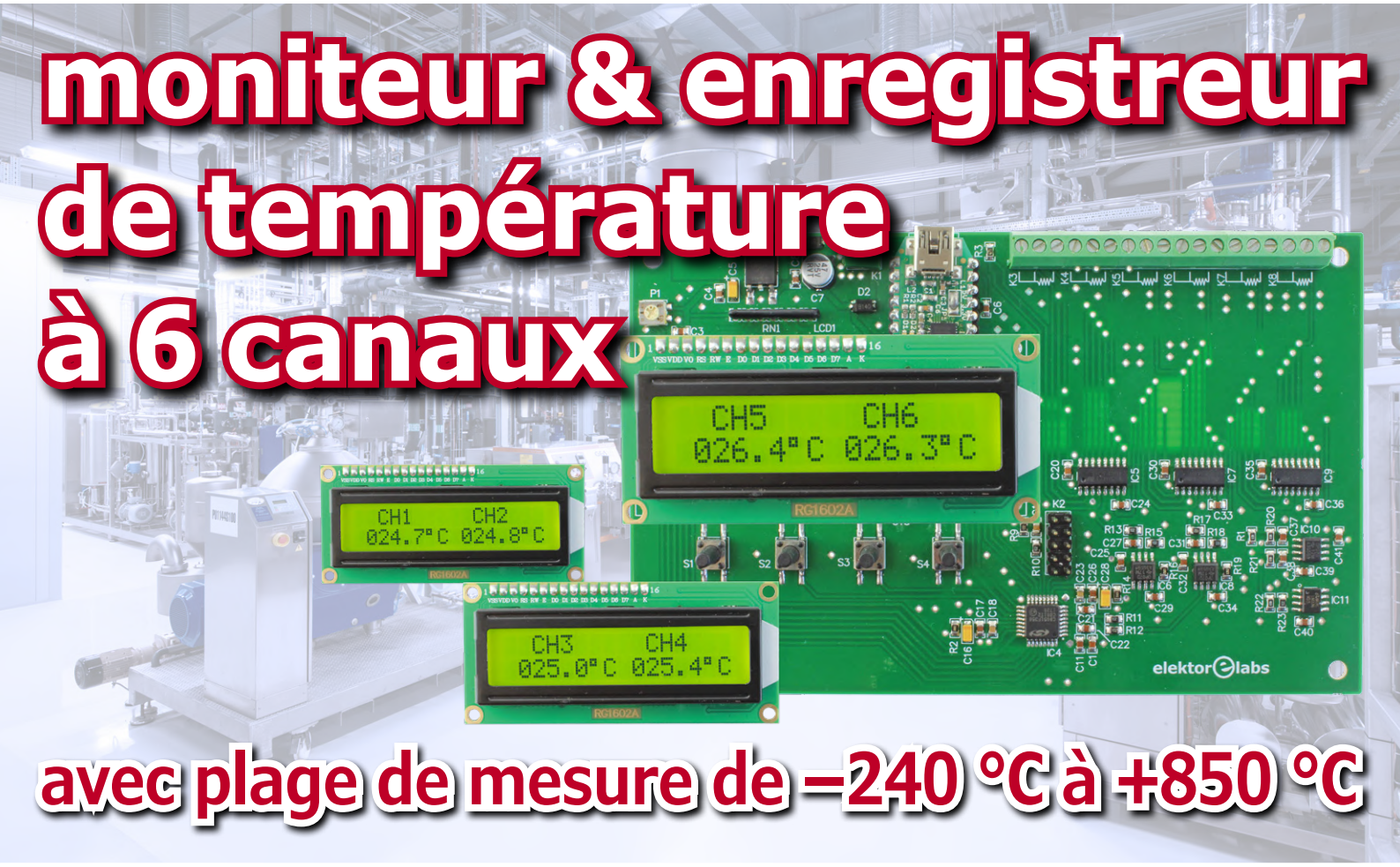
juillet/août 2019 | n° 478 | www.elektormagazine.fr

elektor

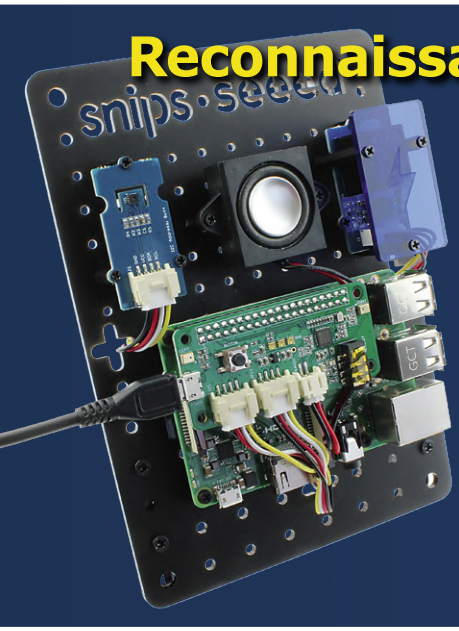
elektor

DÉCOUVRIR ⊳ CRÉER ⊳ PARTAGER

moniteur & enregistreur de température à 6 canaux

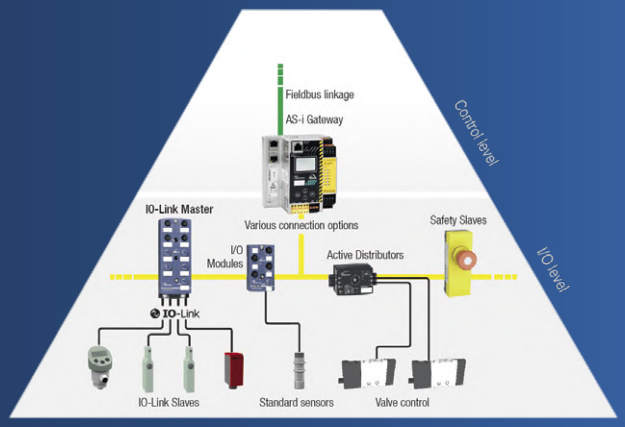


avec plage de mesure de –240 °C à +850 °C



Reconnaissance vocale avec Snips

mise en œuvre avec le Raspberry Pi



Interface AS : les bases

bus de terrain pour les automates industriels

carte de liaison d'E/S MIDI compatible DIN et TRS ⊳ projet SCCC (3) ⊳ TUE/InMotion : objectif Garage 56 ⊳ alimentation pour sondes différentielles par USB ⊳ affichage de données série sur une page web (PHP ou Python) ⊳ nouvelle carte CPLD avec MAX10 ⊳ ESP32 : surveillance de la consommation d'eau (alerte par SMS en cas de fuite) ⊳ cryptomonnaie IOTA (et FPGA pour RPi) : PiDiver ⊳ assistant vocal 100% privé avec Snips ⊳ Q&R : circuits analogiques ⊳ ESP32 + RTC + GPS + écran = serveur NTP privé ⊳ baristors de l'alternatif au continu

ISSN 0181-7450

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

L 19624 - 478 - F: 15,50 € - RD

www.forum-electronique.com

Made by GL events

MÊME LIEU / MÊME DATE

Measurement world

CIM2019 MÉTROLOGIE

42ème année,
n° 478 – juillet/août 2019

ISSN 0181-7450
Dépôt légal : juin 2019
CPPAP 1120 T 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

Téléphone : (+33) 01 70 61 27 41
du lundi au vendredi de 10h à 13h

Télécopie : (+33) 01 70 61 27 42

www.elektor.fr | www.elektormagazine.fr

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

Publicité :
Margriet Debeij
Tél. : +49 (0)241 955 09 174
margriet.debeij@elektor.com

Imprimé aux Pays-Bas par
Pijper Media – Groningen
Distribué en France par M.L.P.
et en Belgique par A.M.P.

Tondeuses robots, épisode…

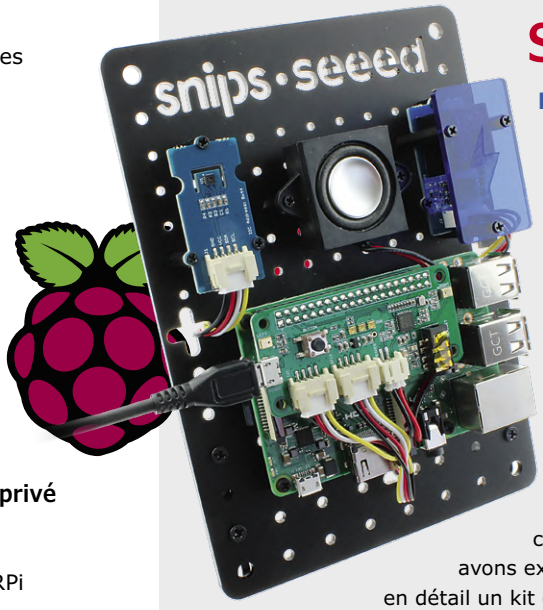
La rédaction d'Elektor est inondée de communiqués de presse sur de nouveaux composants et des cartes électroniques, ou même de produits finis. J'ai un faible pour la domotique et parfois la dent dure pour les appareils domestiques « électronisés ». J'ai par exemple raillé ici l'aspirateur à moteur « numérique ». Je ne rate aucun épisode de la saga des tondeuses robots. Les premiers modèles ne marchaient que dans des jardins rectilignes et sans pente. Puis on les a vus travailler en pente, gérer plusieurs zones de tonte, fonctionner sans boucle de guidage, détecter les vides, apprécier la hauteur de l'herbe, détecter sa présence... Autant d'améliorations de la tonte elle-même, tâche principale du robot. Bravo !
Avec la vogue des objets connectés, les tondeuses robots se sont branchées elles aussi : pour le confort de l'utilisateur, la programmation du nombre d'heures/jour et des jours de la semaine a été déportée sur l'écran d'un ordiphone ou d'une tablette. Et là les fonctions exotiques se sont multipliées : marche/arrêt à distance, alerte en cas de vol… Le summum, ce fut la commande à distance de la tondeuse depuis l'autre bout du monde.
Eh non ! C'était sans compter avec la tondeuse qui répond aux ordres que, depuis votre canapé, vous donnez à l'enceinte connectée Alexa : démarrage de la tonte, retour à la station de charge… Et entretemps, Alexa vous informe de la progression de la tonte.
À quoi cela rime-t-il d'envoyer, pour mettre en marche une tondeuse, des données dans le nuage et de faire tourner des calculateurs ? Je vous laisse réfléchir à cette question pendant que je vais faire le tour du jardin derrière ma bonne vieille tondeuse mécanique.

Mariline Thiebaut-Brodier

ce mois-ci

42e année – nº 478
juillet/août 2019

en coulisse

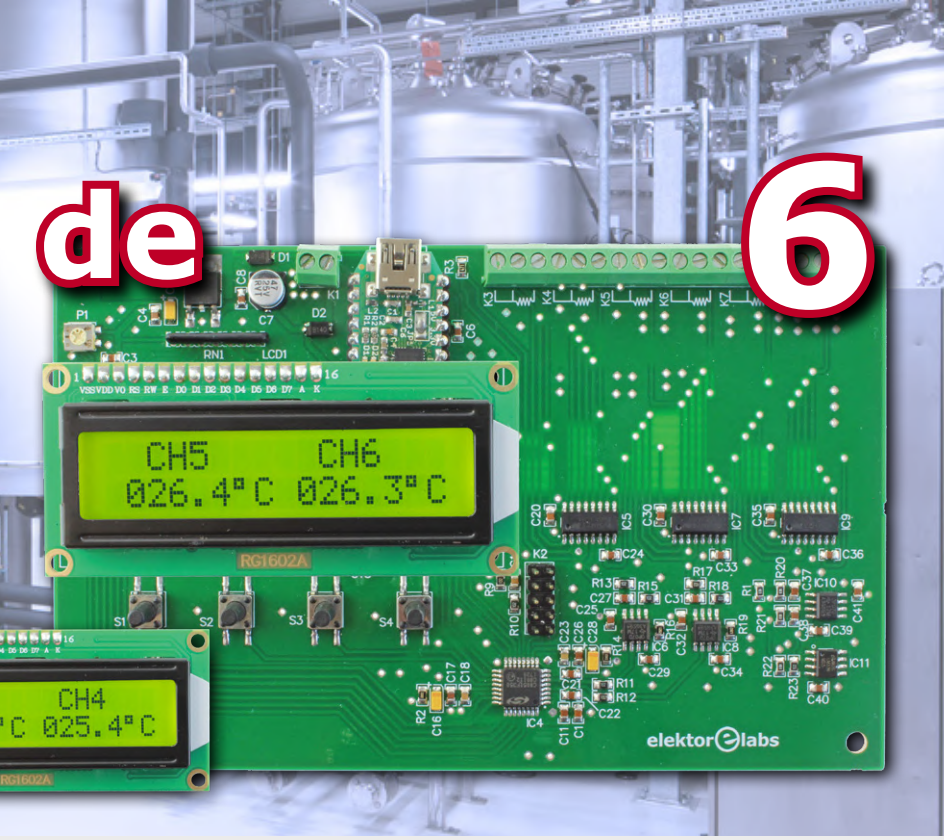


Snips
reconnaissance vocale pour le Raspberry Pi

Les assistants vocaux comme Alexa, Google Home ou Siri sont aujourd'hui devenus monnaie courante. Pour autant, la reconnaissance de la parole n'est pas traitée localement, mais dans le nuage, chez le fournisseur – ce qui soulève un certain nombre de questions à propos de la confidentialité des données. Il existe cependant d'autres solutions, notamment Snips, système de reconnaissance vocale qui fonctionne intégralement sur un matériel local comme le RPi. Nous avons examiné pour vous en détail un kit de reconnaissance vocale et découvert comment l'utiliser pour développer nos propres applications...

22

84
Rétronique
des tubes à vide au silicium
avec quelques étapes intéressantes



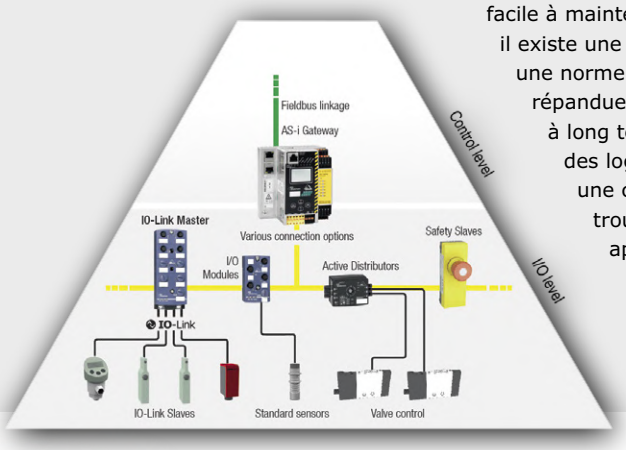
elektor
DÉCOUVRIR ▷ CRÉER ▷ PARTAGER

interface AS : les bases

fonctionnement du bus pour l'automatisation industrielle

68

Au début, tout était commandé à l'aide de câblage fait maison et de formats de données propriétaires. Cette approche est pourtant depuis longtemps révolue en automatisation industrielle. Désormais on utilise des cartes dernier cri et des systèmes de bus spécialisés. Outre l'avantage évident d'être facile à maintenir et à développer, il existe une autre raison à cela : une norme industrielle largement répandue garantit la disponibilité à long terme du matériel et des logiciels, du moins dans une certaine mesure. Vous trouverez ci-après un aperçu de l'interface AS.



sur la scène : les projets

bientôt sur ces pages

Extrait du sommaire du prochain numéro :

- Mini-serre avec LED
- Voltmètre sans fil avec LoRa
- Chargeur universel
- Bibliothèque graphique pour ESP32 et écran tactile Raspberry Pi
- Minuterie à 8 canaux avec OLED
- Télécommande pour modèle réduit de catamaran
- Convertisseur USB/UART à la page
- Circuit de basses graves
- Émission de données WSPR avec SDR

Sous réserve de modification.
Le numéro de septembre-octobre 2019 paraîtra le 22 août 2019.

PROJET DU LABO

moniteur & enregistreur de température à 6 canaux

mesures de –240 °C à +850 °C

Clemens Valens (labo d'Elektor) et **Sunil Malekar** (Inde)

La température est un paramètre important à contrôler dans beaucoup de procédés industriels. De même à la maison, le réfrigérateur, le congélateur, le four de la cuisine et le climatiseur mesurent et ajustent tous la température pour remplir leur mission. Les stations météo, les voitures, les ordinateurs, tablettes et ordiphones surveillent aussi la température. À bien y penser, la température pourrait bien être la grandeur physique la plus surveillée au monde.

Même mon enceinte Bluetooth bon marché déclenche une alarme lorsqu'elle chauffe trop (ce que j'ai découvert après l'avoir placée sur un radiateur). L'être humain aime tellement mesurer la température que nous avons décidé de concevoir un appareil capable d'en mesurer jusqu'à six. D'habitude, les thermomètres ne mesurent qu'une seule température – certains deux, souvent marquées « intérieur » et « extérieur » – mais pour le contrôle de procédé, plus de canaux peuvent être nécessaires. L'étendue de mesure est également importante, raison pour laquelle l'appareil présenté ici peut mesurer des températures dans une vaste plage, de –240 °C à +850 °C. Notez qu'il s'agit de limites logicielles ; les limites effectives dépendent fortement des capteurs utilisés.

Sondes à résistance

Pour mesurer la température, nous allons utiliser ce qu'on appelle des RTD, sous forme de sonde Pt100. RTD est l'abréviation anglaise de Détecteur de Température à Résistance, un dispositif constitué d'un long fil métallique avec une relation résistance/température connue, enroulé sur un support approprié. Le métal utilisé est généralement du platine, du cuivre ou du nickel. La précision du capteur dépend de la pureté du métal. Le platine présente la relation résistance/température la plus stable sur la plage de température la plus grande.

Une sonde

INFOS SUR LE PROJET

8051

surveillance de température

débutant

➡ connaisseur

expert

env. 4h

fer à souder, programmateur C2 (voir texte)

env. 50 €

Pt100 est un RTD en fil de platine (« Pt » dans le Tableau périodique). La valeur « 100 » indique que la valeur de la résistance est de 100 Ω à 0 °C. On peut avoir d'autres valeurs, comme 500, 1000 ou même 2000. Les sondes Pt100 sont couramment utilisées et faciles à trouver. Elles se déclinent en différents

Caractéristiques

- 6 canaux pour RTD Pt100
- Étendue de mesure : −240 °C à +850 °C
- µC 8051
- Écran alphanumérique à deux lignes
- Sortie série des données au format CSV

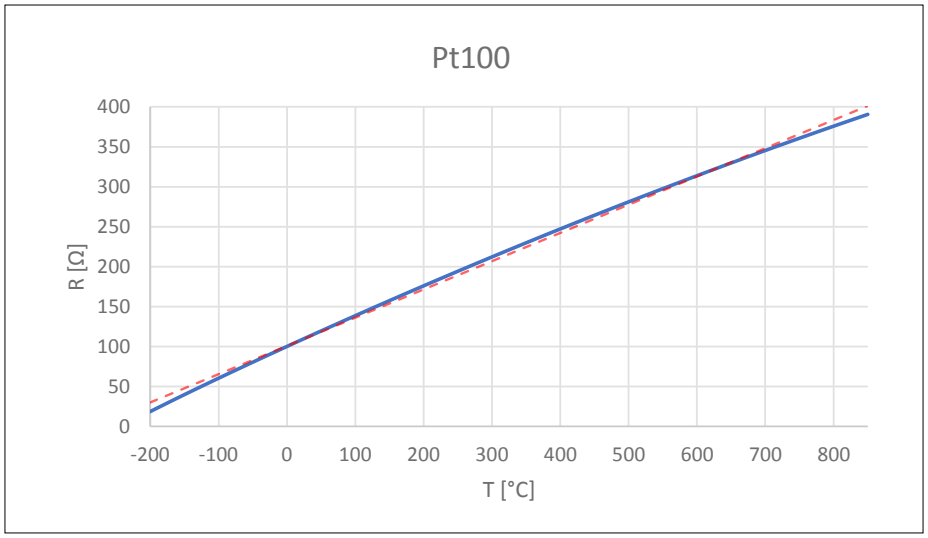


Figure 1. La relation entre la résistance et la température d'une sonde au platine (Pt) n'est pas linéaire. Elle peut être obtenue par approximation avec une table de conversion composée de courts segments rectilignes de la courbe.

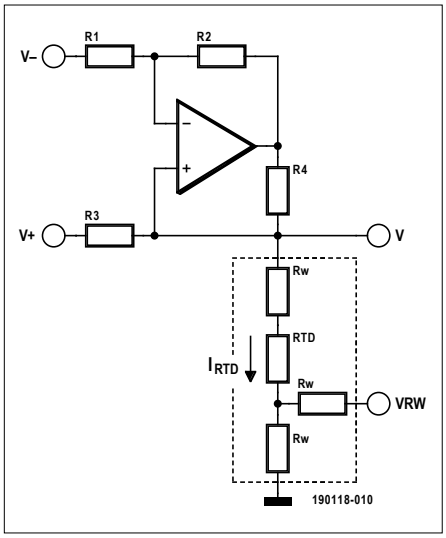


Figure 2. La source de courant de Howland a une impédance de sortie très élevée. Elle est utilisée pour exciter un RTD à 3 fils.

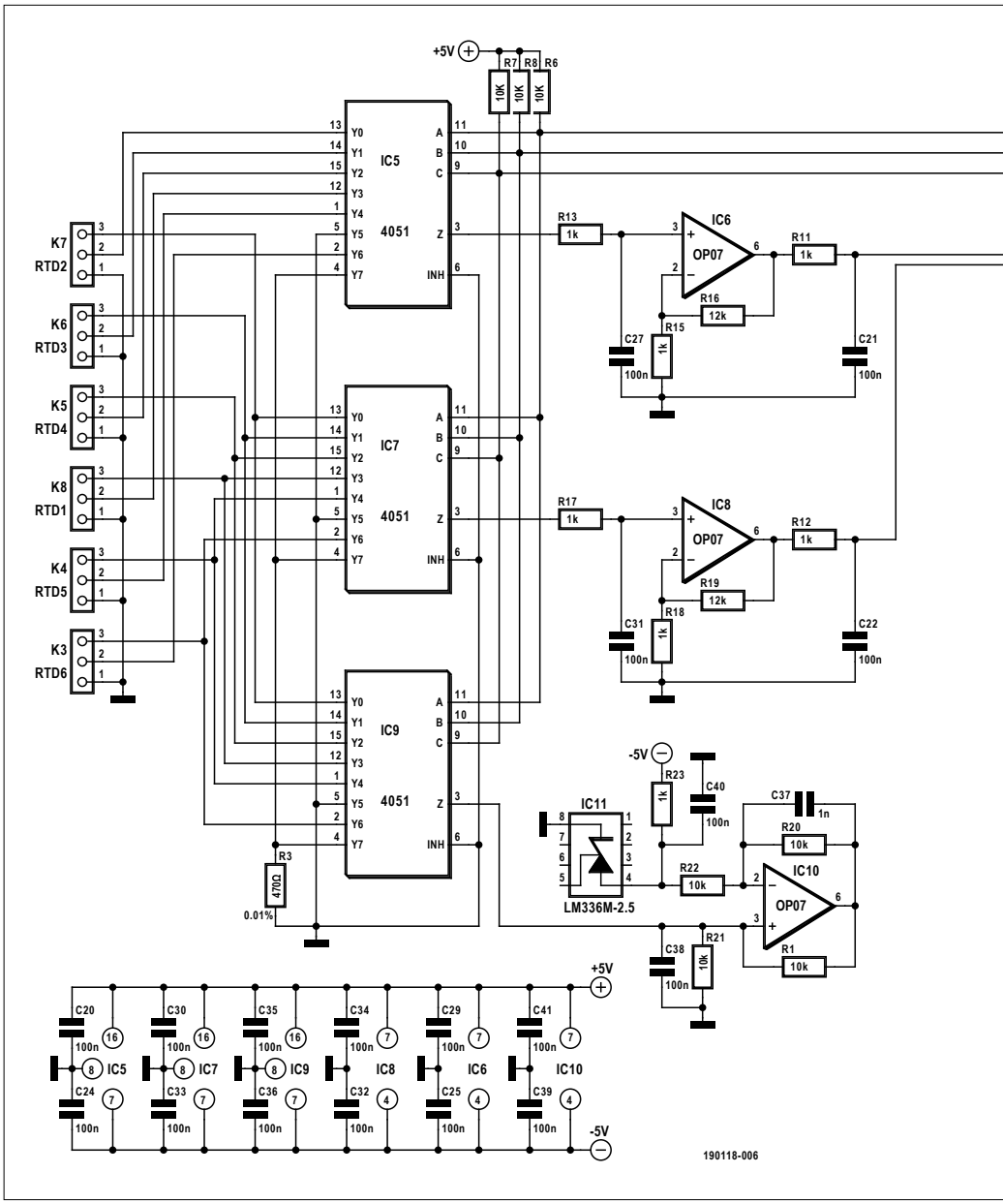
niveaux de qualités (classes) qui déterminent leurs prix, gamme, justesse et précision. Elles sont aussi munies d'un nombre variable de fils de connexion : 2, 3 ou 4. On utilise les fils supplémentaires pour mesurer la résistance des fils de connexion au RTD (voir plus loin). Comme ces fils – qui peuvent être assez longs – se comportent comme des RTD imprécis, il est important de compenser leur effet pour garantir des résultats fiables et reproductibles. Pour éviter de trop compliquer notre système, nous avons opté pour des RTD à 3 fils.

Un RTD n'est pas un thermocouple

Ceci est important à comprendre. Un thermocouple est une jonction entre deux métaux différents qui produit une tension en fonction de la température, à cause de l'effet thermoélectrique (Seebeck). Par conséquent, un thermocouple n'a pas besoin de source d'alimentation pour fonctionner alors qu'un RTD nécessite une certaine forme d'excitation. Les thermocouples ont tendance à être moins précis que les RTD, mais ils sont très peu chers.

Linéarité

Un dernier, mais important point à mentionner sur les RTD est leur légère non-linéarité de la relation résistance/température (cf. **fig. 1**). Ceci implique qu'il faut faire quelques calculs pour obtenir la température à partir de la résistance mesurée. On a longtemps utilisé l'équation de Callendar-Van Dusen pour les RTD



Pt100, mais en 1990 le *Comité International des Poids et Mesures* l'a remplacée par un polynôme du 12e ordre valide dans la gamme de −259,35 °C à 0 °C et un second polynôme du 9e ordre pour les températures de 0 °C jusqu'à 961,78 °C. Comme il faut une forte puissance de calcul (et de débogage) pour évaluer de tels polynômes, la plupart des applications utilisent simplement des tables de conversion. Nous avons aussi fait ainsi.

Source de courant de Howland

Comme brièvement mentionné ci-dessus, on doit exciter un RTD d'une certaine façon pour l'utiliser. Pour cela, on peut avoir recours à une source soit de tension constante soit de courant constant. Nous avons préféré une source de courant constant, car elle pose moins de problèmes avec de longs fils de capteurs. Dans ce cas la tension aux bornes du capteur ne dépend que du courant et, bien sûr, de la température, mais pas en plus de la résistance des fils de connexion.

Il y a de nombreux modèles de sources de courant, mais une communément utilisée avec les RTD est la source de courant de Howland ou pompe de courant (**fig. 2**). Ce type de source a l'avantage d'avoir une impédance de sortie très élevée par rapport aux autres et elle peut absorber aussi bien que fournir du courant. Toutefois, pour un fonctionnement correct, il faut que les résistances R1 à R4 qui entourent l'ampli-op aient une tolérance supérieure ou égale à 1%.

Lorsque le rapport R1/R2 est égal au rapport R3/R4, alors le courant de sortie est donné par :

$$I_{RTD} = (V^{+} - V^{-}) / R3 \text{ [A]}$$

Compensation des fils de connexion

La figure 2 montre aussi comment le RTD est connecté à la source de courant avec ses trois fils. En supposant que les trois fils soient identiques et que nous pouvons mesurer les tensions « V » et « V_{RW} » à très haute impédance d'entrée (c.-à-d. qu'il n'y a pas de courant qui circule dans les directions de V et VRW (plus exactement V_{RW})), alors la tension aux bornes du RTD est donnée par :

$$V_{RTD} = V - 2\,V_{RW} \text{ [V]}$$

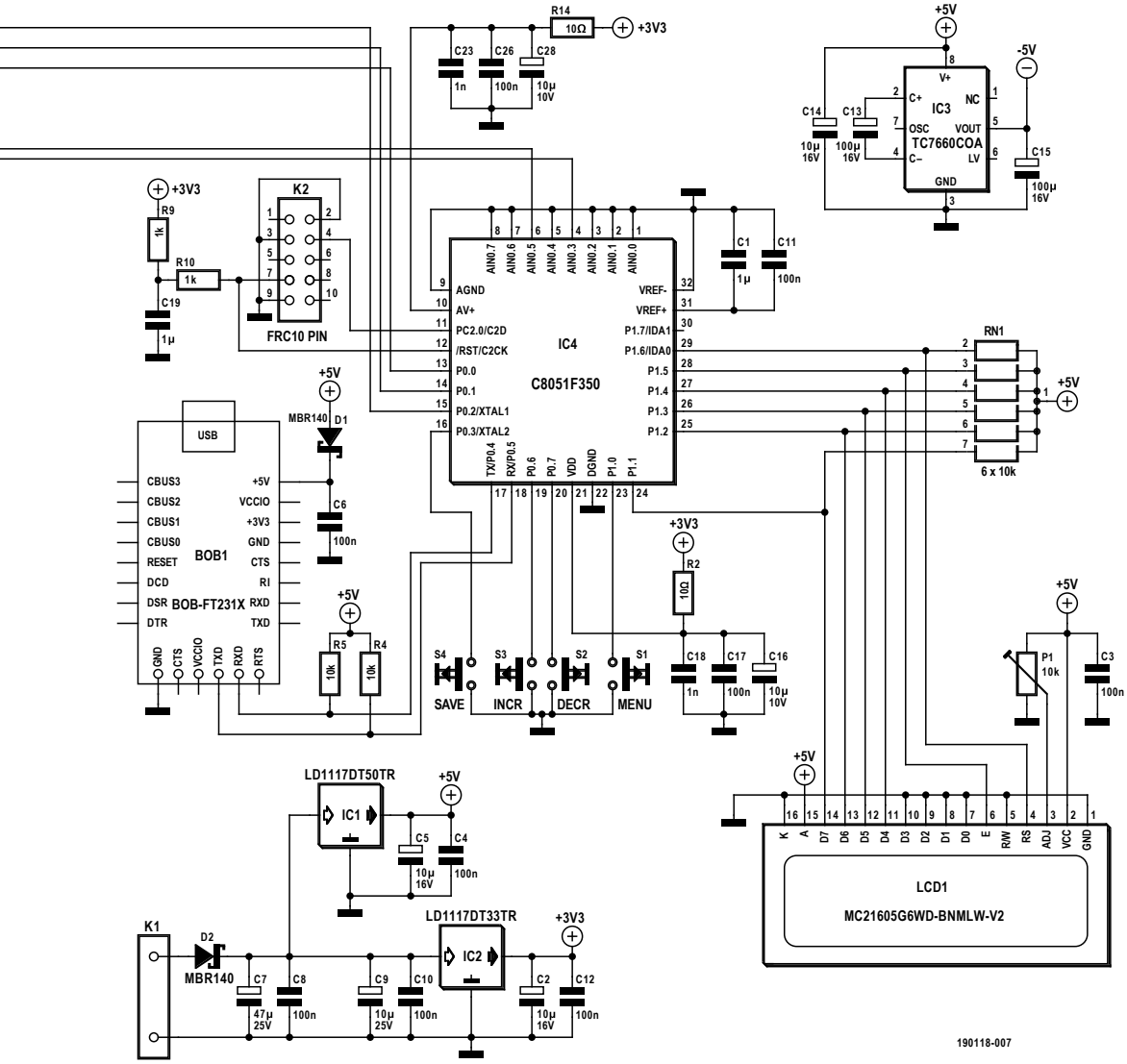


Figure 3. Schéma de l'enregistreur de température multinœuds. L'étage d'entrée analogique est à gauche, la partie numérique à droite.

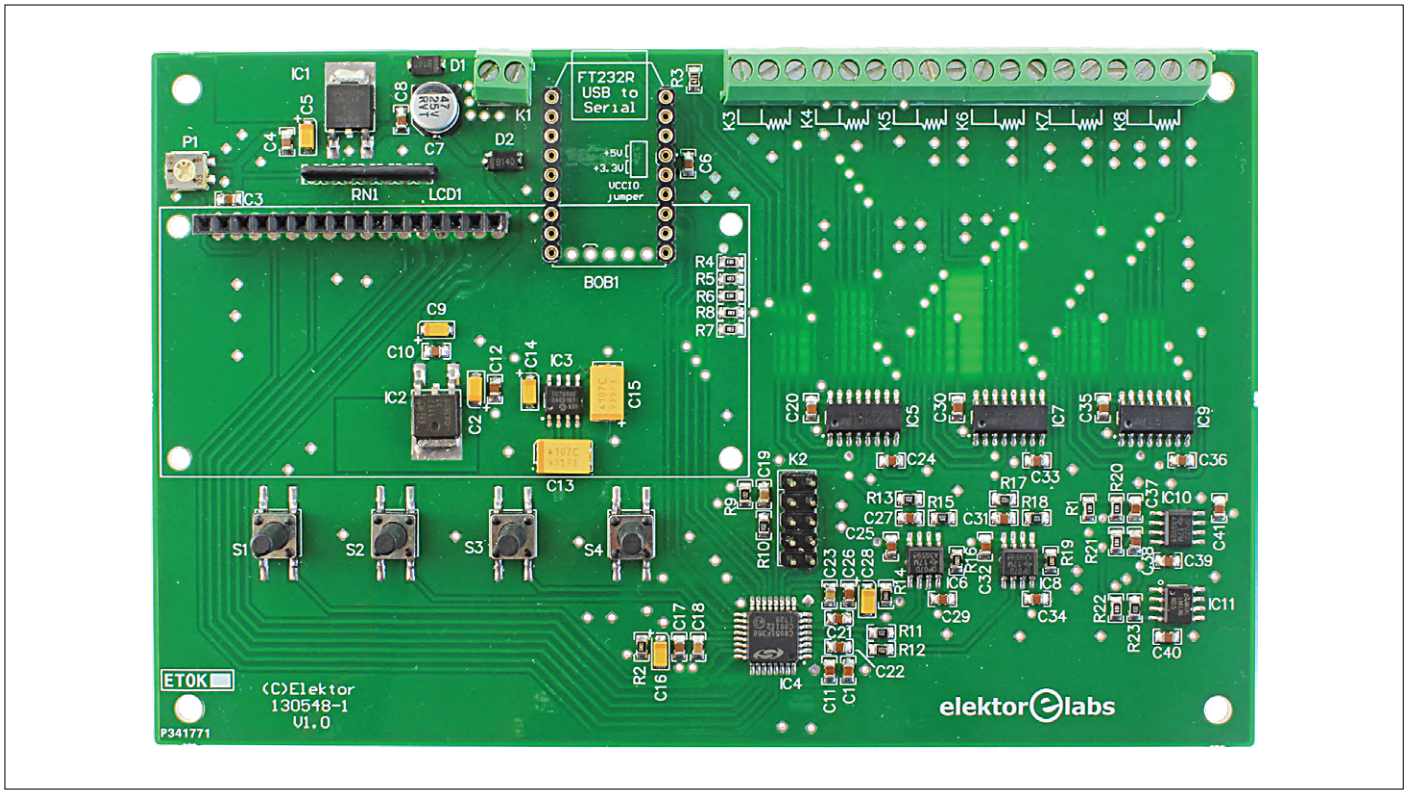


Figure 4. La carte principale assemblée. Notez que cette carte est la version 1.0 avec l'ancien eBoB FT232R au lieu de l'eBoB FT231X.

Ceci termine notre introduction théorique, et nous pouvons poursuivre par l'étude du schéma réel.

Le multiplexeur d'entrée

À gauche de la **figure 3**, nous trouvons les connecteurs K3 à K8 pour les six canaux RTD. Un fil de chaque RTD est relié à GND, les deux autres à un circuit intégré multiplexeur CMOS de type 4051. IC7 transmet le signal « V » de la figure 2 au reste du système ; IC5 fait de même pour le signal « VRW ». Le troisième multiplexeur, IC9, fonctionne dans l'autre sens et relie la sortie de la source de courant (construite autour d'IC10) au RTD sélectionné.

Chaque multiplexeur a huit canaux, dont deux sont utilisés pour les signaux de référence V_{min} (GND) et V_{max} (V_{R3}). Vous comprenez maintenant pourquoi il y a six canaux RTD et pas sept ou cinq. Ces signaux de référence suivent le même chemin que les signaux RTD (R3, une résistance à tolérance de 0,01% est même placée à côté des connecteurs de RTD pour imiter le plus possible un RTD). Comme les valeurs de ces signaux sont connues, ceci permet au logiciel d'étalonner l'étage de numérisation et de supprimer les erreurs introduites par le multiplexeur et la source de courant.

Les sorties du multiplexeur d'entrée sont amplifiées individuellement d'un facteur 13 par IC6 et IC8. Le signal « V » de la figure 2 parvient alors à l'entrée CAN 3 (AIN0.3) du microcontrôleur, tandis que le signal « VRW » est appliqué à AIN0.5.

Source de courant, deuxième partie

La source de courant de Howland dans le projet comprend IC10, les résistances R1, R20, R21 et R22. V^+ de notre description précédente est relié à GND et a donc une valeur de 0. V^- est égal à −2,5 V, produit par la référence de tension IC11. R21 correspond à R3 dans la description précédente. Le courant de sortie est : (0 − −2,5) / 10 000 = +250 [µA].

Alimentation

Théoriquement, la tension d'alimentation du circuit, raccordée à K1, peut aller jusqu'à 15 VCC, mais cela produira d'inutiles contraintes sur les régulateurs de tension à faible chute. La tension d'alimentation minimale est de 7,5 VCC, la valeur recommandée étant 9 VCC. La consommation de courant totale est d'environ 70 mA, y compris les 20 mA consommés par le rétroéclairage de LCD1. (Nota : ce dernier intègre une résistance de limitation de courant).

IC1 abaisse la tension d'alimentation à 5 V pour l'essentiel du circuit, excepté le microcontrôleur IC4 ; IC2 produit une alimentation en 3,3 V pour cette partie intelligente.

L'étage d'entrée analogique nécessite aussi une tension d'alimentation de −5 V, prise en charge par IC3.

Le microcontrôleur

Ces dernières années, on a tellement vu défiler de circuits à base d'Arduino que certains en sont venus à croire qu'il n'existe qu'un seul type de µC à 8 bits. Avec son cerveau pimpant à base de 8051, notre circuit prouve que ce n'est pas vrai ! Maintenant, avant de faire la grimace, sachez que presque quarante ans après son introduction, le 8051 est encore un processeur à 8 bits largement utilisé. Fabriqué par Silicon Labs, le µC C8051F350 utilisé ici fonctionne en interne à 49 MHz et exécute la plupart de ses instructions en un cycle, ce qui en fait un composant plutôt rapide.

Le C8051F350 fait partie d'une famille de quatre qui ne diffèrent que par le type de boîtier et la résolution des CA/N. Le modèle F350 est le plus grand avec un CA/N sigma-delta à 24 bits, huit entrées analogiques dédiées et 17 broches d'E/S numériques. C'est un contrôleur orienté

analogique avec filtres numériques intégrés et deux CA/N à 8 bits. Dans ce projet, nous n'utilisons pas toutes ses capacités, mais vous pourrez étudier plus en détail ce µC pour un futur projet.
Bien que fonctionnant sous 3,3 V, les broches d'E/S numériques du C8051F350 tolèrent le 5 V, raison pour laquelle il s'interface bien avec la circuiterie périphérique alimentée en 5 V (notez les résistances de rappel R4 à R8 et le réseau RN1).
Pas besoin d'un quartz ou de résonateur, car le µC a un oscillateur étalonné sur puce.
Quatre boutons-poussoirs (S1 à S4) et un écran LCD constituent l'interface utilisateur (UI). P1 règle le niveau de contraste de l'écran ; si vous oubliez de le régler la première fois que vous alimentez la carte, l'écran peut ne rien afficher.
Pour la communication avec un ordinateur, nous avons rajouté la possibilité de brancher un convertisseur série USB vers TTL sur la carte. Nous avons utilisé notre propre eBoB, mais il est aussi possible de connecter une interface fonctionnellement équivalente. Notez que les photographies de cet article montrent un prototype qui utilisait l'eBoB FT232R, réf. 110553 d'Elektor (un classique). Depuis, il a été remplacé par notre eBoB FT231X, réf 180537, compatible avec le standard *de facto* des câbles de conversion UBS-série « FTDI ».

À propos du logiciel

À la date de rédaction, l'aimable personnel de Silicon Labs distribuait encore des licences gratuites pour Keil µVision5, afin que les gens qui développe du logiciel en C pour les dispositifs à C8051 puissent le faire confortablement. Lorsque vous demandez une telle licence, elle ne semble valide que pour un mois seulement, mais une fois installée dans l'EDI elle devient permanente. Bien sûr, nous avons exploré cette voie, et tout s'est bien passé – mais ne sachant pas quand ils vont décider d'arrêter de le faire, nous avons porté notre logiciel sous SDCC, le compilateur à code source ouvert pour les microcontrôleurs 8051 et PIC.
SDCC ne fonctionne pas avec des projets et n'utilise aucun autre outil basé sur GCC. Notre projet n'est par conséquent qu'un ensemble de fichiers dans un dossier avec un fichier *batch* pour les compiler.
Comme tout programme en C, la partie visible commence avec la fonction `main`. La première chose à faire ici est de désactiver la temporisation du chien de garde, vous pouvez toujours la réactiver plus tard lorsque vous serez prêt. (Nota : le compilateur Keil fait cela dans le fichier de démarrage de l'exécution (CRT), donc vous ne le remarquez pas. On s'en est aperçu à nos dépens).
Après initialisation des périphériques du µC – top d'horloge système à 1 ms, UART réglé sur 9600 bauds, etc. – le programme étalonne le CA/N. C'est une fonction remarquable du µC, car elle nous permet de gommer beaucoup des imperfections de notre étage analogique. Après étalonnage, V_{max} (la tension aux bornes de la résistance à 0,01%, R3) sera lue comme `0xffffff` (valeur maximale sur 24 bits) tandis que V_{min} (GND) produira une lecture de `0`. Le CA/N applique la correction dans le matériel ; rien à faire pour nous par logiciel.

Un mot sur l'échantillonnage

Le reste du temps, `main` va tourner dans une boucle sans fin, en scrutant les pressions de touches et en rafraîchissant l'affichage lorsque nécessaire. Toutes les secondes, un échantillon est capturé sur un canal, converti en une valeur en degrés Celsius ou Fahrenheit, et affiché sur l'écran. Une seconde plus tard, le canal suivant est mis à jour. En arrière-plan, le CA/N fonctionne à un rythme plus élevé d'environ 18,7 Hz. Cette plus grande vitesse est nécessaire pour deux raisons. Premièrement, pour effectuer une mesure avec un RTD, on doit convertir deux tensions (« V » et « VRW »), car la valeur du RTD doit être compensée de la tension entre les fils de connexion. Deuxièmement, le CA/N dispose d'un filtre passe-bas intégré qui a besoin de six échantillons pour calculer une valeur de sortie. De ce fait, il faut au moins six échantillons pour une mesure de RTD, ce qui correspond à 320 ms. En réduisant la fréquence d'affichage à 1 Hz, le système a suffisamment de marge pour échantillonner et filtrer confortablement.

Table de conversion

Les échantillons sont convertis en valeurs de température à l'aide d'une table de conversion. On peut trouver des tables de conversion pour Pt100 sur l'internet et nous en avons compilée une de −240 °C à +850 °C par pas de 10 °C. En dessous de −240 °C, les choses deviennent hautement imprécises et nous avons donc limité l'affichage à −260 °C (qui correspond en réalité à un débordement). De même, la limite supérieure a été fixée à +850 °C.
Bien sûr, la plupart des lectures du CA/N se trouveront entre deux valeurs de la table, d'où l'utilisation d'une interpolation linéaire pour estimer la température. Si on souhaite des conversions plus précises, on peut diminuer le pas de la table de conversion, tant qu'elle tient dans la mémoire (il n'y en a pas beaucoup). Si la table devient vraiment grande et peu pratique, vous pouvez envisager d'implémenter un algorithme de recherche plus efficace que la recherche linéaire que nous avons utilisée ou d'employer une approximation polynomiale.

Construction

Même si le projet utilise principalement des composants CMS, l'assemblage du circuit imprimé ne devrait pas être trop difficile. Commencez par placer les composants à deux broches comme les résistances et les condensateurs céramique puis continuez avec les composants plus grands. Respectez et vérifiez la polarité des composants polarisés comme les diodes et les condensateurs électrolytiques.
Il vaut mieux placer et tester l'alimentation (D2, IC1, IC2 et les condensateurs autour) avant de monter les autres composants.
Prenez soin d'utiliser des résistances avec une tolérance de 1% dans l'étage d'entrée, sauf R3 qui doit avoir une tolérance de 0,01%. Pour faire une source de courant vraiment bonne, vous pourriez utiliser des résistances à 0,1% de tolérance pour R1, R20, R21 et R22.
Selon la façon dont vous placez ou non l'interface USB/Série, vous devrez peut-être utiliser des connecteurs plus haut pour l'écran, car il est monté au-dessus de l'eBoB série.

Charger le micrologiciel

Le connecteur K2 sert pour la programmation du micrologiciel dans le µC. Ce connecteur respecte le standard officiel *Silicon Labs 2-wire C2 Programming/Debugging Interface Standard* donc connecter un programmateur compatible C2 devrait être simple. En revanche, il est moins facile de mettre la main sur un tel programmateur, raison pour laquelle nous nous en sommes fait un. Le téléchargement gratuit sur la page de cet article [3] comprend un croquis Arduino

qui transforme n'importe quelle carte compatible Arduino Uno en un programmateur C2. Vous pouvez aussi le trouver sur GitHub [2]. Les broches tolérantes au 5 V du µC nous permettent de le faire en toute impunité.
Connectez la broche A0 de l'Arduino à la broche 7 de K2 (« C2CK »), la broche A1 de l'Arduino à la broche 4 de K2 (« C2D ») et la broche GND de l'Arduino à la broche 2 de K2. Connectez l'Arduino à l'ordinateur avant d'alimenter la carte et tout devrait bien se passer.
Un script en Python3, aussi inclus dans le téléchargement, permet de charger les fichiers Intel HEX dans le programmateur fait maison. Ce script nécessite la bibliothèque **pySerial**.

Utilisation du système

Il y a deux micrologiciels pour ce projet, un avec les options de débogage (suffixe « d ») et un sans options. Les fonctions principales de l'UI sont les mêmes pour les deux.

- « Menu » (S1) : appuyez sur cette touche pour accéder à la « page » réglages. Sur cette page, vous pouvez changer les unités de température entre Celsius et Fahrenheit en pressant S1 de nouveau. Les touches « Decr » (S2) et « Incr » (S3) permettent d'ajuster la valeur de rafraîchissement de l'écran d'une seconde à une minute. Ceci fait, vous pouvez soit presser « Save » (S4) pour stocker les nouveaux réglages en mémoire non volatile ou attendre l'expiration de la page. Lorsque la page expire, les nouveaux réglages seront utilisés jusqu'au prochain changement ou redémarrage.
- « Save » (S4) : appuyez pour réétalonner le système.

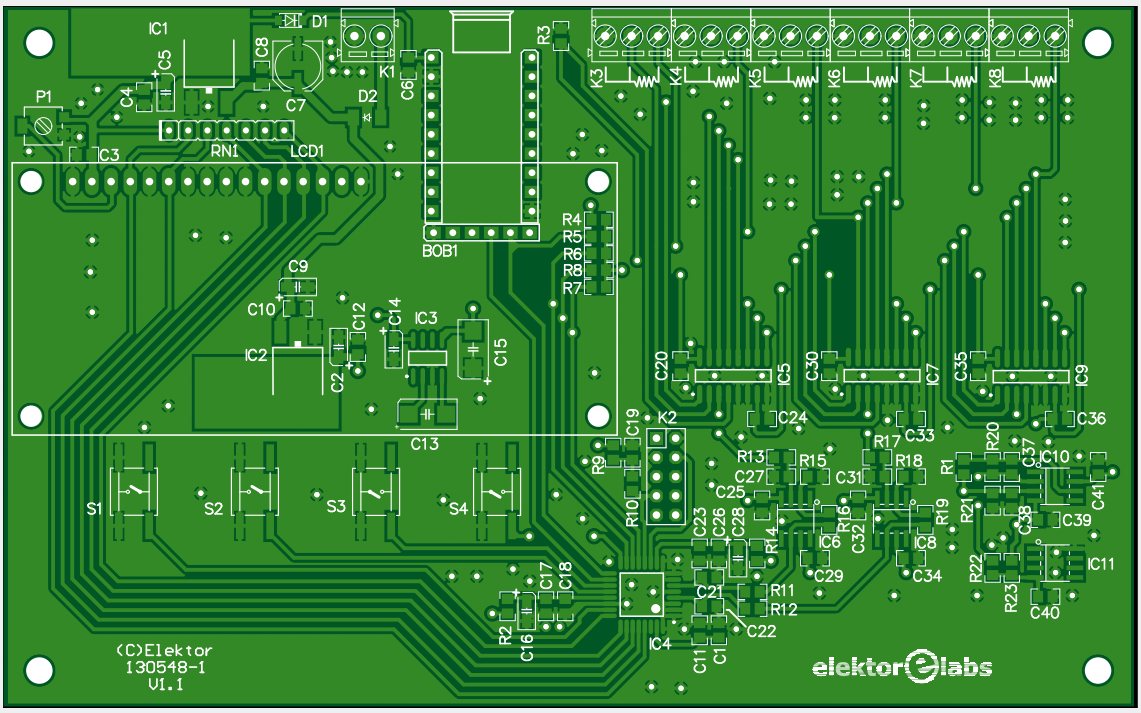
LISTE DES COMPOSANTS

Résistances (0805, 1%)
R4, R5, R6, R7, R8 = 10 kΩ
R1, R20, R21, R22 = 10 kΩ 0,1%
R2, R14 = 10 Ω
R3 = 470 Ω 0,01%
R9, R10, R11, R12, R13, R15, R17, R18, R23 = 1 kΩ
R16, R19 = 12 kΩ
P1 = potentiomètre 10 kΩ
RN1 = réseau de résistances, 6×10 kΩ

Condensateurs (0805)
C1, C19 = 1 µF
C2, C5, C9, C14, C16, C28 = 10 µF, 16 V, boîtier A
C3, C4, C6, C8, C10, C11, C12, C17, C20, C21, C22, C24, C25, C26, C27, C29, C30, C31, C32, C33, C34, C35, C36, C38, C39, C40, C41 = 0,1 µF
C7 = 47 µF, 25 V, diam. = 6,3 mm
C13, C15 = 100 µF, 16 V, boîtier C
C18, C23, C37 = 1 nF

Semi-conducteurs
D1, D2 = MBR140SFT1G
IC1 = LD1117DT50TR
IC2 = LD1117DT33TR
IC3 = TC7660EOA
IC4 = C8051F350-GQ, LQFP32
IC5, IC7, IC9 = 4051, SOIC16
IC6, IC8, IC10 = OP07CD
IC11 = LM336M-2.5

Divers
K1 = bornier double, au pas de 3,5 mm
K2 = barrette mâle, à 2×5 broches, au pas de 2,54 mm
K3-K8 = borniers triples, au pas de 3,5 mm
LCD1 = LCD alphanumérique, 2 lignes, 16 caractères, 5 V, avec rétroéclairage
S1-S4 = bouton-poussoir, 6×6,2 mm, CMS
BOB1 = passerelle eBoB USB/série avec connecteurs appropriés
Support + barrette à 16 broches, au pas de 2,54 mm, pour LCD1
Circuit imprimé, réf. 130548-1 (www.elektor.fr)



Le micrologiciel de débogage ajoute à cela :

- « Decr » (S2) : appuyer sur cette touche pendant le démarrage et la maintenir jusqu'à ce que les écrans d'accueil soient passés fait sauter l'étalonnage du CA/N au démarrage. Utilisez « Save » (S4) pour étalonner quand vous le souhaitez.
- « Decr » (S2) et « Incr » (S3) : appuyer simultanément sur ces touches ouvrira la page *debug* où vous verrez les valeurs brutes en 16 bits du CA/N pour les deux signaux « V » (RTD) et « VRW » (Wire). La pression de « Incr » sélectionne le canal suivant. Appuyez sur « Save » (S4) pour revenir au mode normal.

Liens

[1] Page du projet dans le labo d'Elektor : www.elektormagazine.fr/labs/6-channel-temperature-monitor-logger

[2] Programmateur C2 sur GitHub : https://github.com/ElektorLabs/ElektorLabs_Tools

[3] Page de l'article: www.elektormagazine.fr/190118-04

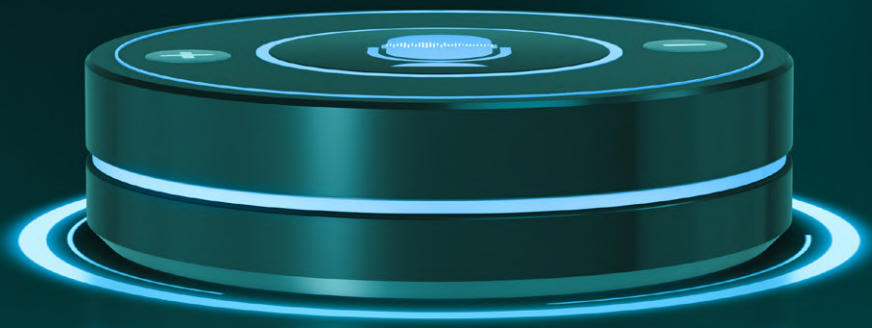
Enregistrement des données

Branchez BOB1 sur la carte (ou un autre convertisseur USB/Série, les connexions sont compatibles avec un câble « FTDI ») et connectez-le à un port USB libre sur un PC. Ce dernier devrait le détecter en tant que port série. Lancez un programme de terminal et configurez-le pour 9600 bauds, huit bits de données, sans parité et un bit d'arrêt (9600n81). Dès que la carte est alimentée, elle va commencer à envoyer des valeurs séparées par des virgules (CSV) dans un format ASCII lisible. La plupart des programmes de terminal pour port série vous permettront d'enregistrer les données dans un fichier pour pouvoir les traiter ultérieurement par exemple dans un tableur.

(190118-04 – version française : Denis Lafourcade)

INTELLIGENCE ARTIFICIELLE

créer un assistant vocal 100% privé avec Snips



ticket d'entrée pour le monde de l'Intelligence Artificielle avec un Raspberry Pi

Mennad Yami et **Robin Guignard-Perret** (Paris)

Snips est une plateforme vocale pour les objets connectés. Elle permet aux développeurs et aux fabricants d'objets d'ajouter un assistant vocal à leurs produits. Votre Raspberry Pi ne vous obéit pas encore au doigt et à l'œil, mais grâce à Snips, il sera à vos ordres. Cet article vous explique comment ajouter ces « oreilles » à un Raspberry Pi.

Créée en 2013, la société Snips souhaite installer un assistant d'intelligence artificielle dans chaque objet, afin de rendre la technologie si intuitive qu'elle s'effacera en arrière-plan.
La technologie Snips est unique, car tout s'exécute localement sur l'objet auquel parle l'utilisateur, ce qui signifie qu'aucune donnée n'est jamais envoyée dans le *cloud*. C'est un concept qui garantit la confidentialité et la continuité du fonctionnement, même en cas de panne d'internet, ce qui fait de Snips une des premières technologies vocales à suivre les normes générales sur la protection des données.
La plateforme Snips est gratuite pour quiconque souhaite tester notre technologie pour un usage non commercial. Nous avons créé une communauté de plus de 25.000 personnes – la plus grande communauté de développeurs d'applications vocales en dehors de Google et Amazon ! Snips fonctionne sous Linux (Raspberry Pi), Android, iOS, MacOS et Debian. Tout le monde peut découvrir en ligne l'ensemble des outils et de la documentation permettant de créer ses propres applications vocales, voir [1].
Snips est disponible en français, anglais, allemand, japonais, espagnol et italien.

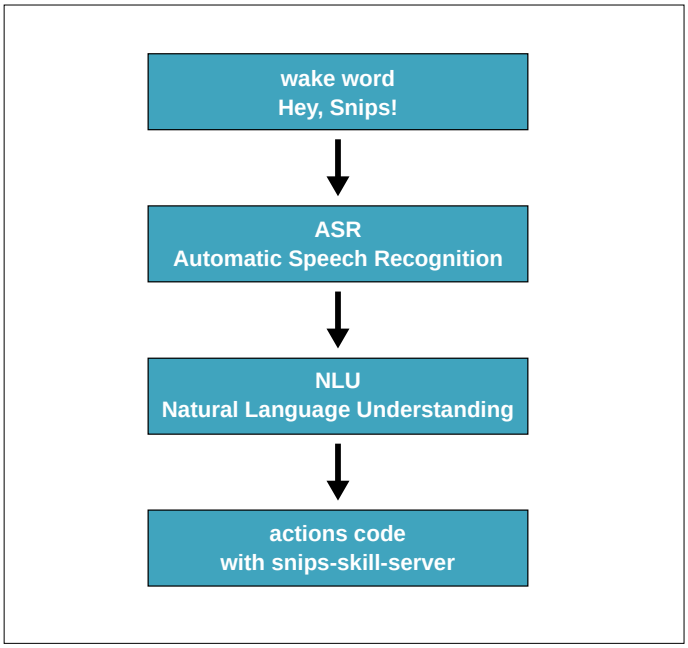


Figure 1. Vue d'ensemble des briques nécessaires à la reconnaissance vocale.

Fonctionnement de nos modèles sur des systèmes embarqués

Le principe de *privacy-by-design* (en clair, « prise en compte du respect de la vie privée dès la conception ») met la confidentialité au cœur du processus de création du système. Cette confidentialité est d'autant plus cruciale que l'on déploie des assistants vocaux dans des contextes très variés, dont la sphère privée des utilisateurs. Ce choix garantit la protection des utilisateurs contre toute utilisation abusive, présente ou future, de leurs données. Chez Snips, le *privacy-by-design* se traduit par l'absence de toute communication de données d'utilisateur vers le *cloud*.
Lors de la conception de la plateforme vocale Snips, la portabilité et la bonne gestion des ressources ont été des priorités. L'inférence intégrée de la plateforme fonctionne sur du matériel IoT commun aussi léger que le Raspberry Pi 3, une carte populaire parmi les développeurs. D'autres cartes Linux sont également prises en charge, et le SDK Snips pour Android fonctionne avec les périphériques dotés d'Android 5 et du processeur ARM, alors que le SDK iOS cible iOS 11 ou une version ultérieure. Pour des raisons d'efficacité et de portabilité, les algorithmes ont été ré-implémentés chaque fois que nécessaire en Rust (langage de programmation compilé de Mozilla Research).

Les assistants vocaux

Les assistants vocaux modernes peuvent se décomposer en plusieurs briques essentielles : la détection d'un mot clé (ou *wakeword*), la reconnaissance de la parole (*Automatic Speech Recognition*, ASR), la compréhension du langage naturel (*Natural Language Understanding*, NLU), et du code d'action qui va s'exécuter dès qu'une consigne est détectée.
Le *wakeword* est le mot clé qui permet d'indiquer à votre assistant le début d'une requête. Un assistant Snips démarre son écoute active dès qu'il entend « Hey, Snips! ». Il commence alors à retranscrire les paroles de l'utilisateur, c'est le travail de la seconde brique qui est alors à l'œuvre : l'ASR.
La brique **ASR** tente de transformer les signaux sonores captés par le micro en texte intelligible par les humains. Lorsque l'utilisateur prononce « Hello world », le signal sonore enregistré traverse d'abord un premier modèle de *deep learning*, un modèle acoustique, qui le convertit en phonèmes, ces unités de représentation du son utilisées par les linguistes. « Hello World » sera d'abord perçu sous la forme « hɛˈləʊ wɜːld ». Par la suite, un modèle spécifique à la langue de l'assistant est appliqué afin de transposer cette phonétique en vocabulaire que nous utilisons, c'est le modèle de langue. Si l'assistant est configuré en anglais, « Hello world » sera correctement retourné puis envoyé à la brique suivante, le NLU.
Le **NLU** permet d'extraire du sens d'un texte. Un assistant vocal étant conçu pour répondre à un éventail de requêtes limitées, le travail du NLU est d'étiqueter une phrase avec une intention. Ces intentions viennent souvent avec des paramètres qu'il convient d'isoler pour pouvoir traiter la requête correctement. Par exemple, lorsque l'utilisateur a prononcé « Hey, Snips! Allume la lumière dans la chambre de Paul en rouge » et que le NLU reçoit la transcription de cette phrase, il distinguera l'intention « allumer la lumière » avec comme paramètres le lieu « chambre de Paul » et la couleur « rouge ». Ce sont ces informations qui pourront être redirigées vers le code d'action.
La brique finale est donc le **code d'action**, qui attend, par le biais d'une boucle infinie, qu'une intention soit détectée afin de s'exécuter, en prenant en compte des paramètres reçus. Ce code est spécifique à chaque application et chaque appareil, et peut réaliser n'importe quelle action *scriptable*. Il peut servir à faire des appels sur des API distantes, commander un objet connecté, faire parler l'assistant vocal, ou tout ça à la fois. Le *wakeword*, l'ASR et le NLU servent de sens cognitifs à votre assistant, le reste est donc géré par le code d'action (voir résumé de la **figure 1**).
Les autres assistants vocaux du marché (Google Home, Alexa, Siri) hébergent leur moteur de détection de *wakeword*, leur ASR et leur NLU dans le *cloud*. Leurs modèles, très larges, nécessitent de trop grosses puissances de calcul pour de la technologie embarquée. Ce n'est pas le cas de Snips. Les modèles

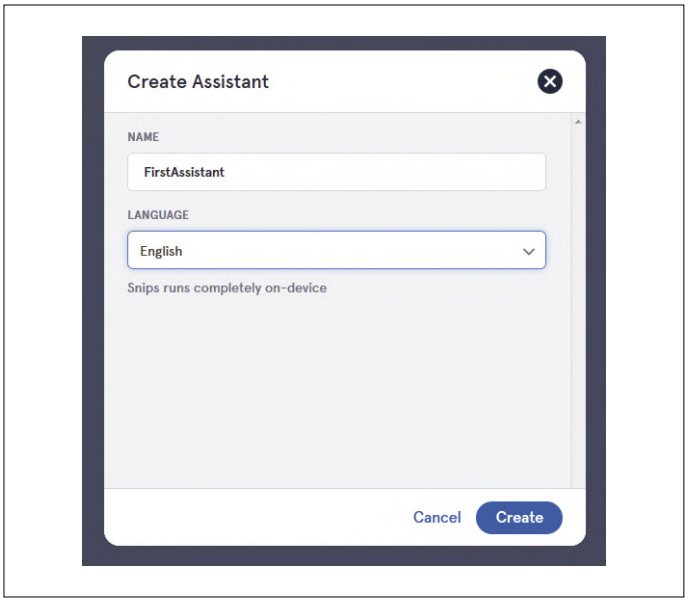


Figure 2. Création d'un assistant sur la console en ligne de Snips.

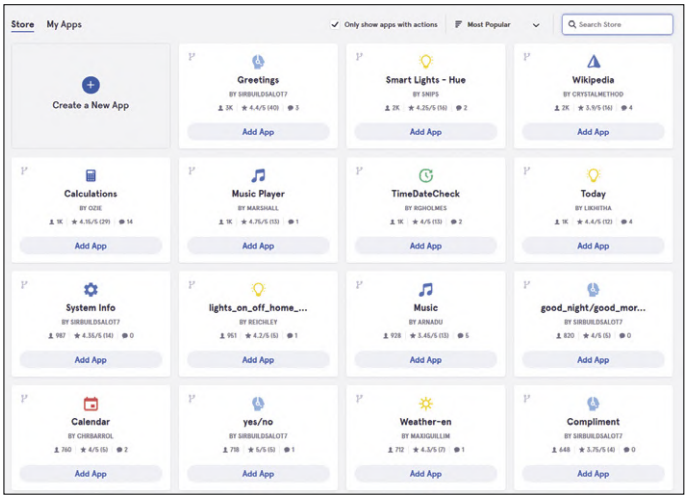


Figure 3. Exemples d'*apps* disponibles sur la console.

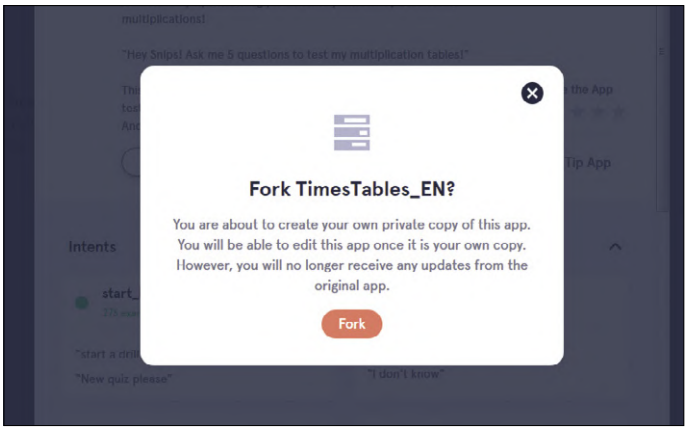


Figure 4. La fonction *fork* permet de créer une copie d'une *App* existante pour la modifier (ce message apparaît automatiquement quand on clique sur *Edit*).

sont créés au cas par cas, pour chaque assistant de chaque utilisateur, et sont spécifiques aux tâches pour lesquelles ils sont conçus. C'est ce qui permet de créer, grâce à Snips, des assistants ultra légers, capables de tourner sur un Raspberry Pi Zero ou 3, ou encore des assistants 100% *offline*. Tout cela est réalisé sans le moindre sacrifice du côté de la performance de la reconnaissance vocale.

La plateforme Snips

La plateforme Snips est l'environnement qui rassemble tous les composants nécessaires au fonctionnement d'un assistant vocal. C'est elle qui, en l'installant, permet à chacun de profiter de son propre assistant vocal, respectueux de la vie privée, sur Linux, MacOS, iOS et Android. Parmi ses éléments, aux côtés du détecteur de *wakeword*, de l'ASR et du NLU, on trouve le **Dialog Manager** en charge de la coordination et de la succession des tâches présentées. Le **TTS**, ou « Text To Speech », s'attèle à la synthèse vocale, pour d'éventuelles réponses. Il reçoit du texte et le lit à haute voix, en adaptant son accent à la langue de l'assistant. Enfin, le **snips-skill-server** est le composant chargé d'exécuter le code d'action.

La communication interne à la plateforme repose sur le **protocole Hermes**, créé par Snips autour du protocole MQTT, léger et rapide. Grâce à Hermes, il est facile de personnaliser le fonctionnement de la plateforme pour par ex. communiquer avec des satellites de votre appareil principal, faire dire une phrase au TTS, ou commander l'un des composants de la plateforme séparément.

La plateforme Snips, une fois installée sur un périphérique, peut accueillir un assistant. C'est sur la console web de Snips [1] que l'on crée cet assistant (**fig. 2**). La console permet de définir la langue de l'assistant, son champ de compréhension et ses fonctionnalités (ou *Apps*).

Par exemple, la plateforme, installée sur votre RPi, pourrait accueillir un assistant en français possédant une *App* pour le contrôle de la lumière ainsi qu'une *App* météo.

Créer et installer son assistant

Passons maintenant à la création de votre assistant personnel, à déployer sur un RPi. Votre RPi doit d'abord accueillir la plateforme Snips. Pour faciliter son installation et la manipulation des assistants, nous avons créé **SAM** (*Snips Assistant Manager*), un outil de déploiement simple d'utilisation, en ligne de commande. SAM est un programme pour votre ordinateur, qui se connecte à votre RPi par SSH. Les deux appareils doivent donc être sur le même réseau, pensez à connecter votre RPi au réseau wifi ou à votre ordinateur par Ethernet. Par ailleurs, votre machine principale doit supporter *Node.js* (environnement d'exécution JavaScript) et *npm* (gestionnaire de paquets de Node.js) pour se servir de SAM. Un `sudo npm install -g snips-sam` fera alors l'affaire pour installer l'outil. Ensuite la commande `sam devices` qui liste tous les RPi présents sur le réseau vous permettra de détecter le vôtre. Lorsque vous avez obtenu l'adresse IP ou le *hostname* de votre appareil, connectez-le à SAM avec la commande `sam connect <adresseIP/hostname>`. Enfin, `sam init` lance l'installation de la plateforme Snips. Pour connecter votre micro et votre sortie audio, utilisez `sam setup audio`.

