radio : l'un pour la communication en UHF (< 1 GHz) et l'autre pour le Bluetooth BLE largement répandu. L'intérêt de cette combinaison est la communication intelligente à l'intérieur d'un réseau. Pour l'IdO ou les applications industrielles 4.0, l'intégration à un réseau est devenue stratégique. Avec le CC1350, tous les capteurs et actionneurs d'une maison peuvent communiquer entre eux sur de



grandes distances par radio UHF (bande des 868 MHz dans l'UE) et le processeur central peut simultanément communiquer avec l'ordiphone de l'utilisateur par BLE. Naturellement, pour ce nouveau microcontrôleur sans fil, une platine de lancement a été développée : *LaunchPad CC1350* au prix de 29 $ [9].

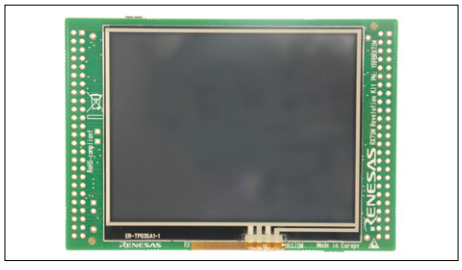
Pour la première fois à *electronica*, on observe un virage technologique du tactile capacitif vers l'inductif. De nombreux fabricants ont déjà intégré cette technologie y compris jusque dans leurs µC. L'avantage est avant tout une meilleure immunité aux perturbations de l'environnement comme l'eau, la saleté, etc. TI rejoint cette tendance en lançant ses propres circuits intégrés, petits et grands, dont les réglages s'effectuent par de



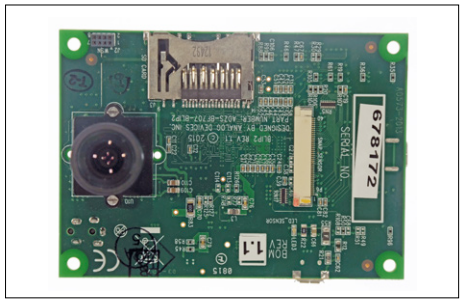
simples potentiomètres. Le LDC0851 est l'un des plus petits de la famille. Il ne nécessite que très peu de composants externes et fonctionne à l'aide d'un petit capteur bobiné (imprimé sur la carte). La LDC0851EVM est une petite carte de développement (au prix modique de 18 €) qui comporte un bouton à capteur inductif ; elle est alimentée par le port micro-USB ou une pile bouton montée sur le support intégré. La carte est sécable, on peut donc n'utiliser que le circuit intégré ou que la bobine du capteur dans des applications particulières [10].

Les nombreuses conférences de **Renesas** étaient focalisées sur le nouveau kit *Revelation RPBRX71M* (59 € net). Le principal avantage du RX71M, nouveau membre de la famille RX700, est son afficheur tactile TFT 320×240 embarqué. Le RX71M est un µC à 32 bits, qui peut être cadencé jusqu'à 240 MHz et peut accueillir jusqu'à 4 Mo de mémoire flash. Les interfaces constituent son point fort, mais il n'est pas en reste en ce qui concerne la sécurité, car ce µC dispose d'un grand nombre de fonctions de cryptage ainsi que de possibilités d'étalonnage et de test des périphériques internes (par ex. CN/A, GPIO) [11].

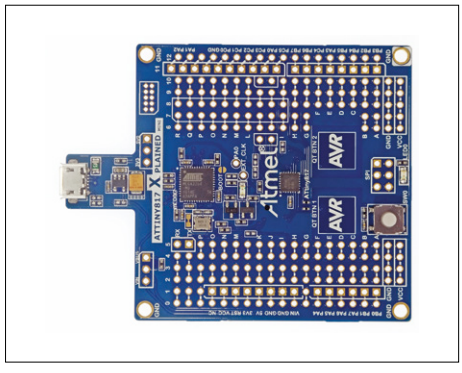
En dehors des tout nouveaux produits de sa gamme analogique, **Analog Devices** a montré les possibilités offertes par la (plus tout à fait nouvelle) carte ADZS-BF707-BLIP2 ainsi que les technologies connexes qu'elle exploite. Cette carte d'un prix avoisinant les 200 $, aux très nombreuses fonctions et interfaces avec son ADZS-BF707 à 32 bits cadencé



à 400 MHz, embarque une caméra, ce qui lui permet de détecter de manière fiable des objets et même des visages. Pour concevoir un projet avec cette carte, mieux vaut utiliser l'émulateur ICE-1000 livré avec le kit et *CrossCore Embedded Studio*, cela simplifie le travail, en particulier pour ce qui est du logiciel de cette technologie [12].



La vaste communauté **Atmel** & **Microchip** a été comblée d'attentions : Microchip était fier de présenter les nouveaux ATtiny 817, 816, 814 et 417. Ainsi, la branche AVR non seulement se maintient, mais continue (comme promis) de se développer. Avec ces quatre nouveaux membres, l'adaptation aux grandes familles de µC AVR et SAM est presque complète. En effet, en dehors des périphériques habituels, ces µC de 14 à 24 broches avec 4 ou 8 Ko de mémoire flash disposent maintenant d'un véritable contrôleur tactile périphérique indépendant (PTC, ***P**eripheral **T**ouch **C**ontroller*),



d'un système événementiel (ES, ***E**vent **S**ystem*), d'une horloge en temps réel (RTC, ***R**eal **T**ime **C**lock*) et de nombreux blocs logiques programmables par l'utilisateur (LUT). Toutes ces caractéristiques visent à rendre les périphériques indépendants de la CPU pour la décharger et améliorer le comportement en temps réel. La carte de développement de ces nouveaux ATtiny est déjà connue : mini-carte ATtiny817 Xplained (env. 12 $) [13].

La famille PIC18F accueille un nouveau membre : le K40. Il est caractérisé par des boîtiers de 28 à 64 broches et 16 à 128 Ko de mémoire flash. Il est cadencé jusqu'à 64 MHz. Les points forts sont avant tout le **C**onvertisseur **A**nalogique **N**umérique avec possibilités de **C**alcul (*ADC²*), qui peut fournir des moyennes ou comparer des valeurs sans intervention du processeur (nécessaire pour les applications tactiles), ainsi qu'un meilleur niveau général de sécurité. Ce dernier point est obtenu grâce à un calcul de somme de contrôle sur la mémoire et un chien de garde spécial et le *Hardware Limit Timer* (un *compteur* matériel qui intervient quand un périphérique externe raccordé ne répond pas dans le délai imparti). Beaucoup des µC K40 sont disponibles en boîtier DIP, ce qui permet de les utiliser sur les cartes de développement universelles Curiosity (env. 30 €) [14].

(160270 – version française : Yves Georges)

Liens

[1] www.cypress.com/products/psoc-analog-coprocessor

[2] www.cypress.com/products/ez-pd-ccg3-type-c-port-controller-pd

[3] www.sensirion.com/de/produkte/feuchtesensoren/development-kit/

[4] http://goo.gl/o9hhGd

[5] http://goo.gl/Xu5Lkr

[6] www.st.com/en/evaluation-tools/32f769idiscovery.html

[7] http://katalog.we-online.de/de/pm/MAGIC-FISM

[8] http://katalog.we-online.de/de/pm/MAGIC-FDSM

[9] www.ti.com/tool/launchxl-cc1350

[10] www.ti.com/tool/ldc0851evm

[11] www.renesas.com/en-eu/solutions/key-technology/human-interface/rx71m-revelation.html

[12] http://goo.gl/LpdZig

[13] www.atmel.com/tools/ATTINY817-XMINI.aspx

[14] www.microchip.com/promo/pic18f67k40

bienvenue dans la section CRÉER

Clemens Valens, labo d'Elektor

Heinrich Rudolf Hertz (1857 – 1894)

La cité allemande de Hamburg est le berceau du sandwich à la viande hachée appelé *hamburger* et du *hertz*, l'unité SI de fréquence. Les deux sont apparus dans la seconde moitié du XIXe siècle, mais nous ne parlerons ici que de l'unité, importante pour les électroniciens. Qui était donc l'homme qui a lassé son nom à l'unité de fréquence ?

Heinrich Rudolf Hertz nait dans une famille aisée, son père est avocat et même sénateur de Hambourg. Heinrich montre très tôt un vif intérêt pour la science et les langues, et fréquente des universités réputées. Il obtient son doctorat à Berlin ; Hermann von Helmholtz et Gustav Kirchhoff, deux scientifiques de renom, familiers de tous les ingénieurs électroniciens ont été ses professeurs. Bien entendu, Heinrich n'a pas inventé la fréquence. Il commence à travailler sur l'électromagnétisme lorsque Helmholtz lui suggère de tenter de prouver la théorie de Maxwell. Au départ il pense que cela n'est pas possible, toutefois il s'attaque au sujet des années plus tard lorsqu'il invente fortuitement les outils appropriés. Une fois la théorie de Maxwell démontrée, il n'y pense plus, car il trouve que ce n'est pas d'une grande utilité.

Avec plus d'une corde à son arc, Heinrich jette aussi les bases de la mécanique des contacts lorsqu'il résout le problème du contact de deux corps élastiques à surface courbe. Lorsqu'il observe qu'un objet chargé perd sa charge plus rapidement s'il est illuminé par une radiation ultraviolette, il contribue à établir l'effet photoélectrique. Ses expériences débouchent même sur le début des rayons X. Heinrich prouve une dernière fois qu'il est un scientifique très pointu en mourant d'une maladie rare, ce qui ne fut officiellement découvert que 40 ans plus tard.

Bien qu'il meure très jeune, Heinrich transmet une partie de son génie à sa progéniture, en l'occurrence sa plus jeune fille Mathilde qui fut biologiste et psychologue de renommée mondiale. Son neveu, Gustav Ludwig, obtient le prix Nobel de physique en 1925. Si l'on regarde les photos d'Heinrich et de son neveu, leur ressemblance est frappante. Il se murmure même que « le père de son neveu n'était pas son père et » que « le père de son neveu ne le savait pas ».

L'unité SI de fréquence devient le hertz (Hz) en 1960 en remplaçant le cycle par seconde (cps). Le *hertz* correspond à 1/seconde ou s^{-1}, exactement comme le becquerel (Bq), cependant, un hertz signifie un événement par seconde, les événements étant *exactement* espacés d'une seconde, en revanche pour un becquerel, il s'agit d'un événement par seconde *en moyenne*.

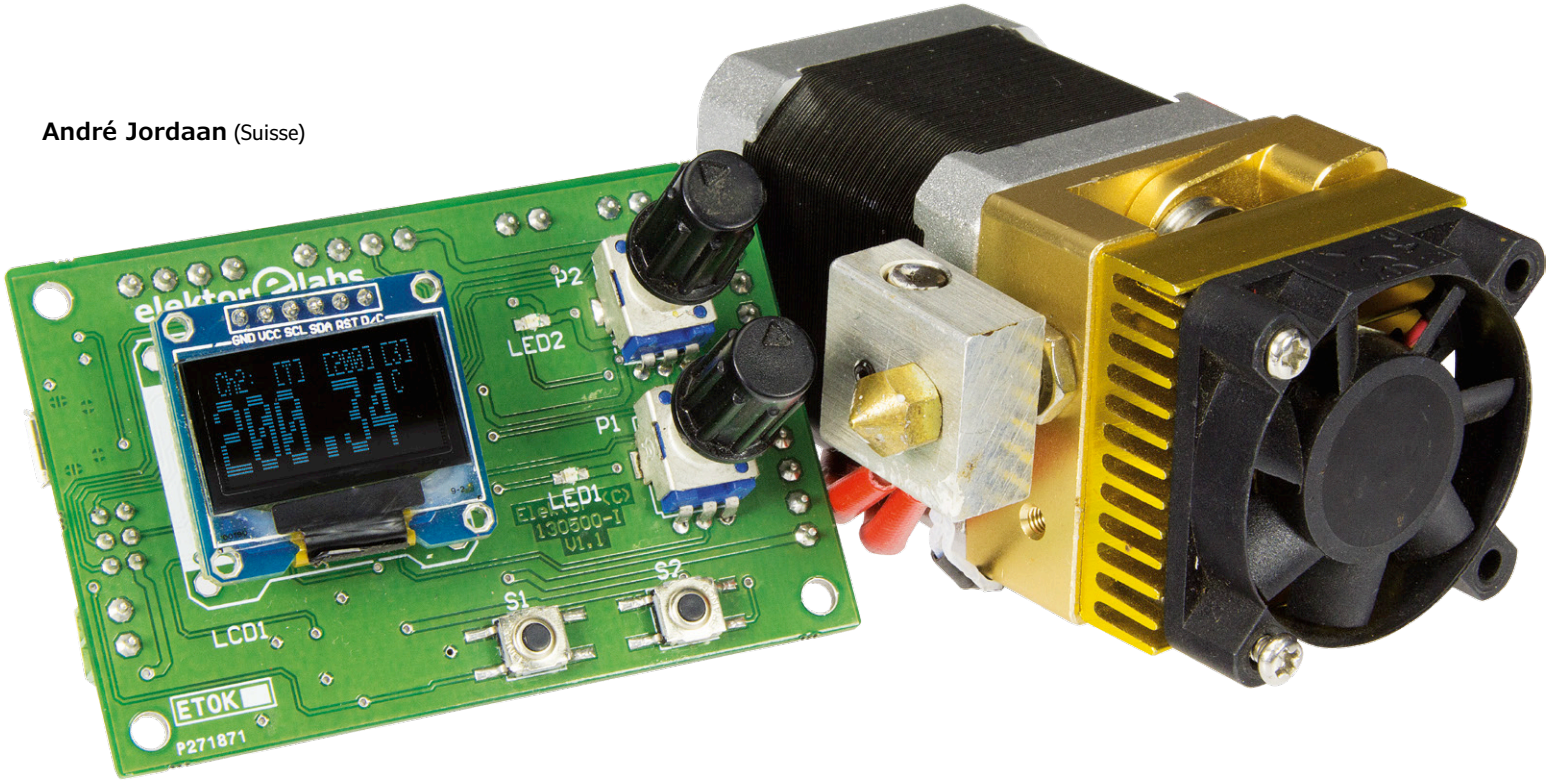
Tandis qu'un massif montagneux est baptisé Joseph Henry, un cratère sur la face cachée de la lune porte le nom d'Heinrich. En cherchant sur l'internet, j'ai même réussi à trouver une recette de *Hertz burger* qui combine les deux grandes inventions de la ville de Hambourg du XIXe siècle en une seule.

(160247 – version française : Yves Georges)

régulateur de température de tête d'imprimante 3D

ou du chauffage de la cage de votre animal favori cet hiver

André Jordaan (Suisse)



Pour un bon résultat, chaque type de filament d'impression 3D (ABS, PLA, etc.) doit travailler à sa température d'extrusion optimale. Pour un même matériau, cette température peut dépendre de la couleur du filament. Il faut donc une régulation précise de la température de la tête d'extrusion. Et pourquoi ne pas contrôler aussi la température du lit ?

Ce projet est né quand j'ai converti une meule X2 en meule à commande CNC. Pour adapter ce montage à une imprimante 3D, il suffit d'ajouter une tête d'impression et un système de régulation précis de la température de la tête et du lit sur lequel l'objet sera imprimé, j'ai donc jeté les bases de ce système. Avec un µC Arduino et un afficheur OLED, quelques thermistances et quelques éléments chauffants, j'ai rapidement implanté un prototype sur une plaque d'essai. Une fois terminé, j'ai pensé que ce régulateur pouvait aussi remplir d'autres missions comme tempérer un clapier pendant l'hiver ou protéger du gel des plantes hivernant à l'extérieur. Finaliser et documenter ce projet conçu par hasard, mais utile, m'a semblé être une bonne idée.

Principales caractéristiques

La **figure 1** donne un aperçu du système : un régulateur bivoie à sorties opto-isolées.

Régulation PID

Les températures de la tête d'impression et du support (une voie chacun) sont régulées par des signaux MLI (*PWM*) qui pilotent les MOSFET de commutation des éléments chauffants (12 V/40 W). Une boucle de régulation PID (voir plus loin) asservit la température à la consigne à ±½ °C. Deux potentiomètres permettent de régler facilement les consignes.

Sondes de température : analogiques ou numériques

Pour mesurer la température, le régulateur travaille soit avec des sondes numériques de type « DHT » (ici DHT22, aussi vendue sous la réf. AM2302 et disponible sur l'internet, mais une DS18B20 devrait aussi fonctionner), soit avec des thermistances (type CTN). Les sondes des deux voies peuvent être chacune d'un type différent.

Les thermistances supportent des températures élevées. C'est parfait pour la tête d'extrusion. Comme les matériaux utilisés pour les filaments des imprimantes 3D ont un point de fusion autour de 200 °C, j'ai choisi une thermistance 300 °C, 100 kΩ, enrobée de verre.

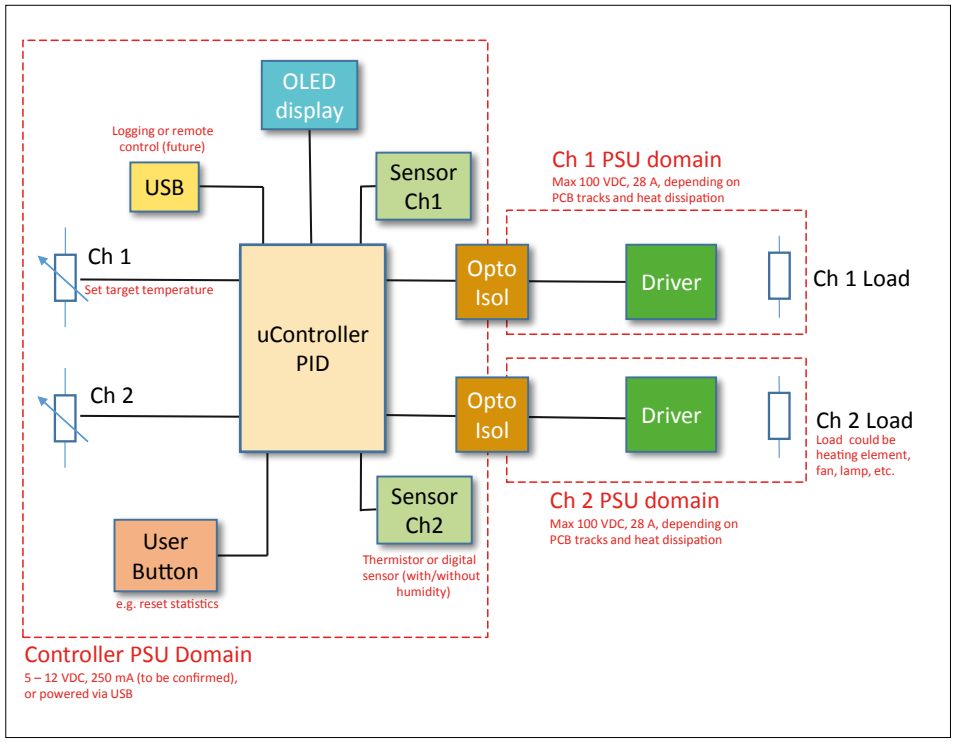


Figure 1. Vue d'ensemble du régulateur de température : un µC commande un transistor de puissance.

L'équation de Steinhart-Hart régit le calcul de la température de la sonde et permet de l'exploiter au mieux.

Collecte des données

Les températures et les paramètres PID sont envoyés par la liaison USB au PC où ils sont enregistrés.

Affichage graphique

Selon le logiciel (il y a deux versions, voir plus loin), l'écran OLED compatible SSD1306 affiche la température instantanée de la tête, la consigne et l'écart, les températures max. et min. enregistrées depuis la mise sous tension (ou depuis la dernière pression sur le bouton S2). L'affichage alterne toutes les 5 s entre les voies 1 et 2.

Le coin supérieur droit contient le numéro du jeu de paramètres PID utilisé :

1. précis – écart réduit, mais lent ;
2. normal – écart raisonnable, réactivité moyenne ;
3. agressif – écart plus grand, mais rapide.

Dans le coin supérieur gauche, une animation indique que le régulateur fonctionne ; si ce n'est pas le cas, un chien de garde relance le système. Si la sonde de température est numérique, et comporte un capteur d'humidité relative comme le DHT22, l'humidité s'affiche également.

Le circuit

Examinons la **figure 2**. Le cœur du régulateur est un µC AVR à 8 bits, ATmega32U4 (IC1), avec interface USB intégrée. Cadencé à 16 MHz, il est pris en charge dans l'EDI Arduino comme un Arduino Micro ou Leonardo ; il suffit d'y transférer le chargeur d'amorçage adéquat (inclus dans l'EDI Arduino). À cet effet, utilisez le connecteur ICSP K2.

Les potentiomètres P1 et P2 règlent la température de consigne des deux voies. Le poussoir S2 est le bouton de l'utilisateur. Selon le logiciel, S2 réinitialise les statistiques collectées par le programme ou bien fait alterner le mode de sonde utilisé (numérique / analogique).

Les sondes de température numériques se connectent sur K7 et K8 et les thermistances sur K3 et K4. Selon la version du logiciel, le mode *thermistance* est essayé en premier. Si le programme ne détecte rien (mauvais branchement, sonde défectueuse), il essaie le mode *sonde numérique*. L'autre version du logiciel devrait être configurée pour travailler avec des sondes, soit analogiques, soit numériques.

Les résistances R8 et R9 règlent la tension sur les sorties des sondes à la moitié de la tension d'alimentation. Les valeurs données sont pour des thermistances de 100 kΩ (à adapter si nécessaire). Le choix de thermistance dépend de l'application ; les résistances R8 et R9 de la thermistance retenue.

Les deux voies de chauffage sont bâties autour d'IC3/T1 et IC4/T2, des circuits de commande opto-isolés qui empêchent tout accident (dysfonctionnement ou détérioration du régulateur) que pourraient causer le bruit et les interférences des alimentations des chauffages. Pour garantir un fonctionnement correct, évitez de connecter V_{IN} à V_{IN1} ou V_{IN2}, et ne faites pas de connexion à la masse. C'est peut-être moins évident qu'il n'y paraît, en particulier si le régulateur est raccordé au même PC que l'imprimante 3D (ou une autre charge) via le port USB (K1)…

Les chauffages et leur alimentation **CC** doivent être **exclusivement** connectés à K5 et K6. **Ne** tentez **pas** de connecter des charges en alternatif ! D'après leurs

Bibliothèques Arduino

Pour ce projet, le logiciel est conçu à partir de bibliothèques mises gratuitement à disposition par la communauté Arduino. Ce genre de bibliothèques continue d'évoluer et pourrait ne plus fonctionner avec notre logiciel au moment où vous nous lirez. Heureusement, nous sommes habilités à les redistribuer et les versions utilisées sont incluses dans le téléchargement disponible sur [1]. La version la plus récente de ces bibliothèques (que nous n'avons peut-être pas essayées) se trouve sur GitHub :

https://github.com/adafruit/Adafruit_SSD1306
https://github.com/adafruit/Adafruit-GFX-Library
https://github.com/br3ttb/Arduino-PID-Library
https://github.com/PaulStoffregen/Time
https://github.com/PaulStoffregen/TimeAlarms
https://github.com/RobTillaart/Arduino/tree/master/libraries/DHTlib

fiches techniques, les transistors supportent 28 A (et une tension de 100 V), mais la carte ne tolère pas une telle intensité. Pour un fonctionnement fiable, le courant débité n'excédera pas 2,5 A par voie. Les risque-tout pourront augmenter le courant maximal en refroidissant les transistors et en renforçant les pistes de la carte.

Cela dit, on peut aussi bien connecter un chauffage et un ventilateur (ou une lampe). L'afficheur graphique OLED LCD1, un module standard disponible partout sur l'internet, peut être configuré en mode SPI ou I²C. J'ai choisi le mode SPI ce qui n'est pas évident si on regarde de près les photos du prototype. Sur le schéma, on voit aussi des étiquettes I²C dans le cadre de LCD1 de sorte que toutes les illustrations montrent la même chose.

Le régulateur peut être alimenté avec un adaptateur secteur/CC, mais sans dépasser 12 V. Le courant à fournir est faible puisque les charges sont alimentées séparément. Il est aussi possible d'alimenter le régulateur par la liaison USB.

Le logiciel

Bien que la tâche du logiciel semble simple (lire quelques capteurs, contrôler deux charges et afficher les données qui nous intéressent), le programme est très long. Par chance, une grande partie du code nécessaire existait sur l'internet (voir l'encadré *bibliothèques Arduino*). J'ai appris à utiliser ce code, puis j'ai écrit de quoi coordonner les différentes parties.

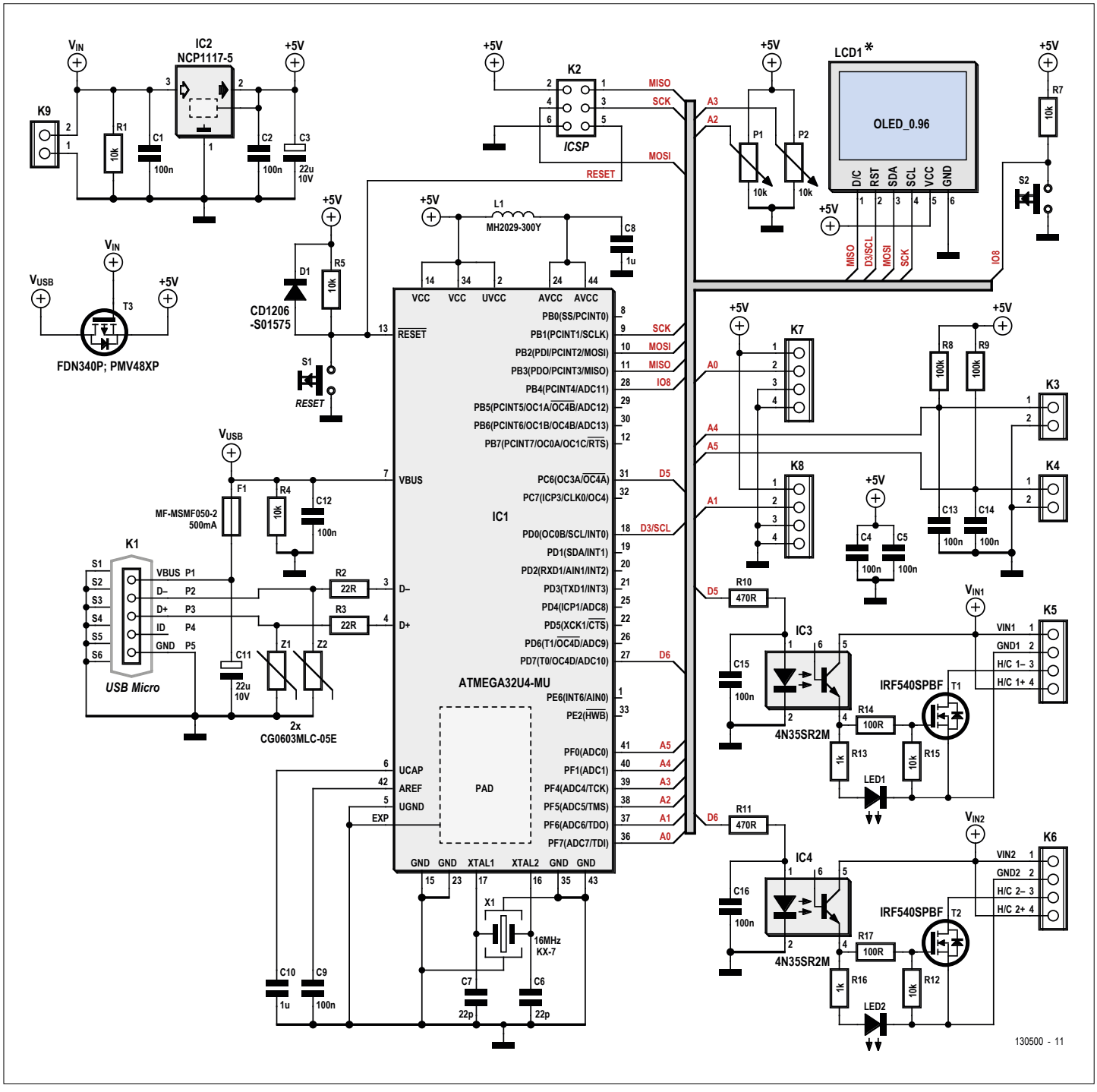


Figure 2. Le schéma du régulateur de température à deux voies montre clairement les deux sorties opto-isolées.

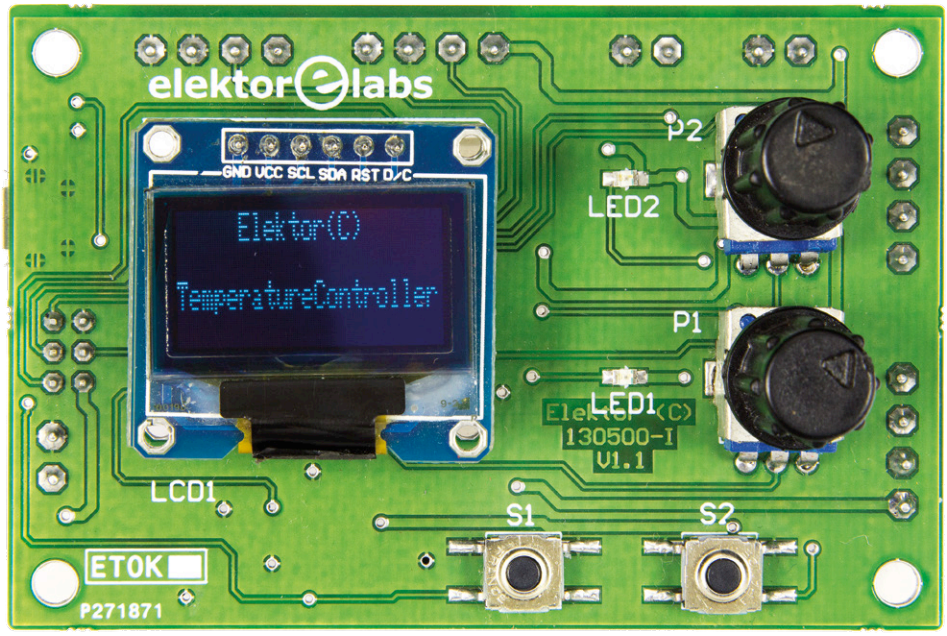


Figure 3. Côté interface utilisateur du régulateur de température.

Il ne reste que deux versions finales :

- le code figé par le labo d'Elektor [1] ;
- le code évolutif de mon cru [2].

Les deux versions ont les mêmes objectifs, mais certains détails diffèrent. Si vous chargez une version et voyez que cela ne correspond pas à ce qui est écrit ici, chargez l'autre version et réessayez.

Sondes numériques

Les sondes numériques sont prises en charge par la bibliothèque spécifique *dht*. Comme ce composant est très répandu, il bénéficie de plusieurs bibliothèques dont les noms peuvent être confondus. Faites en sorte d'utiliser la bonne (incluse dans le téléchargement sur [1]).

Équation de Steinhart-Hart

Les capteurs analogiques sont... heu, analogiques et c'est une tension que l'on mesure avant de la convertir en résistance.

Le polynôme (étendu) de Steinhart-Hart décrit bien la relation entre la température et la résistance de la thermistance :

$1 / T = A_0 + A_1 \times \ln(R) + A_2 \times [\ln(R)]^2 + A_3 \times [\ln(R)]^3$

Une telle équation n'a de sens que si R8 et R9 ont une tolérance de 1 % et que la référence de tension est plus précise qu'une alim USB. Il y a aussi une raison plus triviale d'utiliser ce polynôme : comme on ne stocke pas de grande table de conversion, la mémoire est réservée au programme actuel et à des fonctions qui seront ajoutées ultérieurement.
Les coefficients A_0 à A_3 dépendent de la thermistance utilisée ; un petit outil (en C ou Java, j'ai opté pour le C) permet de les obtenir (voir [3]). Cet utilitaire `coeff.exe` attend en entrée la table température/résistance de référence de la fiche technique de la thermistance (sauvegardez-la sous `simu.txt` dans le même dossier que `coeff.exe` pour remplacer la version initiale), ensuite il faut ajuster le polynôme à celle-ci (exécutez `coeff.exe`). On obtient quatre coefficients (plus d'autres données de vérification). Nota : j'ai eu des problèmes avec la version 1.0 de l'outil, j'ai utilisé la 0.1.

Retour au régulateur PID

Une fois connue, on envoie la température au régulateur PID. Souvent les régulateurs PID intimident, car leur implémentation pose problème. Pourquoi ne pas avoir recours à une bibliothèque trouvée sur l'internet ? C'est ce que j'ai fait. La bibliothèque sélectionnée est en outre accompagnée d'une bonne documentation [4] sur la théorie et la pratique des régulateurs PID. En deux mots, un régulateur PID utilise trois valeurs dérivées des mesures instantanées d'un paramètre pour produire un signal de commande qui réduit l'écart entre la valeur instantanée de ce paramètre et sa valeur théorique.

Ces trois valeurs sont : la valeur instantanée (dite proportionnelle soit *P*), l'écart cumulé entre la valeur instantanée et la consigne (= l'intégrale de P, soit *I*), et le taux de variation de la valeur instantanée (= la dérivée de P soit *D*). En injectant un

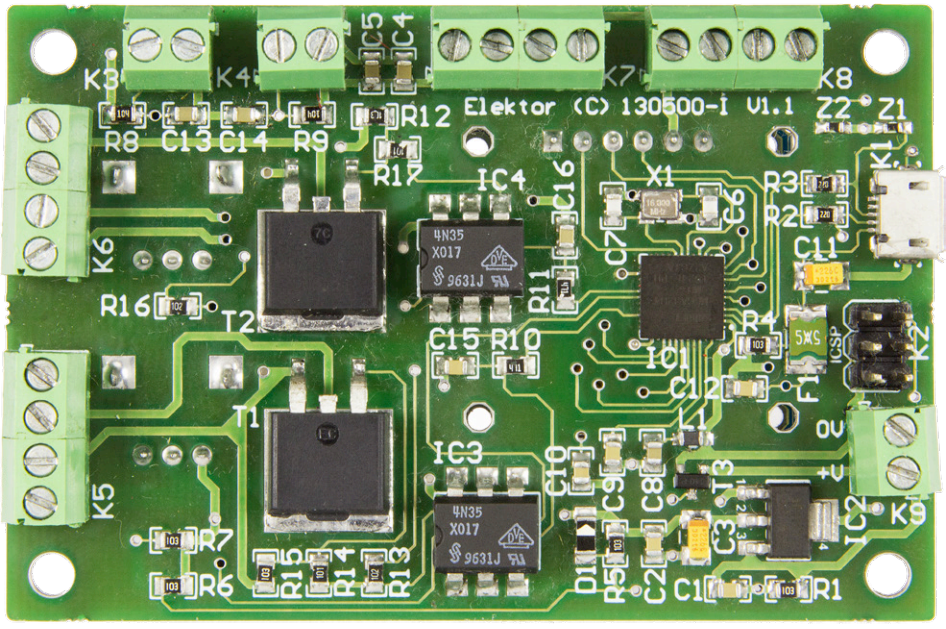


Figure 4. Le verso... ou recto (selon le point de vue), enfin, disons le côté composants.

peu de ces trois valeurs, on produit un signal de commande qui pilote la sortie. La manière dont la sortie réagit au signal de commande dépend des proportions de *P*, *I*, et *D* injectées (pondération). En privilégiant *P*, la réponse est rapide, mais la sortie va dépasser la consigne (par excès ou par défaut selon le sens) et risque de se stabiliser à une valeur erronée. En introduisant *I*, la sortie rejoindra la consigne au bout d'un certain temps (l'écart cumulé tend vers zéro), mais avec des suroscillations. En introduisant *D*, on amortit l'oscillation (la pente diminue), mais l'ensemble est plus lent. Tout l'art du réglage d'un régulateur PID réside dans le dosage de ces trois ingrédients de sorte que le système se comporte le mieux possible. Les valeurs idéales dépendent du système. Des livres entiers sont consacrés aux méthodes théoriques et empiriques d'« accord » des régulateurs PID. La place me manque ici pour expliquer tout cela et je ne suis pas un spécialiste des régulateurs PID, je me suis contenté d'une méthode empirique. Si vous voulez faire mieux, cherchez sur l'internet. Ici, les trois réglages du PID (agressif, normal ou précis) sont obtenus avec des valeurs par défaut pour *P*, *I* et *D*.

Le reste du programme sert principalement à afficher l'interface utilisateur sur l'écran OLED. Comme il s'agit d'un écran graphique, il y a pas mal de programmation. Heureusement je ne suis pas le premier à l'utiliser et les bibliothèques Arduino facilitent la tâche. « Vive le code source ouvert ! »

Conclusion & futur

Cet article montre comment construire un régulateur PID pour réguler avec précision la température d'une tête d'imprimante 3D. Cependant comme ce montage est polyvalent, je l'utilise dans bien d'autres cas. Actuellement je mets au point sa commande par le réseau en lui adjoignant un serveur web. J'utilise pour cela un Raspberry Pi qui communique avec le régulateur par un bus I²C isolé, comme celui du projet 150089 [5] d'Elektor. Pour une imprimante 3D, c'est inutile, mais il y a plein de raisons de vouloir réguler une température à distance. J'ai commencé à écrire le logiciel, mais je suis loin d'avoir terminé. Pour savoir où j'en suis, regardez sur [2].

(130500-1 – version française : Yves Georges)

Liste des composants

Résistances

Par défaut : CMS 0805, 0,1 W, 1 %
R1, R4, R5, R7, R12, R15 = 10 kΩ
R2, R3 = 22 Ω
R8, R9 = 100 kΩ (voir texte)
R10, R11 = 470 Ω
R13, R16 = 1 kΩ
R14, R17 = 100 Ω
P1, P2 = 10 kΩ, pot. linéaire

Condensateurs

Tous du type CMS 0805
C1, C2, C4, C5, C9, C12, C13, C14, C15, C16 = 100 nF
C3, C11 = 22 µF, 10 V, boîtier A
C6, C7 = 22 pF
C8, C10 = 1 µF

Inductances (CMS 0805)

L1 = ferrite MH2029-300Y, 0,025 Ω, 3 A

Semi-conducteurs

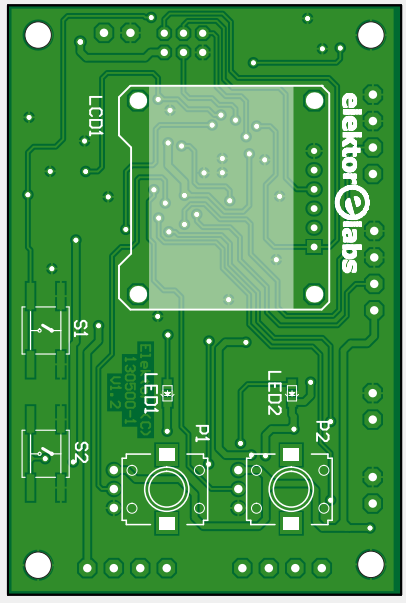
D1 = CD1206-S01575
IC1 = ATmega32U4-MU
IC2 = NCP1117ST50T3G
IC3, IC4 = 4N35SR2M
LED1, LED2 = LED, vert
T1, T2 = IRF540SPBF
T3 = FDN340P

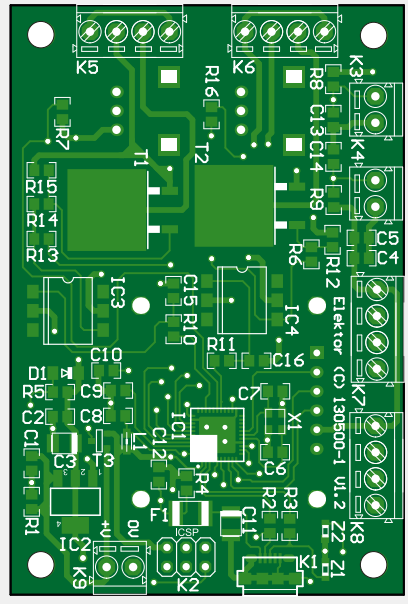
Divers

LCD1 = écran OLED SPI/I²C, 2,4 cm (0,96"), compatible SSD1306
Z1, Z2 = varistance CG0603MLC-05E
F1 = MF-MSMF050-2, 15 V, 0,5 A, 1812
X1 = quartz 16 MHz, 2,5×2 mm
S1, S2 = bouton-poussoir, 6×6 mm, connexions en aile de mouette (*gull wing lead*)
(p. ex. C&K Components KSC321GLFS)
K1 = micro-USB de type B
(p. ex. Amphenol FCI 10104110-0001LF)
K2 = connecteur mâle à 6 contacts (2×3), pas de 2,54 mm, vertical
K3, K4, K9 = borniers à vis à souder, à 2 contacts, pas de 3,5 mm
K5, K6, K7, K8 = borniers à vis à souder, à 4 contacts, pas de 3,5 mm

Divers

Pour LCD1 : connecteur mâle à 6 contacts (2×3), pas de 2,54 mm et connecteur femelle à 6 contacts (2×3), pas de 2,54 mm
Circuit imprimé réf. 130500-1





Liens

[1] www.elektormagazine.fr/130500
[2] www.elektor-labs.fr/3d-printer-head-and-mat-temperature-controller-using-arduino-130500-i
[3] http://thermistor.sourceforge.net/
[4] http://brettbeauregard.com/blog/2011/04/improving-the-beginners-pid-introduction/
[5] isolateur universel pour bus I²C avec adaptateur de niveau, Elektor 12/2016 : www.elektormagazine.fr/150089

sonomètre Arduino tricolore

un nouveau *shield* est né

Clemens Valens (Elektor-lab)

Arduino, c'est un vrai mille-pattes ! Une carte à microcontrôleur pour tout faire et plus encore, avec ses connecteurs d'extension déjà disponibles pour de nombreux *shields*. Le *shield* proposé ici permet de réaliser un feu de signalisation du niveau sonore. Cette application est composée d'une carte Uno surmontée d'une platine polyvalente, développée par Elektor, avec une poignée de composants standard. L'ensemble protégera vos oreilles.

Le livre « maîtrisez les microcontrôleurs à l'aide d'Arduino » [1] constitue une excellente introduction au monde des µC et de leur programmation pour tous ceux qui s'y intéressent, parce qu'il ne demande aucune connaissance préalable dans le domaine. Il contient en outre la description de nombreux exemples de matériel simples que l'on peut assembler sur une plaque d'expérimentation. La deuxième édition est complétée par le chapitre 11 consacré à plusieurs exemples autour d'un *shield* universel, la platine polyvalente Arduino (réf. 190009-1) disponible dans l'e-choppe (carte nue ou kit). Avec elle, vous pourrez construire très vite différents circuits. Trouvent place sur le circuit imprimé un afficheur LCD, deux boutons-poussoirs, deux transistors de puissance, des capteurs de température, de pression atmosphérique et d'humidité, un préampli de microphone, un capteur IR et un récepteur de télécommande. On y voit aussi deux sorties pour LED et vibreur acoustique. Il y a en outre une entrée analogique blindée et filtrée ainsi que deux E/S numériques protégées qui peuvent servir de port sériel.

Arduino, c'est un vrai mille-pattes !
Une carte à microcontrôleur pour tout faire et plus encore.

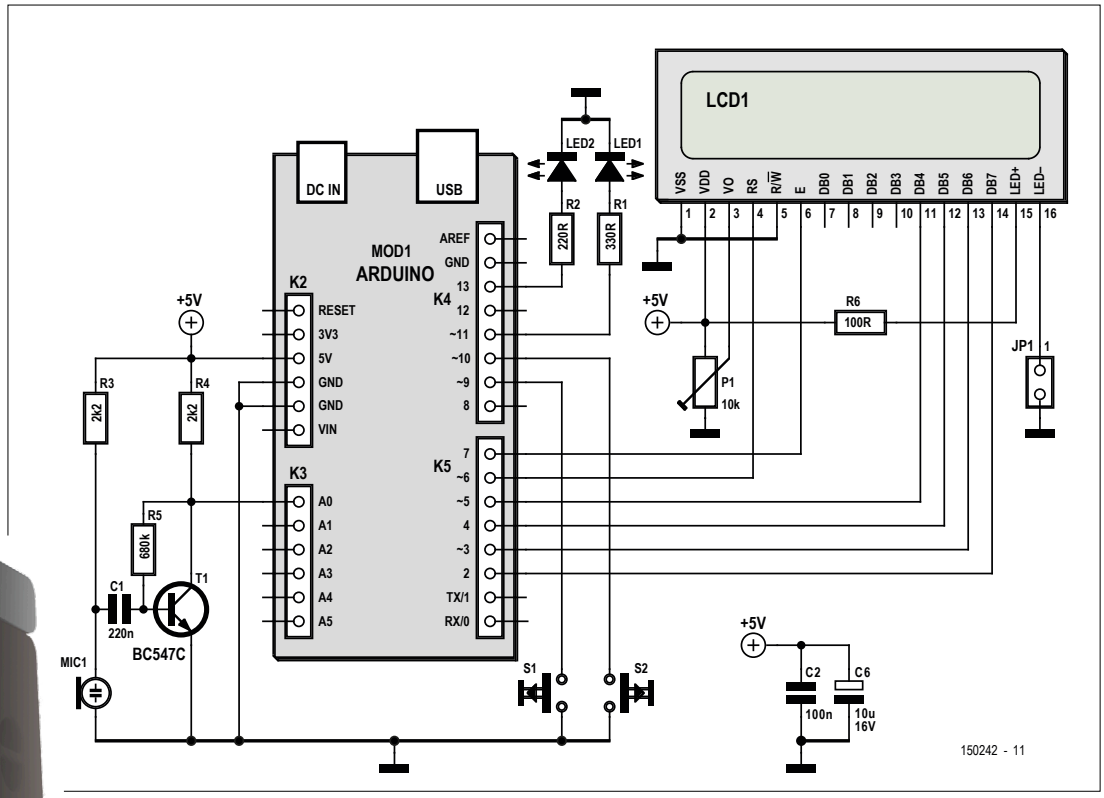


Figure 1. Le sonomètre est composé d'à peine quelques composants montés sur la platine polyvalente de type *shield*.

Tous ces composants ne peuvent pas être câblés en même temps, mais beaucoup quand même.

L'idée pour le circuit présenté ici était de créer à l'aide de quelques LED un genre de sonomètre à sémaphore dont les couleurs correspondent aux différents niveaux du bruit. En particulier, le rouge veut dire « trop fort », danger ! C'est utile pendant un concert ou quand les voisins font la fête un peu trop bruyamment.

Même si ce circuit n'est réalisé qu'avec des composants discrets, il permet toutefois de découvrir les différentes fonctions que la platine polyvalente Arduino peut vous offrir. À côté des LED, vous disposez d'un LCD qui peut servir de vumètre aux gammes sélectionnées par bouton-poussoir. En somme, un excellent tremplin pour acquérir une expérience variée en programmation avec Arduino.

Le circuit

Tout est préparé sur la platine polyvalente 129009-1 pour réaliser un ampli microphonique. Ce sera alors à Arduino de réagir au son. Le micro est un modèle à électret et l'ampli est un classique étage à un transistor au gain voisin de 100, voir la **figure 1**. Sa sortie est dirigée vers l'entrée analogique A0 de la carte Arduino. Au départ, on avait pensé à commettre deux LED à l'indication du niveau, une verte et une rouge, le *shield* le permet encore. Au moment de la construction, il nous manquait une LED verte de 5 mm. En revanche, une LED bicolore rouge/verte à cathode commune attendait sa chance et je l'ai installée à la place de LED2, la troisième patte raccordée à LED1, parce que le vibreur partage une sortie avec LED1 et s'installe près de LED2. Or, avec une LED bicolore, je gagne une couleur : l'orange. Désormais, on voit un « vrai » feu tricolore.

Sans instrument de mesure du niveau sonore au labo Elektor, il ne m'a pas été possible d'étalonner

Liste des composants

Résistances :
(5 %, 0,25 W)
R1 = 330 Ω
R2 = 150 Ω
R3, R4 = 2,2 kΩ
R5 = 680 kΩ
R6 = 100 Ω
P1 = pot. ajust. horizontal 10 kΩ

Condensateurs :
C1 = 220 nF
C2 = 100 nF
C6 = 10 µF/50 V

Semi-conducteurs :
LED1 = LED verte, 5 mm
LED2 = LED rouge, 5 mm
ou LED2 = LED bicolore à cathode commune
T1 = BC547C

Divers :
S1, S2 = bouton-poussoir NO, 6 x 6 mm
MIC1 = microphone à électret, 6 mm
JP1 = embase à 2 picots au pas de 2,54 mm
K2, K3 = embase à 6 picots au pas de 2,54 mm
K4, K5 = embase à 8 picots au pas de 2,54 mm
LCD1 = LCD 2 x 16 caractères, avec éclairage
K6 = embase à 16 picots au pas de 2,54 mm pour LCD : connecteur 16 voies au pas de 2,54 mm
cavalier
circuit imprimé réf. 129009-1

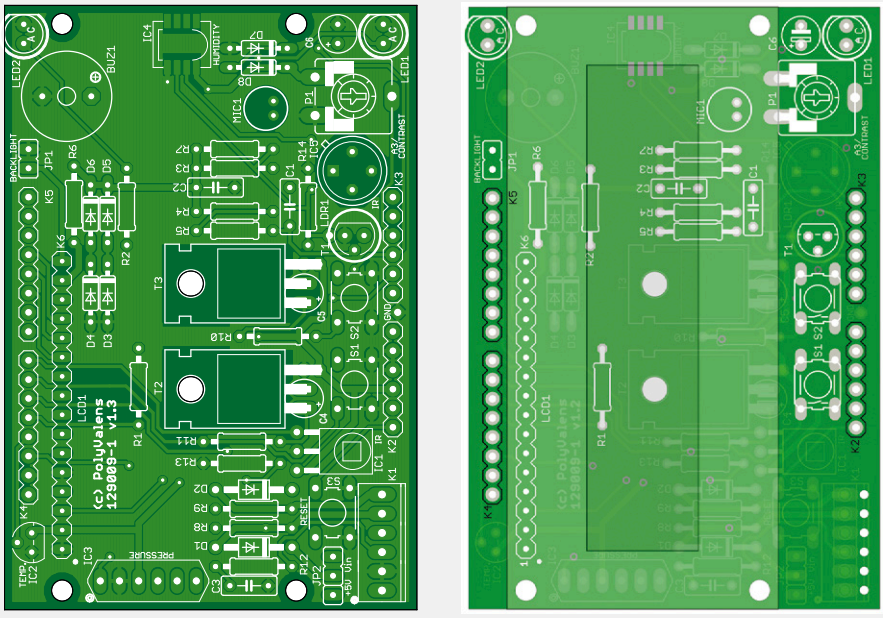


Figure 2. La platine polyvalente peut accueillir différents circuits. Seuls les composants utilisés dans le sonomètre sont représentés.

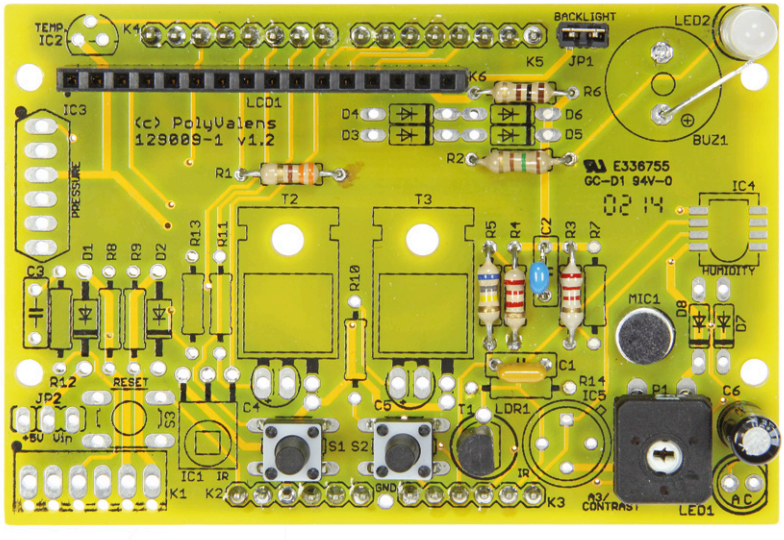


Figure 3. La platine montée, LCD en moins, par souci de clarté.

le système en décibels. Pour réussir à en faire malgré tout un appareil utile, il faut pouvoir en régler soi-même les seuils. Aussi ai-je ajouté au circuit un LCD en mode 4 bits et deux boutons-poussoirs. L'affichage peut indiquer les valeurs mesurées, tant brutes que traitées, les boutons serviront aux choix dans le menu. Le potentiomètre P1 règle le contraste du LCD, le cavalier JP1 branche l'éclairage de fond. Et qui mieux est, tous ces organes s'installent sans difficulté sur le *shield* !

Le logiciel

Le programme, un *sketch* Arduino évidemment, j'ai pris plaisir à l'étoffer pour révéler différentes astuces et techniques. C'est *analogRead* qui mesure le niveau sonore. La routine le fait quatre fois de suite pour en calculer ensuite la moyenne. Le but n'est pas tellement d'effectuer un filtrage du bruit, mais avant tout de ralentir l'exécution. La fonction *analogRead* est assez lente, aussi, la faire tourner à quatre reprises ramène la récurrence de la boucle principale (*loop*) à quelque 2 kHz. Ce sera du même coup le taux d'échantillonnage du signal sonore.
L'amplitude de crête des valeurs de mesure y est déterminée par un intégrateur à fuite, si bien que l'indicateur de niveau sonore suit bien sa variation et redescend tout seul à zéro quand le bruit cesse.

Un filtre passe-bas d'ordre 1 traite la valeur de crête avec une fréquence de coupure d'environ 0,1 Hz en y instaurant une certaine inertie. Pour y arriver, on fait appel à un véritable algorithme de filtre IIR (à réponse impulsionnelle infinie ou RII) à virgule flottante (avec une explication) de manière à voir comment s'y prendre dans ce cas de figure.
Toutes les secondes, on compare la valeur de pointe actuelle filtrée aux seuils définis, puis on montre le résultat sur l'afficheur et les LED. Au cours de cette actualisation, on mesure le niveau sonore analogique, parce qu'il est apparu que la commutation des LED avait un effet sur l'entrée analogique d'Arduino.

Il y a trois seuils pour la mesure du niveau de bruit : vert, orange et rouge. À l'état « éteint », il n'y a que peu ou pas de bruit. Vert, c'est un peu plus fort, avec de la musique, la valeur est de 100. Orange, encore plus fort, pour un chœur chantant, la valeur est de 200. Rouge, c'est très

fort, musique criarde ou bruit assourdissant, valeur 300.
L'afficheur montre le niveau sonore en pourcentage du seuil le plus élevé (rouge). La même valeur est également exprimée sous forme d'une barre horizontale dans la seconde ligne pour pouvoir l'observer à distance, mais aussi pour exposer comment on la programme. L'affichage indique en plus la valeur de crête par un nombre entre 0 et 1 023, plus facile à utiliser pour déterminer la hauteur des seuils.

Pour enregistrer ces seuils, il faut d'abord appuyer en même temps sur les deux boutons, ce qui permet de fixer le seuil vert (0 à 1 023). Appuyer encore une fois sur les deux boutons mène au seuil orange, puis au seuil rouge et finalement à quitter le réglage. Ces niveaux sont alors mémorisés dans l'EEPROM du microcontrôleur.

La construction

Le tracé des pistes sur le *shield* multifonctionnel est visible à la **figure 2**. Vous trouverez les schémas de tous les circuits que vous pouvez installer dessus dans le livre [1]. Sur la figure 2 ne sont représentés que les composants qui participent à l'application décrite ici. Remarquez que les connecteurs K2 à K5 s'installent côté soudures. Il y a une embase à 16 picots destinée à recevoir le connecteur correspondant du LCD. La **figure 4** détaille le raccordement un peu particulier de la LED bicolore. Finalement, le *shield* prêt à l'emploi prend place sur une carte Arduino Uno ; il faut ensuite charger l'Arduino avec le fichier *ino* disponible gratuitement en [2].

Conclusion

Cette platine polyvalente nous a permis de vous dévoiler comment réussir l'implémentation avec Arduino de nombreuses fonctions :

- mesure de niveau sonore
- production d'une alarme
- filtrage numérique
- pilotage de LCD
- commande de LED
- test de l'état d'un interrupteur
- implémentation d'un menu de réglage
- mise en œuvre d'une barre graphique
- usage de caractères personnalisés
- écriture et lecture de valeurs en EEPROM.

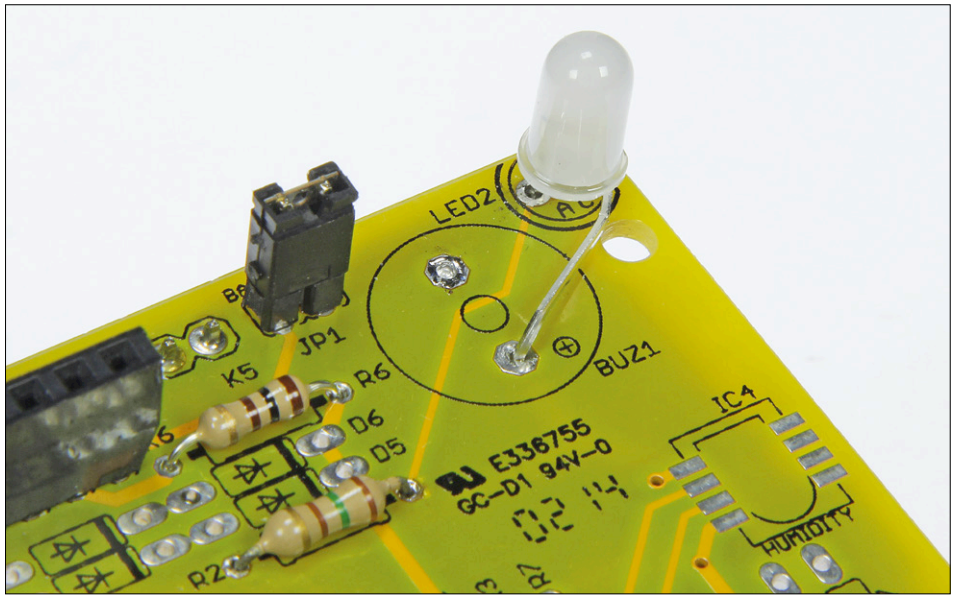


Figure 4. Petite astuce pour la LED bicolore : le fil de la LED verte va au raccord + pour le vibreur BUZ1.

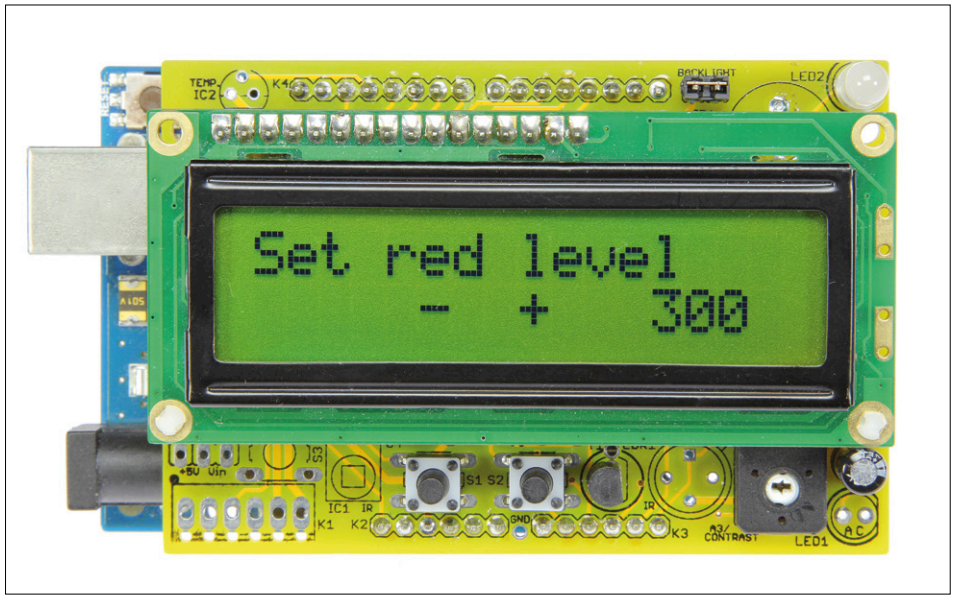


Figure 5. Réglage du niveau auquel la LED rouge doit s'allumer.

Comme exercice complémentaire, je vous propose d'installer un port sériel pour envoyer les mesures sur un PC où elles pourront être horodatées. Pratique comme preuve en cas de tapages nocturnes à répétition ! ⏮

(150242 – version française : Robert Grignard)

Liens

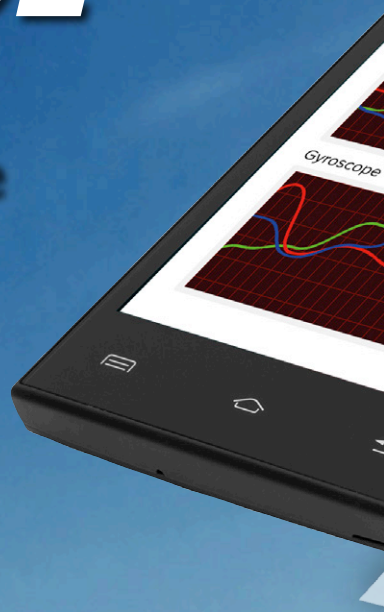
[1] Maîtrisez les microcontrôleurs à l'aide d'Arduino, 384 pages, Publitronic Elektor, ISBN 978-2-86661-195-8, www.elektor.fr/maitrisez-les-microcontroleurs-a-l-aide-d-arduino-b

[2] www.elektormagazine.fr/150242

connectez des objets avec *Genuino 101*

établir la communication entre un circuit électronique et un téléphone

Clemens Valens (labo d'Elektor)



Depuis plusieurs années déjà, tout le monde parle de l'Internet des Objets et des objets connectés ; les observateurs du monde de l'industrie prédisent la connexion de milliards d'appareils dans un futur proche et le marché atteindra des dizaines de milliards de dollars. C'est le moment de prendre le train en marche. Apprenez à connecter un objet.

Appareils connectés et IdO ? Ce sont en général des appareils connectés sans fil qui envoient des données dans le nuage et en reçoivent. « Sans fil » ne désigne pas une technique en particulier, mais *Wi-Fi* et *Bluetooth* sont les techniques les plus répandues. Le *Bluetooth* 4.0 à faible énergie (*Low Energy* ou *BLE*) est tout à fait adapté à la connexion d'appareils à (ultra) faible puissance – par ex. capteurs ou puces à porter – à un appareil plus puissant comme un ordiphone ou une tablette qui sert de point d'accès au nuage.

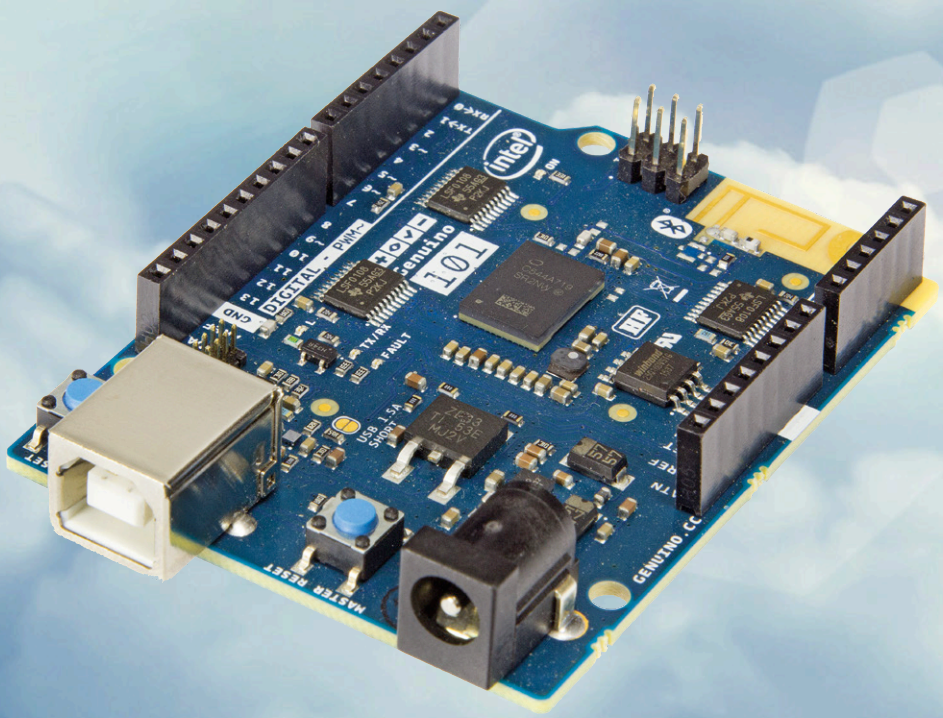
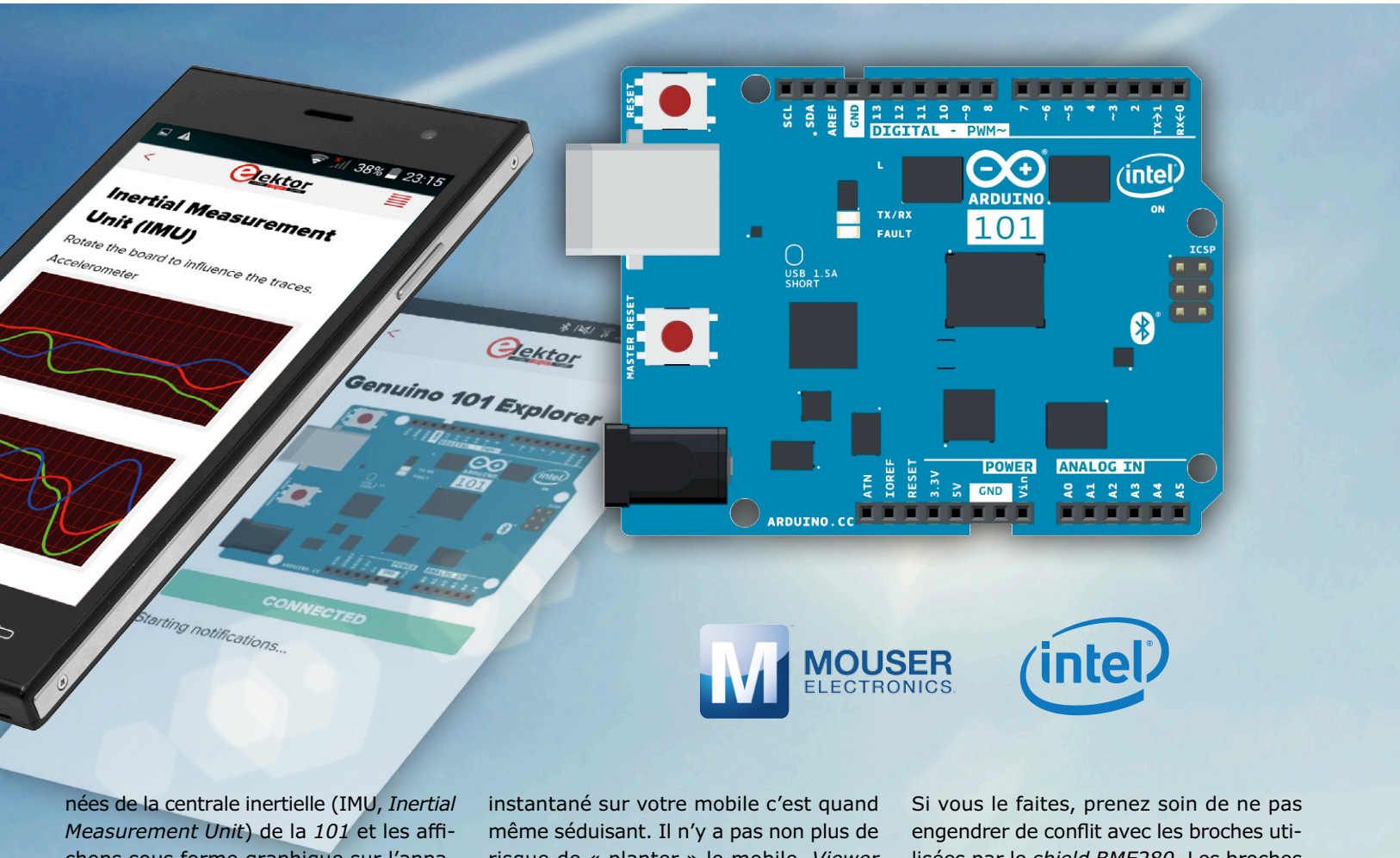


Figure 1. La carte *Arduino/Genuino 101* d'Intel capture les données fournies par sa centrale inertielle à six axes et peut communiquer en Bluetooth faible énergie.

La carte *Genuino 101* d'Intel (*Arduino 101* aux États-Unis) (**fig. 1**) est une carte compatible Arduino, totalement prise en charge par les outils de développement gratuits et très répandus d'Arduino. Comme elle embarque le *BLE*, c'est une excellente base de développement de vos propres objets connectés. Un module *Curie* d'*Intel* l'anime (µC à 32 bits *Quark*, 384 Ko de mémoire Flash, 80 Ko de SRAM, *BLE*, gyroscope-accéléromètre à six axes, et circuit de charge de batterie). Cette carte est dotée de connecteurs d'extension compatibles avec les cartes d'extension (*shields*) Arduino, qui donnent accès à 20 E/S logiques et 6 entrées analogiques.

Vous apprendrez ici à connecter la « *101* » à un appareil mobile, équipé *BLE* sous *Android 4.4* (*KitKat*) ou sup. Officiellement *BLE* est pris en charge depuis *Android 4.3*, mais en raison de sa fiabilité incertaine, mieux vaut ne pas utiliser cette version.

En résumé, nous ajoutons un *shield* équipé d'un capteur *BME280* (mesure de température, pression et humidité relative de l'air) [1]. Nous capturons les don-



nées de la centrale inertielle (IMU, *Inertial Measurement Unit*) de la *101* et les affichons sous forme graphique sur l'appareil mobile [2] ; depuis ce dernier, nous lisons les entrées analogiques de la *101* et en commandons les sorties logiques.
Il faut exécuter une appli sur l'appareil *Android* pour lui connecter une carte *Genuino 101* par *BLE*. Puisque nous avons conçu l'objet connecté selon nos propres spécifications, nous écrivons également cette appli. Il y a plusieurs manières de créer des applis *Android*, mais certaines sont plus aisées. En tant que novices, nous préférons une approche facile : *Evothings*. Il s'agit d'un bon outil qui fournit un cadre d'appli de base dans lequel le développeur n'a plus qu'à ajouter son propre code rédigé dans des langages de développement de pages web bien connus (*JavaScript*, *HTML* et *CSS*). Concevoir une appli dans *Evothings* revient à créer un tout petit site et à l'afficher dans une structure, à savoir *Viewer* d'*Evothings* qui s'exécute sur l'appareil mobile. Comme dans *Evothings* l'appli n'est rien de plus qu'un simple site, on peut la modifier très facilement sans de longues étapes de compilation. Il suffit d'écrire et de modifier du code. Dès la sauvegarde, *Evothings* met à jour l'affichage. Un résultat instantané sur votre mobile c'est quand même séduisant. Il n'y a pas non plus de risque de « planter » le mobile, *Viewer* intercepte les erreurs. Une fois que votre appli vous convient, créez une appli « réelle » et distribuez-la via *Google Play*.

Ça a l'air sympa quand même ? Alors quelle est la marche à suivre ?

Matériel et logiciels nécessaires

- *Genuino 101* ;
- *shield BME280* ;
- PC (Windows, Linux ou OSX) avec au moins un port USB libre, éditeur de texte simple et en option, éditeur d'image (ici Windows 7 édition familiale, SP1, 64 bits) ;
- appareil mobile, équipé *BLE*, sous *Android 4.4* ou sup. (ici Samsung Galaxy J5, Android 5.1.1) ;
- accès à l'internet pour le PC et l'appareil mobile

Assemblage du matériel

C'est simple : branchez le *shield BME280* sur la *101* (**fig. 2**). Bien sûr, vous pouvez ajouter votre propre *shield* avec quelques résistances, LED et potentiomètres pour jouer avec les E/S logiques et analogiques, mais cela reste une option. Si vous le faites, prenez soin de ne pas engendrer de conflit avec les broches utilisées par le *shield BME280*. Les broches logiques 0 à 6 et les entrées analogiques 0 à 3 sont libres.

Installation du logiciel

Côté logiciel, il y a pas mal d'éléments à installer et à configurer avant de pouvoir créer et publier votre appli dans la boutique *Google Play*, mais nous irons au plus simple :

- PC : téléchargez et installez l'environnement de développement interactif Arduino (www.arduino.cc, ici version 1.6.10) ;
- PC : téléchargez et installez l'atelier *Evothings Studio* (evothings.com, ici version 2.0.0) ;
- Appareil mobile : téléchargez et installez *Evothings Viewer* (*Google Play*, ici appli version 1.4.1)

Genuino 101

Pour le moment, ne connectez pas la carte au PC, il faut d'abord installer certains pilotes. Lancez l'EDI Arduino et ouvrez le gestionnaire de carte (Outils → Type de carte → Gestionnaire de carte…). En haut à gauche, choisissez *Arduino Cer-*

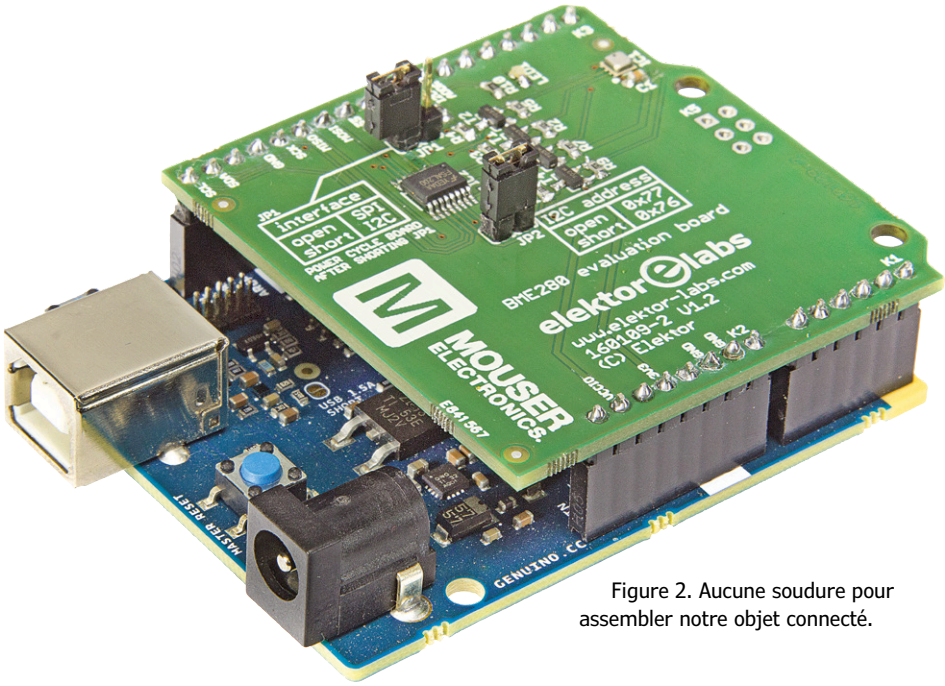


Figure 2. Aucune soudure pour assembler notre objet connecté.

tified pour raccourcir la liste et trouver la carte *Intel Curie*. Cliquez dessus puis sur *Installer*. Le temps d'installation dépend de votre connexion internet. Lorsque c'est fait, refermez le gestionnaire, retournez au menu Outils → Cartes et descendez la liste jusqu'à la carte *Arduino/Genuino 101*. Sélectionnez-la.

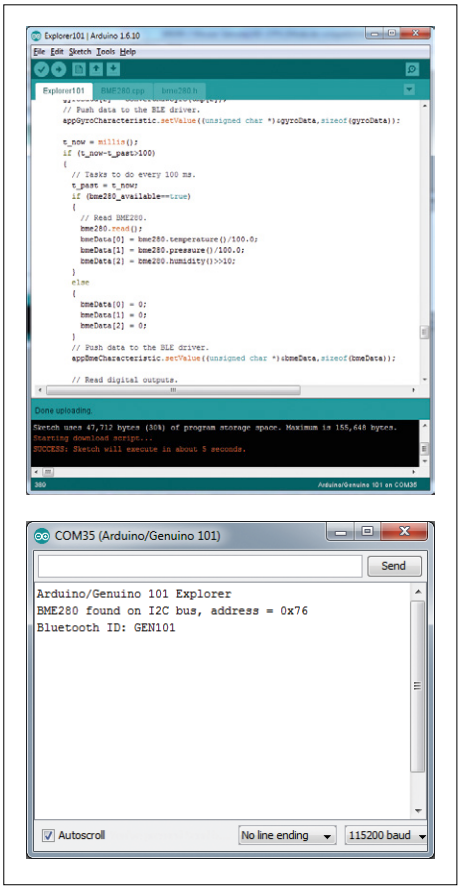


Figure 3. Le croquis s'est chargé normalement et la carte attend maintenant une connexion Bluetooth.

Connectez la carte au PC. Un port série doit apparaître ; sélectionnez-le dans l'EDI Arduino (Outils → Port).

En cas de problèmes lors de l'installation de la *101*, voir le guide d'installation en ligne (www.arduino.cc/en/Guide/Arduino101).

Téléchargez le croquis *BME280* depuis [1]. Le code QR à l'arrière du *shield BME280* contient l'URL (http://bit.ly/2aNNDq7) qui vous y conduira. Téléversez le croquis dans la *101*, attendez quelques secondes et ouvrez le moniteur série de l'EDI Arduino (Outils → Moniteur série). Au bout d'un moment les mesures de l'air ambiant suivantes apparaissent : température, pression et humidité relative. Avant de poursuivre, assurez-vous que le *shield* fonctionne correctement.

Téléchargez le croquis *Genuino 101 Explorer* depuis [1] et téléversez-le dans la *101*. Après quelques secondes, le moniteur série affiche des données indiquant que la *101* attend une connexion Bluetooth (**fig. 3**).

Configuration d'*Evothings*

Lancez l'exécutable de l'atelier *Evothings Studio* (également appelé *Evothings Workbench*). Ouvrez un nouvel onglet et saisissez « *Genuino-101-Explorer* » comme nom de dossier de l'appli. Avant de cliquer sur Create (créer), notez le chemin du dossier où l'appli sera enregistrée. Fermez *Evothings Studio/Workbench*.

Au moyen d'un gestionnaire de fichiers, naviguez jusqu'au dossier de l'appli créé et supprimez tout ce qu'il contient. Depuis [1], téléchargez le projet *Genuino 101 Explorer Evothings* et décompressez-le dans ce dossier. Vous devriez y voir apparaître certains dossiers et fichiers dont les plus importants : `index.htm` et `app.js`. Relancez *Evothings Workbench* et ouvrez l'onglet MyApps. Vous devriez y voir l'entrée *Arduino/Genuino 101 Explorer*.

Sur votre mobile, lancez le module d'affichage, c.-à-d. l'appli *Evothings Viewer* (**fig. 4**). Elle demande une clé de connexion produite par *Evothings Workbench* : cliquez sur l'onglet de connexion (Connect) puis sur l'onglet d'obtention de la clé Get key. Saisissez la clé dans *Viewer* et cliquez sur Connect. Pour que ça marche, le PC et le mobile doivent être connectés à l'internet. Si la connexion fonctionne, *Viewer* indique ce qu'il faut faire : ouvrez l'onglet *MyApps* dans le *Workbench* et cliquez sur le bouton d'exécution Run de l'appli *Arduino/Genuino 101 Explorer*. En le faisant, vous observerez l'icône de chargement qui apparaît dans *Viewer* et l'appli s'ouvre peu après.

Si la connexion à l'internet entre votre mobile et votre PC est de bonne qualité et stable, la connexion entre *Viewer* et *Workbench* perdurera. Cependant, s'il se produit des erreurs de chargement inexpliquées vous aurez intérêt à rétablir une connexion mobile/PC avec une nouvelle clé.

Faites l'essai

Si la *101* est déconnectée du PC, rebranchez-la (ou alimentez-la indépendamment). Sinon, appuyez sur le bouton de réinitialisation (*Reset* et non pas *Master Reset*). Accordez-lui env. 10 s pour démarrer. Avec le moniteur série, assurez-vous qu'elle est en attente de connexion, puis dans *Viewer* sur votre mobile, cliquez sur le bouton rouge de connexion (Connect). Si le *Bluetooth* n'était pas encore activé sur votre mobile, il vous demande maintenant l'autorisation de le faire. Acceptez et observez les messages d'état qui apparaissent sous le bouton de connexion. Si tout se passe bien, quelques secondes plus tard, le bouton passe au vert. Tapez sur le bouton menu dans le coin supérieur droit et sélectionnez une vue de données (**fig. 5**).

Du côté du croquis

L'environnement est fonctionnel, il est temps d'examiner les différents composants du système. Le croquis sur la *101*

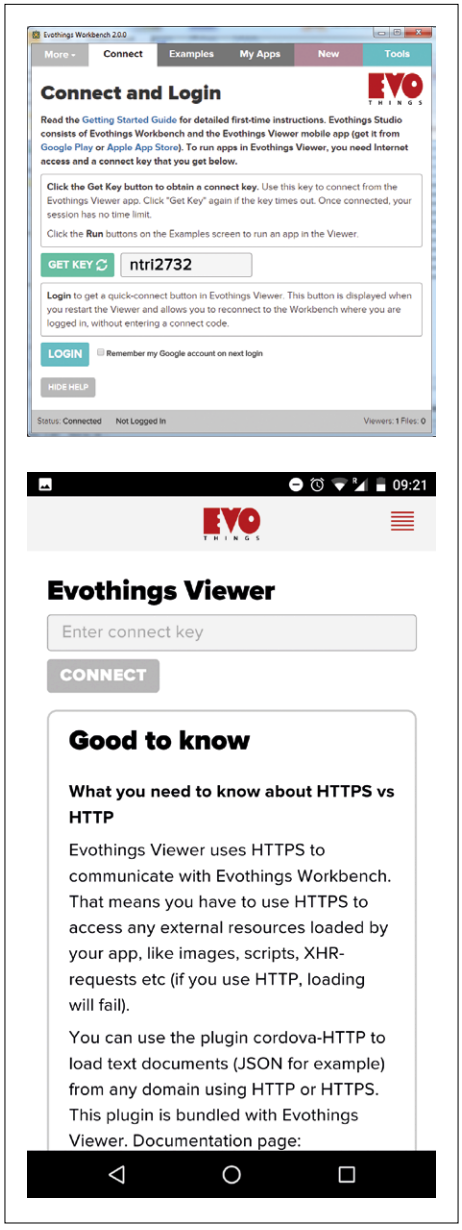


Figure 4. L'atelier *(Workbench) Evothings* sur le PC et son module d'affichage *Viewer* sur l'appareil mobile.

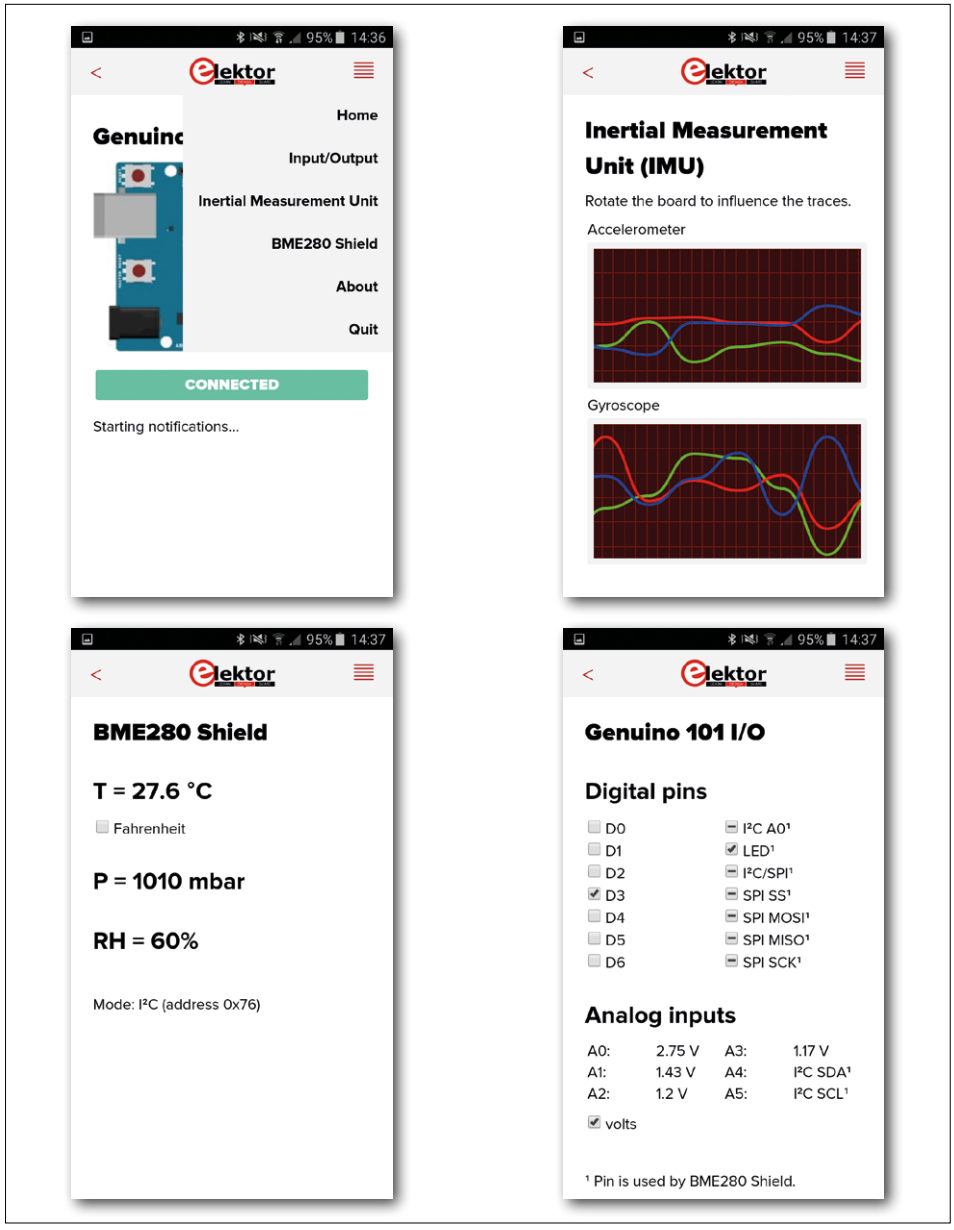


Figure 5. Copies d'écran de l'appli montrant les différentes vues des données.

repose sur les exemples *CurieBLE* et *CurieIMU* (EDI Arduino : Fichiers → Exemples) auxquels j'ai ajouté le pilote *BME280*. Nous considérons ici que l'interfaçage matériel est acquis pour nous intéresser à ce qui touche le *BLE*.

Le BLE utilise des services identifiés chacun par un identifiant (ID) unique. Certains de ces services sont définis par le groupe *Bluetooth SIG* (www.bluetooth.com), mais comme notre service météo IMU n'existe pas encore, nous devons créer pour lui un numéro unique. À cet effet, on produit un identifiant universel unique (*UUID*), par ex. au moyen d'un générateur d'*UUID* en ligne. Ensuite pour chaque type d'information devant être reçu ou envoyé, il faut ajouter un attribut au service, appelé caractéristique. Chaque caractéristique dispose de son ID unique. Les caractéristiques peuvent être en lecture seule (*read-only*), écriture seule (*write only*) ou lecture-écriture (*read-write*) et elles peuvent produire des notifications.

En théorie, une caractéristique peut renfermer jusqu'à 512 octets, mais la bibliothèque *Genuino 101 BLE* limite ce nombre à 20. Notre application délivre trois valeurs flottantes codées sur quatre octets pour l'accéléromètre (c.-à-d. 12 octets au total), idem pour le gyroscope et pour le BME280. Les six données analogiques sont codées sur deux octets (encore 12 octets) et les données logiques sur deux octets. Total : 50 octets, soit trois caractéristiques, mais afin de simplifier et d'éviter un multiplexage compliqué, il y a une caractéristique pour chaque type de donnée.

Une fois les caractéristiques définies, initialisées et ajoutées comme attributs au service, le périphérique *BLE* peut démarrer. Sans connexion, il ne se passe pas rien ; une fois la connexion établie, nous plaçons des données dans les caractéristiques sortantes et lisons les données dans les caractéristiques entrantes. Il n'y a rien d'autre à faire, la bibliothèque prend en charge la partie délicate de la programmation.

Le flux des données provenant de l'IMU

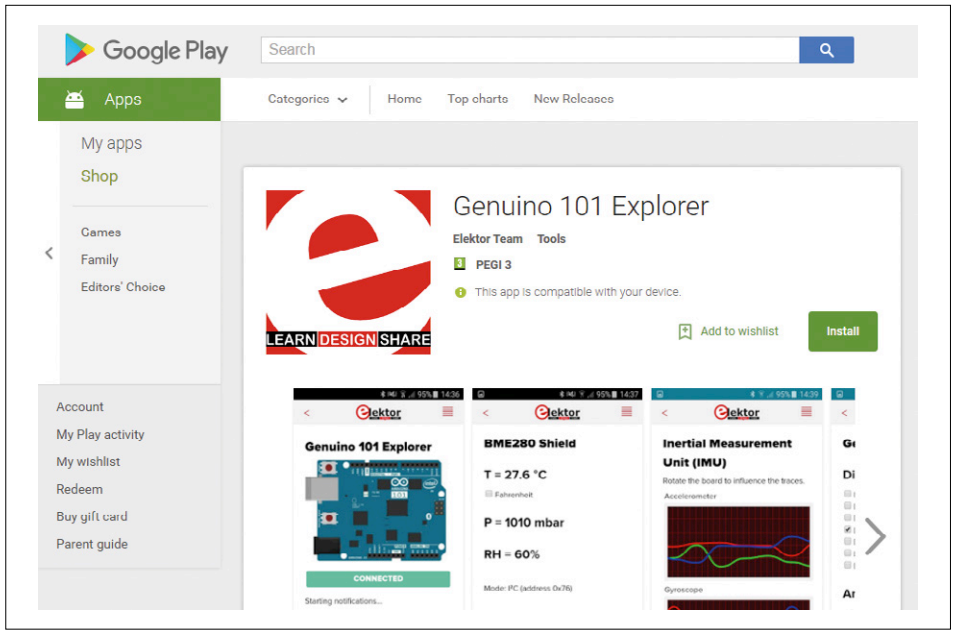


Figure 6. Notre propre appli dans la boutique *Google Play*, nous voici devenus éditeurs *IoT* !

est transmis vers le mobile le plus vite possible ; les autres données sont envoyées toutes les 100 ms.
Il y a deux manières de lire les données d'une caractéristique : soit en l'interrogeant avec la fonction `written()` ou de manière événementielle en déclarant un auditeur (`listener`). Les deux manières sont illustrées pour les caractéristiques des E/S logiques dans le croquis *Genuino 101 Explorer*.
On peut donner un nom (local) au service, ici « GEN101 », mais il faut prendre soin de choisir un nom assez court. En effet, il est stocké dans l'espace libre du tampon de diffusion, et il y en a peu. Si le nom local déborde, la bibliothèque va produire son propre nom empêchant ainsi la détection automatique de l'appli dans le processus.

Du côté de l'appli

Côté appli, les choses sont très similaires. Le code à tout faire de l'appli réside dans le fichier JavaScript `app.js` ; l'interface utilisateur (*UI*) est définie par le fichier HTML `index.htm` et la feuille de style `ui\css\evothings-app.css` détermine la souplesse de comportement des éléments HTML.
Comme la *101*, *Evothings* dispose d'une bibliothèque *BLE* assez fournie et d'un coup de baguette magique notre appli parle *Bluetoth*. Les ID uniques créés pour le croquis *Genuino 101* doivent être connus de l'appli pour qu'elle puisse accéder au service et à ses caractéristiques. Comme les caractéristiques parviennent de façon asynchrone, l'appli active pour celles-ci des notifications de réception afin de traiter les données au fur et à mesure qu'elles arrivent.
En dehors des E/S analogiques et logiques qui s'affichent immédiatement, le reste des données est enregistré avant d'être affiché. Des compteurs d'intervalle sont définis pour les graphiques et les données du *BME280*. Les graphes sont mis à jour relativement lentement de sorte que les mouvements brusques de la carte et le bruit sont filtrés et apparaissent bien lissés. Ce lissage est également dû à la bibliothèque utilisée pour créer des graphiques : *Smoothie Charts* (smoothiecharts.org/).
La communication entre l'UI en *HTML* et l'appli en *JavaScript* est bidirectionnelle. Les fichiers `app.js` et `index.htm` font un usage intensif de la fonction `getElementById()`. Pour que cela fonctionne, l'élément HTML à modifier doit avoir un identifiant. La propriété *innerHTML* permet de ré-écrire le texte à l'intérieur de l'élément (le code HTML est également autorisé, ce qui permet de modifier la page dynamiquement), et avec les méthodes `getAttribute` et `setAttribute`, vous pouvez interroger et modifier les attributs de l'élément, par ex. changer sa couleur ou sa taille.
En retour, le code HTML peut appeler des fonctions dans `app.js` quand l'utilisateur clique sur un bouton ou une rubrique de menu ou bien inverser une case à cocher en déclarant une fonction *JavaScript* pour l'attribut `onclick` de l'élément, par ex. `onclick="app.onStartButton()"`.
Les codes *JavaScript* et HTML peuvent bien entendu se connecter à l'internet. C'est le cas de la page *About* de l'appli dans laquelle quelques liens de base permettent à l'utilisateur d'ouvrir des pages internet. Avec un peu plus de code, vous vous connecterez à des services de stockage de données en ligne, Twitter ou que sais-je encore et publierez vos données à l'échelle d'internet.

Transformation en appli Android

Votre appli est prête à être transformée en une véritable appli *Android* ? Compilez-la ! Nota : il est possible de compiler l'appli pour *Android* ainsi que pour *iOS* (mais pas sous Windows). C'est facile avec la chaîne d'outils *Apache Cordova* (cordova.apache.org/), mais son installation nécessite une quantité considérable de logiciels :

- *Node.js* (ici version v4.4.7)
- *NPM* (ici version v2.15.8)
- *Git* (ici version 2.6.3.windows.1)
- *Cordova* (ici version v6.3.1)
- *Java JDK* (ici version jdk1.8.0_101_x64)
- *Android SDK* (ici version v25.1.7 + API 24 + API 23 + API 19)

Voir la procédure détaillée, disponible sur l'internet : evothings.com/doc/build/build-overview.html#Install.
Donc si *Cordova* est installé sur votre machine ; voici la recette pour construire une appli à partir de votre projet *Evothings*. Il faut commencer par créer un projet *Cordova* :

- créer un dossier de travail et ouvrir une interface de ligne de commande (ILC) ;
- dans ce dossier, saisir la commande `cordova create [mon_projet] [com.elektor.labs.mon_appli] [nom_de_mon_appli]` dans laquelle on remplace tout ce qui est entre [] par des noms au choix. Nota : le nom avec les points est censé être une URL inversée (vous pouvez en créer une) ;
- dans l'ILC, aller dans le dossier tout juste créé : `cd [mon_projet]` ;
- supprimer tout ce qui se trouve dans le sous-dossier « www », mais ne pas supprimer ce sous-dossier ;
- copier le projet *Evothings* dans le dossier « www », c.-à-d. tous les fichiers et tous les sous-dossiers qui

accompagnent `app.js` et `index.htm` ;
- dans l'ILC, saisir la commande **`cordova plugin add cordova-plugin-ble`** pour ajouter la bibliothèque *BLE* au projet. Répéter cette étape pour tout autre *plugin* nécessaire ;
- dans l'ILC, saisir la commande **`cordova platform add android`**. Pour *iOS*, utiliser **`cordova platform add ios`**. Il es possible d'ajouter plus d'une plateforme.

Maintenant les commandes suivantes permettent de construire le projet depuis l'ILC :

```
cordova build android
cordova build ios (ne fonctionne pas sous Windows)
```

Normalement, la construction de l'appli doit se terminer sans avertissement ni message d'erreur et le résultat doit se trouver sur *platforms/android/build/outputs/apk*. Pour tester l'appli, copiez le fichier *.APK* sur votre mobile et installez-le à l'aide du gestionnaire de fichiers. Il faut que votre appareil autorise les sources inconnues (paramètres de sécurité).

Pour que *Google Play* publie votre appli, vous devez la recompiler en mode diffusion (*release*) et la signer. À cet effet, commencez par créer une clé (commande en ligne) :

```
keytool -genkey -v -keystore [Nom].keystore -alias [Alias] -keyalg RSA -keysize 2048 -validity 10000
```

Où vous choisissez `[Nom]` et `[Alias]`. Remplissez les informations demandées et notez-les soigneusement pour pouvoir vous y référer plus tard. Ensuite exécutez l'appli (de nouveau avec la ligne de commande, faites attention aux tirets doubles) :

```
cordova build android --release -- --keystore=»[Chemin]\[Nom].keystore» --storePassword=[Mot_de_passe] --alias=[Alias]
```

Où `[Nom]` et `[Alias]` sont les mêmes que précédemment, `[Chemin]` est le chemin du nouveau fichier contenant la clé, et `[Mot_de_passe]` le mot de passe que vous venez de demander à *keytool* de créer. Une petite fenêtre contextuelle demande alors votre mot de passe ; entrez-le et cliquez sur OK. Cela produit un fichier *.APK* baptisé « *android-release* ».
Si ce n'est déjà fait, ouvrez un compte d'éditeur sur *Google Play* (redevance unique de 25 $), téléversez le fichier *.APK*, remplissez les formulaires, publiez, et voilà, vous êtes devenu éditeur de logiciels ! (**fig. 6**)

Pièges à éviter

Votre appli *Android* pourrait (peut ?) ne pas fonctionner comme espéré, c.-à-d. comme dans *Evothings Viewer*. Vous n'avez sans doute pas résolu les problèmes pris en charge silencieusement par *Viewer*. Pour trouver ce qui ne va pas, utilisez la console *JavaScript* dans l'environnement *Evothings Workbench*, onglet *Tools*, et exécutez votre appli dans *Viewer*. Tout ce qui n'est pas référencé ni pris en charge, etc. doit être corrigé.
Gardez présent à l'esprit que les *timeouts* (temps impartis) peuvent être trop courts pour les situations réelles. Souvenez-vous aussi que les choses fonctionnent différemment lorsque le mobile est connecté à l'objet. La pile Bluetooth peut bloquer quelque chose. Peut-être devez-vous fermer la connexion avant de pouvoir faire ce que vous voulez faire.
Si votre appli ne ressemble pas à ce que vous souhaitez, ouvrez le fichier `index.htm` dans un navigateur de votre ordinateur, il doit apparaître comme dans *Viewer* (mais en plus grand). Si ce n'est pas le cas, tentez de trouver la raison et corrigez-le.
Encore un mot sur *BLE*. La manipulation d'une connexion *BLE* est quelque peu difficile, car il est presque impossible de sortir totalement d'une appli. *Android* maintient son exécution en arrière-plan et elle peut rester (partiellement) connectée. Pour sortir de ce type de piège, il faut « tuer » l'appli dans le gestionnaire d'application du mobile et par précaution, redémarrer l'objet connecté.

Conclusion

Dans cet article, nous avons vu comment connecter une carte *Genuino 101* à un appareil (mobile) compatible Bluetooth *Low Energy* – par ex. ordiphone ou tablette – uniquement en utilisant des outils gratuits à code ouvert. Un *shield BME280* de mesure de température, pression et humidité relative de l'air a été ajouté. Une fois un service *BLE* spécifique et ses diverses caractéristiques définis, nous échangeons toutes sortes de données entre les deux appareils connectés. De cette façon, des données IdO typiques (accéléromètre, gyroscope et informations météo) sont transférées vers l'appareil mobile. Une interface utilisateur s'exécutant sur l'appareil mobile sous forme d'une appli native permet d'afficher les données, d'interagir avec l'utilisateur et s'occupe de l'échange bidirectionnel de données stockées dans le nuage. Enfin nous publions l'appli dans *Google Play* afin que chacun puisse en disposer. ⏮

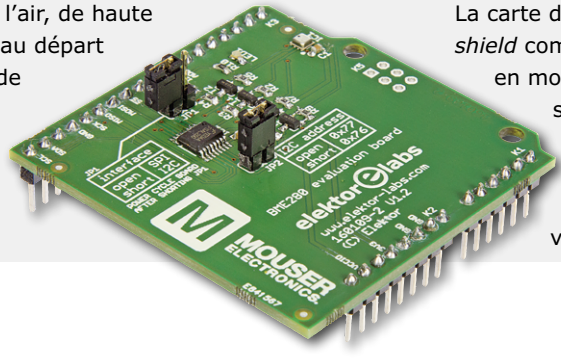
(160109-2)

Liens

[1] www.elektormagazine.com/labs/bme280-evaluation-board-160109-2
[2] www.hackster.io/gov/imu-to-you-ae53e1

Carte d'évaluation BME280

Le capteur *Sensortec BME280* de *Bosch* est un dispositif de mesure de la température, de la pression et de l'humidité relative de l'air, de haute précision. Bien qu'il soit au départ conçu pour la détection de proximité du visage et de la main et la navigation en intérieur sur des appareils mobiles, il s'accommode très bien du rôle de station météorologique. Ce dispositif peut fonctionner aussi bien sur un bus *SPI* que sur un bus *I²C*.



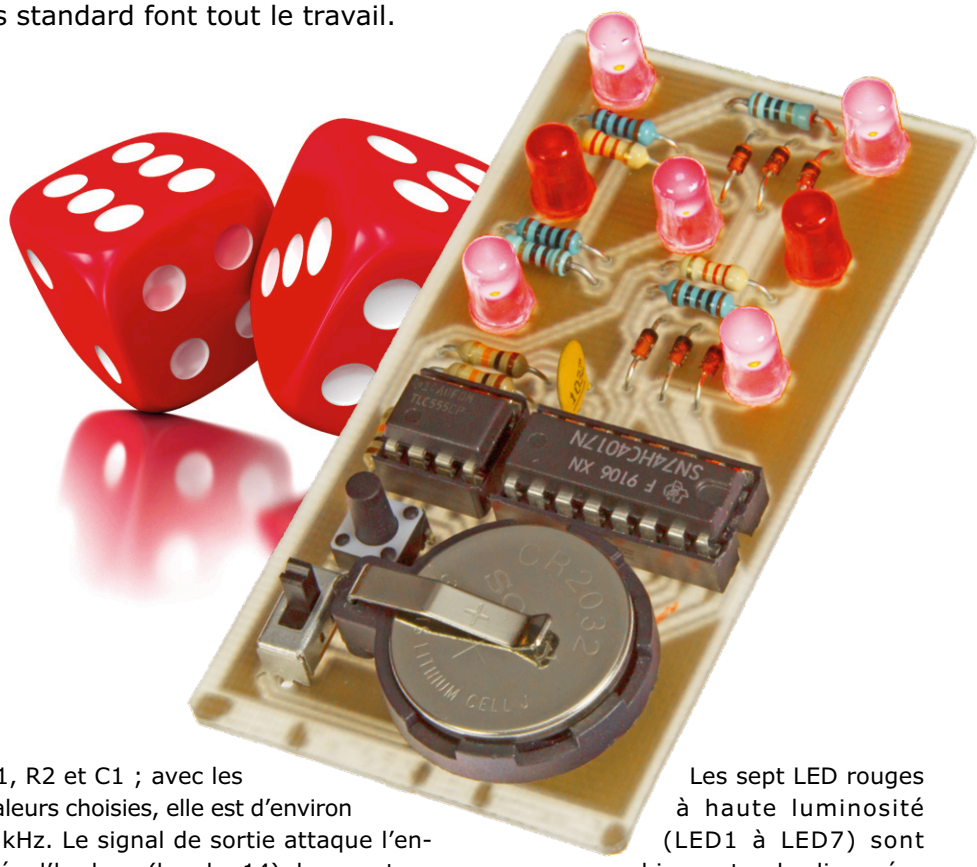
La carte d'évaluation *160109-2 BME280* d'*Elektor* est un *shield* compatible *Arduino*. Un cavalier permet son utilisation en mode *SPI* ou *I²C*. Un second cavalier permet de sélectionner l'une des deux adresses *I²C* disponibles. L'hôte peut détecter le mode et l'adresse *I²C* sélectionnés et il peut également les programmer. Une LED utilisateur est disponible pour un retour visuel d'information.

un dé ultrasimple

sans microcontrôleur !

Roy Aarts (labo d'Elektor)

Le circuit de ce dé électronique très simple a été spécifiquement conçu pour les débutants. Le circuit imprimé est aéré, il n'y a pas beaucoup de composants, ils sont tous traversants, et il n'y a pas de microcontrôleur. Deux circuits intégrés standard font tout le travail.



Depuis des décennies, la réalisation d'un dé est un sujet de choix pour des petits montages électroniques, et Elektor en a publié pas mal au fil des ans. C'était même devenu un véritable sport que de concevoir un circuit avec le moins de composants possible ; cela a entraîné bien sûr quelques carences, comme une représentation peu fidèle du dé : on obtenait bien un résultat exploitable, mais la configuration des LED n'était pas la même que celle d'un vrai dé. Nous avions même réussi à en faire un avec un seul circuit intégré logique de la série 4000, et avec commande tactile ; le seul « inconvénient » était l'alimentation par pile de 9 V, assez encombrante et chère.

Pour ce projet, nous avons opté pour un circuit simple avec des LED, dont le fonctionnement est facile à comprendre, et qui ne présente aucune difficulté d'assemblage. L'alimentation est fournie par une pile bouton CR2032, installée dans un support sur le circuit imprimé ; elle sera donc facile à remplacer, et le circuit est autonome et portable. C'est un montage très attractif, parfait pour initier quelqu'un à l'électronique, même s'il (ou elle) n'a jamais touché un fer à souder ; avec un minimum d'aide, tout un chacun peut le réaliser !

Le circuit

Le circuit est repris en **figure 1** : il est constitué d'un oscillateur (IC2) et d'un compteur décimal (IC1), dont les sorties commandent sept LED.

Pour IC2 nous avons choisi le 555, bien connu et configuré ici en multivibrateur astable : les broches 2 et 6 (respectivement *Treshold* – seuil, et *Trigger* – gâchette) sont connectées ensemble. La fréquence d'oscillation est déterminée par R1, R2 et C1 ; avec les valeurs choisies, elle est d'environ 5 kHz. Le signal de sortie attaque l'entrée d'horloge (broche 14) du compteur IC1, un 74HC4017. En connectant la sortie Q6 (broche 5) à l'entrée de remise à zéro (*Reset*, broche 15), le compteur ne compte que de 1 à 6 ; c'est bien ce dont nous avons besoin pour simuler un dé.

Les impulsions à l'entrée d'horloge du compteur ne le font « tourner » que lorsque l'entrée de validation (*Enable*, broche 14) est au niveau bas. Si on y connecte un bouton-poussoir (S1) et une résistance de tirage, on pourra faire fonctionner le compteur en appuyant sur S1. Lorsque nous le relâchons, une des sorties Q1 à Q5 et la sortie de la retenue (*Carry Out*, broche 12) seront au niveau haut. Comme la fréquence du signal d'horloge délivré par le 555 est assez élevée, il n'est pratiquement pas possible de déterminer quelle sortie sera à l'état haut avec S1 ; c'est donc relativement aléatoire !

Les sept LED rouges à haute luminosité (LED1 à LED7) sont bien entendu disposées de manière à représenter un dé. Elles sont connectées suivant une configuration particulière aux sorties du compteur, via des résistances série de limitation du courant et des diodes Schottky. Ces dernières empêchent le court-circuit d'une sortie active par une autre qui ne l'est pas. La sortie de la retenue est à l'état haut lors d'une remise à zéro et lorsque les sorties Q0 à Q4 sont actives ; en combinaison avec les diodes, nous obtenons donc les nombres suivant : broche 2 haute, 3 ; broche 4 haute, 4 ; broche 7 haute, 5 ; broche 10 haute, 6 ; broche 1 haute, 1 ; broche 12 haute, 2.

Nous avons choisi des diodes Schottky pour minimiser la chute de tension à leurs bornes, vu la faible tension d'alimentation (3 V). Nous avons aussi dû expérimenter pour trouver une valeur correcte des

résistances de limitation du courant, car les sorties du 74HC4017 ne peuvent débiter que peu de courant avec cette tension d'alimentation, et lorsque le courant demandé augmente, la tension de sortie chute. La valeur des résistances est déterminée de telle sorte que la luminosité des LED, commandées simultanément par une sortie, soit sensiblement la même.
L'alimentation de 3 V est fournie par une pile bouton au lithium ; elle est placée dans un support sur le circuit imprimé, pour permettre un remplacement aisé. Les LED ne consommant que quelques mA, la pile devrait autoriser le fonctionnement durant plusieurs heures. L'interrupteur S2 permet la mise sous tension et l'arrêt.

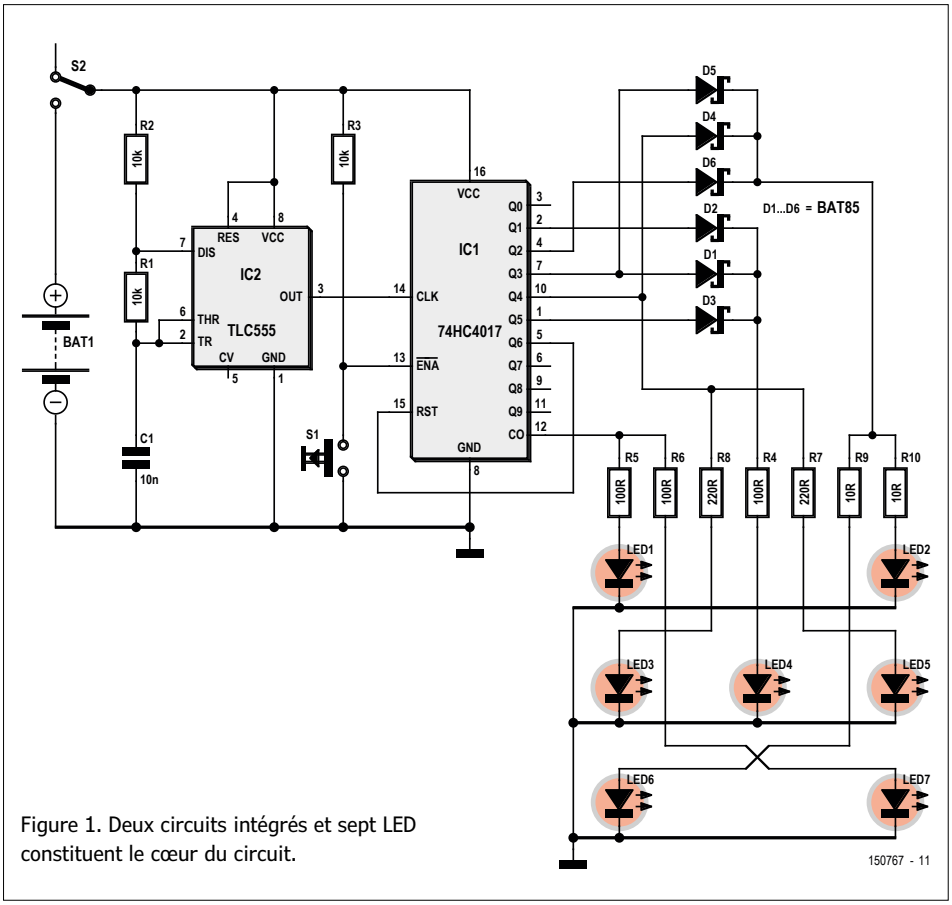


Figure 1. Deux circuits intégrés et sept LED constituent le cœur du circuit.

Montage

Le circuit imprimé, représenté à la **figure 2**, est très aéré ; les pistes et pastilles sont plus larges que d'habitude, un débutant n'aura donc aucune difficulté à souder les composants. Le circuit imprimé est disponible chez Elektor, de même qu'un kit, avec tous les composants et à un prix abordable, pour ceux qui préfèrent cette solution.

Le montage n'est pas compliqué, mais si vous êtes un néophyte en la matière, quelques points devront sans doute être éclaircis. Nous n'allons pas tout expliquer (comme le code de couleur des résistances), alors n'hésitez surtout pas à vous faire aider par quelqu'un de plus expérimenté.

Commencez toujours par les résistances, suivies des diodes Schottky, et du condensateur. La bande argentée sur le corps de la diode correspond à la cathode, qui est aussi représentée sur le circuit imprimé, pour éviter un montage erroné. Soudez ensuite les supports des circuits intégrés et les LED (le méplat correspond à la cathode, qui possède aussi la patte la plus courte ; il est aussi dessiné sur le circuit imprimé). Il ne vous restera plus que l'interrupteur, le bouton-poussoir et le support de la pile. Pour terminer, placez les circuits intégrés sur leur support respectif, en faisant attention à l'orientation (l'encoche sur le boîtier est dessinée sur le circuit imprimé ; s'il n'y a pas d'encoche, un point de couleur ou en creux indique la broche 1). Voilà, c'est tout !
Vérifiez encore bien si tout est à la bonne place et dans le bon sens. Si c'est le cas, insérez la pile dans son support, mettez en marche, et appuyez sur le bouton-poussoir. Un nombre de 1 à 6 devrait être représenté par les LED illuminées.

Félicitations pour votre premier (?) montage réussi ! ⏮

(150767 – version française : Jean-Louis Mehren)

Liste des composants

Résistances (0,25 W, 5%)
R1, R2, R3 = 10 kΩ
R4, R5, R6 = 100 Ω
R7, R8 = 22 Ω
R9, R10 = 10 Ω

Condensateur
C1 = 10 nF, pas de 2,54 mm

Semi-conducteurs
D1-D6 = BAT85 (diode Schottky)
LED1-LED7 = LED rouge, haute luminosité, 5 mm
IC1 = 74HC4017
IC2 = TLC555

Divers
S1 = bouton-poussoir pour circuit imprimé, 6 mm × 6 mm
S2 = interrupteur à glissière, pour circuit imprimé (p. ex. C&K OS102011MS2QN1C)
BAT1 = support de pile CR2032, pour circuit imprimé (p. ex. Multicomp CH25-2032LF)
Pile bouton CR2032
Circuit imprimé réf. 150767-1
ou kit complet réf. 150767-71

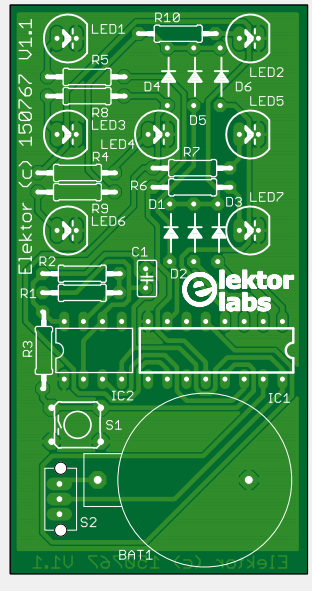


Figure 2. Tous les composants sont sur le circuit imprimé, même la pile et l'interrupteur marche/arrêt.

nouvelles lames pour le Swiss Pi

exemples de programme

Peter S'heeren & **Ilse Joostens** (Belgique)

La carte d'extension Swiss Pi [1] dote le célèbre ordinateur mono-carte Raspberry Pi d'une foule de fonctions utiles. Après la présentation de ce couteau suisse pour RPi (Elektor, 09/2016), nous nous intéressons au serveur Swiss. C'est l'occasion de partager avec vous des exemples de programme écrits en Python et PHP.

C'est toute une série de fonctions que le Swiss Pi ajoute au Raspberry Pi : lignes GPIO, canaux MLI, commande de servo-moteurs, bus RS-485, canaux de CA/N, horloge en temps réel, etc. Pour que l'utilisateur puisse accéder facilement à ces fonctions et même les faire tourner en parallèle, une bonne complicité logicielle entre Swiss Pi et Raspberry Pi est indispensable. On peut comparer le serveur Swiss à un serveur http par ex., puisqu'il permet aux clients, comme à l'utilisateur et à d'autres programmes, de commander la carte avec des instructions. Plusieurs clients peuvent accéder simultanément à Swiss Pi.

Vous apprendrez ici à mettre le serveur en service, puis, comme client, à interagir avec lui pour piloter le Swiss Pi. Vous verrez aussi comment des programmes écrits en Python ou PHP peuvent communiquer avec le serveur pour commander le Swiss Pi.

Le logiciel

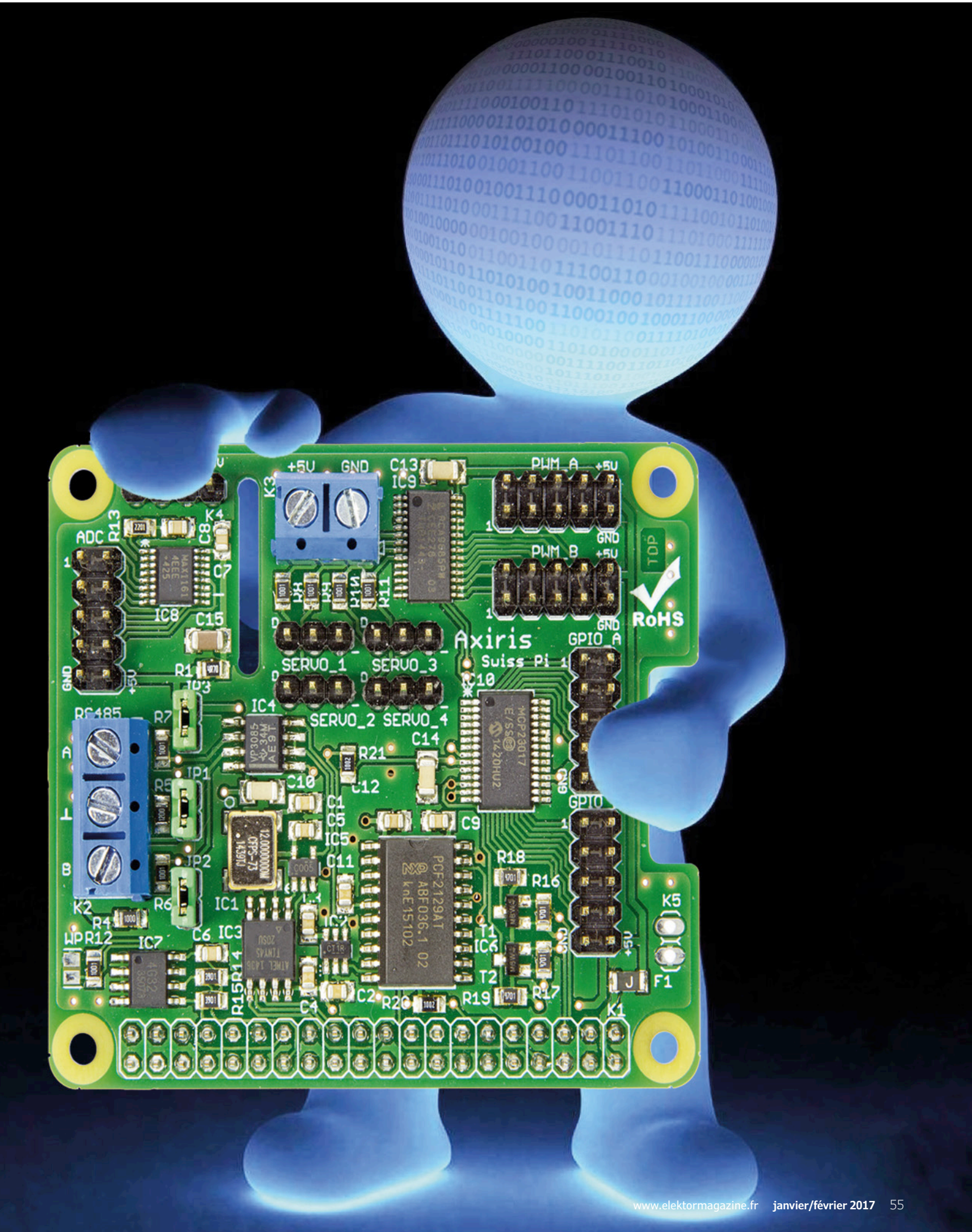
La page du produit de Swiss Pi [2] contient de la documentation et un lien vers la page du logiciel [3]. Cette page met à disposition le code source du serveur Swiss, des programmes en Python et PHP, et la documentation afférente.

Le serveur Swiss

Le programme *Swiss Server* offre les fonctions de Swiss Pi sur des ports de réseau et des E/S normales selon un protocole de communication qui repose sur l'échange de commandes et des réponses correspondantes. Un client envoie une tâche au serveur qui lui répond dès qu'elle a été exécutée. Un client peut envoyer un chapelet de tâches pour accélérer considérablement le déroulement.

Le protocole de communication est codé en ASCII. Sa syntaxe facilite l'usage tant manuel que programmé. Le protocole complet est décrit dans la documentation du logiciel [3].

L'hôte primaire du Swiss Pi est un RPi. Si vous combinez le Swiss Pi avec un AxiCat (**fig. 1**) [4], vous pourrez utiliser un PC sous Linux ou Windows comme hôte. Mais pour cet article-ci, l'hôte sera un Raspberry Pi.


+5V
GND
C13
IC9
K3
PWM_A
+5V
GND
PWM_B
+5V
GND
TOP
RoHS
Axiris
Suiss Pi
GPIO_A
GPIO
ADC
R13
K4
C8
C7
IC8
C15
R1
D3
SERVO_1
SERVO_3
SERVO_2
SERVO_4
IC4
C10
C14
R21
C12
C1
C5
C9
IC5
C11
R18
R16
RS485
R7
IP1
R5
IP2
R6
K2
R4
IC1
IC3
C6
IC7
WP
R12
R14
R15
C4
C2
R20
R19
R17
T1
IC6
T2
K5
F1
J
K1
MCP23017
E/SS
1420HU2
PCF2129AT
ABF036.1 02
knE15102
VP3085
34M
AE9T

Swiss Pi offre une liaison directe entre son bus RS-485 et l'UART de l'hôte

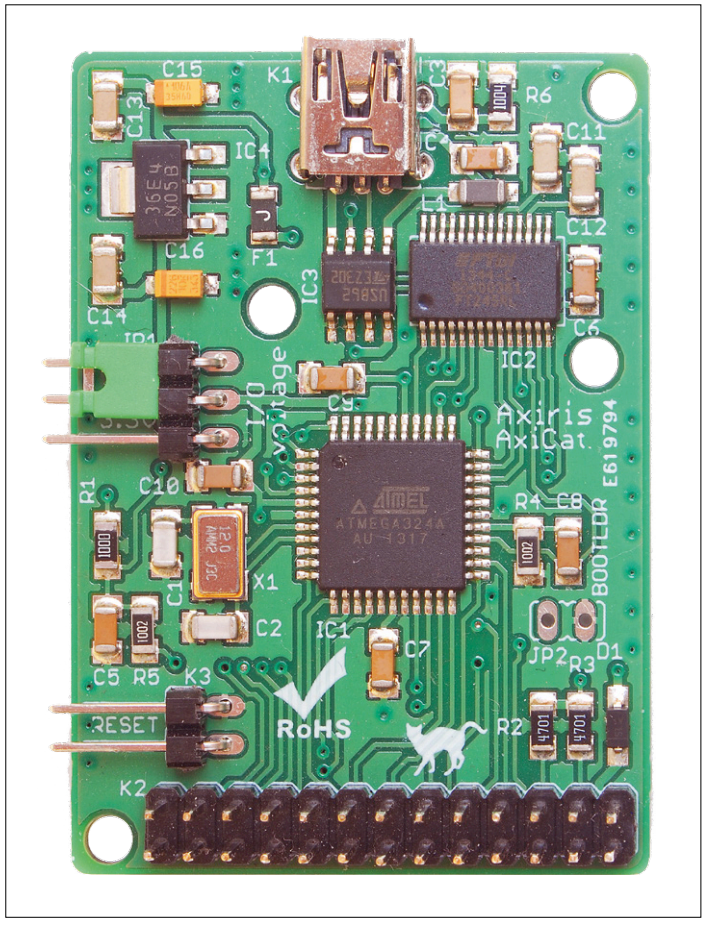


Figure 1. Besoin d'une interface I²C ou SPI pour votre PC sous Linux ou Windows ? Prenez une AxiCat !

Démarrage

Exemple de lancement du serveur avec un RPi comme hôte :

```
./swissserver -v -i2cdev /dev/i2c-1 -p 5003

./swissserver -v -bscdetect -spidev /dev/spidev0.0
   -serial /dev/ttyS0 -rs485 spi -crlf -p 5003 -stdio
   -sp 5004
```

Exemple de lancement du serveur avec AxiCat et Windows comme hôte :

```
swissserver.exe -v -console -axicat \\.\COM12 -rs485
   spi -crlf -p 5003 -stdio -sp 5004
```

Le paramètre `-v`, de *verbose*, signifie bavard. Le serveur montre alors les noms des interfaces matérielles utilisées et toute autre information opportune, toutes les instructions et réponses que le serveur traite. Pratique si vous travaillez en Python ou PHP pour voir si les commandes ont été correctement exécutées. Le paramètre `-console` est spécifique à Windows qui ouvre une console où le serveur peut montrer des informations.

Attention ! Les chemins de périphériques pour Linux tels que `/dev/ttyS0` sont normalement acceptés par les versions récentes de Raspbian et Raspberry Pi modèle 3. La présence et la dénomination de ces chemins ont déjà été modifiées plusieurs fois ces dernières années. Si certains chemins n'existent pas sur votre RPi, vérifiez les paramètres pour I²C, SPI et UART dans `raspi-config`. Vous trouverez sur l'internet plus d'information à ce sujet.

RS-485

Swiss Pi établit une liaison directe entre son bus RS-485 et l'UART de l'hôte. Le contrôleur RS-485 est relié au bus SPI de l'hôte. Pour accorder au serveur la commande totale sur le RS-485, passez-lui les paramètres suivants :

- `serial /dev/ttyS0`
  le serveur commande l'UART de l'hôte.
- `spidev /dev/spidev0.0`
  le serveur commande le bus SPI de l'hôte.
- `rs485 spi`
  le serveur commande le contrôleur RS-485 de Swiss Pi.

Il y a trois méthodes pour faire passer des données par le bus RS-485 :

- avec les instructions `serr` et `serw`, lancer le serveur par `-serial` ;
- par un port réseau (par ex. le port 5004), lancer le serveur avec `-sp` et `-serial`.
- par le UART de l'hôte (par ex. /dev/ttyS0).

Avec la méthode 3, le serveur ne fournit d'accès ni à l'UART de l'hôte, ni au bus RS-485. Il faut configurer convenablement le contrôleur RS-485 avec la commande `rste` de manière à ce que l'UART et le contrôleur respectent le même réglage sériel.

Interactif

Pour vous familiariser avec le serveur, nous conseillons d'envoyer d'abord des instructions de manière interactive. Si vous démarrez le serveur avec `-stdio`, vous pouvez saisir des instructions directement dans le *shell*. Si vous le démarrez avec `-p`, vous pouvez établir une liaison avec un programme de terminal (PuTTY, netcat, Hyperterminal) et transmettre les instructions manuellement. Par exemple, se relier à netcat sur Raspberry Pi : `$ nc localhost 5003`.
Vous pouvez alors saisir des instructions. Dans les exemples qui suivent, les instructions sont en noir et les réponses du serveur en bleu. L'instruction suivante demande la version du serveur :

```
ver
ver «1.0.1»
```

On peut si nécessaire adjoindre à chaque instruction un numéro d'identification qui sera répété dans la réponse. Par exemple

« attendre 100 ms » :

```
id 10 wait 100
id 10 wait ok
```

On peut aussi adjoindre à chaque instruction un préfixe pour qu'il n'y ait pas de réponse :

```
norsp wait 100
```

GPIO

Une ligne d'E/S d'usage général peut servir d'entrée. Par exemple « régler la broche 10 d'E/S comme entrée, activer la résistance de polarisation haute et lire l'état de l'entrée » :

```
iod 10 1
iod ok
iopu 10 1
iopu ok
ior 10
ior 10 1 0 1 1
```

La valeur soulignée dans la dernière réponse veut dire que l'entrée est à l'état haut.
Il est aussi possible d'utiliser une ligne d'E/S d'usage général comme sortie. Par exemple « régler la broche 11 d'E/S comme sortie, la mettre au niveau haut » :

```
iod 11 0
iod ok
iow 11 1
iow ok
```

CA/N

L'instruction `adcr` convertit les huit canaux du convertisseur A/N et renvoie les résultats dans la réponse.

```
adcr
adcr 3460 1843 0756 0233 0060 0000 0000 0000
```

Le serveur enregistre les résultats de la dernière conversion. Vous pouvez à tout moment les réclamer avec l'instruction `adcrc`. Dans l'exemple suivant, on lit les résultats enregistrés pour les canaux 2 à 4 du CA/N :

```
adcrc 2 3
adcrc 2 3 0756 0233 0060
```

MLI

Le contrôleur MLI possède un diviseur préalable pour régler la fréquence MLI et 16 canaux, chacun a quatre paramètres qui déterminent la position de départ et la période de l'onde carrée. La signification précise de ces paramètres est expliquée dans la documentation. Les instructions `ppr` et `ppw` permettent d'écrire et de lire dans le registre du diviseur préalable. Avec les instructions `pcr` et `pcw`, vous pouvez écrire et lire un ou plusieurs canaux MLI.

```
ppr
ppr 030
ppw 121
ppw ok
pcr 5
pcr 05 01 0 0000 1 0000
pcw 9 1 0 1250 0 1280
pcw ok
pcr 8 2
pcr 08 02 0 0000 1 0000 0 1250 0 1280
```

Ces instructions servent à lire le diviseur préalable, positionner le registre sur 121 (50 Hz), lire le canal 5 MLI, écrire le canal 9 MLI, lire les canaux 8 à 9 MLI. La documentation décrit plusieurs variantes de `pcr` et `pcw`.