Console Snips

Ensuite rendez-vous sur [1] pour créer un compte, puis un nouvel assistant. Cet assistant pourra disposer d'une ou plusieurs

fonction(s), vous pouvez en choisir parmi les *Apps* disponibles dans l'*App store* (météo, livre de recette, contrôle des lampes Philips Hue... voir **fig. 3**) ou construire les vôtres. Vous pouvez également personnaliser une *App* de l'*App store* afin d'en produire une version unique grâce à la fonction *fork* (**fig. 4**).
Une *App* capture des intentions, ou *intents*, grâce à l'entraînement du modèle de NLU. Elle en capture également les paramètres, ou *slots*, s'il y en a. Dans le cas d'une *App* calendrier, lorsque l'utilisateur exprime « Ajoute un évènement *anniversaire* pour le 13 novembre », la demande est aiguillée par le NLU vers l'*intent* « addEvent » avec comme *slots* un nom d'évènement « anniversaire » et une date « 13 novembre ». Les équipes de Snips ont déjà créé certains types de *slots* courants, tels que les dates, pour faciliter leur détection.
Le **listage 1** montre à quoi ressemblent les données envoyées par le NLU au code d'action sous format JSON.
Votre assistant a besoin de données d'apprentissage pour entraîner ses réseaux de neurones. Vous pouvez créer vos propres exemples grâce à la console. Pour augmenter les performances du NLU, vous pouvez identifier la position et la nature des *slots* dans une phrase en les surlignant à la souris. Par ailleurs, plus les exemples sont nombreux et variés, meilleures seront les performances de votre assistant. La console Snips vous propose également un service payant de production de données d'entraînement.

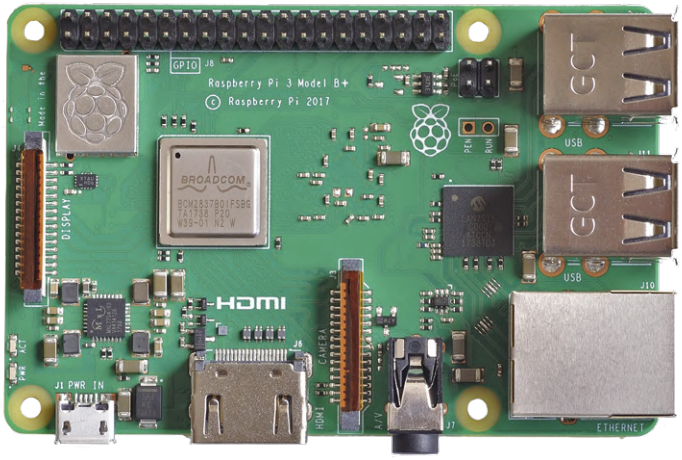
N'oubliez pas de sauvegarder, la console mettra alors à jour votre assistant. Vous pouvez utiliser l'interface de test à droite de la console pour observer les réponses de votre assistant aux requêtes orales ou écrites. Vous pouvez maintenant installer votre assistant sur votre RPi, à l'aide de SAM. La commande `sam login` vous permet de vous connecter sur votre compte utilisé dans la console. Vous pouvez ensuite exécuter `sam install assistant` et sélectionner votre assistant pour l'installer.

Action

Le code d'action est composé de *callbacks* qui s'exécutent lorsque l'*intent* qui leur correspond est détecté. Vous pouvez programmer dans n'importe quel langage, cependant Python et Javascript disposent d'une bibliothèque Hermes qui simplifie les communications internes avec la plateforme Snips. Ces deux langages sont donc recommandés (mais pas obligatoires). Vous pouvez exécuter votre code manuellement, mais le *snips-skill-server* peut également le faire automatiquement au démarrage du RPi. Pour cela, créez un répertoire au nom de votre choix dans `/var/lib/snips/skills/`, et placez votre script dans ce répertoire. Votre script doit alors avoir un nom commençant par *action-* et être exécutable. Son rôle est d'être à l'écoute des *intents*, via une boucle infinie. Il est possible de lier une *App* avec un répertoire Github, et laisser SAM copier vos fichiers au bon endroit lors de l'installation de l'assistant. Le dépôt doit être public, vous pourrez alors l'ajouter dans l'onglet *Actions* de la console. Vous pourrez y placer, en plus de votre fichier d'action, un fichier *setup.sh* qui sera exécuté lors de l'installation, par ex. pour compiler votre projet.

Exemple d'assistant

Passons à la pratique avec la construction d'un premier assistant. Cet assistant vous fera réviser les tables de multiplication via un jeu de questions-réponses. Depuis la console, créez un nouvel assistant en anglais nommé « FirstAssistant », puis



Listage 1. Données envoyées par le NLU au code d'action (format JSON).

```
"input": "ajoute un anniversaire pour le treize
   novembre",
"intent": {
"intentName": "userName:addEvent",
"probability": 0.9662896
},
"slots": [
{
"rawValue": "anniversaire",
"value": {
"kind": "Custom",
"value": "anniversaire"
},
"range": {
"start": 10,
"end": 22
},
"entity": "Noms communs_fr",
"slotName": "summary"
},
{
"rawValue": "pour le treize novembre",
"value": {
"kind": "InstantTime",
"value": "2019-11-13 00:00:00 +00:00",
"grain": "Day",
"precision": "Exact"
},
"range": {
"start": 23,
"end": 46
},
"entity": "snips/datetime",
"slotName": "start_datetime"
}
]
}
```

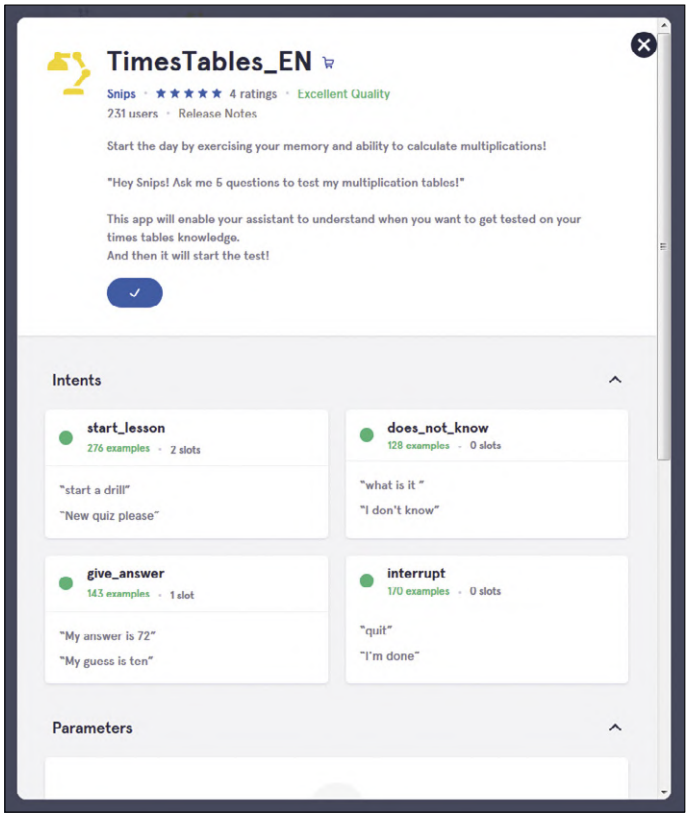


Figure 5. L'*App TimesTables* est disponible en ligne. Il faut la sélectionner puis cliquer sur *Add 1 App* en bas à droite de la fenêtre pour l'ajouter.

une *App* dénommée « TimesTables ». [Note de la rédaction : vous trouverez dans l'*App Store* l'*App* « TimesTableEN » plus évoluée que celle décrite ici, **fig. 5**)]. Le joueur doit pouvoir lancer le jeu, puis répondre aux questions ou les passer lorsqu'il n'a pas la réponse. Nous créons là nos trois *intents* baptisés `start_quiz`, `gives_answer` et `doesnt_know`. Nota : le code d'action recevra le nom de l'*intent* au format *userName:intentName* (par ex. `snips:gives_answer`).

- *intent* `start_quiz` : se limite à lancer le jeu, et n'a donc pas besoin de *slot* pour ajuster le comportement de votre assistant. Il doit comprendre des phrases du type « let's start the game » ou « I want to play ». Il est important d'ajouter des exemples de phrases nombreux et variés pour rendre l'*intent* robuste, ajoutons « I feel like playing the game » ou « let's play » aux exemples avant de sauvegarder (**fig. 6**).

- *intent* `gives_answer` : doit permettre de comprendre une réponse qui est variable. Un *slot* est donc le bienvenu, on peut le nommer `answer` et lui donner le type prédéfini `number`. Dans les phrases d'exemples, n'oubliez pas de surligner les nombres s'ils ne le sont pas automatiquement (**fig. 7**). Pour cela il faut sélectionner le nombre avec la souris, puis cliquer sur *Select a slot* puis sur *answer*. Des réponses telles que « forty-nine » ou « I think it's twelve » sont attendues. Les données au format JSON reçues par votre code d'action comporteront bien la valeur du nombre exprimé par l'utilisateur parmi ses *slots*.

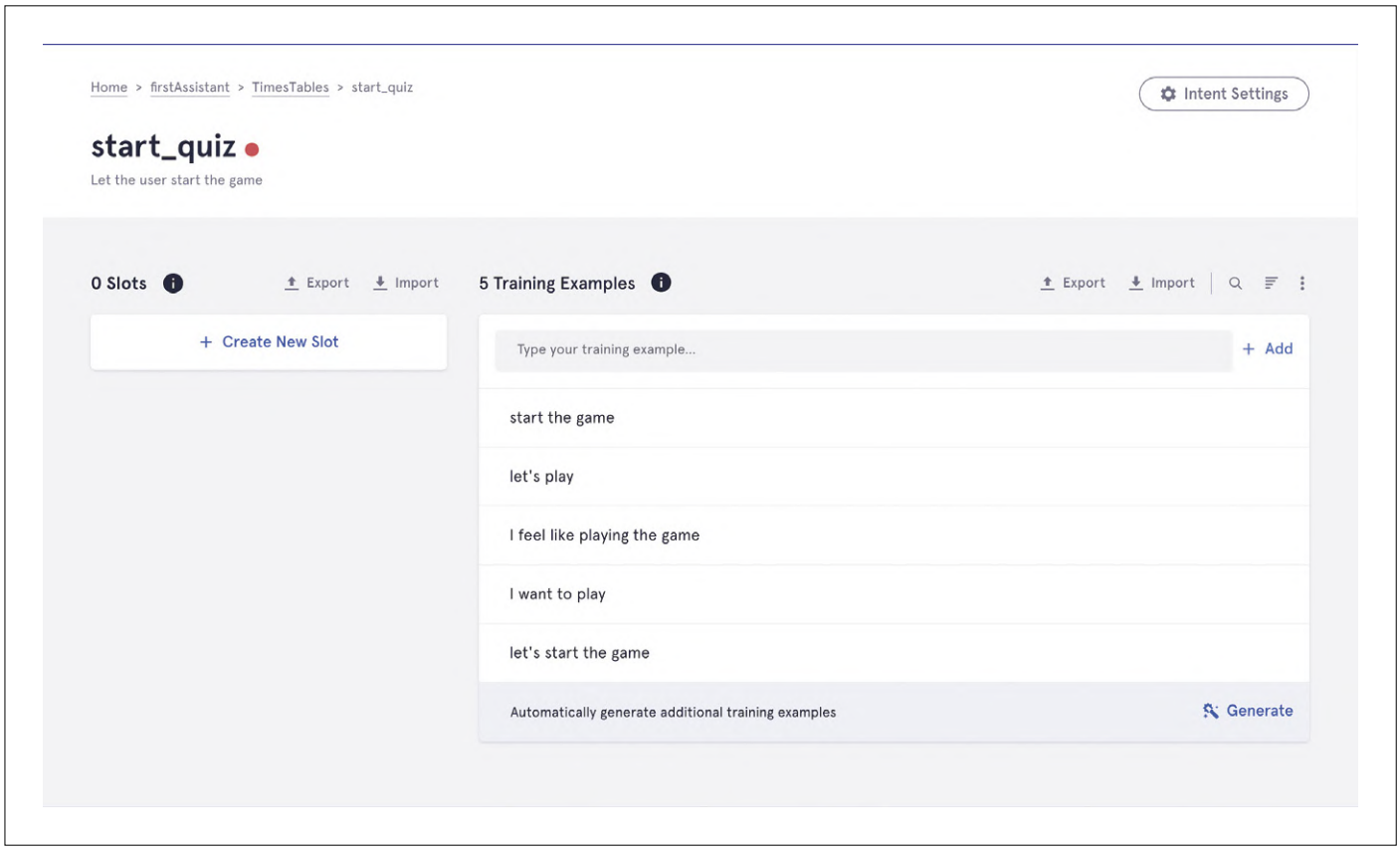


Figure 6. Formules possibles enregistrées dans l'*intent* `start_quiz` pour lancer le jeu. Il s'agit d'une version en anglais, mais vous pouvez créer une version française de cette *App*.

- *intent* `doesnt_know` : suit la même logique que `start_quiz` puisqu'il ne nécessite pas de paramètre. Il suffit donc de donner suffisamment d'exemples pour entraîner les modèles de votre assistant. « I don't know », « I have no idea » ou « I can't get the answer » sont des exemples pertinents.

L'assistant est prêt à être installé grâce à SAM, mais avant cela, nous devons écrire le code d'action sous l'onglet *Actions*. Ici nous avons choisi un script écrit en Python, baptisé `action-timestable.py` (téléchargeable en [2]), nous le plaçons dans le dossier `/var/lib/snips/skills/timestables/`.

Code d'action

Pour réaliser le code de l'action, nous utilisons la bibliothèque `hermes_python`. La première étape est de l'importer, ainsi que la bibliothèque `random` (pour produire les nombres des questions).

Définitions

```
#!/usr/bin/env python2
# coding: utf-8

from hermes_python.hermes import Hermes
import random
```

La combinaison « #! » (1ère ligne) est nécessaire pour que le *snips-skill-server* puisse exécuter le code de votre assistant. Maintenant nous définissons les variables globales de notre action.

```
MQTT_HOST = "localhost"
MQTT_PORT = "1883" # Standard MQTT port
MQTT_URL = "{}:{}".format(MQTT_HOST, MQTT_PORT) # ==>
   "localhost:1883"

SNIPS_USER_NAME = "snips"
INTENT_STARTS_QUIZ = "{}:start_lesson".format(SNIPS_
   USER_NAME) # ==> "snips:start_quiz"
INTENT_GIVES_ANSWER = "{}:give_answer".format(SNIPS_
   USER_NAME) # ==> "snips:gives_answer"
INTENT_DOESNT_KNOW = "{}:does_not_know".format(SNIPS_
   USER_NAME) # ==> "snips:doesnt_know"

sessions_states = {}
```

Les trois variables `MQTT_...` définissent le point d'accès à notre serveur MQTT, dans notre cas on se connecte à *localhost*, car le code sera exécuté sur le RPi qui accueille Snips.
Les trois variables `INTENT_...` définissent les noms de vos *intents* préfixés du nom de l'utilisateur qui a créé l'assistant sur [1]. Il faudra le remplacer par votre nom d'utilisateur.
La variable `sessions_states` permettra à l'assistant de se rappeler de la question posée à l'utilisateur.

Lancer le jeu

Ensuite nous définissons la fonction à appeler quand l'utilisateur demande à l'assistant vocal de lui faire réviser les tables de multiplications.

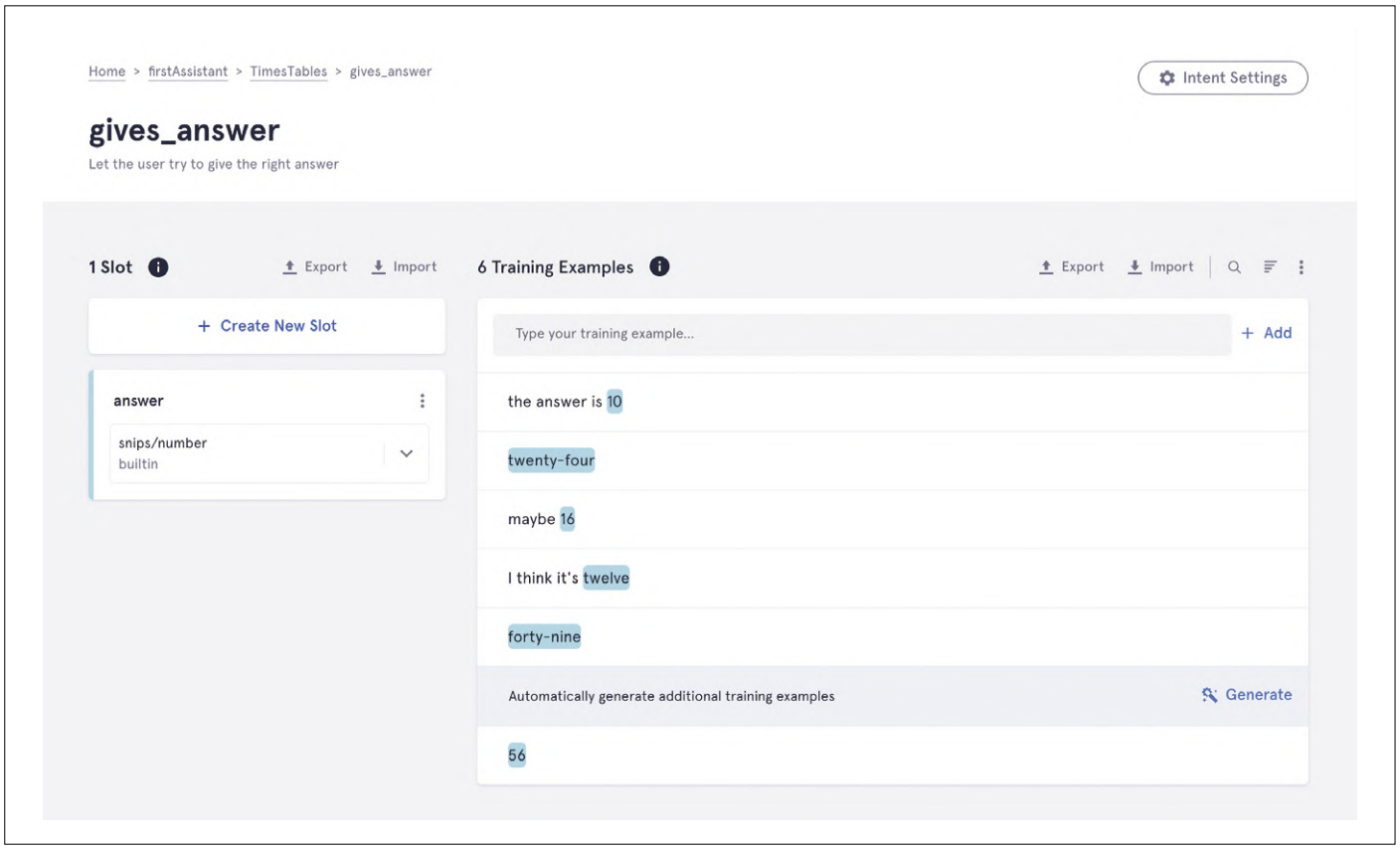


Figure 7. Dans l'*intent* `gives_answer`, les nombres sont surlignés pour le *slot* de type *Answer*.

```
def starts_quiz(hermes, intent_message):
session_id = intent_message.session_id

# Generating question
a = int(random.random() * 10 + 1)
b = int(random.random() * 10 + 1)
tts = "What's {} times {} ?".format(a, b)
sessions_states[session_id] = [a, b]

# Define what the assistant is able to understand
intent_filter = [INTENT_DOESNT_KNOW,
   INTENT_GIVES_ANSWER]

hermes.publish_continue_session(session_id, tts,
   intent_filter)
```

Chaque fonction associée à un *intent* recevra deux objets en paramètres : `hermes` et `intent_message`.

La première étape pour construire un dialogue avec Snips est de récupérer l'» identifiant de la session de dialogue « (`session_id`). Cet identifiant permet de distinguer le dialogue en cours de manière unique ; c'est important, car plusieurs personnes peuvent utiliser votre assistant en même temps (grâce aux satellites, voir https://docs.snips.ai/articles/platform/satellites). Une session de dialogue commence quand un utilisateur prononce le *wakeword* et se termine quand votre code envoie un `end_session`.
Ensuite on crée une question aléatoire sur les tables de 1 à 10 et on stocke la question dans notre dictionnaire `sessions_states`. Cela nous permettra de vérifier la réponse de l'utilisateur.
Enfin on peut publier un évènement pour continuer le dialogue en répondant à l'utilisateur et en écoutant sa réponse. Pour cela, on exécute `hermes.publish_continue_session(...)` avec comme premier paramètre l'identifiant de la session à continuer. Le deuxième paramètre est la phrase/question que l'assistant va poser à l'utilisateur (tts = *text to speech*). Le dernier paramètre est l'`intent filter` qui permet à l'assistant de savoir que l'utilisateur va répondre avec une phrase présente dans un des deux *intents* sélectionnés.

Répondre à la question
Ici les deux *intents* sont `INTENT_DOESNT_KNOW` et `INTENT_GIVES_ANSWER` ce qui veut dire que l'utilisateur ne peut que *donner une réponse* **ou** *dire qu'il ne sait pas*.
Maintenant il faut définir la fonction à appeler quand l'utilisateur donne une réponse.

```
def gives_answer(hermes, intent_message):
session_id = intent_message.session_id

# get answer from user
answer = intent_message.slots.get("answer")
answer = None if answer is None else answer.first().
   value

# defines TTS
tts_good = "Good answer !"
tts_bad = "Wrong answer ..."
tts_no_answer = "I'm sorry I didn't get your answer"
tts = ""
# Check user's answer
if answer is None:
tts = tts_no_answer
elif answer == sessions_states[session_id][0] *
   sessions_states[session_id][1]:
tts = tts_good
else:
tts = tts_bad

hermes.publish_end_session(session_id, tts)
```

Les deux lignes d'initialisation de la variable `answer` permettent de récupérer la valeur (sous forme d'un nombre entier) de la réponse de l'utilisateur. La deuxième ligne vérifie que l'utilisateur a bien donné une réponse avant d'accéder à la valeur. Les lignes `tts_... =` définissent les phrases que l'assistant vocale répondra dans les trois différents cas :

1. l'utilisateur a donné la bonne réponse
2. l'utilisateur a donné une mauvaise réponse
3. la réponse de l'utilisateur n'a pas été comprise par l'assistant.

Le `if` permet de déterminer dans lequel de ces trois cas nous sommes.
Enfin on publie l'évènement `end_session` avec pour paramètre le `session_id` ainsi que la phrase que l'assistant va dire avant de fermer cette session de dialogue.

La fonction appelée quand l'utilisateur dit ne pas connaître la réponse est assez simple. On publie juste un end_session pour fermer le dialogue avec un TTS prédéfini.

Liens

[1] Console, outils et documentation de Snips : https://console.snips.ai
[2] Code python de l'App sur la page de l'article : http://www.elektormagazine.fr/180733-01
[3] Banc d'essai « Snips – reconnaissance vocale pour le Raspberry Pi » : www.elektormagazine.fr/news/banc-d-essai-snips-reconnaissance-vocale-pour-le-raspberry-pi

```
def doesnt_know(hermes, intent_message):
session_id = intent_message.session_id
tts = "Ok, at least try next time !"

hermes.publish_end_session(session_id, tts)
```

C'est terminé

La fonction suivante est appelée une fois qu'une session aura fini de se fermer après un appel à `hermes.publish_end_session(...)`.

```
def session_ended(hermes, session_ended_message):
del sessions_states[session_ended_message.session_id]
```

Ici nous supprimons l'élément du dictionnaire qui était associé à la session qui vient de se fermer.

Arrivée d'Hermes

La dernière étape est de connecter tout ça ensemble.

```
with Hermes(MQTT_URL) as h:

# Subscribe to intents
h.subscribe_intent(INTENT_STARTS_QUIZ, starts_quiz)\
.subscribe_intent(INTENT_GIVES_ANSWER, gives_answer)\
.subscribe_intent(INTENT_DOESNT_KNOW, doesnt_know)
# Subscribe endSession
h.subscribe_session_ended(session_ended)

# Start Listening
h.loop_forever()
```

La première ligne crée une instance de l'objet `Hermes` dans un gestionnaire de contexte. On notera que son argument `MQTT_URL` est l'url défini au début du fichier.
Les trois lignes après `# Subscribe to intents` définissent qu'à un *intent* donné (défini par nos variables globales), on associe une fonction donnée. C'est grâce à ces trois lignes que Snips saura quelle fonction appeler pour chacun des *intents* reçus.
Ensuite de la même manière on définit que la fonction `session_ended` sera appelée quand une session de dialogue sera finie.
Enfin on peut exécuter la fonction qui va se charger d'écouter les différents évènements Snips et d'appeler les fonctions associées à chaque *intent*.
En résumé, le code d'action présente la structure suivante :

- importation des bibliothèques
- définition des variables
- définition des actions pour chaque *intent*
- initialisation d'un contexte (*Hermes*)

Nous espérons que ce premier article sur l'assistant vocal Snips vous aura mis le pied à l'étrier. Vous êtes maintenant prêt à ajouter une commande vocale à nombre de vos projets sur Raspberry Pi. Pour vous faciliter la tâche, n'oubliez pas de vérifier dans l'*App Store* Snips si quelqu'un a déjà conçu une *App* proche de vos besoins.

(180733-01)

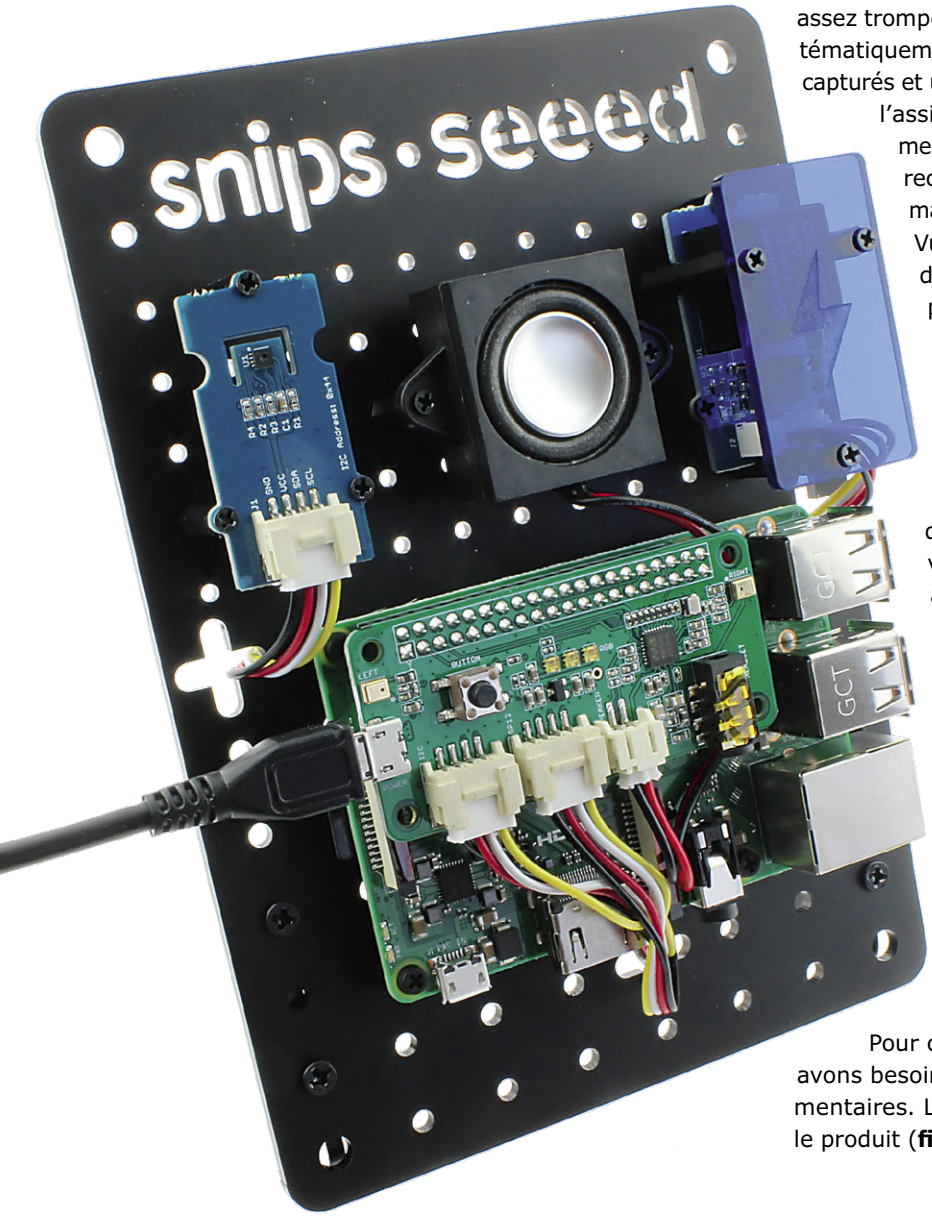
Snips

reconnaissance vocale pour le Raspberry Pi

Mathias Claussen

Les assistants vocaux comme Alexa, Google Home ou Siri sont aujourd'hui devenus monnaie courante. Pour autant, la reconnaissance de la parole n'est pas traitée localement, mais dans le nuage, chez le fournisseur – ce qui soulève un certain nombre de questions à propos de la confidentialité des données. Il existe cependant d'autres solutions, notamment Snips, système de reconnaissance vocale qui fonctionne intégralement sur un matériel local comme le RPi. Nous avons examiné pour vous en détail un kit de reconnaissance vocale et découvert comment l'utiliser pour développer nos propres applications...

« Thé, Earl Grey – chaud » – il y a 10 ans, la possibilité pour un ordinateur d'apporter une réponse appropriée à ce type de demande vocale tenait de la pure science-fiction... Mais c'était avant l'apparition d'Alexa, Google Home et Siri. Les assistants vocaux d'Amazon, Google ou Apple peuvent en effet écouter tout ce que nous disons et y répondre (mais pas toujours à bon escient). Si vous essayez d'écrire votre propre application associée à des fonctions de reconnaissance vocale, vous allez rapidement réaliser que les apparences d'un tel assistant sont assez trompeuses. En effet, tous ces systèmes cherchent systématiquement une corrélation entre les échantillons sonores capturés et une liste limitée de modèles de mots-clés. Une fois l'assistant activé, tout ce qui est collecté est directement envoyé aux serveurs du fournisseur respectif. La reconnaissance vocale n'est pas traitée localement, mais quelque part dans le nuage, chez le fournisseur. Vu ainsi, le nuage n'est rien de plus qu'un ensemble de ressources informatiques rassemblées quelque part dans un centre de données. En tant qu'utilisateurs ou développeurs d'applications, nous ne savons rien de ce que deviennent ces informations. Nous n'avons comme ressource que de faire confiance aux fournisseurs pour qu'ils les traitent dans le respect des réglementations applicables en matière de protection des données. C'est ainsi qu'un juge américain a récemment cité un assistant vocal Alexa d'Amazon comme « témoin » dans une affaire de meurtre.



Si vous craignez un effet « Big Brother », vous serez satisfait(e) de savoir qu'il existe des alternatives, comme Snips – logiciel de reconnaissance vocale fonctionnant intégralement hors connexion, qui veille à ce que vos données restent confinées dans les limites de votre réseau domestique. Il ne nécessite ni connexion internet permanente, ni batteries d'ordinateurs pour assurer la reconnaissance vocale. Pour explorer ce produit plus en détail, nous allons nous intéresser au kit d'interaction vocale Snips qui utilise comme système de base le nano-ordinateur Raspberry Pi 3B+.

Pour cette application de reconnaissance vocale, nous avons besoin, outre le RPi, de quelques composants supplémentaires. Le moment est venu d'examiner ce que contient le produit (**fig. 1**).

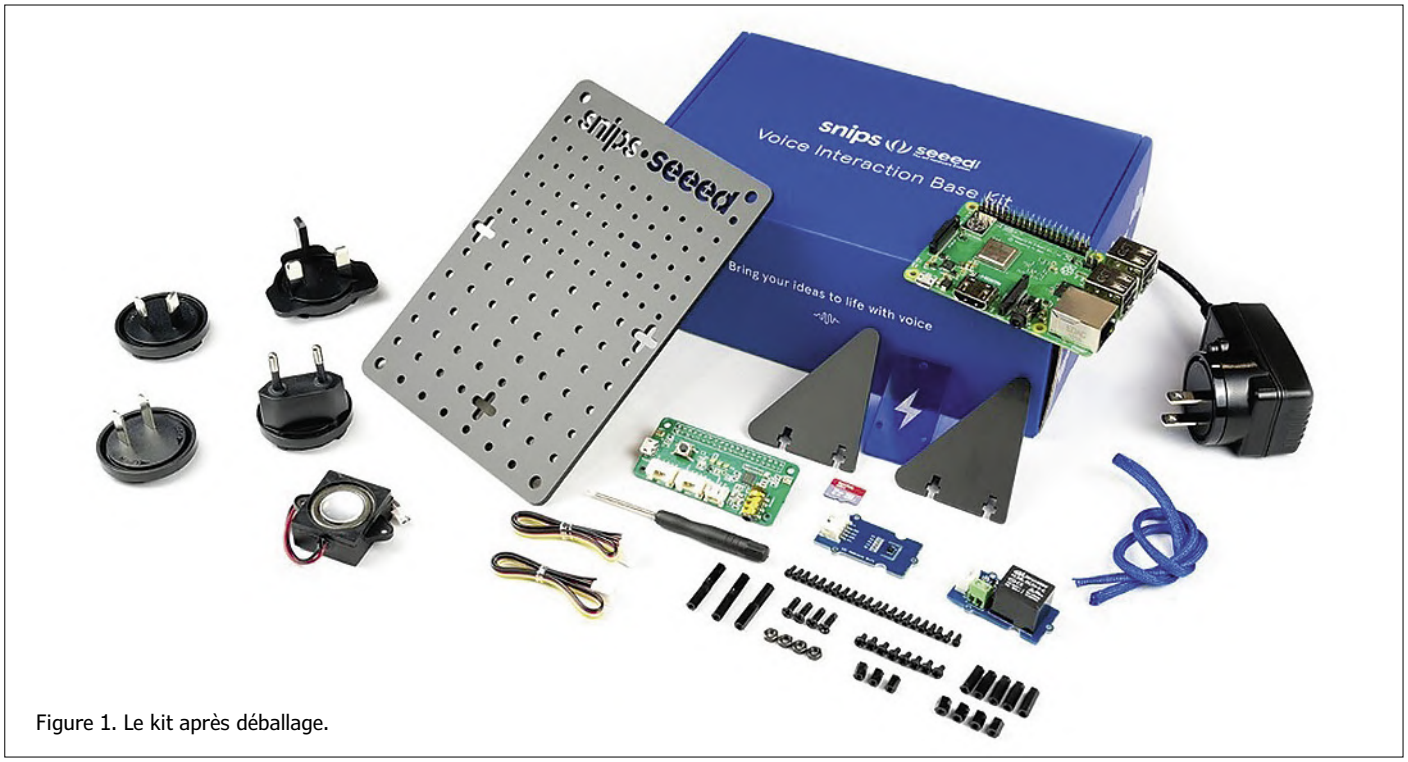


Figure 1. Le kit après déballage.

Le kit

Le kit est livré dans une boîte en carton bleu uni. À l'intérieur, vous trouvez tous les composants, avec le Raspberry Pi 3B+, un adaptateur secteur 5 V 3 A, un relais, un capteur de température et d'humidité, un haut-parleur, un HAT (carte d'extension) ReSpeaker 2-Mics pour RPi, une plaque de montage, une carte SD qui contient le système préinstallé et un guide de démarrage rapide.

Vous avez donc à portée de main tous les éléments nécessaires pour construire votre propre petit assistant vocal. Pour l'assemblage, utilisez les entretoises et les vis appropriées. Malheureusement, le manuel a omis un certain nombre de détails, et il nous arrivera de nous demander à quoi servent certaines vis. Après quelques tâtonnements, nous avons quand même réussi à nous en sortir, même si la carte RPi n'est pas idéalement positionnée pour accéder facilement à la trappe d'accès à la carte SD.

Plaque de montage perforée en acrylique

Le kit contient une plaque de montage en acrylique sur laquelle vous pouvez fixer les différents modules à l'aide de vis autotaraudeuses. La matrice des trous de fixation autorise un certain degré de liberté pour l'implantation des modules.

Capteurs et actionneurs

Le capteur de température et d'humidité est un modèle Sensirion SHT31, monté sur un module Grove. La précision de mesure est de ±0,3 °C pour la température et de ±2 % pour l'humidité.

Le relais est également monté sur un module Grove de Seeed Studio. Ses contacts permettent de commuter des tensions de 250 V_{CA} / 30 V_{CC}, sous 5 A, et offrent la possibilité d'exécuter 100.000 cycles. Même si le relais peut commuter une tension secteur, la carte n'a pas été conçue pour fonctionner dans ce contexte et ne prévoit aucune isolation. Il serait en effet très facile de toucher accidentellement des conducteurs sous tension. Si la tension appliquée à la charge est inférieure au niveau de sécurité en très basse tension (<25 V_{CA} / <30 V_{CC}) et en tenant compte des spécifications du relais, vous pouvez utiliser ce module en toute sécurité pour vos expérimentations.

Un petit haut-parleur est inclus dans le kit, et vous pouvez donc entendre ce que l'assistant vocal vous dit. Comme sa taille le laisse supposer, la qualité du son du haut-parleur n'est pas excellente, mais néanmoins adaptée à sa finalité.

HAT ReSpeaker 2-Mics pour RPi

Le cœur du système — hors RPi — est la carte d'extension HAT ReSpeaker 2-Mics pour RPi, construite autour d'un codec audio WM8960. Malheureusement, il n'est pas pris en charge directement par Raspbian, et le pilote comporte également quelques bogues. Si vous utilisez le HAT avec une image Raspbian nue, il sera nécessaire de compiler le pilote pour le RPi. Vous trouverez en [1] les instructions nécessaires pour configurer manuellement le pilote. Le codec WM8960 comporte un amplificateur de classe D, capable de fournir 1 W sous 8 Ω par canal, et permet de raccorder des microphones. La carte possède donc deux micros ; les connexions nécessaires pour un haut-parleur ou un casque sont également disponibles.

La carte comporte également trois LED RVB (APA102), connectées au Raspberry Pi par l'interface SPI (*Serial Peripheral Interface*). Ces témoins indiquent le moment où l'assistant vocal commence à écouter (ou n'a pas compris). Un bouton-poussoir permet également une interaction avec le système. Vous disposez également de connexions pour le relais, et le capteur de température et d'humidité.

Un port micro-USB situé sur le côté permet de raccorder facilement l'alimentation du RPi. Malheureusement, la connexion de la barrette n'est pas traversante, ce qui empêche l'ajout d'un autre HAT. Il est donc impossible d'utiliser un afficheur SPI-TFT comme interface utilisateur.

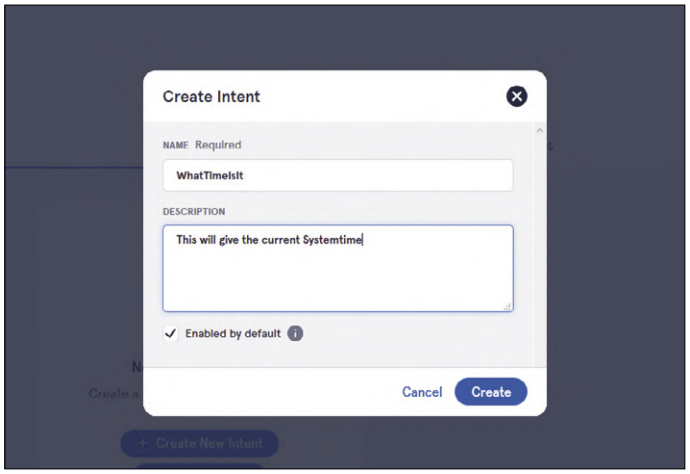


Figure 2. Créer une intention (commande vocale).

Veillez à bien connecter le câble sur le port approprié pour raccorder le relais, et le capteur d'humidité et de température. Une fois tous les éléments assemblés et la carte SD insérée, nous pouvons commencer. Un moniteur HDMI et un clavier USB seront ici très utiles pour pouvoir observer le premier démarrage du RPi. Après lancement, le système est prêt pour un premier essai. Prononcez ceci : « Hey Snips - What's the temperature? » (ou son équivalent français*). Le système devrait vous répondre en vous indiquant la température. Ensuite, dites : « Hey Snips – Turn the relay on. » pour activer le relais. En principe, ces commandes devraient fonctionner directement sans avoir à modifier la configuration. Nous allons maintenant aborder la partie la plus intéressante en découvrant comment réaliser nos propres applications et commandes à reconnaissance vocale.

Configuration

Le RPi doit maintenant pouvoir accéder à votre réseau, et le moyen le plus simple est d'utiliser un câble. Si vous souhaitez le faire en Wi-Fi, vous devez retirer la carte SD du RPi et créer le fichier *wpa_supplicant.conf* sur le lecteur nommé *boot*. Le fichier contient les paramètres suivants :

```
ctrl_interface=DIR=/var/run/wpa_supplicant
GROUP=netdev
update_config=1
country=<your_country_id>
network={
ssid=»<your_ssid>»
psk=»<your_password>»
}
```

Les paramètres *country* (pays), *SSID* et *password* (mot de passe) correspondent aux informations d'identification de votre boîtier Wi-Fi.
Avec l'hôte *snips-base.local,* vous pouvez vous connecter au RPi via SSH et explorer le système. Le logiciel Snips est déjà préinstallé sur le système.
Une fois effectuée l'installation de base du RPi, qui est relativement simple, nous installons *Sam*, interface de ligne de commande pour contrôler et gérer le logiciel Snips sur le Raspberry Pi. Pour installer Sam, nous devons ouvrir un terminal et entrer la commande `sudo npm install -g snips-sam`. Il est maintenant possible de se relier au logiciel Snips sur le RPi avec la commande `sam connect snips-base.local`. La commande `sam status` permet d'obtenir l'état et le numéro de version de chaque composant. Le résultat devrait ressembler à peu de chose près au **listage 1**.
Le logiciel étant encore en cours d'évolution, il est possible qu'une version ou une autre ne fonctionne pas toujours correctement. Pour obtenir la version la plus récente, il suffit d'entrer la commande `sam update`. Le RPi télécharge et installe ensuite la toute dernière version. Enfin, il est nécessaire de saisir la commande `sam update-assistant`. Cela permettra également de mettre à jour les applications, et tous les composants de Snips devraient alors être de nouveau en ligne. Si nous entrons encore la commande `sam status`, tous les composants (hormis *snips-analytics*) fonctionnent et le RPi devrait répondre aux exemples de commandes du tutoriel.

Listage 1. Consultation de l'état et du numéro de version de chaque composant.

```
Connected to device snips-base.local
OS version ........... Raspbian GNU/Linux 9 (stretch)
Installed assistant .. MakerKitBundle_EN
Language ............. en
Hotword .............. hey_snips
ASR engine ........... snips
Status ............... Live

Service status:

snips-analytics ...... 0.60.8 (running)
snips-asr ............ 0.60.8 (running)
snips-audio-server ... 0.60.8 (running)
snips-dialogue ....... 0.60.8 (running)
snips-hotword ........ 0.60.8 (running)
snips-nlu ............ 0.60.8 (running)
snips-skill-server ... 0.60.8 (running)
snips-tts ............ 0.60.8 (running)
```

Construire sa propre application

Après avoir créé un compte en [2], vous pouvez utiliser l'interface web pour créer votre propre application, et mieux encore, soumettre votre assistant à un apprentissage avec vos propres commandes vocales. À titre de petite démonstration, nous demanderons à l'assistant vocal de nous donner l'heure.
Nous commençons par créer un nouvel assistant, que nous appellerons *WhatTimeItIs*. Puisque vous disposez d'un assistant, vous pouvez maintenant créer des applications. Nous allons créer une application appelée « *currentTime* », qui donne l'heure à la demande.
Pour que l'application comprenne une commande vocale, nous devons la soumettre à un apprentissage pour définir une intention (ou *intent* en abrégé) (**fig. 2**). Il existe un certain nombre de modules intégrés pour nous faciliter la vie si nous avons à traiter des nombres ou dates. Puisque nos questions à base de mots-clés n'existent pas sous la forme d'un module prêt à l'emploi, nous devons les créer. Nous souhaitons que le système écoute (et entende) le mot « Time », ainsi qu'un certain

nombre de synonymes.
Maintenant que l'apprentissage est terminé, nous pouvons nous occuper de l'action que notre commande vocale doit déclencher. Nous allons donc sur la zone *Actions* et choisissons *Code snippets* pour les besoins de notre démonstration. Il est possible actuellement d'écrire du code en Python3 et Python2 via l'interface web (**fig. 3**), et de l'exécuter sous la forme d'une « action ». Pour en savoir plus, consultez la documentation de Snips sous [3].
Pour obtenir l'heure, nous utilisons le code Python3 du **listage 2**.
Ce code sera exécuté chaque fois que le mot « Time » ou l'un de ses synonymes sera détecté. En réponse, nous créons une chaîne, qui est ensuite adressée au module logiciel de conversion texte-voix, chargé de donner l'heure vocalement.
Nous avons donc créé notre première application et nous pouvons maintenant la tester. Pour ce faire, il nous faut la transférer sur le RPi. Sur le RPi, il suffit de saisir la commande `sam install assistant` pour installer l'assistant que nous avons créé. Et il ne reste plus qu'à demander l'heure : « Hey Snips, what's the time? »...*

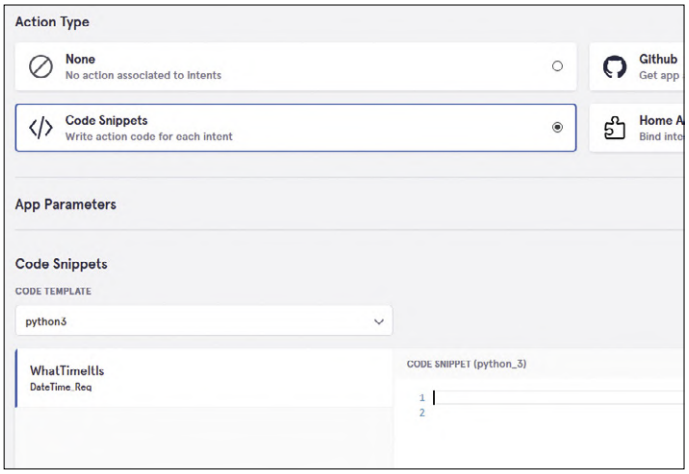


Figure 3. Saisie du code d'exécution de l'action.

En résumé

Ce kit contient tout ce dont vous avez besoin pour découvrir la commande vocale (haut-parleurs, microphones, capteurs et relais). Malheureusement, les signaux produits par les deux microphones intégrés sont de mauvaise qualité et entachés de bruit. La reconnaissance vocale n'en est pas affectée, mais si vous souhaitez enregistrer un nouveau mot-clé pour activer l'assistant vocal à l'aide de la procédure indiquée, vous aurez malheureusement besoin d'un microphone de meilleure qualité. Soyez très attentif(ve) pour l'assemblage du matériel. Une ou deux pages supplémentaires dans le manuel avec des illustrations représentant la mise en place des entretoises auraient été très utiles.
Actuellement, le codec wm8960 n'est pas pris en charge par le noyau RPi, ce qui impose un assemblage manuel pour faire fonctionner la carte son dans la version officielle de Raspbian. Si vous souhaitez créer votre propre image à partir de Raspbian, vous devrez compiler le pilote et incorporer le correctif vous-même. Si vous installez une nouvelle version de Snips, vous devrez mettre de nouveau à jour l'assistant ou le réinstaller.
Une fois familiarisé(e) avec l'interface web Snips.ai, c'est un plaisir de travailler avec le système et d'explorer toutes ses possibilités. Ce que je trouve surtout rassurant à la fin d'une journée, c'est que je suis réellement le seul à avoir accès à toutes mes données...
* L'assistant par défaut est en anglais, mais le produit peut fonctionner en français. Voir en [4]. ⏮

(180208-04)

Listage 2. Code Python de notre application.

```
import time
now = time.localtime()
if len(intentMessage.slots.DateTime_Req) > 0:
    date_or_time = intentMessage.slots.DateTime_Req.first().value # We extract the value from the slot
    result_sentence = "Current time is : {} {} ".format(str(now.tm_hour), str(now.tm_min))
                                        # The response that will be said out loud by the TTS engine.
else:
    result_sentence = "Time is running out"
current_session_id = intentMessage.session_id
hermes.publish_end_session(current_session_id, result_sentence)
```

Liens

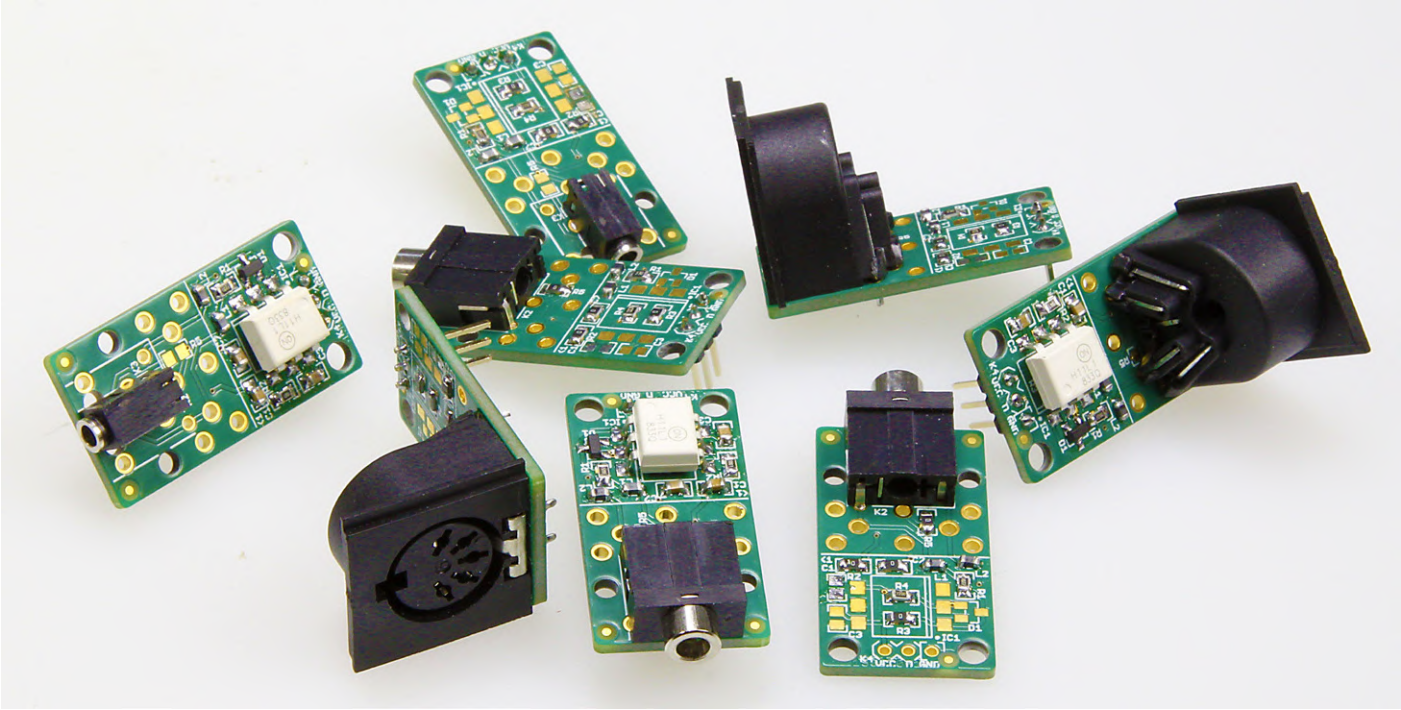
[1] Pilote audio pour le Raspberry Pi : http://wiki.seeedstudio.com/ReSpeaker_2_Mics_Pi_HAT/
[2] Page d'accueil de Snips : https://snips.ai/
[3] Documentation de Snips : https://docs.snips.ai/articles/console/actions/actions/snippets
[4] Version française : http://github.com/snipsco/snips-demo-dev-kit#with-assistant-recommend

PROJET DU LABO

carte de liaison d'E/S MIDI

compatible avec les connecteurs DIN et TRS

Clemens Valens (labo d'Elektor)



Début 2019, quelque 35 ans après son introduction, le protocole MIDI (*Musical Instrument Digital Interface*) est passé en version 2.0. Cette nouvelle mouture apporte plusieurs fonctions matérielles et logicielles inédites, dont deux nouveaux connecteurs. Une nouvelle carte d'interface MIDI universelle s'imposait.

Caractéristiques

- Une seule carte pour MIDI In, MIDI Out et MIDI Thru
- Configurable pour les systèmes 5 V et 3,3 V
- Compatible avec les connecteurs DIN, TRS 3,5 mm et TRS 2,5 mm
- Conforme à la norme MIDI 2.0

La spécification MIDI 2.0 n'ayant pas encore été publiée au moment de la rédaction de cet article (avril 2019), je n'entrerai pas dans les détails. Sachez tout de même que la nouvelle norme restera 100 % compatible avec la version 1.0.

Jack a dit : connecte-moi

Durant l'été 2018, la MMA (*MIDI Manufacturers Association*) a publié une *Lettre d'entente relative aux pratiques recommandées* dans laquelle elle reconnaissait la combinaison prise/fiche TRS (*Tip-Ring-Sleeve*, pointe-anneau-manchon) comme étant « le » connecteur MIDI, et a dans le même temps normalisé son câblage avec les traditionnels connecteurs DIN à 5 broches. La spécification relative aux connecteurs TRS (couramment appelés « jack ») est donc, à vrai dire, pré-MIDI 2.0. Ces nouveaux connecteurs n'en forment en fait qu'un seul offrant deux diamètres différents : 3,5 mm et 2,5 mm, ce dernier étant le type recommandé.

Cet ajout à la norme a été motivé par une simple question d'encombrement, la vénérable prise DIN étant devenue trop grosse pour les appareils modernes plats de type ordiphones.

Que du neuf !

Notre carte de liaison (*break-out board*, BoB) peut être équipée d'un connecteur DIN ou d'un TRS de 3,5 ou 2,5 mm. Elle permet en outre l'utilisation de signaux de 3,3 V, autorise une liaison de masse HF et est configurable comme module MIDI In ou MIDI Out (ou MIDI Thru, équivalent à MIDI Out).
La carte (**fig. 1**) comporte plus de composants que n'en possède une interface

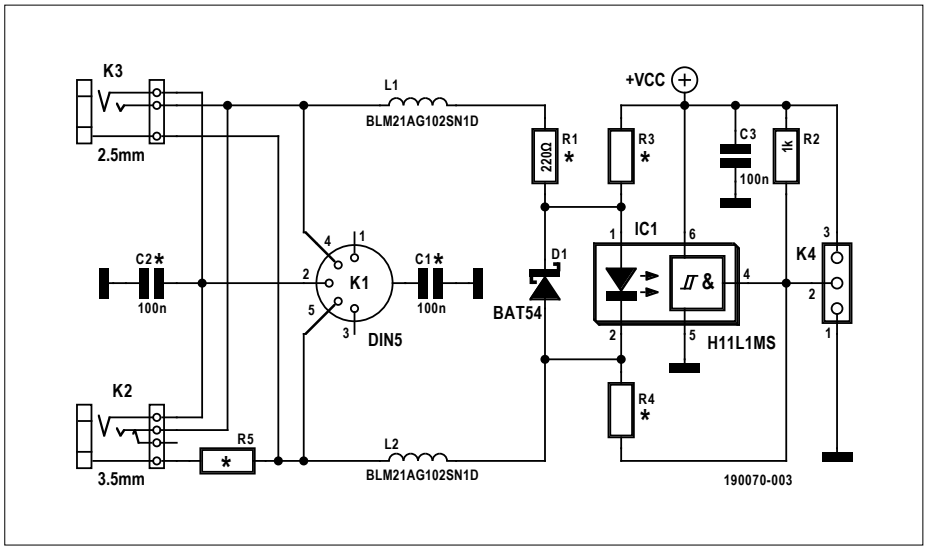


Figure 1. Le schéma du BoB d'E/S MIDI. Un peu pagaille à vos yeux ? C'est le prix à payer lorsque vous combinez plusieurs circuits en un seul circuit configurable.

INFOS SUR LE PROJET

MIDI
TRS DIN BoB
carte de liaison

débutant
connaisseur
expert

env. 1 h

fer à souder

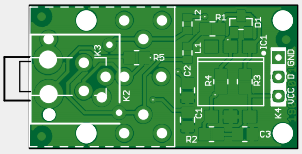
env. 15 €

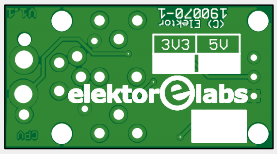
MIDI typique en raison de ses options de configuration et de ses mesures CEM et antiparasites.

Boucles de masse et CEM

La spécification MIDI 1.0 originale stipulait qu'afin d'éviter les boucles de masse entre appareils « *Le blindage de mise à la masse des prises MIDI ne doit pas être connecté à la masse d'un circuit ou d'un châssis.* » Cette prescription a toutefois été revue en 2014. Une connexion optionnelle à la masse entre les blindages des connecteurs MIDI Out et MIDI Thru est désormais autorisée pour améliorer les performances antiparasites/CEM. Les blindages des connecteurs MIDI In peuvent être mis à la masse via un petit condensateur, mais jamais directement. Notre carte le permet via les condensateurs C1 et C2. En configuration MIDI In, C1 et C2 sont soit absents, soit des 100 nF ; en MIDI Out (ou Thru), C1 et C2 peuvent être des résistances de 0 Ω.

LISTE DES COMPOSANTS





MIDI In

Résistances
Toutes CMS 0805
R1 = 220 Ω
R2 = 1 kΩ
R5* = 0 Ω

Condensateurs
Tous CMS 0805
C1, C2, C3 = 100 nF

Inductances
Toutes CMS 0805
L1, L2 = perle de ferrite, 1 kΩ @ 100 MHz

Semi-conducteurs
D1 = BAT54
IC1 = H11L1 ou équivalent

Divers
K1* = prise DIN à 5 contacts, 180°
K2* = prise TRS à 3 contacts, Ø = 3,5 mm
K3* = prise TRS à 3 contacts, Ø = 2,5 mm
K4 = barrette à 3 points, au pas de 2,54 mm
Circuit imprimé, réf. 190070-1 (www.elektor.fr)

MIDI Out / MIDI Thru, 5 V

Résistances
Toutes CMS 0805
R1, R4 = 220 Ω
R3, R5* = 0 Ω

Condensateurs
Tous CMS 0805
C1, C2 = 0 Ω*
C3 = 100 nF

Inductances
Toutes CMS 0805
L1, L2 = perle de ferrite 1 kΩ @ 100 MHz

Divers
K1* = prise DIN à 5 contacts, 180°
K2* = prise TRS à 3 contacts, Ø = 3,5 mm
K3* = prise TRS à 3 contacts, Ø = 2,5 mm
K4 = barrette à 3 points, au pas de 2,54 mm
Circuit imprimé, réf. 190070-1 (www.elektor.fr)

MIDI Out / MIDI Thru, 3,3 V

Résistances
Toutes CMS 0805
R1 = 33 Ω, 0,5 W
R3, R5* = 0 Ω
R4 = 10 Ω, 0,25 W

Condensateurs
Tous CMS 0805
C1, C2 = 0 Ω*
C3 = 100 nF

Inductances
Toutes CMS 0805
L1, L2 = perle de ferrite 1 kΩ @ 100 MHz

Divers
K1* = prise DIN à 5 contacts, 180°
K2* = prise TRS à 3 contacts, Ø = 3,5 mm
K3* = prise TRS à 3 voies, Ø = 2,5 mm
K4 = barrette à 3 points, pas de 2,54 mm
Circuit imprimé, réf. 190070-1 (www.elektor.fr)

Antiparasites HF

Les perles de ferrite L1 et L2 atténuent les interférences HF. Elles peuvent être remplacées au besoin par des résistances de 0 Ω, p. ex. si vous n'avez pas de ferrite en stock.

R3 et R4 sont des résistances de configuration et ne sont requises que pour les applications MIDI Out (et Thru). Pour éviter leur montage accidentel sur un module MIDI In, leur emplacement est identique à celui du photocoupleur IC1.

Un photocoupleur pour MIDI In

Le photocoupleur rapide IC1 assure une bonne qualité de signal et une faible latence entre In et Thru en cas de chaînage de plusieurs appareils MIDI. La diode D1 le protège contre les signaux de polarité inverse.

R2 est la résistance de sortie pour la sortie à collecteur ouvert d'IC1. Une valeur plus faible augmentera légèrement sa vitesse, mais au détriment de la consommation de courant. La tension d'alimentation minimale d'IC1 de 3 V (16 V max.) définit la limite inférieure de V_{CC} pour la configuration MIDI In.

La présence du condensateur de dérivation C3 relève sans doute plus de la *bonne pratique* que de la nécessité, mais comment pourrions-nous prédire quelles longueurs de lignes seront reliées à K4 ? (garder ces longueurs aussi courtes que possible relève aussi de la *bonne pratique*).

La résistance R5 de 0 Ω sert juste à partager une broche de blindage de K2 avec une broche de blindage de K1, évitant ainsi d'involontairement tout connecter à GND. Ne montez R5 que si K2 est présent.

Configuration pour MIDI Out et MIDI Thru

Un module MIDI Out ou MIDI Thru utilise moins de composants puisque D1, IC1 et R2 n'y sont pas nécessaires. R3 est une résistance de 0 Ω reliant R1 à V_{CC}, R4 est la résistance de limitation de courant de la sortie MIDI.

Les valeurs de R1 et de R4 dépendent de V_{CC}. Une alimentation de 5 V attend 220 Ω pour chacune ; il s'agit du port MIDI Out traditionnel des systèmes à 5 V. Le cas des systèmes à 3,3 V est traité ci-dessous.

Calculs pour les signaux de 3,3 V

La norme 1.0 spécifie d'attaquer la LED du photocoupleur de l'entrée MIDI avec un courant de 5 mA de façon à ce que celle-ci brille suffisamment. Même si les optocoupleurs actuels n'ont pas besoin d'une intensité aussi forte (1,6 mA suffisent pour notre H11L1), nous garderons 5 mA puisqu'un module MIDI Out moderne doit être en mesure de piloter un ancien MIDI In.

Une entrée MIDI est équivalente à une résistance de 220 Ω en série avec une LED. La norme MIDI suppose que la tension directe maximale de cette LED vaut 1,9 V. Le niveau V_{TX} du signal de sortie doit donc au moins valoir :

$$V_{TX} \geq 0{,}005 \times 220 + 1{,}9 = 3{,}0\ V$$

R4 assure la limitation de courant de court-circuit requise. R1 et R4 (rappel : R3 = 0 Ω) introduisent une chute de tension que devra également dépasser V_{TX}, d'où :

$$V_{TX} - 3{,}0 \geq 0{,}005 \times (R1+R4)$$

Soit, avec V_{TX} = 3,3 V :

$$R1 + R4 = 0{,}3 / 0{,}005 = 60\ \Omega$$

La MMA recommande R1 = 33 Ω et R4 = 10 Ω, soit 43 Ω au total, de façon à permettre différentes tensions de sortie, tensions d'alimentation et tolérances de résistances.

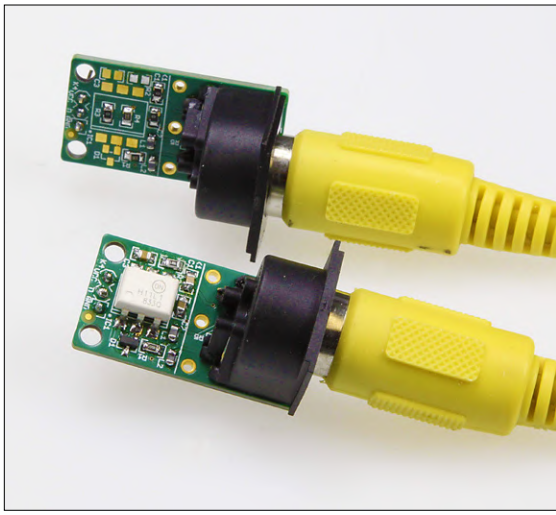
Attention aux courts-circuits

Supposons l'entrée MIDI distante défectueuse et, pour une raison ou une autre, reliée à la masse. L'intensité traversant R1 vaudra :

$$V_{CC} / R1 = 3{,}3 / 33 = 0{,}1\ A$$
$$PR1 = (0{,}1)2 \times 33 = 330\ mW$$

Comme R1 est reliée à l'alimentation de l'émetteur et que celle-ci peut assurément délivrer ces 100 mA, une 0,5 W s'impose pour R1.

Le même raisonnement vaut pour R4 (il faudrait une 1,2 W) si ce n'est que la norme la suppose attaquée par une sortie à collecteur/drain ouvert avec une résistance de rappel vers le haut. Cette résistance d'une valeur (typique) de quelques centaines d'ohms limitera le courant, permettant de ramener la puissance de R4 à 0,25 W. Prudence donc lorsqu'un module MIDI Out de 3,3 V est attaqué par un étage numérique. Cette limitation ne s'applique pas au MIDI Out de 5 V.



La morale de BoB

Tout ceci montre que même en présence de circuits simples, la vigilance doit rester de mise, surtout si l'on tente d'interconnecter des matériels d'origines et d'époques différentes.

Vous trouverez en [1] un PDF contenant neuf tableaux qui indiquent quels composants monter pour quelle configuration.

(190070-04 – version française : Hervé Moreau)

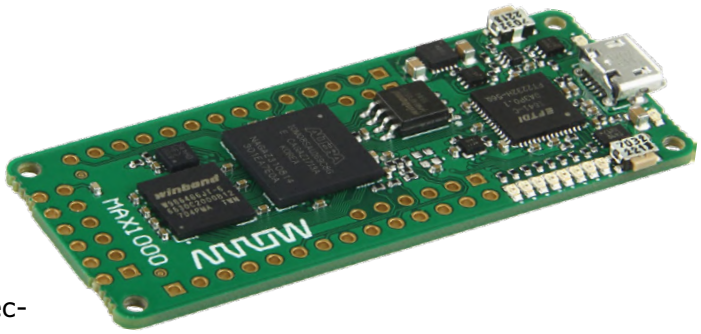
Lien

[1] BoB d'E/S MIDI dans le labo d'Elektor : www.elektormagazine.fr/labs/midi-io-break-out-board

projet SCCC (3)

processeur *softcore* et compilateur C à construire soi-même

Martin Oßmann



Dans l'article précédent de cette série, nous nous sommes limités à la simulation de notre CPU DIY (*Do It Yourself* = faites-le vous-même). Aujourd'hui, nous exploiterons à nouveau la carte MAX1000. Nous connectons la petite carte de développement par une ou deux interfaces séries au PC, pour télécharger les programmes et réaliser l'entrée/sortie de données. En outre, nous lisons la sortie d'un accéléromètre qui se trouve également sur la carte.

Nous avions déjà utilisé la carte *MAX1000* dans le premier article de cette série [1]. Nous allons maintenant aborder la procédure de chargement d'un programme compilé dans la puce FPGA. La compilation du programme en C est relativement rapide. Ce processus produit deux fichiers *codemem.txt* et *datamem.txt*. Nous devons maintenant « charger » leur contenu dans la mémoire du CPU, laquelle se trouve dans la FPGA.

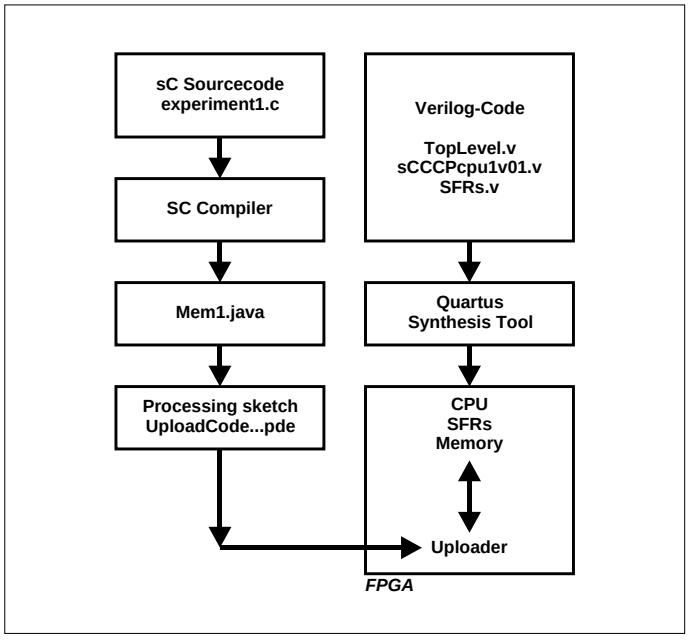


Figure 1. Le programme de téléchargement permet aussi d'envoyer les programmes de l'utilisateur compilés une fois la synthèse du CPU dans la FPGA effectuée. Cela fait gagner un temps considérable.