Servo

Le serveur dispose d'une commande de servomoteurs embarquée pour 16 canaux indépendants. Les canaux 0 à 3 sont à disposition sur un connecteur distinct du Swiss Pi.

On peut mettre en ou hors service la commande de servo par `svme` et `svmd`, configurer les canaux MLI en canaux de servo (`svcw`), activer des mouvements (`svmv`) et demander l'état (`svcr`).

Si vous travaillez avec des servomoteurs, vous devez régler la fréquence MLI sur 50 Hz :

```
ppw 121
ppw ok
```

Pour la configuration d'un canal MLI, `svcw` a besoin des informations suivantes :

- numéro de canal (0 à 15)
- largeur d'impulsion en 12 bits : la plus courte permise (0 à 4 095)
- largeur d'impulsion en 12 bits : la plus grande permise (0 à 4 095)
- type de commande de la position du moteur : par pas (0 à max.) ou largeur d'impulsion en 12 bits (0 à 4 095).

```
svcw 0 118 515 1000
svcw ok
```

Cette instruction définit que la commande de servo du canal MLI 0 peut opérer en largeur d'impulsion entre 118 et 515 et

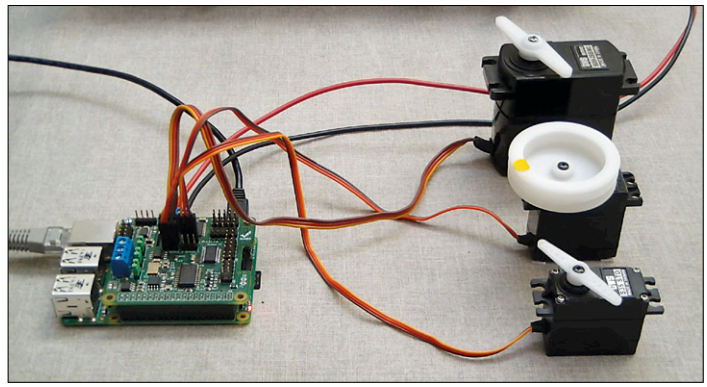


Figure 2. La commande de servomoteurs est un jeu d'enfant.

qu'une valeur de pas entre 0 et 1 000 sera donnée à svmv pour obtenir une largeur d'impulsion de 118 à 515.

```
svcw 1 122 520 0
svcw ok
```

Cette instruction définit que la commande de servo peut utiliser le canal MLI 1 avec des périodes MLI de 122 à 520 et qu'une largeur d'impulsion sera donnée à svmc.

Avant de démarrer un moteur, il faut activer la commande de servo :

```
svme
svme ok
```

Alors, pour changer la position du moteur :

```
svmv 0 500 0 0
svmv 00 ok
```

Cette instruction positionne le moteur au 500e pas, donc à mi-chemin entre 0 et 1 000. Le serveur convertit cette valeur de pas en largeur d'impulsion de 316, le milieu entre 118 et 515.

```
svmv 0 250 2000 0
svmv 0 500 2000 0
svmv 0 400 1000 0
```

Les instructions svmv sont exécutées l'une après l'autre. La durée totale est d'au moins 5 000 ms, vu qu'entre deux instructions, un retard peut survenir. Si vous souhaitez un mouvement fluide durant exactement 5 000 ms, utilisez le quatrième paramètre de svmv. Il indique que l'instruction svmv doit démarrer *x* millisecondes après l'instruction svmv précédente. Le serveur compense d'éventuels retards pour que l'exécution s'effectue toujours dans le temps prescrit.

```
svmv 0 250 2000 0
svmv 0 500 2000 2000
svmv 0 400 1000 2000
```

S'il n'y a eu aucune instruction svmv précédente pour le canal actif, le serveur prend svme comme point de départ. Veillez bien dans ce cas à ce que la largeur d'impulsion actuelle soit la bonne pour le domaine en question, sinon le serveur ne pourra pas faire exécuter le mouvement vers la nouvelle position. Cette façon de faire permet de synchroniser différents moteurs :

Pour travailler avec des servomoteurs, réglez la fréquence MLI à 50 Hz

```
svmv 1 400 0 0
svmv 01 ok
```

Cette instruction met le moteur dans la position qui correspond à une largeur d'impulsion de 400.

Le moteur se rend directement à la position indiquée ; on peut aussi fixer une période pour le mouvement :

```
svmv 0 500 2500 0
svcr 0
svcr 00 0118 0515 1000 0174 0141 1
svcr 0
svcr 00 0118 0515 1000 0257 0350 1
svmv 00 ok
svcr 0
svcr 00 0118 0515 1000 0316 0498 0
```

La première ligne fait exécuter le déplacement pendant une période de 2 500. Les instructions svcr révèlent la position actuelle du moteur. Le serveur n'envoie de réponse que quand le mouvement est terminé. L'exécution d'une svmv a lieu en arrière-plan, c'est donc une instruction asynchrone.

Vous pouvez envoyer plusieurs instructions successives au même canal :

```
svme
svme ok
svmv 0 250 2000 500
svmv 1 400 1550 500
```

Le serveur lance le mouvement des deux moteurs 500 ms après svme.

Horloge en temps réel

Le serveur dispose d'instructions pour lire l'horloge en temps réel (rtcr) et la régler (rtcw).

```
rtcr
rtcr 1 1
rtcw 2016 10 21 15 42 55
rtcw ok
rtcr
rtcr 0 1 2016 10 21 15 42 58
```

La première réponse signale que l'horloge en temps réel ne contient pas de temps valide et qu'il n'y a pas de pile branchée. La deuxième instruction met à jour et à l'heure. La troisième instruction renvoie la date et l'heure.

RS-485

Avec sercfg, il est possible de configurer l'UART de l'hôte et le contrôleur RS-485 d'un seul coup, à condition que le serveur

soit lancé par spi avec `-serial`, `-spidev` et `-rs485`.
Comme exemple, nous avons raccordé un compteur d'énergie SMD120C sur rail DIN au bus RS-485 et voici l'instruction qui configure la ligne sérielle sur 2 400 bauds, 8 bits de donnée, 1 bit d'arrêt, pas de parité et demande aussi au contrôleur RS-485 de configurer :

```
sercfg 2400 8 1 none 1
sercfg ok
```

Le SDM120C travaille avec le ModBus RTU. Pour demander la valeur de la tension :

```
serw 01h 04h 00h 00h 00h 02h 71h 0CBh
serw ok
```

Et pour lire la réponse :

```
serr
serr 001 004 004 067 107 153 154 117 231
```

La documentation du SDM120C indique que la tension est fournie dans un nombre à virgule flottante de 32 bits, ce sont ceux qui sont soulignés.

```
host = 'localhost'
port = 5003
client = textlinenetclient.
   TextLineNetClient(host,port)
```

Cette liaison est locale, le code Python doit être exécuté sur le RPi. Si vous voulez travailler avec le Swiss Pi depuis un autre ordinateur, vous devez donner l'adresse IP de RPi sous la forme `host = '192.168.1.115'`. Vous pouvez demander l'adresse IP de votre Raspberry Pi avec la commande `ifconfig`.
La variable `client` désigne la liaison. La classe dispose de fonctions pour envoyer des instructions et recevoir des réponses en lignes de texte.
L'écriture des instructions comporte deux étapes. La classe travaille avec un tampon d'écriture. La fonction `sendCmd` ajoute une instruction au tampon d'écriture. Pour transférer les instructions du tampon au serveur, on a la fonction `commit`. Autre solution : ajouter le paramètre `True` à `sendCmd`. On peut ainsi envoyer au serveur une kyrielle d'instructions d'un seul coup, c'est plus rapide que de les envoyer séparément.
La fonction `rcvRsp` lit les données du serveur et envoie une seule réponse par appel. Cette fonction bloque le serveur tant qu'elle n'a pas obtenu de réponse.
La fonction `tokenizeLine` découpe une ligne de texte en une série de symboles (*token*). Les programmes utilisent cette fonction pour simplifier l'analyse et préparer la réponse. Si vous avez lancé le serveur avec le paramètre `-v`, vous pouvez surveiller les réponses du serveur.

`sercfg` configure d'un coup l'UART de l'hôte et le contrôleur RS-485

Python
Les exemples de programme proposés sont écrits pour Python 3. Si vous utilisez Raspbian sur votre Raspberry Pi, Python 3 est déjà installé sur votre système. Parcourons-en les principaux aspects.
Pour exécuter un programme, vous lancez Python avec le nom de fichier du programme comme paramètre dans l'invite de commande :

Linux : `python3 swisspi_gpio_read_pin.py`

Windows : `python swisspi_gpio_read_pin.py`

Les programmes font usage du serveur Swiss pour communiquer avec Swiss Pi. Ainsi, vous pouvez faire tourner les programmes localement sur le Raspberry Pi ou sur un autre ordinateur.

Les programmes utilisent `class TextLineNetClient` dans *textlinenetclient.py*. Cela permet d'établir la liaison avec le serveur Swiss pour envoyer des tâches et traiter les réponses. Le nom de la classe indique qu'il s'agit d'un client de réseau qui échange des lignes de texte avec le serveur. La classe produit une exception du type `class TextLineNetClientError` quand survient une erreur telle qu'une rupture de liaison avec le serveur.
Le code suivant en Python établit une liaison avec le serveur Swiss :

Les nombres qui arrivent dans les réponses sont formatés selon un modèle réglable. Il y a différents formats afin d'obtenir une bonne lisibilité en mode interactif. Quand on travaille en Python, mieux vaut recevoir tous les nombres en valeurs décimales. Dans ce cas, les programmes envoient la commande :

```
client.sendCmd('norsp vfmts «*» dec 0 0 0 0 0 0')
```

Remarquez qu'avec `norsp`, le serveur n'envoie pas de réponse. Il n'est pas nécessaire non plus d'envoyer immédiatement après une instruction au serveur. Ce qui est important c'est que le serveur reçoive cette instruction en premier.

GPIO
Le code Python qui suit configure une ligne d'E/S d'usage général en entrée, active la résistance de polarisation haute et lit l'état de l'entrée. La variable `pin` sélectionne les broches de GPIO (0 à 15).

```
pin = 10
client.sendCmd('norsp iod %d 1' % pin)
```

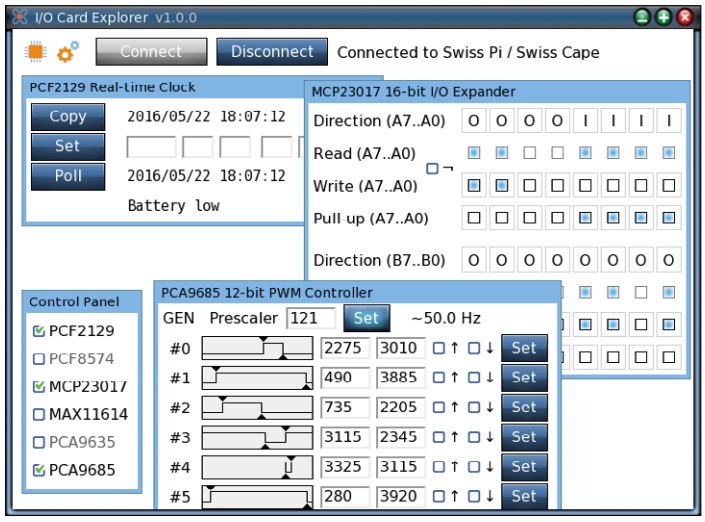


Figure 3. Avec *I/O Card Explorer*, la liaison avec le Swiss Pi est facile.

```
client.sendCmd('norsp iopu %d 1' % pin)
client.sendCmd('ior %d' % pin,True)
tokens = client.tokenizeLine(client.rcvRsp())
print("GPIO pin %d: input state %d" %
   (pin,int(tokens[2])))
```

Le code envoie trois instructions au serveur. La première met la broche GPIO 10 en entrée, la deuxième active la polarisation et la troisième `sendCmd` lit l'état, mais aussi transmet le tampon d'écriture au serveur à cause du paramètre `True`. Le serveur répond à la troisième instruction sous forme scindée, avec des symboles. Le troisième symbole (`tokens[2]`) contient l'état d'entrée de la broche de GPIO.

Configurer l'état de sortie d'une broche de GPIO se fait en sélectionnant aussi par `pin` la broche de GPIO (0 à 15), puis la variable `output` établit l'état (0 ou 1). Mettons la sortie de la broche 2 au niveau haut :

```
pin = 2
output = 1
client.sendCmd('norsp iow %d %d' % (pin,output),True)
```

ADC

L'instruction `adcr` convertit les huit canaux A/N et transmet les résultats dans la réponse. Les canaux sont numérotés de 0 à 7. Pour voir la valeur du canal 4 :

```
ch = 4
client.sendCmd('adcr',True)
tokens = client.tokenizeLine(client.rcvRsp())
print(«A/D channel %d: %d» % (ch,int(tokens[1 + ch])))
```

MLI

Le contrôleur MLI a un registre de division préalable pour régler la fréquence de la MLI. Pour lire le registre :

```
client.sendCmd('ppr',True)
tokens = client.tokenizeLine(client.rcvRsp())
print(«PWM prescale: %d» % int(tokens[1]))
```

Écrire dans ce registre se fait ainsi :

```
pre = 121  # 50 Hz
client.sendCmd('norsp ppw %d' % pre,True)
```

Chaque canal dispose de quatre paramètres pour fixer la position de départ et la période de l'onde carrée. Pour simplement régler la période :

```
ch = 4
on_period = 1250
client.sendCmd('norsp pcw %d 1 0 0 0 %d' %
   (ch,on_period),True)
```

Servo

Le code suivant configure deux canaux et tourne les moteurs dans une certaine position :

```
client.sendCmd('norsp svcw 0 118 515 1000')
client.sendCmd('norsp svcw 1 122 520 0')
client.sendCmd('norsp svme')
client.sendCmd('norsp svmv 0 250 0 0')
client.sendCmd('norsp svmv 1 400 0 0')
client.commit()
```

Horloge en temps réel

Le code qui suit règle l'horloge en accord avec le temps local de l'ordinateur ; il montre aussi l'usage de la fonction `commit`, qui équivaut à l'ensemble des paramètres `True` et `sendCmd`.

```
t = datetime.datetime.now()
client.sendCmd('norsp rtcw %d %d %d %d %d %d'
   % (t.year, t.month, t.day, t.hour, t.minute,
   t.second))
client.commit()
```

RS-485

L'exemple de programme *swisspi_rs485_sdm120c.py* lit la tension d'un compteur d'énergie SDM120C et l'imprime.

PHP

Les exemples de programme utilisent le serveur Swiss pour communiquer avec le Swiss Pi. Vous pouvez ainsi exécuter les programmes localement sur le Raspberry Pi mais aussi sur un autre ordinateur.

Les programmes se servent de `class SwissClient` dans *SwissClient.php*. Cette classe propose une série de fonctions qui exécutent les instructions du serveur. Les fonctions comprennent l'échange d'instructions et de réponses avec le serveur. L'avantage est que les fonctions sont faciles à utiliser. Toutefois il y a un inconvénient : l'impossibilité d'envoyer plusieurs instructions en même temps et d'opérer avec des réponses asynchrones. C'est logique, puisque le PHP s'utilise généralement

pour produire des pages internet et l'accès synchrone au Swiss Pi lui suffit.

Comment réaliser une liaison avec le serveur ? D'abord, il faut créer un objet `SwissClient`, puis appeler la fonction `connect`. Elle renvoie un `true` si la connexion a réussi, `false` si elle a échoué.

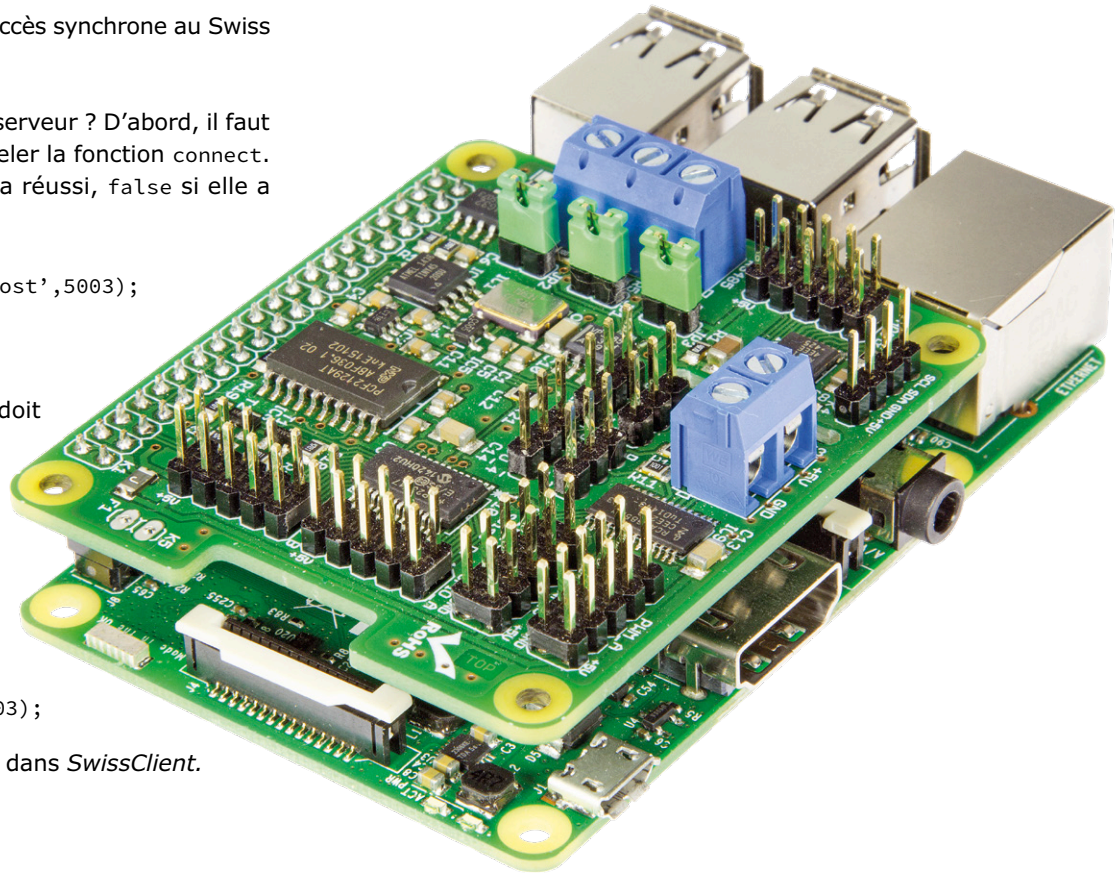
```
$client = new SwissClient('localhost',5003);
$ok = $client->connect();
if (!$ok) return false;
```

Cette liaison est locale, le code PHP doit être exécuté sur le Swiss Pi. Si vous voulez travailler depuis un autre ordinateur avec le Swiss Pi, vous devez donner l'adresse IP du Raspberry Pi. Voici comment ça marche :

```
$client = new

   SwissClient('192.168.1.115',5003);
```

Ce que font les fonctions est expliqué dans *SwissClient.php*. Suivent quelques exemples.



Avec *I/O Card Explorer*, observez l'effet des instructions sur le Swiss Pi.

Lire l'état de la broche 4 de GPIO et imprimer le résultat :

```
$res = $client->readIOPin(4);
var_dump($res);
```

Mettre la sortie broche 4 de GPIO au niveau haut, communiquer la réponse du serveur :

```
$client->writeIOPin(4,true,true);
```

Écrire vers deux canaux MLI :

```
$pwm_data = array
(
    array("always_on" => false, "on_pos" => 1200,
   "always_off" => false, "off_pos" => 2424),
    array("always_on" => true,  "on_pos" =>  444,
   "always_off" => true,  "off_pos" => 100)
);

$client->writePWM2Range(14,$pwm_data,true);
```

Et maintenant ?

Le logiciel contient de nombreux fichiers en Python et PHP que vous pouvez mettre à profit. Vous pouvez à tout moment établir une liaison avec le programme *I/O Card Explorer* [5] par le serveur Swiss (**figure 3**). Vous avez ainsi l'occasion, pendant le transfert interactif d'instructions ou l'exécution de codes Python ou PHP, d'observer l'activité de Swiss Pi et éventuellement d'intervenir.

La prochaine fois, nous vous présenterons quelques extensions matérielles pratiques, parmi lesquelles une carte à relais à 8 canaux, une carte d'entrée numérique à 8 canaux, une commande de moteur à courant continu, une interface à boucle de courant et une petite carte qui convertit la MLI en tension entre 0 et 10 V. ⏮

(160237 – version française : Robert Grignard)

Liens

[1] www.elektormagazine.fr/150584

[2] www.axiris.eu/en/index.php/i-o-cards/swiss-pi

[3] www.axiris.eu/en/index.php/free-software/software-repository

[4] www.elektormagazine.fr/150585

[5] www.axiris.eu/en/index.php/free-software/i-o-card-explorer

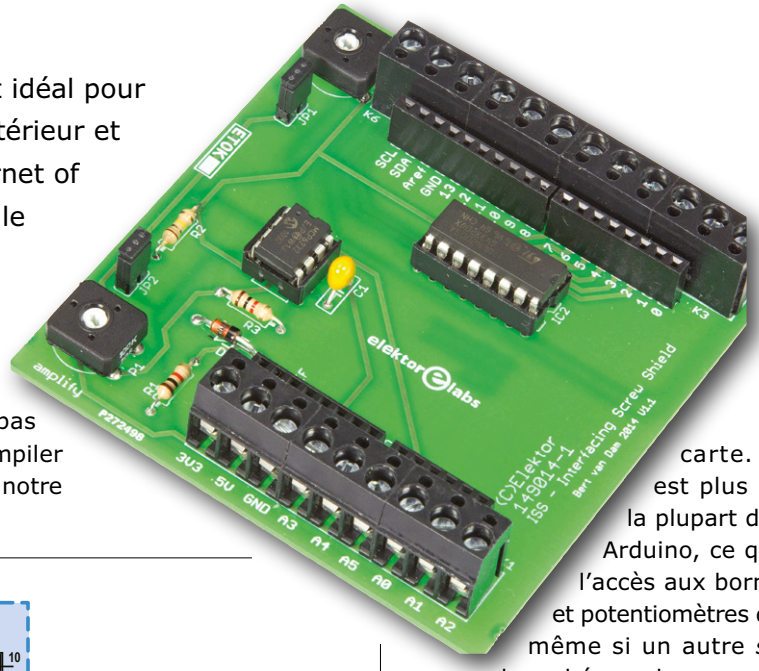
[6] www.elektormagazine.fr/160237

shield IdO pour Arduino

construisez vos objets connectés

Clemens Valens (labo d'Elektor)

Ce *shield*, compatible Arduino Uno R3, est idéal pour des applications de contrôle simples à l'intérieur et à l'extérieur de la maison. Le label « Internet of Things (IoT) » est applicable dès lors que le système est connecté à l'internet.



Le *shield* [1] peut être câblé avec un tournevis, aucune soudure n'est requise. Le système est flexible et extensible, en effet la carte n'utilise pas tous les signaux de l'Arduino, ce qui permet d'empiler d'autres *shields* tant qu'ils n'interfèrent pas avec notre carte. La carte est plus large que la plupart des *shields* Arduino, ce qui permet l'accès aux borniers à vis et potentiomètres de réglage même si un autre *shield* est branché par-dessus.

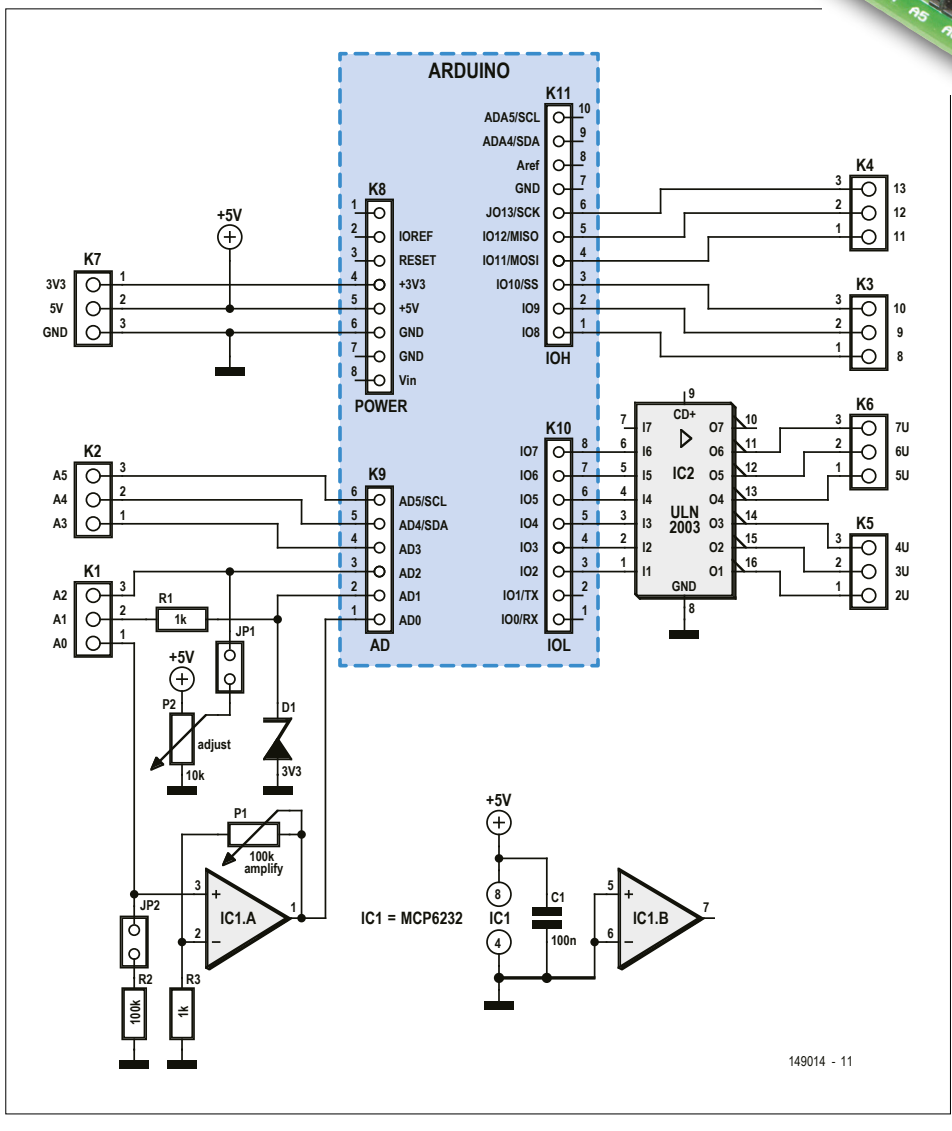


Figure 1. Juste les faits. Un projet simple, sans fantaisie et avec des composants faciles à trouver.

Cette carte offre plusieurs avantages :

- Une source de tension variable pour polariser ou alimenter des capteurs
- Une entrée analogique avec un gain ajustable pour les petits signaux
- Une entrée de détection de haute tension, limitée en courant et tension
- Six sorties à transistor de puissance pour commander par ex. des lampes ou des relais.

Examinons les différentes fonctions du *shield* en nous aidant de la **figure 1**.

Source de tension variable

Le potentiomètre P2 fournit une tension variable entre 0 et 5 V. Si le cavalier JP1 est court-circuité, cette tension est disponible sur l'entrée analogique A3 de l'Arduino. Dans ce cas, la tension est aussi disponible sur la broche 3 de K1 et peut servir pour commander, polariser ou encore alimenter un système externe (à faible courant) comme un capteur. Quand JP1 est ouvert, P2 n'a aucune fonction ; cette broche sert alors d'entrée analogique ou d'entrée/sortie numérique.

Petits signaux

Le circuit IC1A, alimenté par la ligne 5 V, est un amplificateur opérationnel (AOP) de type *rail-to-rail*, ce qui signifie que

ses entrées et sortie vont de 0 à 5 V (à quelques millivolts près). Cette sortie n'est connectée qu'à l'entrée analogique A0 de l'Arduino, c'est pourquoi son signal n'est pas disponible sur un bornier à vis. L'AOP est câblé en amplificateur non-inverseur, son gain est contrôlé par P1. Le gain maximum est d'environ 100, le minimum de 1. Le signal d'entrée sur la broche 1 de K1 peut donc être amplifié ou simplement tamponné. Bien que l'impédance de cette entrée soit très élevée, elle peut être fixée à 100 kΩ en court-circuitant JP2. Faire cela permet d'éviter à l'AOP d'amplifier bruit, bourdonnement et autres interférences quand son entrée est laissée en l'air. Si, au contraire, la source connectée à cette entrée a une très haute impédance de sortie, il est préférable d'ouvrir JP2 pour éviter d'écraser ce signal sensible.

Détecteur de haute tension

La broche 2 de K1 est une entrée analogique limitée en courant et tension. La diode Zener D1 limite la tension continue à max. 3,3 V, un niveau sans danger pour les cartes Arduino. Attention : les tensions négatives (CA) sont interdites sur cette entrée, et sur toutes les autres d'ailleurs. R1 limite le courant sur l'entrée et D1. Sa valeur est un petit peu élevée, ce qui signifie que quand vous connectez l'entrée à 5 V env., la tension sera limitée à 3 V env., mais avec une tension d'entrée de 12 V ou 15 V, vous mesurerez 3,3 V. L'objectif de cette entrée est de détecter des tensions (beaucoup) plus élevées que ce que l'Arduino peut accepter, par ex. la tension de sortie d'un adaptateur CC ou l'état d'un commutateur connecté à une haute tension. Bien qu'en théorie cette entrée résiste facilement jusqu'à 100 V de tension, **ne la connectez jamais à une ligne CA !**

Sorties de puissance

Le composant IC2, un ULN2003A, a une très bonne réputation, c'est pourquoi il est utilisé ici. Il contient sept transistors Darlington de puissance qui peuvent être commandés numériquement. Bien que le circuit intégré possède sept canaux, le *shield* n'en utilise que six. Chaque transistor peut commuter 500 mA et supporter jusqu'à 50 V. Ces transistors sont excellents pour déconnecter un dispositif relié à une alimentation (50 V max.), comme un relais, une lampe ou un moteur, parce qu'ils supportent des courants relativement intenses. Une sortie d'Arduino en est incapable sans l'aide d'un circuit de ce genre. Si vous avez besoin de commuter plus de courant que ce qu'un transistor peut supporter, utilisez deux ou plus de ces sorties en parallèle. Si vous voulez commander des charges inductives (comme un solénoïde, un relais ou un moteur), ajoutez une diode de roue libre en parallèle de la charge (cathode sur la source d'alimentation, anode sur la sortie d'IC2) car les diodes de roue libre du circuit intégré ne sont pas utilisées sur cette carte.

Quelques remarques

- Le *shield* est alimenté par l'Arduino ; n'alimentez pas le *shield* via K7 quand il est branché sur un Arduino. Ce connecteur n'est prévu que pour alimenter les appareils connectés au *shield*, pas pour alimenter l'Arduino.
- La sortie 3,3 V est celle de l'Arduino, donc pas très puissante ; prenez soin d'elle.
- IC1 et IC2 sont montés sur support pour être faciles à remplacer en cas d'incident.
- La plupart des signaux de l'Arduino sont ramenés sur de robustes borniers à vis pour faciliter la connexion des fils et câbles sur la carte. La plupart, mais pas tous : les broches du port série (0 et 1), V_{IN}, Reset, IO_{REF} et A_{REF} ne sont pas accessibles sur les borniers à vis, mais uniquement sur les embases femelles du *shield*.

(160169 – version française : Alexandre Roy)

Liste de composants

Résistances

Toutes 5 %, 0,25 W
R1, R3 = 1 kΩ
R2 = 100 kΩ
P1 = 100 kΩ, potentiomètre, horizontal
P2 = 10 kΩ, potentiomètre, horizontal

Condensateurs

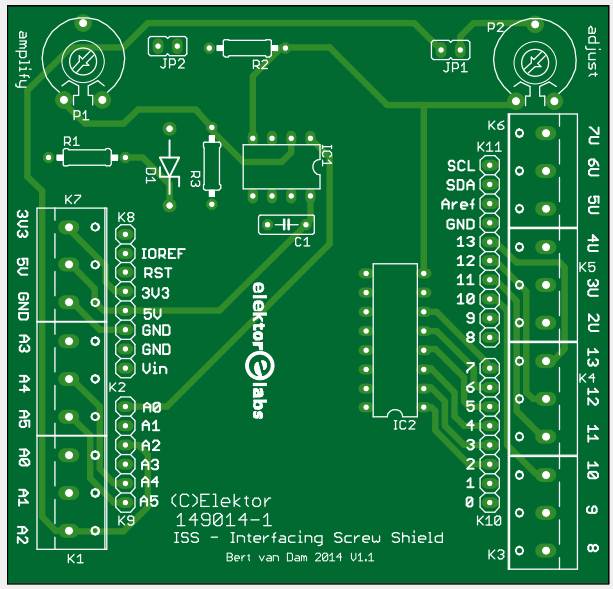
C1 = 100 nF, au pas de 5,08 mm

Semi-conducteurs

D1 = BZX85C, 3,3 V, diode Zener de 1,3 W
IC1 = MCP6232
IC2 = ULN2003

Divers

JP1, JP2 = embase mâle, à 2 contacts, au pas de 2,54 mm
2 cavaliers pour JP1 et JP2
K1,K2,K3,K4,K5,K6,K7 = bornier à vis pour circuit imprimé, à 3 pôles, au pas de 5 mm
K8, K10 = embase femelle à 8 contacts, 1 rangée, au pas de 2,54 mm
K11 = embase femelle à 10 contacts, 1 rangée, au pas de 2,54 mm
Support de CI, DIP-8 pour IC1
Support de CI, DIP-16 pour IC2
Circuit imprimé, réf. 149014-1 (www.elektor.fr)



Liens

[1] www.elektor.fr/interfacing-screw-shield-149014-91

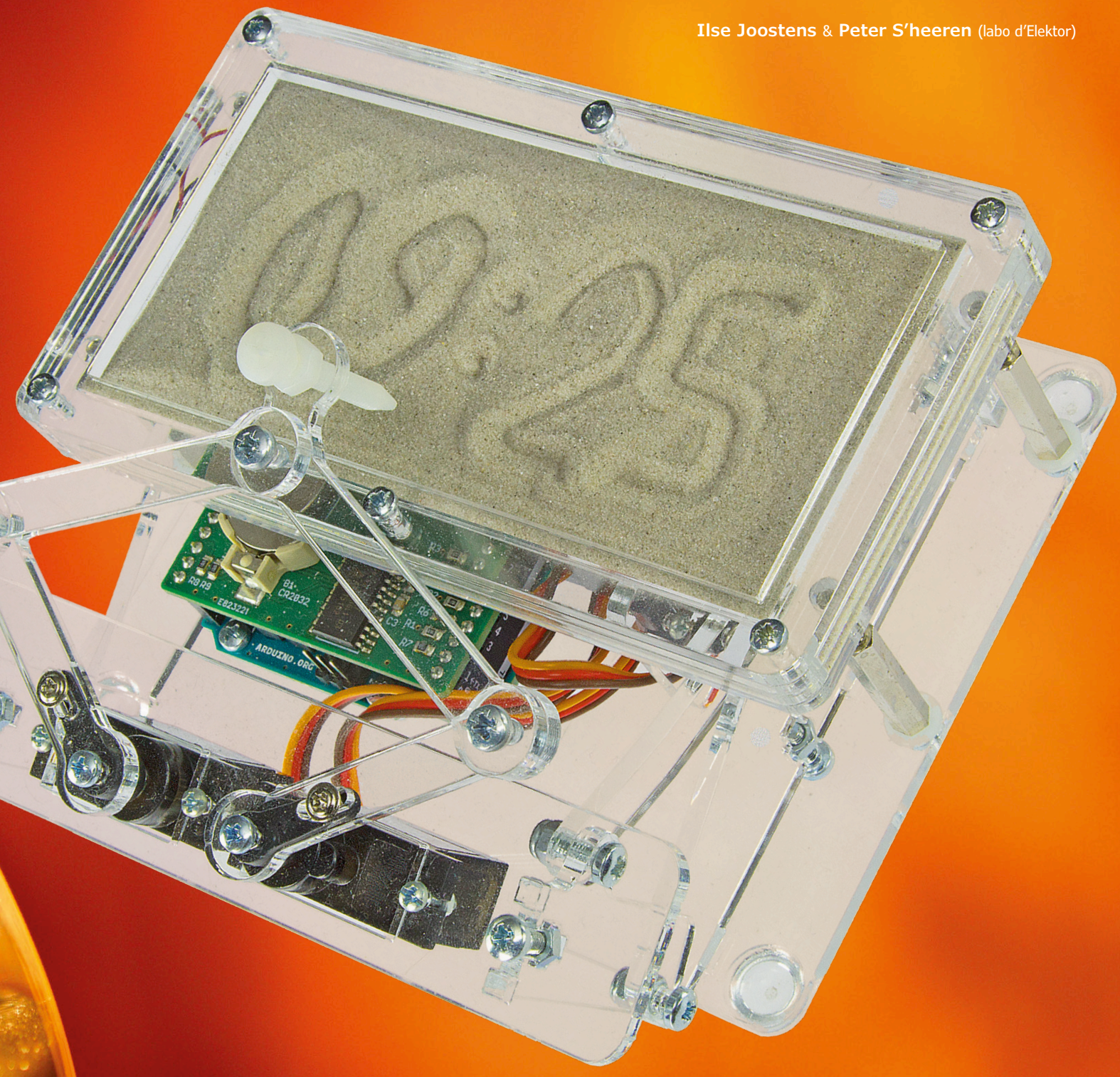
[2] www.elektormagazine.fr/160169

Vous trouverez dans le livre en anglais « IoT GET-U-GOING » (e-choppe Elektor, réf. 17460) d'excellents exemples d'utilisation du *shield* décrit dans cet article, pour toutes sortes d'applications, IdO ou non :

www.elektor.fr/iot-get-u-going

horloge de sable
un modèle fascinant

Ilse Joostens & **Peter S'heeren** (labo d'Elektor)



Ce gadget hors du commun, bâti autour d'une carte Arduino Uno, écrit l'heure sur un lit de sable à l'aide de quelques servomoteurs et d'un mécanisme de pantographe. Après un temps déterminé, deux vibrateurs lissent la couche de sable et l'heure est à nouveau dessinée. En dehors de sa fonction d'horloge, le mécanisme peut aussi exécuter des commandes simples.

L'idée de ce projet vient de nos collègues du magazine allemand *Make*, qui se sont eux-mêmes inspirés de la *Plotclock* (horloge qui écrit l'heure) du FabLab de Nuremberg [1]. Lors d'une réunion de travail avec nos collègues de *Make*, nous avons étudié quelques projets que le labo pourrait retravailler pour en faire des kits ; le premier choisi est celui de cette horloge de sable.

Nous avons tout d'abord modifié les sous-ensembles mécaniques afin que leur montage soit plus facile et pour qu'ils soient adaptés à un kit. Nous avons ensuite conçu une carte d'extension (*shield*) Arduino pour l'électronique additionnelle. Le croquis (*sketch*) original a quant à lui été réécrit ; il est désormais plus lisible, plus convivial, et offre plus de possibilités. Le résultat de nos cogitations vous est présenté ci-dessous.

Le kit, qui comprend tous les composants, une carte Arduino Uno et son *shield* avec les CMS déjà soudés, est disponible dans l'e-choppe d'Elektor (**fig. 1**). Pour ceux qui ont accès à une machine de découpe au laser et qui sont capables de souder des CMS, les fichiers de CAO et Gerber sont également disponibles, voir [2].

Les sous-ensembles mécaniques sont en PMMA extrudé de 3 mm d'épaisseur (polyméthacrylate de méthyle, plus connu sous son appellation commerciale de Plexiglas®). Ce polymère thermoplastique est facile à trouver, et il se prête très bien à la découpe au laser, ce qui permet d'obtenir des arêtes impeccables.
Trois servomoteurs utilisés en modélisme sont mis en œuvre dans l'horloge, c'est aussi le cas pour la *Plotclock*. Le couple nécessaire n'est pas très élevé, on pourra donc se satisfaire de petits servos. Bien que le montage soit direct sur l'axe du moteur, et que le jeu des bras du pantographe soit faible, nous avons sélectionné des modèles de qualité, avec engrenages métalliques.
Le lissage du lit de sable se fait à l'aide de deux moteurs vibrants de 6 mm de diamètre, maintenus à 45°, sous le bac à sable, à l'aide de serre-câbles en plastique.

Le pantographe

Le cœur de l'horloge est constitué des servomoteurs gauche et droit, et des bras en Plexiglas qui positionnent le stylet d'écriture. Ce type d'assemblage est appelé pantographe, par analogie avec l'instrument de dessin, tombé en désuétude aujourd'hui.
Une machine comparable, utilisant un pantographe, est la machine à signer (*autopen*). C'est un automate qui sert à reproduire une signature, elle a été et est toujours utilisée par des politiciens ou des célébrités ; on l'utilise entre autres pour des séances de dédicace, c'est moins impersonnel qu'une signature imprimée. Des présidents américains sont connus pour avoir fait usage d'une telle machine, dont Barack Obama, qui a ainsi signé des lois lors d'une visite en France ou encore pendant des vacances à Hawaï.
Les formules mathématiques qui décrivent le fonctionnement de l'horloge de sable sont exposées dans un encadré sur la cinématique inverse. Elles ont été utilisées pour l'écriture du logiciel.

Lit de sable vibrant

Les mathématiques ont beau être quelque peu effrayantes, le sous-ensemble le plus difficile à mettre au point était le lit de sable ; de nombreux prototypes ont été testés avant d'obtenir un résultat satisfaisant.
Le but des moteurs vibrants n'est pas de faire trembler toute l'horloge, un maximum d'énergie doit être transféré au sable. Le bac à sable repose donc libre-

Figure 1. Le kit contient tous les composants nécessaires, même un adaptateur secteur. Il suffit dès lors de programmer l'Arduino avec le logiciel disponible gratuitement.

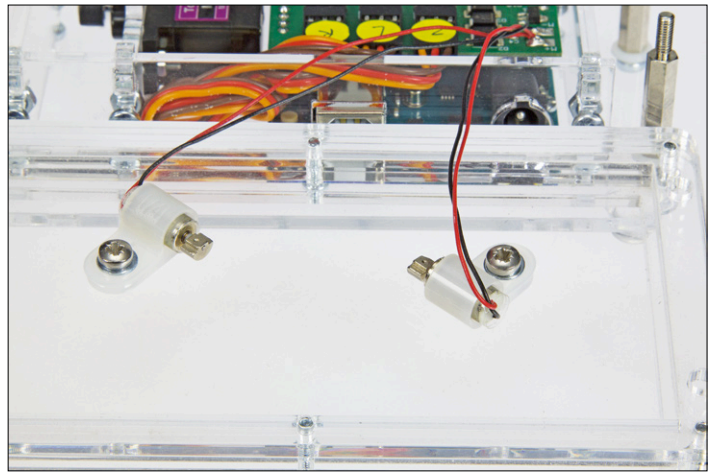


Figure 2. La position des moteurs vibrants sous le lit de sable est importante, pour un bon lissage du sable.

ment sur quatre supports, et jouit d'une certaine liberté de mouvement dans le plan horizontal.

Le plus grand défi était de rendre le lit de sable suffisamment lisse en plus ou moins 5 s pour pouvoir y écrire l'heure, et sans que le sable ne s'accumule par endroits. Nous avons expérimenté avec divers montages et moteurs ; un seul moteur n'est visiblement pas suffisant, et deux moteurs cylindriques de 6 mm de diamètre ont finalement donné des résultats satisfaisants. La position des moteurs est aussi cruciale, et ils sont placés horizontalement sous le bac à sable, à un angle de 45° par rapport aux bords (**fig. 2**).

Le choix du sable est également important : la dimension des grains (qui peut varier du gravier à limon) et leur forme (plus ou moins ronde) déterminent la capacité du sable à s'écouler. Nous avons finalement opté pour du sable blanc fin (grains de 0,1 à 0,3 mm) ; comme le bac n'est pas très épais, nous conseillons d'utiliser du sable le plus fin possible. Nous avons toutefois prévu un rebord, afin d'éviter que du sable soit éjecté du bac.

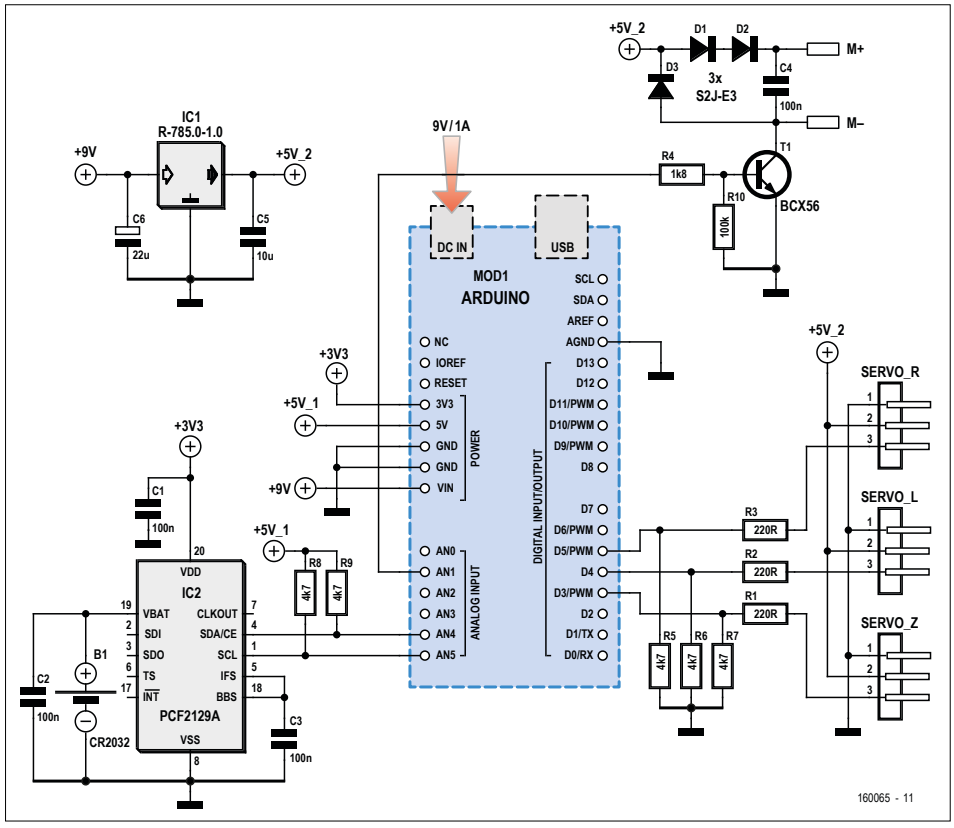


Figure 3. Le schéma du *shield* spécifiquement conçu pour l'horloge : un convertisseur CC-CC pour l'alimentation des moteurs, un circuit d'horloge en temps réel, une interface pour les moteurs vibrants, et quelques connecteurs.

L'électronique

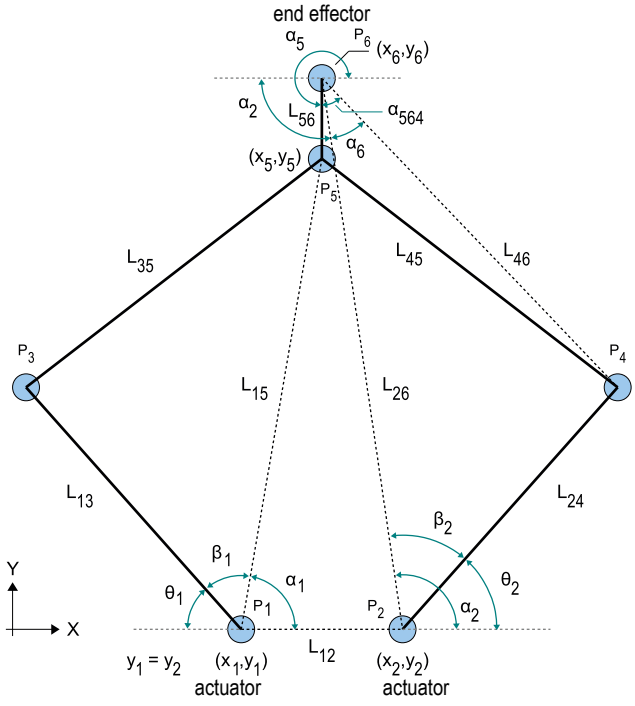
Le circuit est construit autour d'une carte Arduino Uno, dont la puissance de calcul est suffisante pour cette application (malgré les nombreux calculs nécessaires, voir l'encadré). Quelques composants supplémentaires sont nécessaires, pour l'alimentation des servomoteurs, la commande des moteurs vibrants, et le maintien de l'heure en cas de coupure du secteur ; le schéma est repris en **figure 3**.
Afin de rendre le montage le plus simple possible, nous avons monté ces composants supplémentaires sur un *shield* Arduino (**fig. 4**) ; et pour éviter tout souci avec les CMS, le circuit imprimé est fourni avec ceux-ci déjà soudés. Seuls les connecteurs et le convertisseur CC-CC doivent encore être implantés.

L'horloge est alimentée par un adaptateur secteur standard qui délivre de 9 à 12 V. Les servos et les moteurs vibrants consomment une certaine puissance, il vaut donc mieux ne pas utiliser l'Arduino pour les alimenter : le régulateur présent sur la carte Uno est un modèle linéaire qui ne peut débiter au maximum que 800 mA. C'est non seulement trop juste, mais il faut aussi tenir compte d'un échauffement possible de ce régulateur, dont le rendement n'est pas très bon. Si la carte Arduino est connectée à un ordinateur par un câble USB, et qu'on a oublié l'adaptateur secteur, on encourt aussi le risque de voir l'alimentation USB de l'ordinateur s'effondrer.

> On peut aussi commander l'horloge de sable manuellement

L'alimentation des moteurs est donc indépendante de celle de l'Arduino. Elle est assurée par un convertisseur abaisseur CC-CC en boîtier SIP3 (IC1), dont les connexions sont identiques à celles d'un régulateur linéaire en boîtier TO-220 ; son entrée est reliée à l'entrée d'alimentation de l'Arduino. La tension de sortie est de 5 V, et le courant maximal est de 1 A ; le rendement est supérieur à 90% avec une tension d'entrée de 9 à 12 V.
Les moteurs vibrants sont commandés par une broche d'E/S de l'Arduino, via un transistor NPN (T1). Leur tension d'alimentation (suivant la feuille de caractéristiques) est de 3 V ; la tension de 5 V est donc abaissée grâce à la chute de tension aux bornes de D1 et D2, et à la tension collecteur-émetteur de T1 ($V_{CE\ sat}$).
La commande des servos se fait directement par trois broches d'E/S de l'Arduino ; trois connecteurs sont prévus sur le *shield*. Les résistances R1, R2 et R3 protègent l'Arduino en cas d'inversion accidentelle de la connexion d'un servo. Les résistances R5, R6 et R7 maintiennent un niveau stable des lignes de données vers les servos lorsque le microcontrôleur de l'Arduino est remis à zéro (*reset*). On ne veut pas avoir à remettre l'horloge à l'heure chaque fois qu'on la débranche et rebranche ; un circuit d'horloge en temps réel avec oscillateur à quartz intégré, un PCF2129A de chez NXP (IC2), permet de s'affranchir des problèmes de coupure de courant ou autres. Ce circuit

Cinématique inverse



En robotique, une chaîne cinématique est un modèle mathématique d'un système mécanique, où des solides indéformables sont connectés entre eux par des articulations (liaisons mécaniques). Les mouvements d'une telle chaîne cinématique peuvent être modélisés sur la base d'équations mathématiques.

En cinématique directe, la configuration de la chaîne est calculée en fonction de la position des articulations. En cinématique inverse, c'est le contraire : la position des articulations est calculée en fonction de la configuration souhaitée de la chaîne ; le but est en général de déterminer la position et les déplacements d'un effecteur ou d'un manipulateur.

La cinématique inverse est aussi utilisée en infographie et en animation (p. ex. pour les jeux vidéo) ; elle permet la modélisation des mouvements du corps humain ou des animaux.

Dans le cas de l'horloge de sable, la chaîne cinématique comprend quatre solides indéformables et cinq articulations (liaisons pivot), tels que représentés sur le schéma du mécanisme. Deux des articulations sont des actionneurs (les servomoteurs), et l'effecteur est ici le stylet. En fonction de la position souhaitée du stylet (x_6, y_6), nous devons donc calculer la valeur des angles θ_1 et θ_2 des actionneurs.

Vu que les articulations n'ont qu'un seul degré de liberté et qu'il n'y a que deux actionneurs, les équations sont en nombre relativement restreint. L'ATmega328P qui équipe l'Arduino Uno n'aura donc aucun mal à effectuer les calculs nécessaires.

Pour simplifier les calculs, nous avons donné aux axes des deux servomoteurs la même ordonnée : $y_1 = y_2$. Les positions, longueurs et angles sont représentés sur le schéma. Certaines valeurs sont constantes, d'autres sont variables.

Constantes : (x_1, y_1), (x_2, y_2), L_{13}, L_{24}, L_{35}, L_{45}, L_{46}, L_{56}, α_{564}

Variables : (x_5, y_5), (x_6, y_6), L_{15}, L_{26}, α_1, β_1, θ_1, α_2, β_2, θ_2, α_5, α_6

Nous déterminons d'abord la valeur de θ_2, égale à la différence entre les valeurs d'α_2 et de β_2. α_2 est l'angle entre l'axe X et la ligne reliant le servomoteur de droite (P_2) et le stylet (P_6) :

$$\alpha_2 = \arctan 2\left(\frac{y_6 - y_2}{x_6 - x_2}\right)$$

La valeur de β_2 est calculée en fonction de la longueur des côtés du triangle $P_2P_4P_6$. L_{46} est connue, et L_{26} est calculée comme suit :

$$L_{26} = \sqrt{(x_6 - x_2)^2 + (y_6 - y_2)^2}$$

On peut alors calculer la valeur de β_2 comme suit :

$$\beta_2 = \arccos\left(\frac{L_{26}^2 + L_{24}^2 - L_{46}^2}{2 \cdot L_{26} \cdot L_{24}}\right)$$

La valeur de θ_2 est alors :

$$\Theta_2 = \alpha_2 - \beta_2$$

Nous calculons la valeur de θ_1 en fonction de la longueur des côtés du triangle $P_1P_3P_5$. La position de P_5 (x_5, y_5) est une variable que nous devons d'abord déterminer.

Considérons que le triangle $P_4P_5P_6$ est un objet indéformable, dont la valeur des angles et la longueur des côtés sont constantes. Pour calculer x_5 et y_5, on effectue une rotation du triangle d'un angle α_5 autour de P_6, comme montré sur le schéma.

Nous déterminons d'abord la valeur de l'angle α_6 :

$$\alpha_6 = \arccos\left(\frac{L_{26}^2 + L_{46}^2 - L_{24}^2}{2 \cdot L_{26} \cdot L_{46}}\right)$$

La valeur d'α_2 étant connue, nous pouvons déterminer celle d'α_5 :

$$\alpha_5 = \pi + \alpha_2 + \alpha_6 - \alpha_{564}$$

La position de P_5 est alors la suivante :

$$x_5 = x_6 + L_{56} \cdot \cos(\alpha_5)$$
$$y_5 = y_6 + L_{56} \cdot \sin(\alpha_5)$$

La valeur d'θ_1 est calculée en fonction de celle d'α_1 et de β_1. α_1 est l'angle entre l'axe X et la ligne reliant le servomoteur de gauche (P_2) et P_5 :

$$\alpha_1 = \arctan 2\left(\frac{y_5 - y_1}{x_5 - x_1}\right)$$

La valeur de β_1 est calculée en fonction de la longueur des côtés du triangle $P_1P_3P_5$. L_{35} est connue, et L_{15} est calculée comme suit :

$$L_{15} = \sqrt{(x_5 - x_1)^2 + (y_5 - y_1)^2}$$

La valeur de β_1 est alors :

$$\beta_1 = \arccos\left(\frac{L_{15}^2 + L_{13}^2 - L_{35}^2}{2 \cdot L_{15} \cdot L_{13}}\right)$$

Et enfin la valeur de θ_1 :

$$\Theta_1 = \pi - \beta_1 - \alpha_1$$

Ces calculs sont implémentés dans la fonction ***pen_calc*** du croquis.

intégré est bon marché, et sa compensation en température garantit une précision de 3 ppm. Une pile bouton CR2032 prend le relais en cas de coupure de l'alimentation ; sa durée de vie est estimée à 10 ans.
Le PCF2129A doit être alimenté en 3,3 V, tension heureusement présente sur l'Arduino Uno. Les broches I²C supportent elles 5 V sans problème, il ne faut donc pas d'adaptation de niveau pour la connexion à l'Arduino.

Le croquis

Le croquis de l'horloge de sable a été écrit pour l'Arduino Uno, mais il tourne aussi sur la carte Elektor Uno R4 ; il contient toutes les fonctions nécessaires à l'écriture de l'heure dans le sable, et au réglage de l'horloge après son montage. Les procédures détaillées du montage et du réglage, ainsi que le croquis, peuvent être téléchargés sur la page de l'article [3].

Certaines fonctions sont également accessibles via des commandes, ce qui permet de bouger le stylet, de dessiner des figures dans le sable, ou de faire fonctionner les moteurs vibrants. Le croquis accepte donc deux modes : autonome et commande ; dans le premier mode, l'horloge fonctionne normalement, dans le second le croquis exécute les commandes qui lui sont transmises.

Après chargement du croquis, on peut envoyer des commandes à l'horloge via le moniteur série de l'Arduino : on choisit *9600 baud* puis, à l'invite de commande suivante, autre chose que *No line ending*. Les commandes comportent des caractères (p. ex. ***svd***), éventuellement suivis de paramètres (p. ex. ***ps-20.55 +55.8***). Une pression sur la touche *ENTER* expédie la commande ; l'Arduino envoie une réponse à la plupart des commandes. Toutes les commandes disponibles sont documentées au début du croquis ; on peut par exemple dessiner un arc avec ***pa***.

Revenons maintenant à la procédure de réglage initial, qui demande l'introduction d'une série de commandes, certaines avec paramètres. Lorsque tous les réglages sont terminés (voir commande ***sed***), les valeurs sont sauvegardées dans la mémoire EEPROM de l'Arduino (commande ***sew***). Au démarrage, le croquis vérifie que le contenu de l'EEPROM est valide, et si c'est le cas il lance l'horloge dans le mode choisi (aussi entreposé dans l'EEPROM), en principe autonome. Si le contenu de la mémoire n'est pas validé, le démarrage se fera en mode commande, comme lorsqu'on charge le croquis pour la première fois.

Notez que le contenu de la mémoire EEPROM n'est pas effacé lors d'un chargement du croquis ; il n'est pas nécessaire de tout régler à nouveau et d'entreposer les valeurs avec la commande ***sew***. Le contenu de la mémoire peut être effacé avec la commande ***sec***.

En mode autonome, l'heure est écrite périodiquement dans le sable. Ce dessin consiste en une série de déplacements du stylet, tant dans le plan horizontal que dans le plan vertical. Le croquis prend toujours la position actuelle comme point de départ pour les mouvements : tracer une ligne, suivre un arc, dessiner un caractère, ou déplacer le stylet.

Liste des composants

Résistances
R1 à R3 = 220 Ω, CMS 0805
R4 = 1,8 kΩ, CMS 0805
R5 à R9 = 4,7 kΩ, CMS 0805
R10 = 100 kΩ, CMS 0805

Condensateurs
C1 à C4 = 100 nF, CMS 0805 MLCC
C5 = 10 µF / 10 V, CMS 1206 MLCC
C6 = 22 µF / 16 V

Semi-conducteurs
D1 à D3 = S2J-E3
T1 = BCX56
IC1 = convertisseur CC-CC, 5 V / 1 A, SIP3 (Würth Elektronik 173 010 578)
IC2 = PCF2129A

Divers
K1 = jeu de barrettes sécables, contacts mâles, pas de 2,54 mm (1×10, 2×8, 1×6)
SERVO_Z, _L, _R = barrette sécable coudée, 3 contacts mâles, pas de 2,54 mm
B1 = pile bouton CR2032, avec support pour circuit imprimé (p. ex. Multicomp CH25-2032LF)
Arduino Uno R3, ou équivalent

Composants mécaniques
6 vis M2×10, tête Phillips ou Pozidriv
6 vis M2, 5×8, acier zingué, Pozidriv DIN 7985A
6 vis M2, 5×12, acier zingué, Pozidriv DIN 7985A
2 vis M3×6, acier zingué, Pozidriv DIN 7985A
2 vis M3×8, acier zingué, Pozidriv DIN 7985A
15 vis M3×10, acier zingué, Pozidriv DIN 7985A
1 vis M4×30, plastique, tête Phillips (pointe aiguisée au taille-crayon)
6 écrous hexagonaux M2, acier zingué, DIN 934
7 écrous hexagonaux M3, acier zingué, DIN 934
3 écrous de blocage M3, acier zingué, DIN 985
1 écrou hexagonal M4, polyamide
2 rondelles M3, acier zingué, DIN 125A
4 rondelles M3, plastique, DIN 125A
4 entretoises pour M3, 3 mm, polyamide
4 entretoises hexagonales M/F pour M3, 25 mm, laiton nickelé (hauteur totale minimale de 33 mm, p. ex. TME TFM-M3X25/DR213)
2 serre-câbles, polyamide, Panduit CCS25-S10-C
4 patins adhésifs en caoutchouc
3 servos, Tower Pro MG90S ou MG90, avec engrenages métalliques
2 moteurs vibrants, 6 mm de diamètre, VM6ZK273 (450-007 JPR Electronics)
Plexiglas extrudé, 3 mm, transparent, découpé au laser
Sable blanc fin
Colorant alimentaire (facultatif)

Un kit complet, avec tous les composants et les CMS soudés, est disponible dans l'e-choppe d'Elektor.

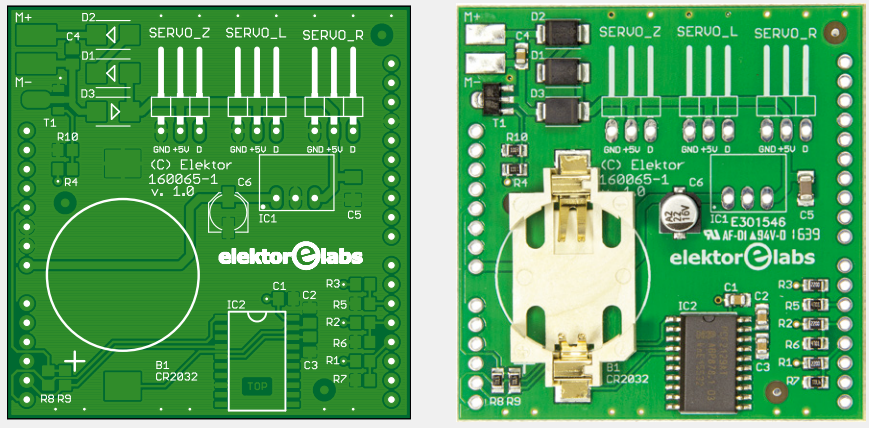


Figure 4. Essentiellement des CMS ; ils sont déjà soudés sur le circuit imprimé fourni avec le kit.

La commande ***pen_set*** déplace le stylet vers les coordonnées (x, y) choisies. Ces coordonnées sont d'abord transformées en radians pour le déplacement angulaire des servos gauche et droite ; les calculs correspondants sont effectués par la fonction ***pen_calc***, qui détermine aussi la faisabilité du déplacement (la possibilité pour le stylet d'atteindre la position souhaitée). Si le déplacement est valide, les mouvements angulaires des moteurs seront transformés en trains d'impulsions de largeur appropriée.

Les calculs de cinématique inverse sont effectués en virgule flottante ; les nombres sont représentés en simple précision sur 32 bits, suivant la norme IEEE 754. Le compilateur *AVR-GCC* s'occupe de ces particularités lors de la compilation du croquis.
Les positions et distances sont exprimées en millimètres ; ces valeurs concordent avec les dimensions réelles de l'horloge. Les dimensions des composants mécaniques sont connues par le croquis. Les angles sont mesurés en radians.
La **figure 5** montre les positions, en mm, des servos gauche et droite par rapport au bac à sable, telles que définies dans le croquis.

Lors de l'écriture, le stylet se déplace par pas inférieurs ou égaux à 0,25 mm. Malgré les nombreux calculs que cela implique, le processeur n'est pas surchargé, et il y a un délai de 5 ms entre les déplacements du stylet lors de l'écriture, pour que le mouvement dans le sable soit plus précis et posé.

On peut interrompre le mode autonome à tout moment en envoyant un caractère de fin de ligne à l'Arduino (qui doit bien entendu être connecté à un ordinateur). Appuyez sur la touche *ENTER*, et le croquis passera en mode commande.
La prise en compte de cette interruption peut prendre un certain temps, si l'horloge est en train d'écrire au moment de l'envoi. On repasse en mode autonome avec la commande ***ma***.

Montage

Dans le cadre de cet article, nous nous limitons au montage du kit. Ceux qui veulent construire l'horloge eux-mêmes devront se référer aux instructions détaillées avec photos [3].

Il faut d'abord souder les connecteurs et le convertisseur CC-CC sur le *shield*. On installe celui-ci sur l'Arduino et on connecte les servos, en veillant à une orientation correcte. On alimente ensuite l'Arduino avec l'adaptateur secteur, et on le relie à un ordinateur, sur lequel est installé l'environnement de développement intégré (*Integrated Development Environment*) de l'Arduino. On compile le croquis [3], on le charge sur la carte Arduino, et on la redémarre ; lorsque cette opération est terminée, les servos se positionnent automatiquement en position médiane (largeur d'impulsion de 1,5 ms). Ensuite déconnectez l'ordinateur, retirez l'adaptateur secteur, le *shield* et les servos, afin de poursuivre le montage. Il est à noter que si vous disposez d'un testeur de servo, vous pouvez l'utiliser pour les caler en position médiane.
Passons maintenant à la partie mécanique. On colle quatre patins sous la plaque de base, et on monte les supports du bac à sable. On installe ensuite le châssis principal avec le servo élévateur, et on le place avec la carte Arduino et le *shield* sur la plaque de base.
On monte alors les servos gauche et droite, on les reconnecte au *shield*, et on installe le pantographe avec son stylet. Pour monter les bras du pantographe, on visse directement dans le Plexiglas ; ceci est permis par les propriétés thermoplastiques du matériau : lors du vissage, il y a en effet un échauffement autour de la pointe des vis suite au frottement, ce qui fait fondre le Plexiglas. Cette technique permet de minimaliser le jeu dans les bras. Les trous du stylet, qui est constitué d'une vis aiguisée au taille-crayon, sont pourvus d'un pas de vis ; si vous découpez les bras vous-même avec une découpeuse au laser, il faudra utiliser un taraud M4 pour ces trous.
Montez le bac à sable, mais sans les moteurs vibrants, car il faut d'abord procéder à quelques réglages. Alimentez la carte Arduino et raccordez-la à un ordinateur ; les commandes pour positionner les servos gauche et droite pourront être introduites via l'EDI. Il faut déterminer les positions des servos pour qu'ils soient exactement à l'horizontale et à la verticale ; ces positions seront alors sauvegardées dans la mémoire EEPROM de l'Arduino. On règle ensuite le servo élévateur : installez le bac à sable et positionnez le servo pour que le stylet soit quelques millimètres au-dessus de la surface, quelle que soit sa position dans le plan. Comme le stylet est une vis, on pourra toujours affiner la position par la suite, et la caler avec un écrou M4. On règle ensuite les positions médiane et haute du servo élévateur, mais celles-ci ne sont pas du tout critiques, et on peut se contenter des valeurs données dans les instructions de montage.

Une fois les positions sauvegardées dans l'EEPROM, réglez l'heure. Vous pouvez alors tester le bon fonctionnement en l'inscrivant « à sec », et si tout fonctionne bien passer en mode autonome (l'heure s'écrira à intervalles réguliers), et supprimer la liaison avec l'ordinateur. Arrêtez l'Arduino, et montez les moteurs vibrants sous le bac à sable ; il ne vous restera plus qu'à souder les fils de connexion aux endroits prévus du *shield*, en veillant à ne pas toucher une partie en Plexiglas avec le fer à souder. Il ne vous reste plus qu'à remettre le bac à sable en place, et à vérifier qu'aucun fil ne gêne le fonctionnement des moteurs. Remplissez le bac de sable, redémarrez l'Arduino, et admirez votre travail !

Si vous souhaitez ajouter un peu de couleur, placez le sable et quelques gouttes de colorant alimentaire dans un récipient

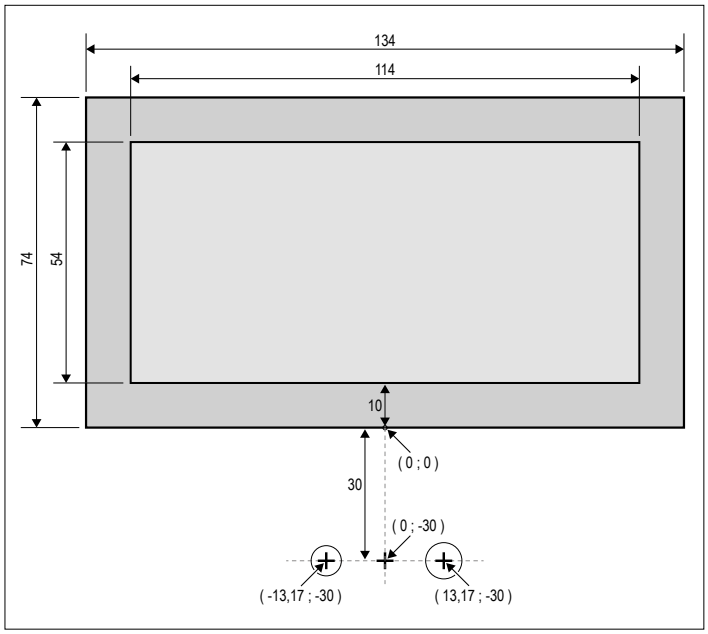


Figure 5. Positions, en mm, des servos gauche et droite par rapport au bac à sable, telles que définies dans le croquis.

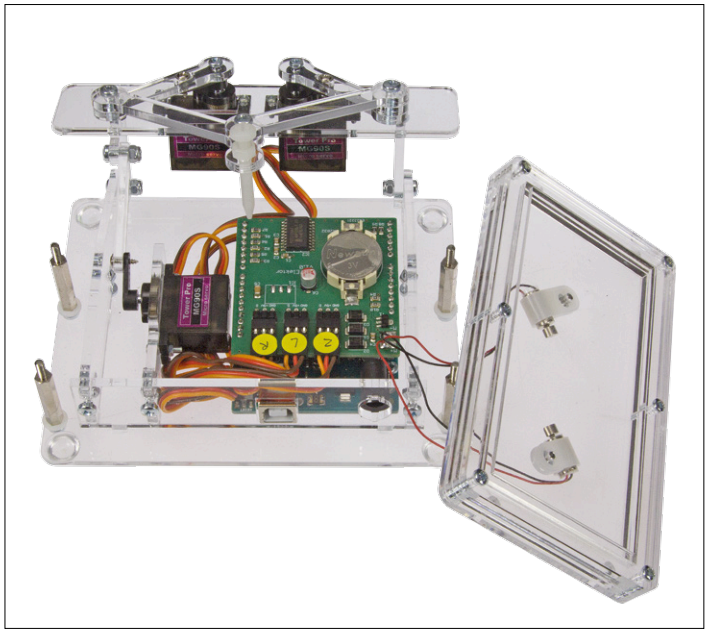


Figure 6. La structure mécanique de l'horloge de sable est bien visible sur cette photo.

clos, et secouez pour obtenir un mélange uniforme. Recommencez jusqu'à l'obtention de la teinte souhaitée. Ne remettez le sable dans le bac que lorsqu'il est tout à fait sec.

(160065 – version française : Jean-Louis Mehren)

Liens

[1] www.thingiverse.com/thing:248009

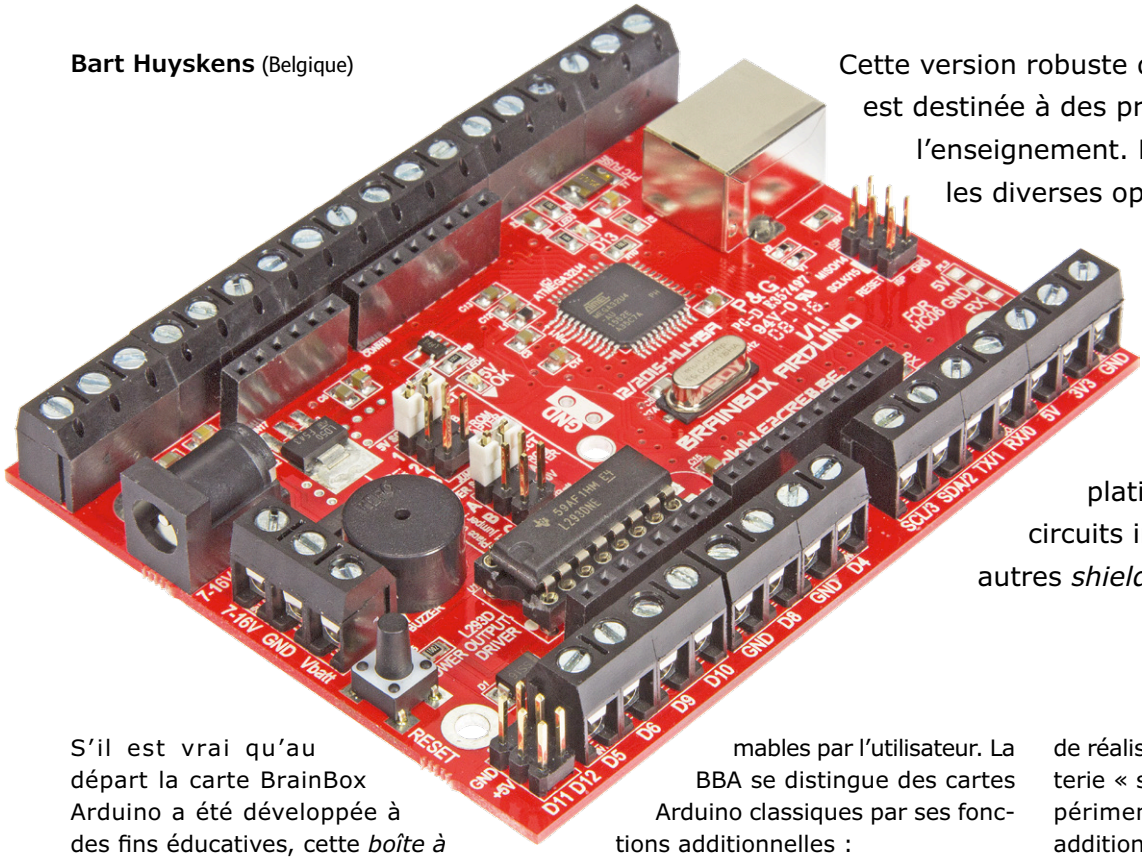
[2] www.elektormagazine.fr/160065

[3] www.elektor.fr/sandclock-160065-71

BrainBox Arduino

un Arduino « costaud » avec bornes à vis

Bart Huyskens (Belgique)



Cette version robuste de l'Arduino Leonardo est destinée à des projets autonomes et à l'enseignement. Les solides bornes à vis, les diverses options d'alimentation, le *buzzer* intégré et le pilote pour la commande directe de moteurs permettent de se passer, pour la plupart des applications, de platine d'expérimentation, de circuits intégrés additionnels et autres *shields*.

S'il est vrai qu'au départ la carte BrainBox Arduino a été développée à des fins éducatives, cette *boîte à malices* n'en est pas moins si polyvalente qu'elle peut être utilisée pour nombre d'autres projets électroniques. Tout le monde peut, en un rien de temps, programmer la BrainBox Arduino (alias BBA pour la suite) dans le langage de programmation qui lui convient ; les exemples de programmes disponibles ont été développés pour cinq environnements de développement différents. La BBA prête à l'emploi, avec chargeur d'amorçage (*bootloader*) préprogrammé, est disponible dans la boutique Elektor ; il n'y a donc pas de soucis de soudage. En outre il existe un kit éducatif avec tout le matériel, les logiciels et les didacticiels pour construire, entre autres, un robot « automobile » (voir photo à la fin de l'article) et le piloter avec une application maison.

Les extras

Le cœur de la BBA est le puissant processeur de la carte Leonardo, un ATmega32U4, cadencé à 16 MHz. Ce µC est directement programmable par USB. À l'image de l'Arduino Leonardo, la BBA possède quatre LED dont deux programmables par l'utilisateur. La BBA se distingue des cartes Arduino classiques par ses fonctions additionnelles :

- Un *buzzer* rend audible n'importe quelle fréquence : idéal pour tester des sonneries maison.
- Un double pont en H attaque quatre sorties de puissance qui peuvent fournir jusqu'à 600 mA/broche. Il permet de piloter quatre moteurs CC en demi-pont ou deux moteurs CC en pont complet (*full bridge*). Ces sorties peuvent aussi être utilisées pour, par ex., piloter des LED de puissance, produire de la chaleur avec des résistances de puissance, voire piloter des moteurs pas à pas. Des cavaliers permettent à l'utilisateur de choisir la tension d'alimentation des quatre sorties de puissance (4,5 à 36 V) : 5 V, adaptateur secteur, piles ou alimentation externe.
- Un cavalier permet de sélectionner l'alimentation du µC : USB, adaptateur ou piles.
- Deux connecteurs pour des servomoteurs.
- Contacts pour des modules Bluetooth HC06, RS-232 et I²C.

Ces fonctions supplémentaires permettent de réaliser nombre de projets sans circuiterie « spaghetti » sur une plaque d'expérimentation et sans *shields* Arduino additionnels.

Le schéma

La **figure 1** donne le schéma complet de la BBA qui peut paraître compliqué. Les débutants et les enseignants préféreront la **figure 2**, on y voit clairement l'affectation de toutes les bornes à vis et les principaux composants de la carte avec leurs fonctions.

Cependant, pour ceux qui souhaitent en savoir plus sur le schéma, passons en revue ses éléments constitutifs. Au cœur de la BBA, on trouve un ATmega32U4. Ce µC à 8 bits RISC, basse consommation, possède 32 Ko de mémoire flash, 2,5 Ko de SRAM, 1 Ko d'EEPROM ; il intègre une interface USB 2.0 pleine vitesse, un convertisseur A/N 10 bits à 12 canaux et une interface JTAG pour le débogage *in situ*. Le µC est cadencé à la fréquence d'horloge maximale de 16 MHz par le quartz XTAL1. Le connecteur USB CONN4 attaque directement le µC via les résistances de protection R3 et R4. Ses bornes d'alimentation sont reliées, au travers d'un fusible réarmable (T1), à JUMPER1 pour la sélection de l'alimen-

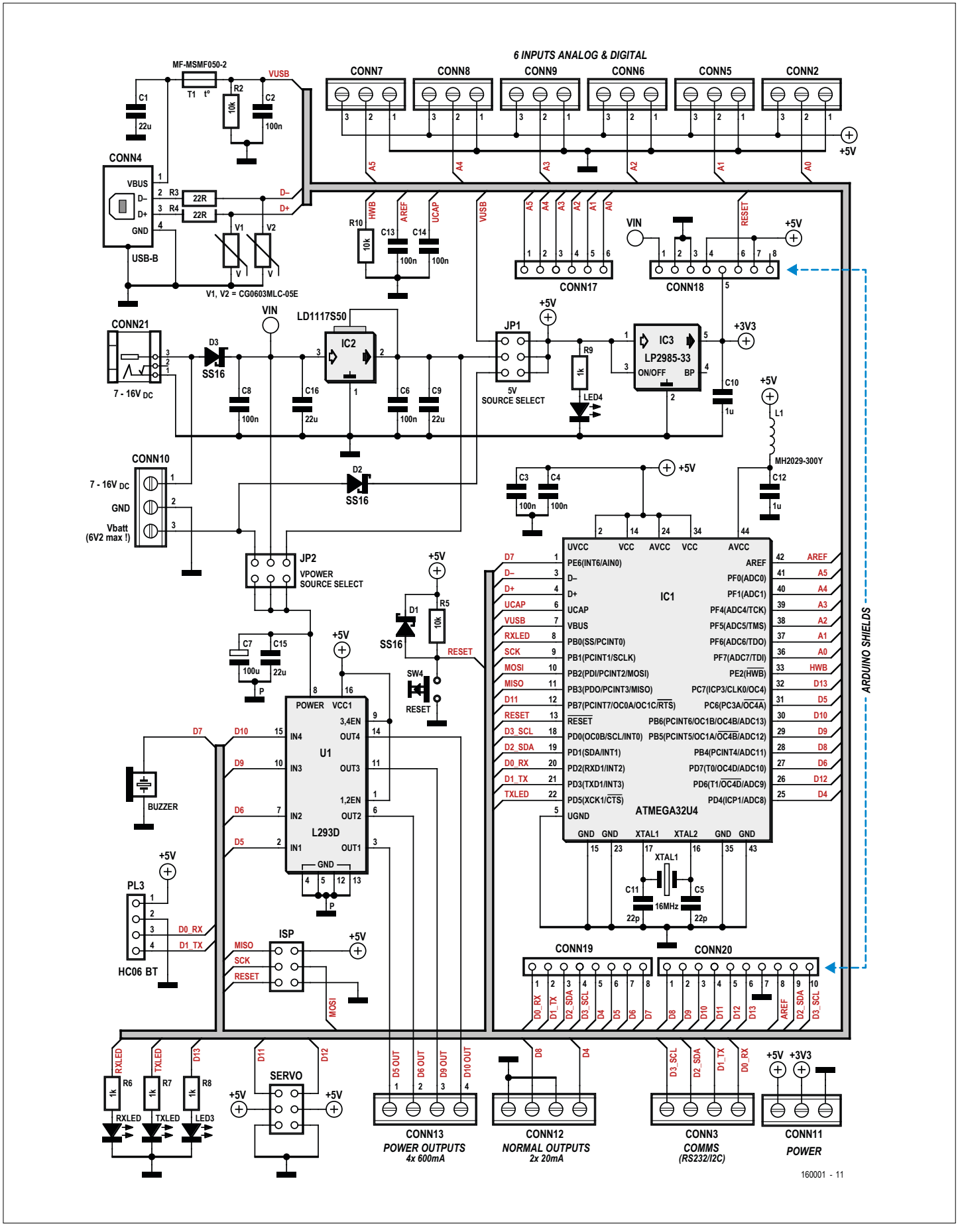


Figure 1. Le schéma de la BrainBox Arduino, au cœur duquel trône un ATmega32U4.

tation de la carte.
Le circuit de commande de moteur U1, un L293D, contient (cf. plus haut) un double pont en H qui pilote quatre sorties de puissance (CONN13), courant maximal de 600 mA/broche. L'électronique du circuit intégré est alimentée par le régulateur IC2. Les étages de puissance du circuit intégré disposent eux d'une connexion à l'alimentation séparée ; JUMPER2 permet de sélectionner la source d'alimentation.
Il y a en outre quatre LED, deux pour superviser la communication série via CONN3 (RXLED est programmable), une LED pour l'alimentation et une LED librement programmable (LED3). Un *buzzer* attaque la broche 1 du µC et un bouton de remise à zéro sa broche 13.
Comme les sources d'alimentation possibles sont diverses, le bloc d'alimentation est relativement complexe. CONN21 permet la connexion directe d'un adaptateur secteur (à tension de sortie entre 7 et 16 V). On peut aussi utiliser le bornier à trois pôles CONN10 pour connecter un bloc d'alimentation secteur ou un jeu de piles qui délivre max. 6,2 V. La tension CC comprise entre 7 et 16 V attaque, au travers de la diode de protection contre une inversion de polarité D3, le régulateur 5 V (IC2) chargé de l'alimentation du µC et du circuit pilote de moteurs (U1). En aval il y a un régulateur de 3,3 V (IC3), dont la tension de sortie est disponible sur les connecteurs de *shield* et sur le bornier d'alimentation CONN11.
Les cavaliers implantés sur JUMPER1 et JUMPER2 donnent le choix entre les différentes options d'alimentation. JUMPER1 définit la source d'alimentation de la carte : connecteur USB, adaptateur secteur ou jeu de piles. JUMPER2 détermine la source d'alimentation des moteurs connectés à U1, le circuit intégré de commande de moteurs. On a le choix ici entre le 5 V régulé d'IC2, l'adaptateur secteur ou les piles.
Il reste des connecteurs enfichables et des bornes à vis. Les embases CONN17 à CONN20 servent à accueillir des *shields* Arduino. Sur l'un des côtés de la carte, on a six groupes de trois bornes à vis pour les entrées analogiques et numériques (CONN2, CONN5 à CONN9) avec pour chacun sa propre borne d'alimentation 5 V. À l'opposé, on trouve les borniers pour la commande de moteur (CONN13), deux sorties numériques (CONN12), les communications série et I^2C (CONN3) et pour finir le connecteur CONN11 sur lequel on a toutes les tensions d'alimentation. Enfin on arrive à une embase

Options logicielles

Il existe, parallèlement à l'EDI Arduino, d'autres environnements de programmation dans lesquels la BBA se laisse programmer. Un atout majeur de ce projet est qu'il est proposé avec des exemples de programmes pour de nombreuses situations différentes et pour chacun des cinq environnements de programmation mentionnés ci-après. On programmera donc la BBA dans l'environnement que l'on préfère.

EDI Arduino avec bibliothèques
L'EDI Arduino original est un simple compilateur C épaulé par un grand nombre de bibliothèques. Si ces bibliothèques et la numérotation de broches unique permettent d'écrire relativement vite des programmes complexes sans bien connaître les µC, elles ont l'inconvénient de ne pas permettre d'apprendre grand-chose sur ces derniers.

EDI Arduino sans bibliothèques
Ce que l'on ignore souvent c'est que l'EDI Arduino peut aussi être utilisé pour programmer Arduino directement avec les noms des registres et les numéros de broches. Cela force l'utilisateur à apprendre à connaître le µC avant de pouvoir écrire des programmes. On peut aussi combiner les deux systèmes – avec et sans bibliothèques.

Flowcode (7) pour AVR
D'un point de vue didactique, Flowcode est indéniablement la meilleure façon de se familiariser avec la programmation enfouie. Les diagrammes permettent de visualiser les structures de programmation et le simulateur permet de tester le code exhaustivement avant de l'« expédier » au matériel. Bien que graphique et épaulé par de nombreuses bibliothèques qui simplifient considérablement la programmation, Flowcode reste sensiblement plus près de la vraie « programmation enfouie » que l'EDI Arduino par exemple. C'est donc **le** tremplin idéal pour l'*Embedded C*.

Atmel Studio 7 avec compilateur GCC
Atmel Studio est l'environnement de développement professionnel pour les µC Atmel et GCC est probablement le compilateur C gratuit le plus largement utilisé avec les µC Atmel AVR. La BBA est facile à programmer dans cet environnement. Et si vous voulez, vous pouvez toujours, même dans cet environnement, utiliser les vastes bibliothèques Arduino pour simplifier certaines fonctions.

Snap 4 Arduino
S4A permet maintenant aussi de piloter la BBA grâce aux très populaires « blocs de puzzle Scratch ». Il établit une liaison stable avec un BBA ; il faudra cependant commencer par charger le programme « Firmata ». On a ensuite une exécution en direct sur l'écran – un peu à l'image d'un émulateur – de toutes les instructions sur le matériel connecté à la BBA.
Cette version bêta convertit les programmes les plus élémentaires en code réel à charger ensuite dans l'EDI Arduino ; certains programmes peuvent ainsi aussi fonctionner « *offline* ». Un début prometteur ; S4A permet idéalement de transmettre un rien de notre enthousiasme pour l'électronique à de jeunes utilisateurs.

femelle à 6 contacts (SERVO) pour la commande de deux servos, un connecteur ISP à 6 contacts (ISP) pour la programmation directe du µC et un connecteur à 4 contacts (PL3) sur lequel pourra s'insérer un module Bluetooth HC06.
Il n'y a pas de liste de composants puisque la BBA est, en raison de son caractère éducatif, livrée montée et préprogrammée.

Vue d'ensemble

Le routage de la BBA est, à dessein, structuré selon le principe ETS : Entrée – Traitement – Sortie ; en haut de la figure 1, toutes les connexions pour les capteurs, le traitement est à la charge du puissant µC qu'est l'ATmega32U4 et en bas toutes les connexions pour les actionneurs.
Les capteurs et actionneurs peuvent être connectés directement via de robustes borniers à vis de 5 mm ; ces derniers sont tous configurables en entrée analogique ou numérique. Chaque entrée possède ses propres connexions de masse (*GND*) et 5 V. On y trouve aussi des connecteurs Arduino pour la connexion de *shields* Arduino existants.

Comme la BBA autorise différentes sources d'alimentation, elle peut être utilisée pour les applications les plus diverses. Sur le côté droit de la carte, on trouve les options de programmation. La BBA est préprogrammée avec un chargeur d'amorçage Arduino Leonardo. On peut donc, dès le départ, programmer la BBA par USB. Si l'on veut programmer le µC ATmega32U4 sans *bootloader*, c'est possible via l'embase ISP à 6 broches.
Côté gauche, on a deux connecteurs pour servomoteurs et quatre sorties de puissance de type MLI (*PWM*). Attaquées par un double pont en H elles peuvent débiter jusqu'à 600 mA/broche.
En bas à droite, on trouve les deux canaux de communication les plus populaires, I²C et RS-232 qui permettent de connecter à la BBA capteurs, actionneurs et écrans LCD les plus divers. L'embase à 4 broches dans le coin droit permet d'établir un lien direct avec les modules Bluetooth (HC06) populaires.
Des exemples de programmes tant pour la BBA que pour APPINVENTOR vous aideront à écrire des applis pour ordiphone afin de communiquer avec votre BBA. Essayez ! Vous serez étonné de la rapidité avec laquelle votre appli sera opérationnelle. Vous trouverez des didacticiels et d'autres infos (en néerlandais) à l'adresse [1].

(160001 - version française : Guy Raedersdorf)

Connexions

Les numéros de broches de couleur de ce synoptique sont les dénominations de broches utilisées par l'EDI Arduino et Snap4Arduino. Les numéros de broches en gris sont les désignations réelles des broches, issues de la fiche technique, utilisées par AVRStudio et Flowcode.

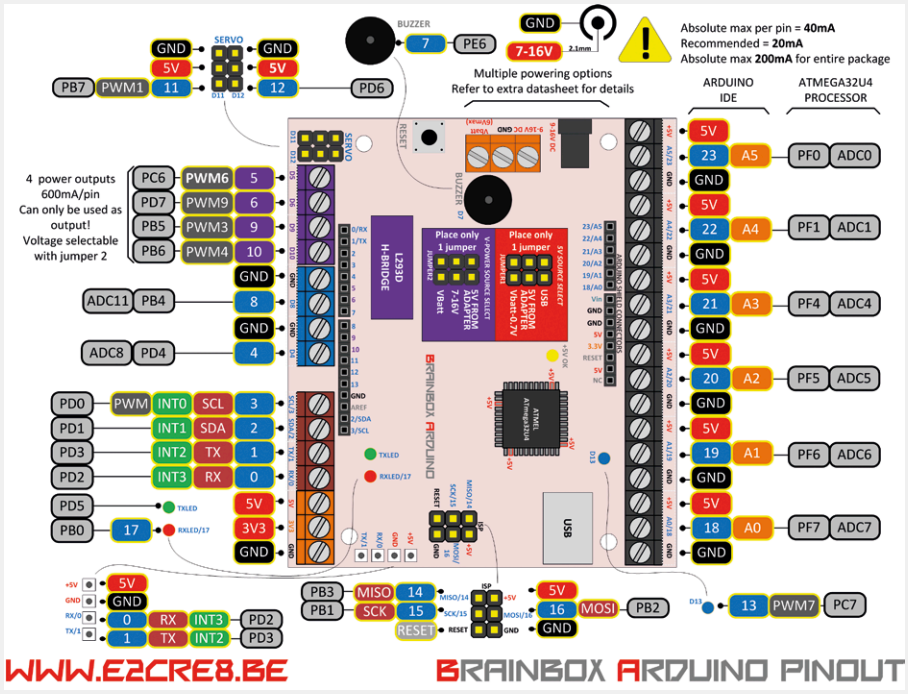


Figure 2. Infos de connexion de la carte avec spécifications succinctes.

Lien

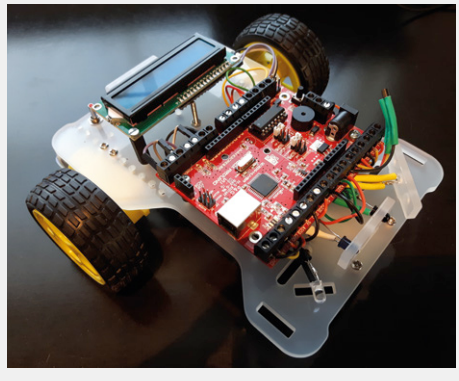
http://e2cre8.be/

L'auteur

Bart Huyskens est professeur d'électronique et de TIC en Belgique. Il organise également des ateliers et développe des matériels, logiciels et didacticiels d'enseignement pour permettre un enseignement des matières STEM d'investigation et de création.

Kit BrainBox Arduino

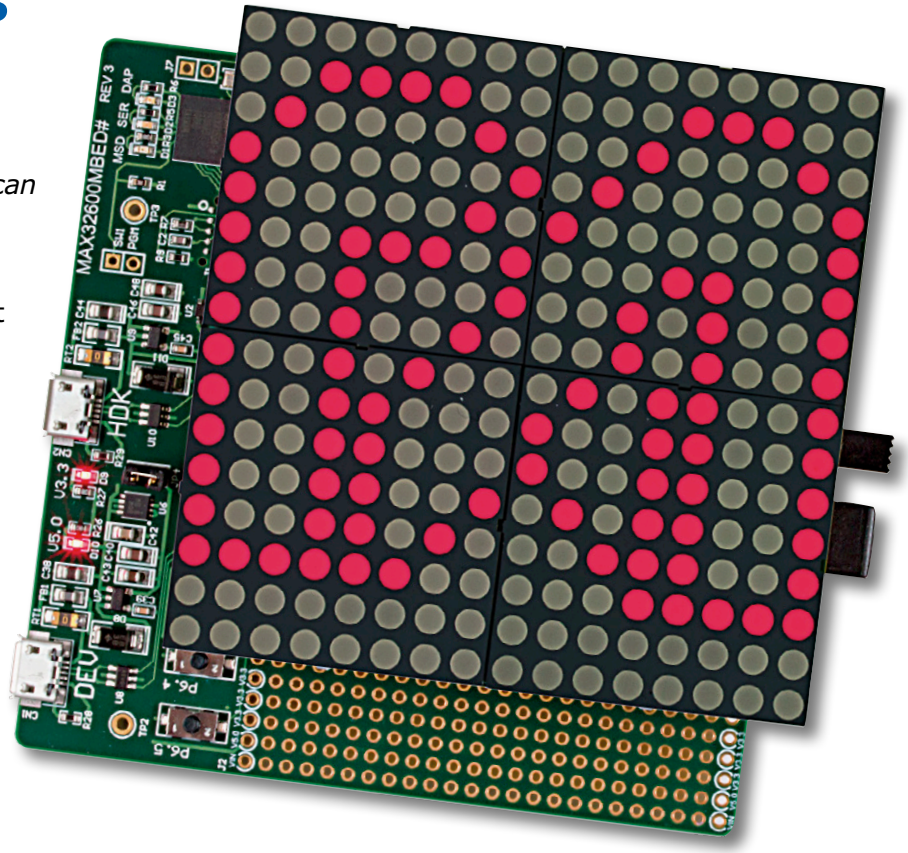
Le kit Arduino BrainBox disponible dans l'e-choppe d'Elektor est livré doté de tous les capteurs, actionneurs, morceaux de fils, pièces mécaniques, informations et exemples de programmes nécessaires pour construire le robot automobile représenté et le commander avec une appli maison. Mais donnez plutôt libre cours à votre créativité pour mettre en œuvre la BBA dans vos propres réalisations.



shield d'affichage MAXREFDES99#

256 LED à vos ordres

Clemens Valens (labo d´´Elektor)



Publié pour la première fois en 1963, l'*American Standard Code for Information Interchange*, connu depuis comme l'ASCII, était basé sur l'alphabet anglais dont les caractères peuvent être affichés avec seulement 5×7 pixels, voire moins. Aujourd'hui, pour pouvoir tout afficher de l'arabe au zoulou, la signalétique électronique demande plus de pixels par caractère. Le MAXREFDES99# avec ses 256 pixels est une solution simple pour traiter les caractères les plus complexes.

Le *shield*...

... RD99 pour les intimes, est une carte d'extension compatible Arduino qui comporte une matrice de 16×16 LED (en réalité quatre matrices de 8×8) et ses pilotes associés. Ce carré de 7,7×7,7 mm (3×3 pouces) est destiné à la signalétique et ses 256 LED permettent d'afficher n'importe quel caractère alphanumérique international. Il peut aussi afficher des messages défilants en police de 5×7 ou autre. Ce qu'il affiche exactement dépend de vous.

Des bibliothèques à code source ouvert (C++) prennent en charge le *shield* sur les plateformes de développement rapide Arduino et ARM mbed. Elles assurent non seulement la communication bas niveau avec l'afficheur, mais aussi des fonctions de haut niveau pour afficher les caractères imprimables de « *ESPACE* » (32) à « ~ » (126) et les messages défilants.

Le *shield* doit être alimenté en 5 V par le système hôte ou par l'alimentation intégrée, suivant la position du commutateur SW1. La tension d'entrée du régulateur est limitée soit par la tension maximale autorisée sur la broche V_{IN} du système hôte (connecteur H3, broche 8) soit par la tension de claquage de C1 (25 V). Pour un hôte Arduino R3, c'est 20 V_{CC} mais il est recommandé de se limiter à 12 V. Le kit est livré avec un bloc secteur de 9 V_{CC}/1,3 A avec une fiche américaine, ce qui est dommage puisqu'il peut accepter entre 100 et 240 V_{CA}.

Le RD99 mesure la tension de la broche IOREF (H3, broche 2) qui, dans les systèmes Arduino, indique le niveau de tension d'E/S de l'hôte. Ce niveau peut descendre jusqu'à 1,2 V grâce à un circuit de décalage automatique du niveau.

La communication avec l'hôte utilise seulement trois signaux : horloge (front montant sur D13, H4, broche 5), données (D11, H14, broche 7) et signal de chargement (front montant sur D10, H4, broche 8). Ces signaux sont compatibles avec les bus SPI à 16 bits dont le signal *Slave Select* (SS) est utilisable comme signal de chargement. Ajoutez à ces signaux la ligne de remise à zéro (active au niveau bas, H3, broche 3), la tension IOREF, l'alimentation, et vous avez les huit connexions de cette extension dernier cri.

MAX7219

Le RD99 a quatre pilotes d'afficheur à LED MAX7219, un pour chaque matrice de 8×8 LED. Chacun peut piloter jusqu'à huit afficheurs à 7 segments (avec point décimal) ou jusqu'à 64 LED individuelles. Le MAX7219 comporte un décodeur BCD code B, un circuit de balayage multiplexé, les pilotes de segments et de digits ainsi qu'une RAM statique de 8×8 bits pour stocker l'état de chaque LED. Une interface série permet de programmer les registres et bien sûr d'activer ou désactiver individuellement les LED. Chaque digit (ou groupe de huit LED) est accessible et peut être mis à jour sans rafraîchir tout l'affichage. Le décodeur BCD intégré limite le travail en amont et peut être désactivé s'il n'est pas nécessaire.

La luminosité des LED est déterminée par une seule résistance externe qui fixe le courant qui les traverse ; quatre résistances pour les quatre matrices de 8×8 LED. On peut aussi régler numériquement (MLI) la luminosité de l'afficheur avec le registre d'intensité.
Une interface série à 4 fils permet de connecter simplement un microcontrôleur. Le protocole série est très similaire au SPI à 16 bits. Grâce à l'absence de contrainte stricte de synchronisation, un µC avec un périphérique SPI à 8 bits, donc incapable de dialoguer avec le MAX7219, pourra envoyer facilement les données par logiciel (*bit bang*). Il faut retenir que les données passent au travers du registre à décalage interne du circuit, quel que soit l'état de la broche LOAD. Un front montant sur cette broche charge la donnée du registre à décalage dans la mémoire interne et met à jour l'afficheur, donc attention à charger au bon moment.

Utilisation avec Arduino

Dans ce qui suit, on suppose qu'on dispose d'un environnement de développement Arduino fonctionnel et d'une carte Arduino Uno R3. Avant de connecter le RD99 sur l'Uno, vérifiez que le *shield* fonctionne. Pour cela, branchez le bloc secteur et mettez SW1 du côté de la prise d'alimentation (sélection de l'alimentation externe). Une LED de présence d'alimentation devrait s'allumer (quelle que soit la position de SW1). Coupez l'alimentation et connectez le RD99 à l'Uno (sans changer la position de SW1), rebranchez l'alimentation et connectez l'Uno à votre ordinateur.

En [1], dans l'onglet *Design Resources,* téléchargez le logiciel pour la plateforme Arduino. Décompressez ce fichier dans un dossier, puis copiez le dossier *MAX7219* dans le dossier *libraries* du carnet de croquis Arduino (pour le trouver : menu *Fichier -> Préférences*). Créez le dossier *libraries* s'il n'existe pas.
Lancez l'EDI Arduino, allez dans *Fichier -> Examples -> MAX7219* et ouvrez *MAXREFDES99_example*. Si vous ne voyez pas cet exemple, c'est que la bibliothèque n'est pas installée correctement ou que l'EDI était ouvert lorsque vous avez copié la bibliothèque. Si c'était le cas, fermez et redémarrez l'EDI. Cliquez sur le bouton *Téléverser* pour transférer le programme dans l'Uno. Pour moi cela ne fonctionnait qu'avec le *shield* alimenté par le bloc secteur.

Lorsque le transfert est terminé, ouvrez le moniteur série de l'EDI et réglez le débit sur 115,2 kbauds. Un menu devrait apparaître. Sinon, pressez le bouton de RàZ sur l'Uno. Tapez « 6 » dans la boîte d'envoi et cliquez sur le bouton *Envoi*. Au début on dirait que rien ne se passe, mais après environ 5 s, votre commande est soudainement acquittée, l'afficheur s'allume et une splendide démonstration démarre.
Si vous êtes attentif, vous remarquerez à la fin de la démo un problème avec le dernier caractère dans le coin inférieur droit. On dirait qu'un pixel a sauté de sa ligne, transformant le « m » en un caractère étrange. L'inspection du caractère « m » dans le fichier *maxrefeds99.cpp* (déjà ouvert dans l'EDI) montre pourquoi (ligne 130) : la séquence « 0x58, 0x44 » devrait être « 0x78, 0x04" ». (Pourquoi ? Écrivez ces nombres sous forme de séquence binaire où un « 1 » est un point et un « 0 » rien). Corrigez, téléversez le programme et relancez la démo n°6. Tout devrait être bon maintenant.

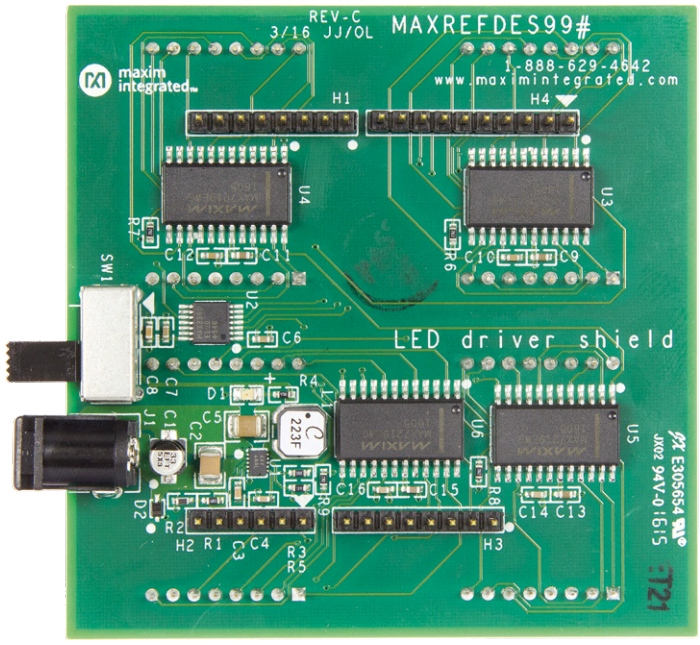


Figure 1. Dessous du *shield*.

Animation simple

Il y a un deuxième exemple nommé *MAX7219_example*. Ouvrez-le, téléversez-le dans la carte, regardez ce qu'il fait et étudiez le code source. C'est assez simple et montre clairement comment allumer des pixels individuels et où ils se situent sur la grille. Inspiré par cet exemple, j'ai créé une petite animation plein écran que vous pouvez télécharger depuis [3]. L'animation consiste en une séquence de onze trames affichées successivement toutes les 25 ms. La dernière trame est affichée pendant 1,5 s, puis l'animation repart. Un second croquis montre une façon de faire défiler des grands caractères personnalisés. Les vidéos de ces croquis sont disponibles en [3].

Après avoir étudié les exemples fournis, vous devriez être prêts à créer vos propres applications de signalisation. Amusez-vous bien !

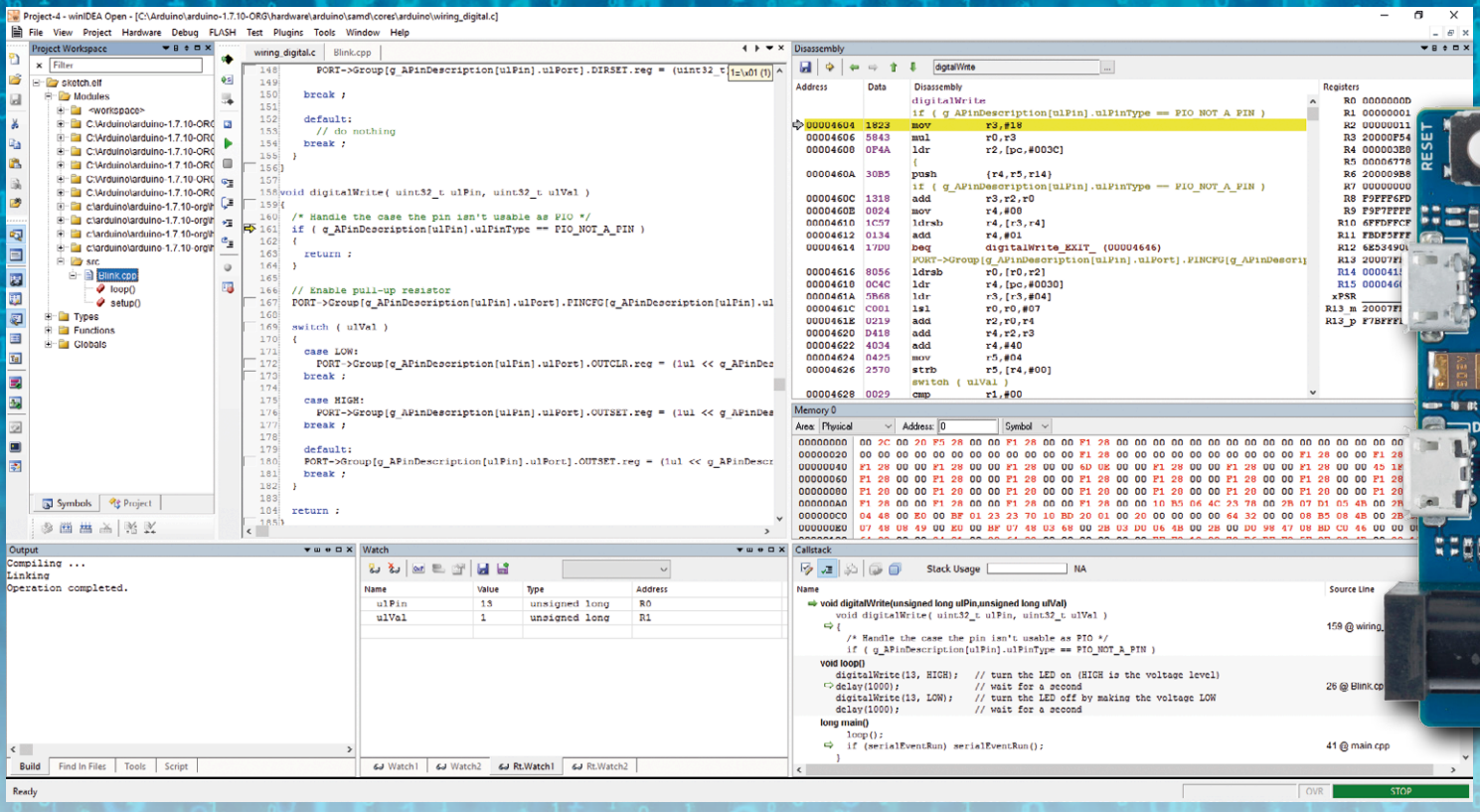
(160267 – version française : Denis Lafourcade)

Liens

[1] www.mouser.com/new/maxim-integrated/maxim-maxrefdes99
[2] www.maximintegrated.com/MAXREFDES99
[3] www.elektormagazine.com/labs/maxrefdes99-led-shield-experiments-160267

débogage sur Arduino Zero & M0 Pro

plongée au cœur du monde Arduino



Stuart Cording (iSystem, Allemagne)

Les cartes telles que les Arduino M0 Pro et DUE possèdent le même brochage que leurs cousines à 8 bits, respectivement Uno et Mega, mais offrent significativement plus de SRAM et de mémoire flash ainsi qu'une horloge plus rapide. Un inconvénient peut-être, tout du moins un défi, est la tension limitée à 3,3 V sur les broches de ces cartes. Ces avantages, combinés à la prise en charge de beaucoup des *shields* existants et aux nombreuses bibliothèques disponibles, rendent toutefois la transition simple et rapide.

Le plus gros problème pour les amateurs endurcis ou les professionnels qui utilisent Arduino pour le prototypage rapide reste l'environnement de développement intégré (EDI) limité. Lorsque l'on commence avec Arduino, son EDI est parfait : simple et clair. Mais, quand les croquis se compliquent et que les défauts ne peuvent plus être corrigés seulement en faisant clignoter une LED ou en envoyant en message sur le port série, le besoin d'un EDI professionnel se fait cruellement sentir.

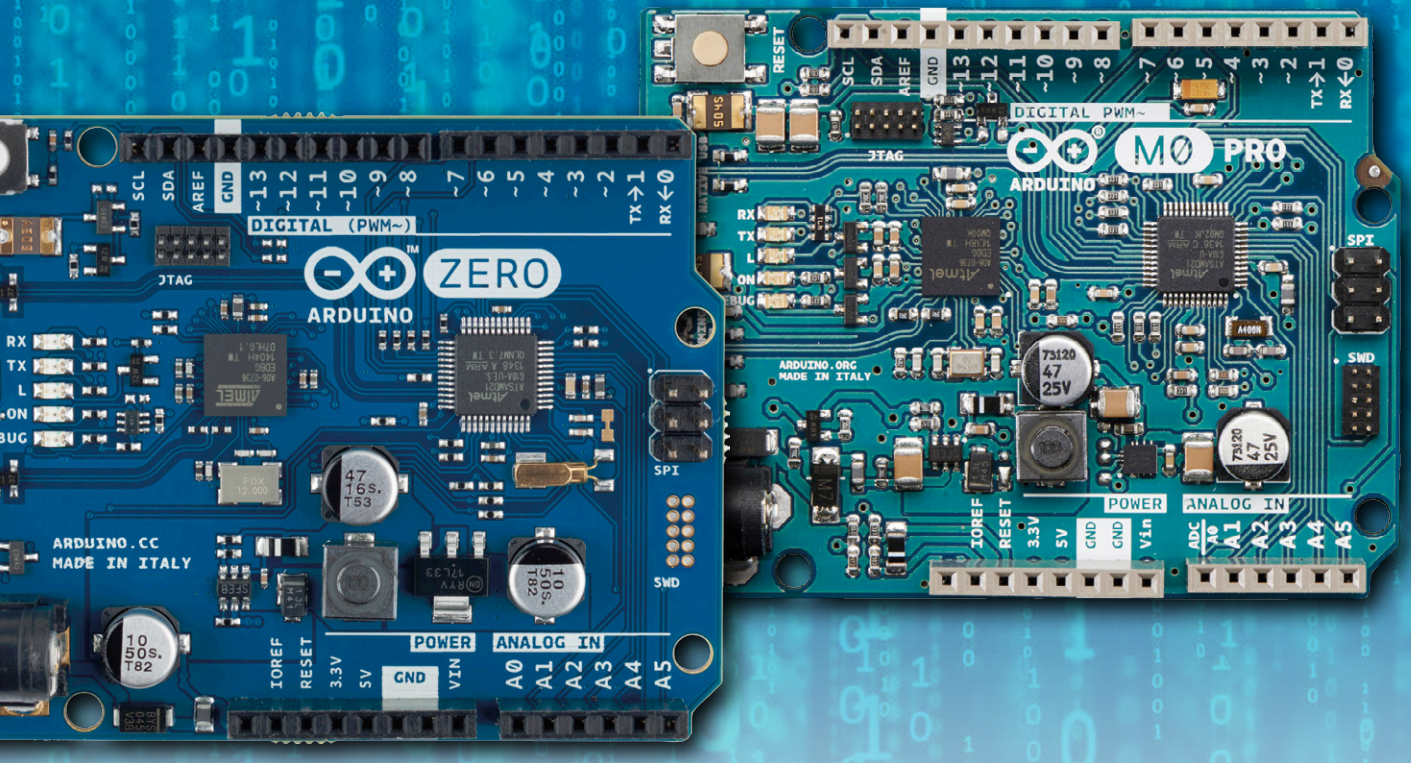
Déboguer comme un pro, sans rien débourser

Les cartes Arduino M0 Pro (d'Arduino.org) ainsi que Arduino/Genuino Zero (d'Arduino.cc), basées sur le SAM D21G d'Atmel (maintenant Microchip), sont équipées d'une interface de programmation un peu différente. La puce derrière le connecteur de « programmation USB » est aussi un débogueur embarqué (EDBG) qui peut être utilisé par beaucoup d'environnements de développement, non seulement pour programmer la carte, mais aussi pour lire l'état interne du micro au cœur du système. Avec un EDI adéquat, il est possible de déboguer votre croquis, mais également la totalité du code du cœur Arduino et des bibliothèques.

L'un des EDI capables de cela est *winIDEA Open* d'iSystem, la version gratuite de leur EDI pour µC à base d'ARM Cortex-M. Pour un outil professionnel, il est plutôt facile à prendre en main puisqu'il suffit de disposer du fichier produit par l'EDI Arduino pour un croquis (voir plus bas). Comme nous le découvrirons, le gestionnaire de constructions peut aussi être invoqué pour accélérer la compilation de croquis. En plus de cela, l'environnement de test intégré, *testIDEA*, peut servir à détecter les bogues créés durant le développement.

La configuration d'un tel outil comporte bien des facettes. Afin

Avec l'introduction des microcontrôleurs à 32 bits ARM Cortex-M dans la famille Arduino, les faiseurs auront un outil puissant dans les mains. Pour entrer dans le monde d'Arduino, la simplicité et la clarté de l'éditeur de code Arduino sont parfaites. Mais, quand les croquis se compliquent, difficile de ne pas se laisser séduire par la puissance d'un environnement de développement professionnel.



de fournir à la fois un aperçu pour ceux simplement intéressés par le concept ainsi que des instructions détaillées pour ceux qui souhaiteraient passer à la pratique, l'aperçu de cet article est complété par une série de didacticiels disponibles en ligne [1].

Premier projet

Pour commencer un projet avec un µC SAM D21 dans d'autres EDI, par ex. Atmel Studio qu'Elektor a déjà couvert [2][3], il faut impérativement partir d'un exemple et inclure tout un tas de fichiers source prédéfinis, issus des bibliothèques. L'EDI winIDEA Open est différent : le fichier en sortie du compilateur, un fichier ELF avec le binaire pour votre croquis Arduino par ex., est considéré comme le fichier central de votre projet. Avec ce fichier seulement, l'EDI peut trouver tous vos fichiers source ainsi que les fichiers du cœur Arduino et les bibliothèques nécessaires au débogage de votre code. La méthode la plus simple pour déboguer un croquis Arduino est donc de le construire dans l'EDI Arduino puis d'importer et charger le fichier ELF résultant sur l'Arduino M0 Pro à l'aide de winIDEA Open.

Dans les trois premiers projets du *Tutorial 1* [1], le traditionnel croquis « Blink » est créé dans l'EDI Arduino standard. À cause des quelques différences entre l'Arduino M0 Pro et Arduino/Genuino Zero, winIDEA Open sera configuré un peu différemment. Un espace de travail préconfiguré est fourni pour ceux qui veulent démarrer rapidement : il suffit d'ouvrir l'espace de travail approprié et de charger le fichier ELF créé par l'EDI Arduino.

Une fois le code dans le µC, il est possible de commencer à explorer le fonctionnement du code du cœur Arduino. Dans la fenêtre *Project Workspace* (**fig. 1**), tous les fichiers qui font partie du projet, ainsi que toutes les fonctions utilisées, sont listés dans une arborescence, comme dans l'explorateur de fichiers de Windows. Si vous vous êtes déjà demandé comment la fonction `delay()` est écrite, il suffit d'ouvrir le dossier *Functions*, de faire défiler la liste jusqu'à trouver `delay(unsigned long ms)` et de double-cliquer sur la fonction pour en afficher le code. La fonction sera affichée dans la fenêtre d'édition. Si vous voulez voir quand la fonction est appelée, il suffit d'ajouter un point d'arrêt sur une ligne de code dans la partie grise à gauche du numéro de ligne. Cliquez avec le bouton droit de la souris et sélectionnez *Set Breakpoint*. Une fois le code redémarré, le µC s'arrêtera lorsqu'il atteindra le point d'arrêt. Vous pourrez alors analyser le contenu des variables, registres ou de la mémoire. Vous remarquerez que, malgré l'apparente simplicité de l'envi-

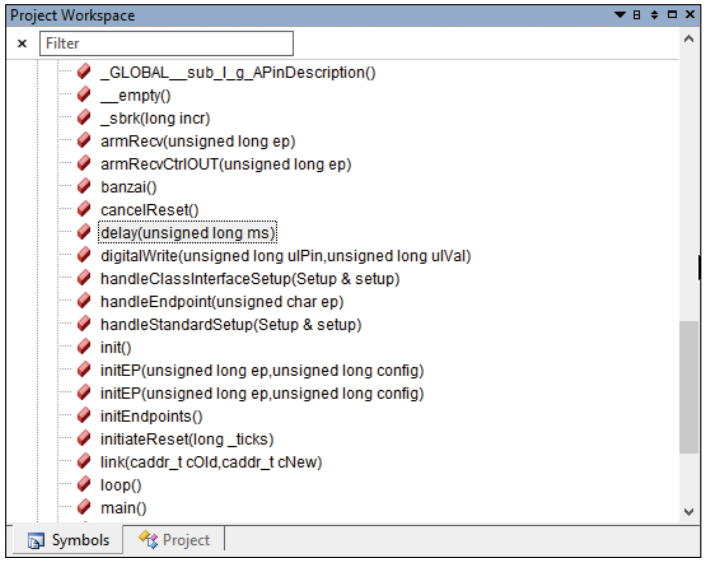


Figure 1. La vue *Functions* montre toutes les fonctions utilisées par le croquis, même celles appartenant au cœur Arduino ou aux bibliothèques.

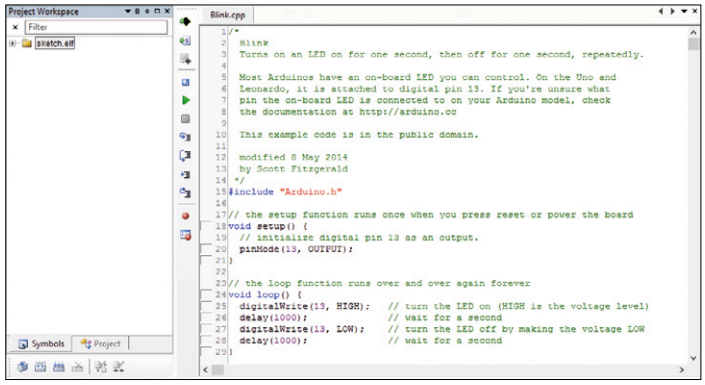


Figure 2. *Blink.cpp* dans l'EDI winIDEA Open.

Figure 3. Le croquis *Blink* dans l'EDI Arduino.

ronnement Arduino, le code du cœur Arduino utilise beaucoup des tours de passe-passe sophistiqués que le langage C permet. Certains symboles (noms des variables et fonctions) peuvent par conséquent être listés dans la fenêtre *Project Workspace* sans qu'ils ne soient associés à une ligne importante dans le code.

Travailler plus vite

Un des problèmes avec la méthode décrite jusqu'ici est qu'il n'est pas possible de déboguer correctement le `Blink.ino` original. Le problème est dû à la manière dont l'EDI Arduino compile le croquis. Pour les débutants, ce n'est pas de problème, mais ceux qui souhaitent explorer les profondeurs du code Arduino apprécieraient de pouvoir construire un croquis en dehors de l'EDI Arduino.

Cela a aussi un autre avantage : l'EDI Arduino reconstruit le projet en entier à chaque fois que l'on change une ligne. Encore une fois, cela importe peu pour les petits croquis. Par contre, dès que l'on inclut une bibliothèque ou deux et que l'on a plusieurs fichiers source, la construction commence à prendre du temps. Ici encore, un EDI professionnel peut aider en permettant à la construction du projet à l'extérieur de l'EDI. Afin de rendre le processus de construction plus intuitif et rapide, le *Tutorial 2* [1] utilise un *Makefile* pour construire notre croquis. Cela nécessite quelques changements à la création de votre croquis :

Votre croquis devra se trouver dans un fichier nommé `<NOM DU CROQUIS>.cpp`, stocké dans un dossier nommé « src ».

Votre croquis devra comporter la ligne de code `#include "Arduino.h"` (voir **fig. 2**) au début du fichier (avant l'appel de `setup()`).

En plus de cela, il vous faudra l'utilitaire *make* qui peut être installé avec MinGW. Vous trouverez des instructions détaillées dans le *Tutorial 2* (voir [1]).

Au lieu de construire le croquis dans l'EDI Arduino (**fig. 3**) et de programmer et déboguer le code dans l'EDI winIDEA Open, il est possible de tout faire depuis winIDEA Open.

Passons au débogage de notre croquis. L'espace de travail du projet fonctionne un peu comme l'explorateur de fichiers de Windows : vous pouvez dérouler les éléments listés comme vous le feriez avec des dossiers. Dans le *Project Workspace*, utilisez le symbole « + » à gauche de chaque élément dans l'ordre suivant : sketch.elf → Modules → src → Blink.cpp (**fig. 4**). Enfin double-cliquez sur `loop()` ou `setup()` et l'éditeur ouvrira le code source, à la ligne où la fonction se trouve. À la gauche de chaque ligne de code, vous trouverez un carré gris permettant d'ajouter un point d'arrêt dans l'exécution du code si vous le souhaitez. Il suffit pour cela de cliquer avec le bouton droit puis de sélectionner *Set Breakpoint*.

Trucs et astuces

Parlons maintenant des limites des fonctions de débogage du SAM D21. Au total, il n'est possible de positionner que trois points d'arrêts simultanément. Si vous essayez d'en ajouter plus, une fenêtre vous annoncera que tous les points d'arrêts disponibles ont été utilisés (**fig. 5**). La solution la plus facile est de désactiver l'un des points d'arrêt actifs (cliquez avec le bouton droit sur une ligne avec un point d'arrêt puis sélectionnez *Disable Breakpoint*) avant d'en ajouter un nouveau. Il est également possible de supprimer le point d'arrêt (*Clear Breakpoint*), mais désactiver a l'avantage de laisser un mar-

queur rouge à gauche de la ligne ; pratique pour retrouver l'endroit où se trouvait le point d'arrêt. C'est très utile lorsque vous vous promenez un peu partout dans le code.
Si jamais vous vous intéressez à l'efficacité du code produit par la chaîne de compilation Arduino (à savoir GCC), affichez le code assembleur avec le menu *View → Disassembly*. Si vous cliquez dans cette fenêtre (**fig. 6**), chaque pas du débogueur dans le code sera d'une instruction au lieu d'une ligne de code lorsque la fenêtre d'édition est active. Vous pourrez également voir comment les registres du µC sont utilisés lors des appels de fonctions pour passer les paramètres, entre autres choses.

Au-delà du débogage

Les applications sont typiquement divisées en fonctions. Pour les petits projets, avec un seul développeur, il est relativement facile de garder un œil sur ce qui fonctionne, ce qui n'est pas fini, et là où sont les bogues. Par contre, pour les plus gros projets, avec plusieurs programmeurs, il est crucial de disposer d'une batterie de tests capables de prouver que le code fonctionne comme prévu. À cet effet, winIDEA fournit l'outil de test *Original Binary Code* (OBC) : testIDEA. Nous l'utiliserons ici pour créer quelques tests pour une fonction qui évalue une valeur d'entrée imaginaire et retourne une nouvelle valeur à passer à `delay()`.
L'algorithme implanté dans le code est plutôt simple (**fig. 7**). Si la valeur passée en paramètre est inférieure à 50, il retourne 150. Si la valeur se trouve entre 50 et 99, il retourne 1000. Pour toutes les autres valeurs, il retourne 1750. Une fois le code écrit, il est logique d'écrire un test qui permet de s'assurer que le code fonctionne toujours après que votre collègue l'aura modifié (c'est toujours le collègue qui introduit les bogues).
Avant d'écrire des tests pour une fonction, il vaut mieux commencer par réfléchir à la manière de la tester en écrivant les étapes à suivre sur un bout de papier ou en créant un tableau avec les entrées et sorties attendues. Dans le **tableau 1**, nous nous sommes penchés sur les valeurs aux limites de la plage d'entrée et les valeurs proches des transitions définies dans l'algorithme (50 et 100).
Pour créer les tests, sélectionnez, depuis la barre de menu, Test → Launch testIDEA. Au démarrage, testIDEA demandera un *fichier de spécification de test*. Ce fichier contiendra les tests et aura l'extension *iyaml*. Voici comment définir un test :

- Depuis le menu, sélectionnez Test → New Test…
- Il faut maintenant définir la fonction à tester. Avant cela, il faudra rafraîchir le lien avec notre projet winIDEA en cliquant sur le symbole correspondant. Ensuite, vous sélectionnerez la fonction à tester, `evaluateNumber()`, depuis la liste déroulante.
- Saisissez la valeur à passer à la fonction dans la case *Parameters*. Pour le test 1, c'est 0.
- Saisissez ensuite le résultat attendu, 50, et choisissez *Default expression*.
- Cliquez sur OK ; le premier test est prêt.

Ce qui est unique avec les tests OBC, c'est qu'ils sont exécutés sur le µC lui-même et pas dans un simulateur, comme avec certains autres outils. Lorsque le test est lancé, son code est chargé dans la mémoire flash du µC et exécuté. Cela se fait avec la commande Test → Run All Tests. Avec un peu de chance, le résultat des tests sera correct, c'est indiqué par une case à cocher verte à côté du panneau *Outline*.
Afin de pouvoir distinguer les tests les uns des autres, on peut leur associer des métadonnées. Depuis le panneau *Form*, sélectionnez *Meta* et ajoutez un *Test ID,* par ex. « Test_1 ». Maintenant ajoutez les autres tests à votre plan de test. Ensuite vous pourrez exécuter tous les tests et, si tout se passe bien,

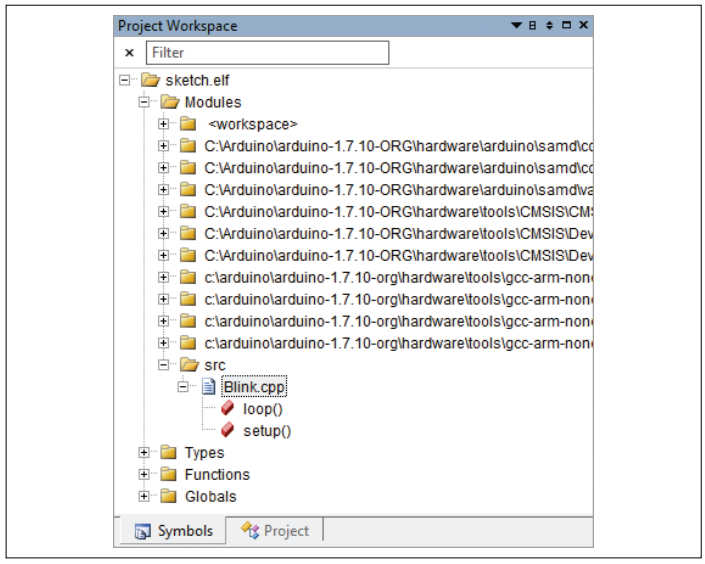


Figure 4. Vous pouvez naviguer dans l'exécutable du croquis comme dans une arborescence de fichiers.