C'est l'objet des lignes qui suivent dans le fichier de code *Verilog sCCCPcpu1v01.v* (lequel définit le CPU) :

```
parameter sCcodeMemAddressWidth=12 ;
parameter sCdataMemAddressWidth=10 ;
reg [40-1:0] sCcodeMem [0:(1 <<
   sCcodeMemAddressWidth)-1] ;
reg [32-1:0] sCdataMem [0:(1 <<
   sCdataMemAddressWidth)-1] ;

initial begin
  $readmemh("CodeMem.txt",sCcodeMem) ;
  $readmemh("DataMem.txt",sCdataMem) ;
end
```

Les quatre premières lignes établissent la largeur et le nombre de mots occupés par le code et les données.
Les lignes derrière `initial begin` indiquent que le code et les données doivent être initialisés au moyen du contenu des fichiers indiqués. Il est nécessaire que ces fichiers soient à disposition au moment de la synthèse de la FPGA. Nous « synthétisons » la FPGA : après la compilation du programme en C, nous obtenons un fichier de bits à charger dans la FPGA. Cette tâche est relativement longue à exécuter (pour l'auteur, elle prend env. 5 min). Il faut répéter cette synthèse après chaque modification du programme en C.
C'est assez usant et nous nous sommes demandé s'il était possible de charger le programme et les données dans la FPGA seulement après l'étape de synthèse. Cela fonctionne si la FPGA dispose de ce que l'on pourrait appeler un *uploader* (chargeur). Essayons ! Un projet Quartus (.qpf) est enregistré dans le dos-

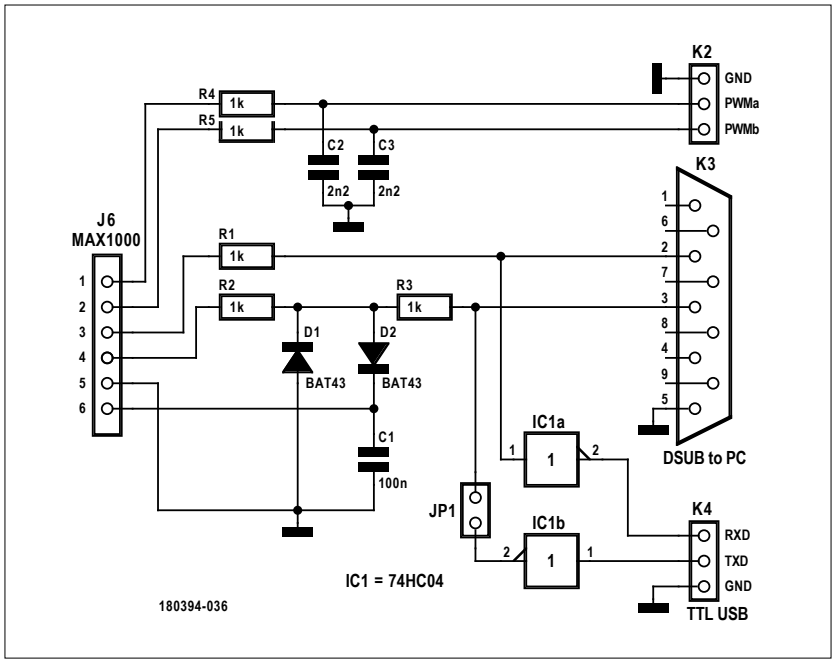


Figure 2. Extension matérielle pour l'interface de service de téléchargement de programmes. Les sorties MLI (*PWM*) ne sont pas utilisées dans un premier temps.

sier de travail *experiment5* (téléchargement sous [3]). Pour pouvoir utiliser l'*uploader*, vous devriez « synthétiser » ce projet et en charger le code dans la FPGA. La **figure 1** montre le diagramme des tâches correspondant.
Pour envoyer les données et le programme dans la FPGA, nous utiliserons une interface série. Mais avant cela, quelques remarques.

Notre première extension matérielle

Nous utilisons les interfaces en mode standardisé *115200 8N1*. Pour connecter l'interface série du PC à la carte *MAX1000*, nous utilisons le matériel de la **figure 2**. Si une véritable interface série est disponible sur le PC, vous pouvez la brancher directement sur le connecteur à 9 broches. IC1 n'est pas nécessaire (il suffit de retirer le cavalier JP1). L'auteur pilote ainsi ses interfaces. Ce n'est pas très académique, mais ça marche. De nombreux lecteurs disposent le cas échéant d'un câble série USB-TTL ; les signaux sont intervertis par rapport aux niveaux RS232. Ce câble peut être raccordé à K4. Il y a encore plus simple : il est aussi possible de raccorder directement le câble USB-TTL à la FPGA, il faut alors programmer dans la FPGA un inverseur pour chacun des deux signaux.

Interface série de service

Pour télécharger les programmes de l'utilisateur via l'« interface de service », l'ordinateur de développement exécute un petit programme PC. L'auteur a réalisé ce programme à l'aide de *Processing* [4]. Le programme est disponible en téléchargement [3] sous *Processing\UploadCodeViaSerial1v01*, il peut être démarré directement dans l'environnement de développement *Processing*. Auparavant, il faut toutefois adapter la ligne :

```
String serialPort="COM3";
```

au port série de communication utilisé sur votre matériel.
Pour vérifier que l'interface de service fonctionne, envoyez

```
1234x
```

sur l'interface (par ex. avec un programme de type terminal). Comme réponse, vous obtenez une suite de caractères hexadécimaux

```
000000001234 000000000000 1234
```

dans laquelle les données hexadécimales envoyées sont répétées et retournées.
Le dossier *Processing\UploadCodeViaSerial1v01* abrite un autre fichier, *Mem1.java*, si au démarrage de l'environnement de développement du SCCCP, l'option *-p* (pour *processing*) a été sélectionnée. Ce fichier contiendra ultérieurement le code objet du programme de l'utilisateur et est exploité par l'environnement *Processing* pour effectuer le téléchargement proprement dit.

Interconnexion des périphériques

Le CPU utilisé pour notre *expérience 5* est équipé non seulement de notre UART de service, mais également d'autres blocs de fonctions, à savoir un compteur (RTC), un modulateur en largeur d'impulsion MLI (*PWM*), un convertisseur A/N, une interface GPIO et enfin, mais très important, une UART utilisateur. Du point de vue matériel, cette UART utilisateur est connectée à l'interface PMOD2 de la carte MAX1000 (J6), comme indiqué à la **figure 3**.
Voyons maintenant comment étendre le CPU FPGA à l'aide de blocs périphériques simples.
Tout d'abord, nous utiliserons trois broches comme broches GPIO (E/S à usage général).
En *Verilog*, cela se programme ainsi :

```
parameter GPIOINPWIDTH = 3 ;
wire[GPIOINPWIDTH-1:0] GPIOinp ;
parameter GPIOOUTPWIDTH = 3 ;
reg[GPIOOUTPWIDTH-1:0] GPIOoutp ;
```

Pour lire des données sur une extension, nous exécutons une instruction `INPA c`, où `c` (comme canal) représente le numéro du port, c'est-à-dire le numéro utilisé pour matérialiser le périphérique du CPU dans la FPGA.
En *Verilog*, le mécanisme est représenté par la variable `inpA` dont la valeur apparaîtra dans le registre *R0* lors du traitement de l'instruction ci-dessus. Un grand « multiplexeur » programmé comme indiqué ci-dessous détermine laquelle des données est concernée :

```
assign inpA=
  (inpAchannel==0) ?  { 32'h12341234   } :
  (inpAchannel==1) ? sCuartRXdata :
  (inpAchannel==2) ? sCTXuartBitsLeft :
  (inpAchannel==3) ?{ 20'h12300 , ADCval[12-1:0] } :
  (inpAchannel==4) ? RTCtimer :
  (inpAchannel==5) ? debugReg1 :
  (inpAchannel==6) ? aDacSignal :
  (inpAchannel==7) ? bDacSignal :
  (inpAchannel==18) ? GPIOinp :
  32'h12341111 ;
```

Via le canal 0, on obtient par ex. toujours la valeur « 12341234 »

(c'est une donnée de test). Via le canal 1, on obtient la donnée reçue par l'UART utilisateur. Via le canal 18, on obtient les broches GPIO que nous voulons utiliser pour l'*expérience 5.*
Le fonctionnement en sens inverse est similaire. Si on indique une donnée pour le canal c avec l'instruction `OUTA c`, le signal `outAstrobe` est activé et les données parviennent « à bon port » grâce à un grande structure de type *if-then-else*. En *Verilog*, cela s'exprime comme suit :

```
always @(posedge sCclk) begin
  if(outAstrobe) begin
    if( outAchannel==0) begin
     ...
     else if( outAchannel==5) begin
      LEDs<=outA[8-1:0] ;
      end
      ...
     else if( outAchannel==18) begin
      GPIOoutp<=outA[GPIOOUTPWIDTH-1:0] ;
      end
       ...
    end
  end
```

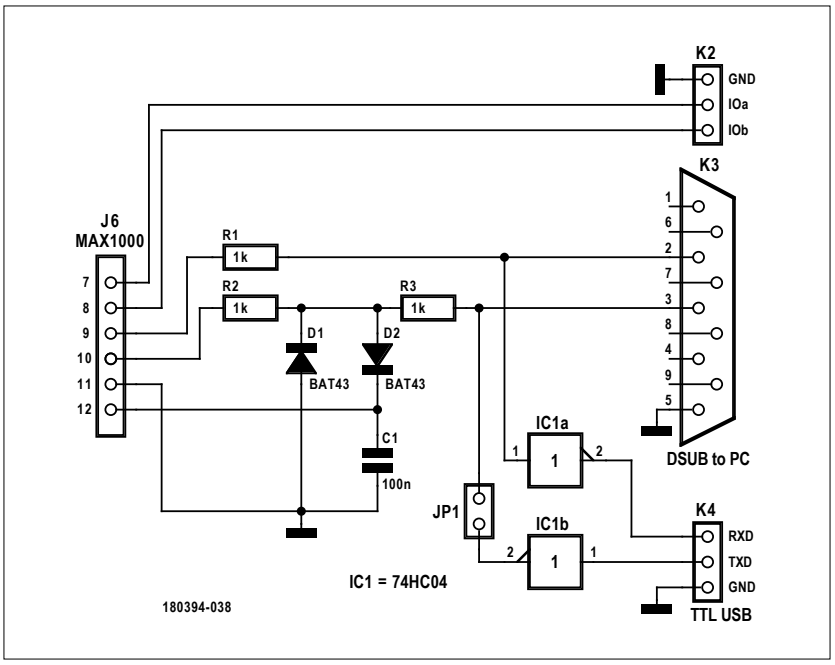


Figure 3. Cette extension permet d'effectuer des expériences d'E/S simples.

Nous voyons par ex. que le canal 5 est relié aux LED utilisées lors de l'*expérience 1*. Le canal 18 sert à piloter nos broches GPIO. Maintenant relions ces broches aux broches matérielles proprement dites.

Expérience 5 : GPIO et SPI logicielle

Un accéléromètre modèle *LIS3DH* de *ST* est monté sur la carte *MAX1000*. Nous le solliciterons avec notre CPU. Pour cela, nous équipons notre CPU comme décrit ci-dessus de broches GPIO que nous relions aux broches du *LIS3DH*. Nous définissons les broches physiques dans le code *Verilog* du module de niveau supérieur (*toplevel*) comme broches d'entrée ou de sortie :

```
...
input LISsenINT1,
input LISsenINT2,
input LISsenSDO,
output LISsenCS,
output LISsenSPC,
output LISsenSDI
...
```

Nous connectons ensuite dans la FPGA les broches du *LIS3DH* aux broches GPIO de notre CPU DIY via les instructions d'affectation (`assign`) correspondantes (listage 18) :

```
assign GPIOinp[0] = LISsenINT1 ;
assign GPIOinp[1] = LISsenINT2 ;
assign GPIOinp[2] = LISsenSDO ;

assign LISsenCS = GPIOoutp[0] ;
assign LISsenSDI = GPIOoutp[1] ;
assign LISsenSPC = GPIOoutp[2] ;
```

Ces deux séries d'instructions *Verilog* permettent au CPU logique de lire et d'écrire des données, respectivement avec les instructions `INPA` et `OUTA`. Nous pouvons ici étendre les périphériques existants du CPU en créant nos propres périphériques. Reste à voir comment dialoguer avec les périphériques du CPU dans les programmes en C.

Rien que du C

Nous avons vu qu'il est possible d'intégrer du code en assembleur dans nos programmes en C. Cela simplifie considérablement le dialogue avec les périphériques du CPU (et donc le matériel externe). L'exemple du **listage 1** provient du fichier *experiment5/IOlib5.c.*
Comment travaille le compilateur ? Lors de l'évaluation d'une expression arithmétique, la valeur se retrouve dans *R0*. Le contenu de *R0* représente la valeur retournée par le code des fonctions. Sachant cela, et avec les commentaires du code, nous comprenons comment cela fonctionne. Nous disposons maintenant de fonctions en C nous permettant d'accéder au capteur d'accélération *LIS3DG*. Du point de vue de la programmation, nous reproduisons l'interface SPI en communiquant par logiciel avec les GPIO (technique dite de *bit banging*). En C, cela donne le **listage 2**.
Nous en arrivons à l'expérimentation proprement dite.

Action : *Go*

L'exécution du fichier *c.bat* dans *experiment5/experiment* appelle l'environnement de développement SCCCP. Ensuite le bouton *MAKE* permet de compiler le logiciel en C.

```
c.bat:
Java
-jar c:\SCCCP\CompilerAsJar\sCCCP.jar
-s experiment5.c
-p ..\..\Processing\UploadCodeViaSerial1v01\
```

Listage 1. Écriture et lecture de données des périphériques.

```
GPIOchannel EQU 18

GPIOout(int p){
p ;                             // La valeur de p est "calculée" puis placée dans R0
#asm                            // la suite est en assembleur
OUTA GPIOchannel                // transfère la contenu de R0 au canal GPIO
#endasm
}

GPIOin(){
#asm // passage à l'assembleur
INPA GPIOchannel                // transfère la valeur du canal GPIO dans R0
#endasm                         // retour au C
}
```

L'option *-p* .... permet d'écrire le code objet résultant de la pression du bouton *MAKE* dans *mem1.java*. Ce fichier est placé dans le dossier de téléchargement *Upload*.
Nous « synthétisons » ensuite le projet Quartus *experiment5.qpf* dans le dossier de travail et chargeons le fichier de bits correspondant dans la FPGA. Dans la FPGA, le programme de téléchargement est prêt à charger les mémoires de données et de programme via l'interface de service. Pour le vérifier, nous envoyons à l'aide d'un programme de terminal un « x » et nous contrôlons que la réponse est une chaîne de chiffres hexadécimaux.
Pour le téléchargement, nous utilisons le croquis *UploadCodeViaSerial1v01.pde* qui s'exécute dans l'environnement de programmation *Processing*.
Désormais, nous pouvons (dans l'environnement *Processing*) démarrer le téléchargement effectif. Ensuite nous chargeons le programme utilisateur proprement dit dans la FPGA. Ce programme doit transmettre le contenu des trois canaux de l'accéléromètre *LISD3* via l'interface utilisateur (à 115 200 bauds) :

```
x= -00000032 y= -00000128 z= +00016576
x= -00000048 y= -00000224 z= +00016688
x= -00000064 y= -00000240 z= +00016544
x= -00000032 y= -00000224 z= +00016640
x= +00000016 y= -00000256 z= +00016704
```

Nous observons les effets des rotations ou inclinaisons de la carte *MAX1000* sur la valeur des accélérations.
Pour modifier le programme qui s'exécute sur le CPU, il suffit désormais de compiler le nouveau programme et de l'envoyer dans la *MAX1000* par téléchargement. C'est notablement plus rapide que de « synthétiser » à nouveau le CPU, comme nous le faisions jusqu'ici. Une fois que le programme fonctionne parfaitement, nous pouvons désactiver à nouveau le téléchargement dans le fichier *MainInclude.txt* du projet *Verilog* de CPU en remplaçant les lignes :

```
`define bootload
//`define standalone
```

par

```
//`define bootload
`define standalone
```

Cela a pour effet d'intégrer à nouveau le contenu des mémoires de données et de programme au moment de la synthèse.
Notre *expérience 5* fait partie des essais complexes, car toute une série de fichiers, programmes et composants est concernée. Si cela ne se passe pas correctement, vérifiez d'abord les points énumérés ci-après. Tous les chemins désignés dans les fichiers *batch* ont-ils été adaptés à votre installation personnelle ?

Liens

[1] Projet SCCC (1), Elektor 3-4/2019 : www.elektormagazine.fr/180394-04
[2] Projet SCCC (2), Elektor 5-6/2019 : www.elektormagazine.fr/180394-B-04
[3] Page de l'article : www.elektormagazine.fr/180394-C-04
[4] Processing : www.processing.org

Listage 2. Communication SPI par logiciel (*bit banging*).

```
int GPIOoutShadow ;
#define bitMaskLISsenINT1 1
#define bitMaskLISsenINT2 2
#define bitMaskLISsenSDO  4

#define bitMaskLISsenCS   1
#define bitMaskLISsenSDI  2
#define bitMaskLISsenSPC  4

LISinit(){
  GPIOoutShadow= bitMaskLISsenCS | bitMaskLISsenSPC | bitMaskLISsenSDI ;
  GPIOout(GPIOoutShadow) ;
 }

LSIclockPulse(){
  GPIOoutShadow = GPIOoutShadow & (~bitMaskLISsenSPC) ;
  GPIOout(GPIOoutShadow) ;
  GPIOoutShadow = GPIOoutShadow | bitMaskLISsenSPC ;
  GPIOout(GPIOoutShadow) ;
  }

SPItxrxByteMSBfirst(int vTX){
  int vRX ;
  int k ;
  int GPIOinp ;
  vRX=0 ;
  for( k=0 ; k<8 ; k++){
    if(vTX & 0x80) {
         GPIOoutShadow =GPIOoutShadow | bitMaskLISsenSDI ;
       }
        else {
          GPIOoutShadow =GPIOoutShadow &( ~ bitMaskLISsenSDI );
     }
    LSIclockPulse() ;
      vRX = vRX << 1 ;
      GPIOinp=GPIOin() ;
      if( GPIOinp & bitMaskLISsenSDO ) { vRX = vRX |1 ; }
      vTX=vTX<<1 ;
  }
  return vRX ;
}
```

Les numéros de port COM et toutes les vitesses de transmission sont-ils réglés correctement ?
Si tout est correct, jetez ensuite un coup d'œil sur le programme *experiment5.c*. Ce programme utilise notamment l'UART utilisateur pour la sortie, et vous pouvez examiner comment c'est réalisé.
Dans le code, la communication avec l'accéléromètre est effectuée par *bit banging* (exécution d'un protocole série par logiciel). Il aurait été possible d'implémenter le protocole SPI dans la FPGA. Cependant, dans notre application simple, non-critique sur le plan temporel, il était plus facile de réaliser une interface logicielle.

Dans le prochain article de cette série, nous nous occuperons de signaux analogiques. Pour cela, nous utiliserons des CN/A sigma-delta et des interruptions. À bientôt !

(180394-C-04 – version française : Yves Georges)

le promontoire des étudiants

TUE/InMotion : objectif Garage 56

Crédit de la photo : Johan van Uden.

Kjeld Teunissen (Pays-Bas)

L'équipe d'étudiants InMotion de l'université de technologie d'Eindhoven (TUE) nous présente son projet de « ravitaillement électrique » visant à recharger un véhicule électrique aussi rapidement que l'est en carburant un véhicule à essence. Prometteur, n'est-ce pas ? Voyons si ces jeunes gens ont fait le plein de bonnes idées…

L'équipe InMotion comprend plus de cinquante étudiants appartenant à l'université de technologie d'Eindhoven et à l'université Fontys des sciences appliquées, toutes deux situées aux Pays-Bas. Un partenariat unique entre le monde universitaire et celui de l'industrie offre à notre jeune et ambitieuse équipe la possibilité de travailler sur des projets à la pointe de la technologie. Nous avons baptisé « Electric Refuelling », littéralement « Ravitaillement Électrique », celui auquel nous travaillons actuellement et dont l'objectif est de rendre la charge d'un véhicule électrique aussi rapide que le plein d'un véhicule classique. Nous espérons démontrer le potentiel de notre idée avec l'inscription en 2023 d'une voiture électrique au Garage 56 du Mans (stand que réserve chaque année l'organisateur des 24 Heures du Mans à

Votre équipe travaille sur un projet cool ?

Vous faites partie d'une équipe d'étudiants ayant réalisé quelque chose d'étonnant ou de sympa en électronique ?
Écrivez-nous à redaction@elektor.fr, sujet : « Le promontoire des étudiants ».

une voiture expérimentale). Pour cela, il nous faudra mettre au point notre propre batterie, travail dont j'ai la responsabilité cette année.

Dilemme

Quelle que soit son application, la conception d'une batterie impose en premier lieu une définition claire de ses caractéristiques, dont les quatre principales sont : sa tension (minimale, nominale et maximale), sa capacité, son poids et sa puissance. Les normes de sécurité doivent également être prises en compte.

La technologie actuelle des batteries place le concepteur face à un dilemme. D'un côté il peut construire une batterie de grande capacité dont l'autonomie sera synonyme de grandes distances parcourues, de l'autre il peut opter pour une batterie de grande puissance qui se montrera mieux adaptée aux applications de forte puissance et à une charge plus rapide. Le cœur de notre projet étant la charge rapide, le second choix s'est imposé. Notre batterie affiche donc une capacité inférieure à celle d'un véhicule utilitaire type, mais cela nous permet d'élaborer une technologie qui devien-

L'équipe InMotion (crédit : Bram Naus).

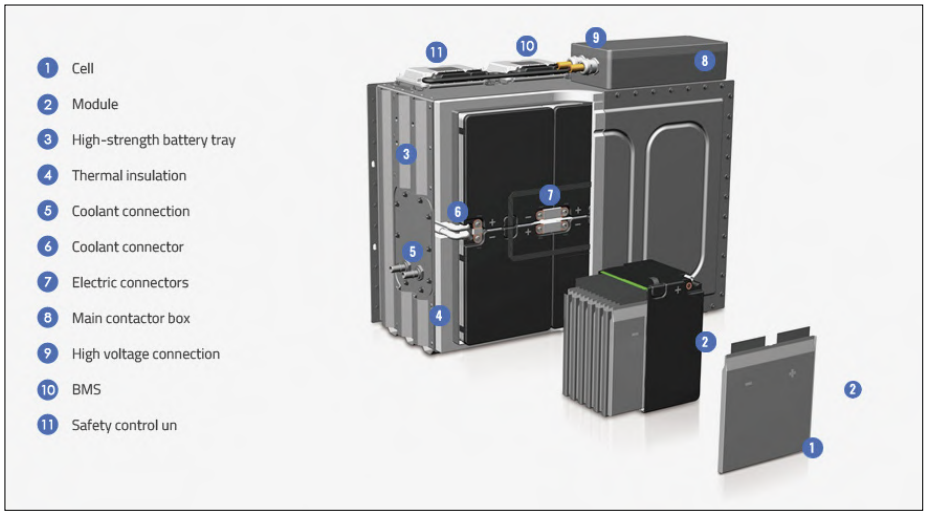


Figure 1. Exemple de batterie Kokam avec système de refroidissement [1].

dra exploitable dès que le progrès aura donné aux batteries de forte puissance une capacité suffisante pour des applications commerciales.

Structure

Rappelons qu'une batterie (**fig. 1**) est formée de nombreuses cellules rechargeables, ou accumulateurs, plusieurs de ces cellules étant regroupées dans des modules assemblés en série, en parallèle, ou les deux.

Cellules

Le choix du type de la cellule est primordial puisqu'il représente le facteur déterminant de la performance de la batterie. InMotion utilise des cellules « poche » (*pouch cells*) car elles conviennent mieux à une charge rapide que les cellules cylindriques. Elles sont aussi plus robustes et ont généralement une résistance interne plus petite, d'où une production de chaleur moindre. Leur plus grande surface facilite par ailleurs le refroidissement des cellules.

Modules

Les variables de conception propres à un module sont déterminées par l'espace libre requis autour de chaque cellule. Dans le cas des cellules « poche », les dimensions de cet espace dépendent généralement de l'emplacement des connecteurs et du type de refroidissement utilisé.

Assemblage

Une fois leurs dimensions déterminées, les modules sont prêts à être empilés. La configuration de cet empilement déterminera la forme du circuit de refroidissement ainsi que la connexion électrique des modules.

Un logiciel maison

Le développement de notre projet est dynamique en ce sens que notre méthode de travail nous permet de modifier certaines choses à la volée et qu'aucune de nos exigences n'est figée dans le marbre.

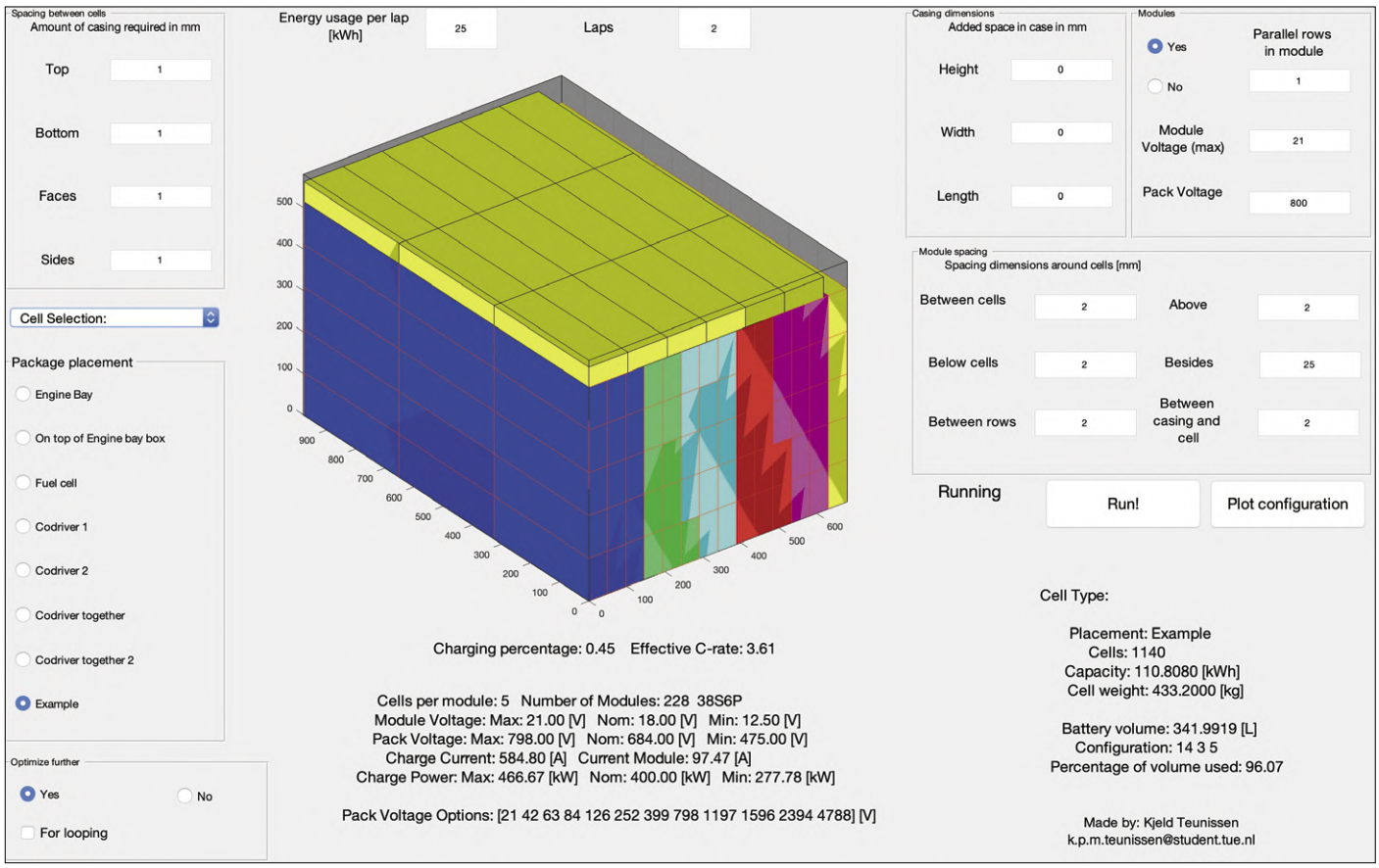


Figure 2. L'interface graphique de notre configurateur.

La batterie est un bon exemple de cette démarche puisqu'elle peut être configurée de multiples façons. Mais comment savoir laquelle de ces configurations est la meilleure ?
Pour ne pas avoir à créer un modèle pour chaque configuration possible, j'ai écrit un programme à interface graphique qui détermine un assemblage optimal en fonction des différentes variables de conception passées en entrée (**fig. 2**). Ce configurateur essaie d'abord d'insérer autant de modules que possible dans un espace donné en les empilant selon une même direction, puis recommence avec l'espace restant, même si pour cela il doit orienter certains des modules dans des directions différentes.
Pour résoudre ce problème d'optimisation spatiale, j'ai choisi une méthode heuristique plutôt qu'une approche analytique. Une méthode heuristique est en effet plus facile à coder, atteint toujours les minima locaux et se montre extrêmement efficace et rapide. Sur un PC ordinaire, le logiciel crée 190 configurations optimales, sélectionne la meilleure pour un ensemble donné de paramètres préférentiels, et affiche sa représentation graphique en cinq secondes. Il enregistre également toutes les configurations optimales créées, ce qui permet de les filtrer en fonction de certains critères préférentiels. Dans ce cas, si une configuration différente s'avère meilleure, un simple clic permet de visualiser le tracé de la solution trouvée.
Le programme calcule les performances attendues de la batterie et indique :

- la capacité prévue ;
- le nombre de modules connectés en série et en parallèle ;
- le poids attendu des cellules ;
- la différence de tension entre batterie chargée et vide.

Il calcule en outre les intensités de charge, indique la quantité de chaleur produite, et donne la puissance nécessaire pour charger la batterie en 7,5 min.

Palmarès

Pour montrer à quel point les courses disputées avec des voitures électriques sont rapides, nous avons créé la Fusion, une Formule E qui détient actuellement les records du tour le plus rapide sur les circuits Zandvoort, TT Assen (Pays-Bas) et Zolder (Belgique). Conçue pour les épreuves d'endurance, notre voiture concept Vision incarne par ailleurs le futur de la mobilité électrique. Son aérodynamisme est en effet dix fois plus performant que celui d'une Formule 1. Notre objectif est donc maintenant la charge d'une voiture électrique en seulement quelques minutes. Lorsque nous l'atteindrons, la transition énergétique pétrole-électricité attendue pour l'industrie automobile sera indubitablement accélérée.

(180732-02 – version française : Hervé Moreau)

Liens

[1] Site web de Kokam : http://kokam.com/modulepack
[2] Site web d'InMotion : www.inmotion.tue.nl/

AGENDA

juillet 2019

◇ **Textival!**
02/07 – Lyon
http://textival.fr/

◇ **Japan Expo Paris**
04 au 07/07 – Paris
www.japan-expo-paris.com/fr/

◇ **Nantes Maker Campus 2019**
05 au 07/07 – Nantes
http://nantesmakercampus.com

◇ **Salon du véhicule électrique et hybride**
11 au 14/07 - Val d'Isère
www.salon-vehicule-electrique.com

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**.

moniteur de batteries à ESP32

mesures sur trois voies

Laurent Labbe (Hong Kong) et **Luc Lemmens** (labo d'Elektor)

Ce projet montre combien il est simple et facile de mesurer des valeurs analogiques avec une carte ESP32 et d'envoyer les mesures par Wi-Fi à un routeur, et de là vers le site web ThingSpeak où les données sont enregistrées et affichées.

Actuellement Laurent travaille sur plusieurs projets alimentés avec de l'énergie solaire. Il cherche entre autres à concevoir une alimentation solaire pour sa montre Nixie [1]. Par nature, l'ensoleillement n'est pas prévisible, il faut parfois des heures, voire des jours pour tester un modèle d'alimentation. Il a conçu trois montages différents, chacun chargeant trois petites batteries identiques (130 mAh) avec une résistance pour simuler la consommation de la montre. Ensuite, il lit toutes les dix minutes le niveau des batteries sur trois montages différents et envoie les mesures sur la plateforme gratuite pour l'Internet des Objets ThingSpeak [2].

Mesures

Les mesures sont recueillies sur une carte de développement ESP32-PICO-KIT-V4 alimentée par USB (5 V). Comme le montage est placé à l'extérieur, sans prise secteur, l'auteur utilise deux batteries externes (*powerbank*), une petite (1 000 mAh) et une grosse (10 000 mAh). La plus petite est connectée en permanence pour alimenter la carte ESP32. Elle est rechargée par la grosse qui est remplacée presque tous les jours pour que le système soit toujours en état de marche. Si on n'utilisait qu'une seule batterie, il faudrait arrêter les mesures lors de la recharge de la batterie alimentant l'ESP32.

Trois broches d'E/S de la carte ESP32 (**fig. 1**) servent à effectuer les mesures sur les trois canaux/batteries. Chaque broche possède son propre diviseur de tension et un condensateur de filtrage de 100 nF. Dans le code, cette atténuation est représentée par la variable `Coef_Vx` et sa valeur peut être ajustée pour compenser les tolérances dans chaque diviseur de tension externe.

Le convertisseur analogique/numérique de la carte ESP32 est doté d'un atténuateur interne que l'on peut régler dans le croquis Arduino avec la fonction `analogSetAttenuation` sur les valeurs suivantes : 0 dB, −2,5 dB, −6 dB ou −11 dB ; soit un facteur de 1 ; 0,75 ; 0,5 ou 0,28.

Par défaut, la tension de pleine échelle du CA/N est de 1,1 V. Cette tension est la valeur maximale que l'on peut lire (en fonction de la largeur de bits configurée pour ADC1 : 4095 pour 12 bits, 2047 pour 11 bits, 1023 pour 10 bits, 511 pour 9 bits). Pour lire des tensions plus élevées (jusqu'à la tension maximale applicable sur la broche, généra-

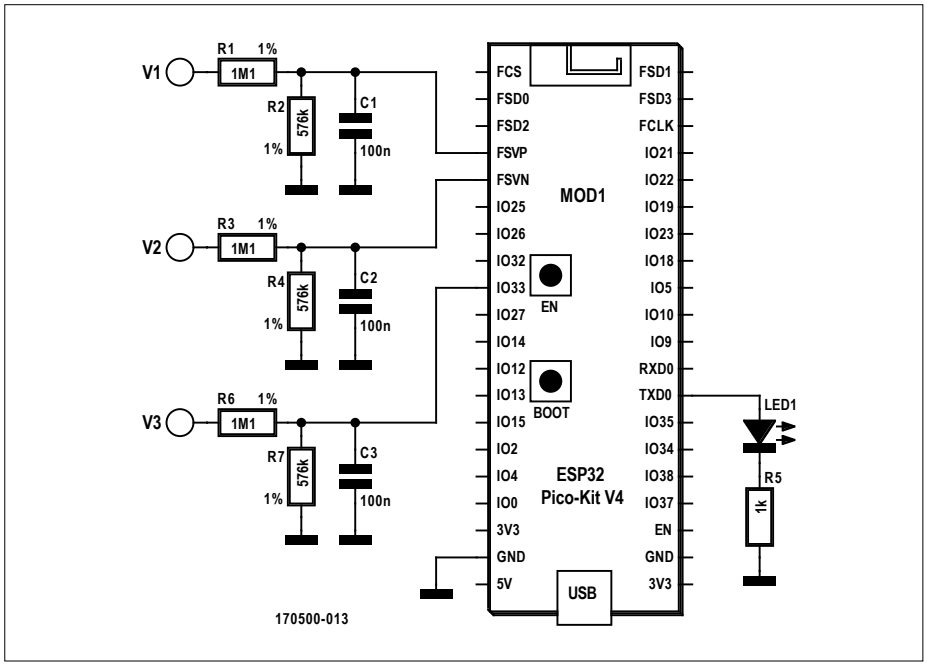


Figure 1. Schéma du circuit : la carte ESP32 se charge de presque tout le travail.

lement 3,3 V = Vdd), il faut régler une atténuation supérieure à 0 dB pour le canal de CA/N concerné. Le **tableau 1** résume les réglages et les plages que l'on peut atteindre. Pour l'atténuation de −11 dB, la tension maximale est limitée par VDD_A, et non par la tension de pleine échelle.

Le circuit a été conçu pour surveiller des batteries LiPo de 4,2 V et ses entrées analogiques sont calculées pour supporter une tension d'entrée maximale de 5 V. Avec des diviseurs de résistance de 1,1 MΩ et de 576 kΩ, les entrées de la carte ESP32 seront soumises à une tension de ±1720 mV pour une tension d'entrée de 5 V, ce qui est la tension d'entrée maximale recommandée pour l'atténuation de −6 dB. Bien entendu, vous pouvez modifier le circuit d'entrée si vous souhaitez avoir une plage de tension d'entrée différente et/ou modifier l'atténuation interne du CA/N de la carte ESP32.

Attention : vous ne pouvez appliquer sur les entrées du circuit que des tensions continues sur la plage définie par le réglage du CA/N. En outre il n'y a pas de protection contre l'inversion de polarité ou les surtensions !

La broche GPIO17 de la carte ESP32 pilote une LED qui s'allume pendant les mesures de tension et la transmission des données. Ensuite, la LED s'éteint et la carte ESP32 passe en mode veille jusqu'à ce qu'elle se réveille automatiquement pour les prochaines mesures. Cela permet d'économiser l'énergie des batteries externes.

Vous pouvez facilement câbler le circuit sur une plaque d'expérimentation ou un morceau de plaque à trous.

Créer un compte ThingSpeak

Laurent utilise la plateforme en ligne ThingSpeak pour un projet de station météo. C'est tout naturellement qu'il y a ouvert une page pour enregistrer les mesures effectuées sur les batteries des montres (voir [3]).

ThingSpeak est un service web gratuit, à code source ouvert, qui permet de collecter et de stocker dans le nuage les données de capteurs et de concevoir des applications pour l'Internet des Objets. Les données des capteurs peuvent être envoyées à ThingSpeak à partir de cartes Arduino, Raspberry Pi, BeagleBone Black et consœurs.

En outre il est possible d'utiliser les fonctions de plusieurs boîtes à outils MATLAB de MathWorks lorsqu'on est connecté à ThingSpeak. Pour cela il faut être connecté avec un compte MathWorks avec une licence active pour utiliser les boîtes à outils.

Si vous vous lancez dans ce projet et que vous n'avez pas encore de compte ThingSpeak ou MathWorks, allez sur www.thingspeak.com et cliquez sur `Get Starded For Free`, puis créez un compte (**fig. 2**). Cliquez ensuite sur `Connexion` pour accéder à votre compte en remplissant vos informations d'identification. Créez maintenant un canal où les données du moniteur de batteries seront collectées (`New Channel`), donnez au canal un nom logique qui sera facilement reconnu comme l'ensemble des données de votre notre moniteur de batteries. Dans les `Channel Settings`, ajoutez trois champs pour trier les don-

Tableau 1. Pleine échelle théorique du CA/N et pleine échelle atteinte, en fonction de l'atténuation réglée si VDD_A = 3,3 V.

Atténuation du CA/N	Pleine échelle théorique	Plage réellement atteinte pour les résultats les plus précis
0 dB (ADC_ATTEN_DB_0)	1,1 V	100 à 950 mV
2,5 dB (ADC_ATTEN_DB_2_5)	1,5 V	100 à 1250 mV
6 dB (ADC_ATTEN_DB_6)	2,2 V	150 à 1750 mV
11 dB (ADC_ATTEN_DB_11)	3,9 V	150 à 2450 mV

(Source : https://docs.espressif.com/projects/esp-idf/en/latest/api-reference/peripherals/adc.html#adc-calibration)

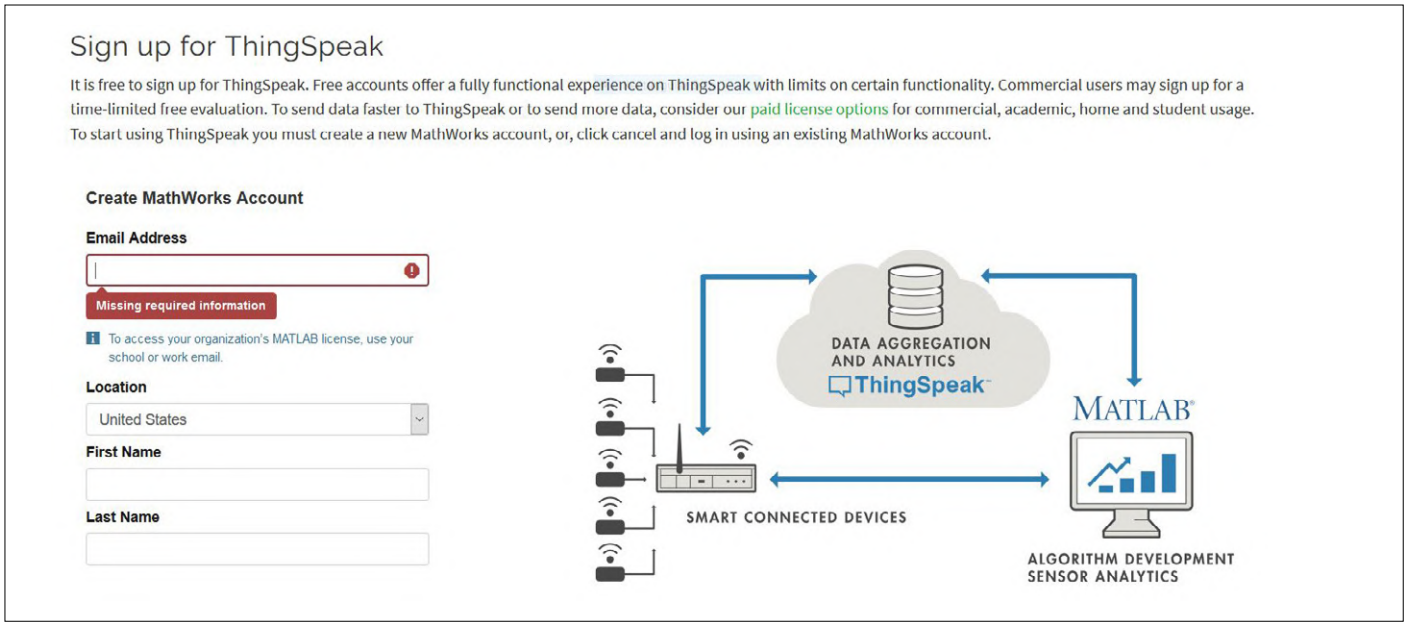


Figure 2. Création du compte chez ThingSpeak.

nées que le croquis Arduino enverra sous forme de chaînes et les étiqueter (dans notre cas V1, V2 et V3 sur la **figure 3**). Choisissez maintenant l'onglet `API keys`, ici vous trouverez la *Write API key* que vous devez copier dans le croquis de la carte ESP32 afin d'identifier le canal ThingSpeak (**fig. 4**).

Croquis du moniteur de batteries

Le croquis `Battery_monitor_elektor_3.ino` écrit en C n'est pas très compliqué, mais avant de le compiler et de le charger dans la carte ESP32, vous devez le modifier pour que la carte se connecte à votre réseau Wi-Fi et envoie les données vers votre canal ThingSpeak.

À la ligne 68 du croquis, saisissez la *Write API key* que vous avez récupérée sur le site web de ThinkSpeak pour identifier votre canal. Ensuite entrez le SSID et le mot de passe de votre réseau local respectivement sur lignes 70 et 71. Ensuite téléchargez le croquis dans la carte ESP32 et ouvrez le moniteur série de l'EDI Arduino dans le menu `Outils`. Assurez-vous que la vitesse de transmission du moniteur est réglée sur 115200 et vous devriez recevoir un message de l'ESP32 : `Battery monitor V1.0 Start` suivi des trois premières tensions mesurées.

Après chaque cycle de mesure, la carte ESP32 passe en mode veille. `Boot number: x` indique combien de cycles se sont écoulés depuis le début de la session (c'est-à-dire combien de mesures ont été effectuées jusqu'à maintenant).

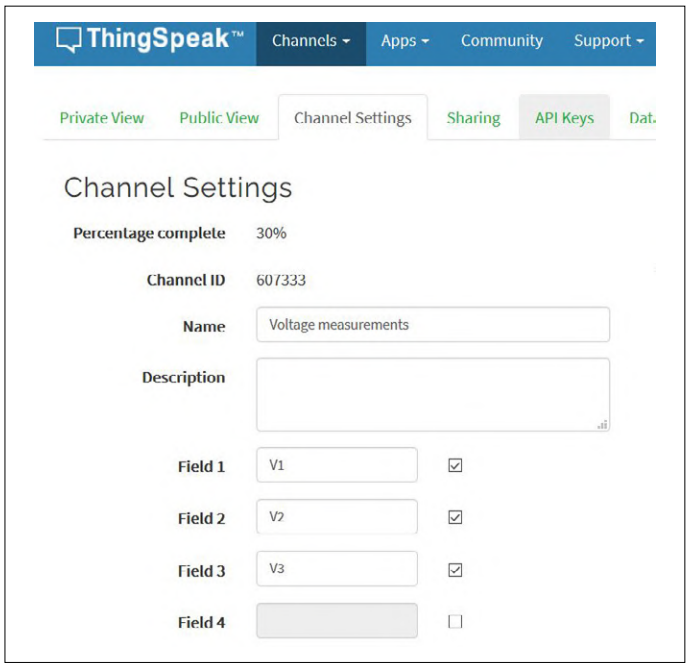


Figure 3. Réglage des canaux sur ThingSpeak.

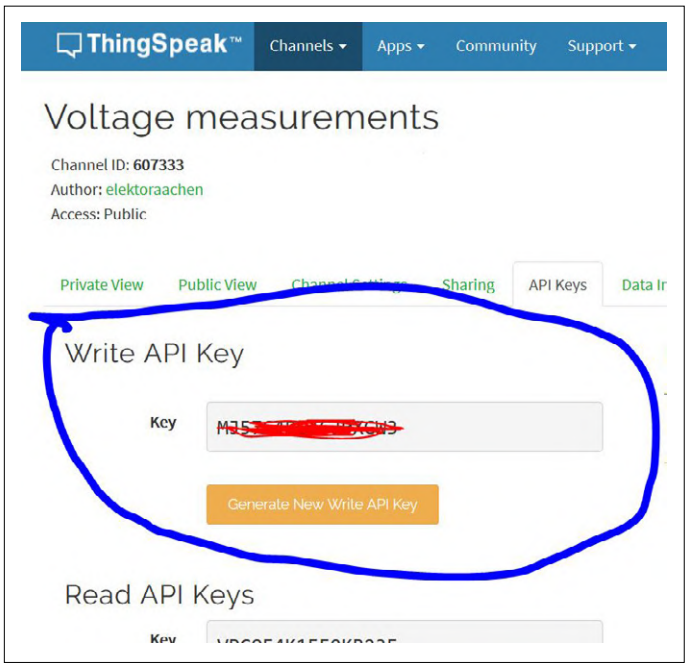


Figure 4. Clé API à recopier dans le croquis.

Ensuite, la carte ESP32 se (re)connecte à votre réseau Wi-Fi et indique la connexion a réussi avec l'adresse IP de l'ESP32, ou envoie le message `WiFi cannot connect` si cette tentative échoue. Vous pouvez régler l'intervalle de mesure en modifiant `TIME_TO_SLEEP` à la ligne 45. Notez que seule la fonction `setup` est exécutée, elle effectue tous les traitements (mesures et transmission de données) et met ensuite la carte ESP32 en mode veille jusqu'à ce qu'une minuterie interne la réveille pour le cycle de mesures suivant. La fonction `setup` sera alors à nouveau exécutée. La fonction `loop` est vide, elle ne sera de toute façon jamais exécutée.

Si c'est votre premier projet avec une carte ESP32 dans l'EDI Arduino, n'oubliez pas d'installer les cartes ESP32 dans cet environnement. Vous trouverez plusieurs tutoriels sur l'internet (voir par ex. [4]). Attention : sélectionnez le bon port COM pour la connexion avec l'ESP32-PICO-KIT-V4 dans le menu `Outils -> Port` et choisissez `ESP32 Pico Kit` dans `Outils -> Carte`.

@ WWW.ELEKTOR.FR

→ **Carte de développement ESP32-PICO-KIT V4, réf. 18423**
www.elektor.fr/esp32-pico-kit-v4

Ce projet est simple et rapide à mettre en œuvre puisqu'il n'y a aucun circuit imprimé à réaliser et à câbler. Il ouvre la porte à toutes sortes d'autres mesures et permet de mettre un pied sur le nuage.

(170500-01)

Liens

[1] Page du labo de la montre Nixie: www.elektormagazine.fr/labs/nixie-watch-1

[2] Plateforme ThingSpeak : www.thingspeak.com

[3] Page publique du moniteur de batteries sur ThingSpeak : http://thingspeak.com/channels/329068

[4] Installation des cartes ESP32 dans l'EDI Arduino: https://bit.ly/2RfTZTd

[5] Page du labo du moniteur de batteries: www.elektormagazine.fr/labs/simple-wifi-multimeter

[6] Page de l'article: http://www.elektormagazine.fr/170500-01

alimentation pour sondes différentielles

par port USB

Martin Oppermann

Dans le numéro de juillet/août 2016 d'Elektor, votre collaborateur Alfred Rosenkränzer a présenté une sonde différentielle pour oscilloscope d'une remarquable simplicité, basée sur l'amplificateur différentiel de précision AD8479, que je me suis empressé de reproduire. Elle est parfaite pour des mesures dans le domaine audio, par exemple. L'article omet toutefois de traiter un petit détail, l'alimentation de la sonde (ou de plusieurs sondes). Tous les oscilloscopes modernes possèdent sur leur face avant un port USB pour la sortie de données. Comme la consommation d'une sonde est très faible, il est tout naturel de penser à ce port pour l'alimenter. La **figure 1** donne le schéma du circuit retenu. Pour la connexion de l'alimentation de la sonde au port USB, on utilise un câble USB ordinaire ; le câble d'un chargeur fait l'affaire. Les condensateurs C1 et C2 filtrent la tension continue fournie par l'USB, LED1 est le témoin de présence de la tension. Le cœur du circuit est le convertisseur continu-continu isolé IH0512S de la société XP Power [2]. Il produit du 12 V à partir du +5 V, avec un débit maximum de 84 mA, tout en assurant l'isolation galvanique ! La tension d'isolation est de 1000 V, ce qui devrait ôter toute crainte de problèmes de masse lors des mesures. La tension de sortie est filtrée par divers condensateurs et inductances ferrites, qui, vu la fréquence de mesure maximale de 130 kHz de l'AD8479, peuvent être les uns supprimés, les autres remplacées par des résistances 0 Ω ou des ponts de câblage.

Pour la connexion des sondes Rosenkränzer, j'ai utilisé des prises femelles stéréo. Leur petite taille est certes agréable, mais elles ont l'inconvénient de mettre les contacts en court-circuit passager lorsqu'on enfonce le jack. Pour éviter d'éventuels dégâts, j'ai ajouté une résistance série de 100 Ω dans chaque circuit de sortie. Comme la consommation d'une sonde est de l'ordre de quelques milliampères (selon sa feuille de caractéristiques, l'AD8479 se contente de 850 µA environ, rien que les diodes témoins de présence de la tension D5 et D6 en consomment davantage), la chute de tension dans ces résistances n'est pas un problème. Comportant une majorité de CMS, le circuit a été réalisé sur un circuit imprimé double face, la face inférieure étant entièrement réservée à la masse. Le format des plus petits CMS est le 1206, ce qui reste encore assez facilement soudable à la main. La **figure 2** montre le dessin du circuit réalisé avec Autodesk Eagle, qu'on peut télécharger sur la page du projet [3].

Le circuit imprimé a été monté à l'intérieur d'un petit boîtier en plastique transparent de Hammond Manufacturing (type 1551RTBU).

Une dernière indication : lors de la réalisation, j'ai utilisé pour les puces IC2 et IC3 des sondes des régulateurs de tension de type 78L09 et 79L09 (donc 9 V) pour élargir le domaine de mesure à 8,7 V.

(180177-04 – version française : Helmut Müller)

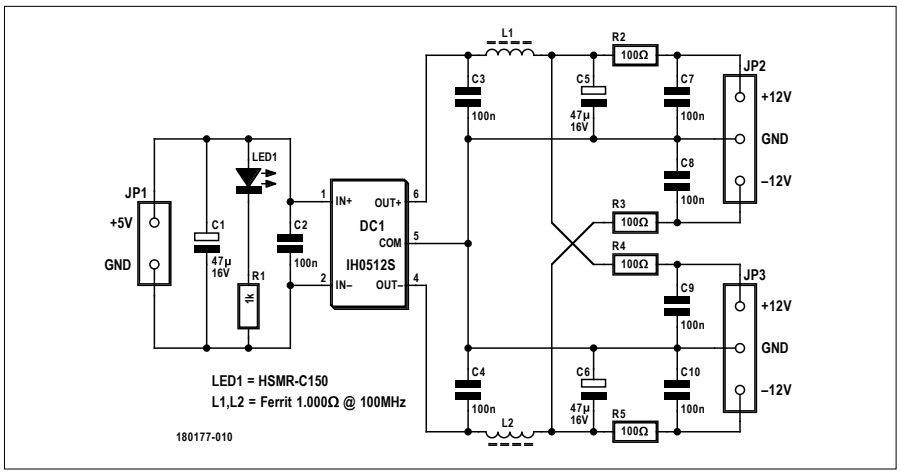


Figure 1. Schéma de l'alimentation des sondes – conçu pour deux sondes.

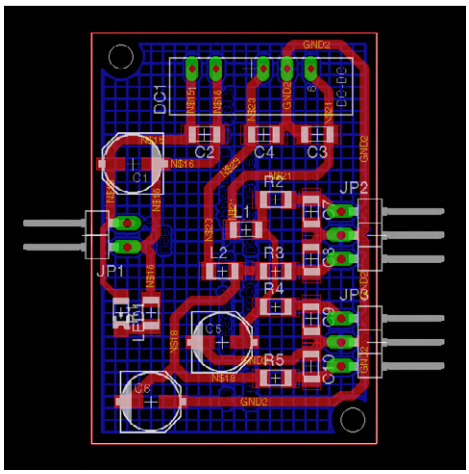


Figure 2. Conception et implantation du circuit imprimé.

Liens

[1] « Sonde de courant pour oscilloscopes », Alfred Rosenkränzer, Elektor juillet/août 2016 : www.elektormagazine.fr/150182

[2] Convertisseur DC-DC 2W IH0512S : https://www.xppower.com/Portals/0/pdfs/SF_IH.pdf

[3] Page de l'article : www.elektormagazine.fr/180177-04

affichage de données série sur une page web

avec des scripts PHP ou Python

Clemens Valens (labo d'Elektor)

Il est parfois utile de visualiser sur une page web dans un navigateur un flux continu de données récupérées sur un port série. Mais comment s'y prendre ? Pourquoi pas avec un petit script ?

Fondamentalement, le problème se résume à la création périodique d'une page web contenant les données série les plus récentes. Un navigateur web peut alors afficher la page, que ce soit sur le même ordinateur ou au travers d'un réseau. Par conséquent, ce qu'il faut c'est un petit programme qui convertit en permanence les données série par exemple en fichiers HTML ou PHP. Ah ! PHP n'est-il pas un langage de programmation pour les applications en relation avec l'internet ? Si, il l'est. Donc, pouvons-nous utiliser PHP pour faire le travail ? Oui, nous pouvons. Mais il y a aussi d'autres moyens, comme nous le verrons.

Utiliser le rafraîchissement automatique de page

Pour mémoire, le langage HTML permet l'utilisation d'une méta-balise qui demande au navigateur de recharger périodiquement une page :

```
<meta http-equiv="refresh" content="10">
```

Cette balise indique au navigateur de recharger la page contenant la balise toutes les dix secondes. (Au cas où votre navigateur ne prendrait pas en charge cette balise, remplacez-la par un bout de JavaScript. Voyez le téléchargement en [1]). Si nous créons une page web qui comporte cette balise et pointons le navigateur sur elle, ce dernier la rechargera toutes les dix secondes (bien sûr, vous pouvez modifier le délai d'expiration). Si nous réécrivons la page toutes les dix secondes avec des données rafraîchies, alors le navigateur nous les montrera aussi.

Partage du travail

On peut aussi mettre la balise de rafraîchissement dans un fichier PHP au lieu d'un fichier HTML et le navigateur fera la même chose. Le fichier PHP contiendra également un script de lecture des données sur le port série. C'est là que les choses se compliquent parce que le langage PHP ne prend pas en charge nativement les ports série. Et même s'il le faisait, cela signifierait que chaque fois que le navigateur demanderait la (dernière version de la) page, le script devrait ouvrir le port série, récupérer quelques données et refermer de nouveau le port. Les données reçues en dehors de cette fenêtre seraient perdues. De plus, certains systèmes de type Arduino peuvent se réinitialiser à l'ouverture du port série, ce qui rend la confi-

Listage 1. Un script PHP qui lit des données depuis le port série et les écrit dans un fichier nommé « data.txt ».

```
<?php

// Linux $comPort = "/dev/ttyACM0";
$comPort = "COM15";

include "php_serial.class2.php";
$serial = new phpSerial;
$serial->deviceSet($comPort);

// On Windows (10 only?) all mode settings must be done in one go.
$cmd = "mode " . $comPort . " baud=115200 parity=n data=8 stop=1 to=off xon=off";
$serial->_exec($cmd);
$serial->deviceOpen();

echo "Waiting for data...\n";
sleep(2); // Wait for Arduino to finish booting.
$serial->serialflush();

while(1)
{
  $read = $serial->readPort();

  if (strlen($read)!=0)
  {
    $fp = fopen("data.txt","w");
    if ($fp!=false)
    {
      fwrite($fp,trim($read));
      fclose($fp);
    }
  }
}

?>
```

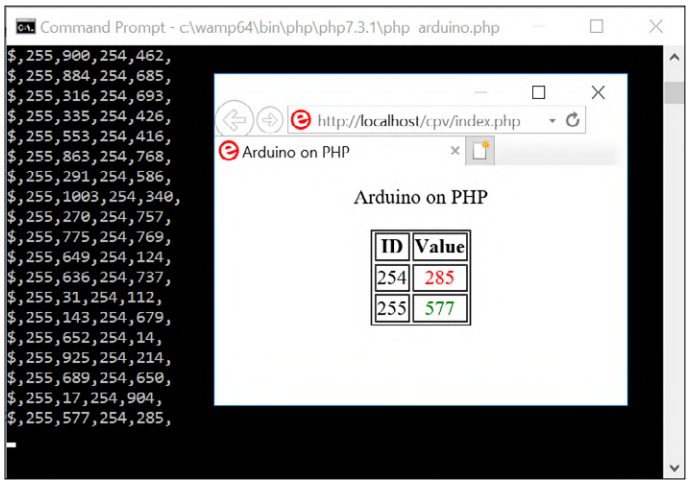


Figure 1. Le fichier PHP dynamique produit par le script PHP est servi par un serveur web WAMP à votre navigateur (voir sa barre d'adresse). Il présente, séparées par des virgules dans un tableur élémentaire, les données lues sur le port série. Les valeurs inférieures à 500 sont affichées en rouge, les autres en vert. Le titre de la fenêtre d'invite de commande montre la commande pour exécuter le script.

guration inutile. Une solution consiste à diviser le processus en deux sous-processus :

- Processus 1 : un script pour lire en permanence le port série et mettre à jour un fichier importé par la page web PHP (**listage 1**) ;
- Processus 2 : un navigateur qui recharge périodiquement la page web PHP de façon à rafraîchir les données (**listage 2, fig. 1**).

Oups, on demande un serveur web...

Ce stratagème résout le problème de l'ouverture et de la fermeture du port série et de la perte de données qui en résulte, mais il faut un script qui tourne en arrière-plan. Si c'est un script PHP, alors l'ordinateur doit être capable d'exécuter des scripts PHP. Il faut aussi un serveur web pour envoyer la page web PHP à un navigateur. Sinon, le navigateur va simplement montrer le code PHP de création de la page au lieu de la page elle-même. La manière traditionnelle d'y parvenir est d'installer ce qu'on appelle un package « AMP « ou « WAMP «. AMP signifie Apache-MySQL-PHP, le W est pour « Windows «, et ensemble ils forment un serveur web tout-en-un.

Oubliez PHP...

Nous avons essayé cette méthode et avons réussi à la faire marcher, mais pas sans problèmes. Outre les difficultés dans l'installation du serveur web, le problème principal rencontré a été d'obtenir de PHP qu'il ouvre de façon fiable un port série pour recevoir des données. En cherchant sur l'internet, il semble qu'il n'y ait qu'une seule bibliothèque PHP pour les communications série, *PHP Serial*. Toutes les autres paraissent en avoir été dérivées. Comme le mentionne l'auteur sur la page GitHub [2] « *Windows : cela semble fonctionner pour certaines personnes, pas pour d'autres* ». Nous étions clairement dans le second groupe. Pour que les communications série avec le PHP fonctionnent correctement, nous avons dû ouvrir puis immédiatement fermer le port série avec un programme de Terminal série (Tera Term par exemple) ; autrement c'est impossible. Nous avons donc abandonné la méthode PHP pour nous diriger vers le langage Python.

... utilisez plutôt Python

Le langage Python 3 avec la bibliothèque *pySerial* [3] s'est révélé parfaitement fonctionnel sur notre ordinateur sous Windows 10, donc nous avons écrit un script pour lire les données sur le port série et créer une page web avec ces données. Maintenant qu'il n'y a plus besoin de PHP, le script Python peut également produire un fichier HTML complet (**listage 3, fig. 2**). Tout le formatage de données effectué par la page

Listage 2. Cette page web PHP transforme le contenu d'un fichier nommé « data.txt » en tableau.

```
<?php

$page_title = "Arduino on PHP";

// Start of HTML page.
echo "<!DOCTYPE HTML PUBLIC '-//W3C//DTD HTML 4.01//EN' 'http://www.w3.org/TR/html4/strict.dtd'>";
echo "<html>"; // Page begin.
echo "<head><title>",$page_title,"</title>"; // Head begin.
echo "<meta http-equiv='refresh' content='1'>";
echo "<meta http-equiv='Content-Type' content='text/html; charset=UTF-8'>";
echo "<link rel='shortcut icon' href='favicon.ico' />";
echo "<link rel='icon' type='image/x-icon' href='favicon.ico' />";
echo "<link rel='icon' type='image/png' href='favicon.png' />";
echo "</head>"; // Head end.
echo "<body><center>"; // Body begin.

echo "<p>",$page_title,"</p>"; // Page title.

// Create a table from data file.
$handle = fopen("data.txt","r");
if ($handle!=NULL)
{
  // Read one line from the file, then close it.
  $data = fgets($handle);
  fclose($handle);

  // Synchronise to the data.
  if ($data[0]=='$')
  {
    // Remove whitespace.
    str_replace(' ','',$data);
    // Split data in fields separated by ','.
    // Expected format: "$,id1,value1,id2,value2,CRLF"
    list($startchar,$id1,$value1,$id2,$value2,$newline) = explode(",",$data);
    // Create array from list.
    $numbers = array($id1=>$value1,$id2=>$value2);
    // Sort array in ascending key order.
    ksort($numbers);

    // Table begin.
    echo "<table border='1' border-spacing='5' style='text-align:center;'>";
    echo "<tr><th>ID</th><th>Value</th></tr>";
    foreach ($numbers as $x => $x_value)
    {
      echo "<tr>"; // Table row begin.
      echo "<td>", $x, "</td>"; // Table column 1.
      echo "<td>"; // Table column 2 begin.
      if ($x_value>=500) echo "<font color='green'>";
      else echo "<font color='red'>";
      echo $x_value;
      echo "</font></td>"; // Table column 2 end.
      echo "</tr>"; // Table row end.
    }
    // Table end.
    echo "</table>";
  }
}
echo "</body>"; // Body end.
echo "</html>"; // Page end.
?>
```

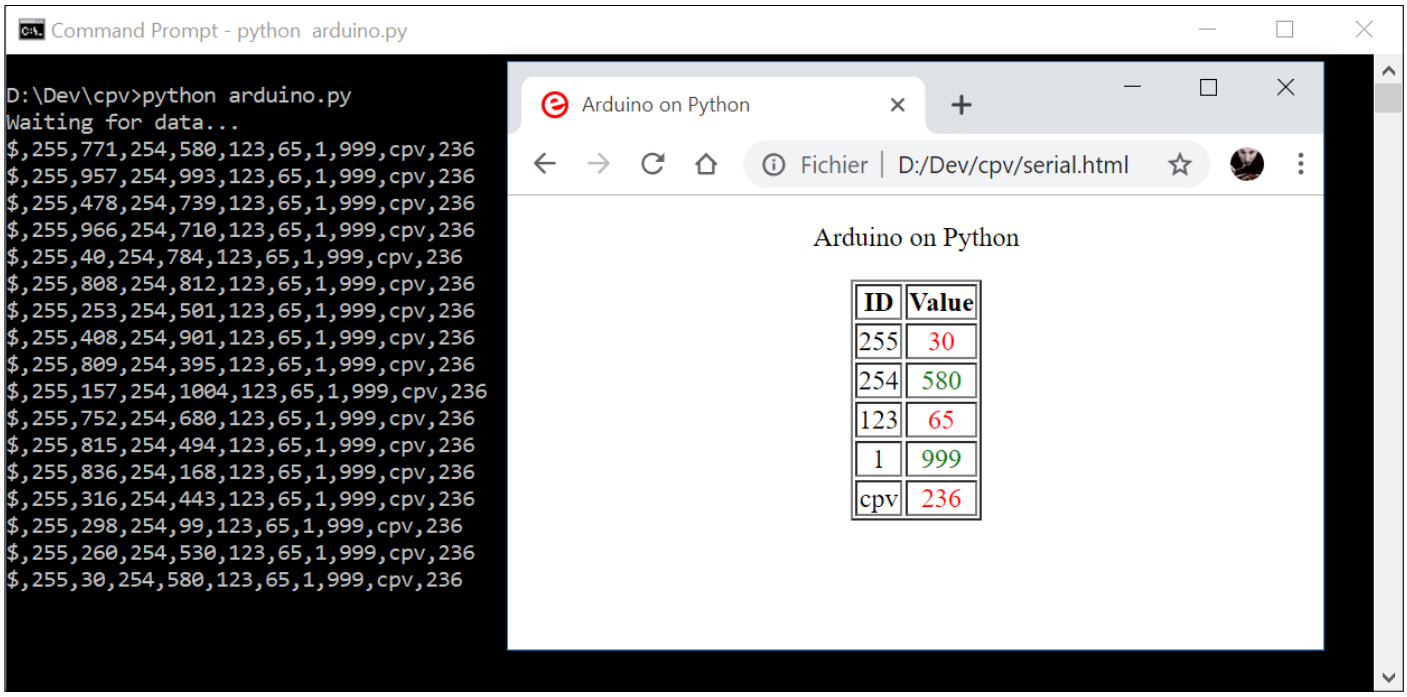


Figure 2. Cette fois-ci, un script Python produit un fichier HTML dynamique (comme on peut le voir sur la barre d'adresse du navigateur).

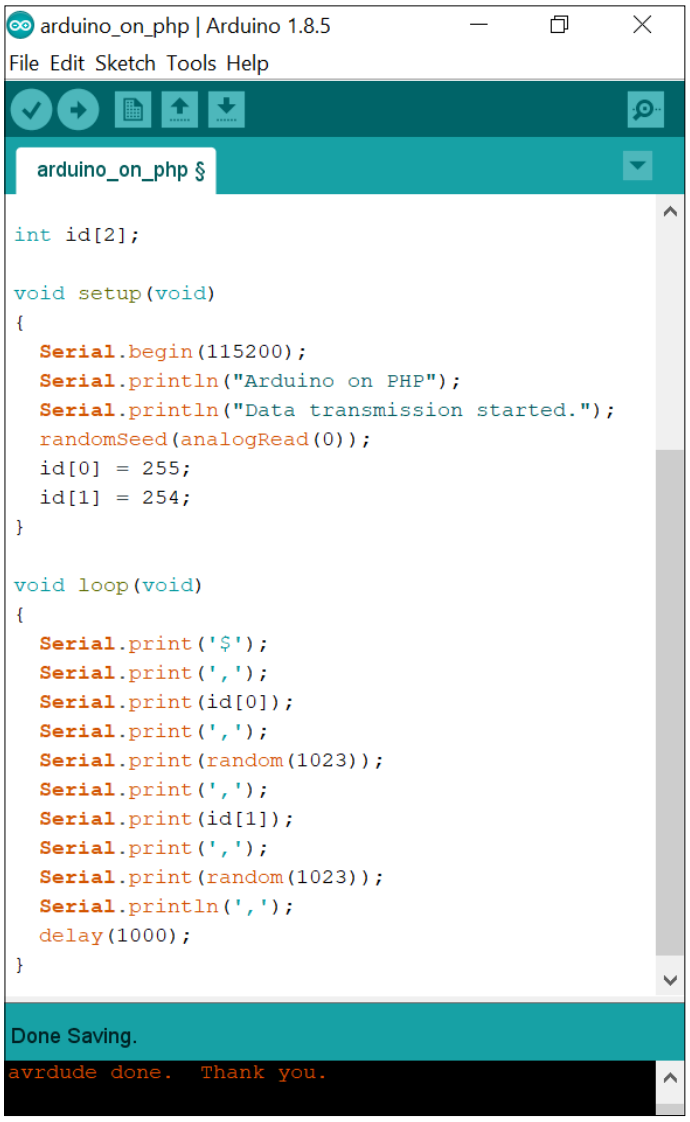


Figure 3. Ce croquis Arduino produit un flux de données série qu'on peut utiliser pour le test, le débogage ou l'écriture de scripts.

web PHP peut se faire directement en Python. Tout est maintenant au même endroit. La page web peut être servie par un serveur web (jeu de mots volontaire ;-)), mais un navigateur peut aussi afficher et rafraîchir la page sans serveur. Par conséquent, il n'y a plus non plus besoin de package (W)AMP, ce qui simplifie tout.