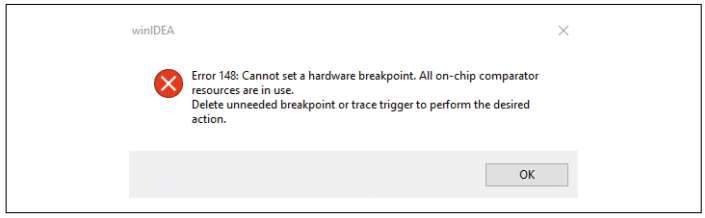


Figure 5. Vous utilisez trop de points d'arrêt.

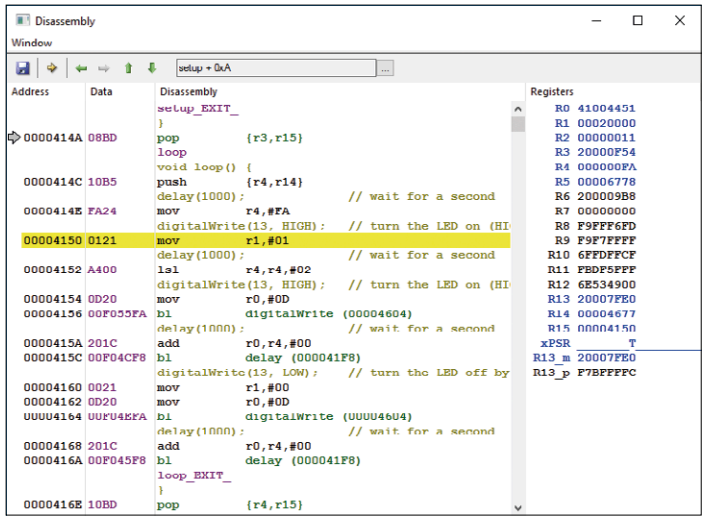


Figure 6. L'assembleur pour le croquis *Blink*.

Table 1. Valeurs d'entrée pour tester notre algorithme, avec la sortie attendue.

n° du test	valeur d'entrée	réponse attendue
1	0	150
2	1	150
3	48	150
4	49	150
5	50	1000
6	51	1000
7	98	1000
8	99	1000
9	100	1750
10	101	1750
11	150	1750
12	200	1750
13	255	1750

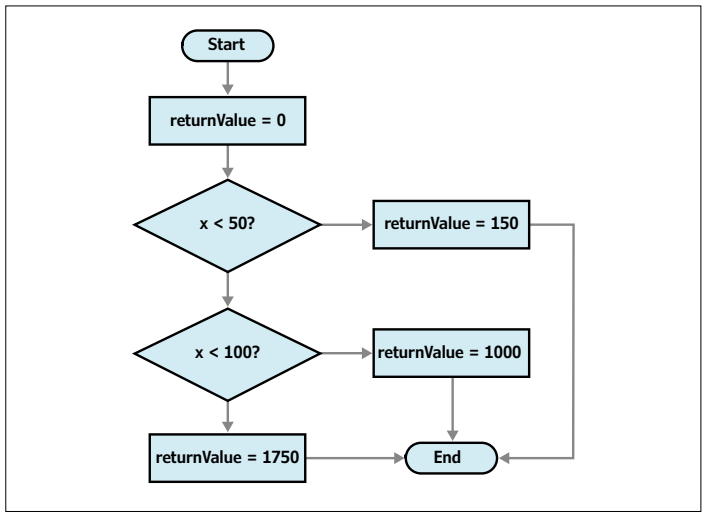


Figure 7. Organigramme d'un algorithme simple.

obtenir un résultat positif à chaque fois.
Bien entendu, l'intérêt d'un tel outil est de découvrir les bogues qui se glissent dans le code lorsque l'on ne fait pas attention. Imaginons par ex. qu'un des membres de l'équipe n'ait pas compris la spécification et décide que toutes les comparaisons « inférieur à » soient remplacées par « inférieur ou égal à » dans votre code. Vous ne vous en rendrez compte que lorsque le projet cessera de fonctionner comme prévu. Pour simuler cette erreur, remplacez `<` par `<=` dans les lignes 20 et 22 du fichier `Blink.cpp` (Tutorial 3, Project 5 [1]).
Pour voir où le bogue aurait pu se glisser, il suffit de reconstruire l'application dans winIDEA, d'ouvrir la spécification de test dans testIDEA et de lancer tous les tests. Les tests 5 et 9 devraient maintenant échouer et vous devriez être en mesure d'utiliser ce résultat pour rapidement trouver le problème.

En résumé

Les cartes Arduino à base de Cortex-M ont un énorme potentiel, particulièrement pour ce qui est du prototypage rapide. STMicroelectronics a récemment annoncé une carte à base de Cortex-M4 STM32 : la STAR Otto. Ces cartes vont probablement être utilisées dans des prototypes complexes avec *shields* Ethernet et Wi-Fi. Un environnement de débogage professionnel tel que winIDEA Open va certainement soulager plus d'un programmeur en quête du problème qui empêche son croquis de fonctionner. ⏮

(160228 – version française : Kévin Petit)

Liens

[1] www.elektormagazine.fr/160228
[2] www.elektormagazine.fr/130392
[2] www.elektormagazine.fr/140037

Le SAMD21 n'est qu'un exemple dans la gamme SAM D des µC Atmel à 32 bits à base d'ARM Cortex-M0+. Ces puces offrent jusqu'à 256 Ko de mémoire flash et 32 Ko de SRAM. En outre elles comportent une vaste gamme d'interfaces, dont USB *full speed*, horloge en temps réel, USART, interfaces I²C et SPI et CA/N à 12 bits. L'efficacité énergétique est également au rendez-vous avec une consommation de 70 µA/MHz. Il existe aussi dans la famille une version pour le marché automobile et certaines des puces peuvent être utilisées pour implanter une interface tactile capacitive avec des algorithmes d'Atmel pour créer boutons, potentiomètres linéaires et roues tactiles.

www.atmel.com/products/microcontrollers/arm/sam-d.aspx

webradio à tubes fluorescents (2)

RPi + ATmega + logiciel

Michael Busser (Allemagne)

Nous voulons une radio de cuisine à VFD, solide et fiable, mais branchée sur l'internet. Le premier article a traité la fonction, l'électronique et la commande de l'affichage. Mais sans logiciel, nous ne voyons encore rien. L'ATmega a besoin de micrologiciel pour l'affichage, mais il faut aussi du code au Raspberry Pi (RPi) pour en faire une webradio opérationnelle.

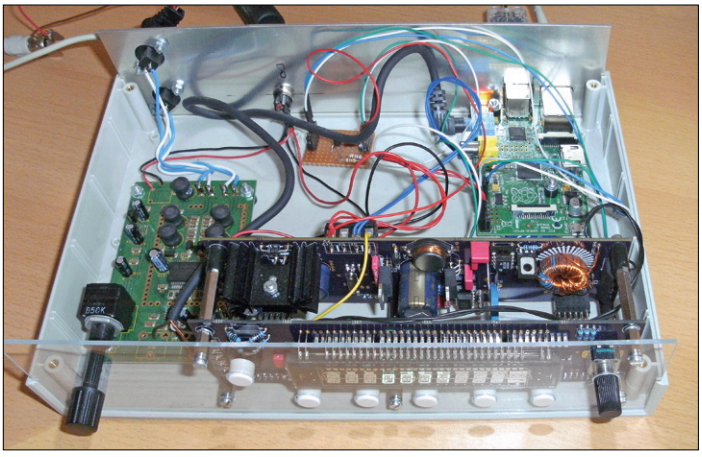


Figure 1. Toutes les cartes sont logées dans un boîtier moulé et y sont câblées.

Figure 2. Couvercle dessus et micrologiciel dedans, la webradio de cuisine à VFD opérationnelle, l'électronique toujours visible derrière la façade en acrylique.

Avec les schémas, les dessins des circuits imprimés et toutes les explications du premier article [1], vous avez construit un beau module VFD compact, tout y est convenablement installé, testé et les tensions sont conformes, pourtant, il ne fait rien d'utile.

Même si toutes les cartes sont câblées (**figure 1**), mises en boîte et le couvercle fermé (**figure 2**), il n'y a toujours rien à voir sur les tubes. C'est que le microcontrôleur de l'unité d'affichage a besoin d'un micrologiciel qui comprenne les commandes. C'est précisément l'objet de ce second article, qui apporte aussi le code pour le RPi.

Protocole de communication

Pour que deux systèmes informatiques puissent échanger des données, il faut s'être mis d'accord au préalable sur la manière de procéder : c'est le rôle du protocole. J'en ai faufilé un moi-même qui utilise un bus de domotique à deux fils. Il dispose de caractéristiques qui ne sont pas forcément nécessaires ici, mais il repose sur la conception habituelle de ces protocoles. Le micrologiciel pour l'ATmega du module d'affichage a été rédigé en entier dans l'environnement de développement Studio d'Atmel (cf. l'écran de la **figure 3**).
Le **tableau 1** montre la structure des données. La couche de transport prend la forme de l'envoi d'un message (type

Tableau 1. Trame du protocole

STX *début de transmission*	contenu du message	ETX *fin de transmission*
0x02	data	0x03

Tableau 2. Éléments du message

STX	SRC	DST	MT	MODE	charge utile (0 à 15 octets)	CS	ETX

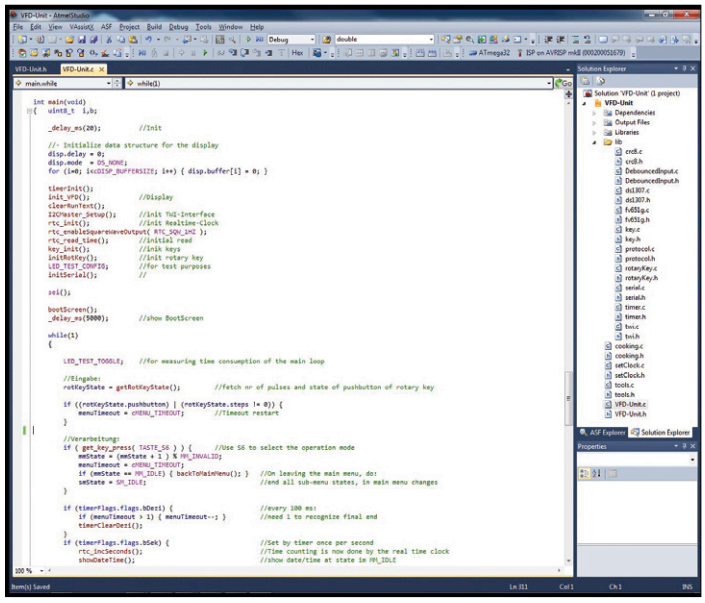


Figure 3. Fenêtre de l'environnement de développement Studio d'Atmel avec le micrologiciel pour l'ATmega du module d'affichage.

TDataBuf), avec comme préambule un STX. On marque la fin d'une transmission d'un ETX. On ne peut donc pas utiliser les signaux STX et ETX dans le reste du flux de données. Si vous le faites, masquez-les en les faisant précéder du caractère DLE (= 01xF). Le récepteur élimine alors le signe de camouflage des données et poursuit avec les caractères suivants du message reçu. Si c'est le caractère de masquage que vous devez transmettre, il faut envoyer deux signes DLE.

Tableau 3. Codes du message

champ	signification
SRC	Source *(Quelladresse)* : 0 à 255 sauf valeurs 2 et 3
DST	Destination *(Zieladresse)* : 0 à 255 sauf valeurs 2 et 3
MT	Type du message : 0x10 StatusReq 0x11 StatusResp 0x12 DisplayReq 0x13 DisplayResp 0x14 Key_Req 0x15 Key_Resp 0x16 Time_Req 0x17 Time_Resp Note : 0x02 et 0x03 sont interdits. Reste libre pour des extensions.
MODE	Bit signification 0 à 3 : longueur de la charge utile en octets : 0 à 15 4 : 0/1 = avec / sans somme de contrôle 5 : 0/1 = confirmation par ACK nécessaire / ou pas 6 : 0 = réservé 7 : toujours 1 Note : 0x02 et 0x03 sont interdits. De là, bit 7 toujours 1.
CS	Somme de contrôle du message

L'interface est formée des fonctions suivantes (cf. serial.c et serial.h) :

```
void initSerial(void);
void sendMsg(TDataBuf msg);
uint8_t peekMsg(TDataBuf *msg);
```

La fonction initSerial() doit être exécutée une fois au lancement du programme. Elle assure l'initialisation de l'UART et déverrouille différentes interruptions. On peut fixer le débit binaire en plaçant dans serial.h la mention :

```
#define UART_BAUD_RATE 9600UL
```

Les autres paramètres de l'interface sont 8N1. La fonction sendMsg() s'occupe des messages de type TDataBuf, les inscrit dans le tampon interne d'émission (quand il est libre) et retourne tout de suite à l'appelant. La réception et l'émission des messages sont commandées par interruption et se passent en arrière-plan. Dès réception d'un message valide, il est placé dans l'un des deux tampons de réception. Un appel de la fonction peekMsg() renvoie un 1 et le message valide est disponible dans msg.
Le mécanisme de masquage est déjà à l'ouvrage ici. Il n'y a qu'un tampon d'émission, mais deux de réception, de manière à ce qu'un message puisse être traité pendant la réception du suivant.
Le module de code est indépendant de la forme du message à transmettre qui est défini dans protocol.c et protocol.h. Un bus à deux fils comme le RS485 nécessite encore un signal de commande pour commuter entre l'émission et la réception. L'état normal est la réception. L'émetteur n'est actif que pendant la durée de la transmission. Il faut dans serial.h les deux définitions suivantes :

```
#define SENDER_OFF
#define SENDER_ON
```

Comme le récepteur est toujours en service, il reçoit aussi le message envoyé par le poste lui-même. C'est utile pour détecter les collisions de bus. Dans protocol.h, on n'utilise que la fonction ownAdr() qui renvoie l'adresse propre du nœud sur le bus.

Le **tableau 2** détaille les différents éléments d'un message et le **tableau 3** contient leurs significations. On voit dans le **listage 1** l'agencement d'un message de type *struct* en C. Avec le regroupement de type union, la structure TDatagramm et la chaîne d'octets TBuf occupent la même adresse en mémoire. On peut ainsi considérer un message simplement comme une chaîne d'octets, ce qui simplifie l'émission et la réception.

Communication ATmega

Voici les différents types de messages :

```
#define MT_STATUS_REQ 0x10
#define MT_STATUS_RES MT_STATUS_REQ + 1

#define MT_DISPLAY_REQ 0x12
```

```
#define MT_DISPLAY_RESPMT_DISPLAY_
      REQ + 1

#define MT_KEY_REQ  0x14
#define MT_KEY_RESP MT_KEY_REQ + 1

#define MT_TIME_REQ 0x16
#define MT_TIME_RESPMT_TIME_REQ + 1
```

Avec `DISPLAY_REQ`, le RPi peut enfin afficher quelque chose sur le VFD. D'autre part, `KEY_REQ` informe le RPi qu'un bouton a été actionné sur le module d'affichage. Avec `TIME_REQ`, l'ATmega peut se procurer le temps réel par le RPi, lequel le reçoit par l'internet via NTP.
L'ATmega peut envoyer et recevoir des messages dans un format défini par l'interface sérielle d'un simple appel de la fonction ad hoc dans `protocol.c` (**listage 2**). La réception a lieu dans une boucle du programme principal (**listage 3**), qui vérifie constamment la validité des messages reçus et les traite, si nécessaire.

Voici un exemple de déroulement d'un message.
Toutes les 60 min, l'ATmega envoie au RPi un message du type `MT_TIME_REQ`, ce qui provoque l'appel de la fonction `sendMsgTimeReq()` :

```
if (timerFlags.flags.bMin) {
    doNTP_Sync--;
    if (doNTP_Sync == 0) {
        doNTP_Sync = NTP_SYNC;
        sendMsgTimeReq();
      // appel temps réel de RPi
    }
    timerClearMin();
}
```

Le RPi envoie en retour un message du type `MT_TIME_RESP` qui sera traité dans la boucle de réception du programme principal. Le texte source complet est trop long pour une reproduction ici. Vous pouvez l'obtenir dans le fichier *Software_2.zip* du téléchargement [1].

Logiciel RPi

On utilise comme système d'exploitation pour le RPi la version *wheezy-raspbian* du 09/09/2014. Comment en faire une image à copier sur carte SD avec la configuration de base, vous l'apprendrez en détail sur [2].
Il y a en [3] une notice explicative sur la manière de configurer RPi en webradio

Listage 1. Construction du message

```
typedef struct {       // construction d'un message
  uint8_t     src;     // adresse source (émetteur)
  uint8_t     dst;     // adresse de destination (récepteur)
  uint8_t     mt;      // type du message
  TMode       mode;    // octet de mode, décrit la structure du message
  TPayload    Payload; // dépend du niveau de l'application
  uint8_t     cs;      // somme de contrôle, par ex. CRC-8
} TDatagramm;

typedef union {
    TBuf        Buf;          // accès vi table d'octets
    TDatagramm  Datagramm;    // accès via structure
} TDataBuf;
```

Listage 2. Fonctions des messages

```
void sendMsgTimeReq(void) {
  TDataBuf msg;

  msgClear( &msg );
  msg.Datagramm.src = ownAdr();
  msg.Datagramm.dst = masterAdr();
  msg.Datagramm.mt  = MT_TIME_REQ;
  msg.Datagramm.mode.bAckReq = NO_ACK_REQUIRED;
  msg.Datagramm.mode.bCRC = NO_CHECKSUM; // WITH_CHECKSUM;
  msg.Datagramm.mode.count = 0; // ne pas compter octets de tête / fin
  msgPrepare( &msg );
  sendMsg( msg );
}
```

Listage 3. Analyse des messages

```
if (peekMsg(&recMsg) == 1) {// si message reçu : analyser
    switch (recMsg.Datagramm.mt) {
        case MT_STATUS_REQ:     break;
        case MT_STATUS_RESP:    break;
        case MT_DISPLAY_REQ: i=0;
            b=1;
            for (i=0; i < cDISP_BUFFERSIZE; i++) {
                if (recMsg.Datagramm.Payload[i] == 0) {b = 0;}
                if (b == 1) {
                    disp.buffer[i] = recMsg.Datagramm.Payload[i];
                } else {
                    disp.buffer[i] = 0;
                }
            }
            disp.delay           = cDISP_SHOWTEXT;
            disp.mode            = DS_TEXT;
            showText();
            break;
        case MT_DISPLAY_RESP:    break;
        case MT_KEY_REQ:         break;
        case MT_KEY_RESP:        break;
        case MT_TIME_REQ:        break;
        case MT_TIME_RESP:       setTimeOnMsgReq( &recMsg );    break;
        }
}
```

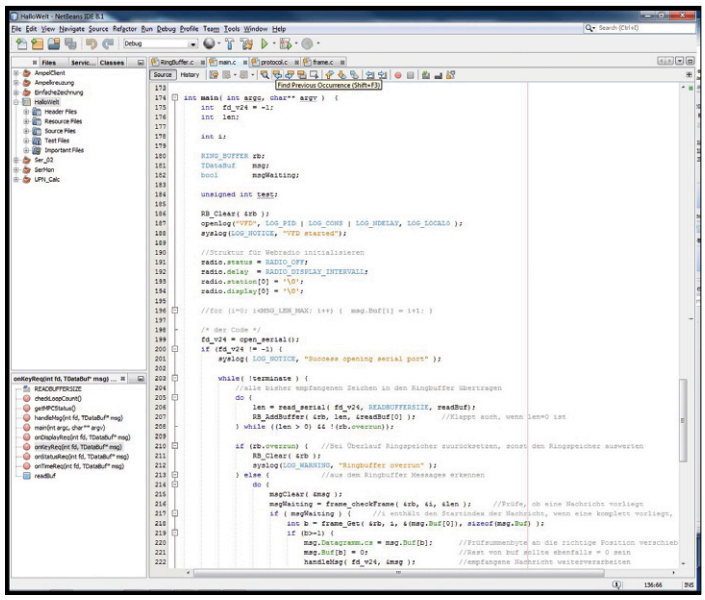


Figure 4. Fenêtre de l'EDI de NetBeans avec le micrologiciel qui transforme un Raspberry Pi en une webradio.

à l'aide de *mpd* (*Music Player Daemon*) et *mpc* (*Music Player Client*). Mais bien d'autres sources existent sur l'internet à ce sujet. Il suffit d'une recherche sur « RPI mpd » pour s'en convaincre.
Pour faire tourner un premier programme RPi en C, il vous faut un environnement de développement (en principe, la ligne de commande suffirait). J'ai choisi NetBeans parce que Java était déjà installé sur mon ordinateur. La **figure 4** vous montre de quoi il s'agit. L'installation de NetBeans comme EDI pour RPi est décrite pas à pas sur [4].

Une fois que vous en êtes arrivé là, vous pouvez confortablement créer sous Windows des programmes en C pour RPi. La compilation et le débogage se feront sur RPi.
Par bonheur, le code C pour l'ATmega tourne aussi sans trop de modifications sur RPi, pour autant qu'on n'ait pas besoin de fonction spéciale ou de module supplémentaire, il n'y en a pas là.

Le programme RPi envoie des messages de journalisation à un serveur Syslog, ce qui aide au débogage. Le Syslog doit être configuré sur le Rpi. La fonction `dumpMsg` réceptionne un message et l'écrit sous forme lisible sur le Syslog.

Dans la partie `main` du programme RPi, tous les caractères qui viennent d'arriver sont lus sur le port sériel et déposés dans un tampon circulaire :

```
do {
    len = read_serial(fd_v24, READBUFFERSIZE,
      readBuf);
    RB_AddBuffer(&rb, len, &readBuf[0]);
                // fonctionne même si len=0
} while ((len > 0) && !(rb.overrun));
```

La méthode `frame_checkFrame()` est appelée régulièrement pour vérifier si un message complet se trouve déjà dans le tampon. Sinon, elle retourne un 0 à la place d'un 1. Les paramètres `startindex` et `length` indiquent le début et la longueur du message.
Un appel de la fonction `frame_get()` renvoie alors le message en question dans `msg`. Il est ensuite traité par la fonction `handleMsg()` (**listage 4**). Le message d'exemple `MT_TIME_RESP` arrive ainsi à la fonction `handleMsg()` qui enchaîne un appel à `onTimeReq()` (**listage 5**). C'est là qu'on fait la réponse à la requête et on la renvoie directement à l'expéditeur par la fonction `frame_Send()`.
Le RPi s'occupe de la configuration du client NTP. La fonction `localtime()` à l'intérieur de la fonction `getTimeResp()` effectue la demande du temps système. On peut voir comment le message de réponse est composé dans la structure `TDataBuf`.

Listage 4. Fonction *handleMsg()*

```
void handleMsg(int fd, TDataBuf *msg) {
    if (msg->Datagramm.mode.bCRC) {// test de la somme de contrôle
        syslog( LOG_NOTICE, "checksum ignored");  // tbd
    }
    if (msg->Datagramm.dst == masterAdr()) {
        switch (msg->Datagramm.mt) {
        case MT_STATUS_REQ:     onStatusReq(fd, msg);     break;
        case MT_DISPLAY_REQ:    onDisplayReq(fd, msg);    break;
        case MT_KEY_REQ:        onKeyReq(fd, msg);        break;
        case MT_TIME_REQ:       onTimeReq(fd, msg);       break;
        default:    syslog(LOG_ERR, "unknown message dropped");
            dumpMsg(msg);
        }
    } else {// ce n'est pas mon message
        syslog( LOG_NOTICE, "message not for me, ignoring it");
    }
```

Listage 5. Fonction *onTimeReq()*

```
void onTimeReq(int fd, TDataBuf *msg) {
    TDataBuf reply;
    msgClear(&reply);
    reply.Datagramm.src = masterAdr();
    reply.Datagramm.dst = msg->Datagramm.src;
    getTimeResp(&reply);
    // dumpMsg(&reply);
    frame_Send(fd, &(reply.Buf[0]), reply.Datagramm.mode.count
      + MSG_HEADER_LEN, reply.Datagramm.cs );
    syslog( LOG_NOTICE, "onTimeReq");
}
```

Webradio

On réalise la webradio proprement dite avec le *mpc* de Music Player Client. C'est là que l'on crée des listes de lecture des stations de radio par internet. Vous trouverez les détails sur [5]. Il y a une aide en ligne *mpc* disponible par invite de commande. Mon programme, écrit en C, commande le mpc en l'appelant par l'option correspondante de `system()` comme on pourrait le faire avec l'invite de commande.

Un fragment de la fonction `onKeyReq()` dans le **listage 6** vous éclairera davantage. On l'appelle en appuyant sur un bouton du module d'affichage.

Si vous appelez mpc sans paramètre, vous verrez dans la première ligne la station et le titre qui passe en ce moment. La fonction `getMPCStatus()` le fait à intervalles réguliers et envoie ensuite le nom de la station par `MT_DISPLAY_REQ` à l'ATmega qui affiche alors le nom de la station de radio choisie à la place de la date du jour.
Avec mpd, les flux radio doivent toujours être enregistrés dans une liste de lecture. On la trouve dans le répertoire */var/lib/mpd/playlists*.

Listage 6. Fragment de la fonction *onKeyReq()*

```
void onKeyReq(int fd, TDataBuf *msg) {
    uint8_t keys = msg->Datagramm.Payload[0];
      // définition variable / vérification bouton
    if (keys & KEY_S0) {syslog( LOG_NOTICE, "onKeyReq:  S0");}
    if (keys & KEY_S1) {terminate = true; syslog( LOG_NOTICE, "onKeyReq: S1");}
    if (keys & KEY_S2) {syslog( LOG_NOTICE, "onKeyReq: S2");}
    if (keys & KEY_S3) {syslog( LOG_NOTICE, "onKeyReq: S3");}
    if (keys & KEY_S4) { // station suivante
        if (radio.status == RADIO_ON) {
            system("mpc next");}
            syslog( LOG_NOTICE, "onKeyReq: S4");
        }
    if (keys & KEY_S5) {                    // on / off radio
        if (radio.status == RADIO_OFF) {    // si radio off
            system("mpc play");             // jouer dernière station
            radio.status = RADIO_ON;
        } else                              // radio off
            system("mpc stop");
            radio.status = RADIO_OFF;
        }
        syslog(LOG_NOTICE, "onKeyReq:  S5");
    }
}
```

Liens

[1] Logiciel et 1ère partie : www.elektormagazine.fr/150720
[2] Installation du RPi : www.netzmafia.de/skripten/hardware/RasPi/RasPi_Install.html
[3] RPi en Webradio : www.youtube.com/watch?v=jf3M1RVpQbM
[4] *Netbeans* pour RPi : http://bit.ly/2aBZ14A
[5] Tutoriel webradio : www.youtube.com/watch?v=jf3M1RVpQbM
[6] Lancement automatique : www.netzmafia.de/skripten/hardware/RasPi/RasPi_Auto.html

Lancement du programme

Actuellement, il faut encore lancer le programme du RPi manuellement parce que son développement n'est pas terminé. La longueur des répertoires créés lors de l'utilisation de NetBeans est assez embarrassante. Si vous suivez la notice pour configurer NetBeans (dans votre répertoire racine), le programme exécutable se retrouve dans le répertoire suivant :

```
/root/.netbeans/remote/192.168.1.24/mib2-Windows-
      x86_64/D/Projekte/RaspberryPi/RPi/HalloWelt/
      dist/Debug/GNU-Linux
```

La partie du chemin en caractères gras change en fonction de l'environnement : adresse IP du PC sous Windows, « mib2 » (nom d'hôte du PC) et « D\Projekte\... » (chemin du projet sous Windows).

Mais on peut facilement faire en sorte que le RPi lance automatiquement le programme. Comment ? C'est Netzmafia [6] qui l'explique. Lors d'un démarrage à partir du shell, le programme adhère à ce shell. Quand on l'arrête, le programme s'arrête aussi. Le remède consiste en un simple « & » après la commande :

```
./hallowelt &
```

Le programme est alors exécuté en tâche de fond. Le shell donne aussi un numéro de processus et un PID. Quand on veut le ramener à l'avant-plan, on utilise alors :

```
fg %1
```

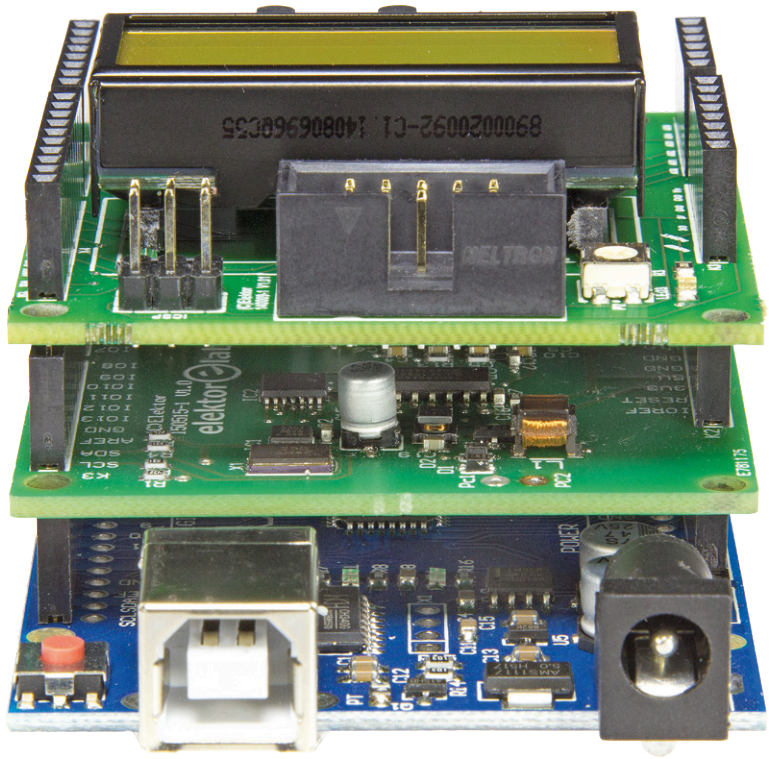
Le programme est alors capable de tourner sur le RPi. Des développements complémentaires sont possibles et bienvenus. On peut importer directement dans NetBeans le programme emballé dans *HalloWelt.zip*. Bonne chance pour vos expériences !

(160207 – version française : Robert Grignard)

SDR d'Elektor réinventé (4)

la radio logicielle en solo

Burkhard Kainka (Allemagne)



Et si l'on se passait du PC pour avoir un *shield* SDR qui ne dépende plus que d'Arduino ? C'est possible si l'on accepte quelques limitations. Avec le *schield* Elektor, une carte d'extension pour Arduino, on a déjà sous la main presque tout ce qu'il faut. Il ne reste plus qu'à s'en donner à cœur joie dans le traitement des signaux IQ.

Avec un strict minimum, il est même possible d'écouter directement le signal de l'hétérodyne transposé en fréquence vers le bas. Carrément branché sur la prise audio du *shield* SDR, un écouteur permet déjà d'entendre un son faible. Le récepteur travaille alors en mélangeur direct. Il faut juste lui ajouter une commande de réglage fin appropriée.

Sélection sur Arduino

Sur *mon premier shield :-)* [1] de juillet/août 2014, il y a deux boutons et un potentiomètre qui permettent de réaliser une commande adéquate. S1 abaisse la fréquence de réception, S2 la rehausse ; quant au potentiomètre, il règle la largeur du saut entre 20 Hz et 100 kHz. L'afficheur à cristaux liquides indique la fréquence actuelle dans la ligne du haut et, en dessous, le pas demandé (**figure 1**). De quoi syntoniser avec beaucoup plus de finesse qu'avec le pas actuel de 1 kHz.
Pareille commande facilite donc l'accord, mais aussi le déplacement rapide dans les bandes de fréquence en *CW* (morse p.ex.) comme en BLU (bande latérale unique ou *SSB*). Au démarrage, la fréquence est de 7 000 kHz, ce qui permet de scruter tout de suite la bande des 40 m pour trouver des stations CW ou SSB. Pour des raisons de compatibilité, le nouveau logiciel (gratuit sur [2]) pour la combinaison Arduino + *shield* SDR + *shield* 140089 [1] permet encore la sélection sérielle de fréquence. Rien n'empêche de continuer la syntonisation par PC, on verra alors la fréquence s'afficher en kHz (**figure 2**) sur le LCD. Toute action sur l'un des boutons du *shield* modifie l'affichage en hertz et montre la vraie fréquence, sans le décalage de 12 kHz (cf. **listage 1**). Bien sûr, avec un programme SDR, le PC peut toujours servir à commander le récepteur, en alternance avec le *shield*, qui sert alors de mélangeur direct.

Mélangeur direct

Le mélangeur direct transpose d'un coup le signal HF dans le domaine audio. Il est vrai que le *shield* est conçu pour une

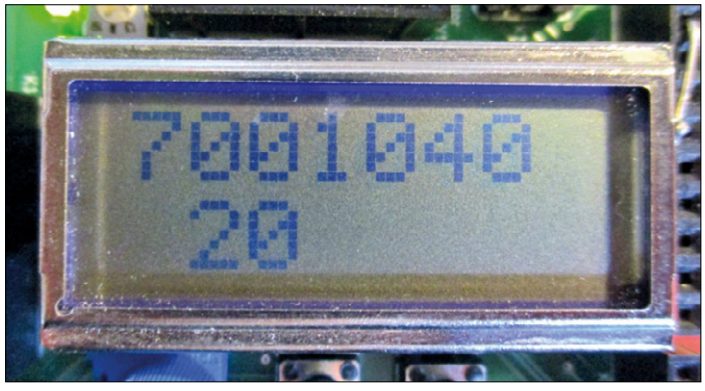


Figure 1. L'affichage sur mon premier *shield* Elektor : 7001,040 kHz, réglable à 20 Hz près.

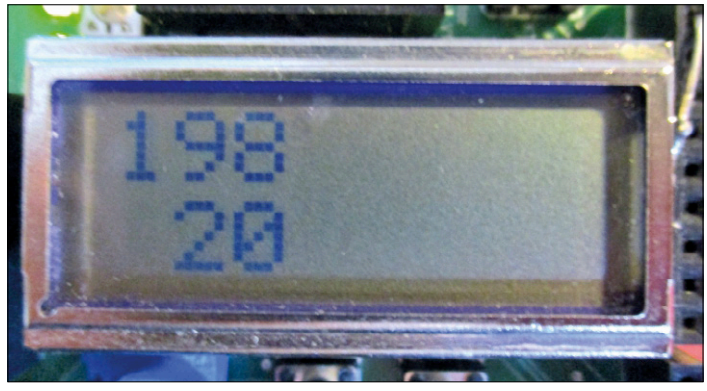


Figure 2. 198 kHz, c'est le PC qui l'a réglé.

Listage 1. Extraits du logiciel de la radio logicielle en solo

```
void loop(void)
{
   pot = analogRead(A3);
   fstep =1000000;
   if (pot > 100) fstep =1000000;
   if (pot > 200) fstep =500000;
   if (pot > 300) fstep =100000;
   if (pot > 400) fstep =20000;
   if (pot > 500) fstep =5000;
   if (pot > 600) fstep =1000;
   if (pot > 700) fstep =200;
   if (pot > 800) fstep =100;
   if (pot > 900) fstep =20;

   if (fstep != fstepOld){
   lcd.setCursor(1, 1);
    lcd.print(fstep);
    lcd.print ("   ");
   }
   fstepOld = fstep;

if (Serial.available()) {
 freq = Serial.parseInt();
 if (freq > 0){
  lcd.setCursor(0, 0);
  freqHz=freq*1000-12000;
  lcd.print(freq);
  lcd.print ("   ");
  setfreq (freqHz);
  }
 }
if (digitalRead(A0)==0){
 freqHz = freqHz - fstep;
 lcd.setCursor(0, 0);
   //freq=freqHz/1000;
   lcd.print(freqHz);
   setfreq (freqHz);
   lcd.print ("   ");
   delay (200);
   }
  if (digitalRead(A1)==0){
   freqHz = freqHz + fstep;
   lcd.setCursor(0, 0);
   //freq=freqHz/1000;
   lcd.print(freqHz);
   setfreq (freqHz);
   lcd.print ("   ");
   delay (200);
   }
}
```

fréquence intermédiaire (FI) de 12 kHz, mais les étages FI ont une telle largeur de bande que l'on peut aussi les utiliser dans une gamme jusqu'à 3 kHz en dessous, il faut seulement un peu plus de gain.

Un petit amplificateur stéréo à réglage de volume procure un meilleur confort, mais offre aussi le luxe de traiter les deux signaux I et Q, l'un en phase, l'autre en quadrature, donc déphasé de 90°. Avec un peu d'habitude, le cerveau arrive à en faire un mélange. Une telle différence de phase est déterminante aussi pour une audition spatiale. Si vous écoutez depuis un bout de temps de nombreux signaux CW, vous développez une impression spatiale qui favorise nettement la sélectivité de l'ouïe. De nombreux utilisateurs disent qu'ils arrivent à distinguer grâce à cela les signaux sous la fréquence de l'oscillateur local (VFO) de ceux au-dessus, ce qui élimine l'inconvénient majeur du mélange direct, il ne faut plus atténuer les fréquences images.

Le *shield* SDR en devient un vrai mélangeur direct de luxe, avec VFO stable sur toute la gamme jusqu'à 30 MHz, réglage fin et en plus sortie stéréo IQ. Et le nec plus ultra, le PC reste éteint et ne cause plus de parasites. Un récepteur sans commande automatique de niveau (ALC) est très pratique en CW et SSB quand, lors des pauses, le bruit ne peut pas remonter trop fort. C'est pourquoi il convient souvent de régler le volume dès qu'on arrive sur une station très puissante.

Le petit amplificateur pour casque d'écoute est construit sur un double amplificateur opérationnel LM358. Une puce spécialisée dans l'audio aurait nécessité moins de composants auxiliaires, mais on a toujours bien un ampli op dans un tiroir. Le LM358 a tendance à la distorsion de croisement quand il est soumis à une charge à basse impédance et doit fournir un gain élevé. On réduit l'inconvénient en mettant une résistance de 1 kΩ entre la sortie et la tension d'alimentation, ce qui n'active plus que la partie inférieure de la sortie *push-pull* à faible modulation. En série avec le casque, il y a encore une résistance de 100 Ω qui soulage la sortie de l'ampli op, mais surtout exclut les dommages à l'ouïe. L'amplificateur arrive alors directement

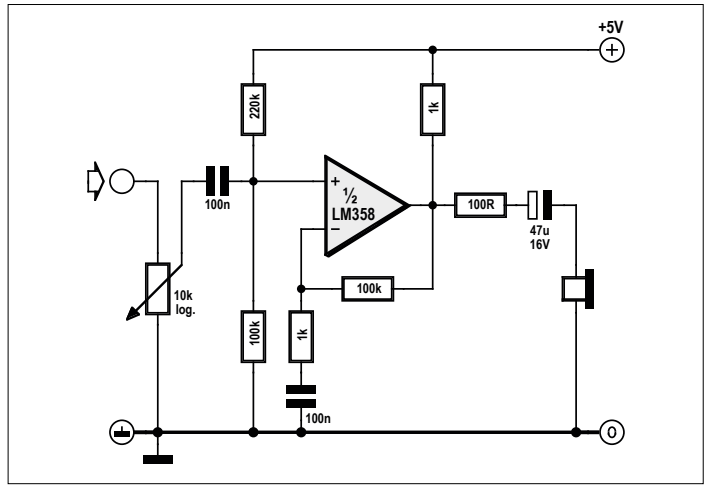


Figure 3. Un canal de l'amplificateur pour casque d'écoute.

à la saturation et la puissance dans le casque reste dans les limites. La sortie du LM358 peut descendre presque jusqu'à la masse, mais n'atteint pas la tension d'alimentation. C'est

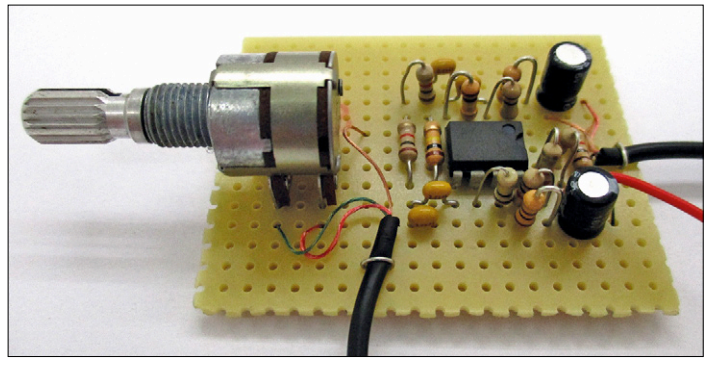


Figure 4. L'amplificateur expérimental pour casque sur une plaque perforée.

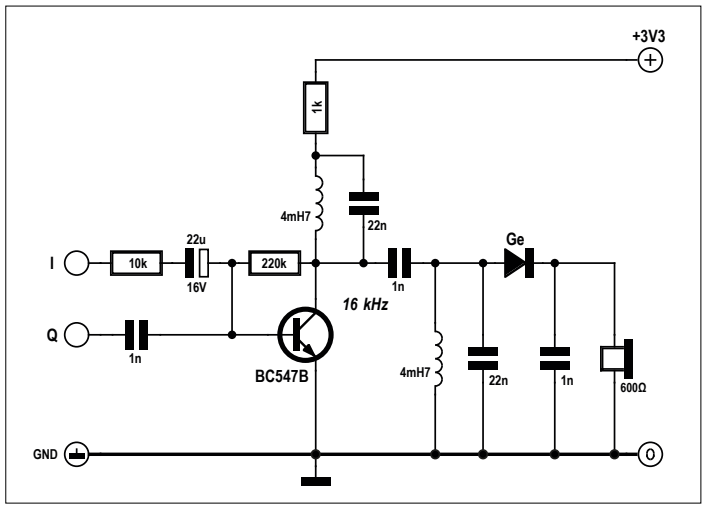


Figure 5. Un détecteur à diode.

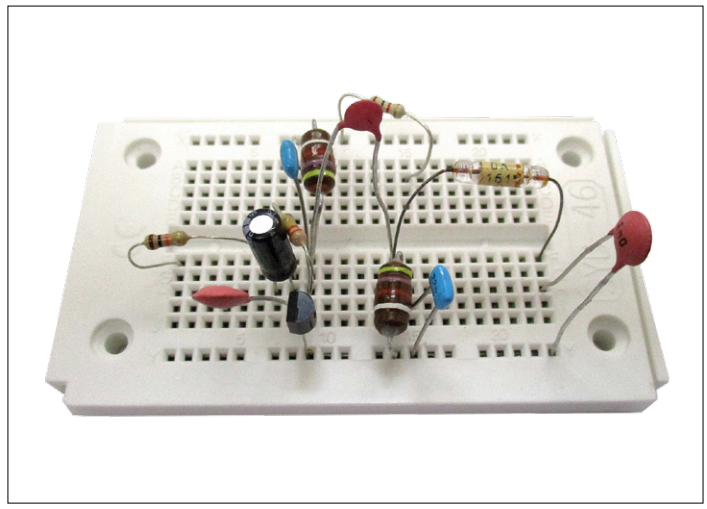


Figure 6. Réalisation du détecteur à diode sur une plaque d'expérimentation.

pourquoi on fixe la tension de repos au tiers environ de celle d'alimentation.
L'amplificateur est capable de remonter le signal de 40 dB, plus qu'il n'en faut pour la plupart des stations, sur lesquelles on doit considérablement réduire le niveau. Tout bénéfice pour la réception des stations CW. On reçoit aussi très bien la SSB pour autant que le réglage de fréquence soit exact. On peut alors l'écouter des heures durant.
Mais qu'en est-il de l'AM, la modulation d'amplitude ? Ce n'est pas la tasse de thé d'un mélangeur direct. Mais ça marche quand même moyennant quelques restrictions. Il faut régler le VFO sur le battement zéro, donc aussi précisément que possible sur la porteuse de l'émetteur. La modulation devient alors claire, mais encore recouverte par un battement. L'asservissement de phase n'est pas vraiment possible, mais on peut quand même recevoir les stations AM. Tout va quand même mieux avec le démodulateur IQ qui suit.

Le détecteur IQ

Le logiciel SDR traite normalement un signal IQ à 12 kHz. Mais ne pourrait-on pas reporter sur le matériel le traitement du signal ? On passe alors la main à un démodulateur AM pour signaux IQ. Et nous avons montré que c'était plus facile qu'espéré.
La première expérience (**figures 5** et **6**) porte sur un filtre FI à deux inductances fixes. Plus ou moins par hasard, la fréquence de travail s'est établie à 16 kHz alors que les condensateurs de 22 nF étaient installés. Avec un condensateur de couplage de 1 nF, on obtient un bon filtre de bande de la largeur convenable. Les deux signaux I et Q se mélangent avec une différence de phase de 90°. À 16 kHz, cela tombe juste parce que là, le condensateur de 1 nF présente une résistance capacitive de 10 kΩ. Le canal Q est donc déphasé de 90° à la fréquence de travail, tandis que le canal I est couplé sans déphasage. Comme on s'y attendait, les signaux s'amplifient à 16 kHz et s'annulent à −6 kHz. On obtient ainsi de fait une réjection pratique de l'image. La première étape du traitement du signal est ainsi atteinte.
La deuxième est de filtrer le signal, c'est la mission du filtre de bande. Cela fonctionnait bien sur les anciennes radios à tubes et c'est encore le cas maintenant sur des fréquences plus basses. Vient ensuite le démodulateur AM sous la forme d'un détecteur à diode. Pour mettre en exergue la continuité

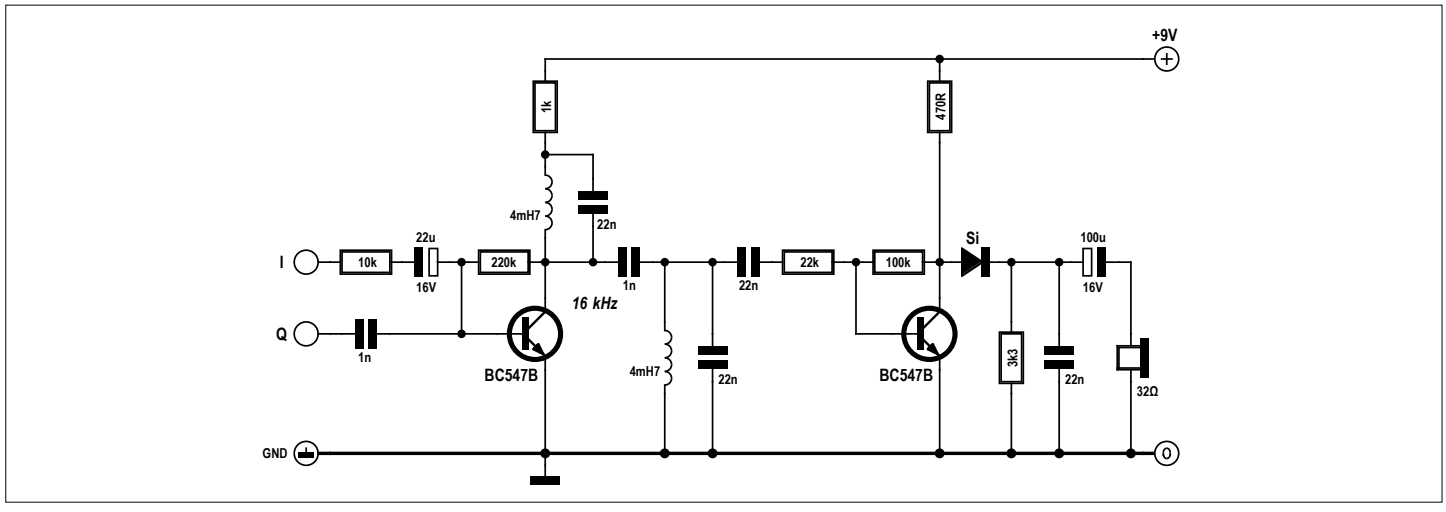


Figure 7. Addition d'un étage à fréquence intermédiaire.

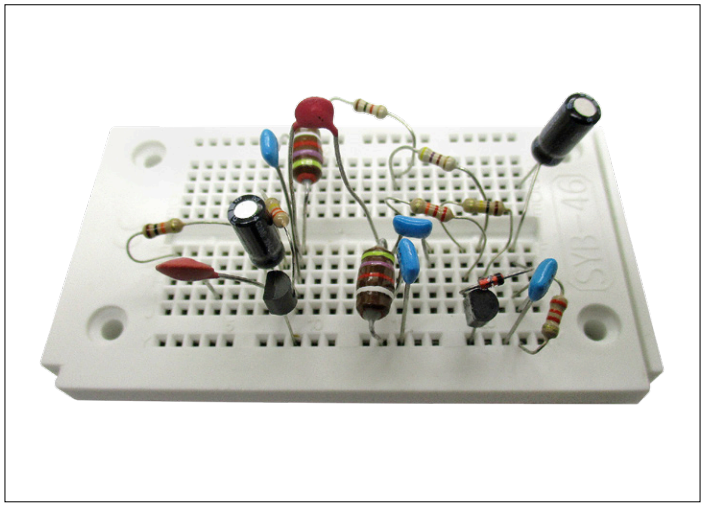


Figure 8. Le circuit complet de la figure 7 sur plaque d'expérimentation.

historique, nous utilisons la diode au germanium OA161 vieille de plus de 50 ans, mais vous pouvez évidemment la remplacer par une 1N60A ou équivalente. Suit alors un écouteur à haute impédance ou un casque à basse impédance précédé d'un transformateur. Et on entend déjà quelque chose. Le son est doux et agréable, c'est principalement au filtre de bande qu'on le doit.
Ce premier circuit ne marche bien qu'avec les signaux les plus forts et ne produit pas encore un volume élevé. Ajoutons donc un autre étage amplificateur à FI (**figure 7**). Il permet de faire travailler le redresseur AM avec déjà une polarisation, ce qui permet à une diode au silicium de fonctionner ici. Comme résultat, on a une bonne sensibilité et une meilleure puissance pour un casque de 32 Ω. Et le détecteur AM peut tout aussi bien attaquer un amplificateur pour haut-parleur.
Toutes les expériences proposées sur la radio logicielle en solo ne sont finalement que des compromis simples. La voie royale serait un véritable traitement numérique du signal au moyen d'un DSP, un processeur fait pour cela. Pourrait-on aller plus loin et le réaliser sur Arduino ? Ce serait en tout cas une prouesse. Si c'est vraiment possible, peut-être qu'un lecteur relèvera le défi.

un démodulateur AM pour signaux IQ, c'est facile à faire

Quoi qu'il en soit, profitez bien de vos expérimentations sur votre *shield* SDR !

(160165 – version française : Robert Grignard)

Liens

[1] www.elektormagazine.fr/140009
[2] www.elektormagazine.fr/160165

émetteur IR quasi universel

Goswin Visschers (Pays-Bas)

C’est toujours au moment où l’émission débute que la zapette ad hoc est introuvable. C’est qu’à chaque appareil correspond une télécommande idiomatique et l’on se retrouve vite avec un assortiment de ces bidules. Sans compter que tous les appareils télécommandés restent sous tension indéfiniment et gaspillent de concert une quantité considérable d’énergie.

On pourrait évidemment mettre fin sans difficulté au premier souci en achetant une télécommande IR programmable, mais j’en voulais une avec entrée numérique pour allumer et éteindre tous les appareils en même temps quand quelqu’un entre ou sort de la pièce. Un détecteur de mouvement est fait pour cela, mais comment le brancher sur la télécommande ? Alors, il n’y a qu’à la construire soi-même !

Quasi universelle

Si je l’ai baptisée ainsi, c’est en raison de limitations techniques

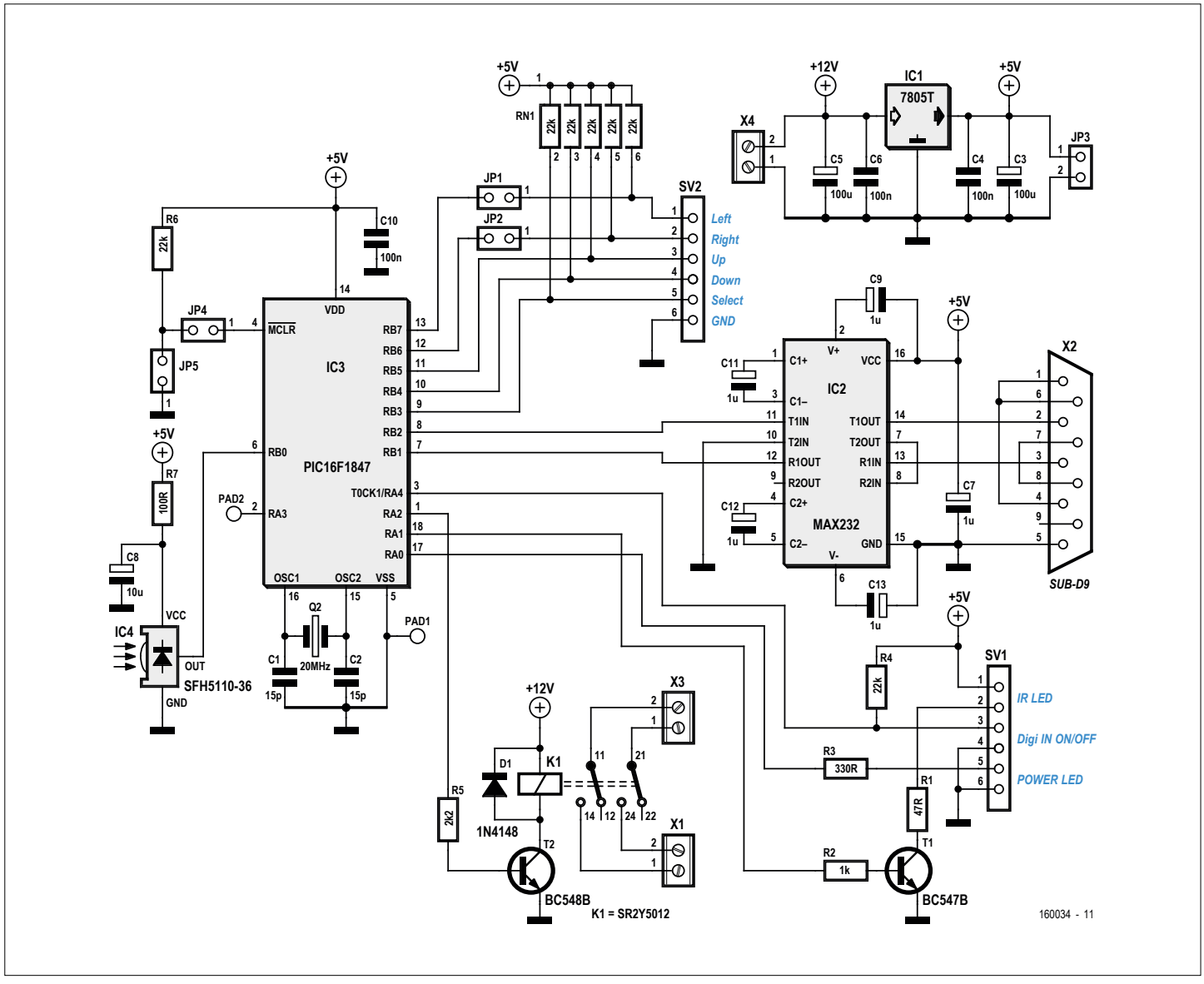


Figure 1. Le schéma de l’émetteur IR quasi universel.

Caractéristiques techniques

- Compatible avec les protocoles suivants :
  - Philips RC5, RC5 étendu, RC6 mode 0 et utilisation du bit de basculement
  - Protocoles NEC et NEC étendu
  - Sony Sirc 12 bits, 15 bits et 20 bits (ce dernier n'est pas testé)
- Envoi des codes d'allumage et d'extinction par entrée numérique
- Possibilité de brancher plusieurs LED IR émettrices en parallèle
- Apprentissage des codes par terminal sériel (RS232)
- Circuit à relais pour débrancher du secteur tous les appareils connectés (économie d'énergie)
- Sélection par bouton *Select* entre quatre sources (TV, radio, USB, CD, etc.)
- Allumage et extinction d'appareils au moyen de maximum six codes IR différents
- Quatre boutons programmables envoient des codes IR différents selon la source choisie.

qui ne permettent pas de couvrir l'ensemble des protocoles IR. Les propriétés du circuit figurent dans l'encadré *Caractéristiques techniques*.

Le circuit

Le schéma de la télécommande quasi universelle, à la **figure 1,** ne renferme rien de spécial. L'alimentation est issue d'un régulateur de tension linéaire de 5 V de type 7805 entouré de quelques condensateurs. Un MAX252 et d'autres condensateurs assurent l'adaptation de niveau de TTL vers RS232 et un récepteur IR standard fournit au microcontrôleur les signaux IR pour la programmation des codes.

Le récepteur IR choisi, un SFH5110-36, est doté d'un filtre de bande à 36 kHz, alors que Sony module à 40 kHz et NEC à 38 kHz. Cela ne donne aucun souci pour peu que la télécommande soit à proximité suffisante du récepteur, les signaux sont alors assez forts pour traverser le filtre.
Le microcontrôleur IC3 régit le tout. Ce PIC16F1847 est cadencé à 20 MHz par le quartz Q2. Les cavaliers JP1, JP2, JP3 et JP4 ne servent qu'à la programmation *en circuit* du microcontrôleur. S'ils ne vous sont pas nécessaires, remplacez-les par des ponts de câblage.
JP3 apporte l'alimentation au programmateur *en circuit* et pourrait éventuellement être abandonné. On détermine avec JP5 si l'entrée Digi IN doit être active au niveau haut ou bas : quand il est mis, Digi IN est active au niveau bas et les codes d'allumage sont envoyés par la LED IR si SV1-3 et SV1-4 sont reliés ensemble. C'est l'inverse si JP5 est ôté.
Les connecteurs PAD1 et PAD2 inutilisés sont là en vue de modifications du logiciel.
Le relais K1 s'active à la réception de codes d'allumage et chute à l'arrivée de codes d'extinction.
La LED IR se branche sur SV1-1 et SV1-2. Si vous raccordez plusieurs LED en parallèle, il faut remplacer la résistance R4 par un pont de câblage et prévoir pour chaque LED une résistance série de 47 Ω, par exemple.

On peut mettre sur SV1-5 et SV1-6 une LED verte qui clignotera pour témoigner de la mise en service du circuit.
RN1 est un réseau de résistances de polarisation haute des entrées RB3 à RB7 quand les interrupteurs sont relâchés.
J'ai dessiné pour ce montage un circuit imprimé dans Eagle. Tous les fichiers du projet (Eagle, tracé des pistes, etc.) sont disponibles sur [1].

Le logiciel

Sur la plupart des télécommandes universelles, on échantillonne à haute vitesse les codes IR pour les stocker dans la mémoire du microcontrôleur. L'avantage de la méthode, c'est d'avoir tous les codes IR qui existent dans les différents protocoles. L'inconvénient, c'est qu'il faut beaucoup de mémoire pour les retenir. Et comme j'aime employer les microcontrôleurs de la série PIC16 de Microchip, j'aurai vite des soucis de capacité de mémoire.

On pourrait y parer en ajoutant une EEPROM externe avec un surplus de mémoire, mais j'ai préféré faire l'impasse sur quelques protocoles et rendre plus astucieux le logiciel, que vous pouvez d'ailleurs télécharger aussi sur [1].
L'astuce du logiciel, c'est de reconnaître le protocole IR qu'il doit apprendre et donc de ne s'intéresser qu'aux codes IR qu'il contient. Le type de protocole peut tenir dans un octet et les données ne dépassent jamais 32 bits, donc 4 octets. Comme cette astuce demande de consigner plus de code de programme, j'ai choisi le PIC16F1847 qui dispose de 14 Ko de mémoire de programme. Elle est pleine à craquer, mais dispose de nombreux protocoles IR.

C'est d'abord la largeur de la première impulsion IR qui permet d'identifier le protocole. NEC, par exemple, utilise une impulsion de 9 ms (**figure 2**) et RC5 une impulsion de départ de 0,9 ms (**figure 3**). Sirc de Sony et RC6 de Philips donnent

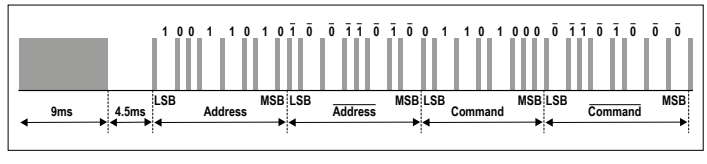


Figure 2. Exemple de code IR du protocole NEC.

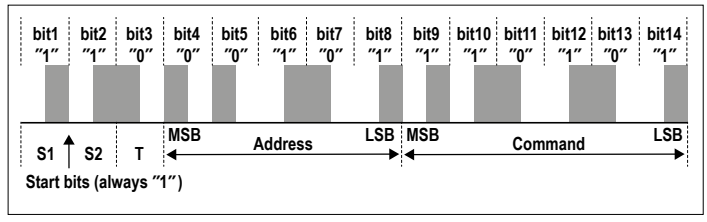


Figure 3. Exemple de code IR du protocole RC5.

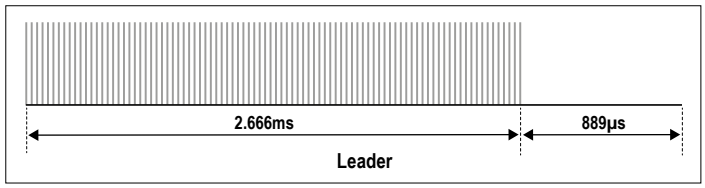


Figure 4. Impulsion de départ du protocole RC6

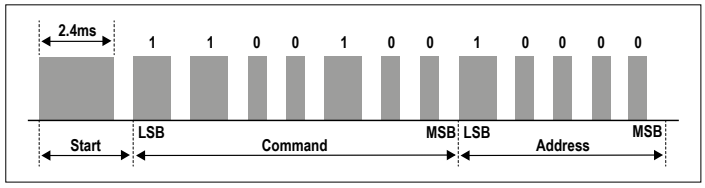


Figure 5. Exemple de code IR du protocole Sony SIRC.

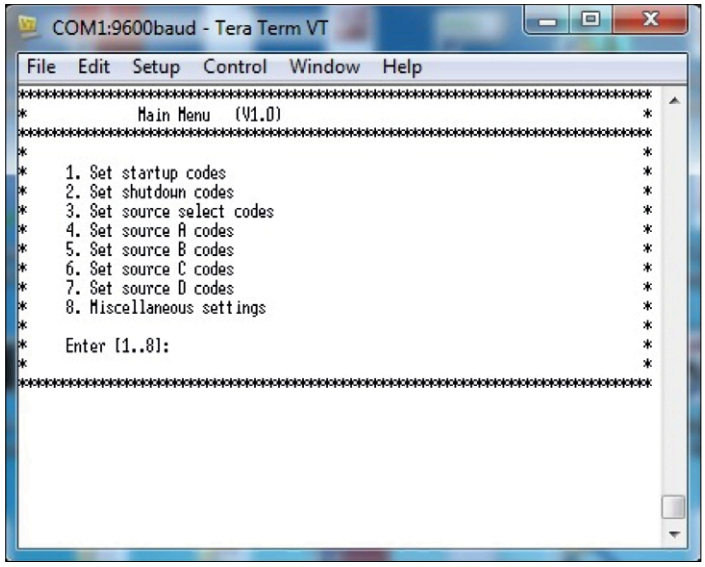


Figure 6. Le menu de sélection qui s'affiche après branchement du port sériel et dès l'allumage du circuit.

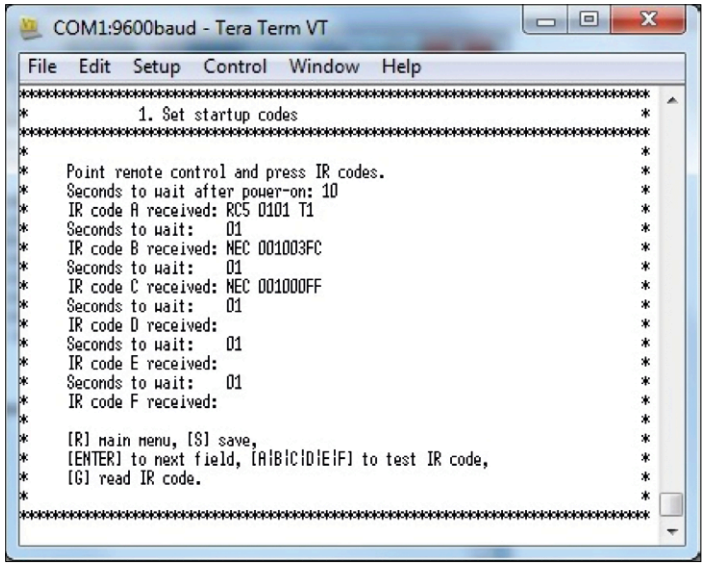


Figure 7. Les codes d'allumage sont programmés avec l'option 1.

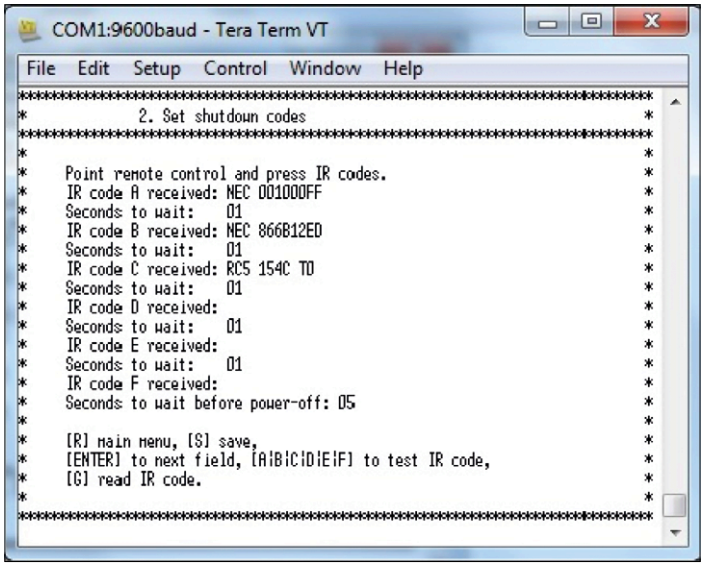


Figure 8. Les codes d'extinction de l'appareil sont programmés avec l'option 2.

des impulsions à peu près d'égale largeur, de l'ordre de 2,4 ms. Pour les différencier, on mesure encore la pause jusqu'à la deuxième impulsion : avec RC6, elle est de 0,889 ms tandis qu'avec Sirc, c'est 0,6 ms, cf. **figures 4** et **5**.

Une fois le type de protocole identifié, le logiciel sait combien de bits de données sont codés. RC5 et RC6 opèrent en codage biphasé, NEC utilise la durée de l'impulsion et Sony la largeur d'impulsion. Pour de plus amples informations, vous pouvez consulter les sources [2], [3], [4] et [5] citées en fin d'article. Le microcontrôleur lit ensuite les bits de données IR et en fait des octets convenables pour les stocker en EEPROM.
L'émission des codes IR se passe exactement à l'envers. Le premier octet dans l'EEPROM représente le type de protocole et dès lors, le logiciel sait combien de bits de données il doit coder et à quelle fréquence les moduler. Ensuite, il transmet les octets de données l'un après l'autre dans l'ordre où il les a appris.

Lors de la réception des codes IR, le SFH5110-36 démodule le signal et donc supprime l'onde porteuse, mais en émission, la LED IR doit de nouveau moduler les codes IR sur une porteuse à 36, 38 ou 40 kHz selon le type de protocole.
Sur une télécommande Philips ou Sony, les codes IR se répètent aussi longtemps que le bouton est actionné. Le temps de pause entre deux codes répétés dépend du type de protocole. Les codes IR Philips contiennent un bit de basculement qui se maintient tout le temps d'action sur le bouton. Après un arrêt et une nouvelle action, ce bit bascule.
NEC s'y prend autrement. Il n'envoie le code IR d'origine qu'une seule fois, mais le répète après un intervalle constant aussi longtemps que le bouton est enfoncé. Certains appareils attendent toujours au moins une répétition du code avant de réagir. C'est pourquoi le code doit toujours être émis au moins une deuxième fois.
Pour vérifier si les codes IR sont réellement émis, on peut simplement approcher à quelques centimètres de la LED émettrice la caméra d'un téléphone portable, elle est aussi sensible au

rayonnement IR, on le voit à l'écran. Une telle LED IR n'a qu'un petit angle de radiation, il faut donc bien centrer la caméra dans sa direction.

Apprentissage des codes IR

Reliez l'émetteur IR quasi universel par un câble inverseur (*null modem*) à un port RS232 d'un PC, ce peut aussi être un convertisseur RS232 vers USB. Lancez un programme de terminal sur le PC, Tera Term par exemple, et réglez le port sériel sur 9600 bauds, 8 bits de donnée, pas de parité et 1 bit d'arrêt (8n1). Quand la télécommande est branchée sur une alimentation, un menu s'affiche d'emblée sur le terminal (**figure 6**). Veillez à ce que le récepteur IR se trouve à l'abri de sources lumineuses et d'écrans de télé avant de commencer l'apprentissage des codes IR. Les sources lumineuses et les écrans plats perturbent la réception des codes, ce qui peut entraîner des erreurs de lecture et des codes farfelus, avec éventuellement un message d'erreur.

On programme dans l'option 1 (**figure 7**) les codes IR nécessaires à la mise en marche et dans l'option 2 (**figure 8**), ceux pour la mise à l'arrêt. On apprend à la télécommande à commuter entre les différentes sources au moyen de l'option 3 (**figure 9**).

Dans les options 4 à 7 du menu (**figure 10**), on programme pour chaque source les codes IR des boutons avec flèche « haut », « bas », « gauche » et « droite ». Ces quatre boutons peuvent envoyer des codes IR différents d'une source à l'autre. Pour lire un nouveau code IR, il faut appuyer sur le « G » ; toutes les commandes sont en lettres capitales. Répétez plusieurs fois le code à apprendre en appuyant sur le G et comparez les codes reçus. Celui qui revient le plus souvent est probablement le bon. Clôturez chaque changement avec le « S ». Et le « R » vous renvoie au menu principal.
Avec l'option 8 du menu, on peut définir les fonctions suivantes :

- déterminer le nombre de fois que le code IR doit être renvoyé, cette option simule le maintien de l'action sur un bouton de la télécommande ;
- activer ou désactiver la fonction *Auto power-on* de l'entrée numérique.

L'usage habituel

Si vous utilisez l'entrée numérique, vous pouvez déterminer avec JP1 s'il faut envoyer le code IR de démarrage pour un niveau logique haut ou bas sur l'entrée numérique et bien entendu le contraire pour un code d'arrêt.

À tout moment, on peut utiliser le bouton *Select* pour envoyer les codes IR de démarrage et d'arrêt. On envoie un code de démarrage en appuyant brièvement sur *Select* pour activer le relais. Ensuite, après le temps d'attente programmé, on peut commencer à envoyer les codes IR de mise en marche.
Dès que les codes de mise en marche ont été envoyés, on peut choisir les différentes sources en appuyant brièvement sur *Select*. Les codes IR des boutons avec flèche (haut, bas, gauche et droite) diffèrent en fonction des sources.

Pour débrancher, appuyer pendant au moins 2 s sur *Select*. Le code IR d'arrêt est alors envoyé et finalement, le relais chute.

(160034 – version française : Robert Grignard)

Liens

[1] www.elektormagazine.fr/160034
[2] www.sbprojects.com/knowledge/ir/nec.php
[3] www.sbprojects.com/knowledge/ir/rc5.php
[4] www.sbprojects.com/knowledge/ir/sirc.php
[5] www.sbprojects.com/knowledge/ir/rc6.php

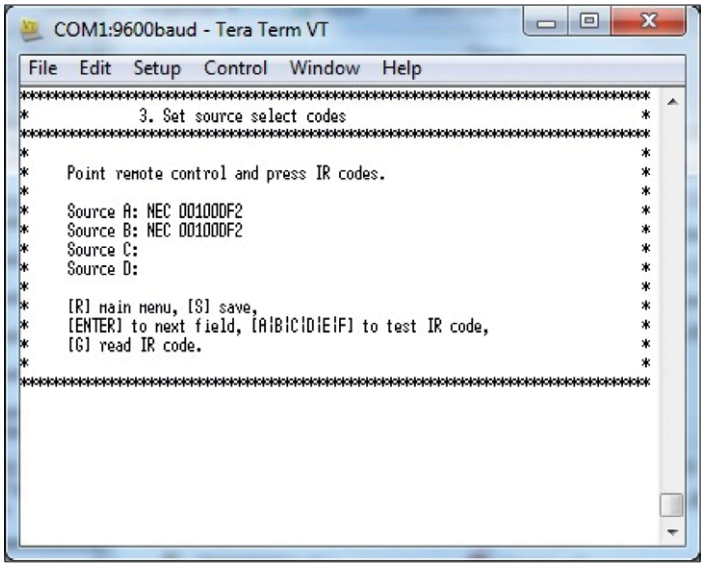


Figure 9. On programme le code de sélection de source avec l'option 3.

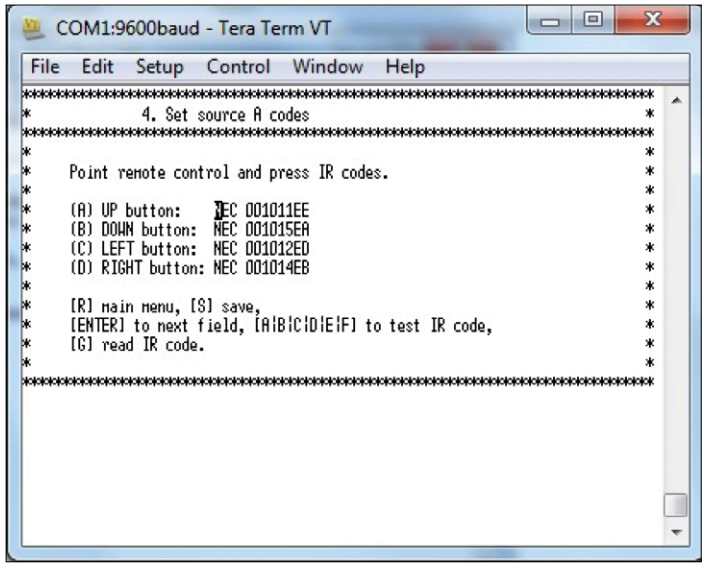


Figure 10. Les options 4 à 7 programment les boutons avec flèche.

diplexeur d'antenne

ajouter la réception numérique (DAB+) à un autoradio

Les temps changent, la radio aussi. La FM va céder la place à la DAB (*Digital Audio Broadcasting*). Dans la voiture aussi les deux vont cohabiter : si vous achetez un autoradio moderne avec DAB+, vous y verrez deux prises d'antenne. Il n'y a qu'une antenne sur la voiture. Que faire ? Utiliser un aiguillage HF.

Alfred Rosenkränzer (Allemagne)

C'est quand je me suis aperçu que la radio FM dans la voiture faiblissait que j'ai découvert ce remarquable filtre pour installer un appareil plus moderne avec réception numérique (DAB+). Par chance, notre auto est encore équipée d'une prise DIN ce qui simplifie le remplacement, puisqu'on trouve des postes compatibles.

Un autoradio DAB+ a donc deux entrées d'antenne, l'une FM, l'autre DAB+. L'antenne auto est normalement utilisée pour les ondes ultracourtes ; pour DAB+, certains appareils sont livrés avec une antenne ronde, si la voiture n'est pas préparée pour la radio numérique, ce qui est notre cas. Cette antenne ronde à l'avant peut réduire la visibilité, et en plus, il faudrait forer un trou dans le tableau de bord pour le câble d'antenne. En aucun cas, je ne voulais risquer de discussion orageuse sur le sujet avec ma femme.

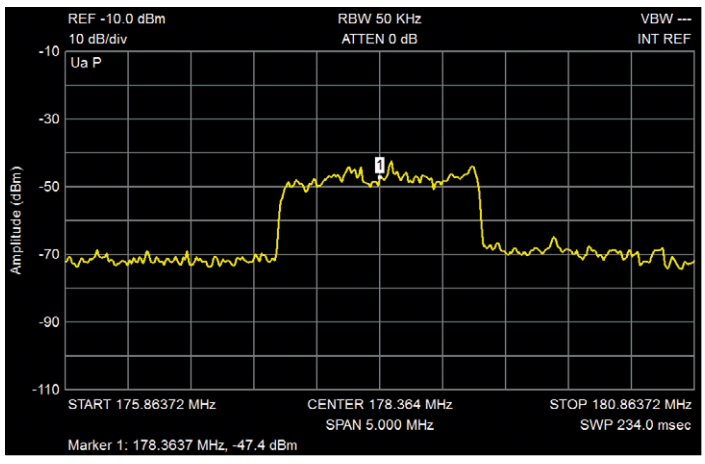


Figure 1. L'occupation spectrale du canal 5c de DAB+ autour de 178 MHz.

Avant d'utiliser l'antenne d'origine, il faut savoir si elle convient pour la réception de la gamme DAB. Une antenne active est dotée d'un filtre passe-bas. Le plus simple, pour le savoir, c'est de la mesurer avec un analyseur de spectre (**fig. 1**), sinon, la tester avec un récepteur DAB+. La bande FM couvre de 88 à 108 MHz, la bande III DAB va de 174 à 230 MHz. La bande L dans la gamme de 1,5 GHz ne convient pas pour la transmission vers une automobile.

Il faut alors répartir le signal disponible sur deux entrées d'antenne. La solution élémentaire consiste à les raccorder par deux câbles en parallèle et, bien sûr, l'impédance de la ligne n'est plus respectée, ce qui entraîne des réflexions et éventuellement une extinction partielle du signal.

Trois résistances pour composer un diviseur permettent d'éviter la désadaptation, mais apportent forcément une atténuation supplémentaire du signal, ce n'est sûrement pas l'endroit pour jouer à cela. Un transformateur HF comme séparateur offre une autre solution.

Premier essai

On peut aussi faire la séparation en combinant un filtre passe-bas et un passe-haut avec des caractéristiques de fréquence appropriées.

Pour l'expérimenter, j'ai d'abord mesuré deux filtres du commerce. La **figure 2** montre l'allure en fréquence des filtres,

chacun mesuré séparément avec un analyseur de réseau. Les fréquences ne correspondent pas à la séparation entre FM et DAB, mais les deux filtres sont bien linéaires dans la bande passante.

Quand on réunit en parallèle les deux filtres par un connecteur en T, la caractéristique de fréquence de chaque filtre est sérieusement influencée par l'autre (**fig. 3**). J'ai alors essayé une répartition par un séparateur à résistances de puissance et un atténuateur supplémentaire de 6 dB derrière le séparateur. L'influence réciproque a considérablement diminué, mais le résultat n'est pas vraiment meilleur. On constate une atténuation supplémentaire de 3 à 6 dB. Et puis le passe-bas prend une pente descendante. Ce n'est sûrement pas la meilleure solution.

Calculer un diplexeur

La clé vient peut-être d'un diplexeur. Il s'agit d'une combinaison d'un filtre passe-bas et d'un passe-haut, ou d'un passe-bande et d'un réjecteur de bande, autrement dit un aiguilleur de hautes fréquences.

Les deux filtres sont organisés pour se fournir à la même source. Pour les calculer, il existe le logiciel *DiplexerDesign* de James L. Tonne [1] qui permet de choisir entre les combinaisons *Lowpass/Highpass* et *Bandpass/Bandstop*. Nous ne considérons ici que la formule passe-bas/passe-haut.

Sur la page *Design* de ce programme (**fig. 4**), on définit les caractéristiques du diplexeur. De l'ordre du filtre dépend le prix (le nombre de composants) et la raideur des flancs. La *Crossover freq* donne le point de croisement des deux filtres. Pour l'exemple FM/DAB+, il se situe à peu près au milieu entre la plus haute fréquence FM (108 MHz) et la plus basse en DAB+ (174 MHz). On peut ainsi ajuster finement la fréquence de transition pour égaliser par le filtre l'atténuation de chacune des bandes indésirables. Vous pouvez aussi chercher à remplacer par des valeurs disponibles de bobines celles qui sont fantaisistes.

Le passe-bas dans la partie supérieure commence toujours par une bobine dans le sens longitudinal. Lors du calcul d'un filtre unique, on peut choisir de commencer par une bobine longitudinale (structure en T) ou un condensateur à la masse (structure en Π), mais ici, ce n'est pas possible. De son côté, le passe-haut commence toujours par un condensateur longitudinalement.

Le *Passband ripple* donne l'ondulation en amplitude dans la bande passante, comme dans un filtre de Tchebychev. Une plus grande ondulation conduit à une meilleure raideur des flancs et inversement.

System Z donne l'impédance caractéristique, dans notre exemple 50 Ω.

En poussant sur le bouton *Plot*, on passe à l'écran suivant (**fig. 5**).

Ici encore, on peut modifier l'ordre, l'ondulation et la fréquence de transition et observer les résultats dans le diagramme. La présentation se règle dans *Plot Options* et avec *Markers*, on peut insérer des marqueurs.

On simule ensuite dans Simetrix le diplexeur ainsi obtenu et on le compare à un passe-bas de Tchebychev.

Le circuit pratique

J'ai ensuite créé un circuit imprimé avec EAGLE. Pour les valeurs inexistantes d'inductances et de condensateurs, on se débrouille

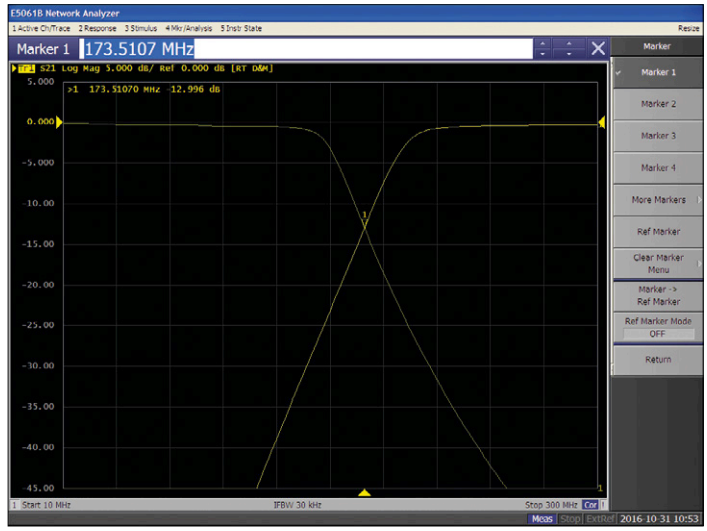


Figure 2. Les filtres passe-bas SLP-150+ et passe-haut SHP-250+ de Mini-Circuits, mesurés chacun séparément avec un analyseur de réseau.

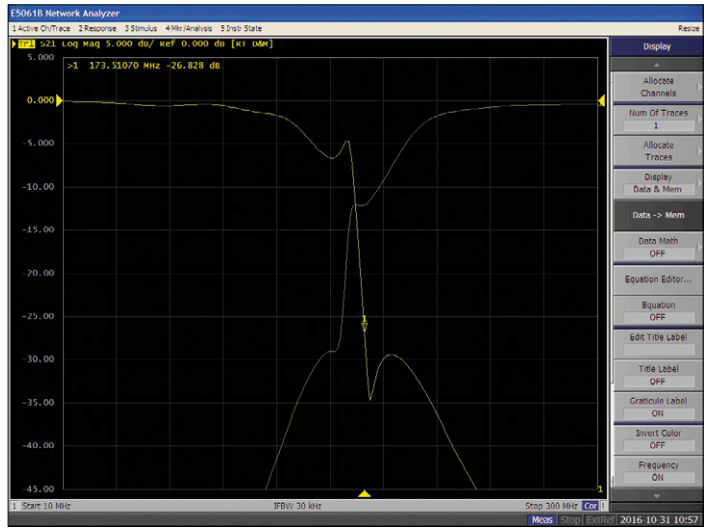


Figure 3. Quand les deux filtres sont reliés en parallèle par connecteur en T, il y a une forte influence mutuelle sur la caractéristique de fréquence.

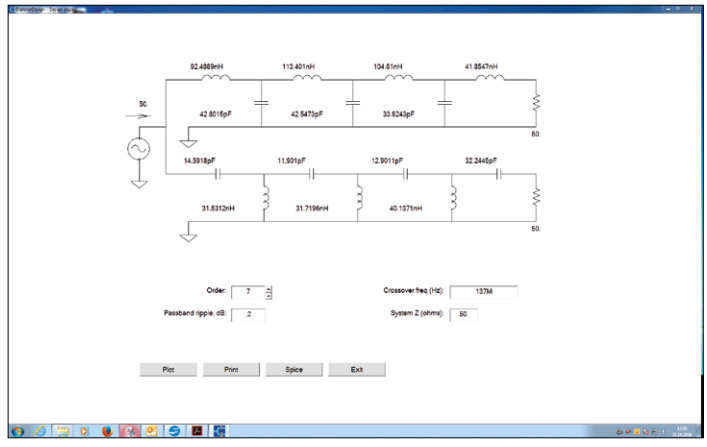


Figure 4. La feuille de dessin du programme *DiplexerDesign*.

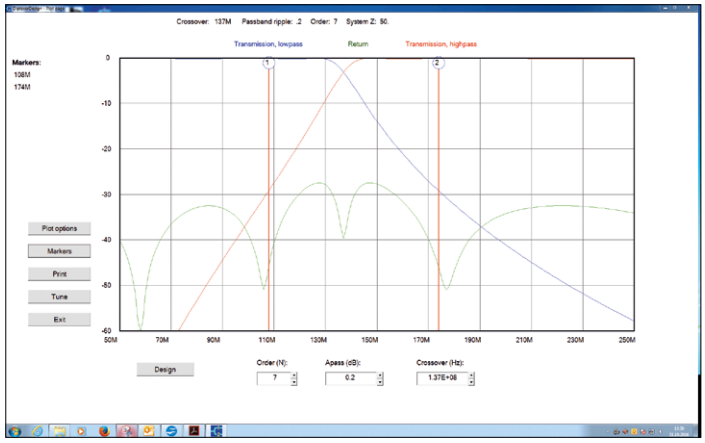


Figure 5. La page graphique du programme *DiplexerDesign*.

par des mises en série ou en parallèle. Les condensateurs, on peut sans difficulté les mettre en parallèle, même à côté l'un de l'autre. Il en va autrement avec les inductances en série, dont les champs magnétiques s'influencent pour former de (très piètres) transformateurs. Du coup, la valeur effective des inductances n'est pas exactement la somme des deux. Aussi, les installe-t-on à distance et à angle droit, dans la mesure où c'est possible avec des CMS.

L'ordre maximum est 7. On peut réaliser des ordres plus petits, il suffit d'éliminer des bobines, remplacées par des ponts de câblage, et des condensateurs. Les inductances sont disponibles en boîtier 0603.

Le schéma du circuit pratique est à la **figure 6**, le circuit imprimé correspondant à la **figure 7**. Le circuit peut s'installer dans un petit boîtier en plastique et y être vissé (**fig. 8**). Les câbles d'entrée et de sortie sont soudés directement sur le circuit imprimé. Les deux filtres sont séparés par un pont de masse sur la face supérieure, la face inférieure est un plan de masse complet.

Les résultats de mesures à la **figure 9** confirment qu'ils sont compris dans les fourchettes de tolérance des composants. À environ 45 dB d'atténuation, le passe-bas entre en saturation. Ce n'est pas joli, mais sans importance pour la fonction.

La version active

Comme la puissance émise par les stations DAB+ est relativement faible par rapport aux émetteurs FM, j'ai cherché un amplificateur adéquat et j'ai trouvé le MAX2630 de Maxim. Comme l'indique la fiche technique [2], il existe en boîtier à quatre contacts SAT143. Il suffit d'y relier un condensateur de découplage pour l'alimentation, un condensateur à l'entrée et un autre à la sortie. Il se monte impeccablement sur une

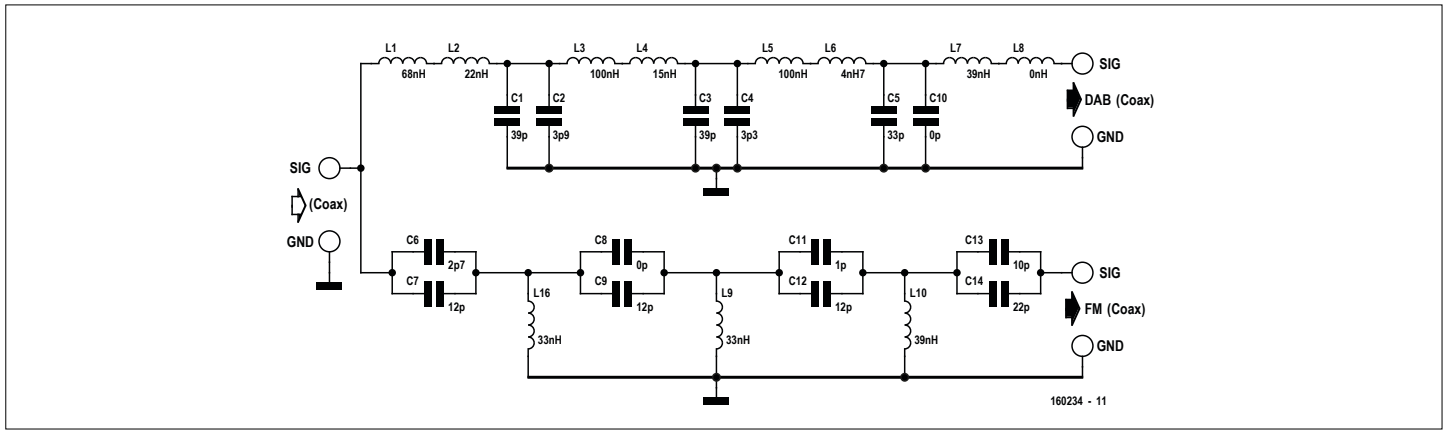


Figure 6. Le schéma de l'aiguilleur HF entre FM et DAB+.

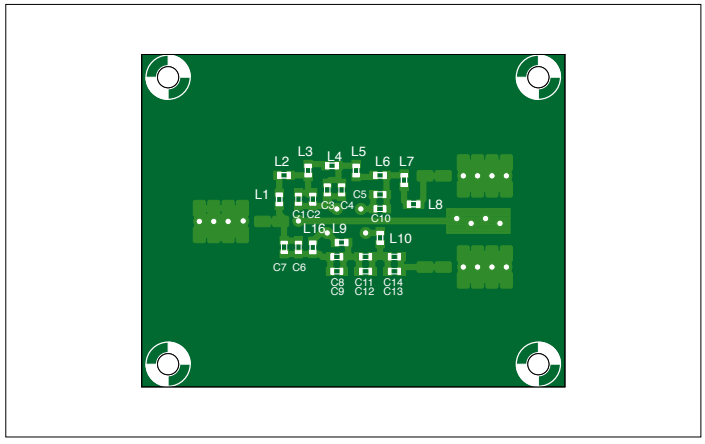


Figure 7. Le tracé d'EAGLE pour le circuit de la figure 6.

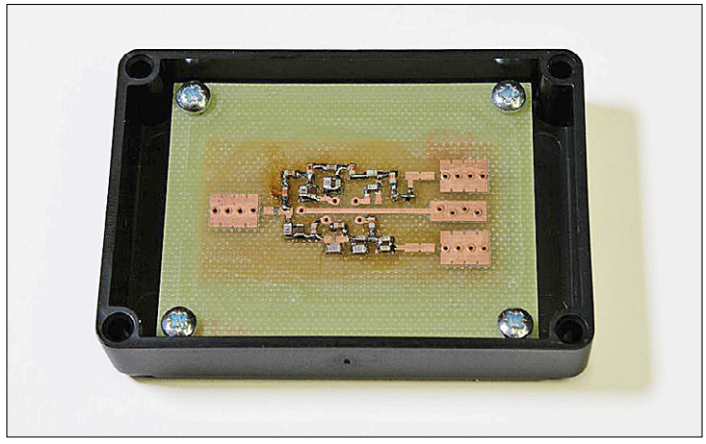


Figure 8. Le prototype du diplexeur dans son boîtier.

plaque à trous, je n'ai donc pas dessiné de circuit imprimé pour lui. Son gain avoisine 15 dB et le facteur de bruit vaut environ 4. La caractéristique de fréquence mesurée montre une ligne presque plate jusqu'à 250 MHz, mais elle se prolonge nettement plus loin.

Encouragé par ces résultats, j'ai construit le diplexeur actif de la **figure 10**. J'ai prévu tant à l'entrée qu'aux deux sorties un MAX2630.

Celui d'entrée ne s'indique que si la voiture est équipée d'une antenne passive. Avec le niveau élevé d'une antenne active, ce pourrait être trop fort. Dans ce cas, on ponte l'entrée à la sortie d'un fil ou d'une résistance CMS de 0 Ω. Idem pour la sortie FM si jamais le récepteur risque la saturation. En revanche, un amplificateur sur la sortie DAB+ est certainement bienvenu. L'alimentation provient d'un régulateur de tension de 3,3 V, chaque amplificateur consomme environ 7 mA. La diode en série à l'entrée empêche l'inversion de polarité, les deux résistances réduisent la dissipation thermique dans le régulateur. On peut obtenir le 12 V à la sortie de l'autoradio, il est prévu pour l'alimentation d'une antenne active. Le circuit du diplexeur actif (**fig. 11**) se loge dans un boîtier en aluminium injecté de Hammond (1550WQ).

(160234 – version française : Robert Grignard)

Liens

[1] www.tonnesoftware.com/diplexer.html

[2] https://datasheets.maximintegrated.com/en/ds/MAX2630-MAX2633.pdf

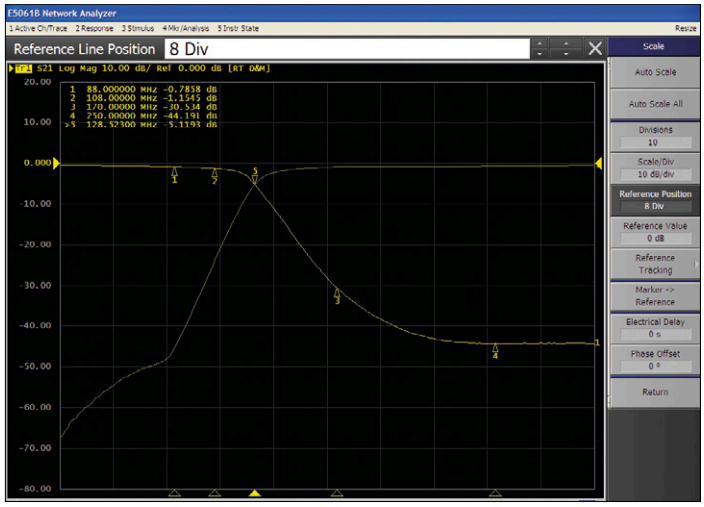


Figure 9. Les caractéristiques de fréquence du diplexeur tracées avec les marqueurs.

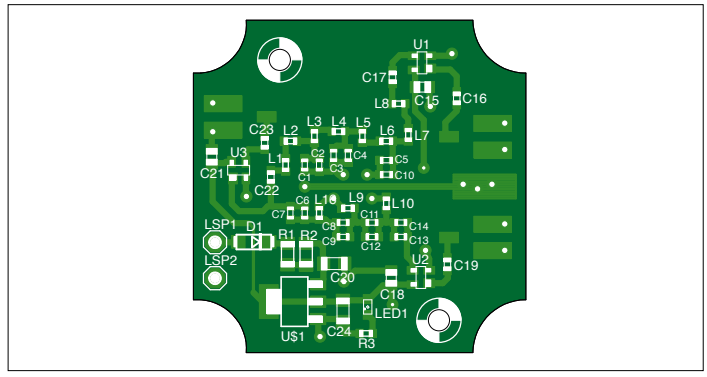


Figure 11. Tracé des pistes et sérigraphie d'implantation des composants du diplexeur actif.

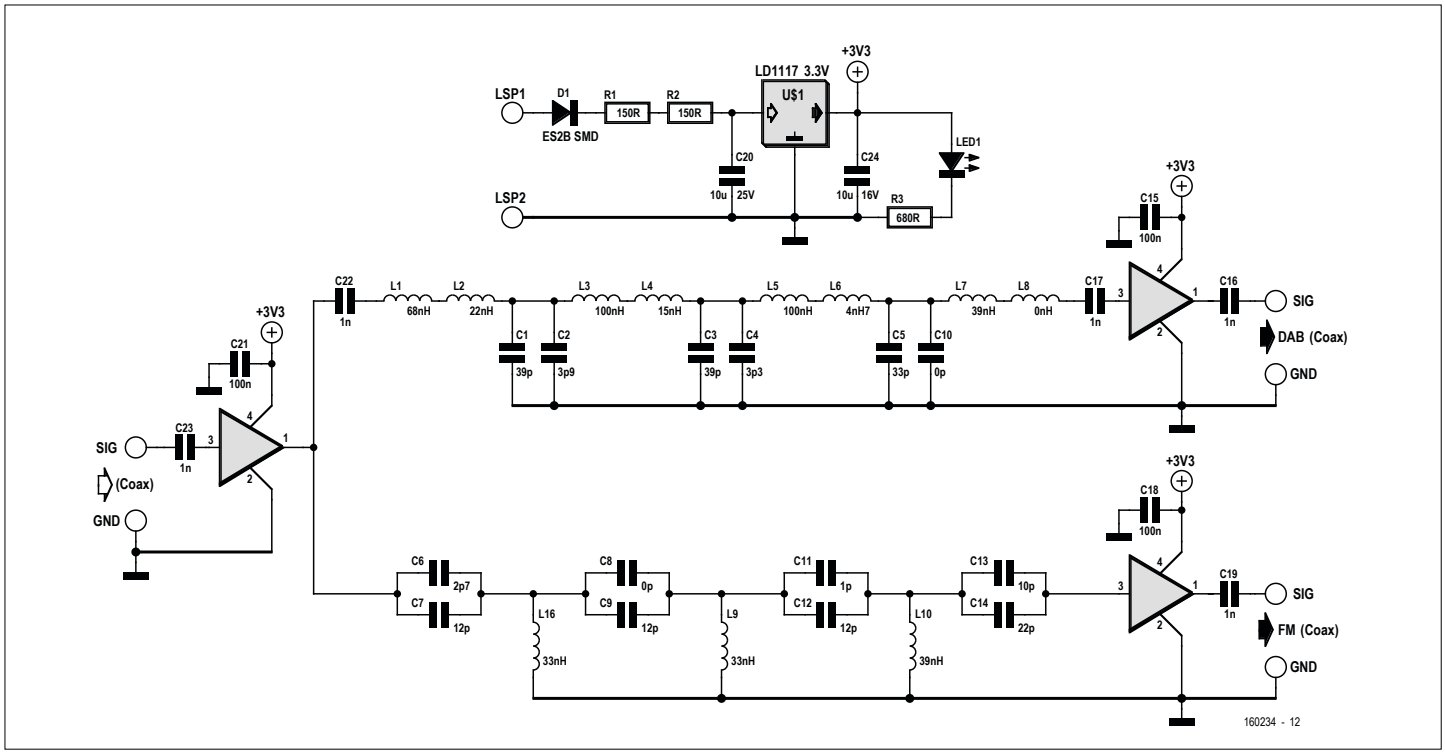
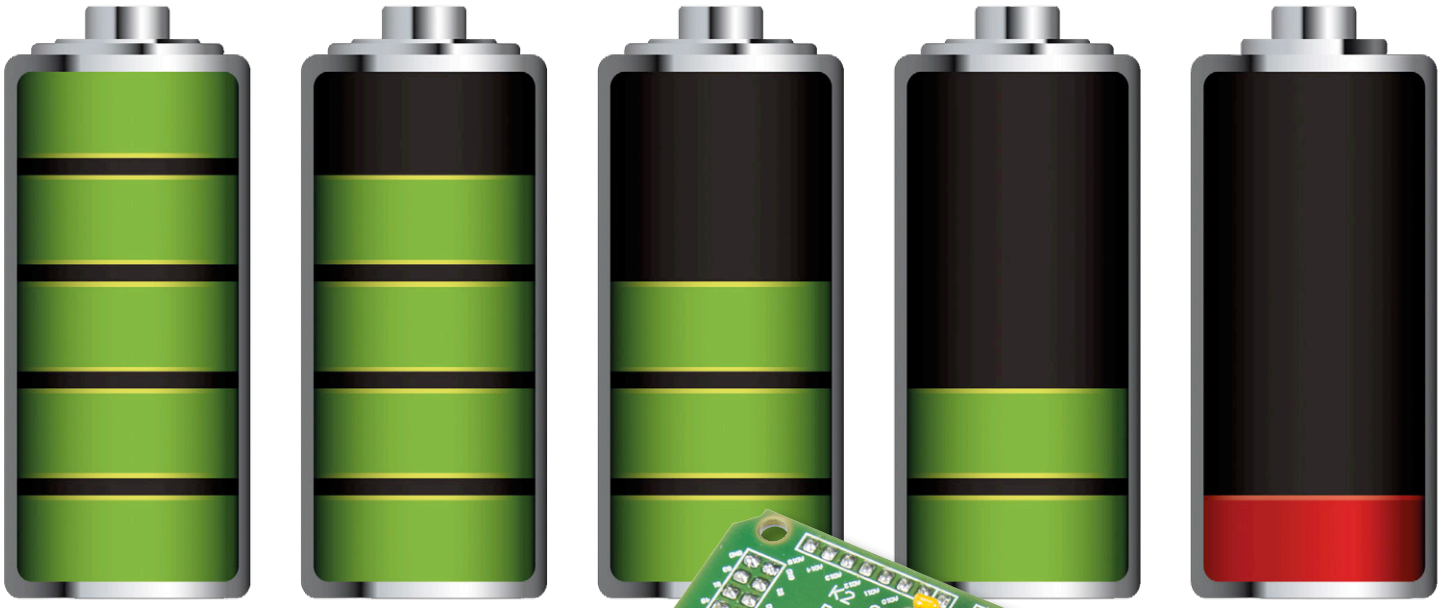


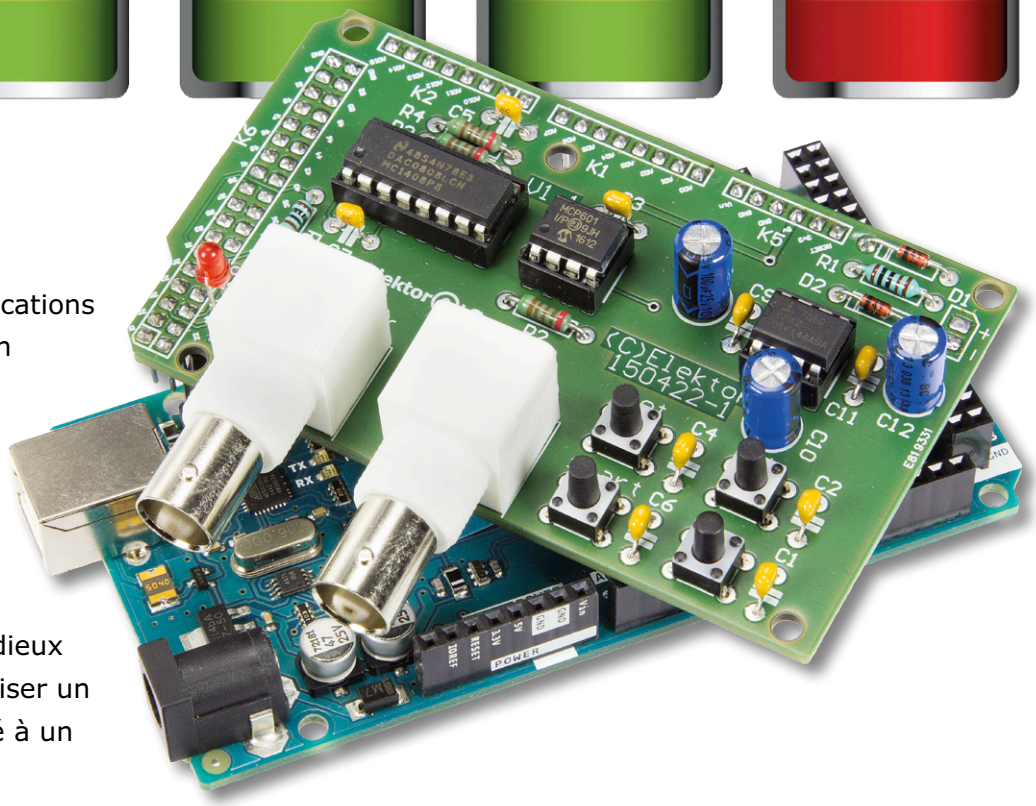
Figure 10. Le schéma du diplexeur actif.

pisteur de tension

mesures à long terme sur oscilloscope avec un *shield* Arduino



Kurt Schuster (Allemagne) et
Luc Lemmens (labo d'Elektor)



Pour observer les lentes modifications de tension d'un capteur ou bien la courbe de décharge d'un accumulateur est-il vraiment nécessaire de rester assis à côté et de relever toutes les x secondes ou minutes la valeur sur l'afficheur d'un multimètre ? Il est moins fastidieux et beaucoup plus élégant d'utiliser un dispositif à base d'Arduino relié à un oscilloscope.

Deux questions ont motivé Kurt Schuster pour lancer ce projet : « *Cet accumulateur est-il encore fonctionnel ?* » et « *Ce chargeur charge-t-il cet accumulateur de manière correcte ?* ». Une représentation graphique de la tension de charge et de celle de décharge serait très utile pour y répondre. Malheureusement, même avec une mémoire de taille respectable, les limites d'un oscilloscope numérique sont vite atteintes. Le balayage horizontal le plus lent est typiquement de 1 à 50 s/div (secondes par division). Avec un écran à 12 divisions horizontales, on arrive à visualiser entre 12 s et 10 min sur une seule vue. C'est insuffisant quand il s'agit de modifications lentes qui peuvent prendre des heures. Une mémoire plus grande ne sera pas utile non plus, à moins d'être prêt à faire défiler des dizaines de vues. La meilleure impression optique s'obtient (le plus souvent) quand toute la courbe tient sur une seule vue.
Il est également possible de résoudre le

problème avec un enregistreur de données ou un multimètre capable d'enregistrer (grâce à une mémoire intégrée et/ou au transfert vers un PC avec le logiciel adapté). Mais ces fonctions ne sont pas standard, et quand c'est le cas, ces appareils de mesures ne sont pas bon marché. La décision a donc été prise d'élargir la base de temps de l'oscilloscope disponible à l'aide d'un circuit à microcontrôleur bon marché et de développer, avec ce *pisteur de tension*, un dispositif adapté aux oscilloscopes. Disons-le tout de suite : même si l'interface utilisateur est « digitale » avec des boutons et que l'affichage des réglages a lieu sur l'écran de l'oscilloscope, le pisteur de tension n'est pas seulement adapté aux oscilloscopes numériques les plus récents, mais il fonctionne également très bien avec des oscilloscopes analogiques plus anciens.

Liminaire

De fait deux questions se posèrent : « Comment un tel dispositif doit-il fonctionner ? » et « Quels composants est-il raisonnable d'utiliser pour réaliser le pisteur de tension ? ». La réponse à la première question est simple : un convertisseur analogique/numérique doit échantillonner périodiquement la tension présente et un microcontrôleur doit mémoriser les valeurs délivrées par ce CA/N ; ensuite le µC doit envoyer périodiquement ces valeurs à l'oscilloscope, à nouveau sous forme analogique via un CN/A, pour obtenir une courbe de tension « en accéléré ». Comme nous avons un microcontrôleur à notre disposition, nous l'utiliserons pour ajouter une interface utilisateur avec des boutons-poussoirs et pour afficher sous forme graphique les paramètres sur l'oscilloscope, grâce à la production de niveaux de tension spéciaux pour le signal analogique de sortie.

Ceci nous amène à la réponse de la deuxième question : comme la carte Arduino est bon marché et que ses outils de développement traditionnels sont mis à disposition gratuitement (EDI Arduino), il est inutile de réinventer la roue, nous utiliserons une carte Arduino comme support prêt à l'emploi. Le matériel supplémentaire requis prend la forme d'un *shield*, ce qui limite les besoins à cette carte additionnelle.

La plupart des microcontrôleurs auront suffisamment de RAM puisque la quantité de données à stocker correspond à la résolution horizontale d'un oscilloscope numérique – le reste est superflu. Kurt Schuster a utilisé une carte Arduino Mega 2560 avec quand même 8 Ko de RAM. Mais pour les données et le programme qu'il a écrit seulement 1,8 Ko sont nécessaires. De fait n'importe quelle carte Arduino peut convenir, même une Uno, à condition que la puissance de calcul soit suffisante et qu'elle tourne à la fréquence de 16 MHz.

Avec une approche très spartiate, un microcontrôleur, en l'occurrence une carte Arduino, suffirait. En effet un microcontrôleur ATmega dispose toujours d'au moins une « entrée analogique » derrière laquelle travaille un CA/N à 8 bits assez rapide. Ensuite on pourrait réaliser une sortie analogique avec une sortie numérique qui travaille en mode PCM ou modulation de largeur d'impulsion. Toutefois un « vrai » CN/A sous forme d'un circuit séparé est idéal et plus précis puisqu'il peut convertir directement un octet en une tension analogique. Si le CN/A est suivi d'un AOP de type *rail-to-rail*, on obtient un tampon ou « étage de sortie » à basse impédance. Avec une alimentation symétrique, il est même possible d'avoir un signal de sortie proprement référencé par rapport à la masse.

Composants et circuit

On pourrait simplement monter sur une carte un CN/A hypermoderne, qui sait presque « tout faire », y ajouter des boutons, etc. et nous aurions notre *shield*. Premièrement ces puces sont chères, deuxièmement elles sont aujourd'hui toujours de type CMS. Comme la place ne manque pas sur la carte du *shield*, nous n'utiliserons que des composants traversants pour faciliter la réalisation ; des connaissances réduites en soudure suffiront. Heureusement il existe encore un CN/A de type DAC0808 réputé, bon marché, ancien mais toujours disponible, qui ressemble à un vrai circuit intégré et qui ne s'envole pas au moindre souffle d'air

Composants

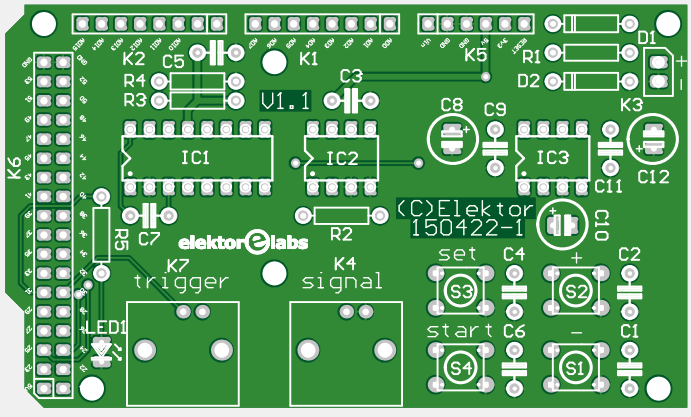


Figure 1. Circuit développé par le laboratoire d'Elektor, disponible dans l'e-choppe.

Résistances :
R1, R5 = 1 kΩ, 5%, ¼ W
R2 à R4 = 5,11 kΩ, 1%, ¼ W, film métal

Condensateurs :
C1 à C7, C9, C11 100 n, 50 V, X7R, pas de 5,08 mm
C8, C10, C12 = 100 µ, 25 V, radial, pas de 2,54 mm

Diodes :
D1, D2 = 1N4148
LED1 = LED, rouge, 3 mm
IC1 = DAC0808LCN, DIP16
IC2 = MCP601-I/P, DIP8
IC3 = ICL7660CPAZ, DIP8

Divers :
K1, K2 = barrette mâle à 1×8 picots
K3 = barrette mâle à 1×2 picots
K7, K4 = connecteur BNC courbé pour montage sur circuit imprimé
K5 = barrette mâle à 1×6 picots
K6 = barrette mâle à 2×18 picots
S1 à S4 = interrupteur unipolaire de 6×6 mm pour montage sur circuit imprimé
Supports de circuit intégré pour IC1 à IC3
Deux câbles coaxiaux avec connecteurs BNC
Circuit imprimé réf. 150422-1 dans l'e-choppe Elektor
Arduino Mega réf. 140566-93 dans l'e-choppe Elektor

Attention : tous les connecteurs doivent être montés sous le circuit imprimé !

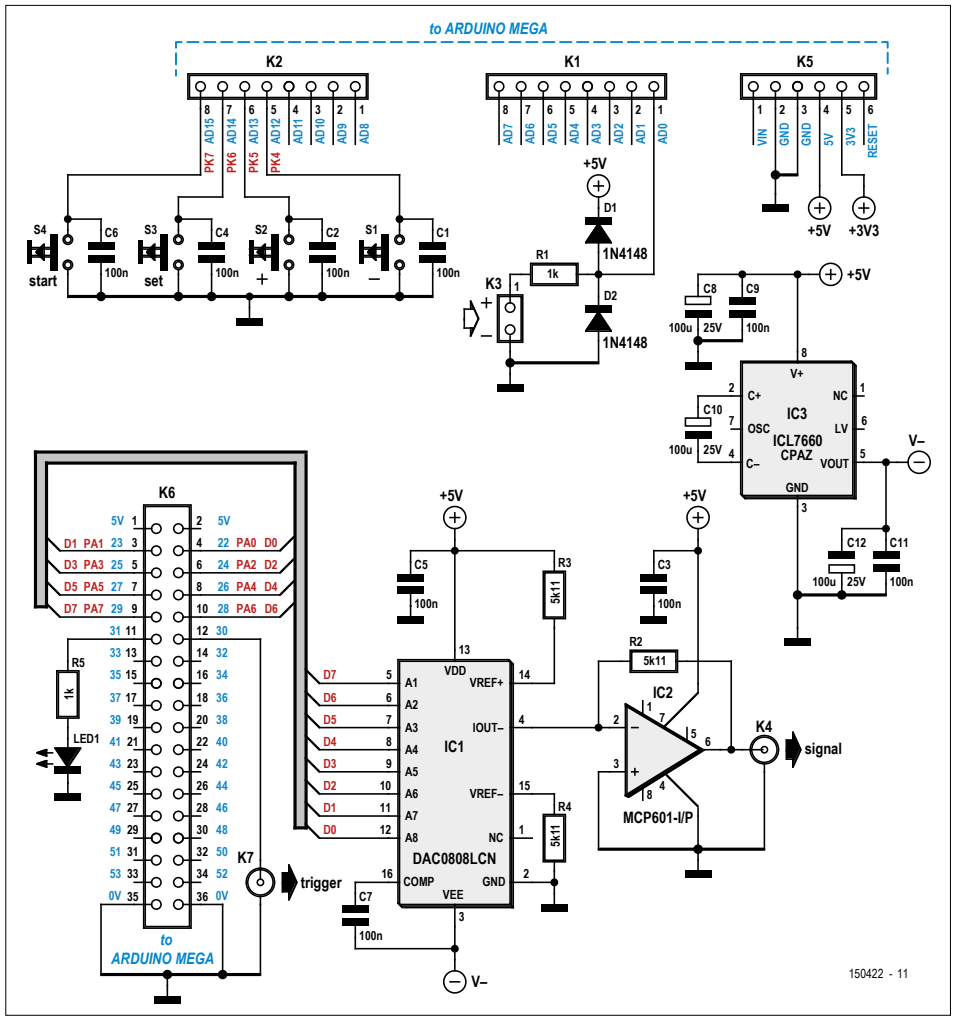


Figure 2. Le schéma du *shield* Arduino se passe de commentaires.

quand on veut le souder. C'est pour cette raison que le montage est très simple, et de plus nous vous proposons un circuit imprimé (**fig. 1**, voir liste des composants). Il suffit de charger le micrologiciel dans la carte Arduino.

Le fichier du croquis ainsi que celui du circuit imprimé sont disponibles sur le site d'Elektor, voir [1].

Le schéma de la **figure 2** montre comment ce CN/A, à savoir IC1, est commandé par le port A (**K6**) de l'ATmega. IC2 est le tampon de sortie. Nous avons opté pour un AOP de type *rail-to-rail* de Microchip. La carte Arduino est généralement alimentée en 5 V, tension également disponible pour le *shield*. La tension négative nécessaire pour le CN/A et l'AOP, mais de faible intensité, est fournie par le convertisseur de tension IC3. Comme souvent, le schéma du *schield* Arduino est facile à comprendre. Pour l'utilisation, il faut quatre boutons ainsi qu'une LED. La tension à mesurer est appliquée sur **K3** du *shield*. Afin d'éviter tout dommage sur l'entrée analogique, et donc sur le µC de la carte Arduino, à cause d'une inversion de polarité ou d'une tension d'entrée trop élevée, nous avons ajouté la résistance de protection R1 et deux diodes qui limitent la tension sur les broches du µC à des valeurs inoffensives.

Comme nous utilisons le CA/N du µC, la plage de mesure est fixée de 0 à 5 V. C'est parfait puisqu'un accu Li-ion par ex. présente une tension maximale typique de 4,6 V. Si l'on veut mesurer la tension d'un bloc d'accus ou d'une batterie de 12 V, il faut insérer un diviseur de tension. Des résistances de l'ordre de 100 kΩ conviennent car l'entrée analogique est à haute impédance, autour de 100 MΩ.

Comme déjà mentionné, le *shield* est alimenté directement par la carte Arduino. La carte est elle-même alimentée en 5 V via l'USB (par ex. directement par un oscilloscope numérique) ou avec une tension comprise entre 7,5 V et 12 V avec

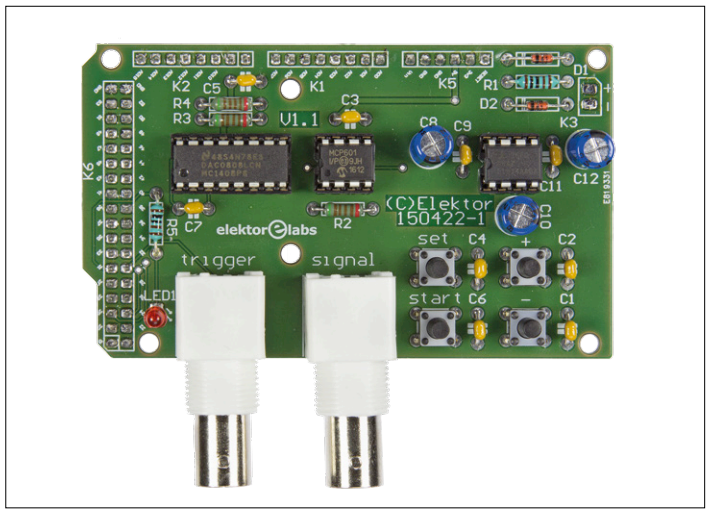


Figure 3. Le circuit imprimé avec ses composants, prêt à être connecté à une carte Arduino.

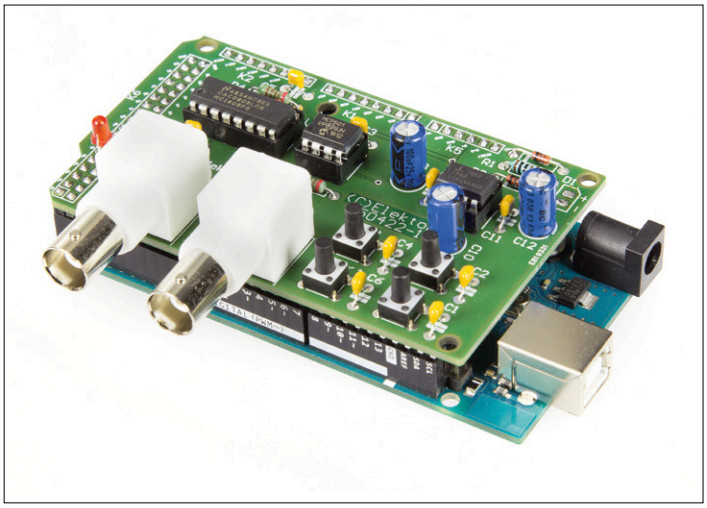


Figure 4. Le sandwich *shield* + Arduino donne un pisteur de tension compact.

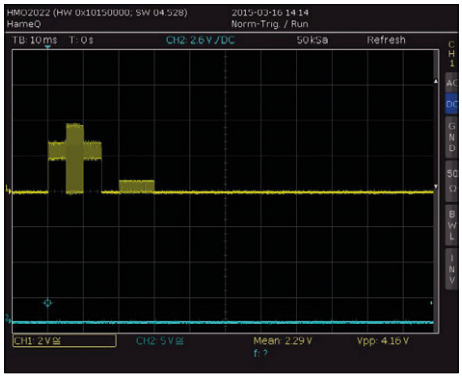


Figure 5. Affichage du réglage de l'option **time**. On peut voir un « t » plus une marche d'escalier, ce qui signifie 1 min/div.

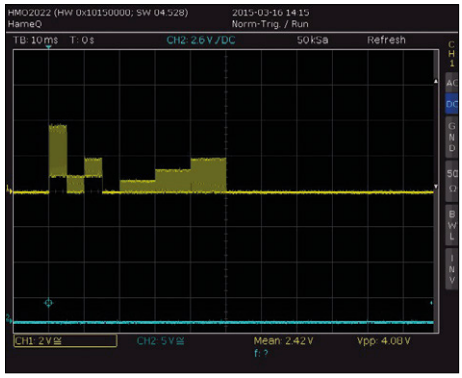


Figure 6. Réglage de l'option **level**. On peut voir un « l » plus trois marches d'escalier, ce qui a pour conséquence un offset de 3 V.

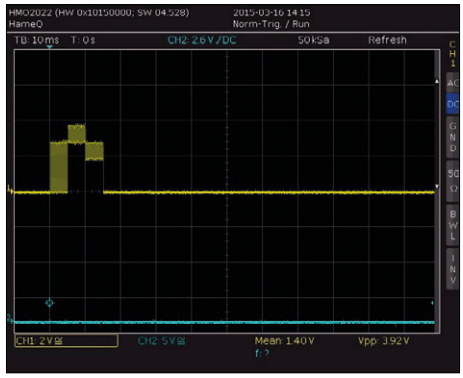


Figure 7. Réglage de l'option **roll**. On peut voir un « r » sans marche, il n'y aura pas de défilement.

un bloc d'alimentation classique, compatible Arduino.
Le *shield* est connecté à l'oscilloscope par l'intermédiaire de deux câbles, l'un pour le signal analogique (**K4**) et l'autre pour le signal de déclenchement (**K7**). Des connecteurs BNC sont prévus pour les deux sorties. Il est donc nécessaire de disposer de deux câbles coaxiaux avec des connecteurs BNC aux deux extrémités (couramment disponibles dans le commerce) pour relier le *shield* à l'oscilloscope. La **figure 3** montre le *shield* assemblé. La **figure 4** montre comment la carte Arduino et le *shield* connecté forment un bloc compact.

Mise en service et réglage

Comme le microcontrôleur mémorise les mesures dans sa propre mémoire, il n'est pas nécessaire que l'oscilloscope reste allumé pendant les éventuelles longues heures de mesure. Par contre ceci n'est pas valable lors du réglage des diverses options, sinon vous ne verrez rien lors de la manipulation des boutons. Supposons que le matériel est prêt, que le croquis est chargé dans la carte Arduino et que le *shield* est connecté, nous pouvons connecter le pisteur de tension à l'oscilloscope avec les câbles BNC (données et déclenchement) et mettre sous tension. Nous avons quatre boutons sur le *shield*. L'un est identifié (ou va l'être) avec le signe « – » et l'autre avec le signe « **+** ». Ces deux boutons permettent de régler le niveau d'une option sélectionnée avec S3 (**set**). Une courte pression sur S4 (**start**) démarre ou arrête une courbe de mesures ; une pression prolongée (>2 s) efface la courbe en cours. Jusque-là tout est simple. Mais comment voir ce qui est réglé ? Pour cela le CN/A est piloté par l'Arduino de telle manière que l'écran de l'oscilloscope affiche la première lettre de l'option choisie ainsi qu'un « escalier » qui indique le niveau réglé.
Nous réglons le niveau de déclenchement de l'oscilloscope dans la gamme de 1 à 4 V, ou nous actionnons le déclenchement automatique. La base de temps devrait être réglée sur 10 ms/div et l'amplification verticale sur 2 V/div, sans oublier de vérifier que les pointes de touche ont un facteur de division de 1:1. Si tout est bien paramétré, nous devrions voir apparaître un petit « **t** » pour la base de temps (***t**imebase*) et une marche d'escalier. La largeur de la marche doit correspondre à une division. Si ce n'est pas le cas, il faut vérifier le réglage de la base de temps de l'oscilloscope. Si l'écran de l'oscilloscope ne présente pas douze divisions horizontales, il faut adapter le pisteur comme suit : appuyez plusieurs fois sur le bouton SET jusqu'à ce qu'apparaisse un petit « **n** » (**n**ombre de divisions). Avec les touches « **+** » et « **–** », adaptez le pisteur à l'écran de l'oscilloscope dans la gamme de 10 à 16 divisions (voir la suite du texte). Après ce réglage, il faut appuyer sur le bouton SET jusqu'à ce que le « **t** » apparaisse de nouveau. Nous pouvons poursuivre le paramétrage de la mesure à l'aide des autres menus. La LED sur le *shield* ainsi que celle de l'Arduino restent éteintes pendant le paramétrage. Elles clignotent pendant l'enregistrement, puis restent allumées quand l'enregistrement est terminé.
Il reste les cinq options de paramétrage avec leurs différents niveaux :

Tableau 1 : il y a exactement sept niveaux pour la base de temps (voir **fig. 5**). La durée correspond à un oscilloscope avec un écran à douze divisions. Comme l'escalier sur l'écran n'a qu'une marche, la définition horizontale est de 1 min/div.

Tableau 2 : l'offset est configurable sur cinq niveaux (voir **fig. 6**) à partir desquels la tension est enregistrée. Attention : comme le niveau paramétré à l'écran est de 3 V, il ne reste plus qu'une marge de 2 V pour rester dans la gamme de mesure de 5 V du µC (vérifiez aussi le paramétrage vertical).

Tableau 1. Base de temps : time

niveau	durée/div	durée
1	1 min	12 min
2	2 min	24 min
3	5 min	1 h
4	10 min	2 h
5	20 min	4 h
6	30 min	6 h
7	1 h	12 h

Tableau 2. Choix du niveau : level

niveau	offset
0	0 V
1	1 V
2	2 V
3	3 V
4	4 V

Tableau 3. Affichage : roll

niveau	défilement
0	non
1	oui

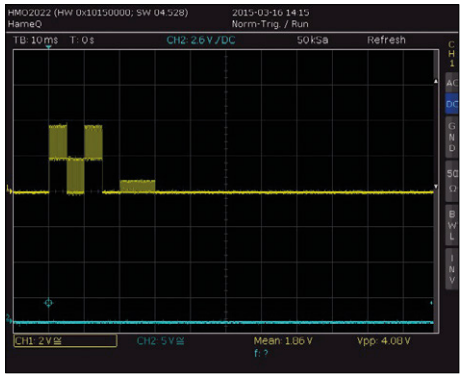


Figure 8. Lors du réglage de l'option **y-Skala**, il faut aussi se préoccuper de l'offset réglé. On voit ici un « y » avec une marche, ce qui correspond à un intervalle de mesure de 2,5 V.

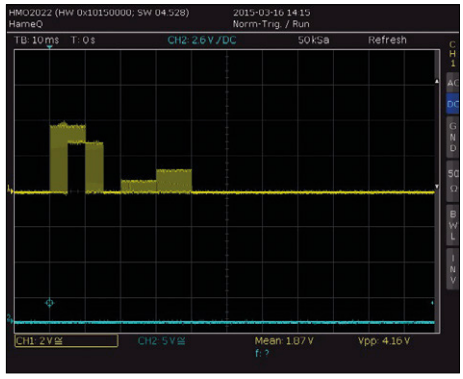


Figure 9. Réglage de l'échelle horizontale de l'oscilloscope avec l'option **number**. On peut voir un « n » avec deux marches, ce qui correspond à un écran avec douze divisions.