Notre opinion

Nous avons présenté ici une méthode d'affichage de données série dans un navigateur web. La méthode n'est ni nouvelle, ni exclusive, ni « la meilleure ». Si vous en connaissez une autre – plus simple, plus élégante, peu importe – merci de la partager [1]. Et, bien sûr, pas besoin d'écrire le script en Python, n'hésitez pas à utiliser n'importe quel langage capable d'établir des communications série et d'écrire des fichiers. L'avantage de Python avec *pySerial* est qu'il fonctionnera sur des machines Windows, macOS et Linux (ou autres).

Le code conçu pour cet article sous forme de scripts PHP et Python ainsi qu'un croquis Arduino peuvent être téléchargés depuis [1]. ⏮

(170111-04 – version française : Denis Lafourcade)

Liens

[1] Page de l'article : www.elektormagazine.fr/170111-04

[2] Bibliothèque PHP Serial : https://github.com/Xowap/PHP-Serial

[3] Bibliothèque pySerial : https://pypi.org/project/pyserial/

Listage 3. Un script Python qui lit des données depuis le port série et produit un fichier HTML à partir de ces données.

```
import serial
import time

file_name = “serial.html” # Once created, open this file in a browser.

# Adapt serial port nr. & baud rate to your system.
serial_port = ‘COM15’
baudrate = 115200

page_title = “Arduino on Python”;

def write_page(data_list):
    fo = open(file_name,”w+”)
    # Start of HTML page.
    fo.write(“<!DOCTYPE HTML PUBLIC ‘-//W3C//DTD HTML 4.01//EN’ ‘http://www.w3.org/TR/html4/strict.dtd’>”)
    fo.write(“<html><head><title>”+page_title+”</title>”) # Page & Head begin.
    fo.write(“<meta http-equiv=’refresh’ content=’1’>”)
    fo.write(“<meta http-equiv=’Content-Type’ content=’text/html; charset=UTF-8’>”)
    fo.write(“<link rel=’shortcut icon’ href=’favicon.ico’ />”)
    fo.write(“<link rel=’icon’ type=’image/x-icon’ href=’favicon.ico’ />”)
    fo.write(“<link rel=’icon’ type=’image/png’ href=’favicon.png’ />”)
    fo.write(“</head><body><center><p>”+page_title+”</p>”) # Head end, body begin.

    # Table begin.
    fo.write(“<table border=’1’ border-spacing=’5’ style=’text-align:center;’>”)
    fo.write(“<tr><th>ID</th><th>Value</th></tr>”)
    for i in range(0,len(data_list),2):
        fo.write(“<tr>”) # Table row begin.
        fo.write(“<td>”+data_list[i]+”</td>”) # Table column 1.
        fo.write(“<td>”) # Table column 2 begin.
        fo.write(“<font color=’”)
        # Values >= 500 will be printed in green, smaller values will be red.
        if (int(data_list[i+1])>=500): fo.write(“green”)
        else: fo.write(“red”)
        fo.write(“’>”)
        fo.write(data_list[i+1])
        fo.write(“</font></td>”) # Table column 2 end.
        fo.write(“</tr>”) # Table row end.
    fo.write(“</table>”) # Table end.
    fo.write(“</body>”) # Body end.
    fo.write(“</html>”) # Page end.
    # Done, close file.
    fo.close()

s = serial.Serial(serial_port,baudrate) # Open serial port.
s.dtr = 0 # Reset Arduino.
s.dtr = 1
print(“Waiting for data...”);
time.sleep(2) # Wait for Arduino to finish booting.
s.reset_input_buffer() # Delete any stale data.

while 1:
    data_str = s.readline().decode() # Read data & convert bytes to string type.
    # Clean up input data.
    # Expected format: “$,id1,value1,id2,value2,...,CRLF”
    data_str = data_str.replace(‘ ‘,’’) # Remove whitespace.
    data_str = data_str.replace(‘\r’,’’) # Remove return.
    data_str = data_str.replace(‘\n’,’’) # Remove new line.
    data_str += ‘123,65,1,999,cpv,236’ # Add some more data
    print(data_str)
    # Split data in fields separated by ‘,’.
    data_list = data_str.split(“,”)
    del data_list[0] # Remove ‘$’
    # Write HTML page.
    write_page(data_list)
```

elektor start-up challenge Paris 2019

Une rampe de lancement internationale pour les start-ups

Denis Meyer
Mariline Thiebaut-Brodier

N'attendez pas qu'Elon Musk vous invite à bord de SpaceX !

Après l'Allemagne et les Pays-Bas, cette année Elektor fera escale à Paris avec un nouveau concours pour les jeunes entreprises dont l'activité est liée à l'électronique embarquée. Cela se passera dans le cadre du nouveau salon Forum de l'électronique en septembre 2019.

Ce concours baptisé *elektor start-up challenge* vise à stimuler la croissance de jeunes pousses après leurs débuts prometteurs.

Le choix des gagnants sera guidé par la viabilité industrielle des projets, leur utilité et bien sûr leur originalité.

L' *elektor start-up challenge* est le fruit de notre coopération étroite avec Cap'tronic, le programme d'aide aux PME en matière de systèmes électroniques et de logiciel embarqué, et GL Events Exhibitions, l'organisateur du salon.

Qui peut embarquer ?
Vous par exemple, qui développez un produit, un logiciel ou un service lié à l'électronique. L'*elektor start-up challenge* vous donne l'occasion de venir présenter votre projet aux professionnels de la branche.

Les prédécents concours organisés par Elektor, notamment dans le cadre d'*electronica* à Munich, le plus grand salon d'électronique au monde, ont récompensé des projets remarquables dans différents domaines.

Pourquoi monter à bord de la fusée ?
Si vous vous inscrivez à l'*elektor start-up challenge* de Paris 2019 et que votre candidature est retenue, vous serez accueilli avec les autres concurrents sur l'aire de l'*elektor start-up challenge*, un grand stand central du salon **Forum de l'électronique** (Paris Expo - Porte de Versailles, du 24 au 26.09.2019).

Vous pourrez :
- y présenter votre projet sur votre propre pupitre
- concourir au côté des autres finalistes
- attirer l'attention de partenaires potentiels

Avant, pendant et après le salon, vous bénéficierez d'une large visibilité :
- vous et votre start-up serez présents dans les supports du Forum de l'électronique : catalogue des exposants, liste des exposants, site officiel...
- vous et votre start-up serez présentés à l'ensemble des lecteurs du magazine Elektor via tous nos supports (papier, web, réseaux sociaux, chaîne YouTube...) en France et dans les pays francophones
- vous et votre start-up serez présentés aux adhérents du programme Cap'tronic

Pour vous et pour votre start-up, le summum ce serait d'emporter le gros lot. En effet, le gagnant de l'*elektor start-up challenge* à Paris recevra son ticket d'entrée pour le prochain salon *electronica* à Munich : vous serez sélectionné d'office pour y participer au concours international *Fast Forward Award* au cours duquel vous présenterez votre projet sur le stand et bénéficierez de la très large publicité donnée à cet évènement.

Elektor Start-up Challenge est organisé en collaboration avec :

CAP'TRONIC Systèmes électroniques et logiciel embarqué

Forum de l'électronique
Salon de l'innovation et des solutions électroniques

Présenter sa candidature

- Pour participer à l'*elektor start-up challenge* dans le cadre du *Forum de l'Électronique*, le candidat peut nous contacter directement à l'adresse redaction@elektor.fr. Il peut aussi créer un compte (gratuit) sur le site www.elektormagazine.fr.
- Il devra fournir une documentation la plus complète possible de son projet afin que le jury puisse apprécier clairement la mission de la start-up, la technologie du projet, se faire une idée du produit et, le cas échéant, des services fournis. Toutes les propositions seront évaluées par le jury de l'*elektor start-up challenge*.
- Pour être considérée comme une start-up, son entreprise doit avoir été créée il y a moins de cinq ans.
- Si sa start-up est admise à participer l'*elektor start-up challenge*, le représentant de l'entreprise accepte d'être personnellement présent à Paris lors du Forum de l'Électronique 2019.
- Toutes les conditions générales de participation au *Elektor Start-Up Challenge* de Paris sont disponibles en ligne.

Participants et vainqueurs du concours *Fast Forward electronica 2018*.

Le rendez-vous de votre start-up avec son avenir

Où ? Paris Expo Porte de Versailles

Quand ? Du 24 au 26 septembre 2019

Comment ? Sur inscription au plus tard le mercredi 31 juillet 2019.

Rendez-vous sur :
www.elektormagazine.fr/escparis2019

Le gagnant participera d'office au concours *electronica* à Munich l'an prochain.

Le jury au travail, présidé par Clemens Valens d'Elektor Labs.

Attention :

la date limite d'inscription est fixée au mercredi 31 juillet 2019.

En cas de forte affluence, il est possible que le nombre limité de places nous contraigne à avancer la date de clôture des inscriptions. Ne tardez pas à vous rendre sur la page **www.elektormagazine.fr/escparis2019** pour vous inscrire.

Les conditions générales de participation au *elektor start-up challenge* de Paris sont disponibles en ligne.

La réunion préparatoire entre Elektor et ses partenaires.

190232-01

nouvelle carte CPLD avec MAX10

Alfred Rosenkränzer

La carte CPLD (*Complex Programmable Logic Device*) s'améliore : grâce à la puce MAX10M08, le cœur de la carte est animé par une puce CPLD avec davantage d'éléments logiques qu'avec la puce précédente, mais à cela s'ajoutent des fonctions FPGA (*Field Programmable Gate Array*) typiques telles que des boucles à verrouillage de phase (*Phase Locked Loop, PLL*), des convertisseurs A/N et des microcontrôleurs à cœur programmable (*softcore*).

Dans le numéro de mai-juin 2018 d'Elektor, l'auteur a proposé une carte équipée d'une puce CPLD de la famille MAX-II d'Altera (aujourd'hui Intel) [1]. La puce utilisée dans ce projet, une EPM570, est une CPLD avec 570 éléments logiques dans un boîtier TQFP100 (*Thin Quad Flat Pack* à 100 pattes). La carte acceptait aussi des puces EPM240, plus petites mais avec un brochage compatible. Du fait de la taille relativement petite du boîtier, il était possible de concevoir une carte de prototypage de telle sorte que la puce était pratiquement utilisable comme un circuit DIL. Avec 570 éléments logiques, on peut déjà réaliser une bonne quantité d'applications, mais s'ils ne suffisent pas, il faut se tourner vers une autre famille de CPLD. Comme la famille MAX-V n'offre que

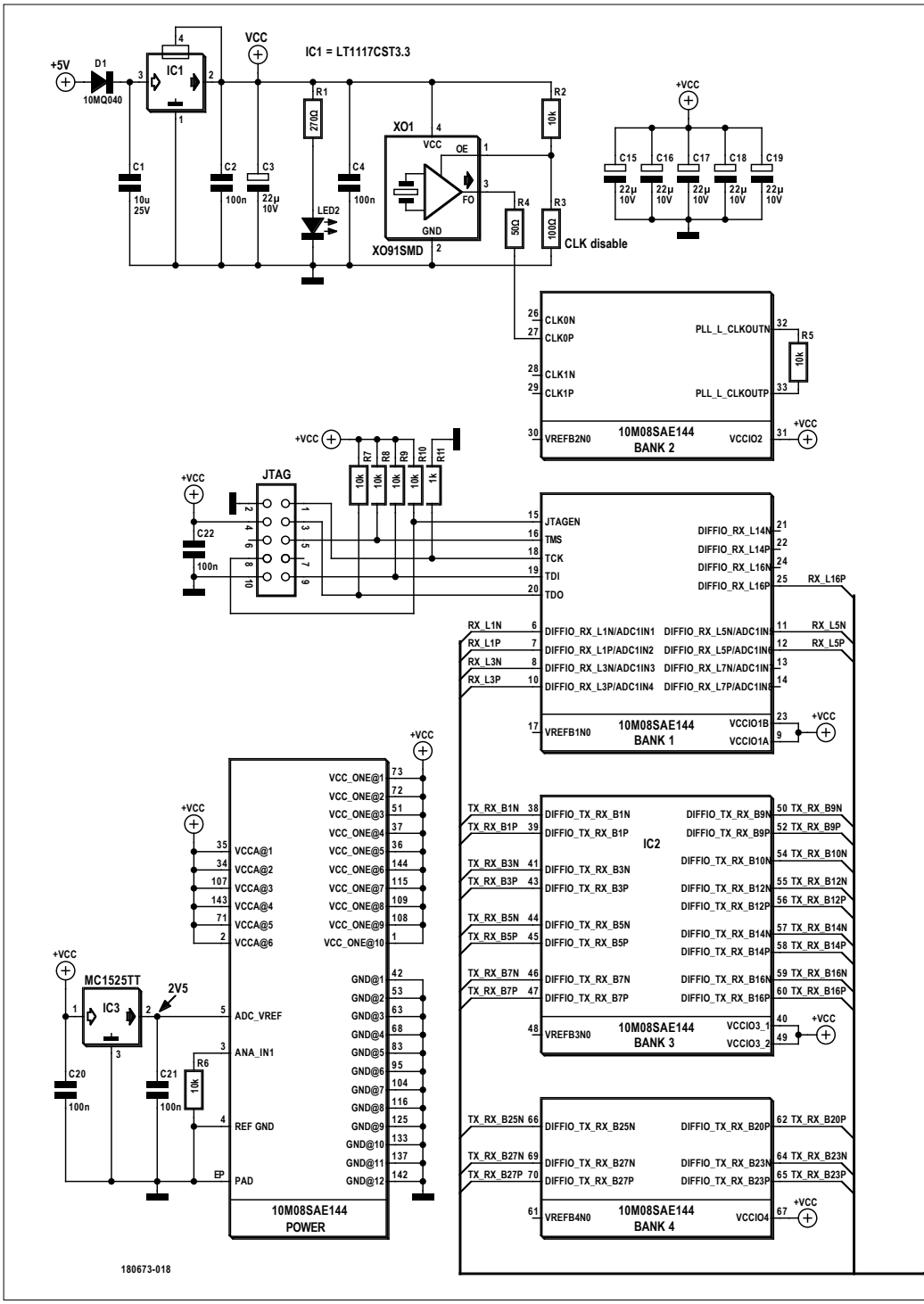
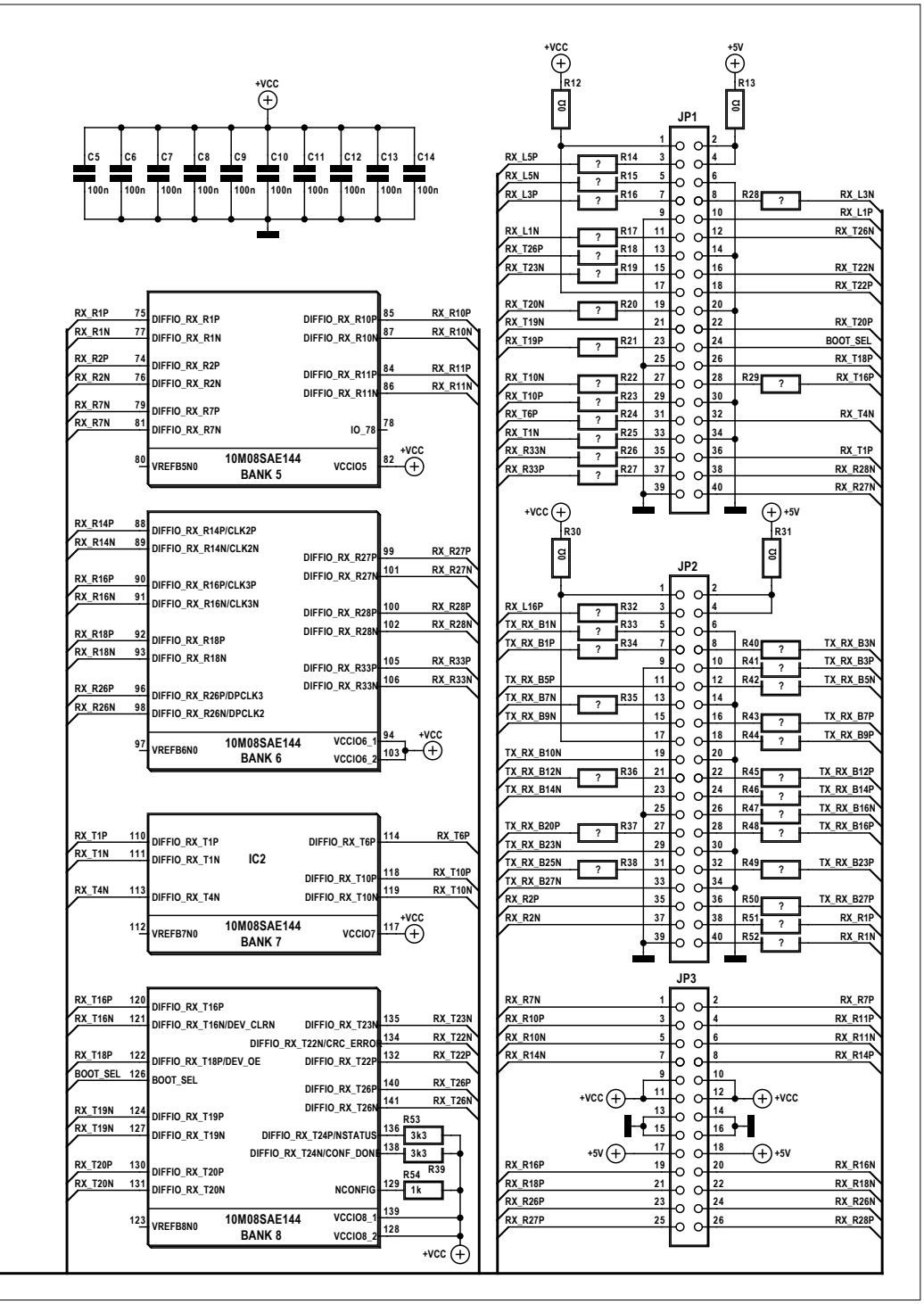


Figure 1. Sur le schéma, la puce MAX-10M08SAE144 est divisée en plusieurs sections pour une meilleure compréhension.

davantage de logique avec la famille MAX10



peu d'avantages par rapport à la famille MAX-II, pourquoi ne pas passer directement à la famille la plus récente d'Intel, la MAX10 ? Cette famille [2] est composée de sept groupes de produits CPLD, chacun avec des variantes diverses. On a le choix : de 2.000 à 50.000 éléments logiques, alimentation à tension simple ou double, avec régulateur 3,3 V intégré (auquel cas, le modèle est obligatoirement mono-tension), différentes tailles de pages de mémoire flash, des convertisseurs A/N, un capteur de température, flash à double configuration (deux images dynamiques utilisables sur la même puce) et bien d'autres fonctions.

De la CPLD à la FPGA

Au-delà des pures fonctions CPLD, la famille MAX10 possède des fonctions FPGA totalement opérationnelles, telles que des PLL internes (jusqu'à 4) pour produire des signaux d'horloge plus rapides, des processeurs programmables NIOS, des blocs de traitement numérique du signal et des pilotes de mémoires DDR3. Toutes ces variantes sont disponibles dans des boîtiers variés (de la galette de semi-conducteur (wafer) aux gros boîtiers avec des centaines de broches d'E/S). Voilà d'ailleurs notre premier critère de choix. Le seul boîtier encore soudable par l'amateur (c'est-à-dire avec un fer à souder) est un EQFP de 144 pattes. Avec ses 22×22 mm^2, ce modèle nécessite un circuit imprimé un peu plus grand que son prédécesseur, mais offre en échange 101 entrées-sorties utilisables (pour toutes les variantes). Pour plus de détails, consulter le manuel utilisateur de la famille MAX10 [3].

Composants à ajouter

La **figure 1** montre le schéma de la carte pour cette nouvelle puce CPLD plus « grosse » en taille et en fonctions.

Pin	JP2		JP1		Rpi GPIO		Pin	Pin	JP3		Pin
1	+3.3 V	+5 V	+3.3 V	+5 V	+3.3 V	+5 V	2	1	P81	P79	2
3	P25	+5 V	P12	+5 V	GPIO 2	+5 V	4	3	P85	P84	4
5	P38	GND	P11	GND	GPIO 3	GND	6	5	P87	P86	6
7	P39	P41	P10	P8	GPIO 4	GPIO 14	8	7	P89	P88	8
9	GND	P43	GND	P7	GND	GPIO 15	10	9	+3.3 V	+3.3 V	10
11	P45	P44	P6	P141	GPIO 17	GPIO 18	12	11	+3.3 V	+3.3 V	12
13	P46	GND	P140	GND	GPIO 27	GND	14	13	GND	GND	14
15	P50	P47	P135	P134	GPIO 22	GPIO 23	16	15	GND	GND	16
17	+3.3 V	P52	+3.3 V	P132	+3.3 V	GPIO 24	18	17	+5 V	+5 V	18
19	P54	GND	P131	GND	GPIO 10	GND	20	19	P90	P91	20
21	P55	P56	P127	P130	GPIO 9	GPIO 25	22	21	P92	P93	22
23	P57	P58	P124	P126	GPIO 11	GPIO 8	24	23	P96	P98	24
25	GND	P59	GND	P122	GND	GPIO 7	26	25	P99	P100	26
27	P62	P60	P119	P120	GPIO 0	GPIO 1	28				
29	P64	GND	P118	GND	GPIO 5	GND	30				
31	P66	P65	P114	P113	GPIO 6	GPIO 12	32				
33	P69	GND	P111	GND	GPIO 13	GND	34				
35	P74	P70	P106	P110	GPIO 19	GPIO 16	36				
37	P76	P75	P105	P102	GPIO 26	GPIO 20	38				
39	GND	P77	GND	P101	GND	GPIO 21	40				

Figure 2. Brochage des trois connecteurs de la carte CPLD et du connecteur GPIO du RPi.

On y distingue un régulateur de tension avec ses condensateurs de filtrage, une horloge et le connecteur JTAG pour la programmation. On y trouve encore une référence de tension pour le convertisseur A/N intégré à la CPLD (*bank 1A*).
Le prototype est équipé d'une puce 10M08 qui possède huit unités logiques (LE), une PLL et un convertisseur A/N. La carte est compatible avec toutes les puces de la famille MAX10 de 2.000 à 50.000 éléments logiques, dans la mesure où elles sont logées dans un boîtier EQFP-144. La conception du circuit imprimé s'appuie sur quatre couches pour garantir un fonctionnement sûr, même aux fréquences d'horloge les plus élevées.
Comme avec la précédente carte, il y a une bonne raison de ne pas router vers les connecteurs d'accès toutes les entrées/sorties possibles. La carte peut être utilisée comme une grosse puce DIL installée sur une platine d'essai ou une carte à trous, mais elle possède un raffinement : le brochage des deux connecteurs latéraux est calqué sur celui du connecteur GPIO à 2×20 broches du Raspberry Pi (à partir de la version 2). On peut donc les enficher directement sur le RPi, ce qui permet de combiner la puissance de calcul et l'interface du RPi avec les possibilités matérielles d'une puce CPLD moderne. La **figure 2** présente l'affectation des broches des trois connecteurs d'accès (avec les broches de la puce CPLD) comparée au connecteur GPIO du RPi. Les broches représentées côte à côte avec la même couleur forment une paire d'entrées ou de sorties différentielles qui peuvent bien entendu être aussi utilisées isolément, comme les broches sans couleur.

Variantes

Il s'ensuit toute une série de possibilités. Comme le montre la **figure 3**, la carte peut être enfichée sur le RPi avec les

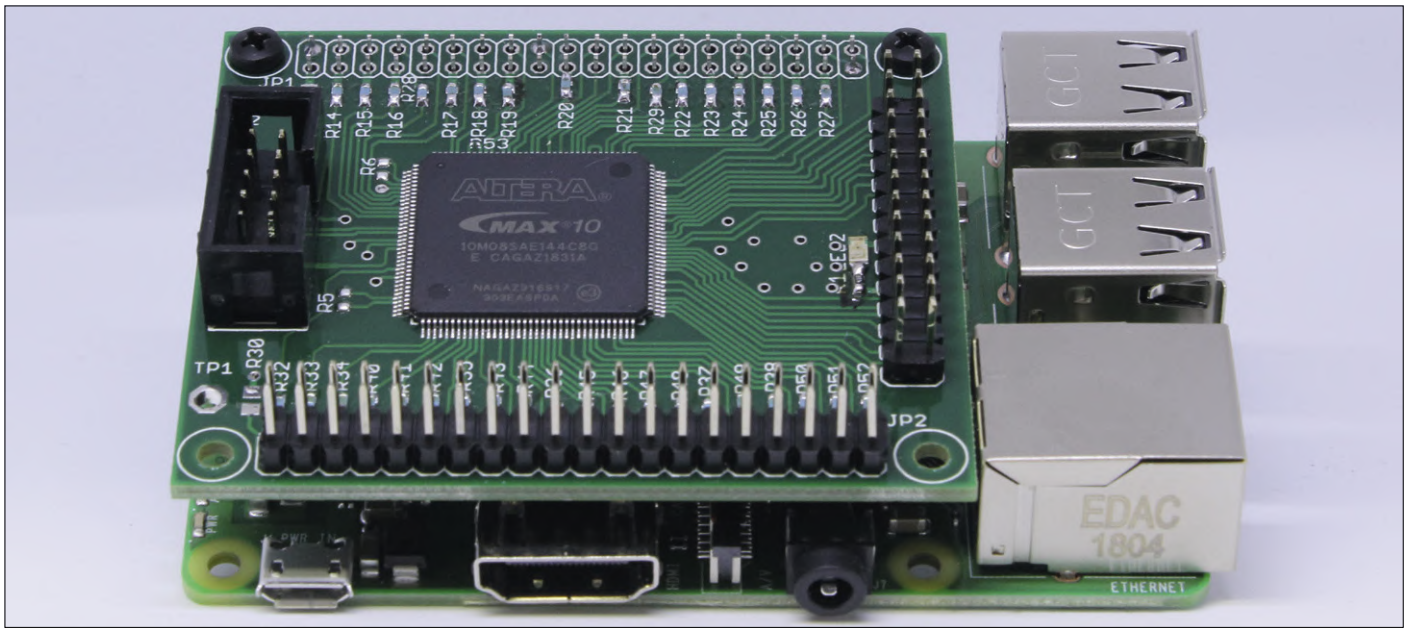


Figure 3. La carte CPLD enfichée sur le RPi 3+. Les trous de fixation sont alignés.

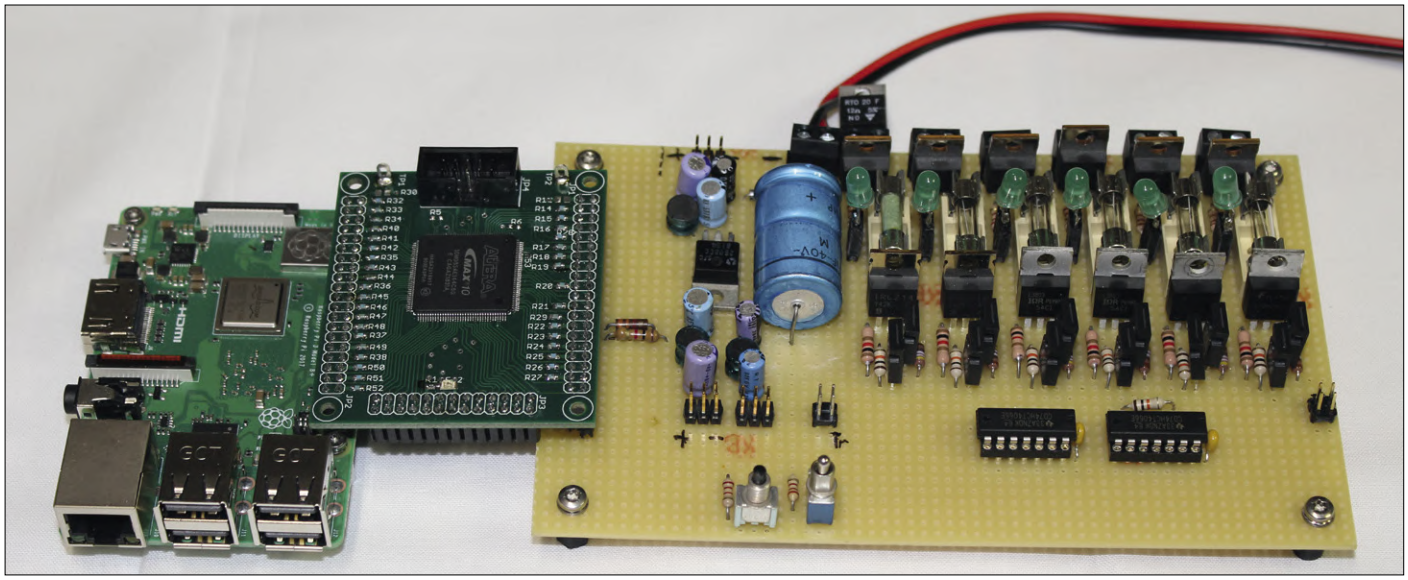


Figure 4. Le RPi pilote la puce CPLD qui elle-même gère le circuit l'électronique sur la carte d'application.

trous de fixation alignés. Le connecteur JP1 est constitué de broches femelles orientées vers le bas et enfichées sur les broches GPIO du RPi. Les autres connecteurs sont orientés vers le haut et librement accessibles.

La **figure 4** montre comment le RPi pilote la puce CPLD via le connecteur JP2, laquelle gère un circuit électronique (par ex. un étage de puissance) sur carte à trous. On peut alimenter les deux cartes en commun ou séparément, c'est configurable avec les ponts (résistances de 0 Ω) R12, R13, R30 et R31.

On peut aussi utiliser un câble plat au cas où les cartes seraient à distance l'une de l'autre (**fig. 5**). Et, pour finir, on peut employer la carte avec la puce 10M08SAE144 toute seule, sans le concours d'un RPi !

(180673-03 – version française : Helmut Müller)

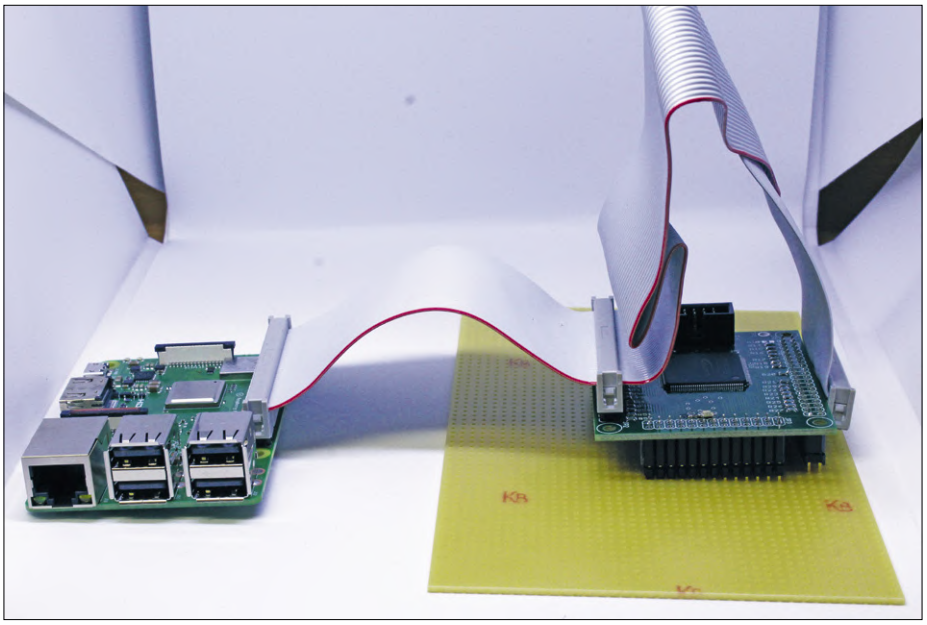


Figure 5. Configuration répartie avec câbles en nappe.

Premier arrivé, premier servi !

Quelques circuits imprimés vierges ou partiellement équipés (de la puce CPLD seulement) sont disponibles chez l'auteur. N'hésitez pas à en demander un.

alfred_rosenkraenzer@gmx.de

Liens

[1] Carte CPLD au format DIL, Elektor 5-6/2018 : www.elektormagazine.fr/160425

[2] Vue d'ensemble de la famille MAX 10 : www.intel.fr/content/www/fr/fr/products/programmable/fpga/max-10.html

[3] Ressources pour la famille MAX 10 : www.intel.com/content/www/us/en/programmable/products/fpga/max-series/max-10/support.html

PROJET DE LECTEUR

surveillance de la consommation d'eau avec l'ESP32

avec alerte par SMS en cas de fuite

Denis Lafourcade (Tournefeuille)

Il y a deux catégories de gens : ceux qui ont déjà eu une grosse fuite d'eau, et ceux qui n'en ont pas ENCORE eu une… Les assurances proposent de couvrir ce risque, mais ne vaut-il pas mieux anticiper et alerter en cas de présomption de fuite ? Le montage proposé ici repose sur une carte qui combine un contrôleur ESP32 pour la connexion au Wi-Fi et un écran OLED pour l'information sur place.

Mieux vaut prévenir que guérir !
Tout ce qui se passe en aval de votre compteur d'eau est sous votre responsabilité. En cas de fuite, la surconsommation sera donc à votre charge, et cela peut coûter très cher :

- Un goutte à goutte : 4 l/h, soit 35 m³ par an, environ 125 €
- Une chasse d'eau qui fuit : 25 l/h, soit 220 m³ par an, environ 800 €
- Une fissure sur la canalisation enterrée entre le compteur et la maison : plusieurs m³ par heure, plusieurs milliers d'euros par semaine !

Les entreprises de distribution de l'eau commencent à installer des systèmes de télérelève, qui incluent une alerte en cas de présomption de fuite. Mais ce déploiement est lent, et la majorité des communes n'est pas encore équipée. Il existe en revanche des compteurs d'eau dotés d'un émetteur d'impulsions qui envoie un signal à chaque litre consommé, et certains compteurs peuvent être équipés d'un tel émetteur à postériori. Le système présenté ici exploite cette possibilité et permet d'assurer une veille permanente de votre consommation.

En l'espace de quelques mois, trois de mes voisins ont découvert lors de la réception de leur facture d'eau qu'ils avaient une fuite entre le compteur et la maison. La fuite était restée invisible pendant plusieurs semaines, entraînant une surconsommation énorme et une facture de plusieurs milliers d'euros !
Craignant de devoir souffrir d'un désagrément similaire, je me suis mis à rechercher des solutions pour surveiller ma consommation d'eau. Par chance, beaucoup de compteurs et le mien en particulier sont munis d'un système de comptage à rouleaux. Le rouleau des unités dispose d'un aimant sur le chiffre zéro, ce qui permet de détecter un tour, ce qui correspond en général à un litre de consommation. On trouve sur le marché des émetteurs d'impulsions (**fig. 1)** qui se clipsent sur le compteur et produisent un signal radio ou filaire à chaque passage de l'aimant devant un détecteur à effet Hall. Ces émetteurs sont alimentés par une pile au lithium d'une durée de vie d'environ quinze ans et ne nécessitent donc pas une alimentation externe.

Avec ou sans fil ?
Dans un premier temps, je me suis intéressé à la transmission de l'information entre l'émetteur et la partie intelligente du

Figure 1. Exemple in situ d'émetteur d'impulsions.

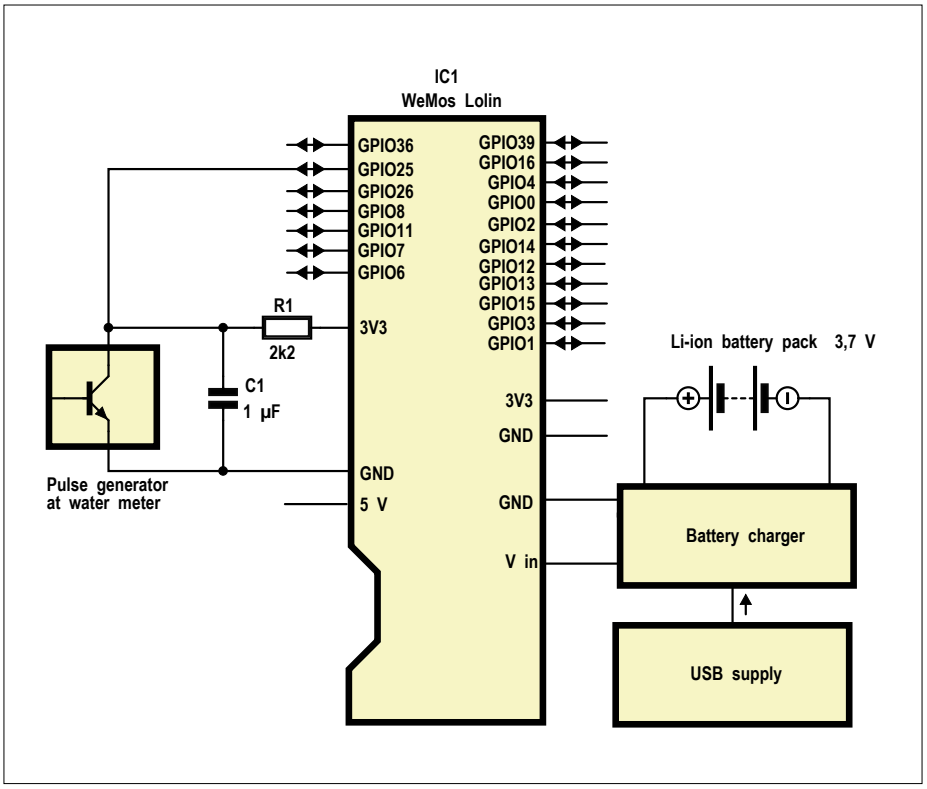


Figure 2. Schéma du montage.

système. La version sans fil des émetteurs d'impulsions paraissait séduisante, mais malheureusement le protocole utilisé n'est pas publié et malgré mes recherches, je n'ai pas trouvé d'information fiable sur le sujet (si vous en savez plus, n'hésitez pas à m'en faire part). Comme le regard de mon compteur d'eau et celui de ma ligne téléphonique sont très proches, j'en ai profité pour simplement tirer un câble bifilaire entre l'émetteur d'impulsions et l'intérieur de mon habitation en passant par la gaine du câble de téléphone. Pour d'autres installations, comme l'opérateur avait laissé en place le câble téléphonique après l'installation de la fibre optique, j'ai utilisé une des paires de ce câble et évité ainsi d'en cheminer un autre.

Principe général

Avant de concevoir le système, il faut en arrêter les spécifications :

- Acquisition et traitement des impulsions de consommation ;
- Affichage de la consommation horaire, journalière et totale ;
- Algorithme de détection de présomption de fuite ;
- Alimentation sauvegardée ;
- Sauvegarde des données en mémoire non volatile ;
- Accès réseau par Wi-Fi et synchronisation de l'heure par NTP ;
- Serveur web pour la visualisation et le contrôle à distance ;
- Système d'alerte par SMS.

Montage électronique

Ces spécifications ont conduit à l'architecture illustrée par la **figure 2**. Le montage repose sur un module Lolin , composé d'un système sur puce (SoC) ESP-WROOM-32 et d'un écran OLED de 128 × 64 pixels (brochage en **figure 3**). Cette carte est disponible en plusieurs versions sur l'internet, pour quelques euros seulement, ainsi que dans la boutique d'Elektor. L'ESP32 dispose du Wi-Fi et d'une capacité de mémoire

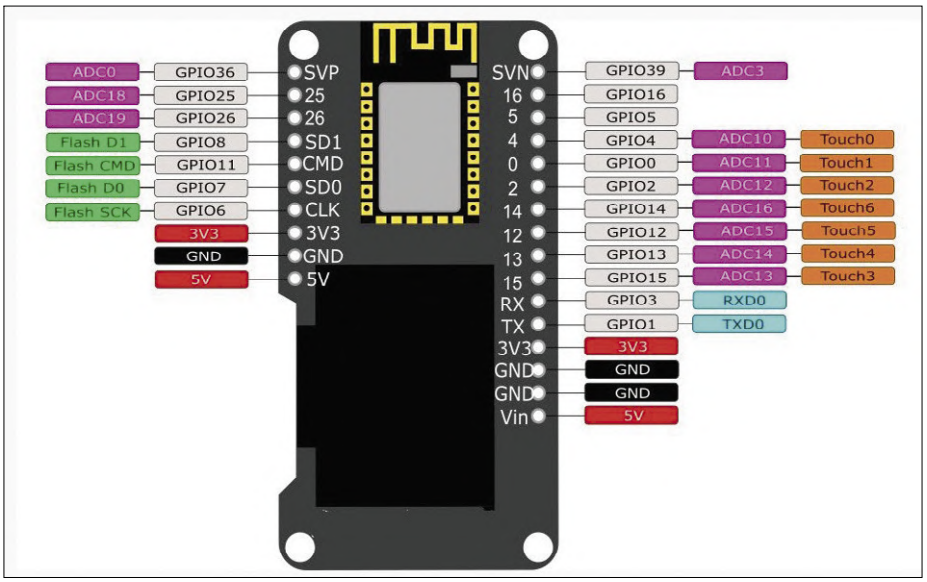


Figure 3. Brochage du module Lolin.

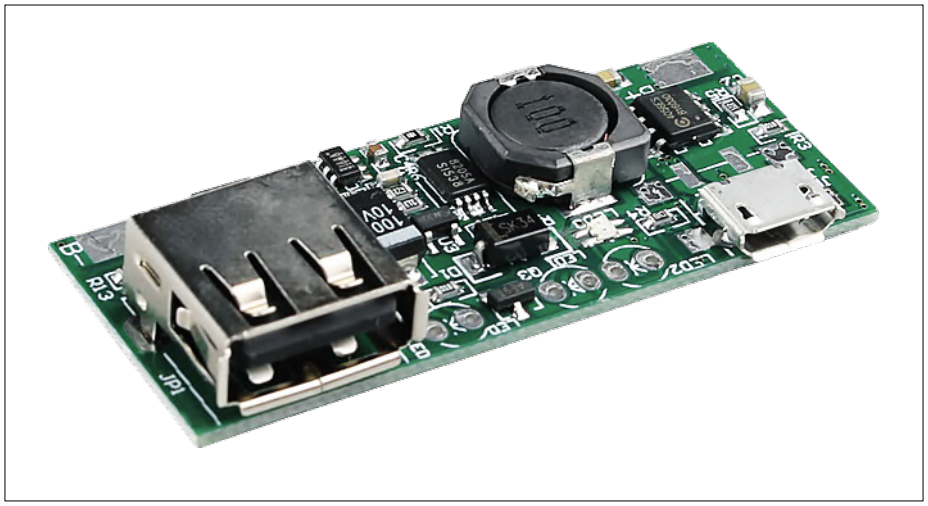


Figure 4. Module d'alimentation-chargeur de batterie.

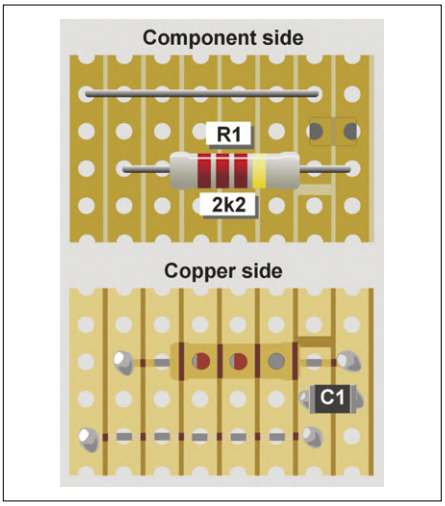


Figure 5. Carte d'interface (connectée entre les broches GND et GPIO25 du module Lolin).

suffisante pour des applications conséquentes. Il est de plus compatible avec l'EDI Arduino, ce qui le rend particulièrement simple à utiliser. L'alimentation est sécurisée par une batterie Li-Ion munie d'un chargeur alimenté en 5 V via une prise micro-USB. Ce chargeur (**fig. 4**) conçu pour les batteries Li-Ion maintient la charge de la batterie et la protège contre les décharges et surcharges excessives. Il est disponible sous plusieurs références sur l'internet (recherchez « 3.7V Li-ion Battery Mini USB To USB A Power Supply Module »). En cas de coupure du secteur, la batterie peut alimenter le système pendant au moins 36 h.

Figure 6. Les trois pages de l'écran OLED.

Les impulsions émises par l'émetteur sont du type collecteur ouvert, avec passage à la masse à chaque litre consommé. L'acquisition de ces impulsions est très simple : une résistance de tirage au plus relie la ligne de données à la broche 3,3 V de la carte et une capacité de 1µF permet de filtrer les éventuelles perturbations sur la ligne. La ligne de donnée est connectée à la broche GPIO 25 de l'ESP32, et la ligne de masse à sa broche GND. La connexion de ces deux fils, de la résistance et du condensateur est réalisée sur un morceau de plaque d'expérimentation soudée sur la barrette mâle du module (**fig. 5**).

Logiciel

Le cœur du système est bien sûr le logiciel (disponible sur la page de l'article [2]). Là encore, il ne s'agissait pas de réinventer la roue, car beaucoup de choses existent et sont mises à disposition par la communauté des « makers ». Les bibliothèques suivantes fournissent la plupart des services nécessaires à nos besoins :

- `Wifi`, `ESPmDNS`, `WiFiUdp` et `Arduino OTA` pour les accès Wi-Fi et la mise à jour du logiciel sans fil ;
- `TimeLib` pour la gestion du temps par NTP (*Network Time Protocol*) ;
- `ESP32WebServer` et `WiFiClient` pour le serveur web ;
- `Wire`, `SSD1306` et `OLEDDisplayUi` pour la partie afficheur OLED ;
- Et bien sûr toute la puissance des fonctions de l'`ESP32 Arduino Core`.

Après l'inclusion de toutes ces bibliothèques et la définition des structures et constantes du programme, on retrouve l'architecture classique d'un programme Arduino, à savoir une routine d'initialisation `setup()` et une boucle principale `loop()`. La routine `IRAM_ATTR handleInterrupt()` prend en charge l'interruption déclenchée par le passage à zéro de la ligne de données raccordée à GPIO 25, et ne fait qu'incrémenter la valeur du compteur de litres d'eau consommés. Un filtrage logiciel permet de se prémunir d'éventuels « rebonds » sur la ligne.
La boucle principale prend alors en charge des opérations suivantes.

- **Affichage sur l'écran OLED**

La bibliothèque `OLEDDisplayUi` permet de réaliser simplement un affichage très élégant : on a défini ici trois pages qui défilent toutes les cinq secondes (**fig. 6**). La première page affiche la date (l'heure est affichée en haut à droite des trois pages), la consommation du jour et l'état de détection de fuite. La seconde page présente la valeur du compteur, la valeur du dernier relevé et la consommation depuis celui-ci. La troisième page indique l'état de la connexion Wi-Fi ainsi que l'adresse IP affectée.

- **Serveur Web**

Le serveur est à l'écoute sur un port défini lors de l'initialisation, et dont vous

devrez assurer la redirection sur votre routeur (*box*). Lorsqu'un utilisateur se connecte grâce à un navigateur, le serveur présente une page avec trois onglets (**fig. 7**). L'onglet `Visualisation`, rafraîchi toutes les 5 s, montre, en plus des informations visibles sur l'afficheur OLED, un tableau de consommation heure par heure sur les dernières 24 heures. L'onglet `Saisie` permet de définir différents paramètres : valeur initiale du compteur, valeur du dernier relevé officiel et valeur du seuil d'alerte de consommation journalière (voir plus loin). L'accès à cet onglet est protégé par un nom d'utilisateur et un mot de passe. L'onglet `À propos` rappelle les fonctions du système et la version du logiciel.

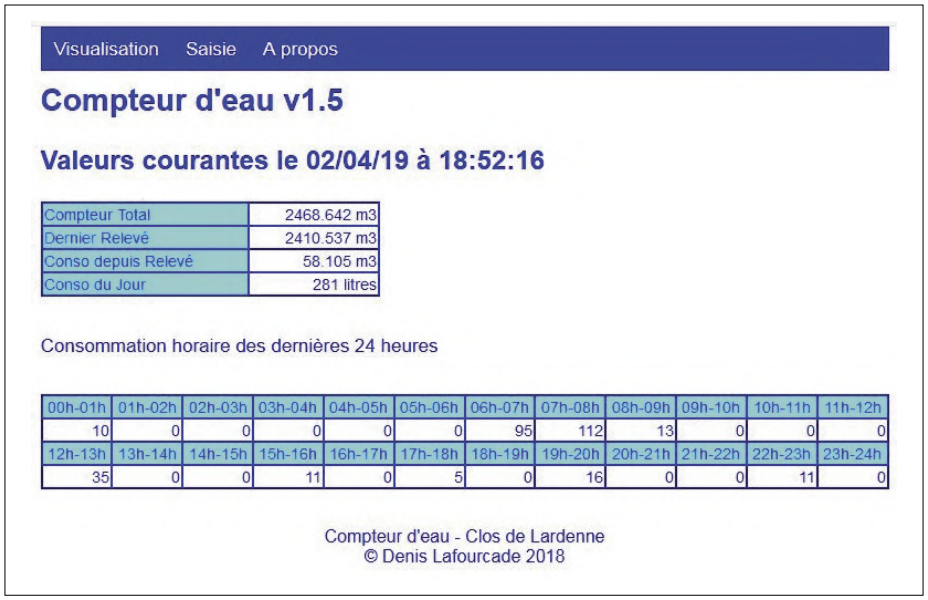


Figure 7. Serveur web.

- **Détection de fuite**

Toutes les heures, le logiciel examine la consommation horaire et établit un diagnostic de fuite sur la base des critères suivants :

- La consommation des dernières 24 heures est supérieure à un seuil ;
- Il n'y a pas de plage de deux heures consécutives sans consommation.

En l'absence de fuite, on doit trouver, notamment la nuit, au moins deux heures consécutives sans consommation. Si ce n'est pas le cas, il y a une forte probabilité de fuite.
En cas de présomption de fuite, le type de fuite est affiché sur la première page de l'écran OLED ainsi que sur la page Web, et un SMS est envoyé au numéro prévu grâce au service IFTTT.

- **IFTTT**

(*If This Then That* : si ceci, alors cela) est un service gratuit qui permet de combiner des actions déclenchées par des évènements, cette combinaison étant appelée « applet ». Ici l'évènement sera l'envoi par notre logiciel d'une requête « POST » spécifique sur le site *maker.ifttt.com*, celui-ci se chargeant alors d'émettre un SMS vers le numéro défini lors de la création de « l'applet ».
En cas de fuite, le logiciel va envoyer un SMS toutes les heures, mais comme IFTTT limite le nombre de SMS mensuels, le logiciel n'enverra plus de requêtes lorsque cette limite sera atteinte.

- **Mise à l'heure et sauvegardes**

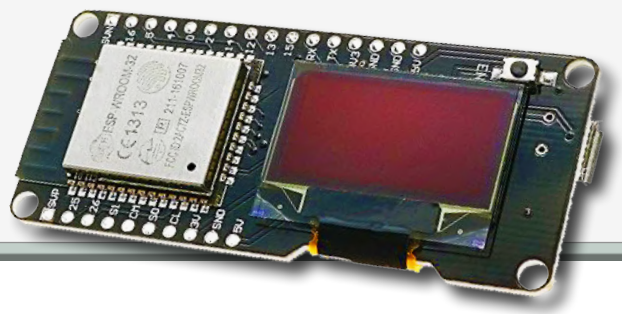
La mise à l'heure automatique est effectuée toutes les heures par NTP (*Network Time Protocol*) et l'heure est maintenue par l'horloge en temps réel (RTC) de l'ESP32. Le passage en heure d'été/hiver est aussi automatique.
Toutes les heures, le logiciel sauvegarde en mémoire morte (EEPROM), les valeurs courantes de consommation.

Raccourci pour accéder à la page web

Les utilisateurs peuvent enregistrer des raccourcis sur leur ordiphone pour accéder à cette page Web. Pour ceux dont le fournisseur d'accès propose un routeur avec « loop-back », un seul raccourci suffit, avec l'adresse IP externe de leur routeur suivie du numéro de port choisi. Pour les autres, il faut un second raccourci avec l'adresse IP du serveur sur leur réseau local (192.168.1.234:4321 par exemple). Certains opérateurs fournissent, par défaut ou sur demande, une adresse IP fixe. Dans ce cas, le raccourci extérieur est toujours valable. D'autres opérateurs changent cette adresse périodiquement, ce qui impose la mise à jour du raccourci ou le recours à un serveur dynamique de nom de domaine (DynDNS), dont certains sont gratuits. La façon de procéder sort du cadre de cet article, mais vous trouverez tout ce qu'il faut pour cela sur l'internet.

Assemblage

Un boîtier spécifique en impression 3D a été conçu pour accueillir le système (**figures 8** et **9**). Des ouvertures sont pratiquées pour l'écran OLED, pour le refroidissement du SoC et le dégagement de l'antenne Wi-Fi. Un motif découpé



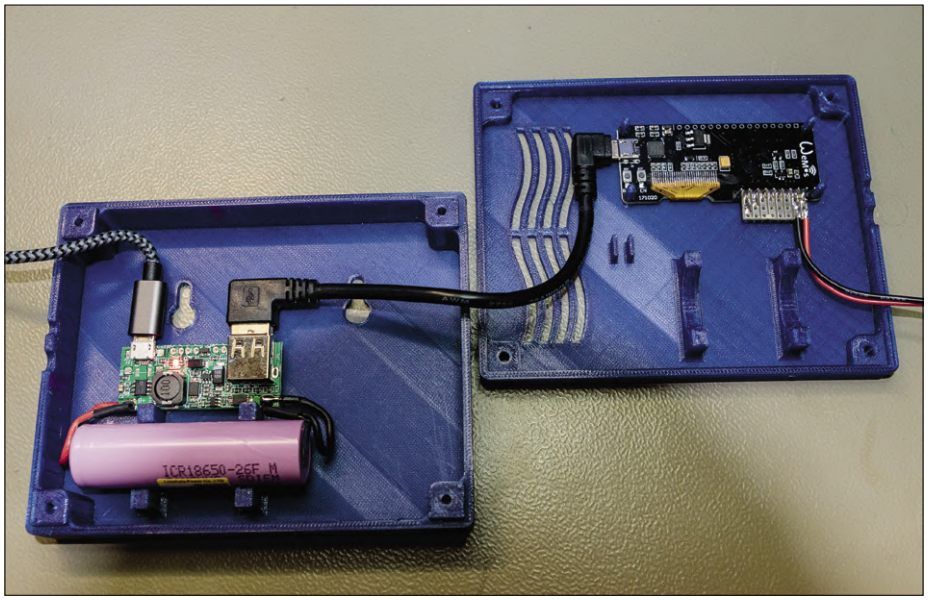


Figure 8. Photo du montage complet, installé dans son boîtier.

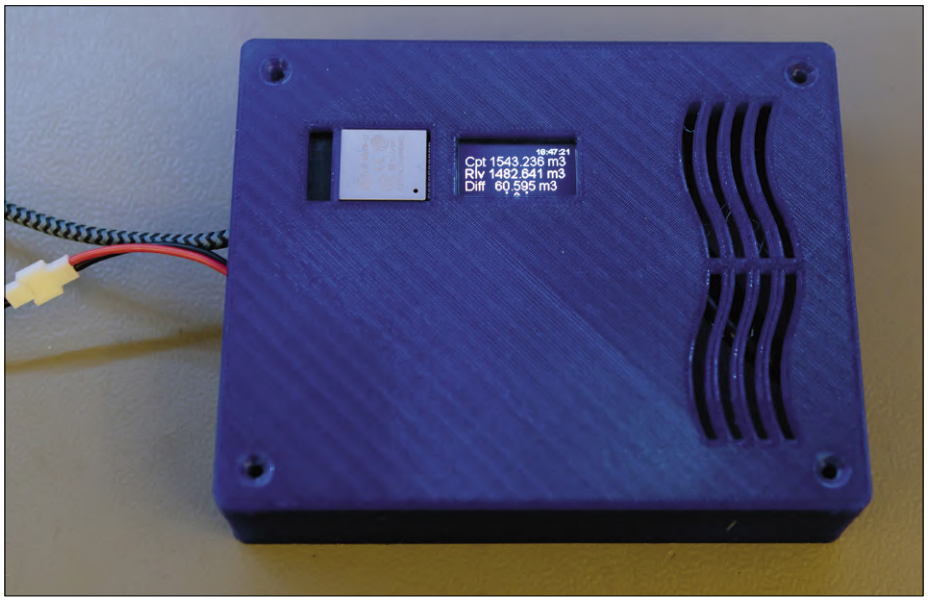


Figure 9. Le montage opérationnel dans son écrin imprimé en 3D.

en forme de vague, en plus de l'aspect esthétique, permet la ventilation du boîtier au cas où la charge de la batterie dégagerait un peu de chaleur. À l'intérieur, le boîtier comporte des supports de fixation pour le module Lolin et la batterie. Le chargeur est simplement collé avec une bande d'adhésif à double face. Il suffit de connecter le câble en provenance de l'émetteur d'impulsion, puis un cordon micro-USB avec un bloc chargeur standard. Un câble USB/micro-USB de 10 cm permet la connexion entre le module Lolin et le chargeur.

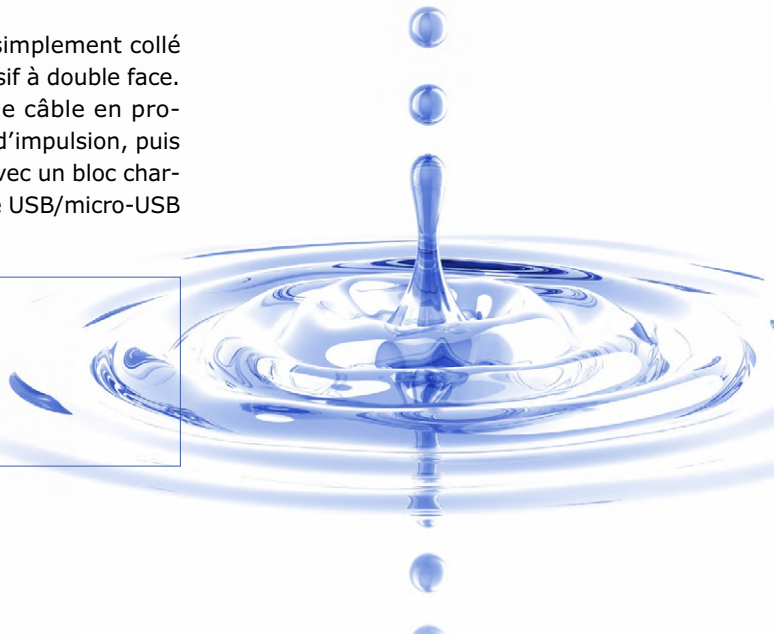
Mise en service/Utilisation

La programmation du système est effectuée avec l'EDI Arduino, en ayant préalablement configuré celui-ci pour les cartes à base d'ESP32 et choisi le type de carte Lolin32. Il faut également charger toutes les bibliothèques indiquées plus haut. Le premier téléversement se fait en branchant le module au PC, les suivants pouvant alors être réalisés par Wi-Fi grâce à la fonction OTA (*Over The Air*).
Le système démarre dès la mise sous tension, recherche le réseau Wi-Fi programmé, se met à l'heure et attend la connexion d'un utilisateur tout en surveillant les impulsions en provenance du compteur. Il faut alors accéder à la page web grâce à l'adresse figurant sur la troisième page de l'écran, et saisir la valeur actuelle du compteur, celle du dernier relevé et le seuil d'alerte de consommation journalière désiré.

Conclusion

Le relevé manuel de nos compteurs tous les six mois montre un écart avec ce système de quelques litres seulement, pour des consommations de l'ordre de 50 m³, ce qui est parfaitement suffisant. Quelques alertes ont été déclenchées et ont permis de détecter précocement des fuites qui auraient pu s'aggraver.
J'ai installé plusieurs de ces dispositifs chez mes voisins, et tous m'ont rapporté qu'ils en ont profité pour surveiller leur consommation, voire l'optimiser en constatant ce que consomme une douche, une lessive, un lave-vaisselle, un complément de plein de la piscine, etc. Voilà une autre fonction intéressante non prévue à l'origine ! N'hésitez pas à partager vos expériences !

(180694-01)



Liens

[1] Projet dans le labo d'Elektor : www.elektormagazine.fr/labs/waterflow-monitor

[2] Page de l'article : www.elektormagazine.fr/180694-01

microcontrôleur PIC1650

drôle de composant n°41

Neil Gruending

PIC1655A de General Instruments. Source : Camillo Ferrari ; licence *Creative Commons Attribution-Share Alike 3.0 Unported*.

Quel électronicien n'a jamais eu affaire à un microcontrôleur PIC de Microchip ? C'est le grand-père de cette famille appréciée de tous, le PIC1650, que je vous présente ici.

Au milieu des années 1970, General Instrument Microelectronics fabriquait le CP1600, un microprocesseur exploité dans de nombreuses consoles de jeu haut de gamme. Bien que formidable pour l'époque, le CP1600 utilisait le même bus d'adresses pour l'accès à la mémoire et aux périphériques, ce qui rendait sa gestion des E/S peu efficace. Pour y pallier, General Instrument conçut une puce périphérique programmable simplifiée, capable de décharger le CP1600 de toutes les tâches d'E/S intensives.

Cette nouvelle puce fut baptisée PIC1650, le sens de PIC restant pour certains sujet à débat : *Peripheral Interface Controller*, *Programmable Intelligent Computer* ou… rien du tout. Quoi qu'il en soit, sa conception était centrée sur les opérations de transfert de registre à registre au niveau du bit et de l'octet, et son jeu d'instructions simplifié, que l'on qualifierait de RISC aujourd'hui, donnait priorité aux fonctions de commande et d'interface sur les fonctions générales de calcul. Ces caractéristiques permirent d'utiliser le PIC1650 en lieu et place de circuits typiquement à logique discrète.

Le PIC1650 était un processeur à 8 bits et mot d'instruction de 12 bits. Un mot d'instruction plus grand simplifiait son architecture (**fig. 1**), car chaque instruction, aussi compliquée fût-elle, pouvait être lue depuis un seul emplacement mémoire, évitant ainsi les multiples accès à la mémoire requis par un mot d'instruction plus petit. Le codage des valeurs littérales sur une seule instruction plutôt que sur deux instructions *Load* et *Store* représentait aussi une amélioration.

Mais la caractéristique la plus notable du PIC1650 était sans doute l'architecture générale de ses registres : elle permettait en effet aux instructions d'accéder directement au compteur ordinal, à tous les registres d'E/S et aux registres spécialisés. Le jeu d'instructions en était simplifié puisqu'une instruction pouvait être utilisée de différentes façons. Pour un calcul logique p. ex., l'instruction de mise à 1 d'un bit pouvait être appliquée à un registre d'E/S ou général. Le jeu d'instructions permettait également de réaliser des opérations arithmétiques sur le compteur ordinal à l'aide d'instructions GOTO.

Vers 1985, General Instrument vendit sa division « microélectronique » à Microchip. La puce fut dès lors fabriquée en technologie CMOS, dotée d'une mémoire de programme EPROM, plus tard d'une Flash et, au fil des ans, de fonctions nouvelles. La lignée PIC compte aujourd'hui plus de 200 modèles. Le PIC1650

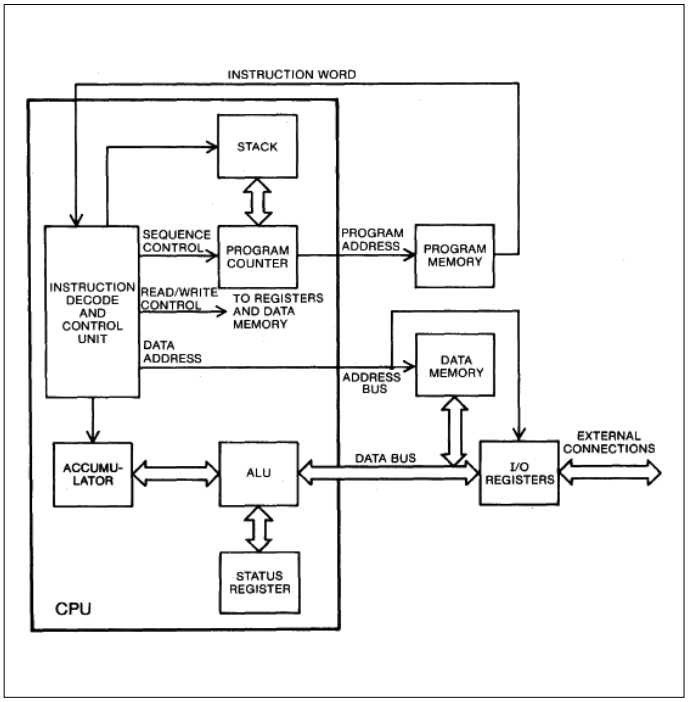


Figure 1. Architecture du PIC1650. Source : Bitsavers.org [1].

est à nos yeux modernes un processeur RISC parmi d'autres, mais à l'époque il s'agissait d'une pièce unique incluant certains éléments visionnaires. Il n'est plus vendu, mais sa fiche technique peut encore être étudiée [1] par curiosité ou… pour en faire une version sur FPGA ?

(180575-C-04 – version française : Hervé Moreau)

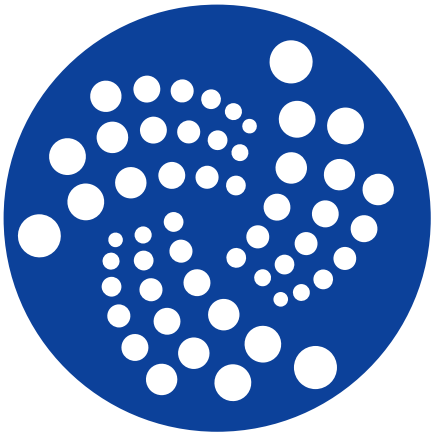
Lien

[1] Fiche technique du PIC1650 : www.bitsavers.org/components/gi/PIC/1983_PIC_Series_Microcomputer_Data_Manual.pdf

la cryptomonnaie IOTA (et FPGA pour RPi)

2e partie : PiDiver – et une carte FPGA pour le calcul rapide

Thomas Pototschnig et **Julia Orlitzky**



Dans le numéro précédent, nous avons présenté la cryptomonnaie IOTA avec laquelle les nœuds de l'Internet des Objets (IdO) peuvent rémunérer d'autres utilisateurs – par exemple pour le stockage de données [1]. Pour pouvoir effectuer des transactions IOTA, il faut au préalable avoir résolu un problème de calcul appelé Preuve de Travail (*Proof of Work*, PoW), opération qui sert à protéger la base de données des transactions (le tangle) du spam. Cette preuve de travail donnant du fil à retordre même aux machines x86, IOTA a prévu la possibilité pour les petits nœuds de la déléguer à des périphériques spécialisés. Dans cette deuxième partie, nous présentons une telle plateforme basée sur une carte d'extension pour le Raspberry Pi.

Cet article décrit une solution rapide et peu gourmande en énergie pour résoudre le problème de la PoW ; les calculs nécessaires sont effectués par une puce FPGA. On obtient ainsi un temps de calcul moyen de la PoW de 300 ms, ce qui est plus rapide qu'une bibliothèque multifil optimisée SSE sur un Core i5 (env. 1,7 s). De plus, la puissance consommée de 2 W est nettement plus faible.

La preuve de travail de IOTA

Pour qu'une transaction soit acceptée par le tangle, les quatorze derniers trits [1] du haché de la transaction doivent être à zéro. Si le haché ne remplit pas cette condition, un compteur interne à la transaction (*nonce*, voir **fig. 1**) est incrémenté et le haché recalculé. Cette opération est répétée jusqu'à ce qu'un haché valide soit trouvé. Cela prend en moyenne 314/hps secondes, où hps est le nombre de hachés calculés par seconde, ce qui correspond à une moyenne de 4,78 millions d'itérations jusqu'à la découverte d'une solution au problème. L'idée sous-tendue est qu'on peut protéger ainsi le tangle contre le spam par le fait que la fréquence de transactions maximale est fortement réduite par la puissance de calcul consommée. Par contre, une vérification ne prend guère de temps, car il suffit de vérifier que le haché de la transaction est correct.

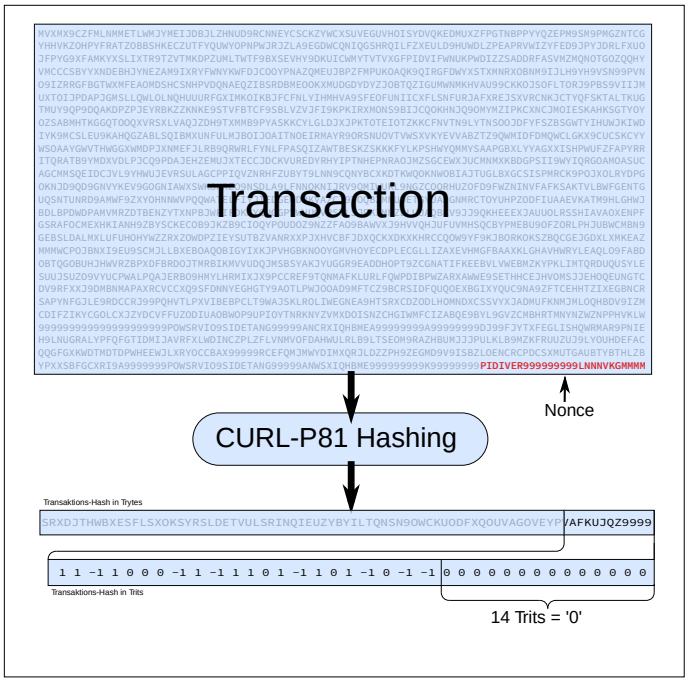


Figure 1. Hachage d'une transaction.

C'est pourquoi un haché de transaction est considéré comme une PoW, car il prouve que les calculs ont bien été effectués. L'algorithme de hachage utilisé est le CURL-P81, qui, à l'instar du SHA-3, est conçu comme une fonction éponge (c'est-à-dire comportant deux phases : l'absorption et l'essorage). De nouveaux blocs de taille constante (*rate*) sont absorbés (*absorb*) dans l'état interne qui a une taille (*capacity*) triple d'un bloc. Ensuite la fonction éponge convertit l'état complet dans l'état suivant (**fig. 2**). Quand tous les blocs ont été absorbés, on passe à la phase d'essorage (*squeeze*), où est produit le haché.

Une transaction IOTA se compose de 33 blocs de 81 trytes (B1 à B33 sur le diagramme). Le dernier bloc contient le compteur qui est incrémenté si le haché de la transaction ne remplit pas les critères du réseau principal (il existe d'autres réseaux de type tangle pour le développement et les tests).

Comme le compteur se trouve dans le dernier des 33 blocs, il est possible d'utiliser l'état intermédiaire (*mid-state*) obtenu après absorption des 32 premiers blocs comme état initial pour le 33e bloc. Après l'incrémentation du compteur dans ce bloc, il suffit alors de hacher un seul bloc pour obtenir le haché de la transaction complète, ce qui fait gagner beaucoup de temps.

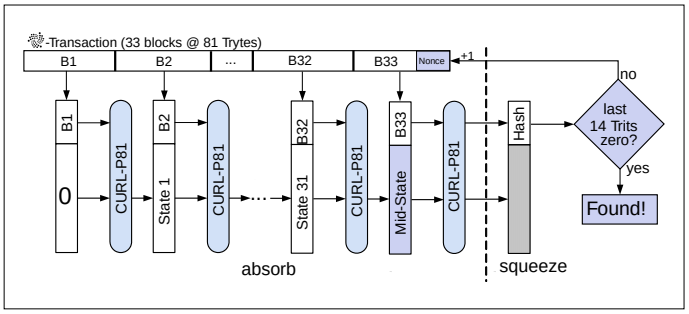


Figure 2. Structure de type éponge du calcul du haché (source : Wikipedia).

Vue d'ensemble du PiDiver

Le PiDiver (contraction de Raspberry Pi et du nom de l'algorithme de référence de la PoW, « Pearl-Diver » [1]) est réalisé sous la forme d'une carte d'extension HAT (*Hardware Attached on Top*) pour le RPi, capable d'effectuer la PoW de IOTA de manière rapide et peu énergivore.

La **figure 3** montre le schéma bloc du PiDiver. Le centre en est occupé par une puce FPGA Cyclone 10 LP d'Intel. Il s'agit là d'un membre de la famille Cyclone, relativement récente (apparue en 2018), qui vise le marché des applications IdO bon marché et basse consommation. Les FPGA, qui offrent beaucoup de ressources en circuits logiques, sont non seulement bon marché et peu énergivores, mais également disponibles dans des boîtiers (relativement) sympathiques pour les bricoleurs (EQFP), ce qui en fait le choix idéal pour ce projet.

La puce FPGA est cadencée à 200 MHz et capable d'exécuter une passe de hachage en un seul cycle (CURL-P81 en nécessite 81). De plus, elle calcule simultanément 7 hachés avec des compteurs (*nonces*) différents. Aux 81 cycles, il faut en ajouter deux pour vérifier le résultat, restaurer l'état intermédiaire (*mid-state* sur la figure 2) et mettre à jour les compteurs. On arrive ainsi à 16,8 MHash/s environ, ce qui, dans le cas du réseau principal IOTA, et transferts de données inclus, donne un temps moyen de PoW de 300 ms, soit 3,33 PoW/s.

Le circuit comprend aussi un microcontrôleur STM32F302 qui permet de piloter la puce FPGA par USB. On a donc le choix de passer soit par le port SPI du RPi, soit par l'USB d'un PC (s'il n'y a pas de RPi).

Vous aurez peut-être remarqué qu'il n'y a pas de mémoire flash pour la configuration de la puce FPGA. Cela signifie qu'il faut recharger la configuration dans la puce FPGA à chaque mise sous tension. C'est possible soit par le RPi, soit à travers l'USB. La configuration est automatiquement reconnue et exécutée par la bibliothèque PiDiver. Pour les futures extensions, il est prévu une mémoire flash SPI attachée au STM32, de taille suffisante pour stocker le flux de bits FPGA complet. Cela permettrait de procéder à une configuration automatique à la

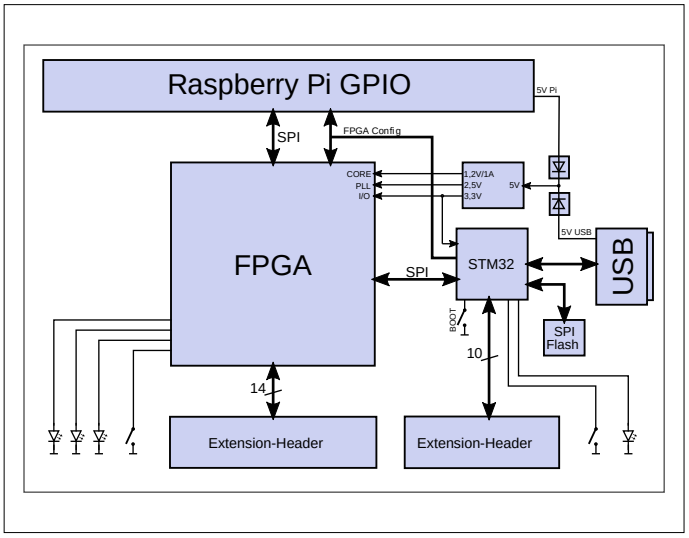


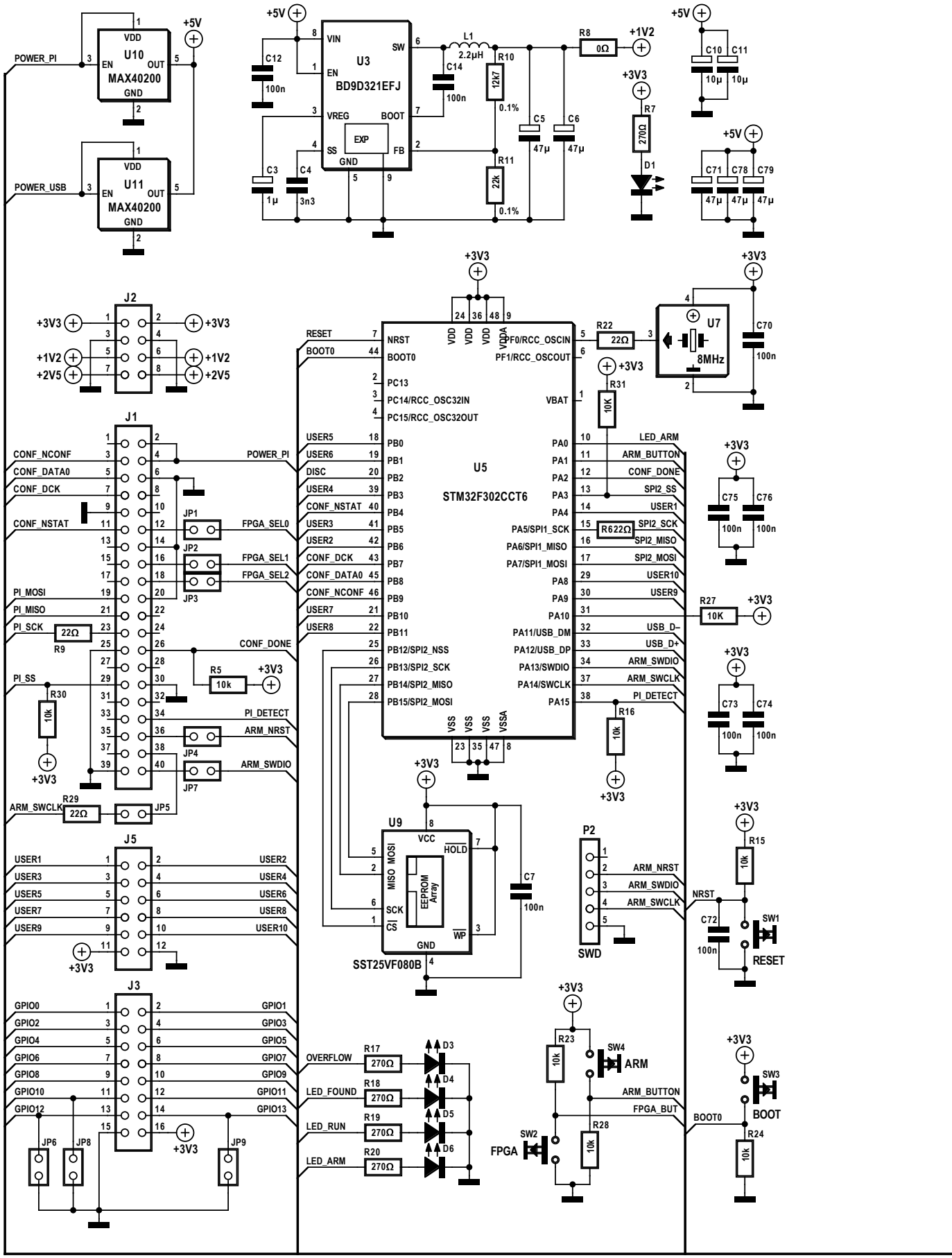
Figure 3. Schéma bloc du PiDiver avec la puce PFGA au centre.

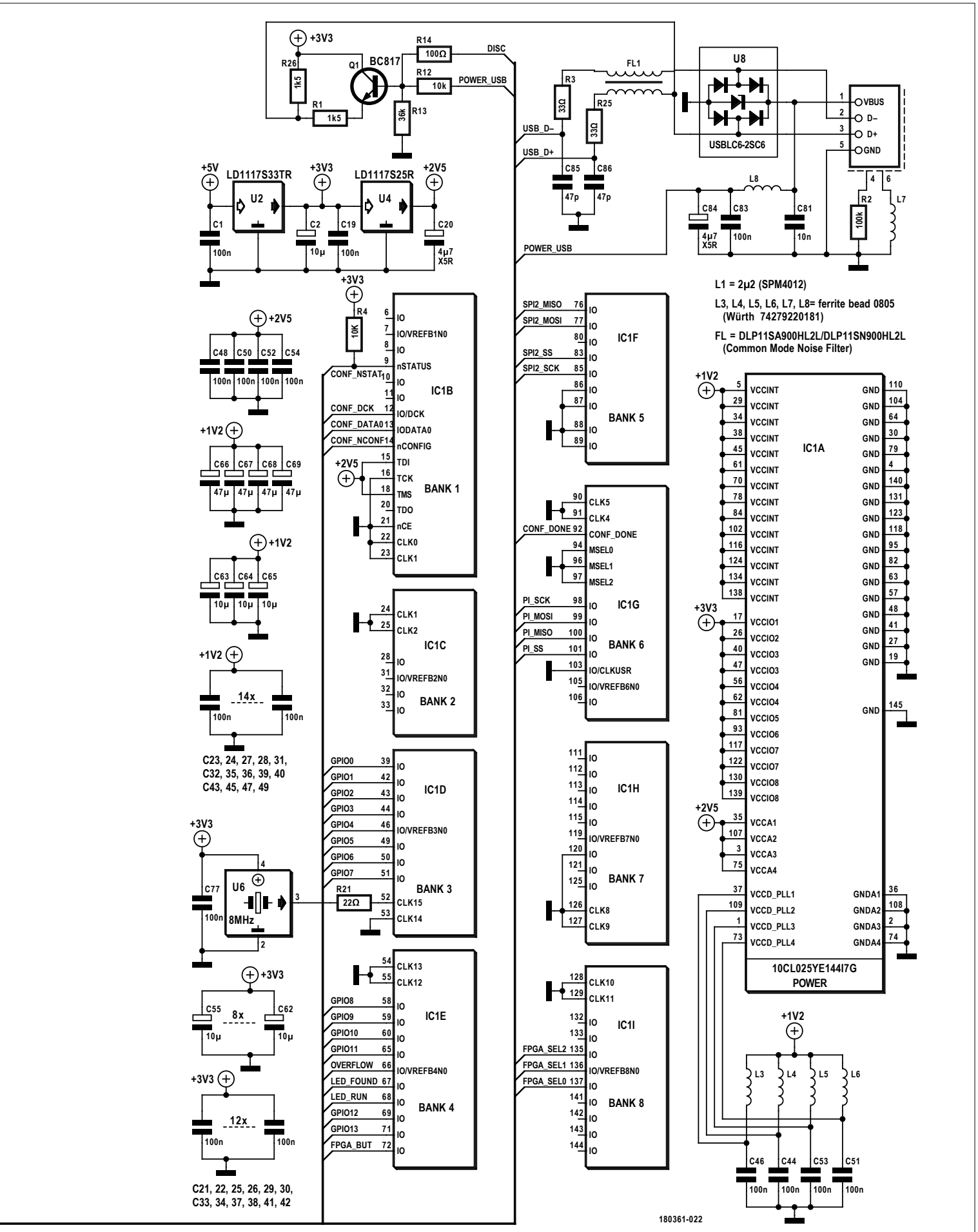
mise sous tension, opération sans doute intéressante dans le cas d'applications embarquées (par ex. avec un microcontrôleur ESP32), étant donné que le flux de bits, qui représente 800 Ko environ, excède la taille de la mémoire flash de beaucoup de microcontrôleurs.

Le schéma de la **figure 4** n'apporte pas grand-chose de plus. Il y a trois tensions d'alimentation avec chacune son propre régulateur. Deux d'entre eux, de type fixe à faible chute de tension (LDO), sont branchés en cascade et fournissent les 3,3 V et 2,5 V ; le troisième est un régulateur à découpage réglé sur 1,2 V et pouvant débiter jusqu'à 3 A (on a besoin de 1 A). La carte peut être alimentée en 5 V soit par le connecteur GPIO du RPi, soit par l'USB. Deux diodes MAX40200, qualifiées d'« idéales » par le fabricant, découplent les deux sources de 5 V, de sorte qu'elles ne soient pas directement interconnectées. En réalité, la MAX40200 n'est pas une diode, mais un MOSFET logique d'une chute de tension de 85 mV pour un courant de 1 A, de quoi faire pâlir de jalousie n'importe quelle diode Schottky !

Le circuit possède quelques possibilités d'extensions pour d'autres projets. Deux connecteurs d'extension donnant accès

Figure 4. Schéma du PiDiVER.




+3V3
R14
100Ω
DISC
R26
1k5
Q1
BC817
R12
10k
POWER_USB
R1
1k5
36k
R13
+5V
LD1117S33TR
U2
+3V3
LD1117S25R
U4
+2V5
C1
100n
C2
10µ
C19
100n
C20
4µ7
X5R
+3V3
R4
10K
CONF_NSTAT
IC1B
BANK 1
CONF_DCK
CONF_DATA0
CONF_NCONF
+2V5
C48 C50 C52 C54
100n 100n 100n 100n
+1V2
C66 C67 C68 C69
47µ 47µ 47µ 47µ
+1V2
C63 C64 C65
10µ 10µ 10µ
+1V2
14x
100n 100n
C23, 24, 27, 28, 31,
C32, 35, 36, 39, 40
C43, 45, 47, 49
IC1C
BANK 2
IC1D
BANK 3
GPIO0
GPIO1
GPIO2
GPIO3
GPIO4
GPIO5
GPIO6
GPIO7
R21
22Ω
CLK15
CLK14
+3V3
C77
100n
U6
8MHz
+3V3
C55
8x
C62
10µ
10µ
+3V3
12x
100n 100n
C21, 22, 25, 26, 29, 30,
C33, 34, 37, 38, 41, 42
IC1E
BANK 4
CLK13
CLK12
GPIO8
GPIO9
GPIO10
GPIO11
OVERFLOW
LED_FOUND
LED_RUN
GPIO12
GPIO13
FPGA_BUT
USB_D–
USB_D+
R3
33Ω
R25
33Ω
FL1
U8
USBLC6-2SC6
VBUS
D–
D+
GND
C85
47p
C86
47p
L8
C84
4µ7
X5R
C83
100n
C81
10n
R2
100k
L7
POWER_USB
L1 = 2µ2 (SPM4012)
L3, L4, L5, L6, L7, L8= ferrite bead 0805
(Würth 74279220181)
FL = DLP11SA900HL2L/DLP11SN900HL2L
(Common Mode Noise Filter)
SPI2_MISO
SPI2_MOSI
SPI2_SS
SPI2_SCK
IC1F
BANK 5
CONF_DONE
CLK5
CLK4
MSEL0
MSEL1
MSEL2
PI_SCK
PI_MOSI
PI_MISO
PI_SS
IO/CLKUSR
IO/VREFB6N0
IC1G
BANK 6
IC1H
BANK 7
CLK8
CLK9
IO/VREFB7N0
CLK10
CLK11
FPGA_SEL2
FPGA_SEL1
FPGA_SEL0
IO/VREFB8N0
IC1I
BANK 8
+1V2
IC1A
VCCINT
GND
+3V3
VCCIO1
VCCIO2
VCCIO3
VCCIO4
VCCIO5
VCCIO6
VCCIO7
VCCIO8
+2V5
VCCA1
VCCA2
VCCA3
VCCA4
VCCD_PLL1
VCCD_PLL2
VCCD_PLL3
VCCD_PLL4
GNDA1
GNDA2
GNDA3
GNDA4
10CL025YE144I7G
POWER
+1V2
L3 L4 L5 L6
C46 C44 C53 C51
100n 100n 100n 100n
180361-022

à 14 entrées/sorties de la puce FPGA et à 10 broches de port du µC STM32 sont disponibles pour des applications personnelles. Il y a des LED et des boutons-poussoirs qui sont associés à la puce FPGA ainsi qu'au STM32. Le schéma a été conçu de sorte que le chargeur d'amorçage interne DFU-Boot-Loader du STM32 soit opérationnel. Pour l'utiliser, il faut appuyer à la mise sous tension sur un bouton de BOOT séparé, pour que le contrôleur démarre dans le mode chargeur d'amorçage. La consommation de courant, de l'ordre de 350 mA, est suffisamment faible pour une alimentation sans problème par un port USB standard.

Réalisation

Le circuit imprimé du PiDiver [2] possède quatre couches et est presque exclusivement équipé de CMS. Certains d'entre eux (U6, U7 et FL1) ne sont pas accessibles au fer à souder, car leurs pastilles sont dissimulées sous le boîtier. Leur soudage requiert donc de la pâte à souder et de l'air chaud.

Il faut laisser de côté R8 (une résistance de 0 Ω à la sortie du régulateur U4), car elle n'est pas à implanter dans l'immédiat. Les composants barrés sur le plan d'implantation (voir [2]) ne sont pas nécessaires au fonctionnement de l'accélérateur de PoW. Les composants R12, R13, R14, Q1 et P2 n'ont d'intérêt

Développement du PiDiver

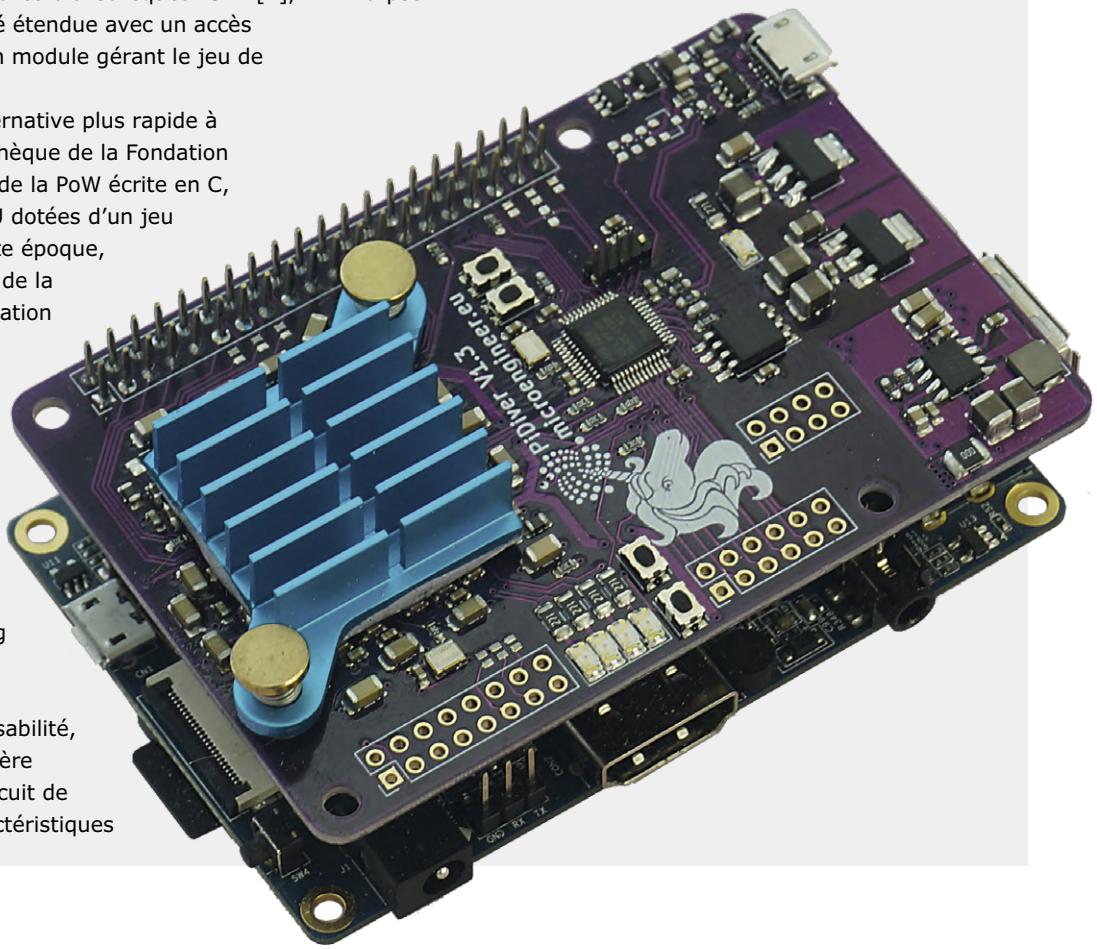
La preuve de travail de IOTA doit protéger le tangle du spam, mais représente en même temps un obstacle pour des applications valides qui doivent envoyer des données au tangle rapidement. C'est une charge de travail considérable, même pour des ordinateurs puissants, et *a fortiori* pour les petits systèmes de l'Internet des Objets. Par exemple, la résolution d'un problème de PoW prend en moyenne 90 s à un Raspberry Pi.

C'est la raison pour laquelle a été lancé le projet PiDiver. Cela a commencé par le développement d'une preuve de faisabilité sur une carte de développement âgée de 10 ans (Altera DE1). On a utilisé l'environnement de développement Altera Quartus 13.0.1, les FPGA Cyclone II n'étaient plus prises en charge par les versions plus récentes. Le cœur du PiDiver a été écrit en VHDL, en partant du code de référence « PearlDiver » (écrit en Java) de Sergey Ivancheglo [12]. Au début, on a atteint une performance de hachage de 4,6 millions de hachés par seconde, rapidement portée à 12,9 MHash/s. Pour pouvoir utiliser la puce FPGA avec les bibliothèques IOTA [7], la bibliothèque « dcurl » [8] a été étendue avec un accès matériel par SPI pour le RPi et un module gérant le jeu de commandes de la puce FPGA.

La bibliothèque *dcurl* est une alternative plus rapide à la bibliothèque *PoW ccurl* (bibliothèque de la Fondation IOTA [9]), car outre une version de la PoW écrite en C, elle supporte des GPU et des CPU dotées d'un jeu d'instructions SSE ou AVX. À cette époque, le porte-monnaie (*wallet*) officiel de la Fondation IOTA [10] – une application Electron – utilisait *ccurl* comme bibliothèque externe, car ce porte-monnaie était lui-même écrit en JavaScript, ce qui est relativement lent pour une PoW. Cette bibliothèque fut remplacée par *dcurl*. Plus tard ont été développés une bibliothèque dédiée à PiDiver et un programme de test en GoLang qui rendirent *dcurl* inutile.

En parallèle avec les premières optimisations de la preuve de faisabilité, on a dessiné avec KiCad la première version du circuit imprimé. Le circuit de base s'inspire de feuilles de caractéristiques d'Intel, pour son dessin, on a suivi des recommandations de conception. La première version de la carte a été testée avec succès en mai 2018, avec une performance de hachage de 14,8 MHash/s. Il y a eu en tout quatre versions du circuit imprimé. Les deux premières versions étaient équipées d'un µC STM32F1, remplacé par un STM32F3 dès la troisième version pour profiter de la fonction de chargeur d'amorçage. La deuxième version a vu la mise en œuvre d'une alimentation optimisée, les régulateurs de la première version étant surdimensionnés. La version 3 dispose d'une protection contre les décharges électrostatiques et d'un filtre CEM sur l'USB ; les deux MAX40200 (« diodes idéales ») ainsi que la fonction de chargeur d'amorçage DFU y ont été ajoutées. La version 4 comprend des perçages pour le montage d'un radiateur ainsi que des boutons-poussoirs plus petits (donc plus nombreux) et des connecteurs à broches pour des extensions personnelles.

Le microgiciel du STM32 a été développé en C++ sous Eclipse.



que si l'on désire utiliser la carte pour développer du logiciel pour le µC STM32. Ils permettent de gérer par logiciel une résistance de rappel sur le fil de données D+ de l'USB. C'est en particulier intéressant pour le débogage du logiciel du STM32, afin de notifier l'hôte d'une réinitialisation du µC et provoquer ainsi une nouvelle énumération du périphérique USB.
Le chargeur d'amorçage standard intégré en usine dans chaque STM32F302 ne gère malheureusement aucune résistance de rappel commutable, c'est pourquoi une résistance spécifique (R26) est prévue pour prendre cette fonction en charge. Cette fonction est complétée par le connecteur à broches P2 d'accès à l'interface de débogage SWD.
La plupart des autres composants n'appellent aucun commentaire particulier (sauf que beaucoup sont très petits), mais il faut prendre garde aux surfaces métalliques sous la puce FPGA (U1) et le régulateur (U3) qui doivent aussi être soudées. Pour l'implantation manuelle, nous avons donc prévu des trous (*via*) à travers lesquels ce soudage est rendu possible. Il est recommandé de chauffer ces surfaces métalliques avec de l'air chaud à travers ces trous jusqu'à la fonte de la soudure. Il est fortement déconseillé d'utiliser un fer à souder, la chaleur se diffuse alors rapidement dans les couches intermédiaires, ce qui rend difficile d'amener les surfaces métalliques à la bonne température. Dans le pire des cas, on risque aussi d'endommager la carte.

Première mise en service

Quand la carte est prête, on commence par vérifier les tensions. Il faut tester en premier la tension de 1,2 V à la sortie du régulateur, laquelle servira à alimenter le cœur de la puce FPGA. On a vite fait d'intervertir les résistances du diviseur de la tension de contre-réaction et de détruire ainsi la puce FPGA. Si la tension est correcte, on soude R8.
On peut ensuite vérifier toutes les tensions sur les pastilles du connecteur à broches J2, qui n'est pas encore installé (**fig. 6**). La tension de 1,2 V n'y est toutefois disponible qu'après l'installation de R8. Quand la carte est alimentée, LED 1 doit être allumée pour signaler la présence du 3,3 V.
Ensuite, on peut vérifier que le µC STM32 est bien reconnu sur l'USB. Pour cela, on connecte la carte au PC au moyen d'un câble USB tout en appuyant sur le bouton BOOT. Le microcontrôleur s'identifie alors au PC en tant que « DFU Bootloader ».

Logiciel

Flashage du µC STM32

Pour graver le µC, il faut le démarrer en mode DFU Bootloader. Pour cela, on maintient le bouton BOOT enfoncé pendant qu'on connecte la carte à l'USB. Peu après, le contrôleur s'identifie au PC sous le nom « DFU Bootloader ».
Sous Linux, on peut utiliser l'outil `dfu-util` [3] pour téléverser le microgiciel :

```
dfu-util -v -d 0483:df11 -a 0 -s 0x08000000 -D
   usbdiver.bin
```

Après un téléversement réussi et une réinitialisation, le µC doit s'identifier comme USBDiver.

Installation de GoLang

Les bibliothèques sont écrites dans le langage GoLang, qu'il faut commencer par télécharger et installer. GoLang est disponible sous [4] pour plusieurs systèmes d'exploitation et architectures. La page web fournit aussi un mode d'emploi. Après l'installation et la configuration des variables d'environnement, on redémarre l'ordinateur.

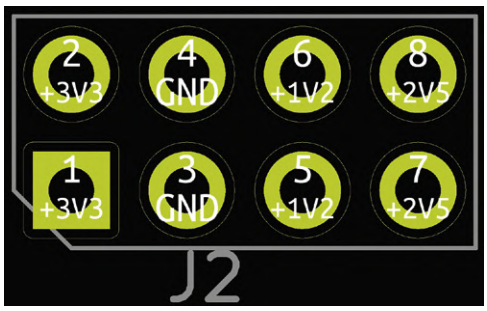


Figure 5. Affectation des bornes du connecteur J2 (non monté) pour tester les tensions d'alimentation.

Installation du programme de test

Après l'installation réussie de GoLang, on peut compiler le programme de test. On commence par rapatrier (sur le RPi ou le PC sousLinux) le paquet Git :

```
go get gitlab.com/microengineer18/pidiver1.3/golang/
   pidiver
cd $GOPATH/src/gitlab.com/microengineer18/pidiver1.3/
   golang/pidiver/main
go get -d ./...
go build
```

Le test proprement dit peut alors commencer. Si l'on veut utiliser le PiDiver sur le RPi, on y démarre le programme avec la commande :

```
$ sudo ./main -t pidiver
```

(Note : `sudo` est important, car sinon on n'a pas d'accès direct au niveau matériel de l'interface SPI.)
Si l'on veut utiliser le PiDiver à travers l'USB, il faut saisir la commande suivante sur le PC sous Linux :

```
$ ./main -t usbdiver
```

La sortie a le même aspect dans les deux cas :

```
...
2018/07/14 09:09:02 Found nonce: 0005e423 (mask:
   00000010)
2018/07/14 09:09:02 PoW-Time: 172ms (15.62MH/s)
2018/07/14 09:09:02 Nonce-Trytes:
   PIDIVER9V9999999OMKVVNGMMMM
2018/07/14 09:09:02 hash: RIRQYZKVJIDQIO9RBPLCJTSRTJ
   CWFVOODAMKXFHRJMFZQLR9AJPLCNYEPW9IJCEJWRBTVTRIWZY
   DA9999

2018/07/14 09:09:03 Found nonce: 00094b5e (mask:
   00000020)
```

```
2018/07/14 09:09:03 PoW-Time: 265ms (16.03MH/s)
2018/07/14 09:09:03 Nonce-Trytes:
   PIDIVER9T9999999UNGEKTVMMMM
2018/07/14 09:09:03 hash: EIW9MEACXYGVAQFCUGAKGMXKNC
   PGMMSPQWOLGMHZWHZKUEOLNRFKJGMNSELCSLRAEVCLYCBREBW
   TZ9999
...
```

Le programme de test calcule les *nonces* de transactions aléatoires et affiche leurs hachés, qui, dans le cas du réseau principal IOTA se termine toujours par « 9999 » (trois trits successifs à 0 affichés comme tryte produisent le caractère 9).
Il existe aussi un serveur (qui tourne sur le RPi, mais aussi sur un PC), qui accepte la commande `attachToTangle` [5] de l'API IOTA et calcule des *nonces* pour de vraies transactions. Mais en dire plus sortirait du cadre de cet article; on se reportera donc à la documentation [6].

Alimentation du RPi

Pour un bon fonctionnement, il est important d'utiliser des câbles qui supportent l'intensité de courant nécessaire. Pour l'alimentation du RPi, il faut prendre un câble USB du genre de ceux utilisés pour les chargeurs à batterie. Le bloc d'alimentation doit pouvoir fournir de vrais 2 A sous de vrais 5 V ! De même, l'alimentation par l'USB exige des câbles calibrés pour au moins 500 mA. L'usage de câbles inadaptés ou d'un bloc d'alimentation un peu faible des genoux se traduit habituellement par une perte de la configuration de la puce FPGA dès le début du calcul de la PoW. ⏮

(180361-B-04 – version française : Helmut Müller)

Liens

[1] Première partie de la série : www.elektormagazine.fr/180361-05

[2] Dépôt GIT du projet : www.gitlab.com/microengineer18/pidiver1.3

[3] Outil DFU : http://dfu-util.sourceforge.net/

[4] GoLang : https://golang.org/

[5] Attach to Tangle: https://iota.readme.io/reference#attachtotangle

[6] Documentation du PiDiver : https://ecosystem.iota.org/tutorials/pidiver-usbdiver-documentation

[7] IOTA Ledger: https://github.com/iotaledger

[8] Blibliothèque dcurl : https://github.com/DLTcollab/dcurl

[9] Bibliothèque ccurl : https://github.com/iotaledger/ccurl

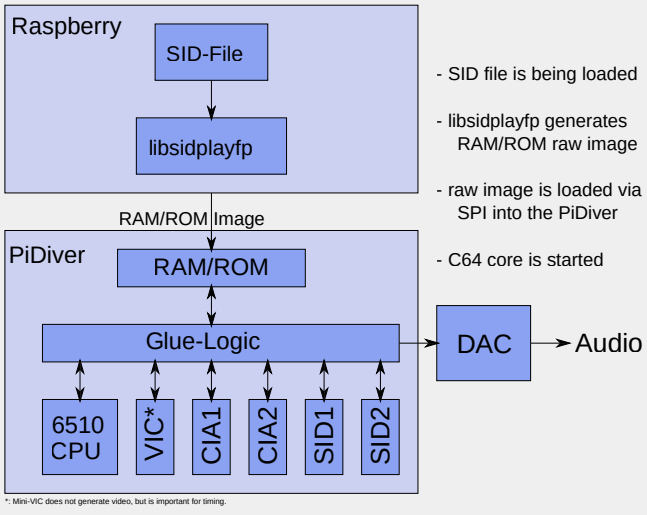
[10] Porte-monnaie de la Fondation IOTA : https://github.com/iotaledger/wallet

[11] PiDiver joue de la musique : www.youtube.com/watch?v=GhgDCf9oBEo

[12] PearlDiver: https://github.com/Come-from-Beyond/PearlDiver

PiDiver musical

Le PiDiver est en fait une carte FPGA à usages multiples, qui met à disposition des connecteurs supplémentaires dont les broches sont utilisables pour des applications personnelles. On a par ex. testé une émulation matérielle d'un C64, capable de jouer de la musique des années 80 en stéréo grâce à un convertisseur numérique-analogique à 8 bits connecté à la carte par I2S.
On a utilisé pour cela la bibliothèque *libsidplayfp*, qui contient une émulation logicielle du C64 avec laquelle de la musique SID peut être jouée par un PC. Cette bibliothèque crée en interne des images de RAM et de ROM qui peuvent être exécutées par le C64 émulé. Quoique ces images n'aient pas été prévues pour une exécution sur du matériel C64, cela fonctionne parfaitement. Félicitations aux développeurs de cette bibliothèque ! On peut voir un clip vidéo avec le PiDiver jouant de la musique sous [11].



bruits de labo

Clemens Valens (labo d'Elektor)

Même si les nouvelles technologies continuent de supplanter les anciennes à une vitesse fulgurante, cela ne signifie pas que les anciennes méthodes disparaissent. Parfois, elles refont même surface, souvent par nostalgie. Le domaine de la photographie en est un exemple.

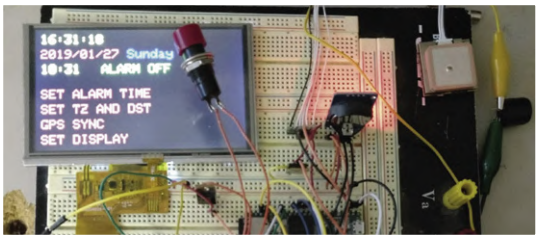
Enceinte Bluetooth portable avec effets de lumière

Les enceintes et les systèmes de sonorisation Bluetooth sont aujourd'hui très populaires. Bien que très pratiques, ces dispositifs n'offrent pas toujours la qualité de son attendue. Dans ce cas, pourquoi ne fabriquez-vous pas l'appareil de vos rêves ? L'électronique nécessaire vient de Chine et à petit prix, mais la partie acoustique ne dépend que de vous. Et tant que vous y êtes, pourquoi ne pas incorporer des effets de lumière ? Découvrez comment avec un excellent tutoriel...

@ Elektor Labs: www.elektormagazine.com/labs/1795

Réveil GPS avec bibliothèque pour écran tactile et fonctions graphiques

Des réveils, encore des réveils, toujours des réveils... Tout le monde semble construire des réveils aujourd'hui. Ils ont pourtant un intérêt : la plupart d'entre eux ont quelque chose de spécifique qui incite à aller y voir de plus près. Dans le cas de ce réveil GPS réalisé sur une carte Teensy 3.5, il s'agit de l'écran tactile et de la bibliothèque graphique qui l'accompagne. Votre projet pourra sans doute s'en inspirer, réveil ou pas...



@ Elektor Labs: www.elektormagazine.com/labs/1792

Dé électronique à afficheur à 7 segments

Êtes-vous lassé(e) de compter les points sur les faces d'un dé ? Dans ce cas, le circuit proposé ici va vous intéresser. Son affichage à 7 segments permet de visualiser la valeur sans aucune ambiguïté. Facile à construire, ce circuit s'appuie sur un microcontrôleur ATtiny2313 et se programme en Bascom. Pour lancer le dé, il suffit d'appuyer sur le bouton-poussoir, puis de le relâcher. Apparaît alors un chiffre aléatoire compris entre 1 et 6.

@ Elektor Labs: www.elektormagazine.com/labs/137

Éclairage inactinique pour chambre noire « maison »

À une époque où prolifèrent des caméras numériques affichant des quantités faramineuses de pixels et de couleurs, associées à des logiciels de traitement d'image extraordinairement puissants, difficile d'imaginer qu'il existe encore des passionnés prêts à passer des heures dans une chambre noire à plonger les doigts dans des produits chimiques plus ou moins nocifs pour développer des photos « à l'ancienne ». Et pourtant, ces enthousiastes existent, comme le prouve ce projet issu du vivier du labo d'Elektor... !

180571-D-04

@ Elektor Labs: www.elektormagazine.com/labs/1797

interface AS : les bases

fonctionnement du bus pour l'automatisation industrielle

Tam Hanna

Au début, tout était commandé à l'aide de câblage fait maison et de formats de données propriétaires. Cette approche est pourtant depuis longtemps révolue en automatisation industrielle. Désormais on utilise des cartes dernier cri et des systèmes de bus spécialisés. Outre l'avantage évident d'être facile à maintenir et à développer, il existe une autre raison à cela : une norme industrielle largement répandue garantit la disponibilité à long terme du matériel et des logiciels, du moins dans une certaine mesure. Vous trouverez ci-après un aperçu de l'interface AS.

Lorsque des processus de fabrication complexes doivent être automatisés, on se tourne vers la structure d'automatisation présentée à la **figure 1**. Au plus bas niveau se trouvent les capteurs et les actionneurs. Pour des raisons de coût et d'efficacité, on renonce à la communication avec ceux-ci à l'aide d'un câblage Ethernet. Par ailleurs les liaisons à courte distance avec des protocoles tels que SPI ou I2C sont incompatibles avec l'environnement souvent sévère des usines de production [1]. Deux techniques largement répandues sont disponibles pour la connexion de composants tels que des capteurs : le protocole ASi (interface actionneur-capteur), connu sous le nom d'interface AS, et le protocole IO-Link (normalisé sous le nom SDCI), piloté par l'environnement Profibus. Le principal avantage de l'interface AS est qu'un maître peut s'adresser à un grand nombre d'esclaves via un seul port. Avec IO-Link, vous avez normalement besoin d'un port et d'un câble par esclave. IO-Link est considéré dans la littérature anglaise comme plus approprié en cas de systèmes à gros volumes de données.
Notons que ASi et IO-Link apparaissent souvent « ensemble ». Le système Bihl-Wiedemann illustré à la **figure 2** et décrit en détail dans [2] utilise IO-Link pour collecter les informations. Les datagrammes compressés sont ensuite transmis au maître via ASi.

Une question de version

L'interface AS a vu le jour en 1990 à la suite de la fusion d'un groupe de sociétés qui ont créé une sorte de pôle d'intérêt sous le nom d'AS International Association [3]. Celui-ci détient les droits de la « marque » AS Interface et vend des produits selon la norme officielle. La cotisation est actuellement d'environ 4000 €/an (au cas où vous souhaiteriez devenir membre). Vous souhaitez simplement jeter un coup d'œil à la norme ? Il y a une solution moins coûteuse : l'interface AS est définie par un ensemble de normes industrielles, dont la plus populaire, et de loin, est la norme EN 62026-2:2015. Au comité officiel de normalisation, le document correspondant vous coûtera « seulement » environ 250 €. Si vous ne craignez pas l'effort d'une recherche personnelle, vous le trouverez aussi dans les bibliothèques des universités techniques ou des grandes écoles. Un prêt entre bibliothèques ou instituts concernés vous aidera si les documents n'étaient pas disponibles directement.
Vous souhaitez approfondir le sujet ? Vous aurez peut-être du mal à le faire, car à l'heure actuelle deux versions d'ASi coexistent sur le marché. Seule la version 3.0 est actuellement disponible que ce soit pour les spécifications ou le matériel. Cependant, depuis le salon SPS IPC Drives 2018, la version 5.0 apportant des améliorations a été largement évoquée.

La difficulté de ce développement somme toute banal, c'est que la spécification exacte du protocole version 5.0 (au moment de la rédaction de cet article) n'est même pas encore accessible à tous les membres. Selon des informations privilégiées, vous ne les obtiendrez que si vous êtes impliqué dans le développement ultérieur du protocole.
Il en résulte des phénomènes « intéressants » : les circuits intégrés émetteurs-récepteurs de la version 3.0 (détails ci-dessous) proviennent de la société ZMD, achetée entretemps par IDT, elle-même reprise par Renesas. Pour la version 5.0, on travaillera donc avec Renesas, dont la philosophie d'entreprise peut être résumée par : « Nous mettons en œuvre tous les protocoles ».

Câble ASi

La capacité « multidrop » (bus multipoint) de l'ASi est due en partie à la structure des câbles. Il s'agit d'un câble à deux fils qui transmet les signaux AS+ et AS−. AS− est la « masse », tandis que AS+, le signal positif. Notez que la « mise à la terre » de AS− n'est autorisée en aucun cas.
Concrètement : dans les installations, les câbles de données ASi sont presque toujours de couleur jaune. Le câble asymétrique typique est illustré à la **figure 3**, il présente une section transversale en forme de nez. Un câble noir optionnel de construction similaire est destiné à fournir une alimentation supplémentaire aux terminaux.
La plupart des appareils ASi disposent de connecteurs pour les câbles conformément à la figure 3, avec lesquels, lors de la pose, des dents percent l'isolant et établissent ainsi un contact galvanique avec les deux brins. Il est donc facile et sûr de rajouter un nouveau périphérique à un bus ASi. Bien que cette procédure soit courante et pratique, vous pouvez aussi utiliser n'importe quel autre câble pour créer un bus ASi.
À l'instar du bus « One-Wire » qui convient à la connexion directe de capteurs à un système à microcontrôleur, l'interface AS requiert également que l'alimentation d'appareils à faible consommation puisse être fournie directement par le bus. La tension nominale est de 24 V ; d'après la spécification, il est théoriquement possible de véhiculer jusqu'à 8 A sur le bus. Mais dans la pratique, au-delà de 2 A, il est recommandé de tenir compte de la chute de tension dans le câble et les connecteurs.
Il est important que l'interface AS soit libre de potentiel et n'aie pas de référence à la terre. Comme déjà mentionné, une mise à la terre est explicitement interdite. Outre des esclaves AS à part entière, il existe aussi l'option de seulement alimenter les périphériques ASi. Vous pouvez alors tirer jusqu'à 400 mA de la ligne 24 V, mais vous n'obtiendrez pas d'adresse et serez déconnecté de la communication. Comme un bus ASi peut avoir une longueur allant jusqu'à 100 m (voire 300 m avec une terminaison appropriée), il convient parfaitement pour alimenter des systèmes tiers à faible puissance. Un tel appareil est alors simplement connecté à AS+ et AS−.
L'alimentation d'un système ASi est généralement assurée par une alimentation appropriée, dont la structure est illustrée à la **figure 4**. Conceptions et constructions personnelles dans ce domaine professionnel sont strictement déconseillées. Vous trouverez sur le site web de l'association ASi [3] des liens vers de nombreuses entreprises proposant des produits certifiés.
Il est intéressant de noter que les alimentations fournissent systématiquement une tension continue de 29,5 à 31,6 V. Cette augmentation par rapport à 24 V est nécessaire pour compen-

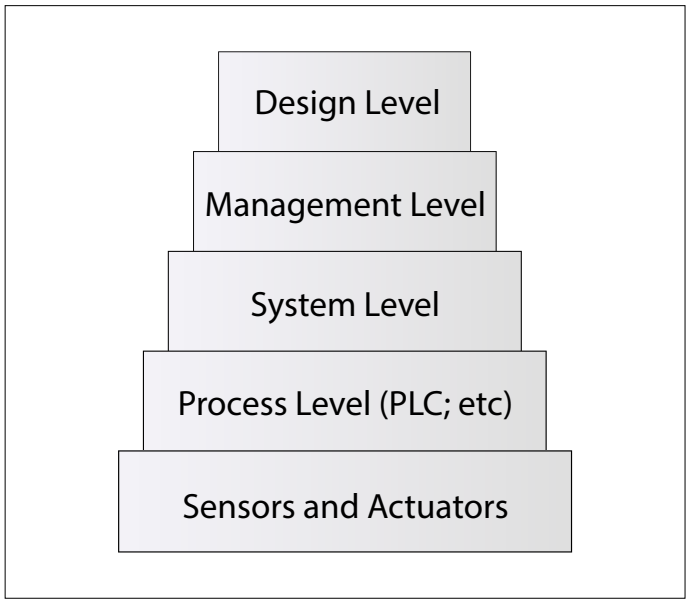


Figure 1. Les niveaux de hiérarchie de l'arbre d'automatisation.

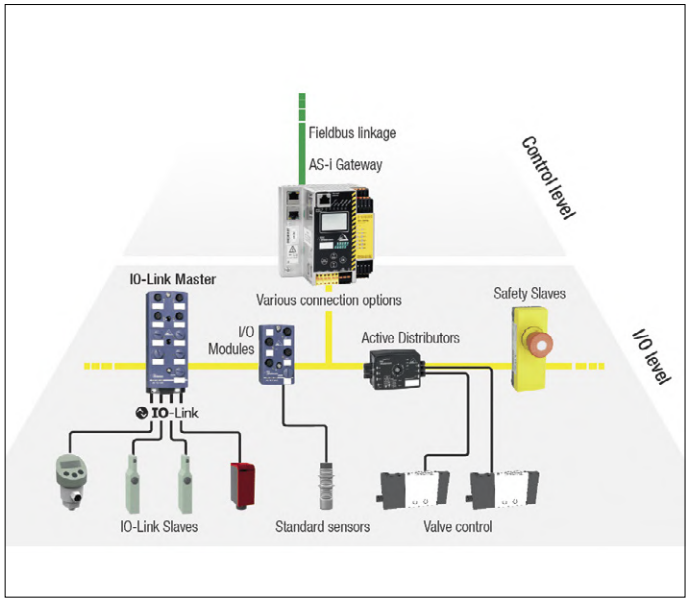


Figure 2. Du bus aux capteurs/actionneurs (source : [2]).

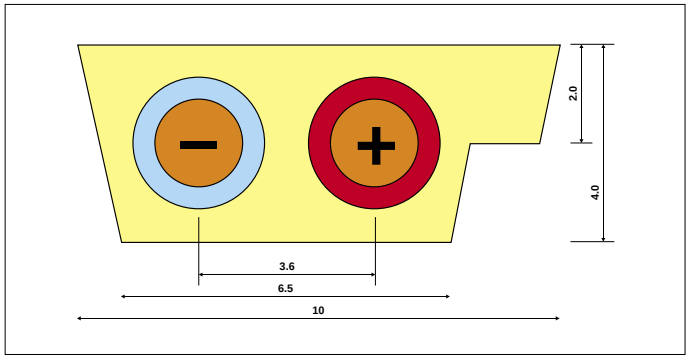


Figure 3. Section transversale d'un câble ASi (diam. de conducteur = 1,8 mm). Le « nez » empêche toute inversion de polarité accidentelle (image : d'après Bihl + Wiedemann).

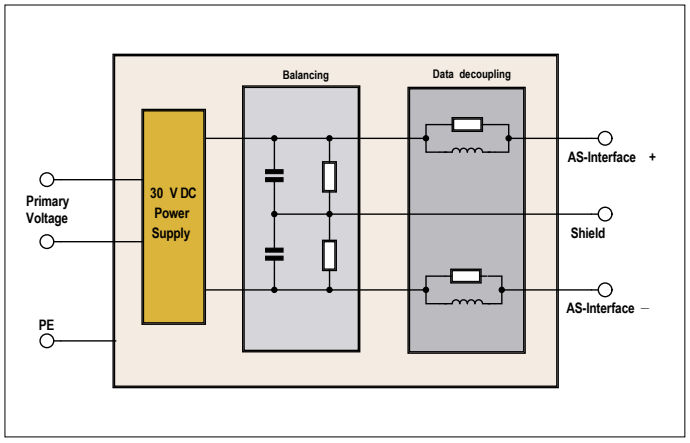


Figure 4. Les sources d'alimentation PELV ont une structure relativement complexe (source : [4]).

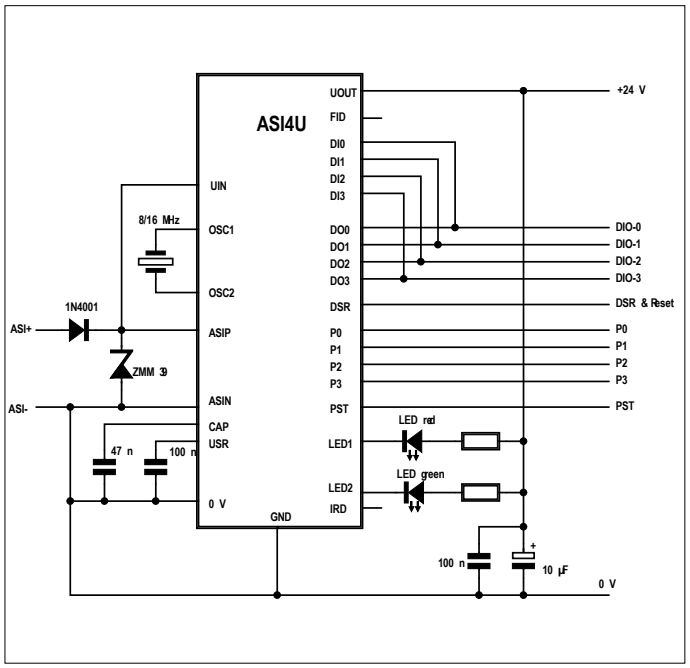


Figure 5. La puce d'émission-réception se contente de très peu de composants externes (source : [6]).

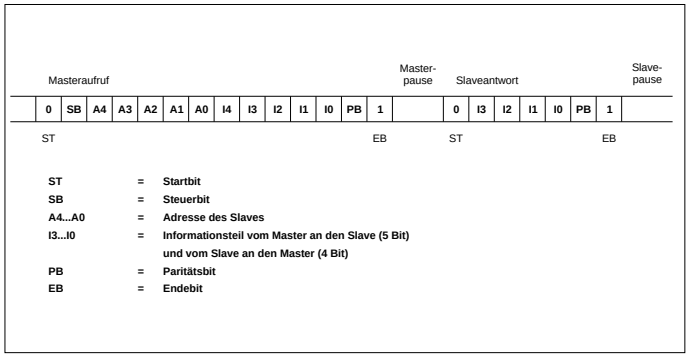


Figure 6. Format des données des messages ASi (source : [7]).

ser les chutes de tension dans les câbles et les connecteurs. Cependant, la très grande immunité aux interférences d'un système ASi ne constitue pas une carte blanche à un câblage négligé ou imprudent. Si vous tirez des câbles ASi dans une même baie, à proximité de câbles pollués par une ligne d'alimentation, vous risquez de vous exposer à des perturbations difficiles à détecter. Une présentation de Phoenix Contact [4] fournit plus d'informations sur la réalisation d'une installation.

Communication

Le câble illustré à la figure 3 montre clairement qu'il n'existe pas de lignes distinctes pour l'alimentation et les données. Il s'ensuit que les informations doivent d'abord être démodulées/transformées avant leur utilisation.

Les données sont transmises par APM (*Alternate Pulse Modulation*). Cette méthode s'appuie sur le codage Manchester qui combine horloge et chaîne de bits à transmettre. L'étape suivante consiste à s'assurer que le flux de bits codé par Manchester est converti en courant. Selon les informations librement disponibles, son amplitude est comprise entre 0 et 60 mA.

Les fronts négatifs entraînent une augmentation du courant de transmission, et les fronts positifs correspondent à une réduction. En raison de l'inductance de la ligne, ces changements de courant se transforment ensuite en impulsions de tension d'environ ±2 V par rapport à la tension nominale. Il est important aussi que le début d'un datagramme s'accompagne toujours d'une impulsion négative, ce qui est utile pour le déclenchement d'oscilloscopes numériques ou d'analyseurs de signaux. Malheureusement, les fabricants d'oscilloscopes ont jusqu'à présent montré peu d'intérêt pour l'intégration dans leurs appareils de décodeurs adéquats prêts à l'emploi. Même les produits de Pico Technology en sont dépourvus. Cependant, cette tâche peut être affectée à un outil tiers grâce à une API [5].

Certes, vous ne voudrez pas implémenter vous-même le décodeur pour votre propre module ASi ; la société IDT, déjà mentionnée, propose avec l'ASSI4U une puce d'émission-réception qui peut fonctionner à la fois en tant que maître et en tant qu'esclave. La **figure 5** montre le circuit de base tiré de la fiche technique.

Notez les deux LED, représentées verte et rouge par IDC dans sa fiche technique. En fait, le bus ASi spécifie que de telles LED soient présentes sur les esclaves. Elles donnent des informations sur le fonctionnement ; vous trouverez des informations supplémentaires dans la fiche technique mentionnée ci-dessus. Les broches marquées *DIx* et *DOx* sont les entrées et sorties. Si un émetteur-récepteur est connecté en mode esclave, alors il dispose des informations transmises sur le bus. Les entrées marquées *Px* sont utilisées pour définir des paramètres. Elles jouent un rôle important en particulier dans le mode maître, que nous ne présenterons pas ici.

La norme de communication

Maintenant que vous en savez un peu plus sur les caractéristiques électroniques d'un système ASi, passons à la logique du format de communication. La communication est en fait strictement gérée par le maître. Le maître exécute des processus cycliques de scrutation (*polling*) : il scrute les esclaves présents sur le bus les uns après les autres. Normalement les esclaves répondent avec des données utiles d'une longueur de 4 bits, après quoi le cycle recommence (voir figure 6).

Le format d'adresse définit le nombre maximal de périphériques

pouvant être adressés. Dans les datagrammes « normaux », seuls cinq bits d'adresse sont disponibles et l'adresse « 0 » est réservée à un nouvel esclave. Il reste donc 31 adresses possibles pour des esclaves. Les périphériques apparaissant avec l'adresse 0 se voient attribuer une nouvelle adresse disponible par le maître.
Avec un système ASi, la scrutation cyclique est suivie par un temps de réponse déterministe. Pour ASi 3.0, il est d'environ 5 ms, tandis que la nouvelle version 5 ne prévoit un temps de cycle que de 1,2 ms pour 24 utilisateurs. En outre, avec ASi 5.0, le nombre maximal d'esclaves est nettement plus élevé, à savoir 96.
Il est particulièrement important de pouvoir transmettre des données ou des paramètres. Un bit d'information indique le type d'information présente sur le bus. Par ailleurs un bit de contrôle différencie les adresses des commandes.

Comment procéder ?

Il vous faudra avoir une approche économique, que vous utilisiez un système ASi déjà existant, ou envisagiez de construire un nouveau système ASi. Certes la conception de modules avec capteurs peut être amusante, mais en pratique elle n'est pas rentable. Il existe déjà de nombreuses entreprises actives dans ce domaine. Elles proposent presque tous les capteurs et actionneurs imaginables équipés de l'interface AS. Il est donc beaucoup plus efficace de composer son propre système complet à partir de ces composants, et de le proposer à vos clients accompagnés de service de conseil.
Vous voulez vraiment expérimenter avec l'ASi ? Vous joindrez alors tôt ou tard l'organisation de normalisation. La norme a beau livrer toutes les informations nécessaires, la puce de l'émetteur-récepteur ASI4U a beau être disponible sur le marché selon *oemsecrets.com*, il s'en suit quand même une grande question.
Le problème, ce sont les droits sur la marque. Il devient vite juridiquement problématique d'utiliser le nom ASi ou même le logo ASi sans autorisation explicite. Ce n'est probablement qu'une question de temps avant que des plaintes ne soient déposées, si d'aventure vous vous risquez à proposer votre système sur le marché.
Cette situation n'est certes pas satisfaisante pour le technicien en électronique féru de travaux de laboratoire. D'un autre côté, travailler avec des systèmes ASi peut avoir un impact très positif sur votre compte en banque, car dans le domaine de l'automatisation industrielle, les honoraires relativement élevés des développeurs et consultants sont la norme. Cela peut compenser correctement l'effort consenti. ⏮

(190124-04 – version française : Xavier Pfaff)

Conseil : lisez la fiche technique !

En raison de sa taille, cet article ne peut donner qu'un aperçu général de la norme ASi. Par conséquent, il est impératif d'étudier dans le détail la fiche technique *asi4you* d'IDT avant toute démarche pratique [6].

En savoir plus

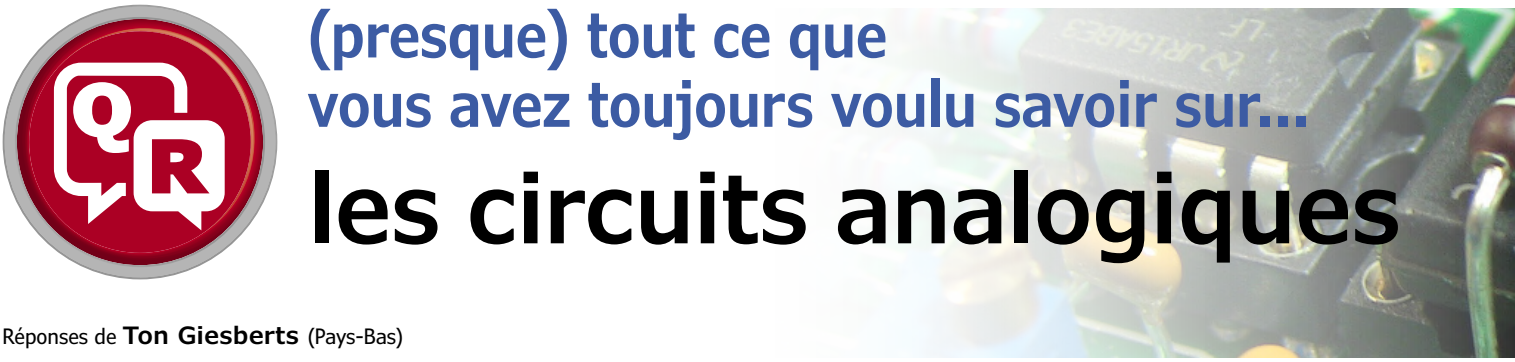
Outre la norme officielle, différents documents sur la norme ASi sont disponibles en ligne :

- www.agilicom.fr/tutorial-ASi.html
- www.bihl-wiedemann.de/fr/support/videos.html
- www.automation-sense.com/blog/automatisme/bus-de-terrain-asi-siemens.html

Recherchez sur la toile avec le mot clé « AS-Interface ».

Liens & litérature

[1] 'Industrial Control Electronics', J. M. Jacob, page 260 et suivantes
[2] Informations sur ASi : www.bihl-wiedemann.de/en/applications/communication/as-interface-and-io-link.html
[3] AS International Association : www.as-interface.net
[4] Phoenix Contact : www.phoenixcontact.com/assets/downloads_ed/global/web_dwl_technical_info/instrecomm10_e.pdf
[5] Outil ASi de Pico Technology : www.picotech.com/library/picoapp/as-iexpert-network-diagnosis-tool
[6] Feuille de caractéristiques ASi de IDT : www.idt.com/document/dst/asi4u-datasheet
[7] DESY : http://www-mks2.desy.de/content/e3740/e5177/e7190/e7997/e8006/e9274/e9656/index_ger.html



(presque) tout ce que vous avez toujours voulu savoir sur...

les circuits analogiques

Réponses de **Ton Giesberts** (Pays-Bas)

Q Comment maintenir le niveau de bruit d'un amplificateur le plus bas possible ?

R La valeur des résistances doit être la plus faible possible, mais pas trop, sinon l'amplificateur ou les divers étages risquent de ne pas pouvoir fournir le courant nécessaire ; il faut aussi tenir compte de la charge que représente la contre-réaction. On peut alors calculer la charge maximale, et prévoir une bonne réserve. Un courant trop élevé peut cependant causer une augmentation du bruit et de la distorsion. Les oscillations HF sont également la source de beaucoup de bruit, sans parler de la distorsion et du décalage des points de repos. Si vous utilisez des amplis op avec un gain proche de l'unité, choisissez un modèle qui est stable à gain unité ; d'autres types ne sont stables qu'avec un gain supérieur à 5 voire 10. Un truc, souvent utilisé pour les préamplis phono, est de mettre plusieurs transistors en parallèle.

Q Comment minimiser la distorsion sur une charge ?

R Il faut relier les masses du signal d'entrée, de la contre-réaction et de la charge en un seul point ; c'est ce qu'on appelle le câblage en étoile. Ce point central sera alors relié à la masse de l'alimentation. Toutes les liaisons devront aller vers ce point central et pas une des branches.

Q Comment avoir un faible décalage en tension dans un amplificateur ?

R La solution la plus simple est un condensateur en série avec l'entrée et la sortie, mais ce n'est pas possible si on doit passer du continu ou de très basses fréquences. Dans ce dernier cas, il faudrait de trop gros condensateurs. Une meilleure solution est de minimiser les courants d'entrée. Avec des transistors bipolaires, ils sont de l'ordre de quelques µA, avec des FET quelques nA, et encore moins avec des MOSFET. Les transistors à effet de champ produisent cependant un bruit à basse fréquence plus élevé. La technologie choisie dépendra aussi des autres caractéristiques du circuit (puissance, réponse en fréquence, etc.). Il faut en tout cas lire attentivement les fiches de caractéristiques et les recommandations du fabricant.

Q Comment protéger les entrées contre les surtensions, et à quoi prêter attention ?

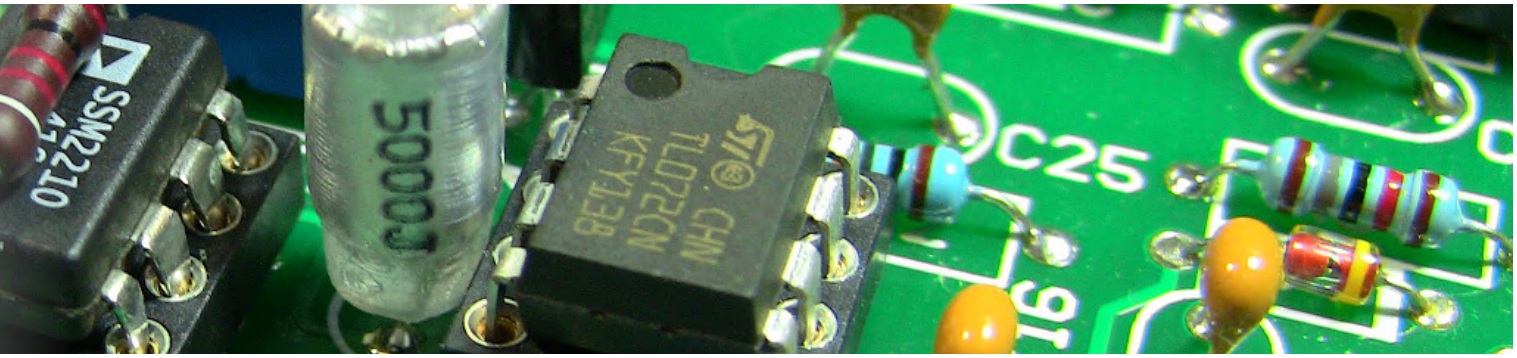
R Une résistance en série avec l'entrée est une option, mais le bruit augmente, ainsi que le risque d'interférences ; la bande passante peut aussi être limitée par cette résistance. Des diodes de protection entre entrée et rails d'alimentation sont une meilleure option, la valeur de la résistance série pourra être plus faible ; elles peuvent cependant entraîner un décalage de la tension avec de hautes impédances d'entrée, par exemple dans des sondes. Il faut alors recourir à des diodes spéciales, dont l'influence ne se fera sentir qu'en HF. Si les diodes sont parcourues par un courant d'intensité trop élevée, elles ne pourront pas non plus remplir leur rôle, à moins que le régulateur de tension ne soit conçu pour cela ou protégé. On peut aussi monter deux diodes Zener (ou plusieurs diodes en série) tête-bêche en parallèle sur l'entrée, c'est plus simple ; l'impédance d'entrée et la linéarité s'en ressentiront sûrement. Tout est affaire de compromis...

Q Pourquoi y a-t-il une masse analogique ?

R Dans un circuit qui utile des circuits analogiques et numériques, c'est une pratique courante de séparer les masses, c'est bien visible sur les circuits imprimés. Ceci permet d'éviter des interférences dues aux circuits numériques, et on prévoit aussi souvent deux alimentations séparées. Et si les circuits analogiques sont vraiment sensibles, il faudra encore une masse séparée pour l'entrée, ou une isolation galvanique.

Q Comment protéger un circuit analogique contre les parasites ?

R Un découplage correct des circuits d'alimentation est un bon début, par exemple des perles de ferrite sur les lignes d'alimentation des circuits numériques. L'utilisation de CMS, de plus en plus petits au fur et à mesure que la fréquence augmente, est parfois la seule solution ; les composants doivent alors être au plus près de la source des interférences pour minimiser les liaisons inductives. Il faut encore tenir compte de la réjection en mode commun, les parasites étant souvent en mode commun dans ces circuits. Il est utile de rappeler que les liaisons entre composants ou circuits fonctionnent comme des antennes pour capter les parasites ; c'est pour ça qu'on place de grosses « perles » de ferrite sur les câbles d'alimentation ou ceux de liaison entre ordinateurs et périphériques. En ce qui concerne les circuits, leur bande passante et leur temps de montée (*slew rate*) doivent être limités au strict minimum. Et bien entendu, le dessin des circuits imprimés doit se faire dans les règles de l'art : pas de boucles, retours de courant les plus courts possible, multiplication des couches si on ne peut éviter les longues liaisons autrement, etc. En HF, il faudra souvent des écrans et blindages, et toujours cette recherche de la réjection des signaux en mode commun...



Q Quelle peut être l'origine du ronflement dans un amplificateur ?

R Nous avons déjà répondu en partie à cette question, et nous ne parlons bien entendu pas du blindage déconnecté d'un câble ou de l'inversion par inadvertance des connexions signal et masse. Il s'agit plutôt du champ magnétique émis par un transformateur à proximité d'un circuit qui perturbe ce dernier. On aura déjà pensé à un transformateur torique, mais eux aussi émettent un champ magnétique. On peut prévoir un écran en mu-métal ou une plus grande distance entre transformateur et circuit, si c'est possible... Un transformateur à noyau en E à tôle feuilletée est encore meilleur qu'un modèle torique, et on peut le placer dans un coin à un angle de 45° par rapport aux côtés du boîtier, où les champs parasites seront moins gênants. Les diodes et ponts de redressement sont aussi une source de perturbations, c'est pourquoi on place souvent des filtres (*snubber*), constitués de condensateurs avec ou sans résistance série, en parallèle sur les diodes ou les bornes du transformateur. Les ondulations de la tension d'alimentation sont souvent une source de ronflement dans les étages d'amplification sensibles, par ex. ceux des micros, mais aussi dans les récepteurs et autres circuits HF. Le dessin du circuit imprimé et le câblage des masses sont d'une grande importance, et il faut toujours éloigner les circuits sensibles des sources potentielles de parasites. Les oscillations, dans un amplificateur par ex., sont une autre source d'interférence ; elles sont plus difficiles à localiser et peuvent entraîner une consommation plus élevée, avec une augmentation de l'ondulation de l'alimentation, etc.

Q Quelle est la meilleure manière de transmettre un signal de faible amplitude ?

R La manière la plus simple est via une ligne couplée à une masse ; il s'agit là d'une liaison asymétrique. Le câble coaxial a en général la préférence, mais deux conducteurs parallèles ou une paire torsadée suffisent parfois. Il peut exister une différence de potentiel entre les masses de deux appareils, à cause des courants circulant dans les circuits. Un câble dans un champ peut aussi causer ces perturbations, et elles peuvent apparaître dans une liaison entre deux parties d'un même circuit imprimé. Une transmission symétrique du signal peut permettre d'éviter les perturbations, surtout sur de longues distances. C'est également la seule option si on veut une certaine plage dynamique du signal (c'est-à-dire la possibilité de traiter de très faibles signaux). On utilise deux signaux en opposition de phase, et toute perturbation commune aux deux signaux sera annulée. L'amplitude du signal est aussi deux fois plus élevée qu'avec une liaison asymétrique. Pour toutes ces raisons, les CA/N à haute résolution ont en général une entrée symétrique.