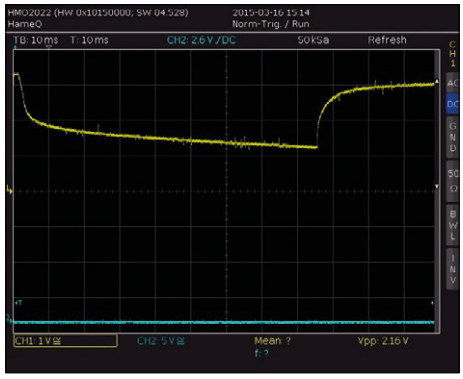


Figure 10. Affichage à titre d'**exemple** de la courbe de décharge d'une cellule Li-Ion. Durée représentée = 12 min.

Tableau 3 : comme sur la **figure 7**, si le réglage est 0, l'enregistrement des données s'interrompt lorsque l'écran est plein. Par contre avec le niveau 1, il y a un défilement de l'image : lorsqu'une nouvelle donnée apparaît à droite de l'écran, la donnée la plus à gauche disparaît.

Tableau 4 : le réglage vertical est un peu plus compliqué. Sur la **figure 8**, le niveau réglé est 1, ce qui correspond à un gain vertical de deux. Comme la variation maximale enregistrable est maintenant de 2,5 V, la valeur maximale de l'offset (tableau 2) est de 2 V si on veut éviter le *clipping* (dépassement de l'intervalle de mesure). Avec l'offset est réglé sur 2 V, l'intervalle de mesure est de 2 à 4,5 V. Si l'offset est égal à 0, l'intervalle de mesure n'est pas décalé et il s'étend par rapport à la masse de 0 à 2,5 V.

Tableau 5 : les écrans d'oscilloscope ne présentent pas tous les mêmes divisions horizontales. Afin d'adapter la base de temps du pisteur de tension à l'oscilloscope connecté, on indique ici le nombre de divisions. Ceci a aussi des conséquences sur la durée d'enregistrement possible. Sur la **figure 9**, le niveau 2 correspond au nombre habituel de divisions, soit 12. Le niveau le plus haut correspond à 16 divisions ce qui permet une durée maximale d'enregistrement de 16 h.

Exemple

Sur la **figure 10** est représentée la courbe de décharge d'une cellule (usagée) d'un accu Li-Ion. Tout d'abord a lieu la décharge ; elle est stoppée quand la tension descend sous le seuil de 3,3 V. On voit bien comment la tension remonte à nouveau pendant la période de repos. On a utilisé ici un oscilloscope avec douze divisions. Le paramétrage du pisteur de tension était de 1 min/div, ce qui correspond à une durée totale de douze minutes. L'offset était fixé à 3 V et le réglage vertical à ¼. Comme le gain vertical sur l'oscilloscope est de 1 V/div, nous avons sur cette image une résolution de 0,25 V/div.

Remarques finales

La réalisation d'un *shield* Arduino avec uniquement des composants traversants est certainement un exercice facile. Une fois le *shield* installé sur une carte Arduino bon marché, nous obtenons un pisteur de tension tout aussi bon marché et très pratique pour enregistrer et afficher à l'aide d'un oscilloscope la courbe de tensions qui varient lentement. La manipulation est simple puisque l'écran de l'oscilloscope permet d'afficher le réglage des options. Le pisteur de tension a de la mémoire : tous les paramètres sont sauvegardés dans l'EEPROM du microcontrôleur de l'Arduino et sont donc disponibles à la mise sous tension suivante. ⏮

(150422 – version française : Patrick Bechler)

Tableau 3. Affichage : y-scale

niveau	Y osc.	gamme	niveau max.
0	1/4	1,25 V	4 V
1	1/2	2,50 V	2 V
2	3/4	3,75 V	1 V
3	1/1	5,00 V	

Tableau 5. Divisions de l'écran : number

niveau	divisions
0	10
1	11
2	12
3	13
4	14
5	15
6	16

Liens

[1] www.elektormagazine.fr/150442

[2] www.elektormagazine.fr/labs/voltage-tracker-for-oscilloscope

À propos de l'auteur :

Kurt Schuster travaille comme développeur en matériel et logiciel, spécialisé dans l'assembleur pour AVR. Les commentaires et questions peuvent être envoyés à : qrt@qland.de

Un événement oublié ?

Vous organisez une conférence, un salon… ou bien vous participez à un séminaire ou tout autre événement qui aurait sa place ici, partagez cette information avec tous les lecteurs. Envoyez-nous tous les détails à **redaction@elektor.fr**.

janvier 2017

◇ Expolangues
20 au 21/01/2017 – Paris
www.expolangues.fr

◇ Salon du travail & mobilité professionnelle
20 au 21/01/2017 – Paris
www.salondutravail.fr

◇ Semaine du son
23/01 au 05/02/2017 - Paris et partout en France
www.lasemaineduson.org

◇ Congrès ATEC ITS France 2017
exposition, débats et conférences sur les systèmes de transports intelligents
24 au 25/01/2017 – Paris
www.congres-atecitsfrance.fr

◇ SEPEM industries Nord
salon des services, équipements, process et maintenance
24 au 26/01/2017 – Douai
www.sepem-industries.com

◇ 9e forum international de la cybersécurité
25 au 26/01/2017 – Lille
www.forum-fic.com

◇ Biogaz Europe
25 au 26/01/2017 – Rennes
www.biogaz-europe.com

◇ Ecohome & Vivez Nature
27 au 30/01/2017 – Paris
www.vivez-nature.com

◇ Salon de la Radio
29 au 31/01/2017 – Paris
www.salondelaradio.com

◇ C!PRINT
salon international de la communication visuelle
31/01 au 02/02/2017 – Lyon
salon-cprint.com

février 2017

◇ Microtech
rendez-vous des techniques de précision et des micro/nano techniques
01/02/2017 –Lyon
www.microtech.events

◇ Salon des entrepreneurs
01 au 02/02/2017 – Paris
www.salondesentrepreneurs.com

◇ Aerotech (innovation aéronautique)
02/02/2017 – Lyon
www.aerotech.events

◇ Mondial des métiers
02 au 05/02/2017 - Lyon
www.mondial-metiers.com

◇ Euromaritime & Eurowaterways
salon européen de la croissance bleue
03 au 05/02/2017 – Paris
www.euromaritime.fr

◇ Retromobile
08 au 12/02/2017 – Paris
www.retromobile.fr

◇ Japan Expo Sud
rendez-vous de la culture manga et japonaise
14 au 26/02/2017 – Marseille
www.japan-expo-sud.com/fr

◇ International Railway Summit
Sommet international de l'industrie du transport ferroviaire
15 au 17/02/2017 – Paris
www.irits.org

◇ Tech Innov concours de jeunes pousses
23/02/2017 – Paris
www.techinnov.events

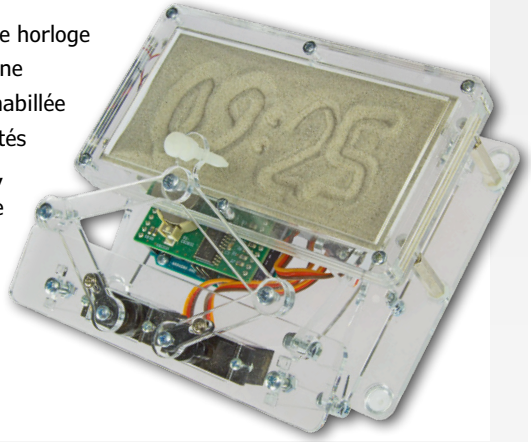
bienvenue dans votre e-choppe

Elektor recommande

Horloge de sable Arduino : l'heure écrite sur le sable

Ce fascinant gadget construit autour d'une carte Arduino Uno donne l'heure en traçant les chiffres dans le sable comme on le fait avec le doigt sur la plage. Les afficheurs à 7 segments et les aiguilles habituelles d'une horloge sont remplacés par un bac de sable, un stylet actionné par un pantographe et deux petits moteurs. Ce sont leurs vibrations qui toutes les minutes effacent en quelques secondes les chiffres de l'heure précédente.

À l'origine, cette horloge est une création de nos amis du magazine allemand *Make*, que le labo d'Elektor a habillée de plexi transparent. Les servos sont dotés d'une transmission à pièces métalliques, lesquelles ont peu de jeu et donnent une meilleure précision.



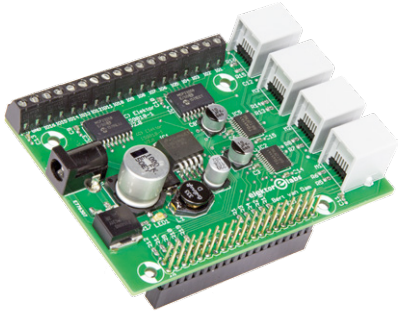
Retrouvez l'article consacré à cette horloge dans ce numéro du magazine.

Denis Meyer
labo d'Elektor

www.elektor.fr/horloge-de-sable-arduino

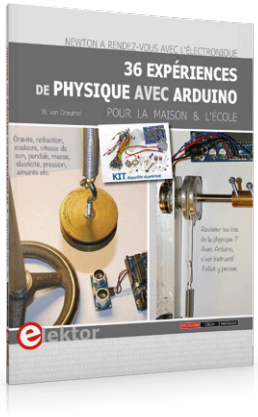
Vos favoris :

1. Carte de commande LEGO pour Raspberry Pi
   www.elektor.fr/lego-rpi-board



2. Détecteur de chauve-souris[PLUS]
   www.elektor.fr/detecteur-chauve-souris
3. 36 Expériences de Physique avec Arduino
   www.elektor.fr/arduino-36-experiences
4. Elektor Uno R4
   www.elektor.fr/elektor-uno-r4
5. BBC micro:bit
   www.elektor.fr/bbc-micro-bit
6. Swiss Pi
   www.elektor.fr/swiss-pi

36 Expériences de Physique avec Arduino



Ce livre n'est pas un manuel de physique : pas d'équations différentielles ni de courbes abstraites, mais des phénomènes physiques de la vie quotidienne. C'est une approche nouvelle et créative des leçons de physique grâce aux techniques modernes de mesure et de traitement des données. L'électronique utilisée (Arduino) est simple. Ajoutez-y le logiciel gratuit *CoolTerm* pour enregistrer les mesures et les retravailler ensuite sous Excel.

Prix (membres) : 22,41 €

www.elektor.fr/arduino-36-experiences

Kit de démarrage

du livre 36 Expériences de Physique avec Arduino.

Ce kit a été spécialement conçu pour réaliser les expériences décrites dans le livre « 36 expériences de physique avec Arduino ». Profitez de cette offre pour acquérir le livre et le kit. La physique rébarbative, c'est fini !

Prix (membres) : 26,96 €

www.elektor.fr/kit-demarrage-36-experiences-arduino

DVD Elektor 2010-2014

Ce DVD-ROM contient tous les numéros d'Elektor des années 2010 à 2014. Elektor propose à ses lecteurs des montages électroniques de conception professionnelle et aisément reproductibles, dans les domaines de l'électronique et de l'informatique appliquées. Il leur apporte également des informations sur l'évolution technologique et les nouveaux produits.

Prix (membres) : 62,10 €

www.elektor.fr/dvd-elektor-2010-2014

Pas un, mais deux !

Voyagez dans le temps
avec les DVD-ROM Elektor des années 1990 à 2009 !

50% DE REMISE

C'est le moment de commencer ou compléter votre schémathèque Elektor.
Ce n'est pas un, mais deux DVD-ROM que nous vous proposons :

- **DVD-ROM 1990-1999**
- **DVD-ROM 2000-2009**

Chaque DVD-ROM contient les centaines d'articles publiés dans le magazine Elektor au fil des ans. Les montages électroniques présentés couvrent les domaines les plus divers : alimentation, audio, vidéo & hi-fi, auto, moto & vélo, expérimentation, hautes fréquences, loisirs, microcontrôleurs… C'est une mine d'idées inépuisable !

Tous les articles sont au format PDF. Un navigateur internet et un lecteur Adobe Reader suffisent pour effectuer des recherches dans tous les articles d'un DVD-ROM.

Prix (membres) : 89,00 €
www.elektor.fr/elektor-dvd-offre-groupee

SmartScope Maker Kit

Le SmartScope n'est pas qu'un excellent oscilloscope USB utilisable avec un ordiphone, une tablette ou un ordinateur, c'est aussi une formidable carte de développement FPGA. Ajoutez quelques connecteurs à la carte et vous voilà prêt à expérimenter, aidé par un logiciel facile à utiliser. Elektor propose un *Maker Kit* avec un Smartscope préconfiguré (version exclusive), deux programmateurs (JTAG et PICKIT3), les câbles nécessaires ainsi que deux sondes analogiques.

Prix (membres) : 269,10 €

www.elektor.fr/smartscope-maker-kit

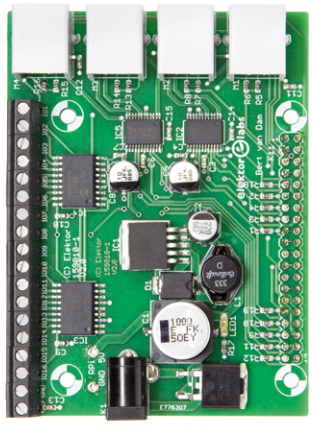
Red Pitaya for Test & Measurement

Red Pitaya est un instrument de mesure à code source ouvert, de la taille d'une carte de crédit. Il peut remplacer nombre d'instruments de mesure de laboratoire coûteux. Avec ce livre (en anglais), les débutants découvriront les principes de l'électronique et leurs applications, tout en expérimentant avec Red Pitaya : théorie, mais aussi utilisation des composants électroniques, le tout accompagné d'expériences amusantes et intéressantes.

Prix (membres) : 31,45 €

www.elektor.fr/red-pitaya-livre

Carte de commande LEGO pour RPi



Cette carte pour Raspberry Pi met à disposition 4 lignes de commande pour des moteurs LEGO EV3 Mindstorms ainsi que 16 lignes d'entrée-sortie avec tampon. En tant que HAT (= *hardware attached on top*), cette carte conforme à des critères bien défi nis est reconnue par le RPi dès sa connexion. La configuration des entrées-sorties polyvalentes et des pilotes est automatique. De quoi se lancer dans de superbes réalisations combinant RPI et LEGO.

Prix (membres) : 40,46 €

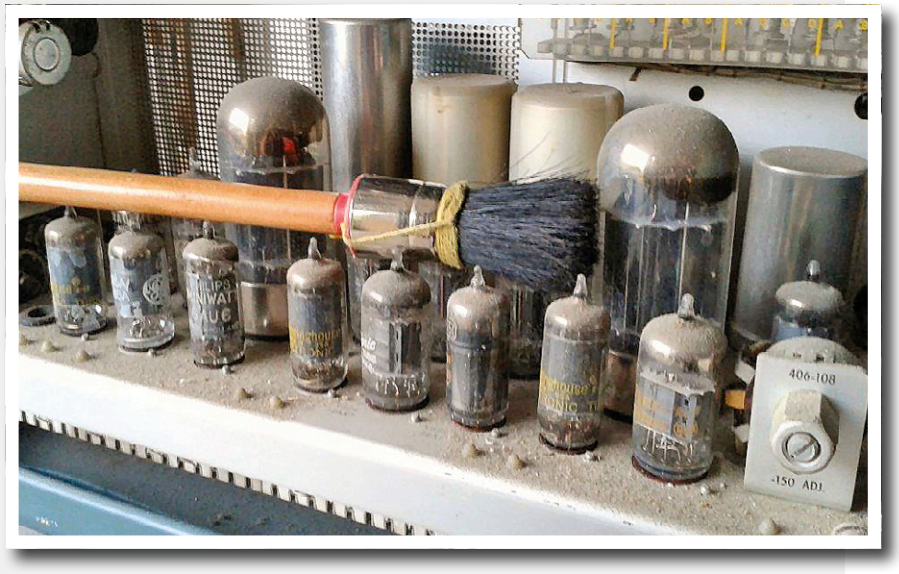
www.elektor.fr/159010-91

bienvenue dans la section PARTAGER

Thijs Beckers (Elektor)

Un de nos collègues du labo a eu un problème assez inhabituel avec son chauffe-eau électrique (soi-disant instantané) : lorsqu'il ouvrait le robinet d'eau chaude, il s'écoulait tout d'abord un peu d'eau brûlante, puis de l'eau froide pendant un certain laps de temps, et enfin de l'eau à la température souhaitée. Comme l'appareil était en location, notre collègue n'a pas hésité et a fait appel à un technicien qualifié.
Ce dernier a eu vite fait de trouver la panne : l'appareil détecte trop tard l'appel d'eau chaude, et ne s'enclenche qu'après coup ; l'eau très chaude initiale provient d'un petit réservoir. Il fallait commander une pièce de rechange, et la solution provisoire (accrochez-vous !) était d'inverser la prise de courant !
Bizarre autant qu'étrange, nous direz-vous : il s'agit après tout de courant alternatif ! Le technicien a bien tenté d'expliquer que cela avait probablement un rapport avec la polarisation d'un composant, mais il n'en savait pas plus. On peut imaginer que la polarisation ait une influence sur la mesure de la température ou du débit, mais de là à admettre que l'inversion d'une prise de courant puisse résoudre ce genre de problème... C'est pour le moins suspect, et nous penchons plutôt pour un circuit électronique « bâclé » (sans doute par souci d'économie).
Nous restons bien entendu intéressés par la cause réelle du phénomène, et n'hésitez pas à nous contacter si vous en savez plus à ce sujet. Nous partagerons volontiers vos trouvailles avec la communauté d'Elektor.

Quelque chose de tout à fait différent maintenant. Il n'est pas rare que Jan Buiting, rédacteur attitré de notre rubrique Rétronique, ait affaire à des tubes ; parfois très anciens et recouverts d'une couche de poussière à couper au couteau, rendant le marquage illisible. Comment nettoyer le tube ? Les encres utilisées n'étaient pas toujours de bonne qualité, et on risque d'enlever plus que de la poussière ! Jan n'est bien entendu pas né de la dernière pluie, et il a partagé son expérience avec nous.



En tous cas, ne frottez JAMAIS sur le marquage avec un chiffon, vous risqueriez de l'effacer irrémédiablement ! Le chiffon sert uniquement à enlever la poussière sur le reste du tube, en principe il n'y a pas de risque. Prenez ensuite un pinceau neuf à poils souples et époussetez PRUDEMMENT l'emplacement du marquage ; il faut être patient, et vous devrez peut-être repasser une cinquantaine de fois avant d'obtenir un résultat. N'utilisez surtout pas une brosse à dents, il faut des poils DOUX, comme ceux d'un pinceau éventail utilisé pour le maquillage. On peut ensuite nettoyer le tube – sauf l'emplacement du marquage ! – avec un chiffon humide, éventuellement avec un peu de solution pour verres de lunettes.
Voilà, c'est fait ! Et ne touchez surtout plus au marquage, ou alors adieu... ⏮

(160252 – version française : Jean-Louis Mehren)

interface de programmation pour USBasp

les lecteurs écrivent aux lecteurs

Encore une solution astucieuse qui facilite la vie des électroniciens.

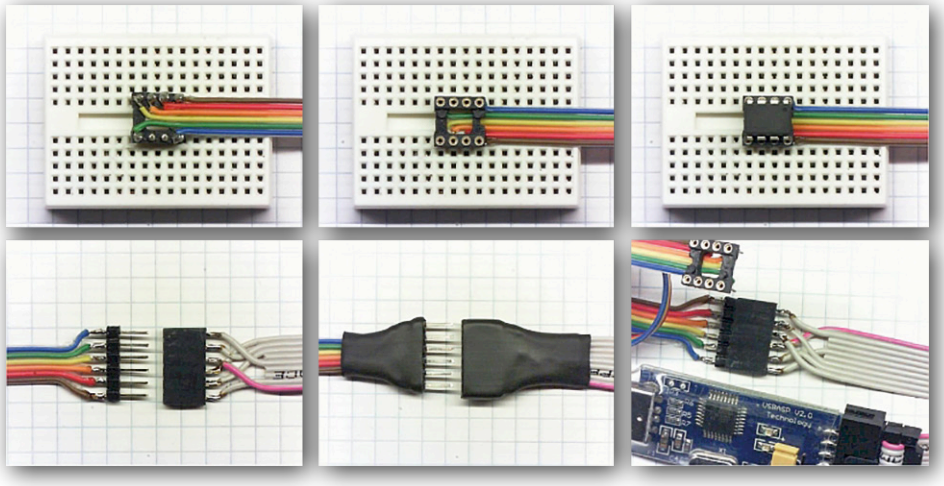
Interface de programmation pour USBasp

Antonello della Pia

Je trouve formidable de partager des idées avec la communauté des électroniciens (tout en gagnant 40 €). L'un de mes microcontrôleurs préférés est l'ATtiny85, que je programme volontiers avec le programmateur simple et bon marché USBasp. Malheureusement, pour un montage sur plaque d'expérimentation, connecter le programmateur aux broches ICSP de l'ATtiny n'est pas facile.

J'ai réalisé un adaptateur pour l'USBasp. J'ai utilisé un support de circuit intégré standard à 8 broches, avec des trous ronds, ainsi qu'une barrette à 6 contacts mâles/femelles ; j'ai aussi récupéré un vieux câble plat à 9 conducteurs (utilisé jadis pour un port série) dont j'ai enlevé trois fils. Les six fils restants conduisent les signaux MISO, MOSI, SCK, RESET, GND et VCC (si vous voulez alimenter le circuit à partir du programmateur).

Ces fils sont soudés sur les broches du support. La barrette est connectée à l'autre extrémité du câble. Pourquoi donc ? Comme on ne peut pas croiser les fils du câble plat du côté du support de circuit intégré, j'ai réalisé un second câble qui met les fils dans l'ordre requis par l'USBasp ; ce câble se termine par une barrette femelle. Les tableaux 1 et 2 donnent les différents brochages (support et USBasp).



La photo montre clairement l'interconnexion de l'ensemble. J'ai même découvert un second usage pour cet adaptateur ! Si l'on réalise une très petite carte, on peut mettre l'adaptateur en sandwich entre le microcontrôleur et son support et ainsi programmer le microcontrôleur in situ.

Je pense que mon idée est facilement transposable à d'autres programmateurs et microcontrôleurs.

(160277 – version française : Helmut Müller)

Tableau 1. Fonctions des broches de l'interface de programmation (ICSP) de l'ATtiny85.

Signal	Broche	Broche	Signal
RESET	1	8	VCC
nc	2	7	SCK
nc	3	6	MISO
GND	4	5	MOSI

Tableau 2. Brochage de l'USBasp.

Signal	Broche	Broche	Signal
MOSI	1	2	VCC
GND	3	4	TXD
RESET	5	6	RXD
SCK	7	8	GND
MISO	9	10	GND

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

broches d'alimentation d'un ampli-op

Arbitraire ou logique ?

Qu'est-ce qui peut clocher lors de l'implantation d'un composant sur un circuit imprimé ? Diverses choses, bien sûr, mais nous sommes tombés sur un cas vraiment curieux : les broches d'alimentation de l'ampli-op n'étaient pas là où on s'y attendait !

Luc Lemmens (labo d'Elektor)

Lors du test d'un premier prototype des « tangibles Tabula », un projet du labo d'Elektor et de l'Université technique de Rhénanie-Westphalie à Aix-la-Chapelle, nous avons été confrontés à un sérieux problème d'alimentation : la sortie du régulateur semblait court-circuitée. Nous avons d'abord vérifié les soudures : en effet, lors du montage de CMS à la main, il n'est pas rare qu'un pont de soudure se forme entre deux broches d'un circuit intégré ou ailleurs. En général, ce n'est pas bien grave, on dessoude et on ressoude. Une inspection visuelle détaillée du circuit imprimé n'a cependant rien révélé, et nous nous sommes alors penchés sur le schéma ; rien de suspect de ce côté-là non plus. Un contrôle supplémentaire de l'implantation des composants et des connexions n'a toujours pas permis de déceler une erreur. Nous avons mesuré la résistance entre la sortie de l'alimentation et la masse, et celle-ci semblait relativement faible ; il devait donc bien y avoir « quelque chose », mais quoi ? Nous avons finalement trouvé deux causes du mauvais fonctionnement, dont une qui nous a vraiment surpris !

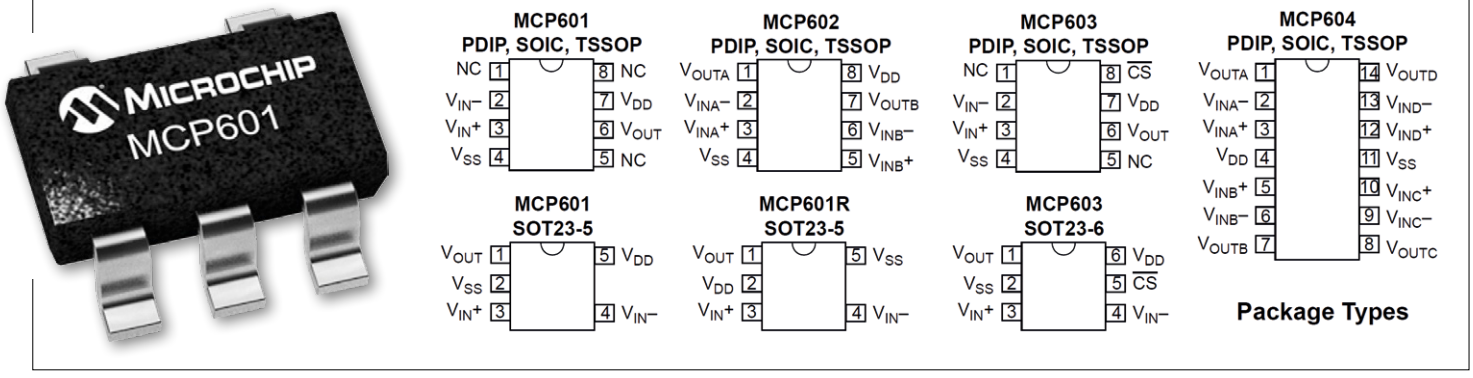
Pour contrôler le circuit, nous avons appliqué une méthode simple qui a fait ses preuves : on retire un composant à la fois, et on mesure à nouveau la résistance en sortie de l'alimentation, jusqu'à trouver le coupable. Il n'y avait heureusement pas beaucoup de composants, et en plus nous avons eu de la chance : le premier circuit intégré que nous avons dessoudé était le composant fautif. Il s'agissait d'un ampli-op MCP601 en boîtier SOT23-5 de chez Microchip ; sa feuille de caractéristiques nous a fourni la réponse : il existe deux brochages différents du circuit pour ce même type de boîtier, 601 et 601-R, avec les alimentations V_{DD} et V_{SS} inversées.

Nous avons déjà rencontré nombre « d'anomalies » en électronique. Par exemple, il faut connaître le fabricant d'un FET BF254 ou BS170 pour être certain du brochage. En ce qui concerne les circuits intégrés, le suffixe après l'identifiant est en général une indication supplémentaire sur les caractéristiques ou le type de boîtier ; qu'un R (sans doute pour *reversed*) signifie que les alimentations sont inversées, ça c'était du jamais vu ! Nous ne voyons pas non plus le pourquoi de la mise sur le marché de ces deux brochages différents, d'autant plus que la feuille de caractéristiques ne dit rien à ce sujet.

Nous avons dit qu'il y avait un second problème avec l'alimentation : le choix du convertisseur de tension abaisseur (*step-down*) était incorrect pour la plage de tension souhaitée. C'est aussi la raison pour laquelle nous n'avions pas vraiment prêté beaucoup d'attention au problème du brochage de l'ampli-op. La rédaction a cependant insisté pour que nous vous en touchions un mot, et nous avons encore une fois revérifié : le circuit intégré existe bien avec deux brochages différents en boîtier SOT23-5.

Comprenne qui pourra !

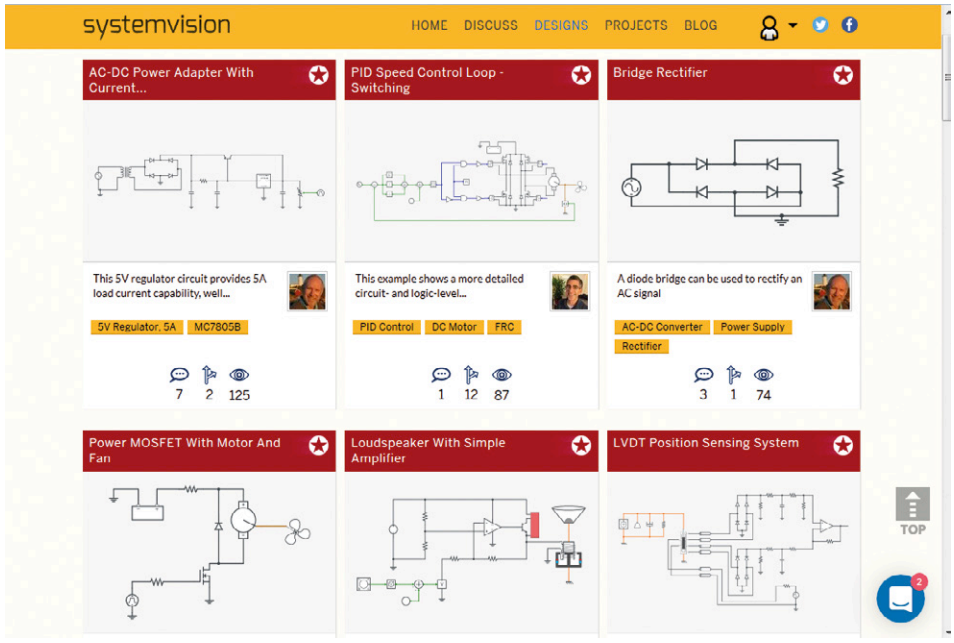
(160257 – version française : Jean-Louis Mehren)



simuler avec SystemVision®

hébergé dans le nuage et gratuit

La simulation des circuits est une étape essentielle pour détecter les erreurs avant la fabrication. À cet effet, il existe beaucoup de programmes qui reposent sur SPICE, mais ils sont d'ordinaire très coûteux. Il y a aussi des versions gratuites, mais généralement sérieusement bridées. Avec son outil *SystemVision® Cloud*, Mentor Graphics propose depuis peu une solution gratuite, disponible uniquement en ligne. Nous nous y sommes intéressés de plus près.



Tam Hanna (Slovaquie)

Un vieil adage prétend que la correction d'une erreur est d'autant plus bénéfique qu'on la découvre tôt. Dans le cas des circuits qui ne comportent pas de composants programmables, il est possible d'effectuer une vérification approfondie sur ordinateur.

Depuis longtemps déjà, Mentor Graphics propose SystemVision, un logiciel de simulation de circuits avec de nombreuses fonctions, mais jusqu'ici toujours payant. Depuis peu, avec SystemVision Cloud (**fig. 1**), il existe une version de base gratuite, dont la différence la plus importante avec la version complète est que les éléments conçus avec elle sont automatiquement partagés avec la communauté.

Premiers pas

Démarrage en douceur : ouvrez la page web **www.systemvision.com** et cliquez sur le bouton *Log In* pour ouvrir une fenêtre d'identification. À l'heure où nous écrivions cet article, SystemVision acceptait, outre les comptes d'utilisateurs ordinaires, les utilisateurs qui voulaient s'identifier avec leurs comptes Facebook, Microsoft, LinkedIn, Twitter ou Google. Toutefois, le processus d'identification varie en fonction du réseau utilisé : pour Twitter par ex. apparaît une fenêtre *pop-up* dans laquelle vous devez autori-

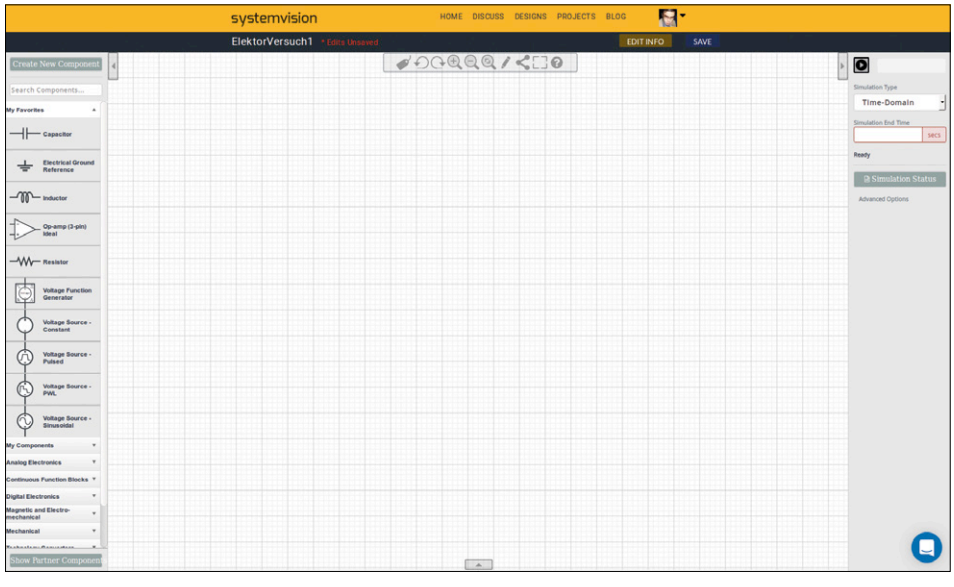


Figure 1. L'espace de travail de SystemVision.

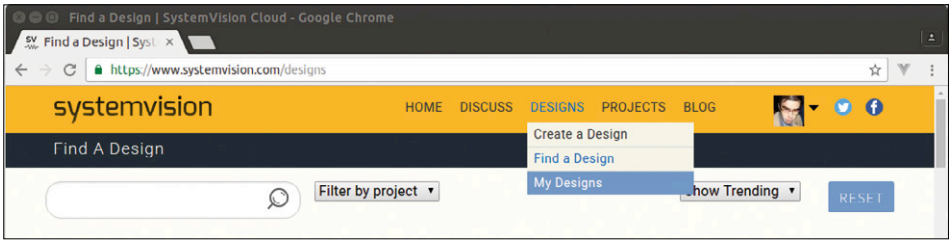


Figure 2. Le menu contextuel de SystemVision est très important.

ser SystemVision à utiliser votre compte.

À cause de l'accent social très marqué de SystemVision, il est quelque peu compliqué de démarrer un nouveau projet. Cliquez sur la barre d'outils jaune en haut de l'écran et choisissez l'option *Designs* → *Create a Design* (**fig. 2**).

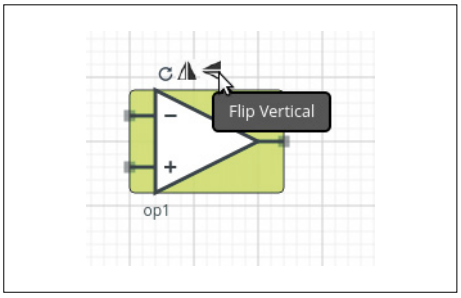


Figure 3. Les composants peuvent être orientés et retournés à volonté.

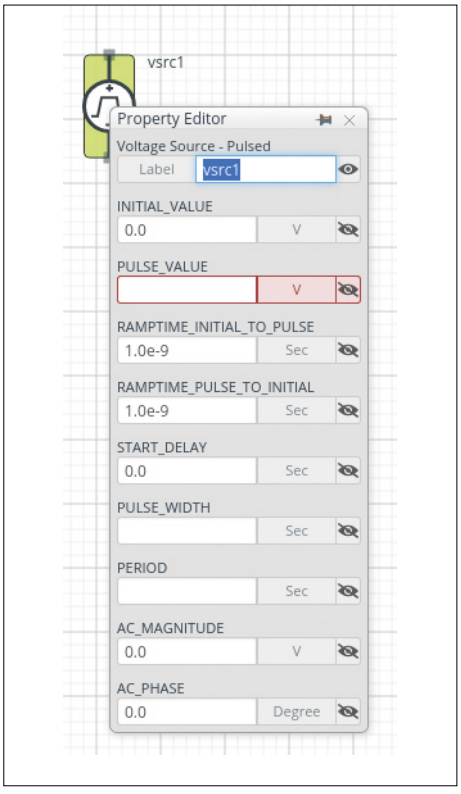


Figure 4. La fenêtre de saisie des paramètres est bien dissimulée.

À l'étape suivante, SystemVision permet de choisir un projet déjà disponible. Si l'on veut commencer avec une planche à dessin totalement vierge, on clique sur *My Designs*. Sur l'écran qui apparaît alors, cliquez sur *Create* pour lancer le processus qui affiche à l'écran l'assistant d'ouverture d'un nouveau projet.
Le chargement du programme prend un peu de temps : selon la configuration du système, cela pourra être plus rapide avec Google Chrome ou Firefox qu'avec le classique Internet Explorer. Si vous créez un nouveau projet, il faudra spécifier un titre et la visibilité. Si l'on renseigne le champ *Visibility* avec la valeur « Show only to me », on crée un projet privé.
L'interface utilisateur proprement dite de SystemVision est simple. À gauche se trouve une liste de composants qu'on peut amener dans l'espace de travail par glisser-déposer. Les options de rotation et de miroir du composant sélectionné sont affichées en gris (**fig. 3**). Utilisez-les pour orienter le composant selon les besoins de votre schéma.

On relie ensuite les bornes des composants par glisser-déposer. Positionnez d'abord le curseur sur la borne de départ, il prend la forme d'une croix. Tirez ensuite une ligne vers la borne d'arrivée, lâchez et la connexion est réalisée.

Pour commencer, prenons un exemple simple, celui d'un circuit RC qui montre la croissance lente et classique de la tension après la mise en marche.
Déposez par glisser-déposer les composants requis dans l'espace de travail et raccordez-les selon la bonne habitude de chez Fritzing et Cie. Il se pose alors la question de savoir comment attribuer des valeurs aux différents composants. L'astuce est de cliquer sur l'étiquette du composant pour accéder à un menu contextuel (**fig. 4**). Renseignez les divers paramètres et validez en cliquant en dehors de la fenêtre, ce qui enregistre les nouvelles valeurs.

Étendons notre circuit comme le montre la **figure 5** pour réaliser les conditions d'une simulation efficace du réseau RC. Dans ce contexte, il est important de savoir que pour SystemVision, les valeurs numériques sont exprimées par défaut en ohms, farads, etc. Ces unités sont modifiables en entrant à la suite d'une valeur numérique une lettre ou combinaison de lettres de la liste suivante :

- f
- p
- n
- u
- m
- K
- Meg
- G
- T

Maintenant, si vous êtes attentif, vous vous demandez pourquoi le circuit est alimenté par une source à impulsions plutôt que par une source continue. Étrange à première vue, ce comportement est dû à un petit bogue : si l'on alimente un circuit RC avec une source continue, on met hors-jeu une partie de l'environnement de simulation et, à la place des courbes caractéristiques, on n'obtient que des valeurs constantes.

Au travail !
Avant le lancement proprement dit de la simulation, il faut encore déclarer un symbole de masse : le moteur de simulation à l'arrière-plan de SystemVision ne fonctionne que lorsqu'on lui a spécifié un potentiel de référence. Dans la boîte à outils *Emulation Toolbar* à droite de l'écran, cliquez ensuite sur le champ *Simulation End Time* et saisissez la valeur 5 s. Cliquez enfin sur la flèche tout en haut pour lancer une simulation de type *domaine temporel*. SystemVision peut également effectuer des simulations de type *domaine fréquentiel*, une fonction que nous n'aborderons pas ici, ne serait-ce que par manque de place.

Une fois la simulation effectuée par l'ordinateur de Mentor Graphics, le programme ouvre une fenêtre de résultat (**fig. 6**) où

sont affichées les diverses valeurs produites au cours de la simulation. Les erreurs éventuelles dans le modèle du circuit apparaissent sous forme de bulles rouges ; les messages d'erreur sont en général faciles à interpréter.
Si vous voulez vous épargner l'examen de la fenêtre de résultats, vous pouvez utiliser l'icône du crayon pour placer un poste d'observation (*watcher*, **figure 7**). Il s'agit d'un élément qui flotte au-dessus du schéma et qui affiche les résultats de la simulation en cours.

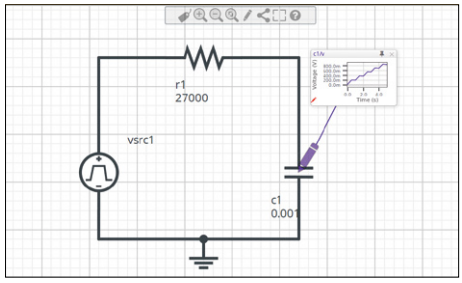


Figure 5. Le circuit RC est prêt pour la simulation.

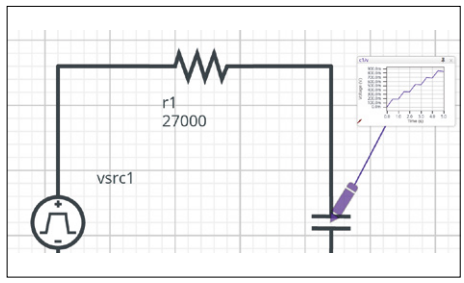


Figure 7. L'icône du crayon permet de positionner un poste d'observation (*watcher*).

Lorsque nous écrivions ces lignes, SystemVision présentait un bogue énervant : si l'on démarre une deuxième simulation, un nouveau fichier de résultats s'ouvre dans la fenêtre de simulation. C'est pourquoi il est conseillé de faire un clic droit sur les éléments désormais inutiles et de les éliminer au moyen des options du menu. Un autre petit souci concerne l'enregistrement des modifications du circuit : SystemVision ne les prend en compte qu'après un clic sur le bouton de sauvegarde situé dans la barre d'outils jaune déjà mentionnée.

Du social et plus

Même si la fonction de simulation de SystemVision est très intéressante, sa fonction la plus importante est la possibilité de partager des projets avec des amis et des collègues. Le partage de notre circuit est activé en cliquant sur le bouton *Share Design*. SystemVision affiche alors un lien (par ex. **http://sysvis.io/smKBn9**) que l'on peut partager avec ses amis et collègues.
La version de base gratuite de SystemVision permet de créer jusqu'à cinq projets privés. Au-delà, la version professionnelle est requise (mais on peut supprimer un projet périmé pour rester sous cette limite, ou mettre un projet en partage). Un clic sur le bouton *Upgrade Account* ne fournit malheureusement qu'une adresse de courriel : on cherche encore les tarifs sur l'internet.

Si l'on est en mal d'inspiration, il y a la rubrique *Find a Design* qui présente les circuits mis en ligne par les autres concepteurs. C'est particulièrement utile si l'on veut en savoir davantage sur un composant particulier ou si l'on cherche des avis pour la conception de simulations efficaces.

Conclusion

À première vue, SystemVision semble un peu compliqué parce qu'il n'est utilisable qu'en ligne, cependant il offre de nombreuses fonctions dont cet article ne donne qu'une petite idée.

> La fonction la plus intéressante est la possibilité de partager des projets avec ses amis et collègues.

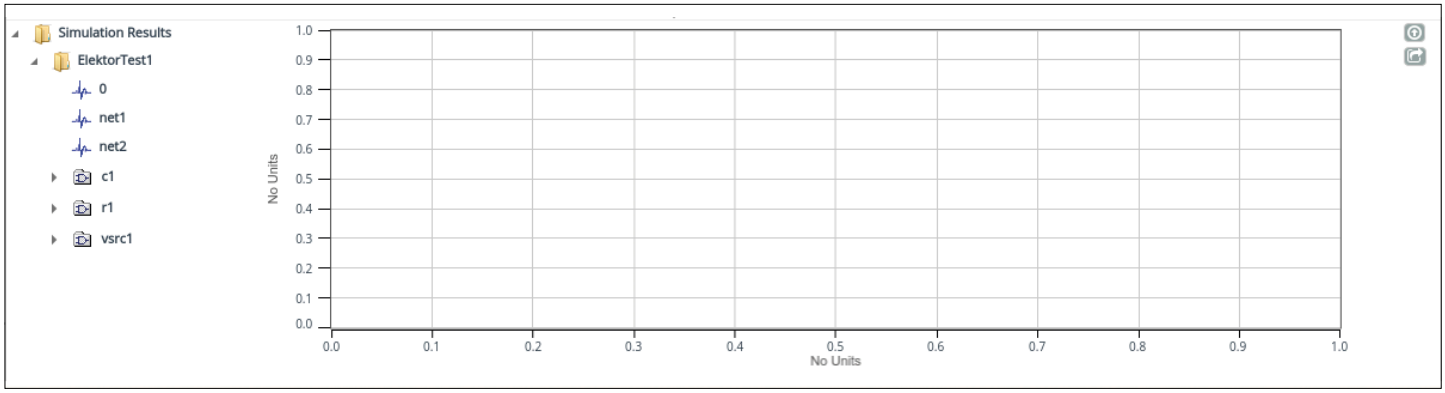


Figure 6. Les résultats de la simulation sont disponibles.

Si vous vous intéressez depuis toujours à la simulation des circuits électroniques, vous devriez essayer SystemVision qui a une longueur d'avance sur PSpice, en particulier ce qui concerne l'ergonomie.

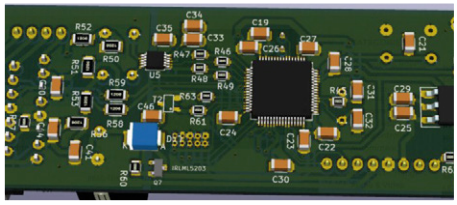
(160203 – version française : Helmut Müller)

bruits de labo...

La nouvelle année commence et la (vieille) liste des résolutions continue de grossir. Jetez les plus anciennes, remplacez-les par des projets concrets – d'électronique par ex. Voici quelques suggestions pour vous lancer.

Construisez un puissant lecteur audio

Vous recherchez un carillon tapageur ou un avertisseur sonore puissant ? La « carte son » présentée ici comprend un amplificateur audio TDA7266 de 2 × 7 W, un convertisseur A/N CS4344, un µC ARM Cortex-M4 STM32F401 et un connecteur de carte micro-SD. L'idée est de jouer différents sons (de haute qualité) enregistrés sur une carte SD avec un volume suffisant pour interrompre le plus profond sommeil.



https://goo.gl/yc1or6

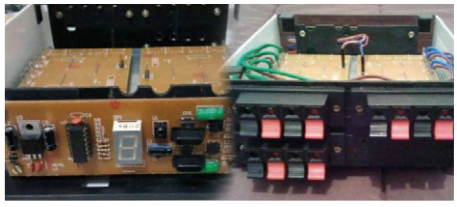
Confiez l'arrosage de votre pelouse à un Raspberry Pi

Un peu énervé par le temps précieux perdu à arroser sa pelouse alors qu'il aurait pu travailler sur ses projets électroniques, l'auteur s'est mis à réfléchir au moyen d'automatiser cette tâche fastidieuse. Un premier prototype de commande temporisée de buses d'arrosage à 12 voies, conçu autour d'un écran graphique tactile intelligent (µC PIC), donnait de bons résultats, mais la commande à distance lui faisait défaut. La seconde version est plus évoluée technologiquement (PIC32, Ethernet, Windows 10 IoT, et Raspberry Pi).

https://goo.gl/JwND75

Boîtier de commutation de haut-parleurs avec télécommande

Il y a des gens qui voudraient écouter la musique en mode *surround 5+1*, mais qui n'ont pas pour autant envie d'abandonner la stéréo. Avoir les deux dans la même pièce nécessiterait huit haut-parleurs, ce qui n'est ni pratique ni indispensable. Comme les deux systèmes ont des haut-parleurs pour les voies gauche et droite, l'idée nous est venue d'y adjoindre un commutateur capable de configurer l'ensemble en 5+1 ou en stéréo (utilisant les HP 5+1).



https://goo.gl/6h6oJos

Analyseur de batterie intelligent Intelli-Cell

Voici un instrument vraiment intelligent et reprogrammable pour recharger vos batteries et évaluer leur état ; sa plaque d'accueil permet de le raccorder aux bornes de toute batterie de téléphone (Apple inclus) ainsi que de tout autre type de batterie, avec une tension comprise entre 1,25 V et 12 V. Cet appareil détecte automatiquement la polarité et mesure la température de la plaque d'accueil et celle du câble externe de recharge. Il est facile à programmer pour toute combinaison de batteries (NiCd, NiMH, Li-Ion, Li-Fe, plomb-acide) et toute capacité avec un courant de 400 à 5000 mA. La plaque d'accueil comprend des contacts coulissants *Pogo* (à ressort) pour connecter les bornes de n'importe quelle batterie, même les plus difficiles à atteindre. Le câble extérieur est équipé de pinces crocodiles et d'une thermistance pour les batteries de plus grande taille ou pour utiliser l'adaptateur Apple. ⏮

(160253 – version française : Yves Georges)

https://goo.gl/ar15x7

du Verobox au Heavy Metal

instruments de labo Elektor des années 80 et 90

Entre 1984 et 1988, Elektor a présenté plusieurs instruments de labo à réaliser soi-même ; ils étaient tous logés dans le même boîtier Verobox, en plastique bicolore et à faces avant et arrière en alu. En 1989 a été lancée une seconde vague d'instruments de mesure habillés du même uniforme, à savoir de solides boîtiers de métal. J'en ai retrouvé quelques exemplaires dans le grenier Rétronique et vous présente la photo de famille « 15 ans de boîtiers ».

Jan Buiting, rédacteur en chef

Retour à 1995 : juste avant de quitter le Royaume-Uni, j'ai eu la chance de faire une visite privée des laboratoires de RadioSpares à Corby, Northants ; c'est en effet le siège social actuel de RS Components. La première version sur CD-ROM du *RS Catalog*, avec une fonction de recherche intégrée (incapable d'ailleurs de trouver un *BC547*), venait de voir le jour. Les membres du labo de RS devaient fournir des données d'ingénierie vérifiées relatives aux composants et aux systèmes, à utiliser dans la *RS Datasheet Collection* en abonnement, tout sur papier bien sûr et dans des classeurs rouge sombre, et avec onglets SVP. Surprise ! Les cinq membres de la section « Composants » du labo possédaient une collection complète d'appareils de test Elektor publiés depuis 1984 sous forme de projets à monter soi-même. Les instruments étaient faciles à identifier grâce à leurs boîtiers Verobox bicolores, gris et blanc cassé ! Les gens de RS aimaient ces appareils qui, une fois calibrés, fonctionnaient tous parfaitement sans avoir rien à envier à « l'équipement hors de prix du siège social ».
En fait il y a eu plusieurs « séries d'appareils de mesure Elektor ». Je décrirai ici deux séries facilement identifiables grâce aux boîtiers choisis par le labo d'Elektor. Vraiment ?

Ingénierie de produit

Si vous avez commencé l'électronique au fond de votre garage et qu'aujourd'hui vous planchez au bureau sur des feuilles de tableur, des budgets et autres joyeusetés, vous vous souvenez de l'émotion éprouvée lorsque votre circuit fonctionnait, la mise en boîtier était secondaire. En effet, de nombreux projets Elektor durant ces années étaient « fin prêts » dès lors que l'autorouteur affichait *zéro erreur* et que le prototype fonctionnait. Aujourd'hui le fait qu'un nombre croissant de projets soient à base de microcontrôleur (µC) ne semble pas inciter les programmeurs à vouloir donner à leurs projets un aspect professionnel. Ils ne s'intéressent pas aux écrous et boulons.

Quel boîtier ?

Retour aux années 80 : une équipe du labo d'Elektor épaulée par les rédacteurs a imaginé de publier une série d'appareils de mesure faits maison, à un prix abordable et de grande qualité ; il était hors de question de ne proposer qu'un circuit imprimé. Comme de vrais *designers* industriels, ils commencèrent par le boîtier et l'ergonomie des commandes, avant d'ouvrir la *RS Components Datasheet Collection* pour y chercher qui un ampli-op qui un connecteur. Pour certains concepteurs, le boîtier prévu constituait un facteur restrictif, mais ils finirent par accepter une taille unique et un « *look* » Elektor qui permettrait de reconnaître les instruments au premier coup d'œil !

Au tout début, Vero

La première série devait être logée dans un boîtier « Verobox » à deux couleurs, de 205 (l) × 137 (p) × 75 (h) mm. À titre d'exemple, j'ai repris le **générateur d'impulsions Elektor** en **figure 1**. Le boîtier (ABS) est composé d'un couvercle blanc cassé et d'un fond de couleur grise. Des plaques d'alu de 1,5 mm constituent les panneaux avant et arrière. Elektor vendait des feuilles sérigraphiées auto-adhésives à coller sur le panneau avant (préalablement percé). Les textes et symboles à utiliser firent l'objet de débats mémorables entre les quatre départements linguistiques au sein d'Elektor, et heureusement, il fut convenu d'utiliser une combinaison de textes (minimalistes) en anglais et de symboles « uniformisés ». Le nom de l'instrument resta un défi, comme le prouve la figure 1 (coin supérieur gauche).
Les fentes sur les petits côtés de la moitié inférieure du Verobox servaient à fixer les circuits imprimés à la verticale (voir

Figure 1. Commençons par l'ABS : générateur d'impulsions, au milieu des 80. Il manque un bouton sur la commande *0.1–1.0 Repetition Time*.

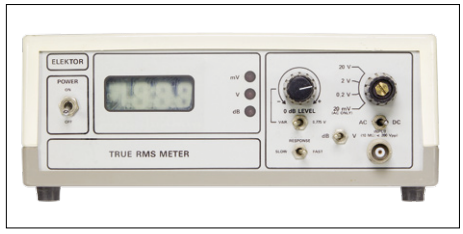


Figure 3. Un oiseau rare, rarement utilisé d'ailleurs ; personne ne se soucie plus guère de l'expression exacte des tensions analogiques : *True RMS Meter*, milieu des années 80.

Figure 5. Passons au boîtier Telet en métal avec le testeur H_{FE} de 1990.

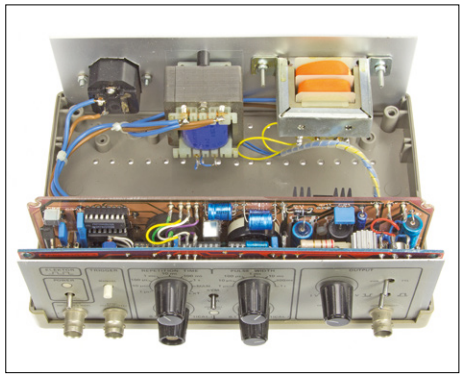


Figure 2. Coup d'œil à l'intérieur du générateur d'impulsions ; notez le montage vertical des cartes et les deux transformateurs placés à l'arrière.

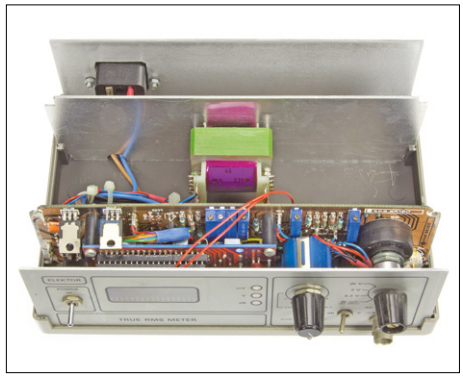


Figure 4. Coup d'œil à l'intérieur du *True RMS Meter*. Notez le bouton de gamme soudé à même la carte.

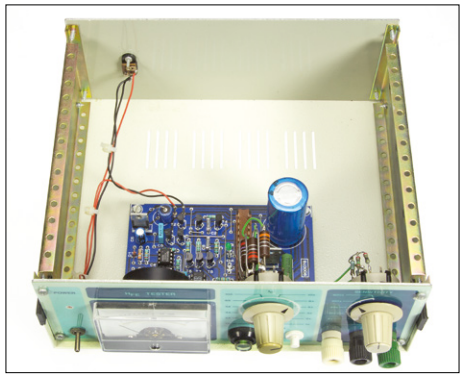


Figure 6. Ce n'est pas l'espace qui manque ici !

fig. 2). Elles permettent aux organes de commande de la face avant (potentiomètres et autres commutateurs) (a) de contribuer à fixer la carte « de commande » au panneau avant et (b) d'être connectés directement aux pistes du circuit plutôt que par le biais de fils volants comme dans bien d'autres projets *amateurs*. Les LED et autres voyants brillaient à travers les découpes translucides de la face avant.

La partie tension secteur + transformateur de puissance était confinée sur la face arrière. À noter : les embases pour vis de la demi-coque inférieure du Verobox restent inutilisées – toutes les platines étaient montées à la verticale. Remarquez ces trous de ventilation soigneusement percés dans les deux demi-coquilles par Jan Visser et/ou M. Feron.

Le ***True RMS Meter* Elektor** est un autre bel exemplaire de la série Test & Mesure Verobox retrouvé dans le grenier Rétronique **(fig. 3**). Désolé pour cet écran LCD devenu flou au fil des ans. Ici, le transformateur d'alimentation a trouvé place sur une plaque d'aluminium séparée, en raison de son poids probablement (**fig. 4**). Il manque malheureusement à ma collection, le summum de la série T & M Verobox Elektor, le **générateur de fonctions à XR2206 Elektor**. Il ressemble aux deux autres instruments sauvés en 2006. On continue de rencontrer ce générateur de fonctions, un véritable classique, sur les sites de ventes aux enchères.

J'ai en mémoire divers autres instruments de la série Verobox, un wattmètre et un multimètre, mais je ne les ai pas.

Ce Verobox coûtait cher d'où de nombreuses plaintes auprès d'Elektor, de la part de Néerlandais en particulier. Certains annonceurs s'enthousiasmèrent pour la série T & M et se lancèrent dans la vente des kits avec leurs infâmes pièces « elektoriennes ». Maplin au Royaume-Uni, Selectronic et Magnetic-France en France, DIL Elektronica aux Pays-Bas, Geist en Allemagne, alors que C-I Electronics fournissait le « reste du monde » par colis postal uniquement. Résultat : un succès incontestable pour Elektor, ces instruments sont devenus légendaires.

Pas mieux que le métal italien

Comme tout fabricant d'instruments T & M, Elektor ne cessa d'innover et de réactualiser sa gamme, assurant du même coup la notoriété de sa marque. Vers 1988, le boîtier Verobox avait fait son temps et l'équipe d'Elektor s'est mise à la recherche d'un nouveau « boîtier standard » pour y loger une nouvelle gamme d'instruments maison. Et toujours le même leitmotiv : Abordable – Qualité – Professionnel

À en croire la rumeur, le distributeur de pièces Texim Electronics, fantastique source de pièces elektoriennes « suggéra aimablement » d'utiliser des boîtiers en métal de Telet (Italie). Contrairement au Verobox, ces boîtiers étaient (a) entièrement métalliques et (b) disponibles en différentes tailles. Ils devinrent le nouveau standard.

La **figure 5** montre un exemple pris au hasard de la « nouvelle » série T & M Elektor, le **testeur H_{FE}**. Ses « mensurations » : 198 (l) × 180 (p) × 80 (h) mm. Notez l'utilisation correcte d'$_{indices}$ sur la face avant - un exploit de Patrick Wielders (graphismes & dessins) ! Remarquez ces deux nuances de vert Arduino (turquoise disent certains) utilisées pour le film de la face avant.

Pour la série d'instruments Telet, les cartes pouvaient être installées à l'horizontale sur des entretoises et les organes de commande étaient fixés directement sur la face avant pour le câblage (**fig. 6**). À noter que dans le cas du testeur H_{FE},

le boîtier est relativement vide en raison de l'absence d'alimentation interne, l'instrument est alimenté par un adaptateur secteur CC (bonne idée les gars, sécurité avant tout !).
Quels autres instruments « Telet » dans la collection Rétronique ?
Un **wattmètre efficace** avec affichage LCD, gammes ×1 et ×10 commutables et embase CA de sortie sur la face avant (**fig. 7**) (Telet 198 × 132 × 80 mm) [2].
L'**énergiemètre multifonction à 80C535** de 1993 dans le plus grand boîtier Telet utilisé par Elektor pour sa série (Telet 297 × 180 x 80 mm) (**fig. 8**) [3]. En avance pour son époque, du moins pour ce qui est du monde amateur, il possédait une interface série V24 (embase sub-D à neuf broches en bas au dos de l'appareil) pour les mesures et l'étalonnage.
Le ***TV Pattern Generator*** que j'utilise d'ailleurs toujours pour vérifier les performances de déclenchement et de temporisation d'oscilloscopes « antiques » (**fig. 9**) (Telet 247 × 180 × 80 mm). Ici, les LED serties dans les trous débordent légèrement du panneau avant. Alimentation secteur.
Pour finir, un ***Inductance Meter*** feignant d'avoir des gammes allant jusqu'au nanohenry (**fig. 10**). Certes, dans la gamme de nH la plus basse, cet instrument est battu à plate couture par notre tout récent **LCR-mètre 0,05%**.

Deux autres projets de style Telet non représentés ici auraient sans doute mérité une mention : une **alimentation échelonnée pour l'amateur** [4] et le **LCR-mètre high tech** [5] que j'ai encore. Il me reste à monter ce projet fantastique de Hans Bonekamp, et à le faire défier notre récent LCR-mètre 0,05% qui coûte *2,5 dB* de plus.

À ne pas oublier

À l'exception du *True RMS Meter*, tous les instruments dont il est question ici fonctionnent et font partie de l'initiative *Retronics Classic Repair* que j'ai initiée chez Elektor. Bien que je me rende compte que l'ensemble de leurs fonctions peut être exécuté en une vingtaine de cycles d'horloge par une carte Red Pitaya ou par un jeune de 15 ans avec un Arduino ou Raspberry Pi doté des cartes d'extension adéquates, ces instruments rétro restent amusants à utiliser, même occasionnellement. Par exemple, le wattmètre est monté de façon permanente dans ma ligne d'alimentation alternative pour me signaler des « monstres » (consommation de plus de 1 000 W).
Contrairement aux projets de technologie enfouie récents, les articles consacrés dans Elektor à ces humbles instruments expliquaient leur assemblage et leur utilisation pratique, mais aussi les principes de mesure et les problèmes d'ingénierie rencontrés (et élégamment solutionnées à la sauce Elektor). Sans même parler des efforts consacrés aux travaux des métaux et à l'ingénierie de produit.
Si vous avez un instrument mis en boîtier Verobox ou Telet construit à partir d'un article Elektor que je n'aurais pas mentionné ici, faites-le-moi savoir, car je sais bien que la liste présentée ici n'est pas exhaustive.

(160197 - version française : Guy Raedersdorf)

ESTD 2004
www.elektor.tv

Retronics is a monthly section covering vintage electronics including legendary Elektor designs.
Contributions, suggestions and requests are welcome; please telegraph editor@elektor.com

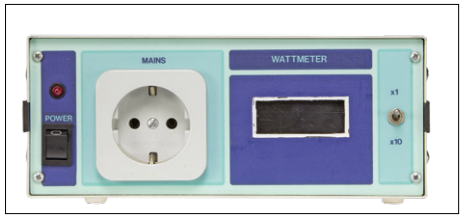


Figure 7. Le wattmètre simple et efficace, voire spartiate de 1991.

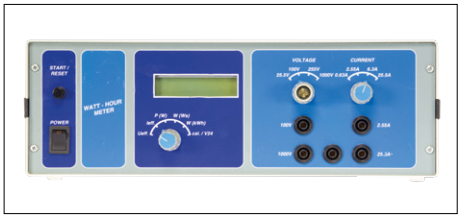


Figure 8. L'un des instruments les plus « encombrants » montés dans un boîtier Telet, l'énergiemètre multifonction à 80C535 de 1993.

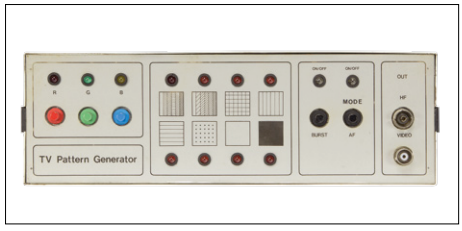


Figure 9. Toujours en service au labo Rétronique : le *TV Pattern Generator*, bien qu'il ne serve plus à tester les TV ou les moniteurs.

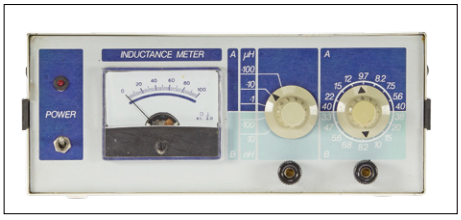


Figure 10. *Inductance Meter*. Je ne vois pas la raison de l'utilisation, au centre, du petit point au lieu du signe « × » devant les gammes.

Liens

Note : les articles originaux de 199x sont disponibles sous forme de fichiers PDF sur le DVD-ROM « COLLECTION 1990-1999 ».