Q Y a-t-il quelque chose à ne pas oublier lors des mesures sur un circuit ?

R Lors de l'utilisation d'un multimètre, il faut tenir compte de son impédance interne, qui peut causer des erreurs lors de la mesure de tensions ou de courants. Si le circuit est particulièrement sensible, le fait d'approcher la pointe de mesure ou un doigt peut causer des perturbations ou même des oscillations. Cela donnera des résultats de mesure erronés, et il faut en tenir compte. En cas de doute il vaut mieux vérifier le comportement du circuit à l'oscilloscope lors des mesures ; sans oublier que les sondes ont aussi une impédance d'entrée... Une sonde active a en général une impédance plus élevée, avec une capacité plus faible, et est une meilleure option pour les mesures sur les circuits sensibles et/ou en HF.

Q Quels condensateurs faut-il éviter en audio ?

R La qualité des condensateurs, qui dépend de leur construction et du diélectrique utilisé, est un sujet qui a fait couler beaucoup d'encre, tant il est vrai que ces composants peuvent affecter – en général négativement – les performances d'un circuit. La capacité varie avec la tension et la température, en fonction du diélectrique utilisé ; les plus mauvais de ce point de vue sont sans doute les condensateurs céramique. Les contraintes sur les armatures peuvent aussi causer de la microphonie, et les caractéristiques varient avec l'âge. Dans le cas des condensateurs électrolytiques, on note encore un phénomène d'instabilité de la capacité, et on peut observer un effet de distorsion de croisement (comme sur les amplis en classe B) dû à l'hystérésis de charge. Un autre problème possible est l'effet de batterie : il y a réapparition d'une tension aux bornes du condensateur après une décharge brusque. En général, il vaut mieux utiliser des condensateurs à isolant plastique (polyester, polystyrène ou polypropylène) ; la qualité du condensateur est en général inversement proportionnelle à la constante diélectrique de l'isolant. L'inconvénient de ces condensateurs est leur faible capacité, quelques nF tout au plus pour ceux au polystyrène. Il faudra donc choisir en fonction de la valeur de capacité nécessaire pour l'application et/ou des dimensions du circuit. ⏮

(190099-04 – version française : Jean-Louis Mehren)

HORS CIRCUITS

comment choisir un amplificateur opérationnel ?

tension de décalage, courant d'entrée, taux de réjection de mode commun…

Robert Iacoste (Chaville)

Même si les amplificateurs opérationnels ont de nombreuses qualités (compacts, bon marché, simples…), ils ne sont pas parfaits. Cet article passe en revue les caractéristiques à prendre en compte pour choisir le modèle le mieux adapté à l'application envisagée. Le lecteur devrait s'y retrouver plus facilement dans les données des fiches techniques.

Démarrons par un petit retour dans le passé. Au commencement étaient les tubes et les transistors. Les concepteurs choisissaient avec soin le composant le mieux adapté à l'application, calculaient laborieusement les composants passifs nécessaires et optimisaient avec délicatesse chaque étage du circuit. Puis les circuits intégrés ont été inventés (Jack Kilby, chez Texas Instrument, a fabriqué les premières puces en 1958, même si le concept a été breveté dix ans plus tôt par Werner Jacobi). Rapidement, cette technologie a été utilisée pour créer des composants complexes : le premier amplificateur opérationnel intégré (AOP en abrégé), le µA702, a été conçu par Bob Wildar (Fairchild) en 1963. D'ailleurs vous serez peut-être surpris d'apprendre que les amplificateurs opérationnels existaient bien avant les circuits intégrés : la **figure 1** vous en donne un exemple. Évidemment avec l'intégration sur une puce, leur succès a pris une tout autre dimension. Le même Bob Wildar a ensuite conçu le µA709 (1965), et est passé chez National Semiconductors pour créer le LM101. En réponse, Fairchild, lâché par son gourou, a lancé le célébrissime µA741 (1968). Et oui, vous avez sûrement déjà utilisé ce composant, et il a 50 ans !

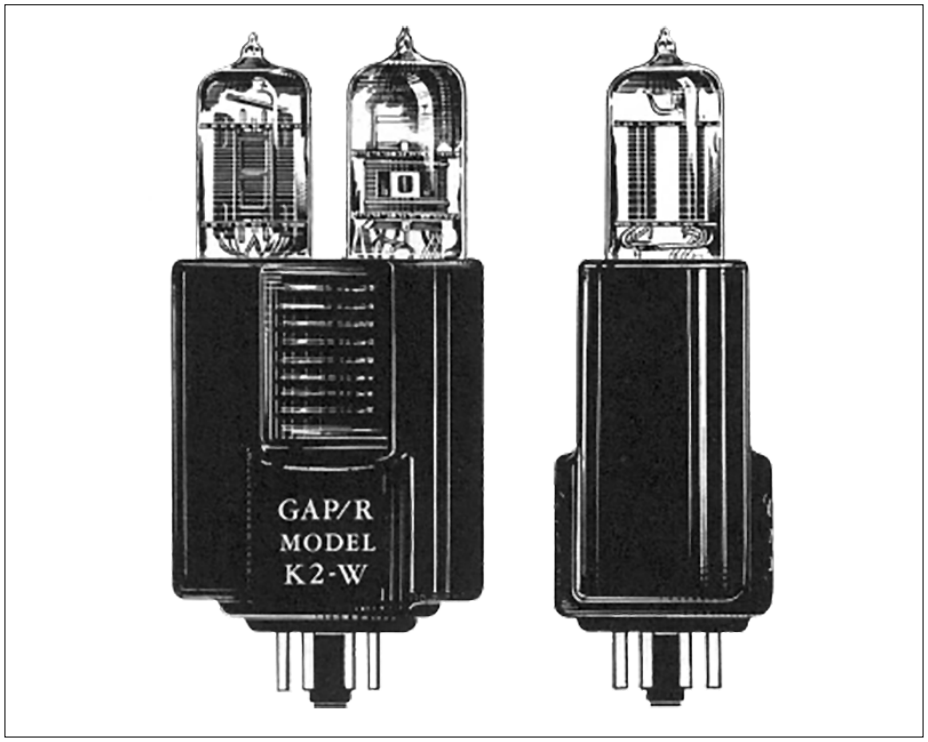


Figure 1. Le K2-W est le premier amplificateur opérationnel « intégré » (1952). Il a été conçu par la société George A. Philbrick Researches (GAP-R), de Boston. Source : www.philbrickarchive.org

L'histoire des AOP est donc une belle aventure : compacts, bon marché, simples et très flexibles, que demander de plus ? Malheureusement, les concepteurs que nous sommes oublient parfois que ces petites puces ne sont rien de plus qu'un ensemble de transistors. Ce ne sont pas des boîtes noires magiques aux caractéristiques parfaites. De là, le choix du bon AOP pour son projet ne doit pas être laissé au hasard. Dans cet article, je vais tenter de vous donner quelques pistes pour y voir clair : quelles sont les principales caractéristiques d'un AOP ? Lesquelles sont importantes pour votre projet ? Comment lire et exploiter les dizaines de pages de chiffres et de graphiques qui remplissent les fiches techniques de ces composants ?

Quelques bases

Commençons par quelques notions sur les AOP. Fondamentalement, un AOP est un soustracteur de tension suivi d'un amplificateur (**fig 2**). Il a deux entrées : une entrée non-inverseuse (IN+), et une entrée inverseuse (IN−). Sa tension de sortie est simplement la différence de tension entre les deux, multipliée par un gain appelé **gain en boucle ouverte** :

$$V_{OUT} = GAIN_{BOUCLE\ OUVERTE} \times (V_{IN+} - V_{IN-})$$

La caractéristique importante de tous les AOP est que ce gain de boucle ouverte est très élevé. Quand je dis très élevé, c'est vraiment très élevé : un gain de 1 000 000 n'est pas rare. Donc, dès que l'écart de tension entre les deux entrées n'est pas quasi nul, la tension de sortie part en butée (c'est-à-dire se rapproche autant qu'elle peut de la tension d'alimentation ou de la masse, selon le signe de la différence). De là, un AOP, s'il est utilisé sans composants externes, est un comparateur de tension : sa sortie indique simplement laquelle des deux entrées est soumise à la tension la plus élevée. En règle générale, on l'utilise avec des composants externes qui créent une contre-réaction. Je vais vous l'expliquer dans quelques minutes, mais pour le moment gardez à l'esprit la première règle d'or des AOP, qui découle de ce gain très élevé :

Règle 1 : si sa sortie n'est pas saturée, la différence de tension entre les deux entrées d'un AOP est toujours quasi nulle.

Évidemment cette règle n'est vraie que tant que l'on considère que l'AOP est parfait. Une autre caractéristique importante d'un AOP est que les courants circulant sur ses entrées sont très faibles (couramment quelques nA ou moins). Cela a conduit à la deuxième règle d'or, une fois de plus pour un AOP idéal :

Règle 2 : le courant à travers les entrées d'un AOP est quasi nul.

Enfin, un amplificateur opérationnel doit bien sûr être alimenté. Il a deux broches d'alimentation, une positive et une négative. Vous pouvez soit connecter ces broches V+ et V− à une source d'alimentation symétrique (par ex. ±10 V), soit mettre V− à la masse et relier V+ à une tension d'alimentation positive. Ne vous laissez pas berner par les publicités clamant qu'un circuit est conçu pour une alimentation bipolaire ou unipolaire : un AOP ne fait pas la différence entre une alimentation de ±10 V et une alimentation unipolaire de 0/+20 V, car il n'a pas de prise de terre !

Circuits classiques

Un AOP permet de réaliser nombre de circuits : amplificateurs, différentiateurs, intégrateurs, filtres, oscillateurs, etc. Pour cet article, je resterai sur les

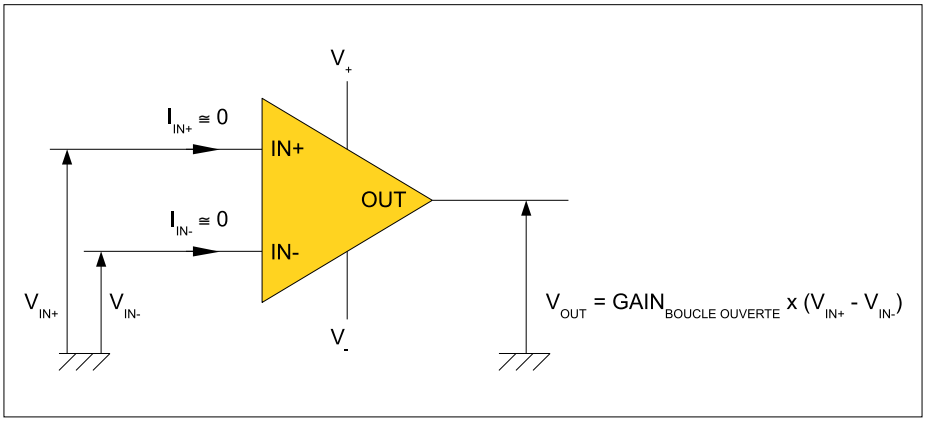


Figure 2. Un amplificateur opérationnel n'est rien de plus qu'un amplificateur différentiel, dont les principales caractéristiques sont un gain en boucle ouverte très élevé et des courants d'entrée très faibles.

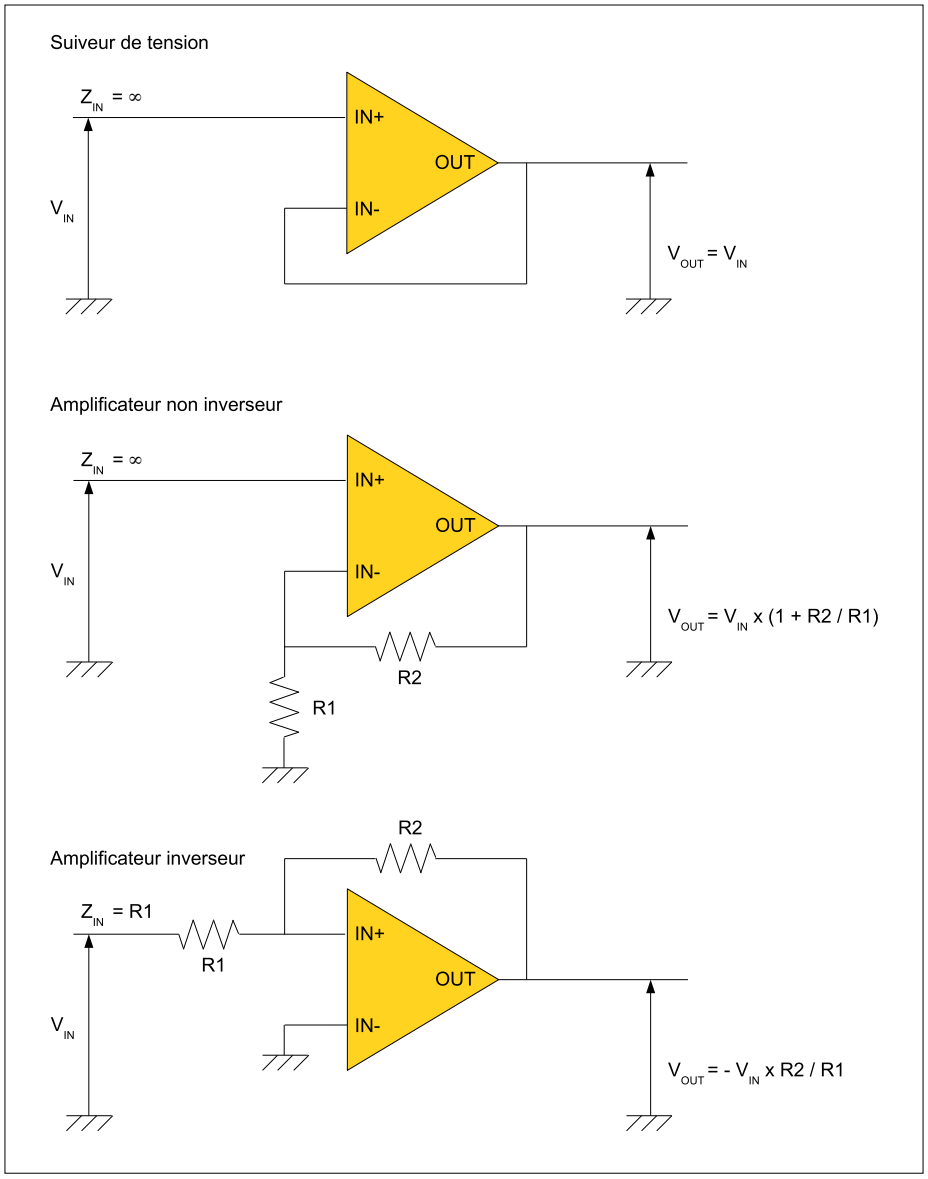


Figure 3. Quelques schémas types d'applications d'un AOP. Comme on le voit, il y a presque toujours une boucle de contre-réaction autour de l'amplificateur.

trois exemples les plus élémentaires, illustrés à la **figure 3** : le suiveur de tension, l'amplificateur non-inverseur et l'amplificateur inverseur. Comment fonctionnent-ils ? Comment calculer la valeur des résistances ? C'est ultra simple.

Commençons par le **suiveur de tension**. Regardez de nouveau la figure 3 : la tension d'entrée est simplement connectée à l'entrée non-inverseuse, et l'entrée inverseuse est connectée à la sortie de l'AOP. Vous vous rappelez la règle 1 ? La tension entre les deux entrées est quasi nulle, donc la tension de sortie est simplement égale... à la tension d'entrée. De plus, comme le courant d'entrée est nul (c'est la règle 2), l'impédance d'entrée est quasi infinie : ce circuit recopie une tension, tout en ayant une impédance très élevée en entrée et très faible en sortie.

Quid de l'**amplificateur non-inverseur** ? Reprenons la figure 3. Cette fois, deux résistances supplémentaires sont utilisées. La tension d'entrée est toujours connectée à l'entrée non-inverseuse, donc la tension sur l'entrée inverseuse doit également être très proche de la tension d'entrée V_{IN}. Mais, attendez, cette entrée inverseuse est reliée à la masse par R1. Vous vous rappelez la loi d'Ohm ? Le courant traversant R1 doit forcément être $I = U/R = V_{IN}/R1$. Mais souvenez-vous aussi qu'aucun courant ne circule sur l'entrée de l'AOP (règle 2). Le même courant I doit donc passer par R2, qui relie l'entrée inverseuse et la sortie de l'AOP. La chute de tension à travers R2 est donc $U = R \times I = R2 \times (V_{IN}/R1)$. Quelle est donc la tension à la sortie de l'amplificateur opérationnel ? C'est la tension sur l'entrée inverseuse (qui est égale à V_{IN}), plus cette chute de tension aux bornes de R2 que nous venons de calculer. On a donc $V_{OUT} = V_{IN} + (R2 \times V_{IN}/R1)$, c'est-à-dire $V_{OUT} = V_{IN} \times (1 + R2/R1)$. Voilà, c'est donc un amplificateur non-inverseur et son gain est fixé par R1 et R2.

Vous avez compris qu'il est très facile de calculer les grandeurs (tension, courant…) présentes dans ces circuits en utilisant simplement les deux règles d'or. Allez, faites donc un petit exercice et calculez de même le gain de l'**amplificateur inverseur** illustré également sur la figure 3, c'est super simple...

Zéro ? Enfin presque !

Jusqu'à présent, j'ai supposé que l'amplificateur opérationnel est parfait. Imaginons qu'on utilise plutôt un AOP « réel », par ex. notre vieil ami le UA741 (équivalent du µA741). Quels sont les écarts ? Pour cela, il faut jeter un œil à la fiche technique du composant [1]. Regardez donc la **figure 4**, extraite du site web de STMicroelectronics. Que diantre signifient tous ces chiffres ?

Pas de panique ! Le 1er paramètre est intitulé « **input offset voltage** » (voir l'encadré « **glossaire** » pour la traduction). C'est simplement une mesure de l'erreur de cet AOP par rapport... à la première règle d'or. Comme le circuit n'est pas parfait, la différence de tension entre les deux entrées n'est pas strictement nulle. Ici, l'erreur typique est de 1 mV. Le fabricant nous dit aussi que cette erreur peut aller jusqu'à 5 mV, et même jusqu'à 6 mV lorsque la température n'est pas de 25 °C. Par simplicité, considérons juste l'erreur typique de 1 mV. Est-ce suffisant ? Et bien tout dépend de votre application : 1 mV peut être complètement négligeable, ou rendre votre circuit inutilisable. Imaginez que vous concevez un amplificateur pour un thermocouple. La tension de sortie d'un tel capteur est de l'ordre de 10 mV. L'erreur de 1 mV introduirait une erreur de 10% sur la sortie ! C'est pour cela que certains AOP sont optimisés pour de telles applications, par ex. ceux caractérisés par « zero-drift ». Ils sont spécialement conçus pour réduire au minimum la tension d'offset, grâce à des circuits de compensation intégrés. Un exemple ? Regardez sur le web la fiche technique du TLC2652 (Texas Instruments) [2]. Cette

puce a une erreur d'offset de 1 µV, et une variation de cette erreur avec la température encore plus impressionnante de 0,003 µV/°C. Pas mal non ?
Revenons à la fiche technique du UA741 (figure 4), le 3e paramètre s'appelle « **input bias current** ». Comme vous pouvez l'imaginer, c'est lié à la deuxième règle d'or. Le courant passant par les entrées n'est pas strictement nul. Sur l'UA741, ce courant est généralement inférieur à 10 nA, mais peut aller jusqu'à 200 nA. De plus, le courant circulant dans les deux entrées n'est pas égal, la différence est appelée « **input offset current** » (figure 4, 2e paramètre), et égale à 2 nA. Ce sont des courants très faibles, mais sont-ils assez faibles ? Encore une fois, cela dépend de votre application. Si vous concevez un amplificateur non-inverseur avec deux résistances externes, cette petite fuite introduira une erreur sur le gain théorique. Faites donc le calcul, ces 10 nA doubleront approximativement l'erreur d'offset de 1 mV si les résistances sont de l'ordre de R = U/I = 2 mV/10 nA = 200 kΩ. Ce ne sera donc pas un problème si vous utilisez des résistances de faible valeur, mais cela peut devenir un sérieux souci sinon. Une autre source d'embêtement est que le même courant de fuite sera appliqué à l'entrée de votre circuit. 10 nA sera généralement négligeable, mais ce ne sera pas le cas si vous voulez amplifier un très faible courant provenant par ex. d'une photodiode. Encore une fois, il existe des amplificateurs opérationnels spécialisés pour ce type d'applications, appelés « amplificateurs opérationnels à faible *bias* ». Regardez par ex. le LTC6268 d'Analog Devices (ex Linear Technologies) [3]. Son courant de polarisation typique est de 3 fA. Oui, trois femto-ampères, c'est trois millions de fois moins que l'UA741...
Enfin, le 4e paramètre de la fiche technique reproduite sur la figure 4 est le gain en boucle ouverte dont je parlais au début de cet article. Pour l'UA741, il est de 200 V/mV, ce qui signifie un gain de 200 000. C'est beaucoup, mais pas infini. Encore une fois, cela peut être acceptable ou non, selon votre projet.

Symbol	Parameter		Min.	Typ.	Max.	Unit
V_{io}	Input offset voltage (R_S ≤10 kΩ) T_{amb} = +25 °C T_{min} ≤T_{amb} ≤T_{max}			1	5 6	mV
I_{io}	Input offset current T_{amb} = +25 °C T_{min} ≤T_{amb} ≤T_{max}			2	30 70	nA
I_{ib}	Input bias current T_{amb} = +25 °C T_{min} ≤T_{amb} ≤T_{max}			10	100 200	
A_{vd}	Large signal voltage gain (V_o = ±10 V, R_L = 2 kΩ) T_{amb} = +25 °C T_{min} ≤T_{amb} ≤T_{max}		50 25	200		V/mV
SVR	Supply voltage rejection ratio (R_s ≤10 kΩ) T_{amb} = +25 °C T_{min} ≤T_{amb} ≤T_{max}		77 77	90		dB
I_{CC}	Supply current, no load T_{amb} = +25 °C T_{min} ≤T_{amb} ≤T_{max}			1.7	2.8 3.3	mA
V_{icm}	Input common mode voltage range T_{amb} = +25 °C T_{min} ≤T_{amb} ≤T_{max}		±12 ±12			V
CMR	Common mode rejection ratio (R_S ≤10 kΩ) T_{amb} = +25 °C T_{min} ≤T_{amb} ≤T_{max}		70 70	90		dB
I_{OS}	Output short circuit current		10	25	40	mA
$±V_{opp}$	Output voltage swing T_{amb} = +25 °C T_{min} ≤T_{amb} ≤T_{max}	R_L = 10 kΩ R_L = 2 kΩ R_L = 10 kΩ R_L = 2 kΩ	12 10 12 10	14 13		V

Figure 4. Voici un extrait de la fiche technique de l'UA741, qui montre toutes les caractéristiques en continu de cet AOP historique. Source : STM

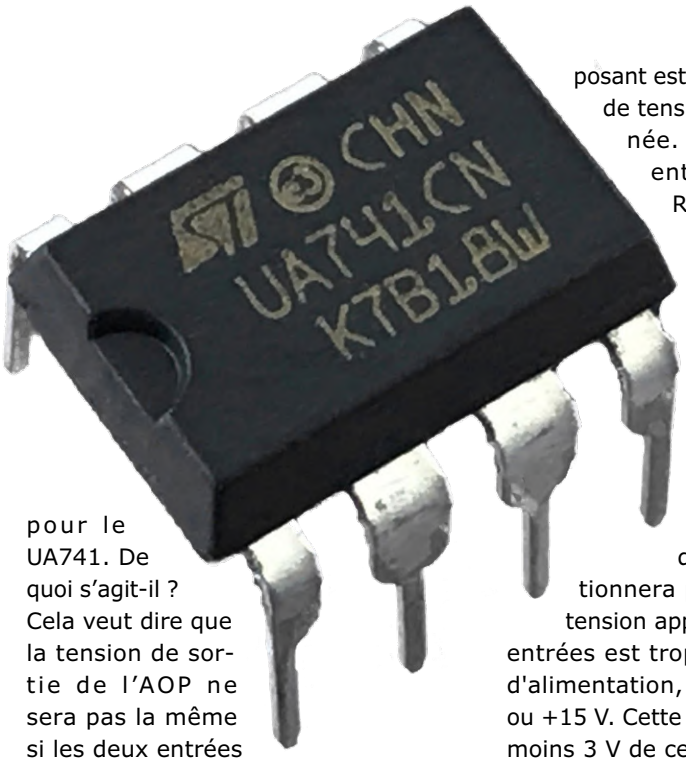
D'autres sources d'erreurs

Comme on l'a vu, la tension de sortie d'un AOP est normalement proportionnelle à la différence de tension entre ses deux entrées et rien de plus. Mais évidemment, c'est un peu plus compliqué dans la vraie vie. Quelles sont les sources d'erreurs ? Tout d'abord la tension de sortie dépend aussi un peu de la tension d'alimentation de l'ampli. Cette dépendance est spécifiée dans la fiche technique, c'est le « **supply voltage rejection ratio** ». Revenons à la figure 4 : pour l'UA741, c'est 90 dB. Cela signifie que si l'amplificateur est configuré pour un gain de 1 (suiveur de tension), l'erreur de tension sur la sortie pour une variation de tension de 1 V sur l'alimentation sera de 1 V × $10^{-90/20}$, soit 31 µV. Attention, cette erreur serait cent fois plus forte pour un gain de 100, c'est-à-dire 3 mV, et là ça commencerait à être significatif. Ce phénomène de couplage entre alimentation et tension de sortie peut également provoquer des oscillations pénibles si l'alimentation n'est pas correctement filtrée.
L'autre source d'erreur classique est appelée « **common mode rejection ratio** », ce taux est également de 90 dB

Glossaire	
common mode rejection ratio	taux de réjection du mode commun
gain bandwidth product	produit gain × bande passante
input bias current	courant de polarisation des entrées
input common mode voltage range	plage de tension d'entrée de mode commun
input offset current	courant de décalage d'entrée
input offset voltage	tension de décalage d'entrée
output voltage swing	excursion de la tension de sortie
large signal voltage gain	gain en boucle ouverte
rail to rail input	entrée à excursion totale
rail-to-rail output	sortie à excursion totale
slew rate	vitesse de montée
supply voltage rejection ratio	taux de réjection de la tension d'alimentation

Symbol	Parameter	Min.	Typ.	Max.	Unit
GBP	Gain bandwidth product V_i = 10 mV, R_L = 2 kΩ, C_L = 100 pF, f =100 kHz	0.7	1		MHz
THD	Total harmonic distortion f = 1 kHz, A_v = 20 dB, R_L = 2 kΩ, V_o = 2 V_{pp}, C_L = 100 pF, T_{amb} = +25° C		0.06		%
e_n	Equivalent input noise voltage f = 1 kHz, R_S = 100 Ω		23		$\frac{nV}{\sqrt{Hz}}$
Øm	Phase margin		50		Degree
SR	Slew rate V_i = ±10 V, R_L = 2 kΩ, C_L = 100 pF, unity gain	0.25	0.5		V/µs
t_r	Rise time V_i = ±20 mV, R_L = 2 kΩ, C_L = 100 pF, unity gain		0.3		µs
K_{ov}	Overshoot V_i = 20 mV, R_L = 2 kΩ, C_L = 100 pF, unity gain		5		%

Figure 5. La fiche technique d'un AOP comprend également les caractéristiques en alternatif. Ici de nouveau celles du vénérable UA741.



pour le UA741. De quoi s'agit-il ? Cela veut dire que la tension de sortie de l'AOP ne sera pas la même si les deux entrées sont respectivement à 2 V et 2,001 V, ou à 3 V et 3,001 V. Ce devrait être le cas, car l'ampli-op ne fait que mesurer la différence, mais ça ne l'est pas parce qu'il n'est pas parfait. Évidemment certains circuits sont spécifiquement optimisés pour avoir une réjection de mode commun bien plus grande qu'un AOP ordinaire. C'est par ex. le cas des composants appelés « instrumentation amplifiers », comme l'AD8422 (Analog Devices) [4] qui a un taux de réjection de mode commun de 150 dB. C'est une sacrée différence par rapport à 90 dB.

Excursion totale ?

Parlons maintenant un peu de l'alimentation de l'AOP. Bien sûr, chaque composant est spécifié pour une plage de tension d'alimentation donnée. Mais qu'en est-il des entrées et des sorties ? Regardez encore une fois la spécification du UA741 (fig. 4). Vous voyez le paramètre intitulé « **input common mode voltage range** » ? Elle est de ±12 V pour une alimentation de ±15 V. Qu'est-ce que cela signifie ? Simplement que cet AOP ne fonctionnera pas correctement si la tension appliquée sur l'une de ses entrées est trop proche des tensions d'alimentation, c'est-à-dire de −15 V ou +15 V. Cette tension doit rester à au moins 3 V de ces limites supérieure et inférieure. Imaginez maintenant que vous utilisez ce UA741 avec une pile de 9 V. Vous connectez sa patte V− à la masse, et la pile de 9 V à V+. Vous devrez alors vous assurer que la tension d'entrée n'est jamais inférieure à 0+3 = 3 V, et jamais supérieure à 9−3 = 6 V. 3 V à 6 V, c'est une plage très réduite ! Une fois de plus, il existe des amplificateurs opérationnels spécialement conçus pour tolérer une plage de tension d'entrée aussi large que possible, caractérisés par des « **rail-to-rail inputs** ». Il y a même des amplificateurs opérationnels allant plus loin, comme le LT1783 (Analog Devices) [5] qui accepte des tensions d'entrée de 0 à 18 V, même s'il est alimenté en 2,5 V. Passons maintenant à la sortie de l'AOP : le même problème se pose. Toujours avec une alimentation de ±15 V, notre cher ami l'UA741 est ainsi spécifié pour un « **output voltage swing** » de ±13 V avec une charge de 2 kΩ. Cela signifie que sa sortie ne peut pas s'approcher trop près de V+ ou V−. Nombre de circuits ne fonctionnent pas parce que leurs concepteurs ont oublié cette limitation. Imaginez qu'un UA741 est alimenté en unipolaire et que sa sortie pilote un transistor NPN câblé en émetteur commun. Comme sa sortie sera toujours au moins à 2 V, c'est-à-dire supérieure à 0,6 V, le transistor sera toujours passant. Évidemment, je ne vous surprendrais pas en vous disant que certains AOP sont spécifiquement conçus pour avoir une sortie pouvant s'approcher très près de V+ et V−, ils sont caractérisés par une « **rail-to-rail output** ». Par exemple, la sortie du MCP6001 (Microchip) [6] se rapproche jusqu'à 25 mV des deux rails d'alimentation.

Et en alternatif ?

J'ai fait le tour des principaux paramètres en continu (DC) d'un AOP, mais quid de son comportement en alternatif ? Gardons donc le même exemple : les principales spécifications en alternatif (AC) de l'UA741 sont reproduites à la **figure 5**. Le premier paramètre s'appelle « **gain bandwidth product** » ou GBP. Pour l'UA741, il vaut 1 MHz. Pour comprendre ce dont il s'agit, vous devez savoir que le gain en boucle ouverte d'un AOP en fonction de la fréquence est approximativement une droite sur une échelle logarithmique. Qu'est-ce que ça veut dire ce charabia ? Simplement que ce gain, exprimé en tension, est divisé par deux quand la fréquence du signal double. Pour ceux qui aiment les dB, cela signifie que le gain en boucle ouverte est réduit de 6 dB chaque fois que la fréquence est doublée. Il y a donc une fréquence où ce gain en boucle ouverte est de 1, cette fréquence est par définition appelée GBP. Si vous voulez voir cela sous forme graphique, jetez donc un œil à la **figure 6**. C'est le gain en boucle ouverte du UA741 en fonction de la fréquence. Ce qui est encore plus intéressant, c'est que ce même graphe permet d'estimer la bande passante maximale d'un amplificateur en boucle fermée. Il suffit pour ça de repérer la valeur de gain désirée, et de tracer une ligne horizontale. Vous obtiendrez ainsi une approximation de la fréquence de coupure de l'amplificateur. Par exemple si vous voulez

concevoir un amplificateur avec un gain de 40 dB (soit un gain de tension de 100, car 20 ×log(100) = 40), vous trouvez 10 kHz. Donc un amplificateur basé sur UA741 avec un gain de tension de 100 ne fonctionnera que jusqu'à 10 kHz. Plus vous voulez de gain, moins grande sera la bande passante. Bien sûr il y a des tas d'AOP beaucoup plus rapides que ce vénérable UA741. Allez, un exemple : le LMH5401 (Texas Instruments) [7] a un produit gain ×bande de 8 GHz...
Un paramètre étroitement lié à la GBP est le « **slew rate** » de l'AOP, à savoir la vitesse maximale à laquelle la tension de sortie peut bouger. Elle est de 0,5 V/µs pour le UA741... mais de 17.500 V/µs pour le LMH5401. Par exemple 17.500 V/µs signifie que sa sortie peut passer de 1 V à 3 V en 120 ps. Impressionnant, non ?

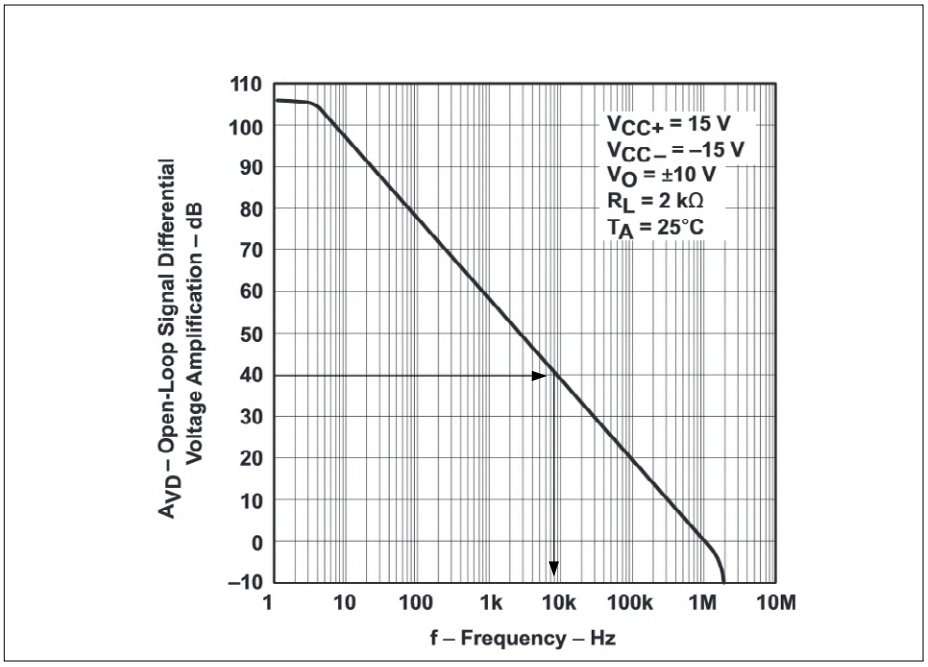


Figure 6. Ce graphique, extrait de la fiche technique du µA741 de Texas Instruments, donne le gain en boucle ouverte en fonction de la fréquence du signal. Il est évident qu'avec un tel circuit, on ne peut pas espérer une bande passante de plus de 10 kHz pour un gain de 40 dB.

Pour conclure

Plein d'autres paramètres sont évidemment importants, comme le bruit ou la stabilité, mon seul but dans cet article était de vous prouver que le choix d'un AOP ne doit pas se faire au hasard. Tout d'abord, vérifiez quels sont les paramètres critiques pour votre projet : tension de décalage ? Courant d'entrée ? Erreur de mode commun ? Caractéristiques *rail-to-rail* ? Performances en hautes fréquences ? L'AOP idéal n'existe pas, définissez vos besoins, hiérarchisez-les, et enfin utilisez les superbes outils web proposés par tous les fournisseurs d'AOP pour trouver la puce la mieux adaptée à votre application. Bon, regardez le prix des composants quand même et vérifiez que le composant idéal est bien disponible chez votre revendeur favori sans avoir besoin d'en commander 10 000 pièces...
J'en ai terminé, si vous voulez aller plus loin sur ces sujets, une excellente lecture : « The Art of Electronics » de Paul Horowitz et Winfield Hill [8]. Ce livre n'est pas bon marché, mais ses 1190 pages se lisent comme un roman, y compris les quelque 200 pages traitant des amplificateurs opérationnels.
Dans le prochain article, je continuerai à vous parler d'AOP, en abordant le cas un peu pénible des charges capacitives. Alors restez à l'écoute, et à bientôt ! ⏮

Cet article a été publié dans la revue Circuit Cellar (n°303, octobre 2015).

(190223-01)

Liens

[1] Fiche technique de l'AOP UA741 (équivalent du µA741) : www.st.com/web/en/resource/technical/document/datasheet/CD00001252.pdf

[2] AOP stabilisé par découpage, TLC2652AC, 'Advanced LinCMOs Precision Chopper-stabilized Op-Amps' : www.ti.com

[3] AOP à très faible courant de polarisation, LTC6268, '500MHz Ultra-Low Bias Current FET Input Op Amp' : www.linear.com

[4] Amplificateur de mesure, AD8422, 'High Performance, Low Power, Rail-to-Rail Precision Instrumentation Amplifier' : www.analog.com

[5] AOP à plage d'entrée extralarge, LT1783, '1.25MHz, Over-The-Top Micropower, Rail-to-Rail Input and Output Op Amp in SOT-23' : www.linear.com

[6] AOP à excursion de sortie presque totale, MCP6001, '1 MHz, Low-Power Rail-to-Rail Input/Output Op Amp' : www.microchip.com

[7] AOP à produit gain ×bande très élevé, LMH5401, '8-GHz, Low-Noise, Low-Power, Fully-Differential Amplifier' : www.ti.com

[8] 'The Art of Electronics', 3rd edition, Paul Horowitz & Winfield Hill, Cambridge University Press, ISBN 978-0-521-80926-9 [disponible en français sous le titre « Traité de l'électronique analogique & numérique »]

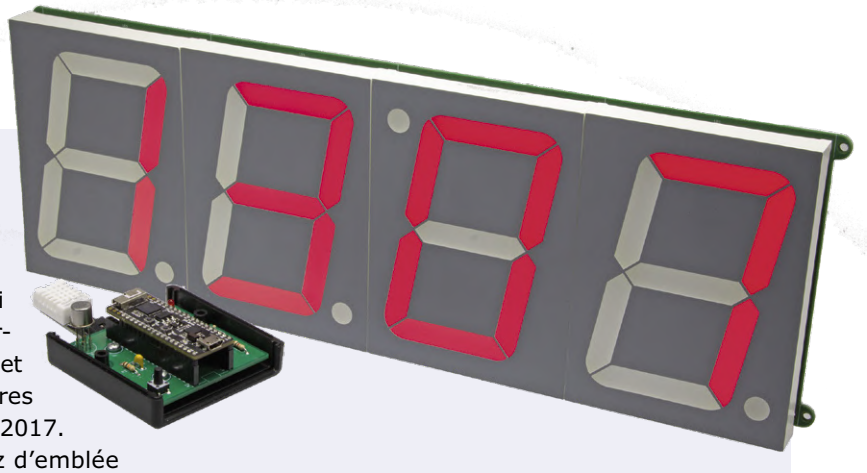
[9] Article Wikipédia, 'Invention of the integrated circuit' : https://en.wikipedia.org/wiki/Invention_of_the_integrated_circuit

[10] 'Op-Amp history', Walter G Jung : www.analog.com/library/analogDialogue/archives/39-05/Web_ChH_final.pdf

[11] Société George A. Philbrick Researches, GAP/R archive : www.philbrickarchive.org

Projet 2.0

corrections, mises à jour et courrier des lecteurs



horloge à LED géante avec Wi-Fi et mesures météo

Elektor 05-06/2019, p. 52 (180254-04)

La lecture de cet article m'a passionné. J'ai trouvé la liaison *MQTT* entre la carte capteur et l'horloge particulièrement séduisante. Cette solution permet également à ce projet de communiquer avec d'autres projets, par ex. avec la passerelle IdO de mars/avril 2017. Si vous utilisez déjà un Raspberry Pi, vous pouvez d'emblée installer *FHEM* (sigle allemand de *Freundliche Hausautomation und Energie-Messung*, c.-à-d. *système convivial de domotique et de mesure énergétique*) et exploiter ainsi les données des capteurs. Avec la passerelle IdO, j'aurais peut-être développé un capteur à faible consommation pour fonctionner sur batterie. Il serait intéressant d'afficher d'autres données comme l'arrivée de courrier, mais ce n'est qu'un exemple parmi d'autres. Il serait même possible de concevoir un affichage matriciel permettant d'afficher le texte complet. Cette horloge pourrait être à la base de toute une série de projets. Il ne fait pas de doute que vous trouverez d'autres applications.

Frank Klee

Merci pour ce retour. Effectivement, de nombreuses modifications sont dans les cartons et nous souhaitons intégrer l'échange de donnée par *MQTT* dans bien d'autres projets. En préparation : la liaison entre l'horloge et la station météo ainsi qu'une modification pour que l'horloge puisse signaler que quelqu'un a sonné à la porte. Nous envisageons aussi d'exploiter les broches E/S libres pour que l'horloge produise ses propres données ; par exemple en y connectant un capteur de luminosité ou un *DHT22* pour mesurer la température et l'humidité ambiantes. Bien sûr, ces développements ne seront pas encore prêts pour la prochaine publication.

Mathias Claußen (labo d'Elektor)

carte de commutation à relais à 9 canaux

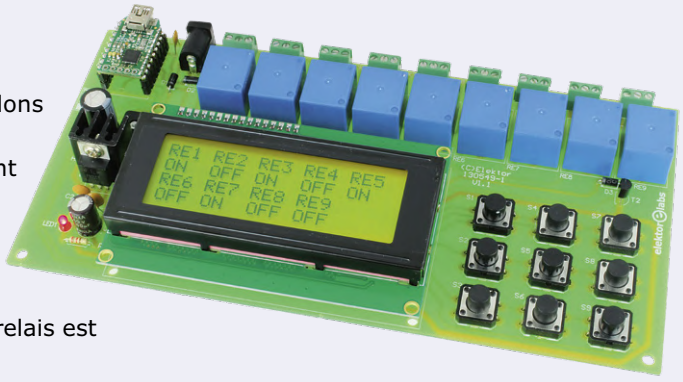
Elektor 05-06/2019, p. 70 (190027-03)

Bonjour d'Autriche !

Envisagez-vous un kit complet ? C'est exactement ce dont j'aurais besoin.

J'ai par ailleurs une réserve à formuler : les relais sont prévus pour 1200 W en CA, mais la tension de service n'est pas indiquée. J'imagine qu'il s'agit de 230 V, cependant, je trouve que la distance d'isolation (pour empêcher les arcs entre les contacts) est un peu trop faible.

Manfred Tischler



Oui, d'ici peu ce projet sera disponible en kit (nous attendons sa livraison).

La carte est prévue pour 230 V CA, les raccordements sont calculés pour 250 V CA, 7 A.

La distance d'isolation recommandée dépend de l'application et des normes auxquelles vous souhaitez répondre. Dans un milieu propre et sec, elle peut être très faible ; dans un milieu sale et humide, elle doit être très grande. La carte à relais est conçue pour travailler dans un milieu propre et sec.

Clemens Valens (labo d'Elektor)

diviseur de fréquence avec facteur de division entier réglable

Elektor 01-02/2019, p. 13 (180559-04)

thyristor MOSFET à double anode

Elektor 05-06/2019, p. 93 (190017-03)

Par erreur, seul *Michael Shustov* a été cité comme auteur de ces projets ; en fait, *Andrey Shustov* est co-auteur de cet article.

thyristor MOSFET à double anode

Elektor 05-06/2019, p. 93 (190017-03)

Dans cet article, la notion de « haute tension » n'a pas été employée à bon escient. Comme l'indique à juste titre Wikipedia (https://fr.wikipedia.org/wiki/Haute_tension), l'expression « haute tension » n'est utilisée qu'à partir de 1000 VCA. S'il fallait distinguer les différentes versions du projet en fonction de la tension de service, pour la version 40 V CA, il conviendrait d'utiliser le terme de *très basse tension (TBT)* et pour la version 230 VCA, *basse tension (BT)*.

Peter Bitzer

Notre lecteur a tout à fait raison.

Ralf Schmiedel (rédaction)

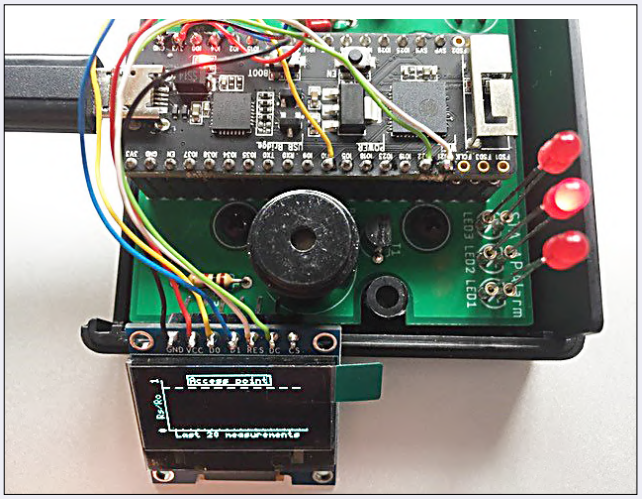
surveillance du niveau de pollution de l'air

Elektor 03-04/2019, p. 82 (170182-04)

Pour cet article, j'ai une proposition d'amélioration. Le schéma l'indique, mais le texte n'apporte aucune explication complémentaire : dans la configuration décrite et les croquis associés, les LED ne peuvent pas être utilisées en même temps que l'afficheur OLED. Mais cela n'est pas obligatoire, le kit *ESP32 Pico* dispose de largement assez de broches libres pour pouvoir piloter les deux. J'ai revu le moniteur et entrepris certaines modifications de sorte que sur le mien, je puisse utiliser les LED et l'afficheur OLED en même temps.

Comme le câblage des LED via la platine est déjà défini, il ne peut être modifié, mais il suffit d'affecter d'autres sorties de l'*ESP32* à quelques-unes des connexions de l'afficheur. Par ex. la broche DAT de l'afficheur peut être affectée à l'E/S IO4 de l'*ESP32*, le DC à IO22 et RST à IO21. Le croquis doit naturellement être modifié pour correspondre à ces affectations :

```
//pin definitions
#define gasSensorPin 35
#define btn 9
#define alarmPin 19
#define dispClk 10
#define dispDat 4
#define dispD_C 22
#define dispRst 21
#define STALed 5
#define APLed 18
#define alarmLed 23
```



Lorsque c'est fait, le mieux est de rechercher et remplacer dans le croquis les occurrences de `STALed`, `APLed` et `alarmLed`. Lorsque des instructions correspondantes sont incluses dans `#ifndef USEDISPLAY ... #endif` ou dans les branches `#else` de `#ifdef USEDISPLAY ... #endif`, il suffit de s'en débarrasser. C'est le cas quelquefois dans *tgs2600.ino* et *network.ino*.

Une fois les modifications effectuées, vous pourrez apprécier ce fonctionnement simultané de l'afficheur avec des LED qui indiqueront correctement l'état de la connexion au réseau et de l'alarme.

Les connexions peuvent être simplement soudées sur l'*ESP*.

Hans Schneider

Merci beaucoup pour cette excellente proposition d'amélioration.

Ralf Schmiedel (rédaction)

(180573-D-04 – version française : Yves Georges)

équiper sa paillasse

...à petit prix

Une fois que vous aurez contracté le virus de l'électronique, vous voudrez équiper votre labo d'appareils de test et de mesure. Pour vous lancer, pas besoin d'acheter du matériel de professionnel très onéreux ! Voici une petite sélection à prix tout doux.

Oscilloscope OWON XDS3064E à 4 voies avec écran tactile

La gamme d'oscilloscopes XDS de la marque chinoise OWON offre un certain nombre de fonctions dont sont (jusqu'ici) dépourvus les modèles concurrents comparables, et ce, à un prix très optimisé. Examinons l'oscilloscope à 4 voies XDS3064E, qui offre de généreuses capacités pour moins de 400 €.

Dès le déballage, l'appareil capte immédiatement l'attention par son grand dispositif d'affichage, un écran LCD de 8 pouces d'une résolution de 800×600 pixels, doté de fonctions tactiles. C'est une première pour moi dans cette gamme de prix. Une fois l'instrument mis en service, cette première impression ne se dément pas : un écran généreux, doté d'un angle étendu de visualisation ! Cet appareil à 4 voies dispose d'un écran tactile, il est doté d'une bande passante d'entrée de 60 MHz et d'une fréquence d'échantillonnage de 1 Géch/s. Il est vrai que cette fréquence est divisée par 2, voire 4 si vous utilisez plusieurs canaux, mais c'est courant dans cette gamme de prix. L'instrument est fourni avec quatre sondes, un câble d'alimentation secteur, un câble USB et un manuel de prise en main succinct. Il existe également un manuel complet au format PDF.

Se familiariser avec le fonctionnement de l'appareil ne demande que peu de temps. Cependant, certaines fonctions nécessitent quelques explications supplémentaires et il n'est pas inutile, dans ce cas, de consulter le manuel complet. Il est possible de régler la sensibilité des voies d'entrée jusqu'à 1 mV/div (bande passante limitée). Le bruit des étages d'entrée reste assez faible, car dans le cas contraire, une telle sensibilité n'aurait aucun sens. Le bouton *Auto-set* (réglage automatique) permet de rechercher automatiquement les bons paramètres pour un affichage stable du signal d'entrée. Le bouton *Math* permet non seulement d'appliquer des opérations aux signaux, mais également d'entrer des équations. Le XDS3064E possède également une fonction FFT (transformation de Fourier rapide).

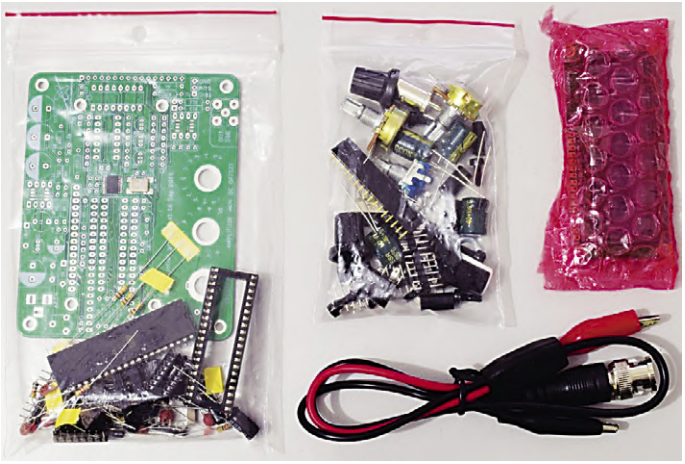
En conclusion, si une largeur de bande d'entrée de 60 MHz vous suffit, cet instrument est idéal pour vous.

@ WWW.ELEKTOR.FR

→ oscilloscope OWON XDS3064E à 4 voies
www.elektor.fr/18829

Générateur de fonctions en kit de Pulsar Labs

Un oscilloscope (tel que le modèle OWON décrit ci-dessus) est indispensable pour voir ce que se passe dans un circuit électronique, mais un tel instrument devient vraiment utile lorsque l'on peut appliquer un signal bien défini à l'entrée du circuit. Et c'est là qu'un générateur de fonctions entre en jeu. Ici, nous examinons un générateur de fonctions à monter soi-même : le générateur de fonctions à code source ouvert de Pulsar Labs. L'assemblage du générateur de fonctions (bien qu'il ne soit pas explicitement décrit dans le manuel fourni) devrait être simple, même pour ceux qui ne sont pas encore des experts en soudure. Le plus important est de prendre son temps, de vérifier soigneusement comment les composants doivent être



montés et, en cas de doute, d'essayer de les monter en premier. Les deux seuls composants CMS sont déjà câblés sur la carte, il n'y a donc pas lieu de s'en inquiéter. Une fois que le circuit imprimé a été entièrement assemblé et vérifié (conseil : en cas de doute, laissez quelqu'un d'autre y jeter un coup d'œil), la tension d'alimentation peut être appliquée et le générateur peut être essayé.

Le générateur peut produire les trois formes d'onde les plus importantes (sinusoïdale, carrée et triangulaire) et pour ce prix (pas tout à fait 60 €), il présente une gamme de fréquences respectable de 1 Hz à 10 MHz. Le mode Balayage (*sweep*) est extrêmement pratique pour déterminer rapidement le comportement d'un circuit sur une large gamme de fréquences.

La tension de sortie maximale est d'environ 10 V ; la plage d'offset varie de −5 V à +5 V. La distorsion de la sinusoïde de sortie à 0,7 V et 1 kHz est d'environ 0,1% – pas si mauvaise que ça. Dans l'ensemble, il s'agit d'un générateur très agréable à utiliser qui trouverait toute sa place sur la paillasse d'un débutant.

Impression à main levée avec le kit 3D-Pen Simo

Est-ce pour bricoler ou apprendre ? Avec le kit Simo 3D, ce sont les deux ! Quand vous entendez parler d'impression 3D, vous pensez sans doute d'abord à un appareil assez encombrant. Et à quelque chose d'assez bruyant. Pas à un crayon avec lequel vous dessineriez en trois dimensions et en couleurs ! Depuis quelque temps, un accessoire d'impression 3D de petite taille a fait son apparition sur les établis des bricoleurs. Il fait penser au classique pistolet à colle thermofusible.

Cet outil électrique est à l'imprimante 3D normale ce que la perceuse de poing est à la machine-outil montée sur colonne. En plus silencieux ! Autrement dit, un extrudeur manuel de filament PLA, une espèce de crayon qui permet de dessiner ou d'imprimer à la main, en 3D.

Conçu comme un outil de table, léger et facile à manipuler, il s'adresse aux adeptes de l'impression 3D qui souhaitent mettre la main à la pâte, au propre et au figuré. C'est à la fois un outil de retouche et de finition, et un outil de création à main levée. Sa taille est de 138×25×21 mm, il ne pèse que quelques dizaines de grammes. Il est alimenté sous 5 V via un câble USB capable de fournir 1,5 A.

Le modèle proposé ici est aussi un kit à assembler soi-même. La carte principale est un Arduino Nano. Ce kit 3D Simo est libre de droits : toutes ses données de conception sont disponibles gratuitement sur GitHub (fichiers du circuit imprimé,

pièces imprimées en 3D et même le logiciel). Autrement dit, tout est modifiable et personnalisable à volonté.

Ce kit est non seulement un crayon pour dessiner en l'air, mais aussi un objet d'étude et d'expérimentation personnelle. Il est donc très intéressant l'enseignement. ⏮

180689-D-04

@ WWW.ELEKTOR.FR

→ générateur de fonctions en kit de Pulsar Labs
www.elektor.fr/18653

@ WWW.ELEKTOR.FR

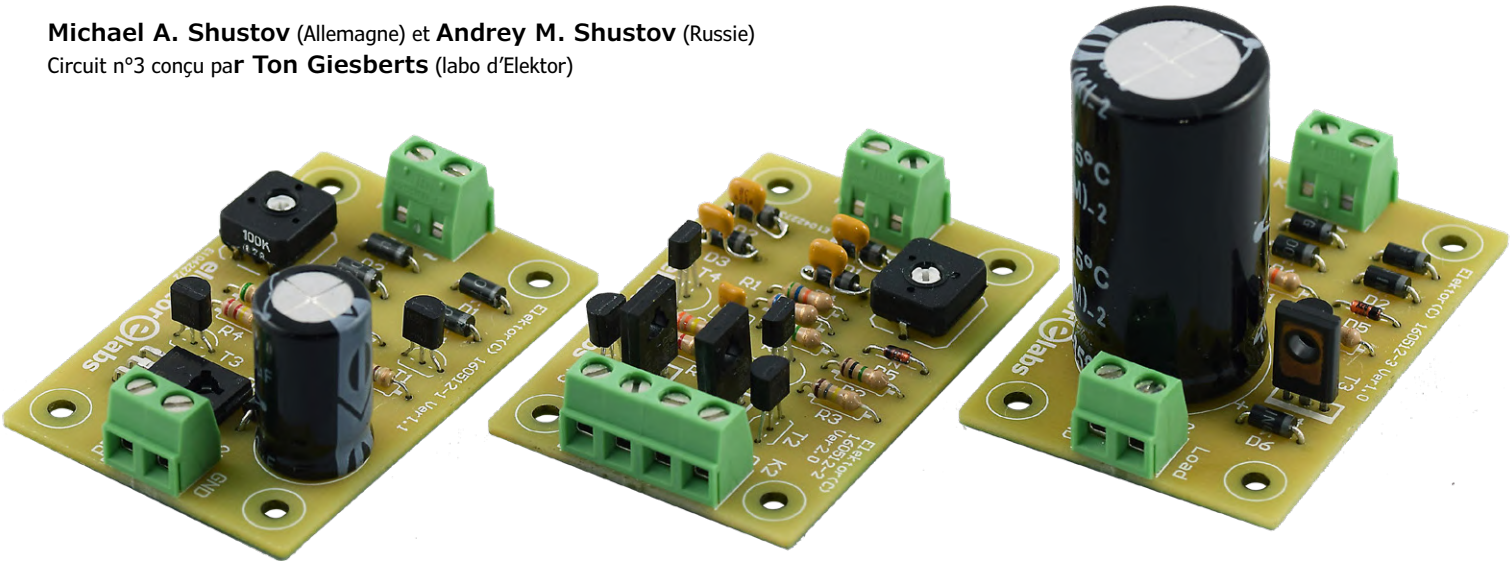
→ kit 3D-Pen Simo
www.elektor.fr/18867

PROJET DU LABO

baristors de l'alternatif au continu

modules expérimentaux pour alimentations

Michael A. Shustov (Allemagne) et **Andrey M. Shustov** (Russie)
Circuit n°3 conçu par **Ton Giesberts** (labo d'Elektor)



Dans l'électronique industrielle et grand public, les thyristors et triacs sont connus pour altérer la tension du réseau alternatif en raison des harmoniques, du bruit, des salves et des transitoires qui se superposent à ce que vous pensiez être une onde sinusoïdale propre issue de la prise de courant. Un dispositif électronique composite récemment expérimenté, appelé baristor, peut aider à maintenir saine la tension du secteur tout en offrant de bonnes propriétés de régulation. Cet article décrit le fonctionnement du baristor ainsi que trois circuits imprimés pratiques pour l'expérimentation : entrée~, sortie= (AC in, DC out).

Le module barrière-résistance (*barrier* + *resistor* = baristor) est un élément de commutation pour l'électronique de puissance et l'analogique dont la résistance électrique d'entrée à sortie passe brusquement de conduction à blocage et inversement. L'état souhaité est obtenu quand une valeur de seuil déterminée (la barrière) du signal d'entrée est atteinte. Les symbaristors sont des baristors symétriques destinés à fonctionner en courant alternatif.

Baristor contre SCR

Dans l'électronique de puissance, les thyristors et les triacs sont des commutateurs capables de passer de l'état non conducteur à la conduction lorsque leur électrode de gâchette est activée. Ils s'éteignent au moment où leur courant anodique approche de zéro. Les thyristors et les triacs permettent d'amputer une partie du signal sinusoïdal pour contrôler la puissance effective fournie à la charge. Mais ils ont certains inconvénients :

- faibles performances de commutation ;
- pas de mise hors tension sans interruption du courant à travers l'appareil ;
- distorsion de l'onde sinusoïdale ;
- réduction du facteur de puissance ;
- faible rendement économique.

On peut largement surmonter ces inconvénients avec les baristors et les symbaristors qui :

- permettent de diviser une tension alternative en segments de largeur réglable et de les utiliser en fonction de ses besoins ;
- permettent de réguler ou de stabiliser la consommation électrique des appareils de chauffage et d'éclairage ;
- sont capables de produire différentes tensions de sortie sur des blocs d'alimentation.

Pour simplifier, la commutation des thyristors et triacs s'exerce sur l'axe horizontal, celui du temps, alors qu'avec les baristors, elle a lieu sur l'axe vertical, donc en tension. Cette caractéristique du baristor ouvre de nouvelles possibilités dans les appareils de communication et l'électronique de puissance. Les baristors sont destinés à séparer les signaux dont l'amplitude est supérieure ou inférieure à une valeur déterminée (la barrière) fixée par l'utilisateur.

Types de baristors et applications

On distingue les baristors en classes :

- contrôlée et non contrôlée (c'est-à-dire avec une valeur de seuil contrô-

lée ou pas) ;
- alternatifs ou continus (baristors asymétriques et symétriques ou symbaristors) ;
- baristor monocanal ou multi-canal (baristors multi-niveaux, multi-seuils) ;
- interrupteurs de type marche/arrêt ou commutateur.

Voici leurs noms avec la photo de famille.
Baristor de niveau haut : pas de tension à la sortie avant que la tension d'entrée dépasse une certaine valeur.
Baristor de niveau bas : la tension à sa sortie est présente tant que la tension d'entrée dépasse un seuil donné. Ensuite, le baristor change d'état et coupe la charge.
Commutation d'un baristor à seuil unique : lors d'une montée progressive de la tension d'entrée, elle se retrouve d'abord à la sortie de signal de bas niveau. Quand le seuil fixé est dépassé, le signal d'entrée est automatiquement commuté sur la sortie à haut niveau.
Commutateur à baristor à plusieurs seuils : en augmentant/diminuant continuellement la tension d'entrée, la commutation de la sortie du baristor est séquentielle et le signal d'entrée est ainsi transmis à la sortie du baristor. Le signal d'entrée en cours de passage indique la plage de tension correspondante.

L'utilisation de baristors et de symbaristors permet de distribuer ou de redistribuer la puissance alternative entre les consommateurs d'énergie électrique, de supprimer les harmoniques supérieures de la tension du secteur, d'augmenter (c'est-à-dire corriger) le facteur de puissance, d'utiliser rationnellement une partie de l'onde sinusoïdale que les SCR « balaient ». Toutes ces caractéristiques devraient contribuer à accroître sensiblement l'efficacité de l'utilisation de l'énergie électrique, ainsi qu'à améliorer l'efficacité économique. Les éléments de barrière-résistance peuvent être utilisés dans les domaines suivants :
- alimentations économiques miniatures à faibles pertes ; convertisseurs de tension et stabilisateurs de tension ;
- protection des composants et circuits électroniques ainsi que des appareils électriques et des lignes de communication ;
- systèmes de télécommande et de communication multicanaux comprenant un signal HF superposé sur une ligne bifilaire ;
- instruments de mesure et transducteurs ;
- appareils à impulsions ;
- isolation, séparation ou production de signaux ;
- électronique analogique pour la séparation des signaux à l'aide d'une valeur de seuil définie par l'utilisateur.

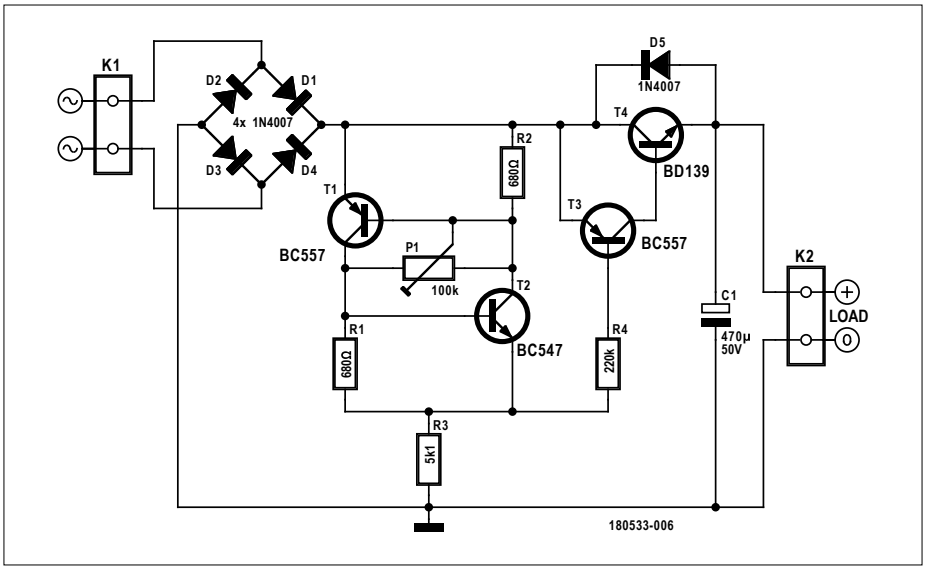


Figure 1. Version de base d'un baristor à seuil unique conçu à partir des explications trouvées dans l'ouvrage « Electronic Circuits for All ».

Performances du baristor à seuil unique

Référence : circuit de la figure 1.
Quelques résultats de mesure avec charge de 150 Ω et réglage fixe de P1

entrée en alternatif	sortie en continu
12 V	11,4 V
16 V	9,9 V
20 V	9 V
24 V	8,6 V

Avec 20 V~ à l'entrée, la tension de sortie descend de 9 V à 6,7 V quand la charge passe de 150 Ω à 75 Ω.
À certaines tensions d'entrée et pour divers réglages de P1, la tension de sortie fluctue dans le temps. Le courant à vide peut être inférieur à 1 mA, mais aussi à quelques mA, selon le réglage de P1, la tension d'entrée et en absence de charge.

Si vous souhaitez en savoir plus sur les origines du baristor et la théorie de son fonctionnement, vous pouvez vous procurer un exemplaire du livre des auteurs « *Electronic Circuits for All* » publié par Elektor, les détails sont dans l'encadré @www.elektor.fr.

Baristor n°1, la version élémentaire

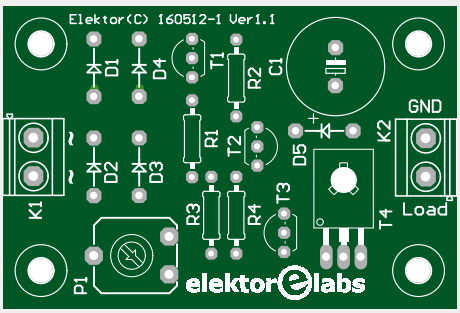
Passons aux choses concrètes. La **figure 1** montre le schéma d'un baristor à seuil unique, après retouches de Ton Giesberts du labo d'Elektor, sur une version trouvée dans le livre mentionné ci-dessus.
Tant que la tension à la sortie du pont redresseur D1 à D4 (dans ce cas, on utilise environ 16 V~) reste inférieure au seuil réglé par T1/T2, le double transistor T3/T4 est conducteur. Le niveau de tension réel de D1 à D4, moins la chute de tension sur T4, est stocké dans le condensateur tampon C1. Si la tension d'entrée dépasse le seuil réglé, T4 se bloque et la tension de sortie n'augmente plus. Remplacer P1 par un photocoupleur ou un JFET permet de faire piloter la tension de sortie par un autre circuit. Si la tension alternative appliquée est supérieure à 16 V, il faut remplacer les résistances R3 et R4 en conséquence. En fonction des valeurs de R3 et R4 et d'autres conditions, la caractéristique courant-tension du circuit autour de T1/T2 varie de celle d'un thyristor à celle d'une diode Zener. L'ennui, c'est que ce n'est pas vraiment une diode Zener.

LISTE DES COMPOSANTS

Baristor à seuil unique /circuit imprimé 160512-1 / circuit de la figure 1

Résistances
R1, R2 = 680 Ω 5 %, 0.25 W
R3 = 5,1 kΩ 1 %, 0,25 W
R4 = 220 kΩ 5 %, 0,25 W
P1 = potentiomètre ajustable 100 kΩ 20 %, 0,15 W, CB10LV104M Citec-TE Connectivity

Condensateur
C1 = 470 µF 20 %, 50 V, diam. 13 mm, au pas de 5 mm



Semi-conducteurs
D1, D2, D3, D4, D5 = 1N4007
T1, T3 = BC557B
T2 = BC547B
T4 = BD139

Divers
K1, K2 = bornier à 2 vis encartable, au pas de 5,08 mm, p.ex. Phoenix Contact MKDSN 1,5/ 2-5,08
Circuit imprimé, réf. 160512-1 (www.elektor.fr)

Si T1 et T2 sont bloqués, le courant circule toujours dans R2-P1-R1-R3. Ainsi, si P1 est de trop faible valeur ou R3 trop élevée, T3 peut être conducteur ou bloqué, c'est-à-dire qu'il n'y a pas de seuil réel et que la tension de sortie varie avec la tension alternative d'entrée. De plus, une instabilité de très basse fréquence peut se produire à certains réglages de P1 en combinaison avec différentes tensions d'entrée. Pire encore, l'impédance de sortie du transformateur affecte le fonctionnement et la stabilité du circuit. Un condensateur de 10 µF entre la base de T4 et la masse peut améliorer la stabilité du circuit. Une modification de la résistance de charge change la tension de sortie.
Quelques résultats de mesure probants, obtenus avec un prototype du circuit sont donnés dans l'encadré : **Performances du baristor à seuil unique**.

Baristor à seuil unique amélioré

Le circuit de la **figure 2** est aussi basé sur un schéma du livre « Electronic Circuits for All ».
Ce n'est pas un montage entièrement fonctionnel, mais plutôt un circuit qui peut être modifié pour s'adapter à des applications personnelles, simplement donner de nouvelles idées ou faire partie d'une solution pour d'autres buts. Bien sûr, le circuit imprimé conçu par le labo d'Elektor pour ce circuit facilite les tests et les expériences, mais le circuit est assez simple pour être monté sur une plaque d'essai.
Un réseau de rétroaction positive R7 et C5 a été ajouté pour supprimer les erreurs et améliorer la commutation des étages de sortie. Ceci fait que la base de T1 agit comme l'entrée d'un *trigger* de Schmitt. En raison de l'hystérésis, qui dépend de la tension alternative appliquée à K1, la phase réglée sur P1 change avec le niveau de la tension d'entrée. En fonction de l'application et de la tension alternative utilisée, les valeurs de certains composants sont sujettes à des modifications, pour lesquelles un logiciel de simulation tel que LTSpice est utile. L'utilisation d'une source de tension au lieu d'un vrai transformateur dans la simulation donnera des résultats faux. Chaque transformateur a une impédance de sortie complexe, composée principalement de la combinaison d'une résistance et d'une inductance. Par conséquent, au moment où le baristor commute, la tension d'entrée change et peut causer de l'instabilité, surtout lorsque le gain de T1 est élevé.
Le circuit original du livre utilisait des transistors Darlington haute tension de type BC373 qui se sont malheureusement révélés difficiles à trouver. Les spécifications sont les suivantes : 80 V/1 A dans un boîtier de type TO92. Le paramètre h_{FE} à 100 mA est >20 000. Heureusement, il existe encore plusieurs types de remplacement disponibles, dont le BCX38C (60 V/0,8 A), le KSP13B (30 V/0,5 A), le MPSA14 (30 V/0,5 A), le MPSA14 (30 V/0,8 A).5 A) ; MPSA29 (100 V/0,5 A) ; ZTX603 (80 V/1 A) ; ZTX605 (120 V/1 A) ; ZTX614 (100 V/0,8 A) ; BC517-D74Z (30 V/1,2 A) mais avec les connexions de collecteur et d'émetteur permutées !
Attention à la dissipation de puissance limitée (0,6 W) du boîtier TO92, préférez-en un plus gros. En fin de compte, nous avons utilisé le bon vieux BD139 en combinaison avec un BC547C standard pour remplacer le Darlington BC373 original par des transistors discrets. Le BD139 a un boîtier de type SOT32 (TO126) et permet une dissipation de puissance maximale de 1,25 W sans refroidissement supplémentaire (c'est-à-dire que la jonction sera de 150 °C à température ambiante de 25 °C). Ne faites jamais fonctionner un transistor à une température de jonction de 150 °C, maintenez-le bien en dessous de 70 °C si vous ne voulez pas réduire sa durée de vie. Et pourquoi ne pas utiliser un BD679 Darlington pour réduire le nombre de pièces ?
Presque tous les Darlington de puissance ont des résistances internes en parallèle avec la jonction base-émetteur des deux transistors. Celles-ci font inévitablement varier le gain du transistor en fonction du courant de collecteur, alors qu'à faible courant, le gain est beaucoup plus faible que celui des Darlington sans résistances internes.
D'après le schéma de câblage, la chute de tension maximale entre K1 et K2 (ou K3) est d'environ 4 V à 100 mA. Ceci est dû aux diodes D1 à D4, à une faible chute de tension entre R4 et R6 et aux tensions de base des émetteurs T2/T3 et T5/T6. Ainsi, sans autre refroidissement de T3/T6, le courant de sortie maximal devrait être limité à quelques centaines de milliampères.
La suppression de D5 augmente la plage de P1 et a plus d'effet à basse tension d'entrée. Pour changer la plage de P1 à des tensions d'entrée élevées, pensez à modifier aussi R1 et R3, et peut-être la tension de fonctionnement

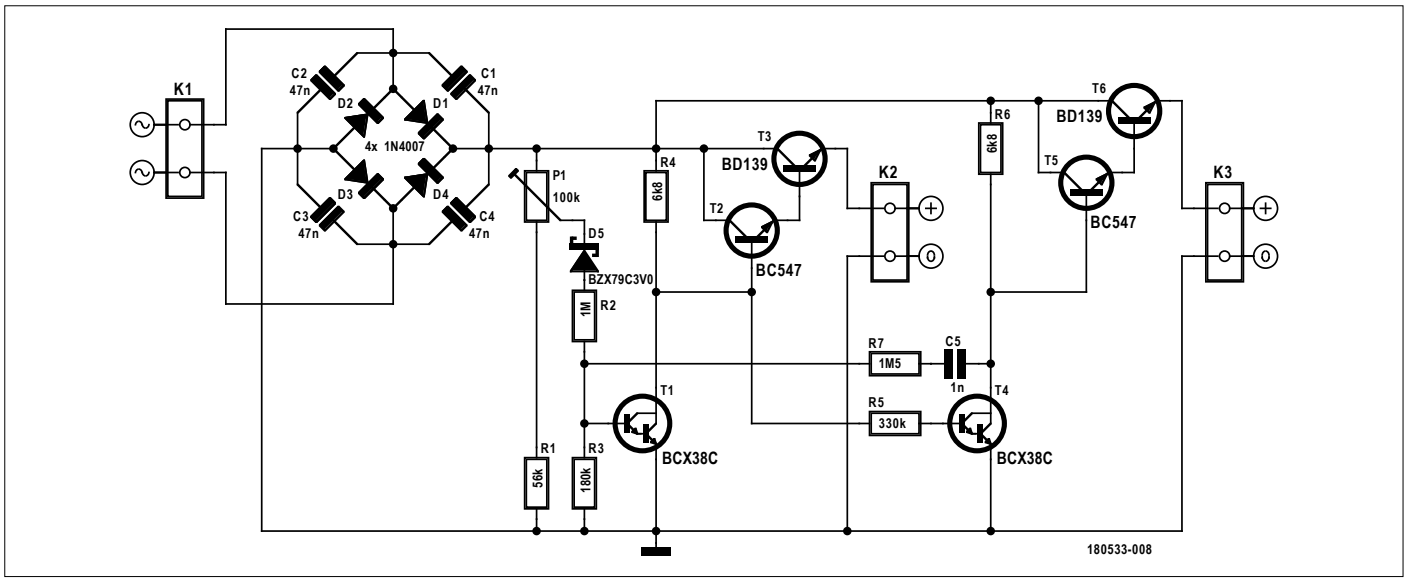


Figure 2. Cette version du baristor comporte des améliorations et des perfectionnements par rapport à la version de base. Elle offre de nombreuses possibilités d'expérimentation et d'adaptation.

Performances du baristor à seuil unique, version améliorée

Référence : circuit de la figure 2.
Un exemple d'angles de phase maximaux pour des réglages minimal et maximal de P1 (18 V sur K1) :

- P1 à fond à droite : 29° à 148°
- P2 à fond à gauche : 79° à 126°
- Courant de repos : 1,71 mA (12 V sur K1)
- Courant de repos : 3,75 mA (24 V sur K1)

Avec impédances de charge égales sur K2 et K3 :

- Courant d'entrée : (*U*in – 3,2 V) / *Rcharge,* mesuré *à* 300 mA (K1) et charges de 75 Ω.
- Courant d'entrée : (*U*in – 3 V) / *Rcharge,* mesuré *à 12*0 mA (K1) et charges de 75 Ω.

Avec deux charges identiques connectées à K2 et K3, le courant d'entrée est virtuellement indépendant du réglage de P1 et ne varie que légèrement, quelques %, en raison des tolérances des composants.

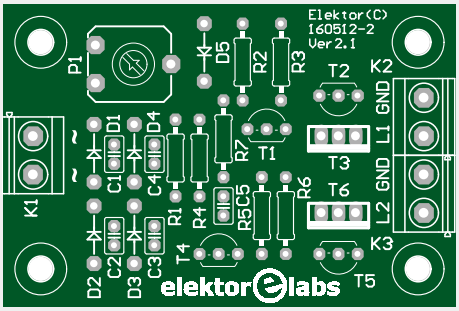
de D5. Mais en raison du faible courant traversant D5, la chute de tension à ses bornes est inférieure à la tension nominale de fonctionnement.
Nous avons réalisé un nouveau circuit au labo et les résultats avec commentaires sont dans l'encadré **Performances du baristor à seuil unique, version améliorée**.
Notre configuration de test partait d'un autotransformateur variable (Variac). Il est utilisé pour faire varier la tension primaire d'un transformateur de sécurité de type FL30/12 de Block. Il possède deux enroulements primaires de 115 V et

LISTE DES COMPOSANTS

Baristor à seuil unique,
version améliorée / circuit imprimé
réf. 160512-2 / circuit de la figure 2

Résistances
R1 = 56 kΩ 5 %, 0,25 W
R2 = 1MΩ 5 %, 0,25 W
R3 = 180 kΩ 5 %, 0,25 W
R4, R6 = 6,8 kΩ 5 %, 0,25 W
R5 = 330 kΩ 5 %, 0,25 W
R7 = 1,5MΩ 5 %, 0,25 W, 250 V
P1 = potentiomètre ajustable 100 kΩ

Condensateurs
C1, C2, C3, C4 = 47nF, 50 V, céramique X7R, au pas de 2,54 mm



C5 = 1nF, 100 V, céramique X7R, au pas de 2,54 mm

Semi-conducteurs
D1, D2, D3, D4 = 1N4007
D5 = BZX79-C3V0, 3 V, 500 mW
T1, T4 = BCX38C, Darlington, 60 V, 0,8 A

T2, T5 = BC547C
T3, T6 = BD139

Divers
K1, K2, K3 = K1, K2 = bornier à 2 vis encartable, au pas de 5,08 mm, MKDSN 1,5/ 2-5,08
Circuit imprimé, réf. 160512-2 (www.elektor.fr)

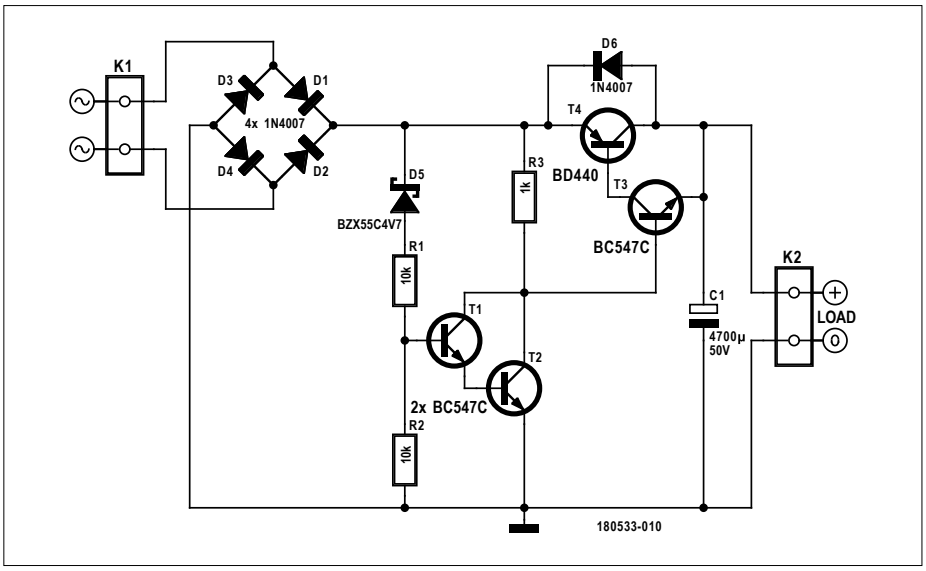


Figure 3. Résultat de la recherche d'un compromis, cette version du baristor allie un plus petit nombre de composants à une meilleure stabilisation.

deux enroulements secondaires de 12 V capables de fournir 1,25 A chacun. Une façon de garantir une installation de test sûre. Mais soyez très prudent dans cette configuration. La tension de sortie maximale de l'autotransformateur est généralement supérieure à la tension d'entrée. Le labo en utilisait un qui est toujours spécifié pour l'ancienne tension du secteur de 220 V. La sortie est spécifiée à 250 V/2 A max. À l'heure actuelle, avec une tension du secteur nominale de 230 V, la tension de sortie à vide sera encore plus élevée, comme 261 V. Mais c'est encore pire lorsque l'on mesure la tension de sortie réelle, on a trouvé un énorme 274 V alors que le secteur atteignant 239 V sur le lieu du test. Dans tous les cas, ne réglez jamais l'autotransformateur à sa tension maximale dans ce montage d'essai. Cela détruirait très certainement le transformateur d'isolement qui lui est raccordé.

Dans un premier test, on a utilisé un petit transformateur et le duo R7/C5 a suffi à se débarrasser des commutations intempestives. Mais la configuration de test décrite ci-dessus n'était pas suffisante. Les quatre diodes du redresseur ont dû être découplées par quatre condensateurs antiparasites de 47 nF.

Nombre de composants en baisse, stabilisation à la hausse

Pour donner une idée des améliorations supplémentaires simples, on a conçu un petit circuit baristor pour produire une tension de sortie continue moins dépendante de la charge et de la tension alternative d'entrée tout en utilisant un minimum de composants. Le principe est simple. L'étage de sortie agit comme un interrupteur et doit être ouvert dès que la tension d'entrée dépasse un certain seuil. Ce dernier est réglé principalement par la diode Zener D5. Ici, R1 limite le courant pour des tensions d'entrée plus élevées tandis que R2 assure la circulation d'une partie du courant dans la Zener avant que le Darlington discret T1/T2 ne conduise. T1/T2 attirent la base du transistor composite T3/T4 vers la masse, coupant ainsi l'interrupteur de sortie. Pour minimiser la chute sur T4, on utilise la combinaison d'un transistor petit signal NPN et d'un transistor de puissance PNP, économisant ainsi quelques dixièmes de volts.

Pour réduire encore la chute à travers R3, due au courant de base, la valeur de R3 n'est que de 1 kΩ. Cela signifie qu'une résistance de 0,25 W dissipera toute la puissance dont elle est capable à une tension d'entrée de 21 V. Toute mesure avec des tensions d'entrée supérieures à 21 V dépassera les spécifications de sécurité pour R3. Dans ce cas, utilisez une résistance de 0,5 ou 0,6 W au lieu de 0,25 W. Il y a aussi des types

LISTE DES COMPOSANTS

Baristor aminci / circuit imprimé 160512-3 / circuit de la figure 3

Résistances

R1, R2 = 10 kΩ 5 %, 0,25 W
R3 = 1 kΩ 5 %, 0,25 W

Condensateur

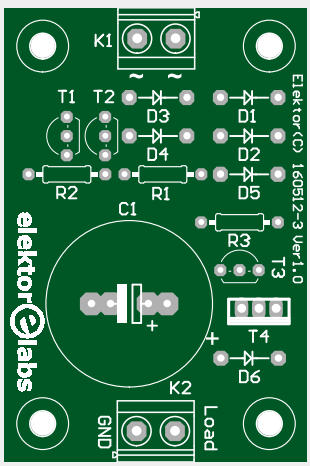
C1 = 4 700 µF 20 %, 50 V, diam. 22 mm max., au pas de 5/7,5/10 mm

Semi-conducteurs

D1, D2, D3, D4, D6 = 1N4007
D5 = BZX55C4V7, 4,7 V, 500 mW
T1, T2, T3 = BC547C
T4 = BD440 (PNP, 60 V, 4 A, 7 A pulse, TO-126)

Divers

K1, K2 = bornier à 2 vis encartable, au pas de 5,08 mm, 630 V
Circuit imprimé, réf. 160512-3 (www.elektor.fr)



avec une spécification de 1 W. Montez ces résistances légèrement au-dessus du circuit imprimé. La dissipation sera la plus grande sans charge.
Puisque le couple T3/T4 est inactif en permanence, la tension sur R3 vaut presque toute la tension d'entrée, moins la tension de déchet sur le pont redresseur. Le fait que T1/T2 conduise au-dessus d'une certaine tension ne change pas grand-chose. Avec une Zener de 4,7 V (0,5 W), la tension de sortie est d'environ 4,4 V pour une charge de 150 Ω. Une charge de 10 Ω réduit la tension de sortie à un peu moins de 4 V. La tension à vide est d'environ 4,67 V. À l'origine, nous utilisions un bon vieux BD140 qui, à faible courant de sortie, fonctionnait bien. Mais avec une charge plus élevée, comme la résistance de 10 Ω, il s'est produit presque immédiatement un court-circuit. Notre dispositif d'essai est capable de fournir un courant nominal efficace de 1,25 A, donc encore plus en pointe. Un BD140 peut supporter 1,5 A de courant de crête, ce qui n'est pas suffisant, surtout avec un gros condensateur à la sortie. Il a été remplacé par un BD440 d'un ancien stock, mais ce type est toujours disponible (BD440S, Mouser # 512-BD440S). Le BD440 est un transistor de puissance PNP 60 V/4 A (impulsions 7 A). Il y a d'autres candidats convenables en TO-126 (SOT-32) pour remplacer T4, comme 2N5195 (PNP, 80 V, 4 A), BD438 (45 V, 4 A) ou BD442G (80 V, 4 A). Selon le fabricant, l'écartement des pattes peut différer quelque peu, on rencontre 2,39 mm (TO-225AA) au lieu de 2,29 mm. De plus, le boîtier du transistor peut être un peu plus grand comme 14,2×8 mm au lieu des dimensions normales du TO-126 de 11,1×7,8 mm. Heureusement, tous ces composants devraient encore tenir sur le circuit imprimé. L'encombrement sur carte du (gros) condensateur tampon C1 est prévu pour des pas de 5, 7,5 et 10 mm et un diamètre de boîtier au maximum de 22 mm (pas de broches à encliqueter, juste des fils droits).
Il est possible de régler la tension de sortie en remplaçant R2 par un potentiomètre. Mais alors, à des tensions plus élevées, elle dépend de plus en plus de la tension d'entrée. La tension aux bornes de R1 augmentera davantage avec une valeur inférieure de P1 et deviendra plus dominante que la tension de Zener. Pour régler une tension de sortie différente, il est préférable de changer de tension de Zener. Si des surtensions surviennent au niveau du pont redresseur, elles peuvent être réduites en insérant une diode entre D5 et R3.
Encore une fois, nous avons testé le circuit dans le labo d'Elektor, retrouvez certains des résultats dans l'encadré : **Performances du baristor aminci.**

À vos fers à souder !

Bien que l'on ait constaté que les trois circuits fonctionnent, ils restent expérimentaux et ne doivent pas être considérés comme des projets prêts à l'emploi ou destinés tels quels à des applications industrielles. Rappelez-vous qu'ils ne sont pas limités en courant ni protégés contre la surcharge. De plus, la température du boîtier des transistors de puissance de sortie doit être surveillée à des charges et des tensions d'entrée plus élevées. Misez sur la sécurité et fixez une petite plaque d'aluminium ou quelque chose du genre au boîtier du ou des transistors de puissance pour le refroidissement. ⏮

(180533-04 – version française : Robert Grignard)

@ WWW.ELEKTOR.FR

- **Livre en anglais, 'Electronic Circuits for All', M.A. Shustov et A.M Shustov**
  www.elektor.fr/electronic-circuits-for-all
- **Baristor à seuil unique, circuit imprimé nu, réf. 160512-1**
  www.elektor.fr/160512-1
- **Baristor à seuil unique, version améliorée, circuit imprimé nu, réf. 160512-2**
  www.elektor.fr/160512-2
- **Baristor aminci, circuit imprimé nu, réf. 160512-3**
  www.elektor.fr/160512-3

Performances du baristor aminci

Référence : circuit de la figure 3.
Résultats des mesures avec
T4 = BD440 et charge de 150 Ω :

U_{in} [V]	I_{in} [mA]	U_{out} [V]	U_{ripple} [mV]
AC	AC	DC	AC_{pp}
5	44	4,28	24
6	64	4,48	56
7	74	4,46	60
8	80	4,44	60
9	85	4,43	60
10	89	4,42	60
11	92	4,42	60
12	94	4,41	60
13	96	4,41	60
14	98	4,40	60
15	101	4,39	60
16	102	4,38	60
17	104	4,37	60
18	106	4,36	60
19	109	4,36	60
20	110	4,35	60
21	110	4,34	60
22	111	4,33	60
23	112	4,33	60
24	113	4,32	60

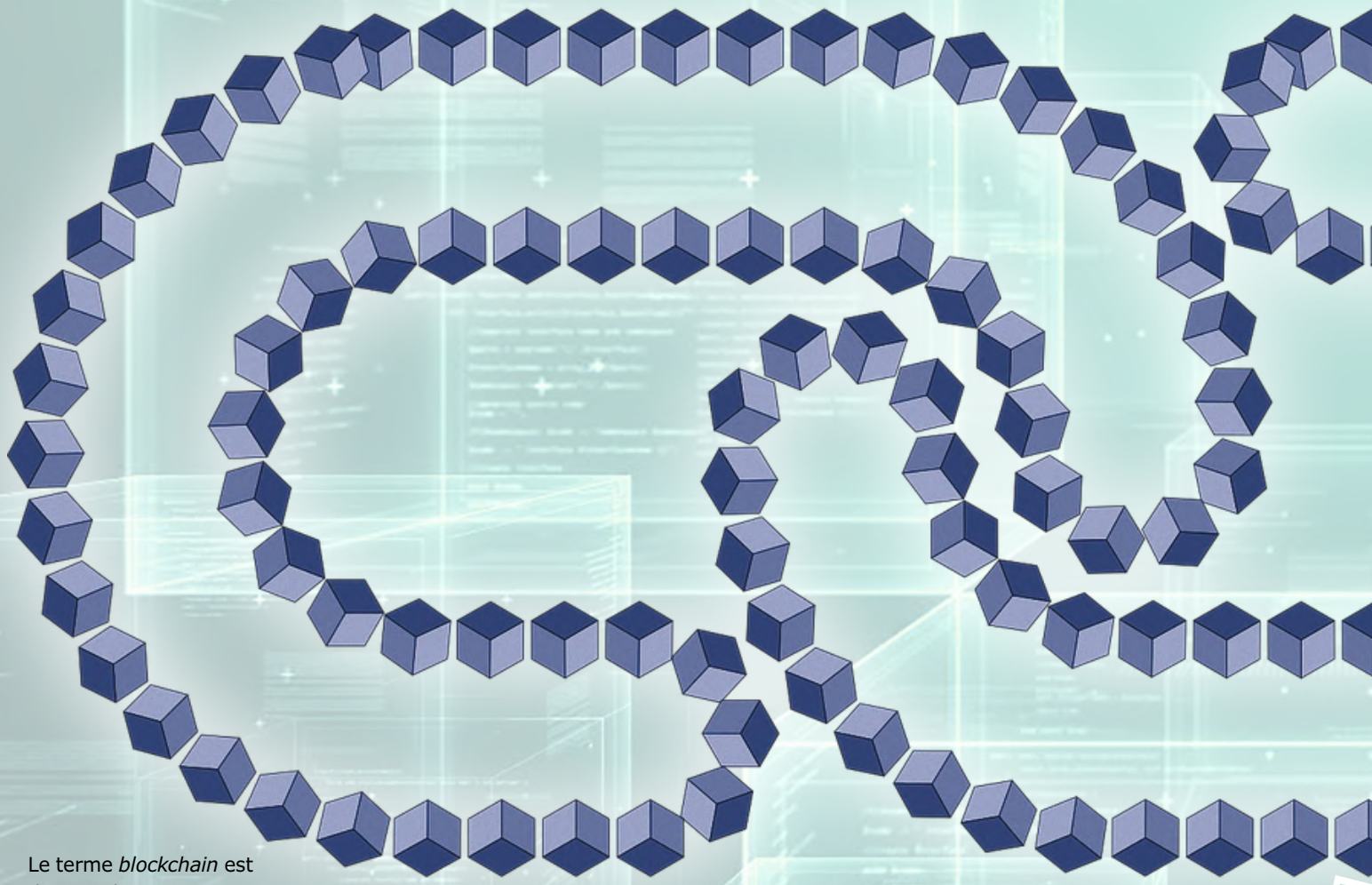
À faible charge, la majeure partie du courant d'entrée est due à R3 qui est tirée à la masse par T1/T2. Pour une utilisation avec de très faibles charges, pensez à augmenter la valeur de R3. La chute aux bornes de T4 pourrait augmenter, mais elle peut aussi accroître l'efficacité globale. Une chose à expérimenter.
Un autre essai rapide avec une charge beaucoup plus élevée, de 10 Ω a également été effectué. Sur 12 V, le courant d'entrée pour une charge de 10 Ω est égal à 0,57 A. La tension de sortie était de 3,99 V. Mais, sans refroidissement, T4 a fort chauffé ! L'efficacité n'est pas grande, mais elle est plus complexe que la multiplication des tensions et des courants, puisque les courants et la tension à l'intérieur d'un baristor sont loin d'être sinusoïdaux.

vol tous azimuts

l'électronique par monts, maux et merveilles

Clemens Valens (labo d'Elektor)

Les *Blockchains*, l'Univers et les Voyages dans le temps



Le terme *blockchain* est depuis plusieurs mois un mot à la mode ayant la réputation d'être un appeau pour vendre du contenu. Mentionnez-le, le lectorat accourra. Le fait même que vous lisiez ceci le prouve.

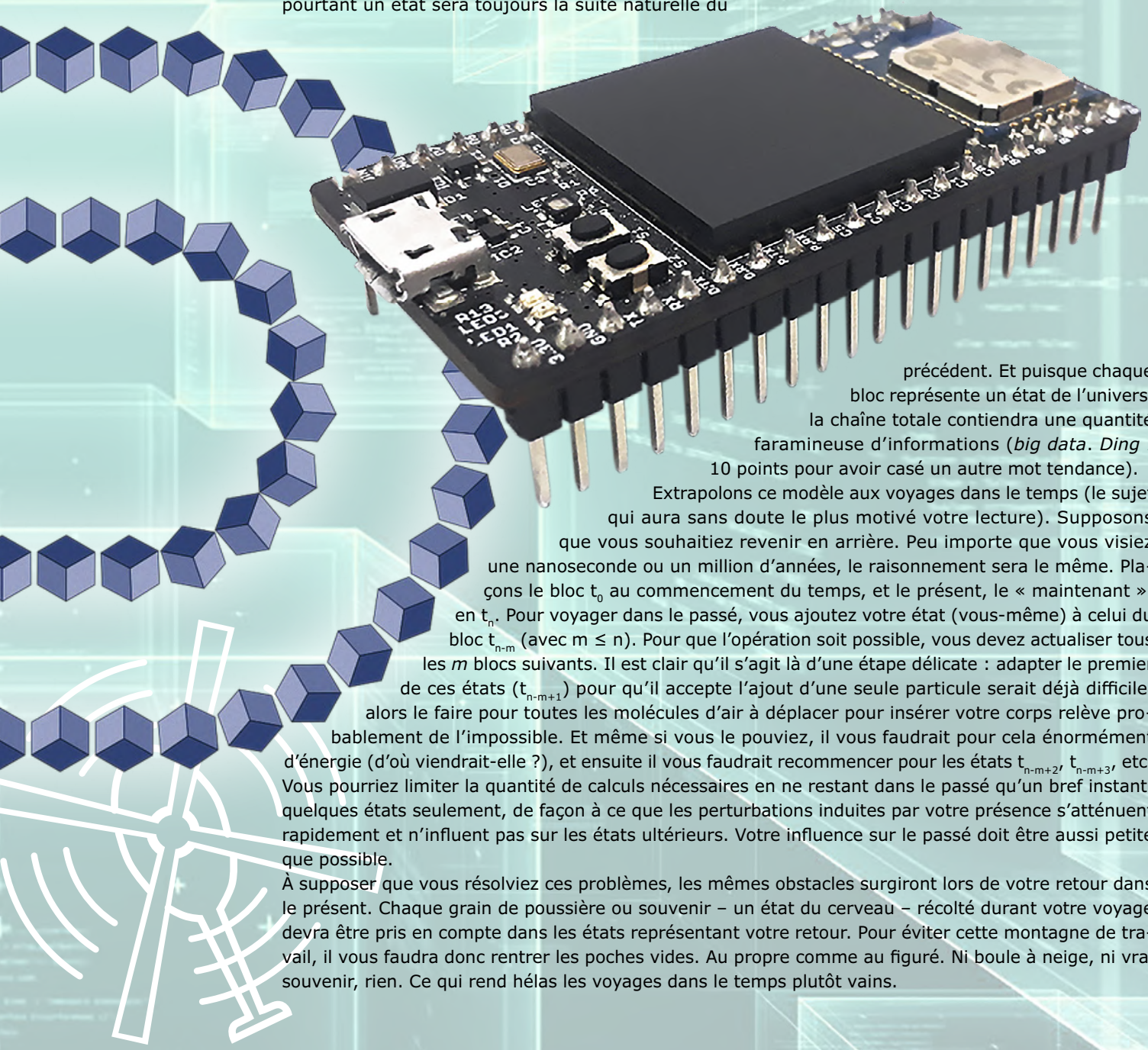
Mais qu'est-ce qu'une *blockchain* ? Dit rapidement, il s'agit d'une chaîne de blocs dans laquelle chaque bloc contient une signature (une somme de contrôle) du contenu du bloc précédent, un horodatage et des données de transaction. La complexité du calcul de la signature d'un bloc rend l'opération longue et coûteuse. De plus, chaque nouvelle signature doit être validée par plusieurs vérificateurs. La modification du contenu d'un bloc requiert l'actualisation des signatures de tous les blocs suivants, sinon la chaîne deviendrait invalide. Si la mise à jour d'une seule signature est « compliquée », imaginez l'effort à déployer pour en actualiser plusieurs centaines ou milliers ! Le contenu d'une chaîne de blocs s'avère donc particulièrement résistant à toute modification.

Bientôt sur carte de prototypage

Les *blockchains* font d'abord penser aux cybermonnaies, mais l'Internet des Objets (IdO, mot tendance n° 2) n'est pas en reste avec la communication de données sécurisées par *blockchain*. Au moment où vous lirez ceci, la campagne Kickstarter lancée par la startup égyptienne Elkrem pour financer sa carte de développement « Blockchain IoT » sera terminée – avec succès ?

Passé recomposé

Un jour, j'ai soudain réalisé que le concept de *blockchain* était transposable au monde réel. Imaginez que nous capturions dans un bloc l'état de l'univers tel qu'il est à un certain instant t_0. À un instant ultérieur proche t_1, l'état de l'univers sera presque identique, à la différence près des mouvements de quelques particules. Comme ces mouvements sont régis par les Lois de l'Univers, le nouvel état est acceptable – sa signature est valide – et peut être chaîné avec le bloc précédent. Plus la distance temporelle entre deux états sera grande, plus la différence entre ces états sera grande, pourtant un état sera toujours la suite naturelle du



précédent. Et puisque chaque bloc représente un état de l'univers, la chaîne totale contiendra une quantité faramineuse d'informations (*big data*. *Ding !* 10 points pour avoir casé un autre mot tendance).

Extrapolons ce modèle aux voyages dans le temps (le sujet qui aura sans doute le plus motivé votre lecture). Supposons que vous souhaitiez revenir en arrière. Peu importe que vous visiez une nanoseconde ou un million d'années, le raisonnement sera le même. Plaçons le bloc t_0 au commencement du temps, et le présent, le « maintenant », en t_n. Pour voyager dans le passé, vous ajoutez votre état (vous-même) à celui du bloc t_{n-m} (avec $m \leq n$). Pour que l'opération soit possible, vous devez actualiser tous les *m* blocs suivants. Il est clair qu'il s'agit là d'une étape délicate : adapter le premier de ces états (t_{n-m+1}) pour qu'il accepte l'ajout d'une seule particule serait déjà difficile, alors le faire pour toutes les molécules d'air à déplacer pour insérer votre corps relève probablement de l'impossible. Et même si vous le pouviez, il vous faudrait pour cela énormément d'énergie (d'où viendrait-elle ?), et ensuite il vous faudrait recommencer pour les états t_{n-m+2}, t_{n-m+3}, etc. Vous pourriez limiter la quantité de calculs nécessaires en ne restant dans le passé qu'un bref instant, quelques états seulement, de façon à ce que les perturbations induites par votre présence s'atténuent rapidement et n'influent pas sur les états ultérieurs. Votre influence sur le passé doit être aussi petite que possible.

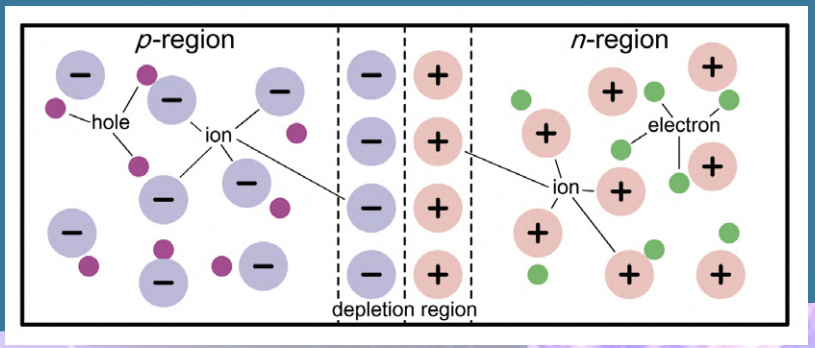
À supposer que vous résolviez ces problèmes, les mêmes obstacles surgiront lors de votre retour dans le présent. Chaque grain de poussière ou souvenir – un état du cerveau – récolté durant votre voyage devra être pris en compte dans les états représentant votre retour. Pour éviter cette montagne de travail, il vous faudra donc rentrer les poches vides. Au propre comme au figuré. Ni boule à neige, ni vrai souvenir, rien. Ce qui rend hélas les voyages dans le temps plutôt vains.

Trous de ver

Je vous entends déjà invoquer les trous de ver, ces objets cosmologiques qui permettraient de voyager dans le temps. Eh bien, les chaînes de blocs peuvent aussi avoir leurs trous de ver puisque rien n'interdit de chaîner plusieurs blocs à un seul. Un embranchement formé de deux chemins ou plus à partir d'un bloc forme ce qu'on appelle une fourche. (Sujet de dissertation : « Peut-on créer une fourche dans l'Univers ? » 500 mots maxi.) Un de ces chemins pourrait ainsi être reconnecté à un bloc situé en amont de la fourche. Cela dit, même si un tel trou de ver permettait de revenir à l'un des blocs du passé, entrer dans ce bloc créerait les mêmes problèmes que ceux évoqués plus haut.

Qu'en est-il des voyages vers le futur ? Eh bien, cela équivaudrait à entrer dans un état t_{n+f} qui n'existe pas encore. (S'il existait, nous vivrions dans le passé. D'après l'analogie de la *blockchain*, cela impliquerait que toutes nos actions seraient déjà connues, sinon le futur serait invalide. Adieu, libre arbitre.) Il vous faudra créer cet état en même temps que le chemin le reliant à l'état présent t_n. La façon dont vous configurez les états formant ce chemin vous revient, la seule contrainte étant de former une *blockchain* valide. Autrement dit, le futur ne dépend que de votre imagination. Ainsi, une façon plus simple, moins énergivore, mais tout aussi valable de voyager dans le futur, serait de simplement fermer les yeux et de rêvasser.

Les porteurs de charge du cristal $NaSn_2As_2$ peuvent être à la fois des trous et des électrons, ce qui élimine potentiellement le besoin de couches multiples dans les semi-conducteurs.



Outil de labo indispensable

Il arrive que l'on ait à mesurer une tension ou une intensité à un endroit d'où il est difficile de regarder l'écran du multimètre tout en maintenant les sondes de mesure. C'est dans ce genre de situation qu'un multimètre Bluetooth s'avère utile. Un appareil connecté tel qu'un ordiphone sert d'écran mobile, qu'il suffit dès lors de placer dans son champ visuel. Et si ce n'est pas possible, l'ordiphone peut lire à voix haute les valeurs mesurées. Ces fonctions suffiraient à elles seules à qualifier ce type d'instrument d'outil de labo indispensable, mais il y a plus : enregistrement des données, tracés, et même affichage de plusieurs multimètres sur le même écran. Plutôt cool, non ?

www.elektor.fr/owon-ow16b-digital-multimeter-with-bluetooth

Fumer est vraiment dangereux

O-k, vous le saviez déjà. Peut-être avez-vous remplacé vos cigarettes traditionnelles par une cigarette électronique (censée être) moins nocive ? Les effets à long terme des e-liquides restent inconnus, mais là non plus je ne vous apprends rien. Ce que vous ne saviez peut-être pas, c'est qu'un habitant de la commune de Plougoulm (Bretagne) a été brûlé lorsque la batterie de sa vapoteuse qu'il avait glissée dans une poche de sa veste est entrée au contact de ses clés de voiture et a pris feu. La victime dit avoir entendu un bruit d'explosion, puis vu une flamme de 40 cm jaillir de sa poche. L'homme a eu la main brûlée en retirant l'objet de sa veste (cela aurait pu être pire à mon avis). Des incidents similaires impliquant des pièces ont été rapportés.
Mais il y a pire. À Fort Worth, au Texas, un homme a été tué lorsque sa e-cigarette lui a explosé au visage et lui a tranché une artère du cou.

(180567-D-04 – version française : Hervé Moreau)

ESP32 comme serveur de temps

ESP32 + RTC + GPS + écran = serveur NTP privé dans votre LAN

Mathias Claussen

Quelquefois il est très pratique de disposer d'un serveur NTP personnel dans son réseau local. Cela ne vaut pas seulement dans le cas d'un réseau isolé de la toile où des logiciels divers doivent disposer de l'heure, mais aussi lorsqu'une manipulation de l'heure récupérée sur l'internet par des projets personnels présenterait un intérêt. Voilà une chose qui, parmi d'autres, est (relativement) aisément réalisable avec les instructions suivantes.

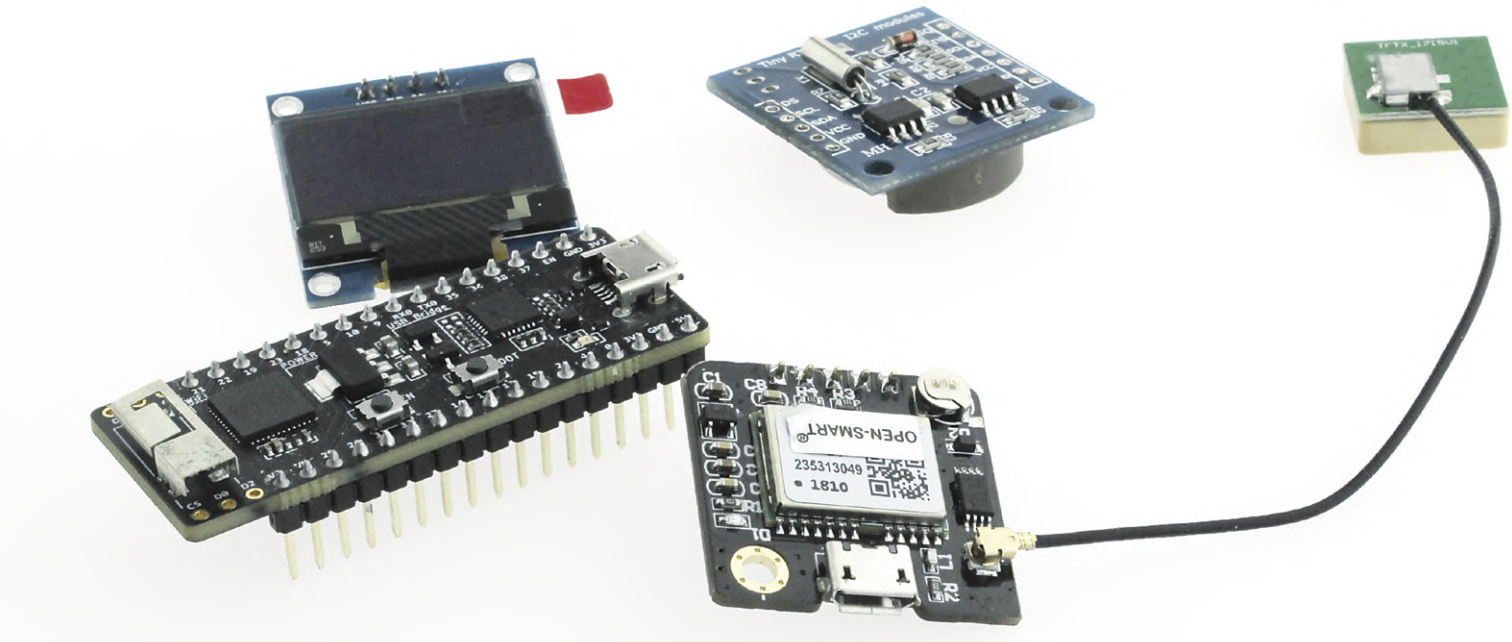


Figure 1. Ces quatre modules constituent le serveur NTP.

Un serveur NTP n'est rien d'autre qu'une adresse IP qui, si elle est correctement sollicitée, renvoie des données horaires relativement exactes par le protocole NTP (*Network Time Protocol*). Si l'on s'intéresse aux détails de ce protocole, on trouvera quantité d'informations et de liens sur Wikipedia, par exemple.

Pour que les PC et autres microordinateurs connectés à l'internet puissent se mettre à l'heure exacte (y compris la date) à la mise sous tension, ils sont pratiquement tous équipés d'une horloge en temps réel (*Real Time Clock*, RTC) sauvegardée par une pile. Cette heure est alors transmise au système d'exploitation et ainsi mise à la disposition de l'ensemble du logiciel sur la machine. Cela assure la bonne marche du calendrier, donne le bon tampon horaire aux fichiers écrits ou modifiés et contrôle la bonne exécution de tâches importantes dépendantes des données horaires.

Mais l'heure devrait rester exacte sur la durée et ne pas dépendre de la précision d'une horloge interne (quartz de la RTC). Pour le garantir, il est souhaitable que les systèmes d'exploitation des ordinateurs alignent périodiquement leurs RTC sur une référence. C'est le service fourni par NTP.

On pourrait aussi se mettre à l'heure

par DCF77 ou GPS mais, d'une part, une fonction logicielle d'interrogation d'un serveur NTP est plus simple et plus économique qu'un matériel spécial et d'autre part, ces deux sources de temps posent des problèmes spécifiques. La portée de l'émetteur DCF77 à Mainflingen (près de Francfort-sur-le-Main, Allemagne) est certes longue, mais tout de même limitée et incapable de franchir les océans. Le GPS est certes disponible dans le monde entier, mais sa réception devient insuffisante à l'intérieur des bâtiments, dont les matériaux de construction atténuent excessivement les signaux à haute fréquence des satellites. Le standard pour la mise à l'heure est donc NTP.

Un serveur de temps personnel ?

Il reste la question : à quoi bon un serveur NTP personnel alors qu'il y a l'internet avec sa quantité de serveurs publics librement accessibles ?

Comme indiqué dans l'avant-propos, il y a des cas où investir (pour un montant raisonnable) dans un serveur NTP personnel a un sens. Il y a, par exemple, les réseaux non connectés à l'internet. Même dans ce genre de réseau, les ordinateurs et autres appareils interconnectés ont en général besoin de connaître l'heure. La saisie ou le réajustement manuel de l'heure, après quelques mois, de chacune de ces machines n'est sans doute pas une opération très passionnante.

De même, le concepteur d'appareils (et/ou d'outils logiciels) qui se procurent l'heure via NTP saura apprécier, au moment des tests, la possibilité de pouvoir modifier les données fournies par un serveur NTP. Et c'est justement dans Elektor qu'ont été présentés dans le passé quelques projets qui requièrent les données NTP. De plus, l'accès à une adresse IP locale est plus rapide et plus stable qu'une requête à l'internet public. C'est quand on a à gérer toute une batterie de microordinateurs qui doivent être synchronisés qu'un serveur NTP personnel prend tout son intérêt.

Il y a donc beaucoup de raisons de réaliser et de mettre en service un petit serveur NTP personnel – peut-être même que vous en trouverez d'autres. Le matériel nécessaire est aujourd'hui peu coûteux; il est très facile de procéder par assemblage de modules complets. Dans ce cas, la puissance de calcul d'un module ESP32 est largement suffisante. De plus, comme ce module gère la connexion sans fil au réseau, il peut être installé à peu près n'importe où. Dans ce projet, le serveur NTP personnel reçoit ses propres données par GPS, fonction pour laquelle il existe aussi des modules à bas prix. Il reste la fonction RTC pour les cas où le GPS est indisponible ou perturbé. Il ne manque plus qu'un petit afficheur pour pouvoir observer le fonctionnement du serveur en permanence. Enfin, le plus important : il nous faut le logiciel qui donnera vie au serveur. C'est ce que nous verrons dans la suite.

Histoire

Notre mini-serveur NTP fut d'abord réalisé pour une exposition en France, où Elektor avait un stand. La première version était basée sur un ESP2866. Les autres composants se limitaient à une puce RTC DS3231 et à un petit afficheur OLED de 0,96 pouce. Ce serveur pouvait fournir des données horaires à tous les appareils exposés par Elektor en l'absence d'internet ou en cas de transmission trop mauvaise. Le prototype rendit le même service au salon *electronica 2018* à Munich. Au labo, il servit de source de temps pour quelques montres Elektor et quelques projets RPi. Et c'est la possibilité de manipuler les données horaires qui nous fut d'une aide précieuse.

La première version du serveur était équipée d'une puce RTC qui était loin de la précision d'une DS3231. Ses dérives devinrent bientôt source d'énervement, car il aurait fallu faire une remise à l'heure tous les mois. Imperceptible le temps d'une exposition, ce défaut devint rapidement inacceptable au labo. Il fallait mettre le matériel du serveur à niveau. Pour cela, on remplaça le module ESP2866 par sa version plus moderne ESP32 et on ajouta un récepteur GPS, disponible dans la boutique d'Elektor, pour la synchronisation automatique de l'heure. Le module ESP32 utilisé était déjà équipé d'un afficheur OLED. Cet afficheur ne répond pas à l'adresse I²C 0x78, comme d'habitude, mais à l'adresse 0x7A, ce qui permet de connecter un second afficheur à l'adresse 0x78. On peut paramétrer l'adresse dans la bibliothèque *U8G2* utilisée. L'afficheur intégré donne l'heure GMT et la date, ainsi que l'adresse IP du serveur et son statut courant (Point d'accès ou Client). Sur le deuxième s'affichent le temps local et les coordonnées GPS. Pour la RTC, on utilisa un module DS3231 disponible au labo. Ainsi se termina la mise à niveau.

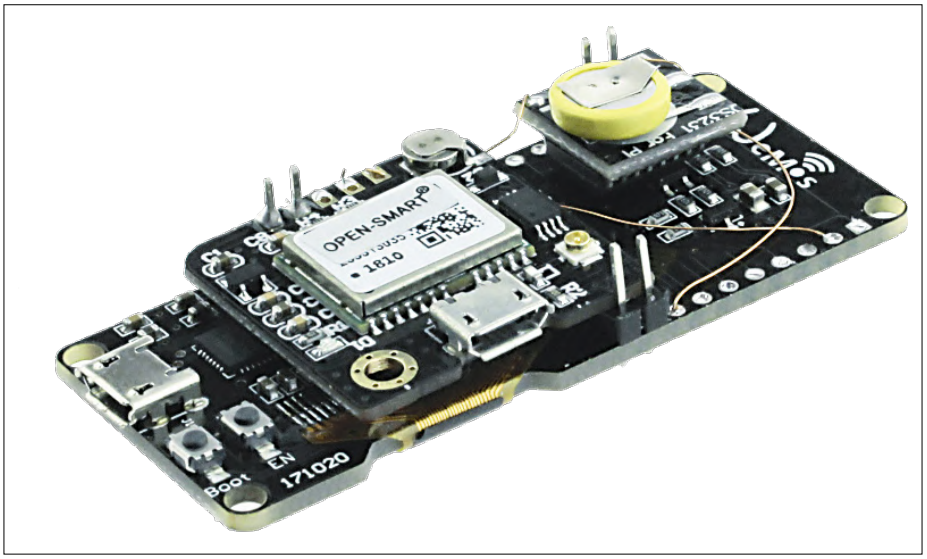


Figure 2. Le sandwich câblé à la main avec l'ESP32 à afficheur OLED intégré, la RTC et le module GPS.

Module

Pour le nouveau serveur NTP, il faut donc combiner les ingrédients de la **figure 1**. Le module central est l'ESP32. S'y ajoutent un afficheur OLED avec son contrôleur SSD1306, une RTC et un module GPS, ce qui fait quatre composants en tout. On peut faire encore plus petit et occuper moins d'espace si l'on utilise un module ESP32 avec afficheur OLED intégré. La combinaison avec la RTC DS3231 et le module GPS prend la forme d'un sandwich compact (**fig. 2**),

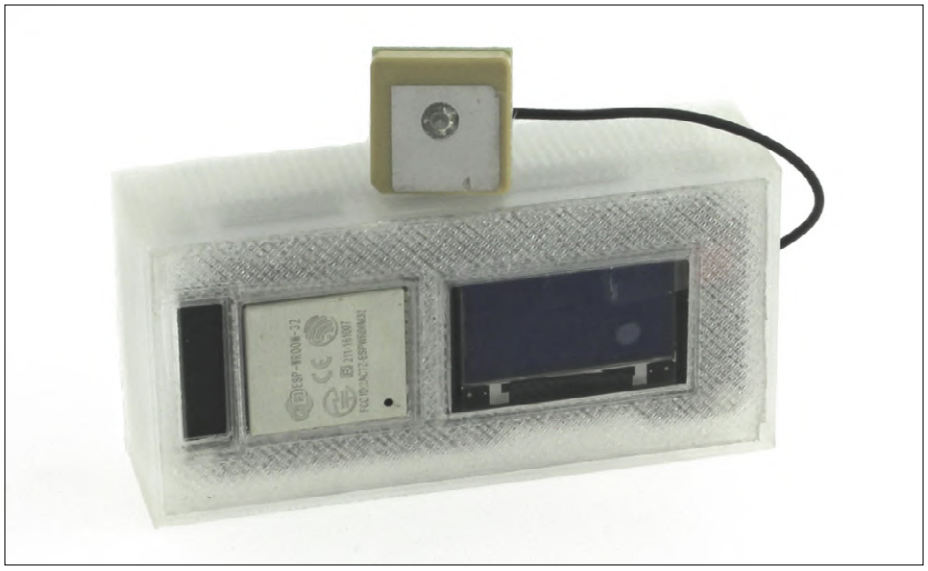


Figure 3. Les éléments de la figure 2 installés dans un boîtier réalisé par impression 3D.

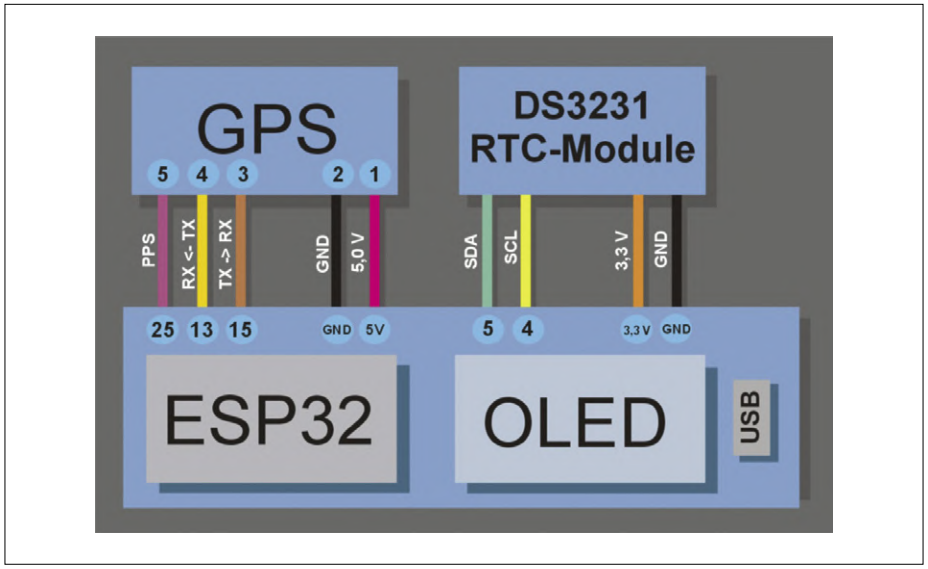


Figure 4. Les liaisons électriques entre les trois modules.

Interruption pour l'« impulsion à chaque seconde »

Pour l'interruption levée à chaque impulsion de seconde, le logiciel utilise la broche GPIO25. Pour que cette broche ne se retrouve pas dans un état indéterminé, on met en fonction la résistance de rappel interne. L'interruption est provoquée par le front montant, ce qui appelle la fonction `handlePPSInterrupt`. Cette fonction doit posséder l'attribut `IRAM_ATTR` – son code doit donc résider dans la RAM de l'ESP32. Le code non résident en RAM serait ralenti par l'accès à la mémoire flash externe ou pire, provoquerait une exception et un redémarrage de l'ESP32 en cas de mémoire flash hors tension. De plus, l'utilisation de variables de type `float` dans une routine d'interruption est problématique, car cela nécessite l'intervention de l'unité de virgule flottante, ce qui, dans la version actuelle de FreeRTOS, conduit à des problèmes et à un comportement imprévisible après l'interruption. À la fin de la routine, `xSemaphoreGiveFromISR` est positionné pour la mise à jour de l'afficheur OLED.

pour lequel nous avons réalisé un coffret par impression 3D (**fig. 3**). Cela donne un serveur NTP petit, mais surtout mobile, qu'on peut installer partout où l'on peut communiquer avec son réseau privé et où l'on peut recevoir les signaux GPS.
Il est important de savoir que la RTC n'utilise pas les broches habituelles de l'ESP32. Pour connecter une RTC, il faut raccorder SCL à GPIO4 et SDA à GPIO5. Le module GPS utilise une connexion série normale, où GPIO13 sert de broche RX et GPIO15 de broche TX. La connexion série exige de croiser les signaux : il faut donc relier ESP32-RX à GPS-TX et ESP-TX à GPS-RX. L'alimentation des deux modules se fait en 5 V directement fournis par le module ESP32, ainsi que la masse. Du côté du module GPS, il faut encore relier le signal d'impulsion des secondes (broche 5) à l'entrée correspondante de l'ESP32 (broche 25). La **figure 4** illustre le câblage nécessaire.

Pour évaluer les problèmes susceptibles de se présenter au cours de l'écriture du logiciel, nous avons commencé par utiliser le logiciel libre *ESPNTPServer* [1]. Pour le « vrai » logiciel du serveur NTP, nous avons réutilisé quelques parties de code d'autres projets Elektor, en particulier les fonctions de correction de l'heure de l'horloge flipper [2], y compris leur capacité à se synchroniser avec différentes sources avec des priorités différentes.
Pour la RTC, on peut utiliser les puces DS3231 ou DS1307. Après le démarrage, le microcontrôleur cherche une RTC à l'adresse I²C 0x68. S'il obtient une réponse, il suppose qu'il a affaire à une DS3231. S'il s'agit d'une DS1307, il faut changer la ligne `RTC_DS3231 rtc_clock;` en `RTC_DS1307 rtc_clock;` dans le module *Firmware.ino* et recompiler le code.
La RTC fournit l'heure quand aucun signal GPS n'est encore reçu ou que celui-ci disparaît. Le serveur fonctionne même si l'ESP32 ne découvre aucune RTC, il se passe simplement du signal du module RTC. L'horloge interne de l'ESP32 démarre alors à minuit le 1er janvier 1970 (début de l'ère Unix) et se met à l'heure à l'arrivée des premières données GPS. Pouvant donc fonctionner sans RTC, le serveur peut aussi fonctionner sans GPS. Il suffit alors de saisir manuellement l'heure de temps en temps. Il va de soi que cette opération s'effectue au

moyen d'une page web de configuration gérée par le serveur.

Données NTP

NTP utilise le protocole UDP (*User Datagram Protocol*) pour la transmission des données. Le serveur NTP est simplement en attente sur le port 123 de requêtes NTP et leur répond par l'envoi d'un paquet de données de 56 octets au format standard NTPv4 (RFC 5905). Le datagramme associé de la **figure 5** indique les données nécessaires. *LI, VN* et *Mode* sont des champs de bits. *Li* indique si à la fin d'une minute il y a une seconde additionnelle. *VN* donne la version du protocole utilisée et *MODE* indique le mode du système (ici « Server »).

Root Delay se réfère à la distance entre ce serveur et le serveur le plus proche. Comme nous sommes le seul dans ce monde (local), ce champ prend la valeur 1. De même, *stratum* vaut 1, puisqu'il n'y a pas de serveur de rang plus élevé. Cette valeur apparaît aussi dans le champ *Root Dispersion*. Les secondes additionnelles sont ignorées. Un *Reference Identifier* avec la valeur « PPS » (*Puls Per Second*) indique qu'on travaille avec une fréquence d'impulsions de 1 Hz.

Le début des temps est fixé au 1er janvier 1900, soit 70 ans, ou 2208988800 secondes avant le début de l'ère Unix. Chaque tampon horaire possède deux champs de 32 bits, l'un pour les secondes de l'heure courante, l'autre pour la partie fractionnaire. Le tampon *Reference Timestamp* devrait contenir l'heure courante. Le serveur ne travaille qu'avec des secondes entières, de sorte que la partie fractionnaire *Reference Timestamp Fraction* est toujours à zéro. Les deux indications sont aussi valables pour les tampons horaires *Receive* et *Transmit.* Dans *Precision*, on a la valeur $\log_2$(fluctuation d'horloge). La fluctuation d'horloge (*jitter*) provient du temps d'appel de la fonction de lecture de l'heure courante plus une seconde due à la mise à zéro de la partie fractionnaire. Cette solution n'est pas parfaite, mais suffisante ici.

Sources de temps

Comme déjà mentionné, le serveur fonctionne avec deux sources de temps. L'une est la RTC, l'autre le signal GPS avec ses impulsions toutes les secondes. Ce signal peut varier et même disparaître à l'occasion. Il provoque une interruption et n'est utilisé que lorsqu'il est disponible. Dans ce cas, il y a basculement du comptage interne des secondes vers celui du GPS. D'autre part, l'impulsion à chaque seconde du GPS est surveillée. Si elle se fait attendre pendant plus de 1400 ms, on admet qu'on a perdu la réception et on bascule sur la source de temps interne. Cela signifie qu'une seconde vient d'être comptée et que ce comptage doit être renouvelé 600 ms plus tard, puisque le logiciel a commuté avec 400 ms de retard par rapport à l'impulsion attendue.

En cas de réception GPS, le module GPS fournit les données horaires correspondantes. Pour se synchroniser sur l'heure mondiale, il suffit d'exploiter ces données une fois toutes les dix minutes pour synchroniser l'horloge interne ou la RTC. Tant que le module GPS reçoit des données, il délivre avec précision une impulsion par seconde qui est comptée par l'horloge interne de l'ESP32.

Pour des besoins particuliers, on peut désactiver la synchronisation par l'heure du GPS. Dans ce cas on n'utilise que l'impulsion à chaque seconde. On peut ainsi tester le comportement de nœuds sur le réseau dans des situations particulières comme le passage de l'heure d'été à l'heure d'hiver.

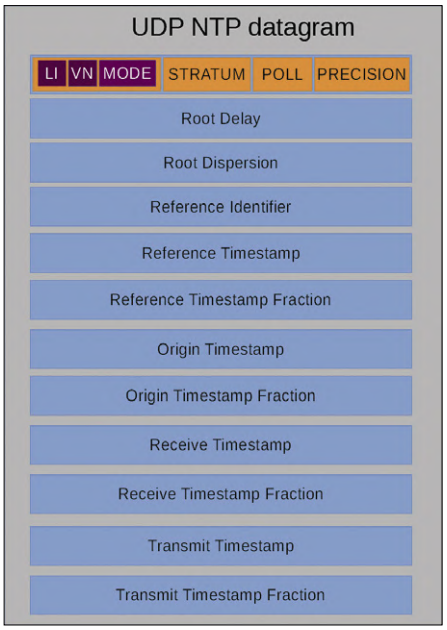


Figure 5. Datagramme d'un paquet NTP.

Logiciel

C'est ici que quelques particularités de l'ESP32 vont se manifester. La mise à jour de l'affichage est effectuée par une tâche spécifique dans une boucle infinie, qui ne bloque toutefois pas l'ESP32. Par communication interprocessus, la tâche est informée quelle ne doit recalculer le contenu des deux écrans qu'une fois par seconde. La tâche est donc inactive jusqu'à l'activation d'un sémaphore. Entretemps, elle ne consomme aucun temps de calcul.

Un mutex (exclusion mutuelle) empêche que la boucle ou la tâche d'affichage accèdent au bus I^2C d'une manière incontrôlée. Ces fonctions sont fournies par le système d'exploitation en temps réel FreeRTOS, qui est normalement masqué par le *framework* Arduino pour l'ESP32. Une description de FreeRTOS, de ses possibilités et de ses chausse-trapes sortirait du cadre de cet article, mais c'est déjà prévu dans un article en préparation.

Liens

[1] ESPNTPServer : https://github.com/liebman/ESPNTPServer

[2] Horloge rétro à afficheurs de flipper, Elektor 3-4/2019 : www.elektormagazine.fr/180307-04

[3] Page de l'article : www.elektormagazine.fr/180662-04

[4] Micrologiciel sur GitHub : https://github.com/ElektorLabs/180662-mini-NTP-ESP32

[5] Préparer l'Arduino pour l'ESP32 : https://github.com/espressif/arduino-esp32/blob/master/docs/arduino-ide/boards_manager.md

[6] Arduino ESP32 Filesystem Uploader : https://github.com/me-no-dev/arduino-esp32fs-plugin

[7] Guide du téléversement de fichiers : https://techtutorialsx.com/2018/08/24/esp32-arduino-spiffs-file-upload-ide-plugin/

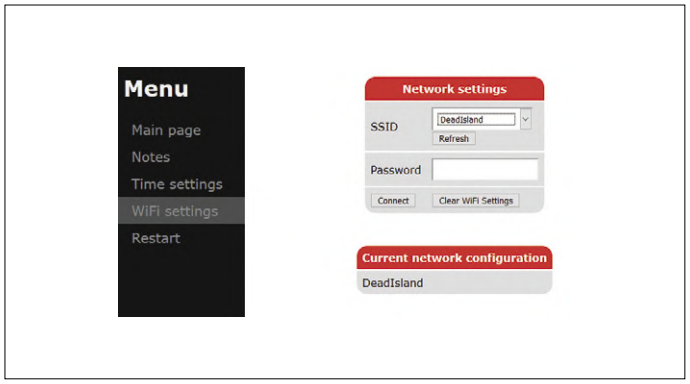


Figure 6. Menu pour les réglages réseau.

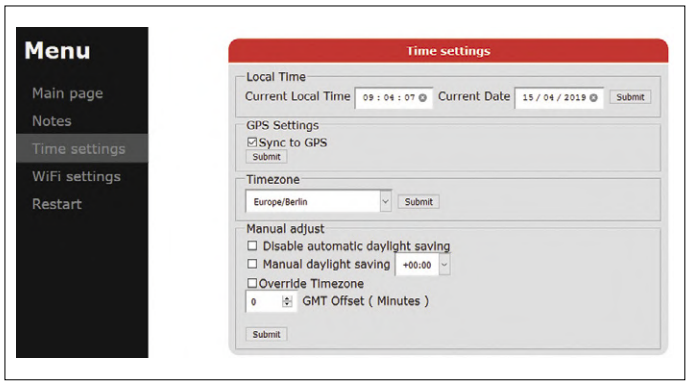


Figure 7. Menu pour les réglages horaires.

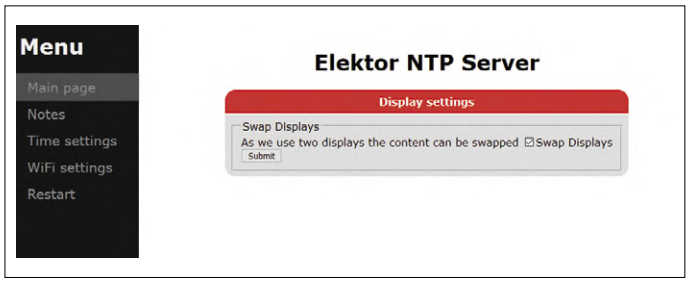


Figure 8. Menu pour le contenu des afficheurs.

Le microgiciel pour l'ESP32 peut être téléchargé gratuitement sur la page Elektor du projet [3]. Sur cette page sont également disponibles les mises à jour éventuelles. Bien entendu, on peut aussi aller voir sur GitHub [4], où l'on trouvera des préversions ou des modifications de la version originale.
Par principe, il faut installer sur son PC la version la plus récente de l'EDI Arduino, ainsi que le paquet pour la carte ESP32 (voir [5]). Pour pouvoir compiler le code du microgiciel, il faut encore installer les bibliothèques suivantes :

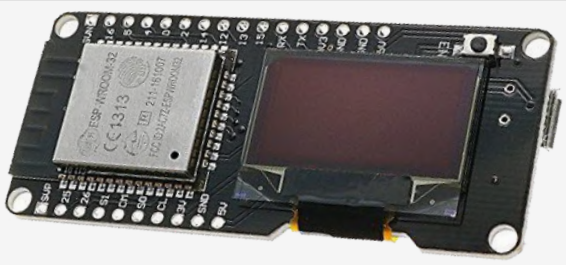
- U8G2
- Time
- Ticker
- TinyGPS++
- RTCLib
- ArduinoJson 6.x
- CRC32

Cette liste est susceptible d'être modifiée pour les éventuelles nouvelles versions du projet. Une fois que le code compilé a été téléversé sur l'ESP32, il faut encore téléverser la page web associée. La façon de le faire est décrite sur l'internet et dans plusieurs projets Elektor utilisant l'ESP32. L'*ESP32-Filesystem-Uploader*, accompagné d'informations, est disponible sur GitHub sous [6] – on trouvera un guide du téléversement des fichiers (un page web n'est rien d'autre qu'un fichier) sous [7]. Après un redémarrage, votre serveur NTP personnel est prêt à prendre son service.

Configuration du serveur

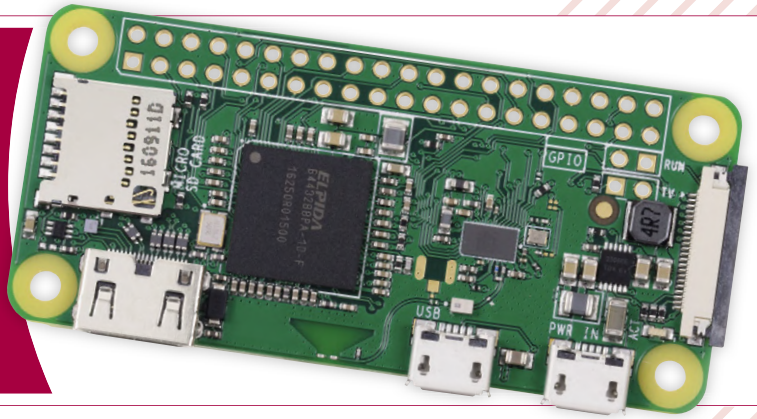
La configuration du serveur est très simple. Après le téléversement avec succès du code et de la page web, le serveur démarre en mode point d'accès. Dans cet état, on peut s'y connecter, ou accéder à sa page web par l'adresse locale 192.168.4.1. Il faut alors saisir l'identifiant SSID et le mot de passe de son réseau local (**fig. 6**). Par l'interface web, il est possible de procéder à tous les réglages horaires.
Une fois que les réglages sont actifs, le serveur essaie d'établir une connexion avec le réseau et d'obtenir une adresse IP par DHCP.
Ensuite, on peut configurer d'autres fonctions par l'interface web (**fig. 7**). Sous la rubrique *Main Page*, on peut permuter les contenus des deux écrans et donc, s'il n'y en a qu'un, choisir son affichage. Ceci termine la configuration. Votre serveur NTP personnel devrait désormais répondre à toute requête NTP entrante. ⏮

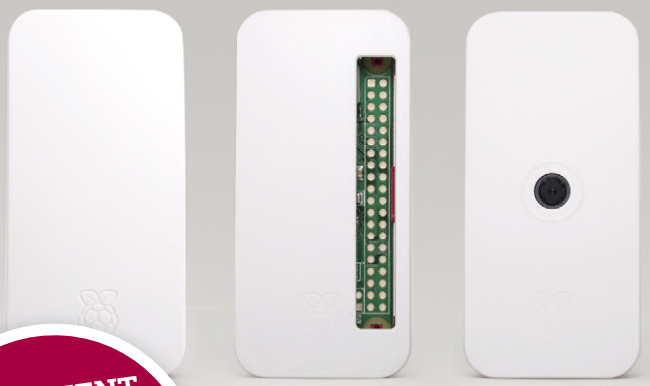
(180662-04 – version française : Helmut Müller)



ABONNEZ-VOUS ET RECEVEZ

RPI ZERO W GRATUIT





SEULEMENT **54,95 €** PAR AN (6 NUMÉROS)

TOUS LES 2 MOIS, LES DERNIÈRES NOUVELLES DU RASPBERRY PI ET LES MEILLEURS PROJETS !