[1] testeur de HFE (909050), Elektor, septembre 1990, www.elektormagazine.fr/magazine/elektor-199009/34773

[2] wattmètre efficace (913071) Elektor, mars 1991, www.elektormagazine.fr/magazine/elektor-199103/34865

[3] énergiemètre multifonction à 80C535, Elektor, février et mars 1993, www.elektormagazine.fr/magazine/elektor-199302/35357 www.elektormagazine.fr/magazine/elektor-199303/35369

[4] alimentation échelonnée pour l'amateur, Elektor, décembre 1995, www.elektormagazine.fr/magazine/elektor-199512/36123

[5] LCR-mètre high tech, Elektor, avril et mai 1997, www.elektormagazine.fr/magazine/elektor-199704/36439, www.elektormagazine.fr/magazine/elektor-199705/36453

Centre de Diffusion

mémoire de ondes longues,

Invitation au voyage : (Indre, 36), plus Saint-Aoustrille, qu'est en ondes courtes ainsi

Pascal Rondane (Tours)

TDF (*TéléDiffusion de France*) exploite des installations de diffusion en ondes courtes à Issoudun depuis 1948. Le site est équipé de plus de 50 antennes de diffusion, dont 12 antennes tournantes, baptisées Alliss (**photo 1**). Ces structures de 80 m de haut sont orientables à volonté vers n'importe quelle zone de diffusion. En moins de deux minutes, les deux cents tonnes de l'antenne se positionnent à 1° près pour enchaîner les différentes séquences de diffusion. Ces antennes portent le nom des grands fleuves qui irriguent eux aussi la planète au-delà des frontières françaises : Danube, Mékong, Gange…
Ce site assure la diffusion de programmes de radio vers des destinations internationales comme l'Afrique, le Moyen-Orient, l'Europe de l'Est, l'Amérique Centrale et l'Amérique du Sud. Les antennes de TDF peuvent desservir, à partir d'Issoudun,

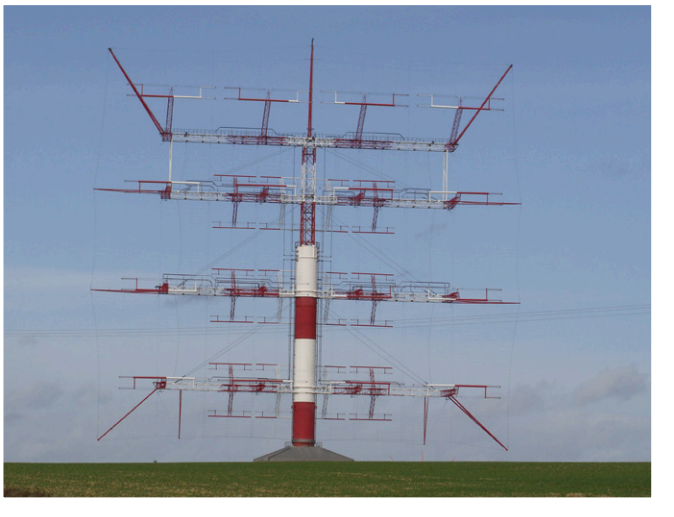


Photo 1. Antenne Alliss : 80 m de haut et 60 m de large.

Photo 2. Anciennes antennes rideau (ou colinéaires).

Historique la Radiophonique

la radiodiffusion en moyennes et courtes

c'est à côté de la ville d'Issoudun précisément sur la commune de implanté le centre de diffusion de TDF que le musée de l'ACHDR.

n'importe quelle zone de la planète.
Les émissions sont réalisées dans la bande dite des « ondes courtes » qui a la faculté de propager à très longue distance les programmes des grandes radios internationales : RFI (Radio France Internationale), VOA (Voice Of America), NHK (radio nationale japonaise), TDA (radio nationale algérienne), KBS (radio de la Corée du Sud) et l'UNR (programme radio de l'ONU).

Photo 3. Salle d'émission du bâtiment B : on peut encore y voir les quatre émetteurs de 120 kW (technologie de 1938) avec les modulateurs et le pupitre de commande. Ils ont été mis en service en 1950 et ils ont été opérationnels jusqu'en 1974.

Photo 4. Salle d'émission du bâtiment B : elle sert de hall d'exposition pour les visites privées, on peut la découvrir sur une des vidéos du blog [4].

Photo 5. Caméra mécanique Barthélémy de 1931 à 30 lignes à disque de Nipkow (16 images/s au format 24 × 30 mm).

Photo 6. Récepteur de télévision EMYRADIO de 1943.

L'*Association du Centre Historique de la Diffusion Radiophonique* (A.C.H.D.R.) a été créée en 1991 par des techniciens de TDF, elle est hébergée dans le bâtiment B qui a été reconverti en 1996 en base de maintenance pour l'ensemble du site TDF.

L'association a pour but de sauvegarder le patrimoine audiovisuel depuis ses origines jusqu'à nos jours et de le faire découvrir au public à travers des expositions temporaires, des visites du centre et des conférences en milieu scolaire ainsi qu'un site internet.

Le matériel provient de l'ex-ORTF, de TDF, de dons de particuliers et de collections privées. Les expositions ont lieu dans l'ancienne salle d'émission en ondes courtes (**photos 3 et 4**)

Photo 7. Caméra tourelle THT 620 développée par la société française Thomson-Houston en 1954.

Photo 8. Pupitre de mixage.

où l'on peut voir encore tous les équipements (oscillateurs, amplificateurs à haute fréquence, amplificateurs à basse fréquence, pupitres de commande et de contrôle).
Décrire l'ensemble de cette fabuleuse collection est impossible dans le cadre de cet article…

J'ai eu la chance d'effectuer une très longue visite de ce musée. Vous trouverez ici quelques photos sur les sujets qui m'ont passionné :
L'espace dédié à la télévision (**photos 5, 6 et 7**) remonte jusqu'à la télévision mécanique, lancée en France en 1935. Il comporte des maquettes et de très beaux récepteurs TV d'époque ainsi qu'une collection de caméras de télévision avec notamment un ensemble fonctionnel de magnétoscopes à bande.
Il s'intéresse aussi à la télévision numérique.
Le studio grandes ondes de l'émetteur d'Allouis a également été reconstitué et est complètement opérationnel. On y découvre deux magnétophones à bandes et deux sélecteurs de 45 tours (l'ancêtre du lecteur MP3…), télécommandés à distance (**photo 9**).
Le musée a une imposante collection de tubes d'émission et d'appareils de mesure (**photos 10 et 11**).
Les tubes de puissance et leur système de neutrodynage (**photos 12 et 13**) de la partie émission du bâtiment B m'ont particulièrement impressionné.

J'espère que vous aurez comme moi l'occasion de visiter ce musée. Attention : comme le musée se trouve sur un site sécurisé de TDF, la visite à titre exceptionnel est soumise à une demande préalable auprès de l'association par courriel (contact.achdr@laposte.net).
La visite dépend aussi de la disponibilité des membres.
Vous pouvez également rejoindre et soutenir l'association (cotisation de 20 €/an).

J'adresse mes remerciements à MM. Mesquita et Fromont de l'ACHDR, Mmes Frugier et Havard de TDF ainsi qu'à M. Bastian Bouchardon.
Sources pour cet article : site ACHDR, TDF.
Photos personnelles de l'auteur.
(160314)

Liens

[1] Site de l'association : http://achdr.over-blog.com/
[2] Page Facebook de l'association : www.facebook.com/ACHDR-Association-du-Centre-Historique-de-la-Diffusion-Radiophonique-104194059618065/
[3] Émetteur d'Issoudun : https://fr.wikipedia.org/wiki/%C3%89metteur_d%27Issoudun
[4] Vidéo : http://achdr.over-blog.com/2016/10/le-centre-tdf-de-saint-aoustrille.html

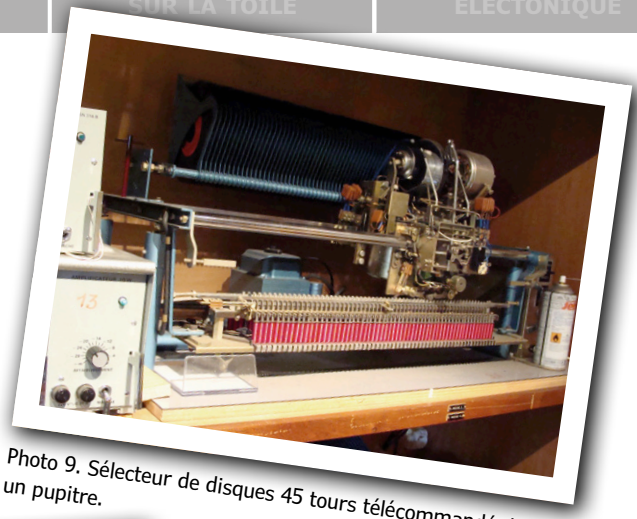


Photo 9. Sélecteur de disques 45 tours télécommandé depuis un pupitre.

Photo 10. Redresseur à vapeur de mercure hexaphasé, utilisé pour les redresseurs de polarité de grille (500 V) des tubes de puissance des émetteurs de 120 W.

Photo 11. Appareils de mesure, au milieu un oscilloscope analyseur de trame TV.

Photo 12. Lampe d'émission du bâtiment B, étage final de puissance de l'émetteur de 120 kW équipé de deux lampes SFR3055 (les plaques que l'on distingue ne sont pas les condensateurs d'accord de plaque, mais les condensateurs de neutrodynage).

Photo 13. Partie émission du bâtiment B. En 1972 l'ORTF inaugure ses émetteurs de 500 kW, au nombre de huit, associés à un champ d'antennes en Y. Les champs d'antennes sont disposés en forme d'Y afin d'obtenir une diffusion mondiale.

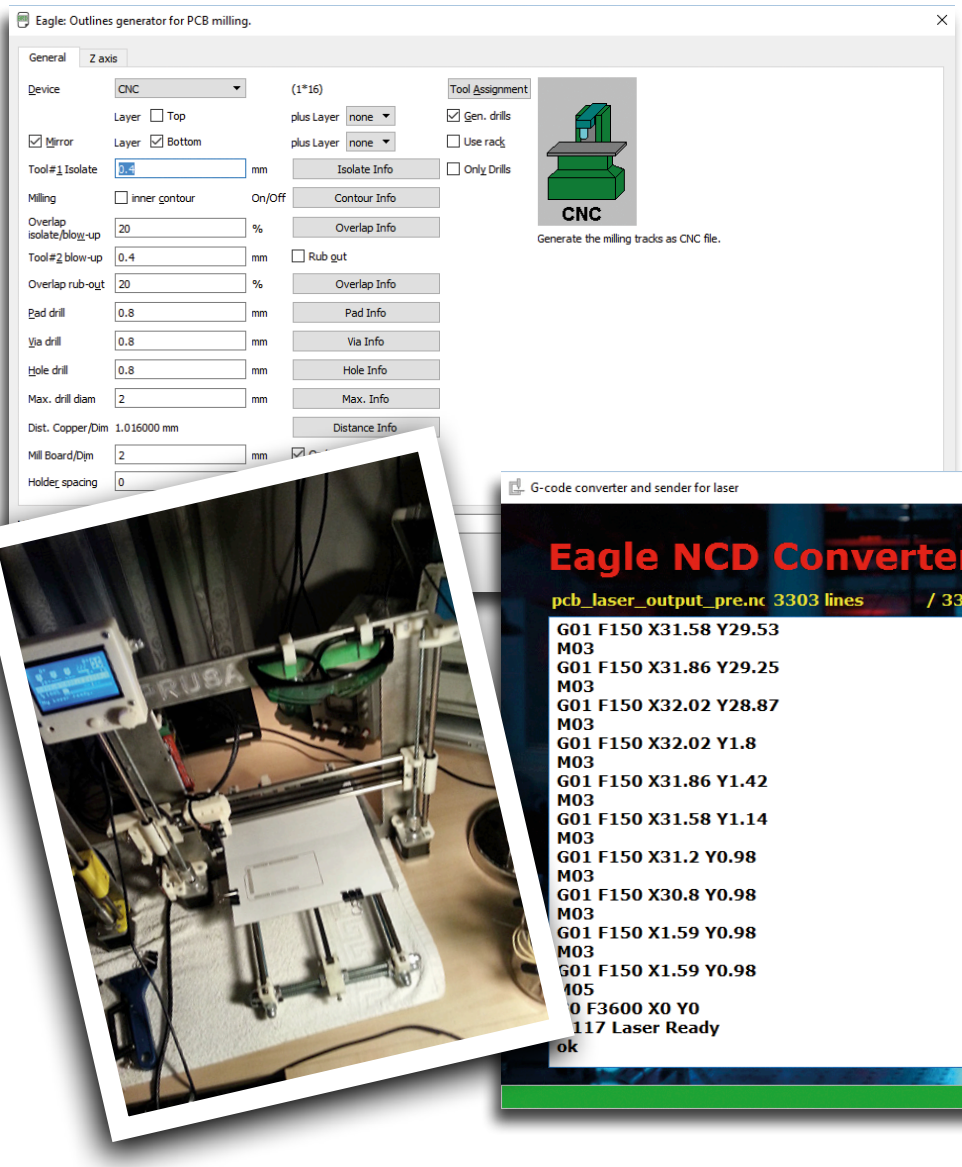
circuits imprimés faits maison

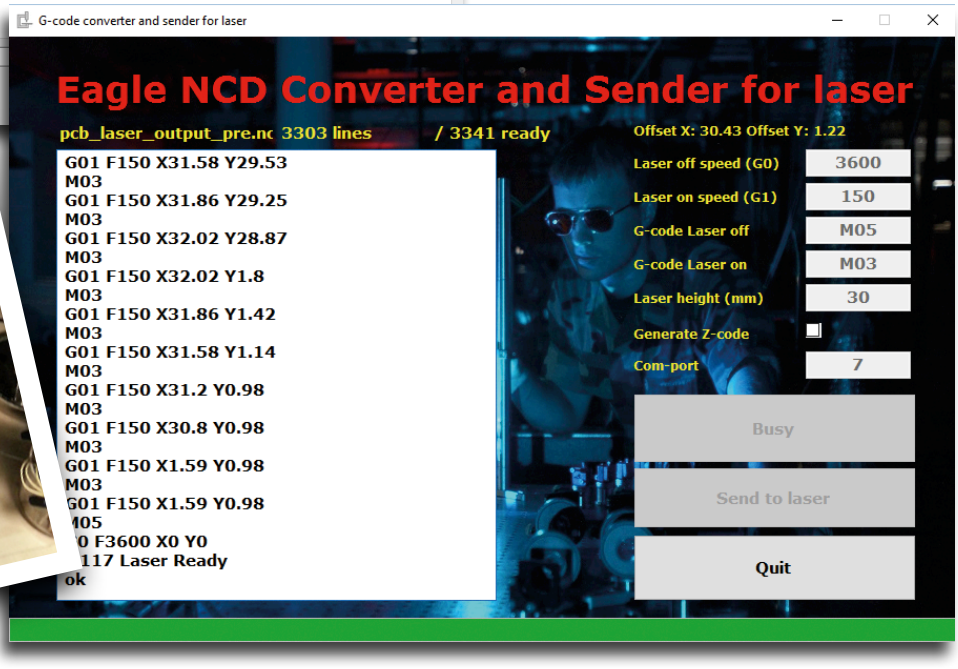
gravure avec un laser à UV

Thijs Beckers (Elektor)

Une fois que vous avez dessiné un schéma, plusieurs solutions s'offrent à vous pour « graver » votre circuit imprimé. L'une d'entre elles, rarement évoquée, est l'exposition d'une plaque photosensible à un laser UV. Sur leur blog [1] (en néerlandais), Ben Zijlstra, Edwin van den Oetelaar (PA2LVD) et Theo Kleijn (PA0KN) font part de leurs résultats.

En fait, c'est loin d'être idiot : plutôt que d'imprimer un typon, de le placer sur une plaque photosensible et d'exposer celle-ci à des rayons UV dans une insoleuse, on pourrait aussi « graver » le circuit à l'aide d'un laser UV (de puissance suffisante) intégré à une imprimante (3D) ou un traceur. Pour étayer leur théorie, Ben, Edwin et Theo ont utilisé un graveur laser.
Première étape : il faut un dessin de circuit imprimé. Pour les essais, ils ont choisi un *shield* pour Arduino Mega. Le logiciel Eagle v. 7.5.0 contient une fonction pour le fraisage qui permet de créer des fichiers CNC ; cette solution semblait prometteuse.





Programme en Visual Basic

La commande du graveur laser requiert d'adapter les fichiers NCD produits par Eagle. Ben et Theo ont écrit un programme de conversion en Visual Basic 2015 (téléchargement gratuit chez Microsoft). La conversion est loin d'être évidente. Ainsi, le fraisage utilise un jeu d'outils complet dont des forets de différents diamètres. Une gravure au laser implique l'adaptation de cette option.
De plus, le point de coordonnées 0,0 se trouve en bas à droite, alors que sur le graveur laser ce point 0,0 est en bas à gauche. Il y a donc bien des calculs en perspective... Il faut en plus des commandes pour éteindre et allumer le laser. Pour couronner le tout, le laser a besoin d'un « certain » temps pour atteindre sa pleine

puissance. Par un heureux hasard, le langage de programmation des machines CNC, G-code, comporte pour cela une instruction que comprend le graveur laser. L'instruction `G04 P100` introduit une temporisation du déplacement du laser, une fois qu'il est allumé.

Il faut envoyer au graveur laser le fichier laser produit. Pour cela on fait appel à un protocole matériel. Dans le programme Basic, on peut, avant d'effectuer la conversion, choisir la vitesse avec laser éteint (*off*, `G00`) et la vitesse avec laser en fonction (*on*, `G01`). Les instructions (`M03`) et (`M05`) permettent respectivement d'allumer et d'éteindre le laser. Pour positionner le laser avec une très grande précision, on peut régler la hauteur de l'axe Z et indiquer aussi s'il faut ou non créer du G-code pour l'axe Z. Si le graveur laser se trouve encore, suite à une utilisation antérieure, à la bonne hauteur, ce réglage peut être omis, d'où un gain de temps. Sinon, l'axe Z est ramené au point zéro (point d'étalonnage) avant de se déplacer à 30 ou 40 mm au-dessus de la plaque. En outre on peut déterminer un décalage (*offset*) pour X et Y. Si l'on veut réaliser plusieurs petits circuits imprimés sur une grande plaque, les *offsets* permettent de les placer les uns à côté des autres. Il faut aussi tenir compte des vitesses de communication entre le PC et le graveur laser.

Fabrication de circuits imprimés

Nous avons procédé à une simulation sur une feuille de papier, puis sur un morceau de contreplaqué. La gravure au laser requiert de procéder à des essais de vitesse de déplacement et de puissance du laser. Le programme en Visual Basic permet un réglage de la vitesse. Une fois que les essais ont été positifs, nous sommes passés au « grand-œuvre ».
Premier essai : un petit circuit imprimé simple face, un *shield* sans fil pour Arduino Mega avec un module nodeMCU, un PCF8583 (horloge en temps réel) avec quartz et pile. Résultat parfait ! Une fois l'écran mis en place et le nodeMCU programmé, le montage, une station météo, fonctionnait. Un test au scanner I²C montra que le PCF8583 se trouvait bien à l'adresse $A0_{HEX}$, prouvant que tout allait bien.

Le second essai (circuit imprimé toujours simple face) s'avéra concluant lui aussi. La durée d'exposition était suffisante, les pistes auraient pu être un peu plus larges. En outre il y avait des différences de largeur du rayon laser d'un graveur laser à un autre. La focalisation est peut-être à revoir.

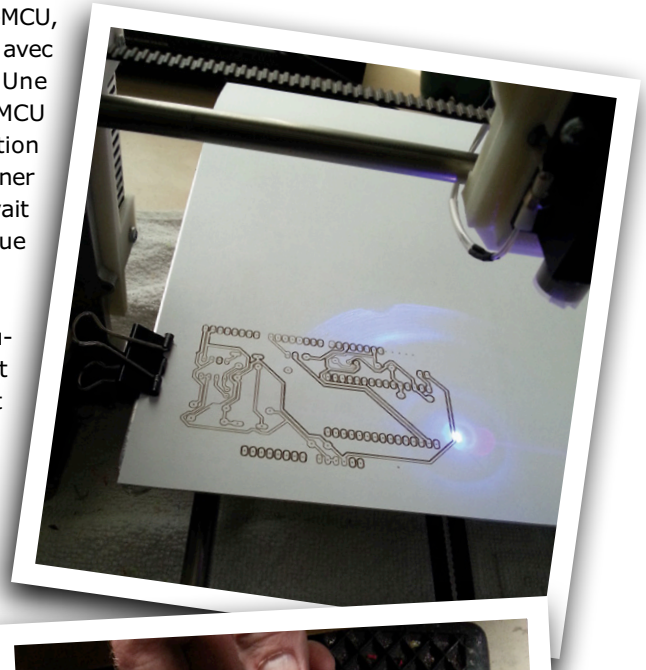
Réussite

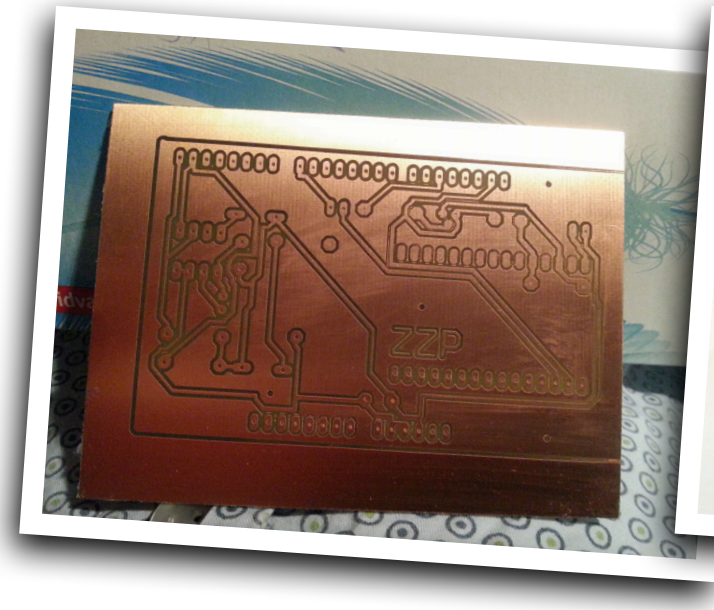
Tous les circuits imprimés produits au cours des essais sont utilisables. Cela montre que l'insolation d'un circuit imprimé avec un graveur laser à UV est parfaitement possible.
Le WisClub (Wis = ***W**eggooien **i**s **s**und!* = Jeter est un crime) sait faire bien plus que des circuits imprimés. Allez donc y jeter un coup d'œil (en néerlandais, pas de version anglaise !). ⏮

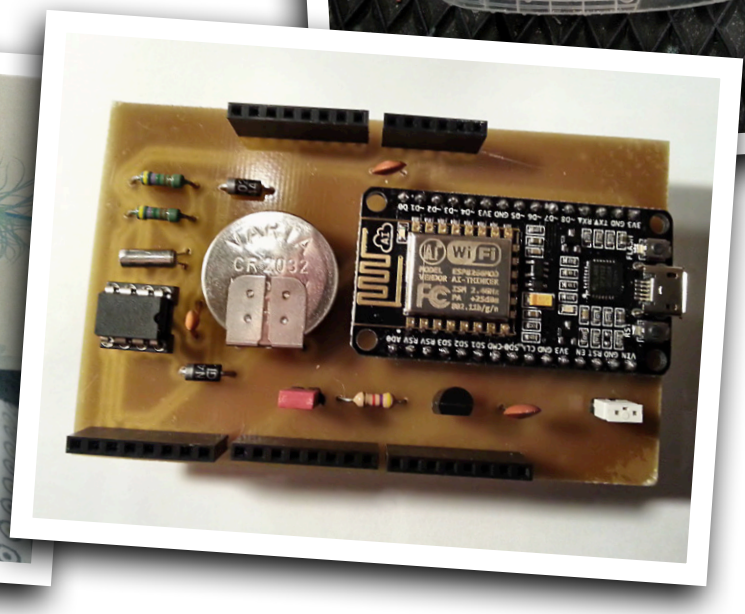
(160096 – version française : Guy Raedersdorf)

Lien

[1] http://blog.wisclub.nl

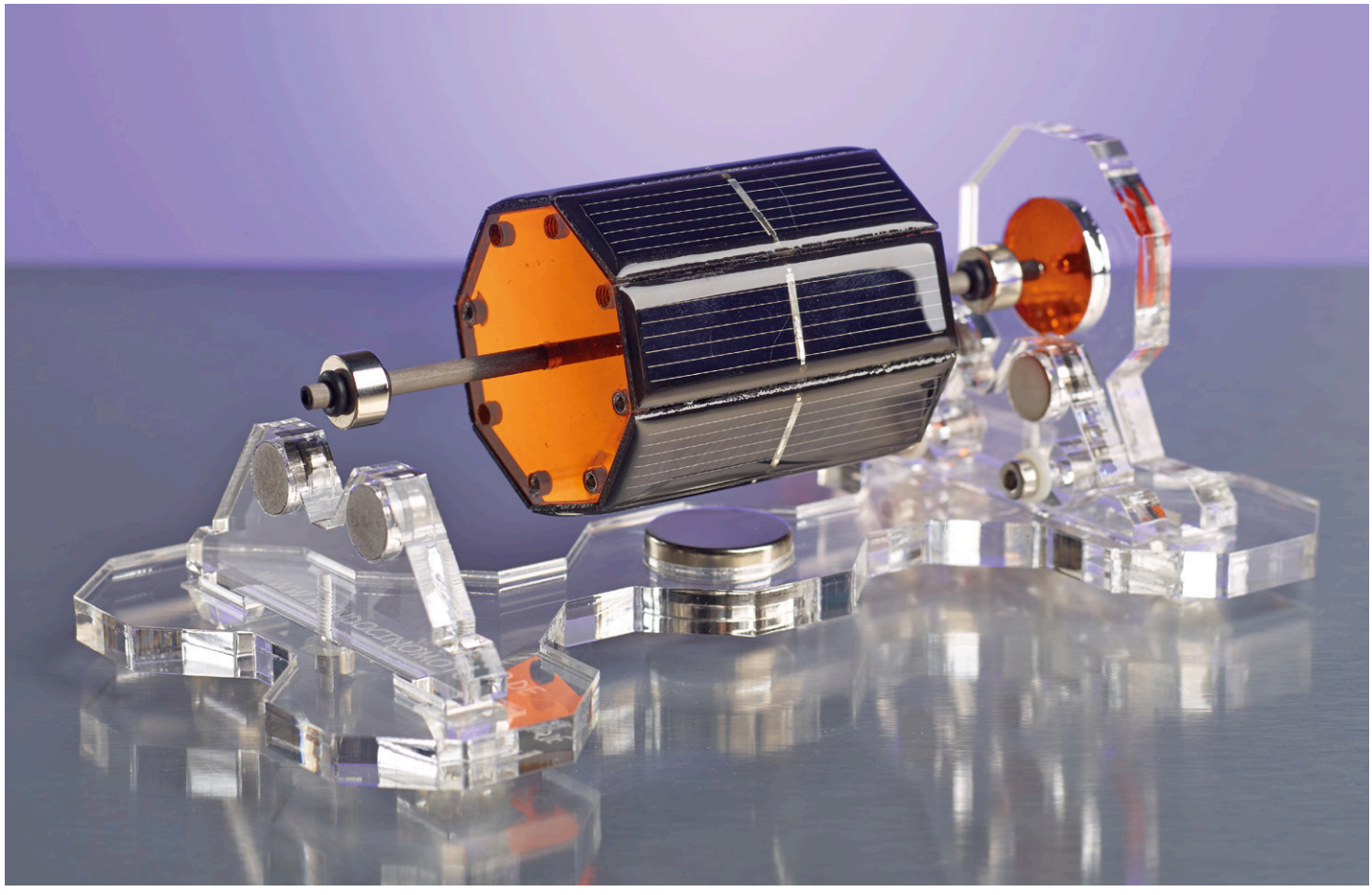






moteur Mendocino

Il flotte et tourne à l'énergie solaire



Manfred Klose (Allemagne)

Il ressemble à une œuvre d'art, est en suspension dans l'air et tourne apparemment de manière autonome, sans l'aide d'une source d'énergie extérieure. Quiconque voit un moteur Mendocino pour la première fois s'étonne et veut savoir comment il fonctionne. Le principe est en fait très simple, mais une belle apparence et une finition soignée sont tout aussi importantes.

Un moteur Mendocino est un moteur électrique alimenté par des cellules solaires ; son rotor repose sur des paliers magnétiques. Le premier moteur a été développé par l'inventeur Larry Spring du comté de Mendocino en Californie.

Il est constitué de deux composants, le châssis et le rotor. Le rotor est un corps octaèdre, traversé en son centre par un axe. Des aimants sont placés à chaque extrémité de l'axe, face aux aimants intégrés au châssis, de même polarité (**figures 1 et 2**). Les aimants se repoussent puisqu'ils sont de même polarité. Ils sont légèrement décalés les uns par rapport aux autres et maintiennent ainsi le rotor en suspension. Selon le théorème d'Earnshaw, aucun objet ne peut demeurer en suspension au sein d'un champ magnétique continu. C'est pourquoi d'un côté se trouve une paroi de butée contre laquelle vient s'appuyer le rotor en raison du léger décalage par rapport aux aimants de soutien, et sur laquelle il peut tourner à l'aide d'une bille. Le rotor reste ainsi prisonnier du champ magnétique.

À titre d'exemple pour cet article, nous examinons le modèle X-8, disponible depuis peu dans l'e-choppe d'Elektor, et produit en petite série par l'auteur.
Le châssis du modèle X-8 est constitué d'une plaque en acrylique dans laquelle est encastré un puissant aimant néodyme, positionné au centre sous le rotor. Celui-ci exerce une force sur les enroulements du rotor, dans lesquels circule un faible courant délivré par les cellules solaires. La force de Lorentz fournit au rotor une impulsion dans la direction de rotation. Le moteur fonctionne en perma-

nence si la lumière est suffisante, grâce à l'agencement des cellules solaires et des enroulements.
La force de traction de Lorentz est très faible, c'est pourquoi un palier magnétique est nécessaire, tout comme une réduction massive du balourd statique. C'est seulement lorsque les forces motrices surmontent l'inertie et le balourd résiduel du rotor ainsi que le frottement de la bille, que le moteur démarre de lui-même.

Structure du moteur Mendocino

Huit cellules solaires sont disposées sur un octogone régulier. Une cellule est éclairée, alors que son opposée est à l'ombre. Ces deux cellules sont connectées en série, ce qui devrait constituer un court-circuit (**fig. 3**). Toutefois dans la pratique, cela fonctionne très bien, car une seule des deux cellules est éclairée à la fois. La différence d'éclairement entre la cellule du dessus et celle du dessous provoque une différence de courant. La cellule éclairée peut délivrer une tension, et celle qui est à l'ombre est bloquée. Un courant circule dans l'enroulement. Si le rotor tourne de 180°, la polarité dans l'enroulement s'inverse. Nous obtenons à nouveau le pôle magnétique souhaité, qui permet de faire tourner le rotor dans le même sens.
Pour nos moteurs, nous avons fait fabriquer des cellules solaires de 65 × 20 mm, avec une tension de 0,5 V. L'intensité du champ magnétique est égale au produit du nombre d'enroulements de la bobine par l'intensité du courant. La bobine ne doit pas avoir une impédance trop élevée afin de pouvoir créer un courant le plus intense possible. La bobine de mes moteurs comprend 60 tours de fil de diamètre 0,3 mm, ce qui donne une résistance d'environ 5 Ω.
L'intensité du courant est égale au rapport tension sur résistance : I = V / R = 0,5 / 5 = 100 mA. Les cellules solaires utilisées débitent un peu moins de 300 mA au soleil. Cette réserve est suffisante. Pour les cellules solaires, on observe une chute rapide de la tension et du courant dès que l'éclairage diminue. C'est pourquoi un plus grand nombre de cellules solaires étroites et donc une densité plus grande d'impulsions de commande successives influencent directement la bonne marche du moteur.
Les bobines intérieures (**fig. 4**) ont la même largeur que l'aimant d'entraînement. Elles se situent directement sous les cellules solaires et ont donc une position optimale dans le champ magnétique. Par rapport à un moteur avec les bobines placées à l'extérieur parallèlement aux cellules solaires, ici la longueur du fil des bobines placées à l'intérieur est réduite de manière significative. Ceci a un effet positif sur le courant.
Le nombre idéal de cellules solaires est de huit. Une cellule solaire qui vient de produire une impulsion de rotation a tourné de 45° par rapport à la lumière lorsque l'impulsion suivante arrive. La diminution rapide du courant fait que le couple « cellule solaire précédente / bobine » ne ralentit pas le rotor. Huit impulsions d'entraînement par tour permettent de bien faire fonctionner le moteur même dans des conditions de faible éclairage. En plein soleil j'ai pu mesurer environ 1400 tr/min. Il est vraiment impressionnant de constater que le moteur démarre automatiquement à la seule lumière d'une bougie de chauffe-plat avec une flamme d'environ 2 cm. ⏮

(160227 – version française : Xavier Pfaff)

Figure 1. Les aimants de l'axe et du châssis se repoussent.

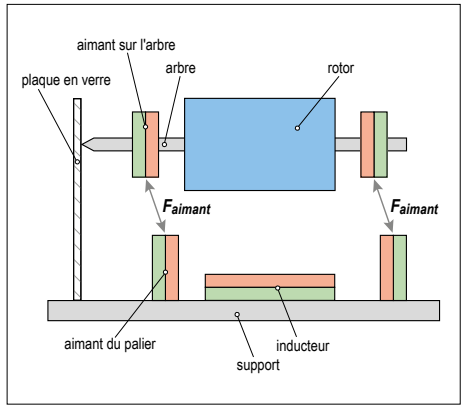


Figure 2. Croquis du roulement d'un moteur Mendocino.

> Le moteur démarre même à la lueur d'une bougie de chauffe-plat.

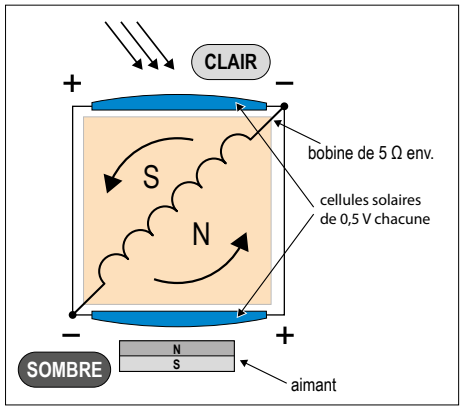


Figure 3. Deux cellules connectées en série, ce qui devrait provoquer un court-circuit. Mais une seule cellule à la fois est éclairée.

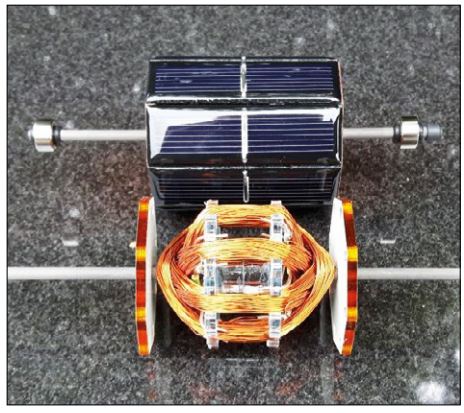


Figure 4. Les bobines se situent au sein des cellules solaires.

Lien

www.elektor.fr/mendocino-motor-x-8

projet 2.0
corrections, mises à jour et courrier des lecteurs

ersatz à LED pour tube fluo

Elektor 01-02/2014, p. 20 (130403)

Lettre de M. Gockenbach

Cher Monsieur Scherer,

Ce matin, j'ai découvert au petit-déjeuner votre article « Ersatz pour tube fluo ». Sa lecture m'a fait sourire. Il y a quelques jours, j'ai acheté ce même tube à LED chez Aldi, en pensant acquérir un dispositif d'éclairage peu coûteux et économe en électricité.

Toutefois, je m'y suis pris d'une tout autre manière. J'ai d'abord mesuré les deux extrémités avec un ohmmètre. L'un des côtés présentait des broches en court-circuit, l'autre quelques centaines d'ohms.

Ensuite, j'ai trouvé dans une échoppe pour bricoleurs une bobine de ballast pour un tube fluorescent de 11 W. En effet il se pourrait qu'une telle bobine soit indispensable pour limiter le courant. Mais comme une bobine de 11 W est dimensionnée pour un courant de 155 mA, elle a donc aussi une très faible résistance. Alors, j'ai branché l'ensemble sur le 230 V et le « tube fluorescent » s'est aussitôt allumé. J'ai alors mesuré le courant dans la bobine, ainsi que les tensions : seulement 11 mA et environ 150 V sur le tube. Ensuite, j'ai branché le tube sur un transformateur variable, sans la bobine, et j'ai fait croître la tension de 0 à 230 V. Le tube s'est allumé à partir de 120 V et le courant est resté à peu près constant jusqu'à 220 V. Cela signifie que l'on peut brancher le tube directement sur le secteur.

Le lendemain, j'ai acheté les deux derniers tubes pour 7,99 € pièce. Il est clairement indiqué sur l'emballage qu'il ne faut pas utiliser ces tubes avec un dispositif électronique d'allumage. Je possède donc maintenant quelques tubes à LED de bon rendement et branchés d'un seul côté.

Klaus Gockenbach

Cher Monsieur Gockenbach,

Je suis heureux d'apprendre que votre expérience rejoint la mienne ;-)

Toutefois, j'ai acheté quelques autres tubes à LED d'autres fabricants, des modèles « longs » (= 120 cm). Ceux-ci présentent un circuit interne différent. Les broches de chaque côté sont en court-circuit et l'électronique se trouve au milieu. Ici, il ne faut installer aucun starter, pas plus un « vrai » qu'un « pseudo » en court-circuit.

Lettre de M. Schrott

1. Circuit d'allumage standard défectueux : problème résolu

Dans l'école de danse où je m'occupe des locaux, l'éclairage est assuré par plusieurs tubes fluorescents de 150 cm avec un système d'allumage standard, c'est-à-dire une bobine de ballast. Il est arrivé que même des tubes neufs refusent de s'allumer. Je suppose que le vieillissement thermique des bobines a provoqué un court-circuit entre spires et qu'elles ne peuvent plus fournir une tension d'allumage suffisante. J'ai donc essayé de remplacer le tube par un tube à LED pour m'épargner un échange de l'ensemble du système d'éclairage – et ça fonctionne parfaitement !

2. Rentabilité

En Allemagne, sur le marché du bâtiment, un tube à LED coûte à peu près trois fois le prix d'un tube traditionnel avec starter. Un éclairage de rechange standard complet coûte environ le même prix qu'un tube à LED. Toutefois, selon les indications du fabricant, un tube à LED a une durée de vie environ trois fois supérieure à celle d'un tube fluorescent, ce qui rend les coûts plus ou moins identiques. En fin de compte, le tube à LED prend l'avantage grâce à une consommation électrique inférieure, égale à moins de la moitié de celle du tube fluorescent.

3. Brochage des tubes de remplacement à LED

Monsieur Scherer mentionne dans le courrier des lecteurs qu'il existe des tubes de remplacement à LED avec différents brochages – soit des broches en court-circuit à chaque extrémité et l'électronique d'allumage au milieu, ou bien celle-ci est branchée entre deux broches du même côté avec l'autre côté en court-circuit. Dans ce dernier cas – le seul que j'ai rencontré pour l'instant – le tube est livré avec un pseudo-starter, comme mentionné dans l'article originel. Dans le premier

cas, au contraire, il est exclu, compte tenu du brochage du tube, d'utiliser un starter réel ou pseudo. Dans ce cas, aucun starter n'est évidemment inclus dans la livraison. Si ce critère ne suffit pas à faire la distinction, on peut tout simplement s'assurer par un test à l'ohmmètre ou à la sonnette si les broches sont reliées d'un seul côté ou des deux. En clair : d'un seul côté: (pseudo) starter en place, des deux côtés : starter enlevé !

4. Montage en tandem

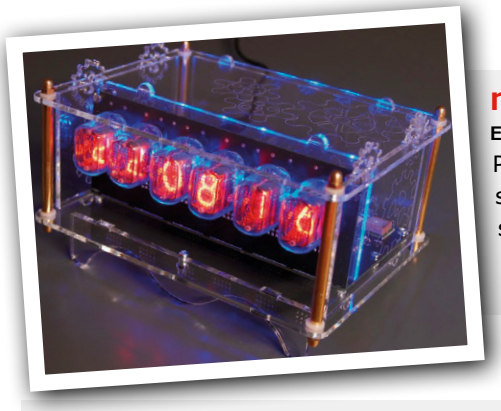
Dans les armoires de salle de bains, entre autres, on trouve quelquefois deux petits tubes fluorescents (60 cm maximum, 22 W) en série avec une seule bobine de ballast (ce que les électroniciens appellent un montage en tandem, voir par exemple http://de.wikipedia.org/wiki/Tandemschaltung). On reconnaît facilement un tel montage en série par le fait que les deux tubes s'éteignent quand on en retire un. Les tubes de rechange courts à LED sont également utilisables dans ce cas, leur tension d'allumage ne dépassant pas les 100 V. Bien entendu, il faut alors prévoir deux pseudo-starters, une solution que je n'ai toutefois pas encore testée.

Robert Schrott

Nouvelle lettre de M. Schrott

Entretemps j'ai également essayé le montage en tandem. Cet essai a montré que les tubes à LED ne sont pas appropriés pour la mise en série. Une mise en série de ces tubes sans ballast ne convient pas non plus – certes les tubes s'allument, mais avec une luminosité réduite pour l'un ou les deux. L'électr(on)icien astucieux arrivera sûrement – bien entendu en respectant toutes les précautions d'usage avec la tension du secteur – à modifier le câblage pour mettre les tubes en parallèle. On peut alors se passer du ballast et du pseudo-starter. Mais on ne pourra plus utiliser de tubes conventionnels.

Robert Schrott



nouvelle horloge Nixie pilotée par GPS

Elektor 05/2016, p. 36 (150189)

Pour habiller l'horloge Nixie, nous proposons un écrin en verre acrylique, livré avec six LED et trois résistances. Ces composants permettent d'éclairer le boîtier. Le schéma de câblage des LED et des résistances se trouve à la page 4 du manuel (en anglais) à télécharger sur notre site :
www.elektor.fr/acrylic-glass-case-for-six-digit-nixie-clock-150189-72

analyseur UART/RS232

Elektor 04/2015, p. 38 (140126)

Nous proposons une nouvelle version du logiciel de l'analyseur : dans cette version 1.1, on peut maintenant choisir le nombre de bits de stop (1 ou 2) ; on peut également changer le bit de parité (pair, impair ou aucun). Pour modifier le nombre de bits de stop, on utilisera la commande « q » ou « Q » et pour le bit de parité « p » ou « P ».

réception d'ELFes

Elektor 09/2014, p. 38 (140035)

En marge de mon activité d'enseignant à l'université Friedrich-Schiller d'Iéna (FSU), je m'intéresse aux ondes électromagnétiques à basses fréquences. Jusqu'ici il s'agissait de fréquences comprises entre 3 Hz et 1 MHz. Comme cela ne me suffisait plus, j'ai pris le risque de me lancer dans la construction d'un récepteur radio proposé par Elektor pour le domaine de fréquences < 1 Hz. Le résultat fut surprenant, car j'ai capté des signaux valides avec des fréquences comprises entre 0,01 et 0,02 Hz. Comme je les ai captés avec une antenne filaire (longueur : 8 cm à 17 m), ils devraient être de nature plutôt électrique que magnétique. De telles ondes ne sont pas traitées dans les ouvrages auxquels j'ai accès. Je n'ai pas non plus trouvé d'explication satisfaisante à l'institut de physique de la FSU.

Alors j'ai pensé que la source de ces oscillations pourrait se trouver dans la géophysique de la Terre (oscillations propres). Mais je ne m'explique toujours pas comment des oscillations mécaniques sphéroïdales ou toroïdales peuvent produire des ondes électromagnétiques. Est-ce que ces fréquences pourraient être justes ? D'autres lecteurs ont-ils fait la même expérience ?

Walter Koch

(version française : Helmut Müller)

compilées par **Robert van der Zwan**

Fast Forward Award 2016 : du vent et… de l'eau

Le gagnant de l'*electronica Fast Forward Award 2016* sait assurément comment exploiter l'énergie éolienne. Mais qu'en est-il de l'exploitation de l'eau avec une grande précision ? C'est l'objectif d'un nouveau débitmètre qui présente une précision rarement atteinte. Le jury FFA a considéré que ce projet était si original qu'il a créé une récompense spéciale, la « Tech for Good ». Son inventeur, l'Australien Lenn Williams (à gauche sur la photo), est reparti chez lui avec un chèque de 5.000 € à utiliser en communication et promotion (RP) pour montrer à tous les acteurs de l'eau les bénéfices (plus) équitables qu'ils pourraient en retirer.

Vous n'êtes pas tenu de nous croire…

Falk Senger, directeur de la foire de Munich, a été très clair en déclarant qu'il était plus que satisfait de la manière dont l'*electronica Fast Forward Award* avait mis en avant toutes ces innovations. En conclusion de la remise des prix FFA 2016, il a indiqué que la qualité et la diversité des projets étaient extraordinaires. Poussé par Elektor, le jury du FFA a contribué au mieux à les mettre en valeur.

electronica 2016 en chiffres

Le salon *electronica 2016* en quelques chiffres : 2.913 sociétés de plus de 50 nationalités différentes ont présenté leurs solutions à environ 73.000 visiteurs venus de plus de 80 pays. Les thèmes dominants ont été la sécurité et l'automobile. 99 % des visiteurs ont jugé que ce salon était « bon » ou « excellent ». Comme le nombre des exposants a progressé de 7 % par rapport à l'édition 2014, Falk Senger, directeur de la foire de Munich, a estimé que les exposants et les organisateurs s'étaient surpassés.

Angela Marten, directrice du salon *electronica* de Munich, et Don Akkermans, directeur des éditions Elektor, ont signé un accord pour poursuivre leur collaboration autour de l'*electronica Fast Forward Award*, la plateforme consacrée aux jeunes pousses, lancée par Elektor. FFA continuera à se développer en 2017 et reviendra sur l'édition 2018 du salon *electronica*.

Le soleil se cache ?

Pourquoi ne pas profiter du

Les gagnants de l'*electronica Fast Forward Award* peuvent s'avancer maintenant ! Deux viennent d'Allemagne, le troisième des États-Unis. Pour sûr, cette matinée du vendredi 11 novembre a été passionnante au salon *electronica 2016* de Munich. L'excitation était palpable jusqu'au dernier moment – sans artifice !

Durant tout le salon, le stand d'Elektor a été entièrement consacré à l'innovation : 35 inventions venues de 16 pays différents. Sous la houlette d'Elektor, le jury international qui comprenait également des sociétés renommées comme STMicroelectronics, Würth Elektronik, Conrad et Trinamic, a eu fort à faire pour choisir les finalistes de chacune des catégories (prototype, *start-up* et idée).

Pendant trois jours, les candidats avaient présenté leur projet. Le vendredi matin, la captivante cérémonie de remise des prix a commencé à 10h précises sur le stand d'Elektor. Vers 10h20, le jury d'*electronica 2016* a pu conclure après quelques délibérations : Mowea de Berlin a dominé la catégorie *Prototype*, BotFactory de New York a gagné dans la catégorie *Start-up* et Kewazo de Garching (Allemagne) a fini premier de la catégorie *Idée*. Ils ont tous les trois été assaillis de questions pertinentes posées par les membres du jury et le public.

À 11h, il était temps de dévoiler le classement final. Le président du jury de la FFA 2016, Clemens Valens, a remis la médaille de bronze à Kewazo pour son (si, si…) robot constructeur d'échafaudage. Tout sourire, Artem Kuchukov a remporté un chèque de 25.000 € à dépenser en relations publiques.

D'un air pince-sans-rire, Clemens a déclaré : « j'appelle le second, et … par conséquent le premier aussi ». La médaille d'argent a été décernée à BotFactory pour son prototype d'imprimante de bureau pour circuits imprimés (une machine qui place aussi les composants !). Son PDG J.F. Brandon a reçu un chèque de 50.000 €.

INDISCRÉTIONS • Félicitations à Tanja Pohlen pour l'organisation de l'electronica FFA, et à Angela Pendant toute la durée du salon, Tessel Renzenbrink et Jens Nickel ont partagé en ligne leurs trouvailles d'Elektor TV, et Patrick Wielders, le cameraman, ont aidé les personnes timides à s'exprimer et à Grotenrath, Chantalle Reuling et Raoul Morreau (service commercial & clients/membres) pour leur

moindre souffle d'air ?

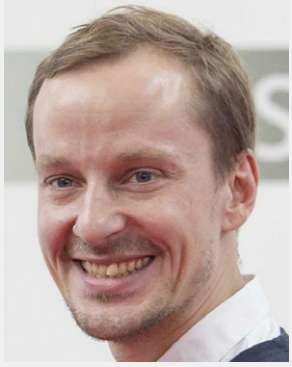
Le grand gagnant, Mowea, propose des microéoliennes qui permettent aux particuliers et aux TPE de combiner les énergies solaire et éolienne, et de bénéficier d'une alimentation stable même hors réseau de distribution. Le PDG, Till Naumann, et son équipe sont repartis avec un chèque de 75.000 € et un stand gratuit pour le prochain salon *electronica* (2018) !

Photo : Till Naumann et Lara Obst de Mowea en train de présenter leur éolienne « pico wind ».

Marten pour celle du salon electronica 2016 • munichoises • Tandis que Jan Buiting, l'animateur partager des infos pointues • Merci à Julia disponibilité sur le stand d'Elektor … … …

PROFIL D'EXPERT

Elektor est au cœur d'un réseau de plus de 1 000 experts et d'auteurs engagés dans la publication de livres, d'articles, de DVD, de webinaires et autres événements. Coup de projecteur !



Nom : **Till Naumann**

Formation : **docteur de l'université technique de Berlin**

Intérêts professionnels : **je me passionne pour le vent et l'aérodynamique… mon père était un pilote de voltige.**

Je suis un ingénieur originaire de Berlin. J'ai écrit ma thèse de doctorat sur l'aérodynamique des éoliennes à l'université technique (TU) de Berlin. J'aimerais que le « pico-éolien » devienne abordable et que mes connaissances des grandes éoliennes y contribuent. J'ai rencontré mon partenaire [Amberger, Ed.] à la TU de Berlin. Nous avons développé un système modulaire d'énergie éolienne d'une puissance nominale de 10 kW.

Avez-vous été surpris de gagner le premier prix electronica FFA ?

Notre équipe a été très surprise ! Venus de Berlin par le train, nous avons dormi dans un hôtel bon marché et mis toute notre énergie dans la présentation au jury – après trois jours presque sans dormir. Nous avons réussi et nous en sommes très heureux !

Savez-vous déjà comment vous allez dépenser vos 75.000 € ou bien est-ce encore trop tôt pour en parler ?

Comme nous sommes une jeune pousse, nous avons besoin de promotion dans toutes les directions possibles : charte graphique, amélioration de notre site, vidéos sur nos prototypes et nos essais de terrain, contenu pour nos partenaires B2B, présence dans les salons professionnels, et aussi édification d'une communauté solide sur Facebook et autres médias. Nous prévoyons une campagne de financement participatif fin 2017.

Qu'espérez-vous accomplir dans les cinq prochaines années ?

Nous espérons construire une entreprise à croissance rapide, mais solide, qui fournira des sources d'énergie hors réseau abordables et propres. Nous voulons contribuer à décarboner la production de l'énergie.

Qu'est-ce qui vous a convaincu de mettre sur pied ce programme éolien ?

Je suis un passionné du vent et de l'aérodynamique. Comme mon père était un pilote de voltige, enfant, j'ai passé beaucoup de temps dans les airs. Quoi qu'il en soit, Andreas et moi avons décidé de concrétiser notre passion pour le petit éolien quand nous avons vu l'intérêt que suscitaient partout les solutions d'énergie renouvelable de petite taille, surtout dans les pays à faible infrastructure énergétique.

Quels seront les développements électroniques fondamentaux dans un futur proche ?

Andreas est expert en électronique et logiciel. Nous voulons doter nos éoliennes d'une intelligence suffisante pour collecter les données de production et de comportement des utilisateurs. Nous voulons également intégrer nos éoliennes dans des micro-réseaux. En un mot, tout reste à faire ! ⏮

(160260 – version française : Yves Georges)

chatdoku casse-tête pour elektorniciens

En ce début d'année, notre traditionnelle grille hexadoku est remplacée par une grille chatdoku (le contour de cette grille dessine une tête de félin).

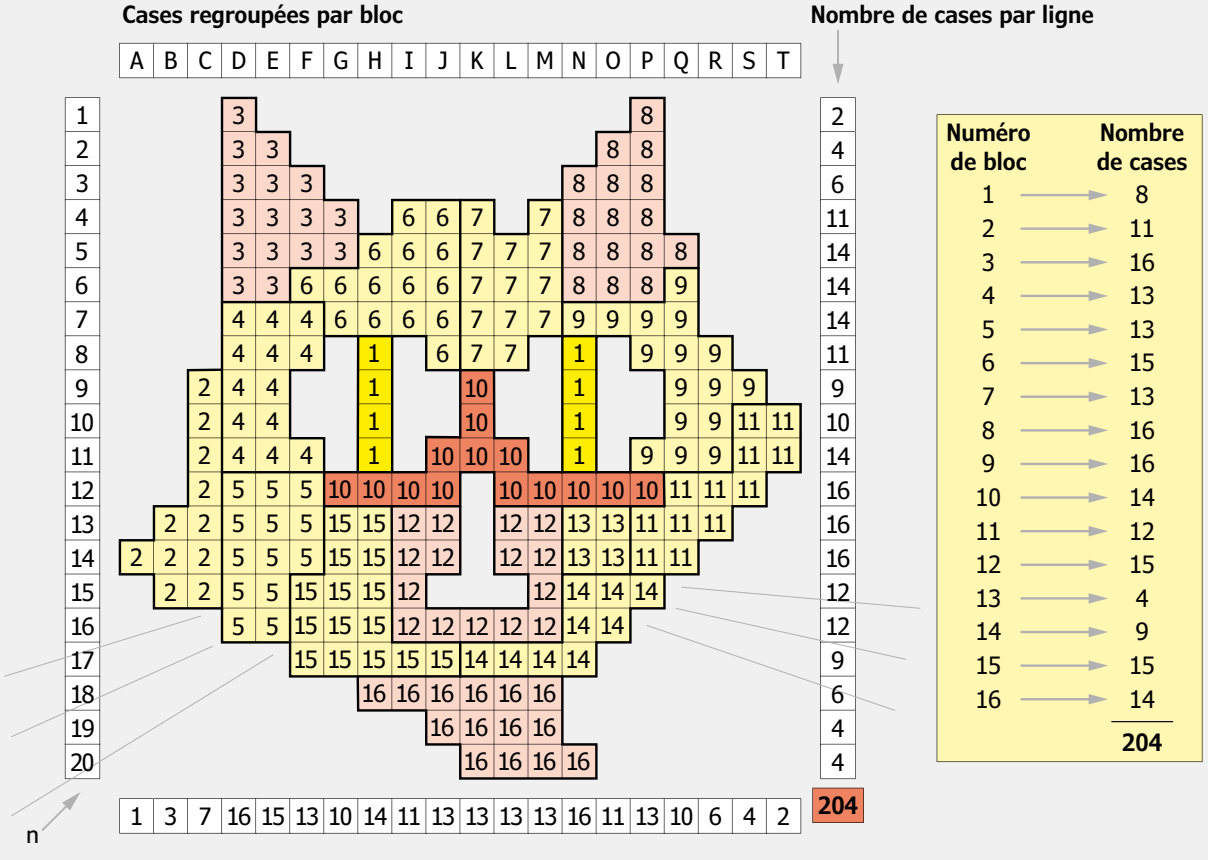
Cette grille est proposée par un fidèle lecteur, Claude Ghyselen. Les règles appliquées pour remplir la grille ainsi que les caractères autorisés restent les mêmes que pour la grille hexadoku *normale*.
Rappel : une grille hexadoku est composée de chiffres du système hexadécimal, de 0 à F.
Il faut la remplir 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 bloc (les blocs sont délimités par un filet gras).

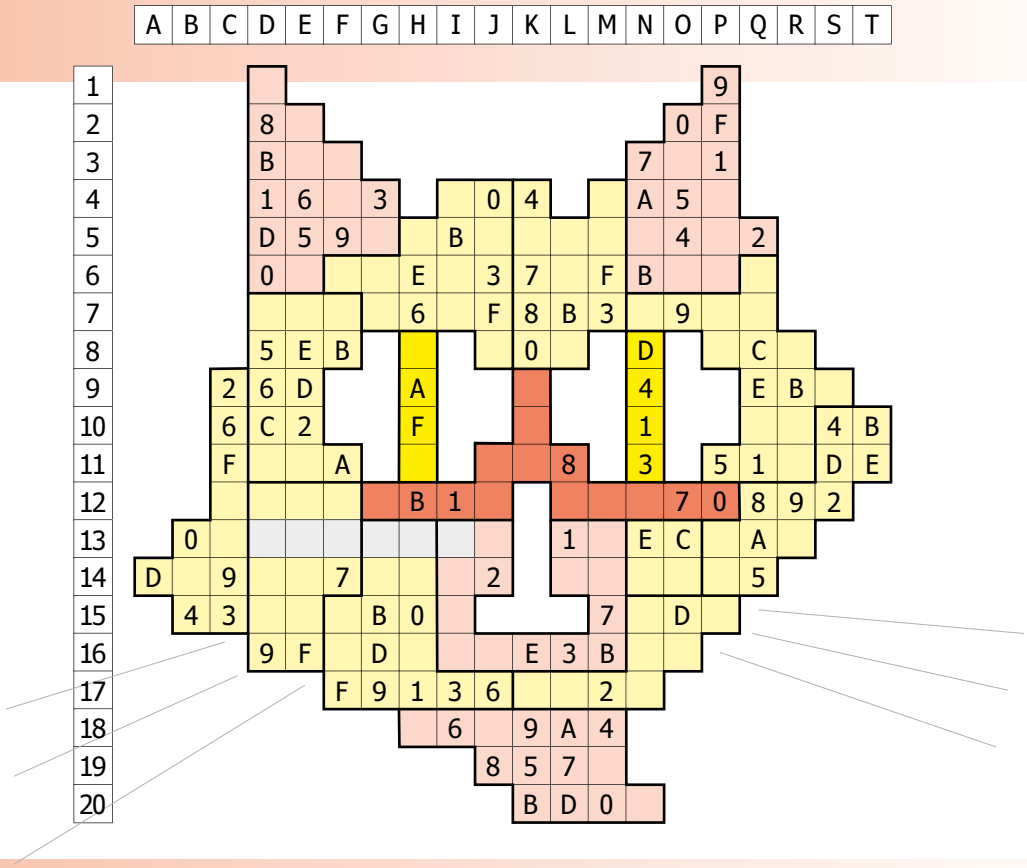
Le chatdoku se distingue de l'hexadoku par les points suivants :

- La grille comporte 16 blocs, mais 20 lignes (1 à 20) et 20 colonnes (A à T) (voir grille ci-dessous).
- Cinq zones restent vides (en blanc).
- Les blocs ne sont pas de forme carrée ou rectangulaire, ils ressemblent plutôt à des pièces de puzzle (on parle d'*hexadoku jigsaw*).
- Le bloc n°1 (pupille des yeux, en jaune vif) est décomposé en deux parties disjointes, une pour l'œil gauche, l'autre pour le droit.
- Le nombre de cases de chaque bloc dépend de sa position et de sa forme, mais n'excède pas 16. Cette grille peut donc toujours être qualifiée d'hexadoku.
- Le nombre total de cases actives est de 204, contre 256 pour un hexadoku.

Comme certains blocs contiennent moins de 16 cases, il n'est pas possible d'y inscrire tous les caractères hexadécimaux. C'est une petite difficulté supplémentaire dont il faut tenir compte lors de la résolution.

Certains chiffres, déjà placés dans la grille, définissent la situation de départ. Pour participer, inutile de nous envoyer toute la grille, il suffit de nous envoyer la série de six chiffres sur fond grisé.





Participez et gagnez ! Nous tirons au sort **trois** des réponses internationales correctes reçues dans les délais ; leurs auteurs recevront chacun un bon d'achat Elektor d'une valeur de **50 €**.
À vos crayons !

Envoyez votre réponse (les chiffres sur fond grisé) avec vos coordonnées par courriel, avant le **1er février 2017** à l'adresse **hexadoku@elektor.fr**

Les gagnants

La solution de la grille du numéro de novembre est **389BD**

Les trois bons Elektor d'une valeur de 50 € vont à :
- Bernd **Lörler** (Allemagne)
- Johann **Parfuss** (Autriche)
- Claude **Viou** (France).

Bravo à tous les participants et félicitations aux gagnants !

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

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.

reichelt elektronik
Technology connects.

- Plus de 45 ans d'expérience
- Envoi en 24 heures
- Plus de 70 000 produits

LE SPOT PUBLICITAIRE DE REICHELT

rch.lt/vfr

FIND OUT MORE ▶

Photo : Foto- und Bilderwerk

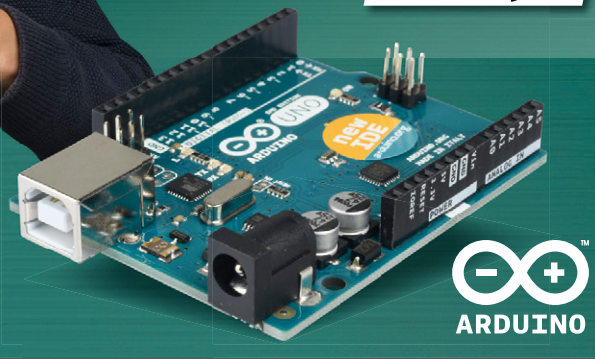
« Grâce aux microcontrôleurs d'Arduino, les idées deviennent tout simplement une réalité ! »

Le petit préféré parmi les Arduinos !

ARDUINO UNO REV. 3

- ATMega 328
- 14 entrées et sorties numériques (dont 6 en PWM)
- 6 entrées analogiques
- Prise USB

ARDUINO UNO **16,39**



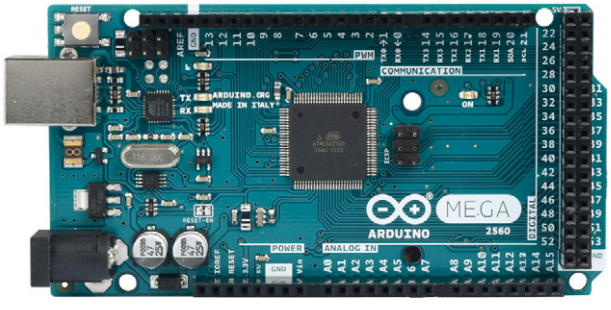
ARDUINO

Il suffit de brancher et c'est parti !

ARDUINO™ MEGA 2560

- ATMega 2560
- 54 entrées et sorties numériques (dont 14 en PWM)
- 16 entrées analogiques
- Prise USB

TOUS LES PRODUITS ARDUINO
rch.lt/yb

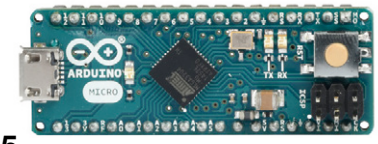


ARDUINO MEGA **29,33**

Petit, mais performant !

ARDUINO™ MICRO

- ATMega 32u4
- 20 entrées et sorties numériques (dont 7 en PWM)
- 12 entrées analogiques
- Prise micro-USB

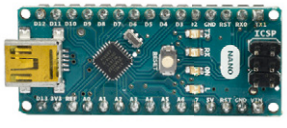


ARDUINO MICRO **15,55**

Petit, mais ... attention !

ARDUINO™ NANO

- ATMega 328
- 14 entrées et sorties numériques (dont 6 en PWM)
- 8 entrées analogiques
- Prise mini-USB



ARDUINO NANO **19,24**

Prix du jour! Prix à la date du: 01. 12. 2016 Prix en € hors T.V.A., frais de port en sus · reichelt elektronik, Elektronikring 1, 26452 Sande (Germany) Les langues de notre boutique:

MODES DE PAIEMENT INTERNATIONAUX:
SOFORT ÜBERWEISUNG | VISA | MasterCard | PayPal
ACHETER EN LIGNE TOUT SIMPLEMENT !

www.reichelt.fr
ASSISTANCE TÉLÉPHONIQUE EN ANGLAIS: +49 (0)4422 955-360