Souscrivez dès maintenant un abonnement d'un an au magazine MagPi, nous vous offrons :

- Six numéros du magazine MagPi
- Une carte Raspberry Pi Zero W
- Un boîtier avec trois couvercles différents
- Un connecteur pour module de caméra
- Un câble HDMI/mini-HDMI et un câble micro-USB/USB OTG

Vos avantages :

- Prix au numéro réduit
- Chaque numéro directement dans votre boîte aux lettres
- Tous les numéros disponibles sous forme numérique (PDF)
- Cadeau de bienvenue d'une valeur de 22,95 €
- Découverte de chaque nouveau numéro avant sa sortie en kiosque

ABONNEZ-VOUS : WWW.MAGPI.FR

recette « KiCad Like a Pro »

création d'un nouveau symbole de composant

Peter Dalmaris (Australie)

Cette « recette » vous apprendra à créer sur mesure le symbole d'un composant absent des bibliothèques de KiCad, et à l'utiliser dans l'éditeur de schéma Eeschema. L'essentiel de la démarche décrite ici s'applique aussi à la modification d'un symbole existant, sujet abordé dans une autre recette du livre.

Note de la rédaction : cet article est la reproduction d'un chapitre du livre anglais *KiCad Like a Pro* de Peter Dalmaris, disponible dans l'e-choppe d'Elektor. Sa mise en page a été adaptée à celle du magazine.

Vous souhaitez très vraisemblablement créer un symbole de schéma, car vous n'en avez trouvé aucun pour tel ou tel composant de votre circuit. Peut-être l'avez-vous d'abord cherché parmi ceux de KiCad, puis lancé une recherche sur Google en espérant le trouver quelque part, mais sans succès.
Il est de même très probable que vous souhaitiez associer une empreinte à votre nouveau symbole. Ce sera possible si votre composant est en boîtier standard, DIP par exemple, mais s'il ne l'est pas il vous faudra aussi créer son empreinte. Cette création d'empreinte sur mesure est expliquée dans une autre recette du livre.
J'ai choisi d'illustrer cette recette avec le temporisateur 555. Son symbole est présent dans un grand nombre de bibliothèques KiCad, mais pour les besoins de l'exposé nous ferons comme si nous n'avions pas réussi à le dénicher.
Notre objectif est de créer un symbole semblable à celui de la **figure 35.1**.
Comme nous le voulons conforme aux conventions de représentation des circuits intégrés, nous adopterons les critères suivants :

- Broches disposées autour d'un rectangle.
- Broches regroupées par fonctions (entrées, sorties, alimentation, etc.)
- Broche Vcc sur le côté supérieur du rectangle.
- Broche GND sur le côté inférieur du rectangle.
- Nom et code identificateur appropriés. Les codes d'identification des composants font l'objet de normes ; lisez [1] pour en savoir plus.

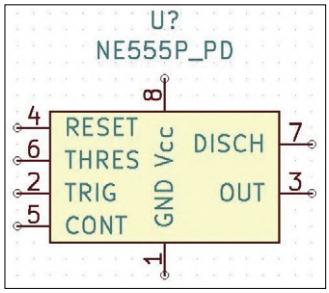


Figure 35.1. Un symbole du CI 555 créé sur mesure.

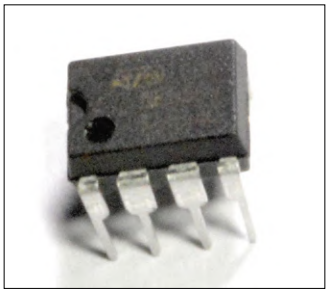


Figure 35.2. Le temporisateur 555, monté dans un boîtier DIP à 8 broches.

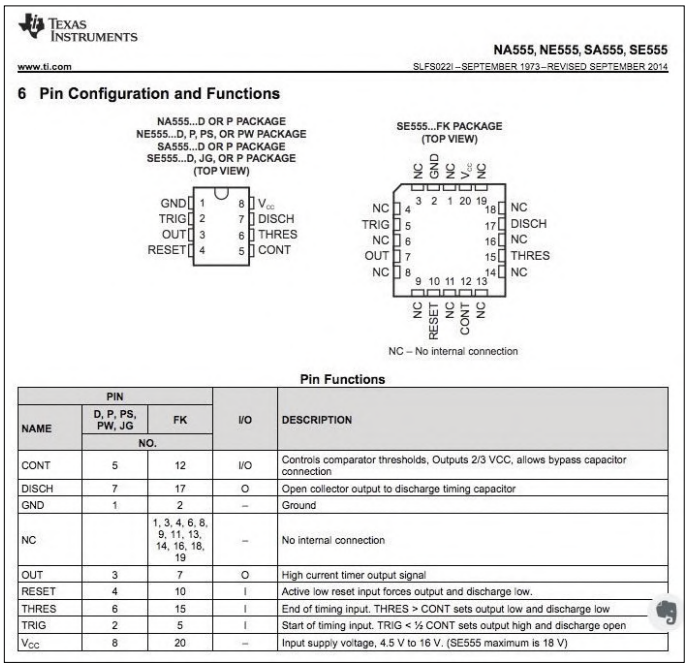


Figure 35.3. Configuration et fonctions des broches du *timer* 555.

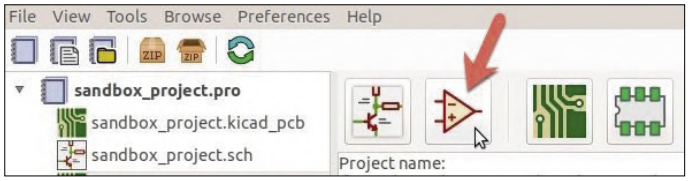


Figure 35.4. Lancement de l'éditeur de symbole.

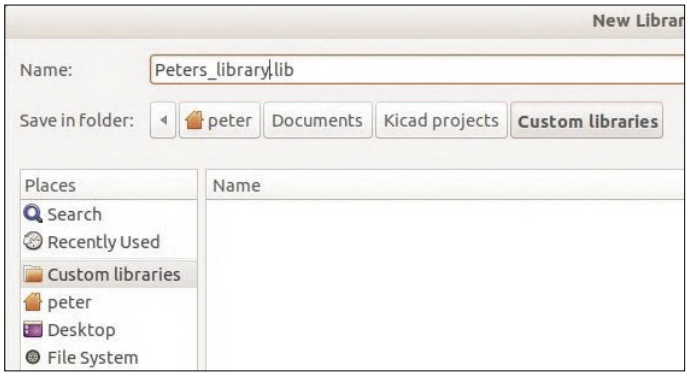


Figure 35.5a. J'ai placé ma bibliothèque dans un répertoire appelé *Custom libraries*.

Le 555 que nous voulons représenter est logé dans un boîtier DIP classique à 8 broches (**fig. 35.2**).
À l'exception de son nombre total de broches, il n'est nullement nécessaire de connaître les caractéristiques physiques du composant dont on souhaite créer le symbole. Ces caractéristiques n'ont d'utilité qu'au moment où l'on travaille sur son empreinte.
Cela ne veut pas dire pour autant que sa fiche technique soit sans intérêt. Bien au contraire puisqu'elle fournit les noms et rôles de chacune des broches, qu'il s'agisse d'une broche d'entrée, de sortie, d'alimentation, ou encore d'une broche bidirectionnelle. Toutes ces informations sont utiles, et plus vous en aurez sous la main, plus aisé et fructueux sera votre travail.
J'ai récupéré la fiche technique du 555 auprès de son fabricant. Les informations qui nous intéressent sont à la page 6, reproduite pour votre commodité sur la **figure 35.3**.
Ceci étant posé, attaquons le processus de création d'un nouveau symbole. Depuis la fenêtre principale de KiCad, cliquez sur le bouton *Symbol Library Editor* (**fig. 35.4**).

Chaque symbole devant être sauvegardé dans un fichier de bibliothèque, il nous faut en premier lieu créer une nouvelle bibliothèque. Cliquez donc sur le bouton *New Library*, ou passez par le menu *File* (**fig. 35.5a**).

KiCad vous demandera si vous souhaitez que cette bibliothèque soit accessible depuis tous les projets (*Global*) ou seulement depuis le projet en cours (*Project*). Choisissez l'option qui vous convient (j'ai choisi *Global*). Vous travaillez maintenant depuis cette nouvelle bibliothèque, où vous stockerez votre nouveau composant. C'est ce que devrait confirmer, dans l'en-tête de la fenêtre de l'éditeur de symbole, la présence du nom et du chemin de la bibliothèque nouvellement créée.
Cliquez ensuite sur le bouton *Create new symbol* de la barre d'outils supérieure. KiCad vous demande de sélectionner la bibliothèque dans laquelle vous souhaitez enregistrer votre symbole. Cliquez sur celle que vous venez de créer, puis sur *OK*. La fenêtre *Symbol Properties* s'ouvre. Les champs à compléter en priorité sont le nom et le code identificateur du symbole. Le nom est typiquement celui du modèle du composant physique, ainsi que toute autre information aidant à son identification. Un nom bien choisi facilitera sa recherche dans la bibliothèque lorsque vous souhaiterez l'utiliser dans un autre projet. Pour différencier mon symbole de ceux déjà présents dans d'autres bibliothèques, j'ai ajouté mes initiales après le nom du 555 (NE555P_PD).

Pour le code identificateur (ou « référence »), n'improvisez pas, référez-vous au tableau de la page Wikipédia [1] (**fig. 35.5b**). Vous y verrez que le code d'un circuit intégré est la lettre U, donc entrez « U » dans le champ *Default Reference Designator*.

La **figure 35.6** montre comment j'ai paramétré la fenêtre *Symbol Properties*. À l'exception du nom et du code, les valeurs indiquées sont celles par défaut.
Cliquez sur *OK* pour valider vos choix. KiCad place le code et le nom du symbole au milieu de la feuille, l'un au-dessus de l'autre. Utilisez le raccourci « M » pour repositionner ces deux blocs de texte selon la disposition de la **figure 35.7**.

Dessinez ensuite le contour du symbole en utilisant l'outil de création de polygones ou celui d'ajout de rectangle. Votre contour devrait ressembler à celui de la **figure 35.8**. Ajoutez une couleur de fond cohérente avec celle des autres symboles de CI en ouvrant la fenêtre des propriétés graphiques du rectangle (placez votre curseur au-dessus du rectangle et appuyez sur la touche E). Sous *Fill*

RV	Varistor
S	Switch (all types, including push-buttons)
T	Transformer
TC	Thermocouple
TP	Test point
TUN	Tuner
U	Integrated circuit (IC)
V	Vacuum tube
VR	Variable resistor (potentiometer or rheostat)

Figure 35.5b. Le code d'un circuit intégré d'après la norme IEEE 200-1975/ANSI Y32.16-1975.

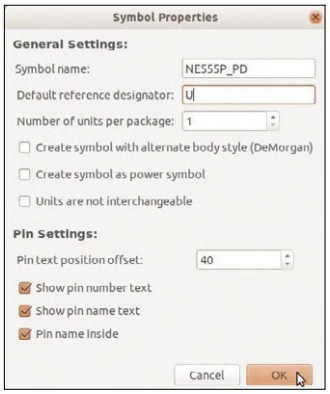


Figure 35.6. Les propriétés associées à un nouveau symbole.

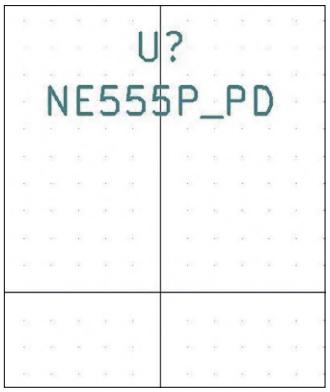


Figure 35.7. Référence et nom du symbole.

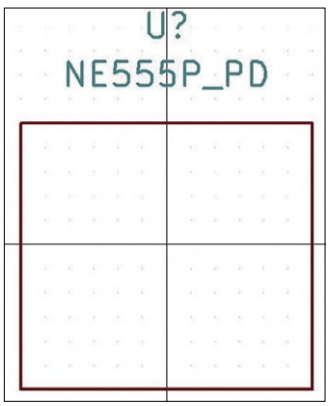


Figure 35.8. Le contour de l'empreinte.

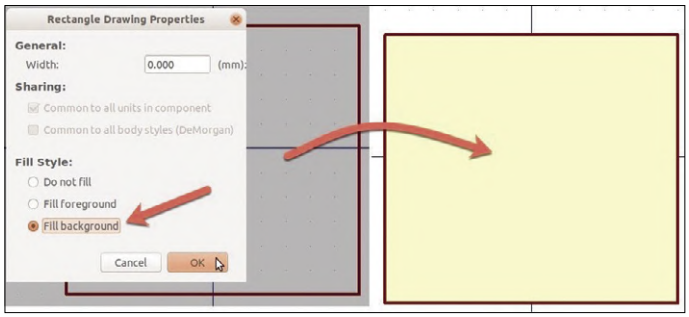


Figure 35.9. Remplissage avec la couleur de fond.

Style, activez le bouton radio *Fill background* (**fig. 35.9**).
Passons au placement des broches. Gardez ouverte la fiche technique du 555 (ou référez-vous à la **figure 35.3**), vous en aurez besoin. Cliquez sur le bouton d'ajout de broches (touche P) de la barre d'outils de droite. Placez les huit broches le long du contour du symbole selon la disposition de la **figure 35.1**. Rappelez-vous que la convention veut que les broches soient regroupées par fonction, et que les deux broches d'alimentation soient placées sur le bord supérieur et la base du rectangle. Commençons par la broche Vcc, la n° 8 selon la fiche technique, donc placée sur le haut du contour. Cliquez sur l'outil de broche, puis cliquez sur le milieu du bord supérieur pour faire apparaître la fenêtre *Pin Properties*. Remplissez-la comme indiqué sur la **figure 35.10**.

J'ai entouré en rouge les champs auxquels vous devez prêter attention. Même si le nom de la broche est arbitraire, un nom approprié s'avère de mise. J'utilise en général celui employé dans la fiche technique. Le numéro de la broche est par contre très important puisque c'est lui qui permet d'associer les symboles et les empreintes d'un schéma aux broches physiques. Lorsque nous créerons l'empreinte de ce composant physique en suivant la recette « Création d'une nouvelle empreinte », c'est le numéro du champ *Pin number* qui dictera quelle connexion de la *netlist* ira à la pastille correspondante de l'empreinte. Entrez donc dans ce champ le numéro indiqué dans la documentation (8 pour la broche Vcc).

Dans le champ à liste déroulante intitulé *Orientation*, sélectionnez l'option correspondant au côté du rectangle où est attachée la broche. La broche Vcc l'étant sur le côté supérieur, son connecteur circulaire doit pointer vers l'extérieur du rectangle (la ligne horizontale de cette icône représente le rectangle). On choisirait de la même façon l'icône dont le connecteur circulaire pointe vers la gauche pour placer une broche sur le côté gauche du contour.

Pour le champ *Electrical type*, j'ai sélectionné *Power input* puisque Vcc est une broche d'alimentation.
Validez en cliquant sur *OK*, puis placez la broche au centre du côté supérieur (**fig. 35.11**). J'ai déplacé les blocs de texte afin qu'ils ne chevauchent pas la broche.

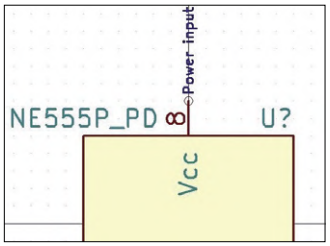


Figure 35.11. La broche Vcc en place.

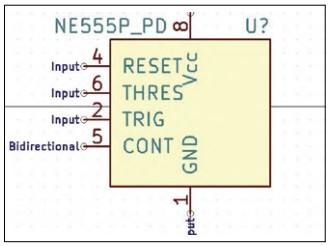


Figure 35.12. Les broches d'entrées sont placées à gauche.

Suivez les mêmes étapes pour ajouter la broche GND (broche 1) sur la base du rectangle. Utilisez le nom et le numéro de la fiche technique, et affectez à cette broche le type *Power input* puisqu'il s'agit aussi d'une broche d'alimentation.

Continuez avec les broches d'entrée du côté gauche que sont, d'après la fiche technique du 555, les broches RESET, THRES et TRIG. La broche bidirectionnelle CONT peut indifféremment être placée à droite ou à gauche, j'ai choisi le côté gauche (**fig. 35.12**).
Affectez à ces quatre broches leurs propriétés respectives (**fig. 35.13**). La broche 5 étant bidirectionnelle, n'oubliez pas de choisir *bidirectional* comme type électrique.

Les deux dernières broches à paramétrer sont, toujours d'après la fiche technique (**fig. 35.3**), les broches de sortie 7 et 3. Ceci fait, votre symbole devrait ressembler à celui de la

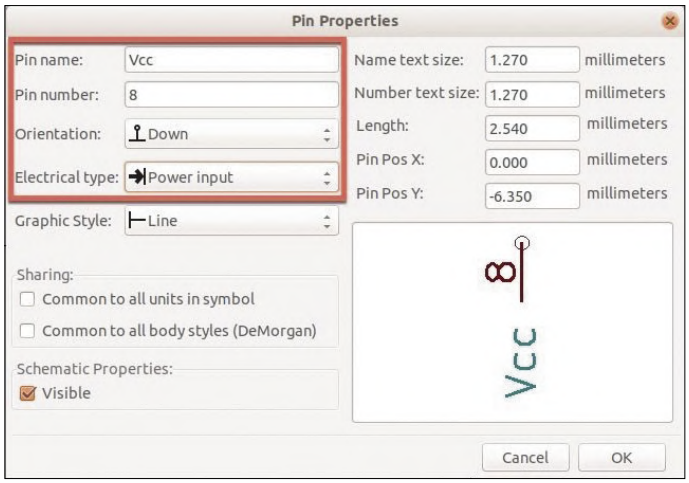


Figure 35.10. Les propriétés de la broche Vcc.

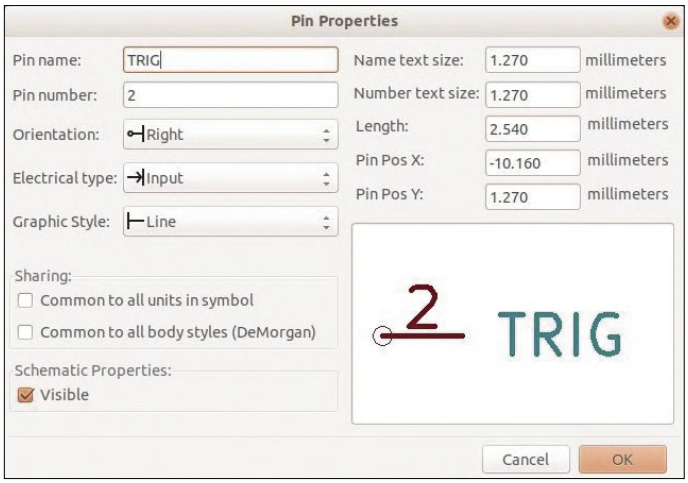


Figure 35.13. Propriétés de la broche d'entrée TRIG.

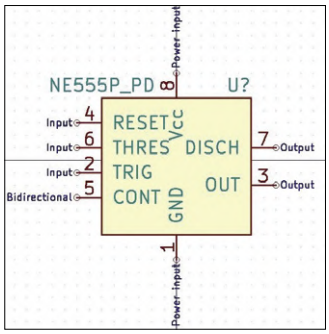


Figure 35.14. Le symbole sur mesure du 555.

figure 35.14.

La dernière étape consiste à ajouter aux propriétés du composant (symbole) l'URL pointant vers sa fiche technique. Ce lien vous servira plus tard de référence, notamment lorsqu'il s'agira de créer une empreinte sur mesure pour le symbole en question.

Cliquez sur *Symbol*, puis sur *Fields…* pour ouvrir la fenêtre *Field Properties*. Cliquez sur la ligne *Datasheet*, puis copiez/collez l'URL dans le champ *Field Value* (**fig. 35.15**). Validez vos changements avec *OK*.

Voilà qui conclut la création du symbole. Sauvegardez-le dans la bibliothèque sélectionnée en cliquant sur le bouton *Save Current Symbol*. Vérifiez ensuite qu'il peut être appelé depuis l'éditeur de schéma. Lancez donc Eeschema, ouvrez le menu *Preferences*, puis *Symbol Libraries*, et ajoutez la nouvelle bibliothèque (une recette du livre explique comment procéder). Placez votre curseur sur la feuille, et tapez « A » pour ajouter votre nouveau symbole. Recherchez votre bibliothèque en entrant son nom dans la zone de recherche (**fig. 35.16**), puis double-cliquez sur le symbole pour l'envoyer dans l'éditeur de schéma.

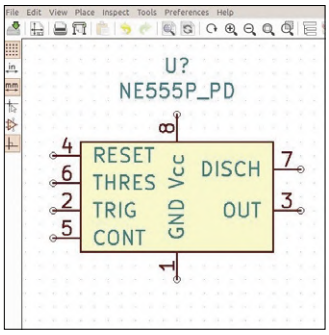


Figure 35.17. Votre nouveau symbole dans Eeschema.

Le symbole sur mesure du 555 devrait apparaître sur la feuille, prêt à être utilisé comme n'importe quel autre symbole (**fig. 35.17**).

Cette recette vous aura appris à créer un symbole à partir de rien, mais dans certaines situations il peut s'avérer plus rapide de modifier un composant existant pour aboutir au symbole recherché. C'est ce que vous apprend la recette « 36. Modifier un composant (symbole) existant ». ⏮

(190041-04 – version française : Hervé Moreau)

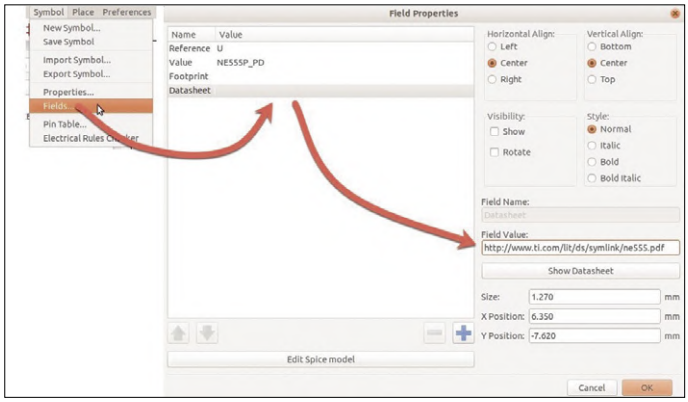


Figure 35.15. Un accès rapide à la fiche technique est très utile.

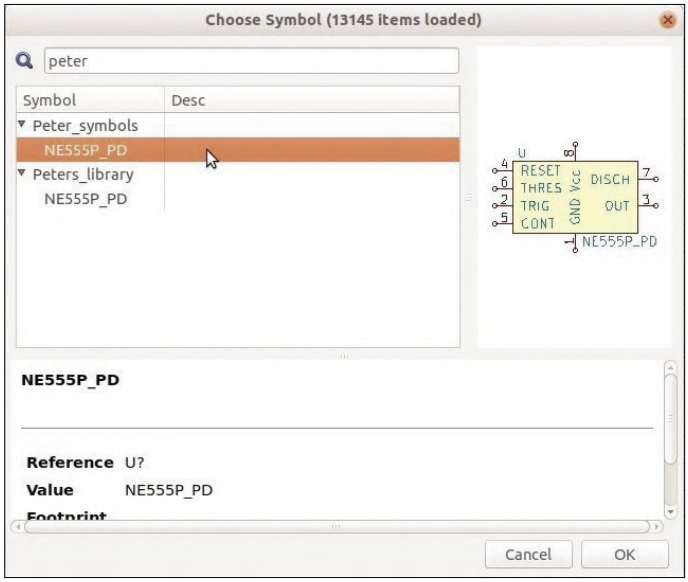


Figure 35.16. Recherche de la bibliothèque et du symbole.

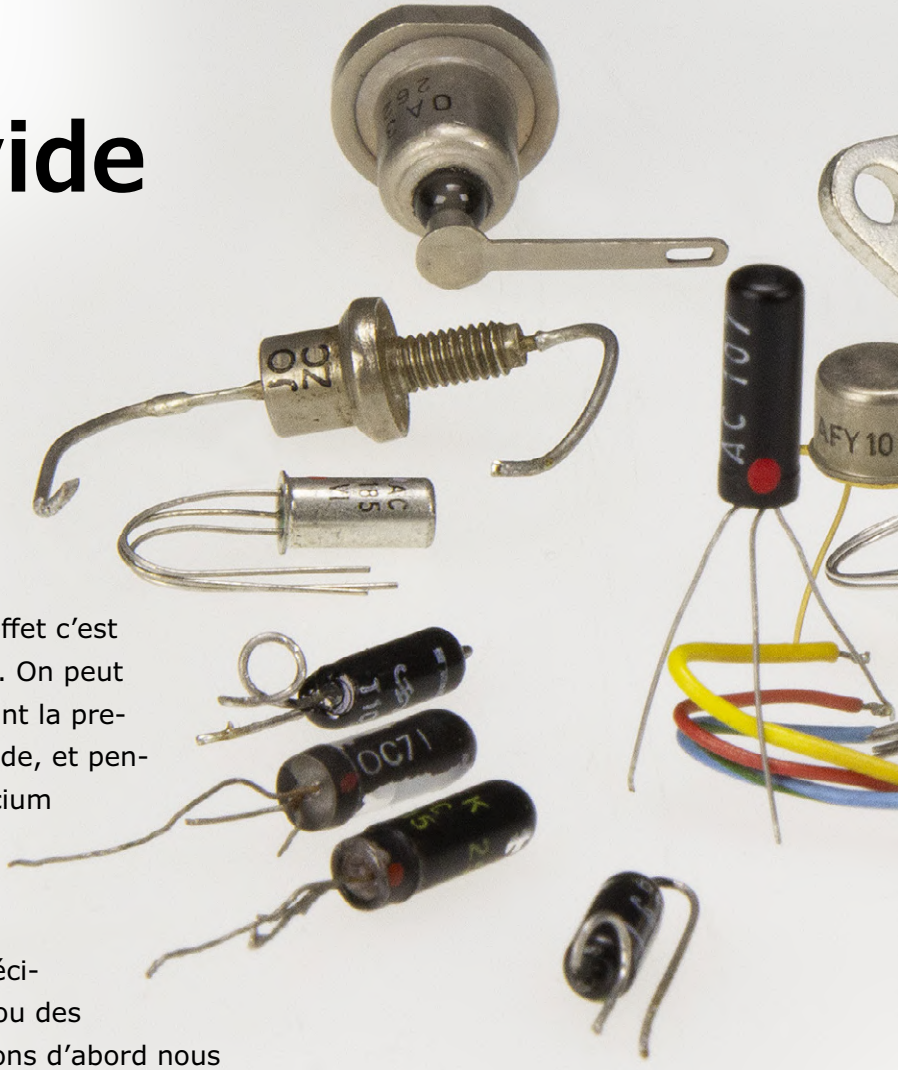
Lien

[1] Code d'identification des symboles : https://en.wikipedia.org/wiki/Reference_designator

des tubes à vide au silicium

avec quelques étapes intéressantes

Martin Beusekamp



L'électronique a un peu plus d'un siècle : en effet c'est en 1907 que Lee de Forest a inventé la triode. On peut en gros couper cette période en deux ; pendant la première moitié, on n'a utilisé que des tubes à vide, et pendant la seconde des semi-conducteurs au silicium – composants discrets ou circuits intégrés. Pendant une courte période de transition, on a conçu et construit des appareils qui n'utilisaient aucune des deux technologies précitées, mais bien des redresseurs au sélénium ou des diodes et transistors au germanium. Nous allons d'abord nous intéresser à ces dispositifs.

Redresseurs au sélénium

Un redresseur au sélénium (**fig. 1**) est un empilement de sandwiches constitués de fines couches de bismuth ou de nickel de ±1 µm, et de sélénium d'environ 60 µm, le tout entre deux plaques d'acier ou d'aluminium. Les sandwiches constituent une barrière pour les électrons, mais ils peuvent la franchir plus facilement dans un sens que dans l'autre ; exactement comme dans une diode.
Chaque élément supporte une tension inverse de 30 à 50 V, mais on se limite dans la pratique à environ 20 V. Le redresseur de la photo, qui a huit éléments, peut donc supporter une tension inverse d'au moins 160 V. Le courant maximal est proportionnel à la surface des plaques : 25 à 30 mA par cm² [1]. Pas besoin de consulter les fiches de caractéristiques pour connaître les capacités d'un redresseur au sélénium : il suffit de compter le nombre de plaques et de mesurer leur surface. Pratique, non ? Vu leur capacité en courant relativement élevée, des redresseurs au sélénium – de dimensions respectables – ont été utilisés pendant une courte période dans des applications nécessitant plusieurs ampères. Leur fiabilité et leur durée de vie étaient cependant limitées, en comparaison avec les diodes actuelles (si on les utilise en deçà de leurs capacités maximales), et on se demandait toujours quand et pas si un tel élément allait lâcher.
Un redresseur au sélénium en fin de vie voit la chute de tension à ses bornes et son courant de fuite augmenter. Il va surchauffer, et dans le pire des cas prendre feu, le tout accompagné d'un dégagement de gaz nauséabond et toxique

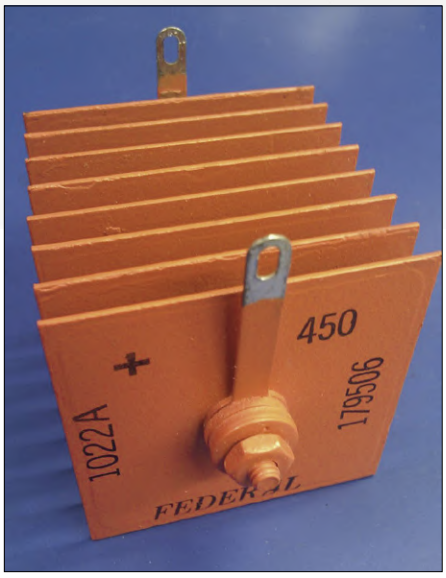
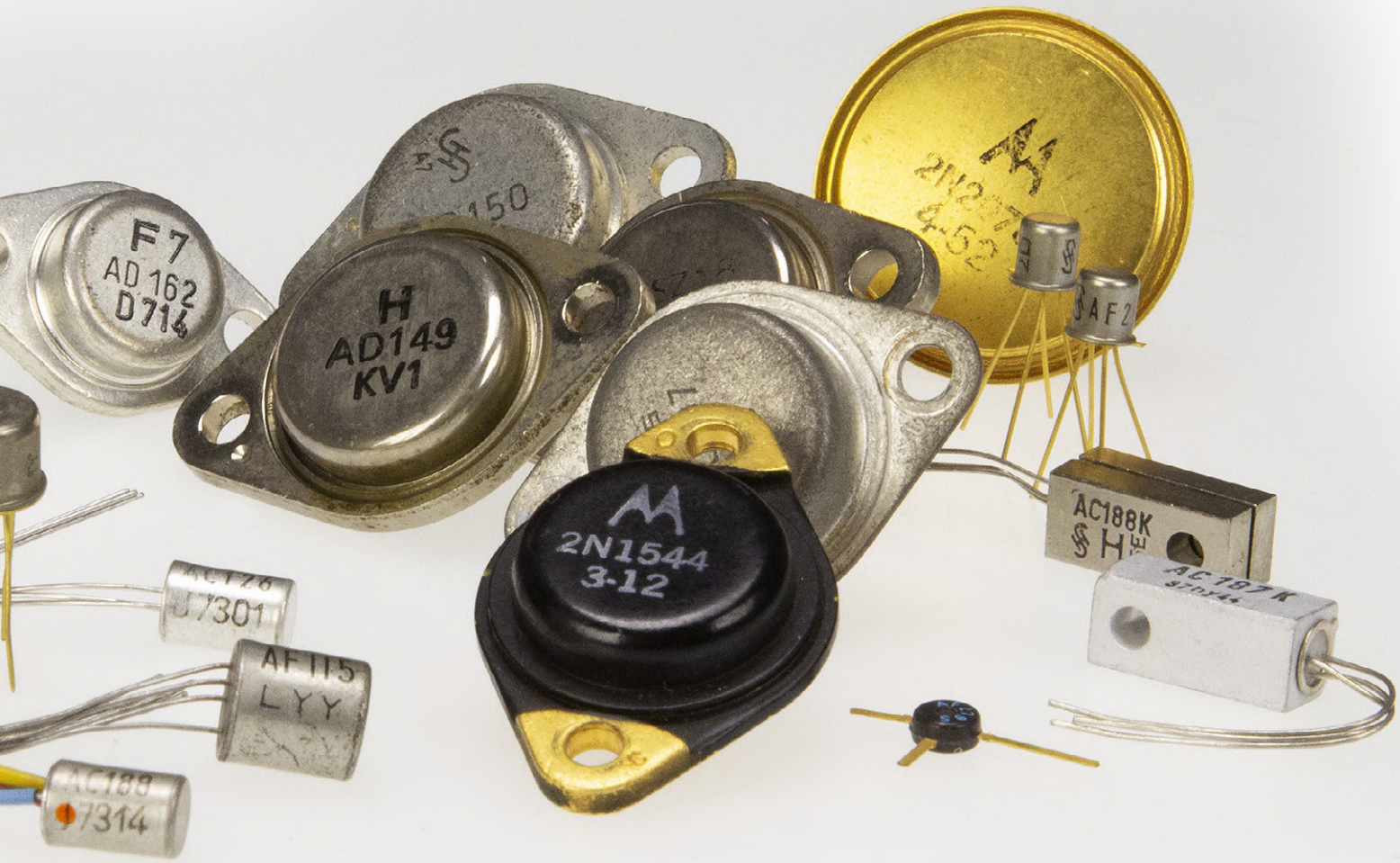


Figure 1. Un beau modèle de redresseur au sélénium. (Creative Commons Attribution-Share Alike 3.0 Unported)



Rétronique est une rubrique mensuelle sur les pages glorieuses et jaunies de l'électronique, avec occasionnellement des montages de légende décrits dans Elektor. Si vous avez des suggestions de sujets à traiter, merci de les télégraphier à redaction@elektor.fr

www.elektor.tv

(voir l'encadré « **Bouchez-vous le nez, ouvrez la fenêtre, et fuyez !** »). Si vous avez un appareil qui utilise des redresseurs au sélénium, il est conseillé de les remplacer par des diodes au silicium, avec une résistance en série pour obtenir la même chute de tension.

Diodes et transistors au germanium

Les premiers semi-conducteurs produits à grande échelle ont été les diodes et transistors au germanium (voir l'encadré « **Codes des tubes appliqués aux diodes et transistors** »). Dans le tableau périodique des éléments, le germanium se trouve juste en dessous du silicium, chaque atome a donc aussi quatre électrons de valence. À l'état pur, le germanium a une structure cristalline régulière.

Bouchez-vous le nez, ouvrez la fenêtre, et fuyez !

Comme nous l'avons signalé, les redresseurs au sélénium ne sont pas sans danger. Lorsqu'ils vieillissent, leur résistance série équivalente augmente et par suite la chute de tension à leurs bornes. La puissance à dissiper par le composant et sa température aussi. À un moment donné, la température limite est atteinte et le sélénium s'oxyde et se transforme en dioxyde de sélénium.

À température ambiante, le dioxyde de sélénium est un solide incolore, mais sous l'action de la chaleur il s'échappe du redresseur sous la forme d'un gaz verdâtre. Ce gaz est très toxique : les effets immédiats peuvent être la nausée, l'étourdissement et l'irritation des yeux et des voies respiratoires ; en cas d'exposition prolongée ou répétée, des séquelles irréversibles au foie et à la rate sont possibles.

Notre nez est heureusement un détecteur de dioxyde de sélénium très efficace : une concentration de 200 ng/m³ est perceptible par la plupart des personnes, et on peut donc prendre des mesures avant qu'il y ait un réel danger. En outre, l'odeur est tellement nauséabonde que personne ne souhaite vraiment rester à proximité de la source...

Dans le tableau périodique des éléments, le sélénium se trouve juste en dessous du soufre. Ils ont donc des propriétés semblables, tout comme le germanium et le silicium qui sont aussi l'un au-dessus de l'autre dans le tableau. L'odeur du dioxyde de sélénium n'est donc pas sans rappeler celle du dioxyde de soufre : oignons pourris et ail.

Voir aussi Barry L. Ornitz, *Selenium Rectifier Replacement* [7].

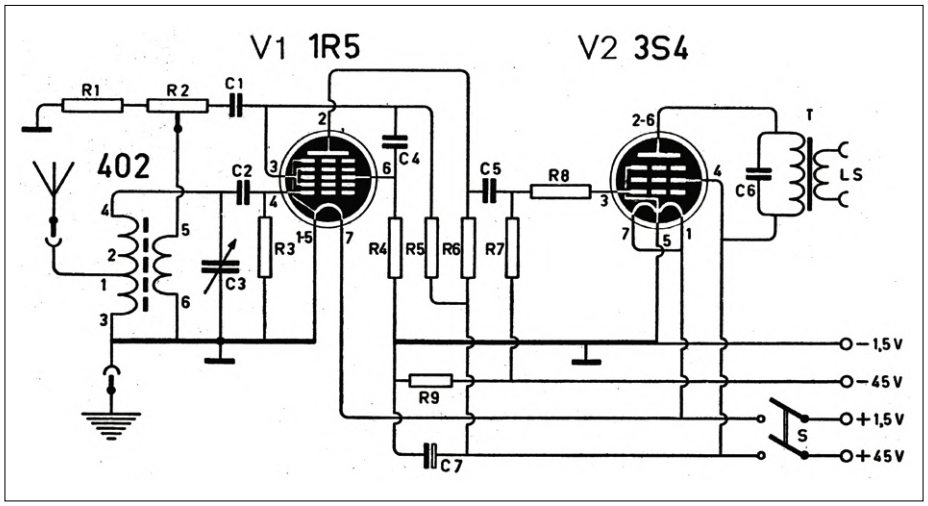


Figure 2. Schéma de l'appareil pouvant être construit avec le contenu de la boîte Nucleon d'Amroh. L'alimentation nécessitait deux batteries.

Figure 3. Couverture de Radio Blan de juillet 1960, un magazine destiné aux jeunes.

Codes des tubes appliqués aux diodes et transistors

Les premières diodes au germanium sont apparues au milieu des années 50 et ont été baptisées OA79 et OA81. Les transistors au germanium ont suivi quelques années plus tard : OC3, OC4, OC13 et OC14.

Ces désignations sont basées sur les codes (européens) utilisés pour les tubes. Le O indique une cathode froide (pas de filament), le A une diode, et le C une triode.

Ce n'est que plus tard qu'on a adopté des codes spécifiques pour les semi-conducteurs. En Europe, le A indique les matériaux avec une bande d'énergie interdite (*bandgap*) de 0,6 à 1 eV (dans la pratique, du germanium), le B une bande interdite de 1 à 1,3 eV (du silicium), et le C les matériaux avec une bande interdite plus haute (par ex. de l'arséniure de gallium). Ces codes sont toujours appliqués : CQY37 désigne par ex. une LED IR, et CQY80 un photocoupleur.

Les diodes et transistors au germanium (OA, OC) étaient généralement encapsulés dans un petit tube en verre recouvert d'un vernis noir. Si on gratte ce vernis, on obtient une photodiode ou un phototransistor !

Tout comme le silicium, le germanium peut être dopé avec d'autres éléments. Avec du phosphore ou de l'arsenic, qui ont cinq électrons de valence, on obtient un semi-conducteur extrinsèque de type N ; avec du bore, qui n'a que trois électrons de valence, on obtient un type P. Les diodes et transistors au germanium fonctionnent de la même manière que ceux au silicium et ont un aspect semblable (voir photo en tête de l'article), seules les caractéristiques électriques diffèrent.

Ainsi, la chute de tension aux bornes d'une diode, ou entre base et émetteur d'un transistor à l'état passant, n'est que de 0,1 à 0,2 V pour le germanium, contre 0,6 à 0,7 V pour le silicium. Il y a cependant des inconvénients. La valeur de la résistance série d'une diode au germanium est plus élevée, de même que le courant de fuite : de l'ordre de 1 µA, contre 1 pA pour le silicium. Un transistor au germanium ne peut quant à lui être complètement bloqué : sans courant de base, la valeur du courant résiduel I_{CEO} entre collecteur et émetteur va de quelques µA pour les transistors petits signaux à quelques mA pour les transistors de puissance.

Les semi-conducteurs au germanium ont d'autres problèmes, comme une très grande dispersion des caractéristiques et la variation significative de celles-ci en fonction de la température ou lors du vieillissement. On peut aussi citer la formation de filaments en métal, voir à ce sujet l'encadré « **Il s'en est fallu d'un cheveu…** ». Il n'est donc pas surprenant que les semi-conducteurs au germanium aient disparu de la scène lorsque la fabrication de types au silicium est devenue suffisamment mature. Seules quelques applications où la faible chute de tension est essentielle les utilisent encore.

Boîtes et kits d'initiation

Avant les semi-conducteurs au germanium, il y avait déjà des kits avec des tubes, par ex. la série Senior de Philips, récepteurs pour ondes moyennes et amplificateurs. Ces kits étaient réservés aux adultes, à cause de la présence de tensions dangereuses et la nécessité de souder.

Amroh a alors sorti quelques kits, toujours avec des tubes, mais alimentés par batteries. Le récepteur pour ondes moyennes Nucleon (**fig. 2**) utilisait deux tubes, un 1R5 (ou DK91) et un 3S4 (ou DL92). Les filaments étaient

alimentés sous 1,5 V, avec un courant de 50 mA pour le premier tube, 100 mA pour le second ; une pile électrique LR20 (type D) de bonne qualité pouvait tenir le coup pendant plusieurs heures, jusqu'à ce que la tension chute à 1,2 V ou moins. Une batterie de 45 V fournissait la tension d'anode des deux tubes. Des batteries pouvant délivrer une tension de plusieurs dizaines de volts (par ex. 90 V) ont été disponibles jusqu'à la fin des années 50, mais il fallait un porte-monnaie bien garni...

Divers magazines proposaient également des projets de récepteurs, par ex. Radio Plans en France, Practical Wireless en Grande-Bretagne, et Radio Blan aux Pays-Bas (**fig. 3**). Le prix (non actualisé) de ce dernier n'était que de 0,07 €. Bon marché, même à l'époque, mais il y avait des dizaines d'annonceurs pour ce nouveau marché en pleine expansion, dont beaucoup de détaillants en composants.
Avec les diodes au germanium, des jeunes pouvaient facilement réaliser un récepteur à cristal, par ex. le Pioner I de Philips. D'autres kits ont vu le jour avec l'avènement des transistors au germanium, comme les Philips Pioner II, avec deux OC13 (**fig. 4**), et Pioner III, avec un transistor supplémentaire OC14 comme amplificateur de puissance.
Philips a sorti peu après des boîtes d'initiation à l'électronique, les EE8 et EE20 : EE = *Electronic Engineer*, et le chiffre représente le nombre d'appareils pouvant être construits avec les composants dans la boîte. On y trouvait des transistors au germanium « modernes », les AF116 et AC126, et les appareils à construire étaient très diversifiés : amplificateur audio, orgue électronique, intercom, relais acoustique, indicateur d'humidité, temporisateur, pont de mesure, ou récepteur pour ondes moyennes (il y avait encore des émetteurs sur ces ondes à l'époque...).

Le meilleur de trois mondes

L'industrie utilisait aussi des composants au sélénium ou au germanium ; l'important était d'avoir un appareil plus performant, plus petit, plus fiable et/ou moins cher. Et les composants au silicium avaient fait leur apparition avant que les transistors au germanium aient supplanté les tubes.
Il n'est donc pas étonnant qu'un appareil de la seconde moitié des années 60

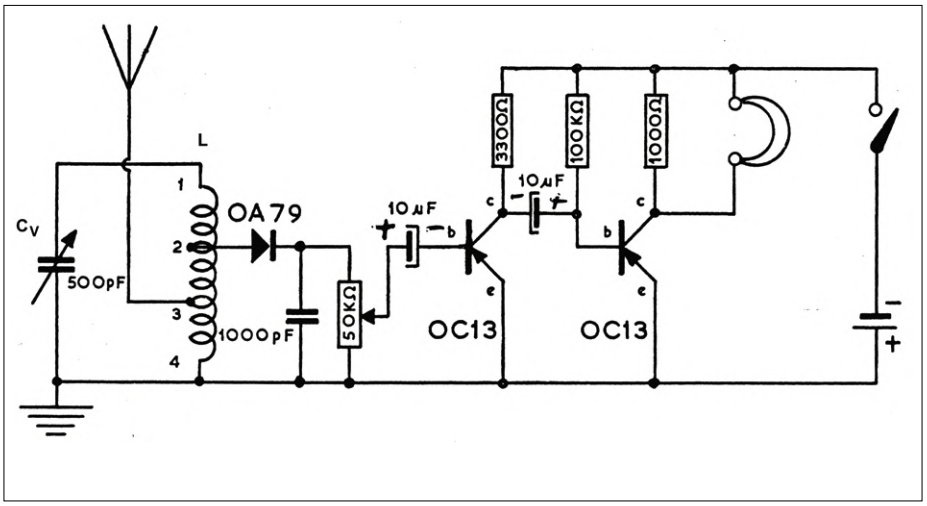


Figure 4. Schéma de l'appareil pouvant être construit avec le contenu de la boîte Pioner II de Philips, dont une diode au germanium OA79 et deux transistors au germanium OC13.

Figure 5. L'oscilloscope Philips PM3230, qui fonctionne avec des tubes, des diodes au silicium, et des transistors au germanium et au silicium. Photo : Gerrit Busscher.

Pour la cuisson : mollet ou dur ?

Les semi-conducteurs sont sensibles à la température, et les opérations de soudage doivent être rapides. Ceux au germanium sont particulièrement fragiles de ce point de vue.
Il y avait pas mal de trucs et astuces pour ne pas endommager les composants au germanium lors d'un dessoudage. Une pince crocodile accrochée à la patte à dessouder suffisait en général à éviter la surchauffe du composant.
Le magazine néerlandais Radio Blan, destiné aux jeunes, donnait nombre de conseils judicieux : « Demande à ta maman une tranche de pomme de terre crue. Incise-la jusqu'au centre et enfile-la sur la patte de la diode ou du transistor à souder. Retire la pomme de terre avec précaution après soudage ».
Ces trucs et astuces étaient bien nécessaires, car les amateurs n'étaient pas experts en soudage et les composants étaient relativement chers : en 1960 une diode au germanium coûtait environ l'équivalent de 3 € actuels, et un transistor près de sept fois plus !

Figure 6. Le récepteur pour ondes moyennes Philips IC2000 de 1967, et son étui. Ce que l'on voit à droite est le chargeur avec adaptateur secteur.

utilise les trois technologies. C'est le cas de l'oscilloscope Philips PM3230 (**fig. 5**). On y trouve neuf tubes (dont des ECC88 et des E810F) et quatre-vingt-un semi-conducteurs, dix-sept au germanium (essentiellement des transistors HF AF11x) et soixante-quatre au silicium (diodes et ponts de diodes).

Pourquoi un tel choix ? Un oscilloscope analogique comprend divers circuits aux caractéristiques fort différentes. Les amplis d'entrée doivent traiter des signaux de quelques mV, l'anode post-accélératrice de l'écran nécessite quant à elle une tension de plusieurs kV. Les alimentations fournissent du continu, mais l'appareil accepte des signaux jusqu'à 10 MHz ; et d'autres appareils, bien plus chers, allaient encore plus loin dans le domaine des HF. À l'époque, seules les technologies à tubes ou au germanium étaient suffisamment éprouvées pour remplir certaines de ces fonctions ; le manuel de maintenance du PM3230 [2], en néerlandais, fournit plus de détails à ce sujet.

L'utilisation de certains composants était parfois un simple argument de vente. Au salon international de la radiodiffu-

Il s'en est fallu d'un cheveu...

Les premières automobiles et les premiers trains n'étaient sans doute pas aussi fiables que les diligences, mais on n'a rien pour rien. Comme disait Henry Ford : « Si j'avais écouté mes clients, je me serais reconverti dans l'élevage de chevaux... ».

Il en allait de même avec les premiers transistors. Ils étaient plus petits et consommaient moins que les tubes, mais ils étaient nettement moins fiables.

Les premiers transistors au germanium produits en série (OC170, OC171, AF11x, etc.) fonctionnaient très bien, mais les fabricants ne pouvaient évidemment pas encore déterminer leur durée de vie. Un des facteurs qui limitait cette durée de vie était la formation spontanée de filaments de métal sur la face interne des boîtiers en étain. Ce métal, tout comme l'antimoine, le cadmium, l'indium et le zinc, est en effet sensible à la formation de filaments à sa surface (voir **figure 8** et explications de Wikipédia [8]).

Le boîtier en étain était creux et la puce du transistor elle-même était protégée par de la graisse de silicone. On ne pouvait cependant empêcher la formation d'une bulle d'air lors de la fabrication, qui entraînait la formation spontanée de filaments de quelques micromètres. C'était suffisant pour causer un court-circuit entre boîtier et électrodes du transistor : base, collecteur et/ou émetteur.

Inutile de dire que les fabricants – et même la NASA – ont eu fort à faire avec ce problème dans les années 50-60.

L'introduction de boîtiers en plastique ou en alliage non affecté par la formation de filaments a heureusement résolu le problème.

Voir aussi le mystère de l'OC171 [9] et le problème de l'AF11x [10].

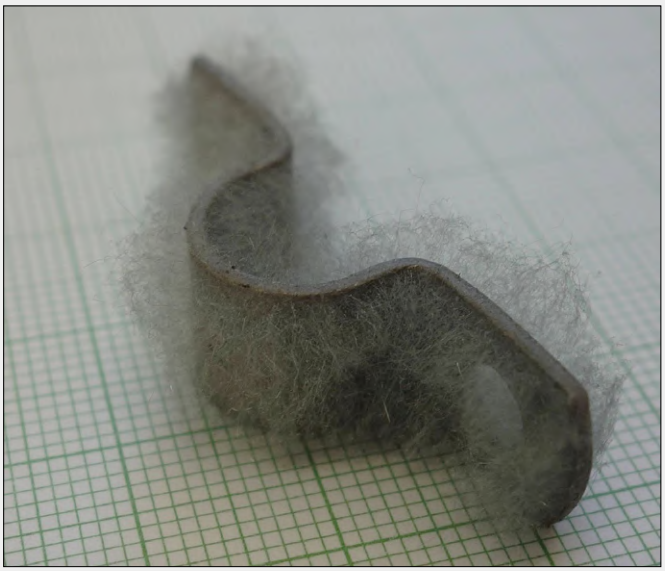


Figure 8. Formation de filaments sur un étrier en acier galvanisé (source : Wikipédia).

sion de Berlin en 1967, Philips a présenté un modèle de radio pour ondes moyennes à l'aspect futuriste, l'IC2000, dont la **figure 6** montre un exemplaire en parfait état. Trois quarts du boîtier rond de 3 cm d'épaisseur et 7 cm de diamètre étaient occupés par le haut-parleur. Autour de l'aimant de celui-ci, un circuit imprimé recevait deux circuits intégrés, dont le TAA263 qui ne contient que trois transistors, une diode au silicium et trois transistors, un au silicium et deux au germanium pour l'ampli de puissance.
L'emploi de circuits intégrés n'était pas essentiel, mais c'était un bon concept de commercialisation, tout comme le choix de la forme de l'appareil et sa dénomination : IC de *integrated circuit*, un mot magique à l'époque, et 2000, bref la technologie du futur... Le schéma de cet appareil et de bien d'autres se trouvent sur les sites www.doctsf.com [3] et www.radiomuseum.org.

... et d'un seul

Des appareils relativement plus simples, comme une alimentation (linéaire) industrielle, n'ont pas besoin de tout ce tralala ; ce qui compte ce sont les performances et la robustesse. Les alimentations Philips des années 60, comme les PE4802 (0-15 V, 6 A) et PE4804 (2×0-30 V, 2×2 A), ne contiennent que des redresseurs au sélénium et des semiconducteurs au germanium.
La **figure 7** montre les entrailles d'une PE4804 de la collection de l'Université de Twente, aux Pays-Bas [4]. On y voit bien les redresseurs au sélénium, en bleu, et la structure symétrique indiquant une alimentation double. Les radiateurs des transistors de puissance au germanium sont sous les pales du ventilateur, dont le moteur est une machine synchrone standard, alimentée par le secteur et que Philips utilisait à l'époque dans tous ses tourne-disques. Les manuel et schéma de l'appareil sont disponibles via les liens [5] et [6].
Les alimentations Philips PE4802 et PE4804 sont des objets de collection, pas des appareils ésotériques. Elles illustrent bien les choix technologiques d'une époque, sans tubes et sans silicium. ⏮

(180574-D-04

version française : Jean-Louis Mehren)

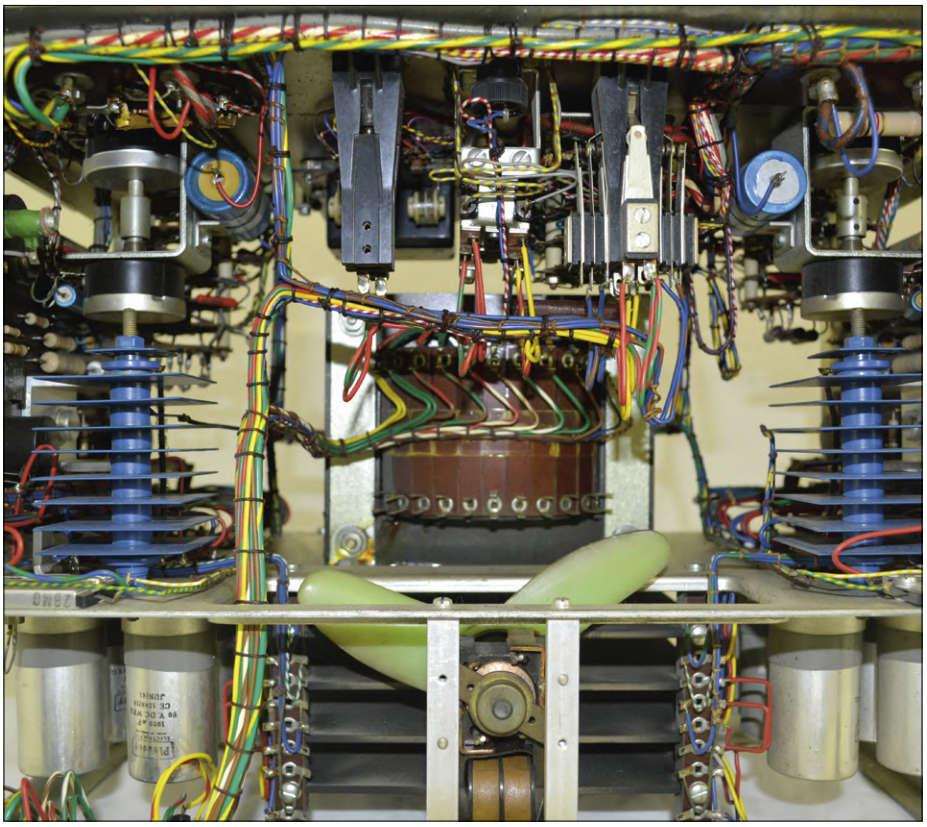


Figure 7. L'alimentation Philips PE4804 : uniquement des redresseurs au sélénium et des semiconducteurs au germanium. Don de E.G. Pleiter – Hengelo. Photo : Rob Quentemeijer.

Liens et litérature

[1] Source : E.A. Harty – Characteristics and applications of Selenium-Rectifier Cells – Transactions on Electrical Engineering, volume 62, October 1943

[2] Oscilloscope Philips PM3230 : www.doctsf.com/grandlivre/fiche.php?ref=41422

[3] Radio Philips IC2000 : www.doctsf.com/grandlivre/fiche.php?ref=17171&variante=3&afficher_photos=0

[4] Historic Study Collection of the Faculty Electrical Engineering, Mathematics and Computer Science of the University of Twente: http://studieverzameling.utwente.nl/

[5] Philips PE4804/01 : www.radiomuseum.org/r/philips_gleichspeisungs_speisege.html

[6] Philips PE4804/05 : www.radiomuseum.org/r/philips_pe4804.html

[7] Remplacement de redresseurs au sélénium : https://yarchive.net/electr/selenium_rectifiers.html

[8] Barbe (cristallographie) : https://fr.wikipedia.org/wiki/Barbe_(cristallographie)

[9] Le mystère de l'OC171 : www.elektormagazine.fr/magazine/elektor-200807/11104

[10] Le problème de l'AF11x : www.markhennessy.co.uk/articles/vintage_transistors.htm#the_af11x_problem

algorithmes honnêtes

Tessel Renzenbrink

L'utilisation responsable d'algorithmes est devenue un sujet d'actualité : comment s'assurer qu'ils ne soient pas bâtis sur des préjugés, ou qu'ils reflètent bien nos valeurs éthiques et morales ?

Cathy O'Neil, mathématicienne et analyste, a écrit le livre *Algorithmes, la bombe à retardement* (*Weapons of Math Destruction*) [1], et elle se réjouit que l'on prenne enfin ce sujet au sérieux. « C'est un débat difficile, souvent répétitif et frustrant, mais il est indispensable qu'il ait lieu », a-t-elle répété lors d'une interview. En avril de l'année passée, elle a conseillé la ville d'Amsterdam pour la mise en œuvre d'un algorithme auto-apprenant pour le traitement des plaintes.

Le livre de Cathy a fortement contribué à une certaine prise de conscience concernant l'utilisation d'algorithmes. Elle a démontré, de manière compréhensible et convaincante à l'aide d'exemples réels, comment des modèles mathématiques pouvaient causer du tort. Ses « armes de destruction mathématiques » (ADM) ont trois caractéristiques : elles sont dommageables, elles opèrent à grande échelle, et leur fonctionnement est opaque. Elle illustre très bien ce dernier point dans son livre : « Les verdicts des ADM tombent comme des diktats énoncés par les démiurges algorithmes ». Ces décisions sont sans appel et personne n'a à en répondre s'il s'en suit des dommages quelconques.

Une des AMD citées par Cathy est l'utilisation de la solvabilité pour fixer la prime d'une assurance auto. Dans le modèle mathématique utilisé par les compagnies d'assurance américaines, la solvabilité de l'automobiliste est considérée comme plus importante que son comportement dans la circulation. En Floride, des conducteurs sans soucis financiers, mais ayant été condamnés pour avoir roulé sous influence de l'alcool, paient annuellement jusqu'à 1 500 dollars de moins que des conducteurs exemplaires ayant un mauvais score en matière de solvabilité ! Cathy montre clairement, exemple après exemple, que ce sont surtout les pauvres et les faibles qui sont les victimes des ADM.

Automatiser le favoritisme

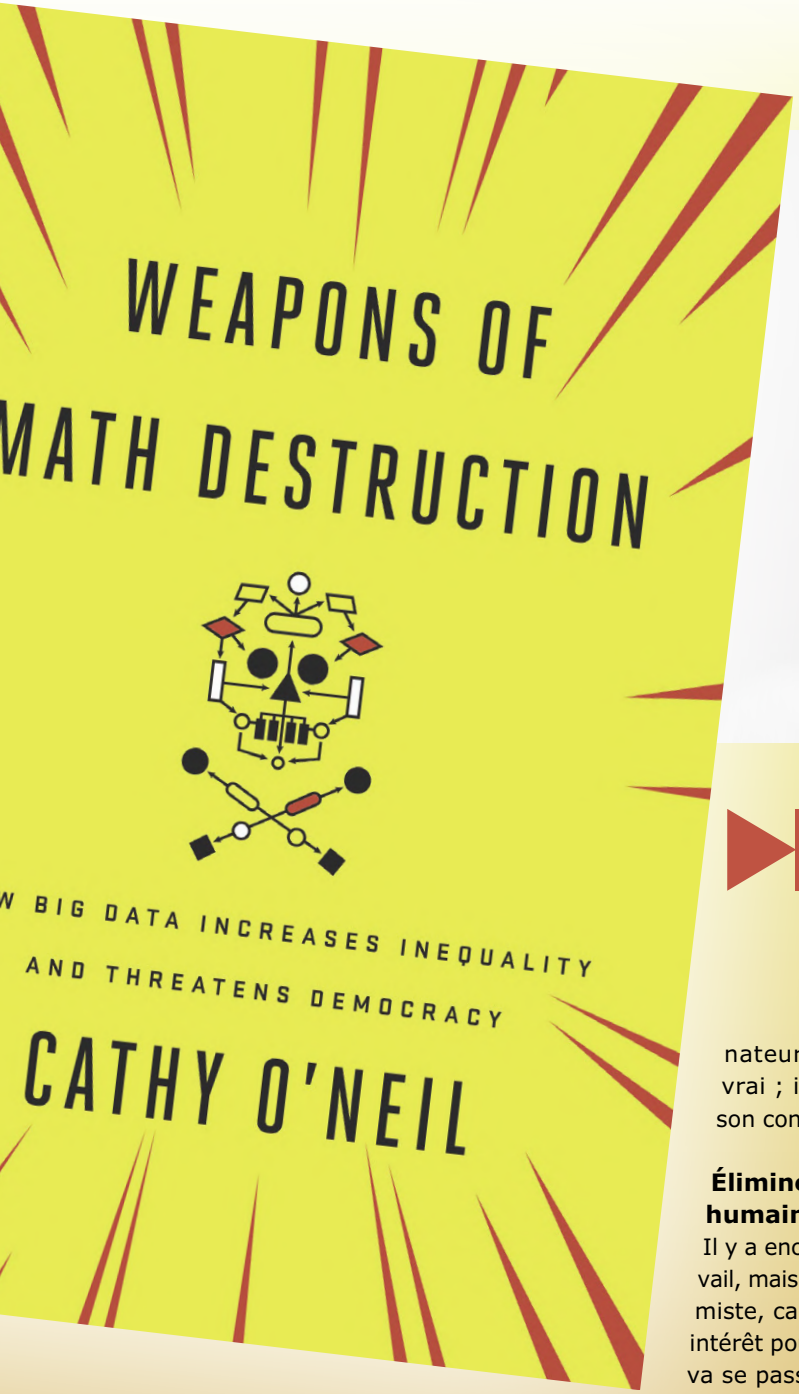
Cathy O'Neil n'a pas que des avis négatifs sur l'utilisation des mégadonnées (*big data*), bien au contraire : avec une approche correcte, le traitement de ces données peut aider utilement les prises de décision. Le challenge reste le reflet de nos valeurs éthiques dans les modèles mathématiques, sans oublier la transparence... Quelqu'un doit aussi avoir à répondre des conséquences néfastes de l'utilisation d'un algorithme. Il faut bien entendu que les applications soient testées de manière approfondie avant leur mise en œuvre.

« Ces tests étaient justement ce que d'aucuns voulaient éviter lors de la mise en œuvre d'algorithmes », nous dit encore Cathy. « D'une certaine façon, de nombreuses entreprises, organisations et autorités trouvaient cela trop beau pour être vrai : les processus étaient plus efficaces et la responsabilité en cas d'échec était limitée. Les décisions ont toujours été prises par des êtres humains, qui ont parfois des préjugés, leurs propres règles, ou tendance à favoriser leurs amis ; on peut le leur reprocher en cas de problème. L'utilisation d'algorithmes permettait tout à coup des prises de décision quasi automatiques, mais sans que l'on puisse blâmer qui que ce soit par la suite ».

Algorithmes sous la loupe

Le vent commence cependant à tourner, et de plus en plus de groupes d'intérêt réfléchissent à la manière dont on pourrait rendre les algorithmes honnêtes. C'est le cas de la ville d'Amsterdam, qui souhaite automatiser le traitement des plaintes des citoyens, comme celles relatives aux dépôts sauvages d'immondices sur la voie publique. La municipalité veut maintenir un traitement équitable dans l'ensemble de la ville ; les quartiers aisés pourraient en effet être favorisés, leurs habitants étant plus prompts à contacter le service de la propreté publique en cas de souci.

La ville d'Amsterdam a donc décidé de tester l'algorithme avant sa mise en œuvre définitive, et elle a réalisé des audits avec la société de conseil KPMG [2]. C'était aussi la raison de la présence de Cathy O'Neil. Sa société ORCAA [3] est spécialisée dans la réalisation d'audits sur les algorithmes – qui sont testés entre autres sur leur impartialité, leur transparence et leur exactitude, et elle a donc pu conseiller utilement la ville et KPMG.



Cathy O'Neil : « Il y a encore du travail, mais on y arrivera ! »

Cathy a rappelé que la conception d'algorithmes honnêtes n'est pas chose aisée. Il n'y a pas de solution miracle, et il est indispensable de réaliser des tests en situation opérationnelle : ce n'est pas parce qu'un modèle fonctionne à Amsterdam qu'il pourra être mis en œuvre à New York sans autre forme de procès ! « C'est ce que beaucoup ne comprennent pas », nous dit Cathy : « On croit qu'un algorithme peut être testé sur un ordinateur, mais ce n'est pas vrai ; il faut l'évaluer dans son contexte réel ».

Éliminer les erreurs humaines

Il y a encore beaucoup de travail, mais Cathy O'Neil est optimiste, car il y a désormais un intérêt pour ce sujet : « Ce qui va se passer, c'est que l'on va trouver une manière de responsabiliser la mise en œuvre d'algorithmes. Et il se pourrait même que cela aille plus loin qu'auparavant : les processus de prise de décision étant analysés minutieusement, ils deviendront plus transparents et impartiaux ».
Cathy illustre ce point avec un exemple : « Amazon a développé un algorithme pour le recrutement de personnel, mais il s'est avéré sexiste et n'a finalement pas été utilisé. C'est une bonne chose et un progrès certain qu'une évaluation ait été effectuée. On pourrait aller plus loin, car il est clair que c'est le processus sur lequel l'algorithme était basé qui est sexiste ! Amazon aurait pu modifier le modèle et dire qu'ils avaient amélioré un processus de prise de décision humain... On n'en est malheureusement pas encore là, mais on y arrivera : on corrigera les fautes que nous commettons, plutôt que de se contenter de les reproduire et de les propager ».

(180572-D-03 – version française : Jean-Louis Mehren)

(illustrations : https://weaponsofmathdestruction.com)

Liens

[1] Livre 'Weapons of math destruction', Cathy O'Neil : https://weaponsofmathdestructionbook.com
[2] Réseau KPMG : https://fr.wikipedia.org/wiki/KPMG
[3] Société ORCAA : www.oneilrisk.com

bienvenue dans votre e-choppe

la rédaction recommande

Passerelle Wi-Fi pour SmartScope de LabNation

Le SmartScope est un oscilloscope USB à deux voies, exceptionnellement pratique et polyvalent. Il peut fonctionner avec tous les systèmes informatiques, tablettes et smartphones utilisant Windows, OS X, iOS, Android et Linux. Il dispose désormais d'un module d'extension très commode qui lui permet de fonctionner sans fil grâce au Wi-Fi : la passerelle WiFi-Bridge.

Le module WiFi-Bridge de LabNation est un compagnon remarquable pour le SmartScope, et il est particulièrement intéressant pour tous ceux qui souhaitent effectuer des mesures sans fil à distance (relative). Je le recommande donc tout spécialement aux possesseurs d'un SmartScope !

Harry Baggen (labo d'Elektor)

www.elektor.fr/wifi-bridge-smartscope

Vos favoris :

1. SDR Hands-on Book (*livre en anglais*)
   www.elektor.fr/sdr-hands-on-book

2. Raspberry Pi 3 B+
   www.elektor.fr/rpi-3-plus
3. *Shield* SDR 2.0 d'Elektor
   www.elektor.fr/170515-91
4. Testeur USB Bakeey UM25C
   www.elektor.fr/bakeey-um25c
5. Fer à souder TS100 de Miniware
   www.elektor.fr/miniware-ts100
6. Kit PCBite
   www.elektor.fr/pcbite-kit

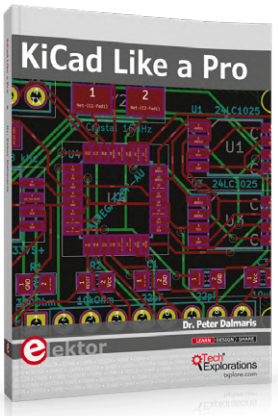
Embedded in Embedded
(livre en anglais)

Ce livre est une étude de cas dans le domaine de la conception embarquée. Il aborde le matériel, l'initialisation du processeur, le développement de pilotes de bas niveau et la conception de l'interface d'application pour un produit. Les explications de l'auteur reposent sur une application spécifique d'une carte de développement Cortex-M3. Le lecteur acquerra des compétences essentielles pour devenir un excellent développeur de produits.

Prix (membres) : 44,96 €

www.elektor.fr/embedded-in-embedded

KiCad Like a Pro
(livre en anglais)

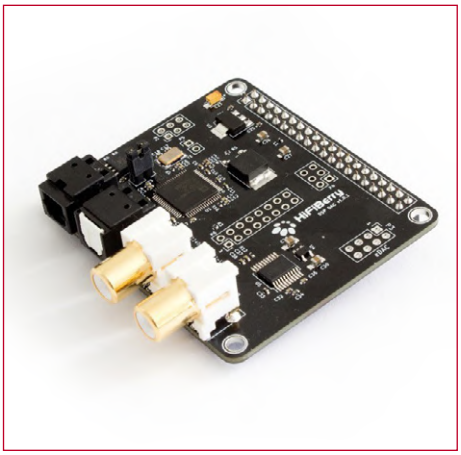


Ce livre (en anglais) vous apprendra à utiliser KiCad. Que vous soyez amateur ou ingénieur en électronique, grâce à ce livre, vous serez rapidement en mesure de dessiner vos schémas électroniques et de concevoir vos circuits imprimés. L'approche pratique de cet apprentissage repose sur quatre projets de difficulté progressive.

Prix (membres) : 35,96 €

www.elektor.fr/kicad-like-a-pro

HiFiBerry DAC+ DSP

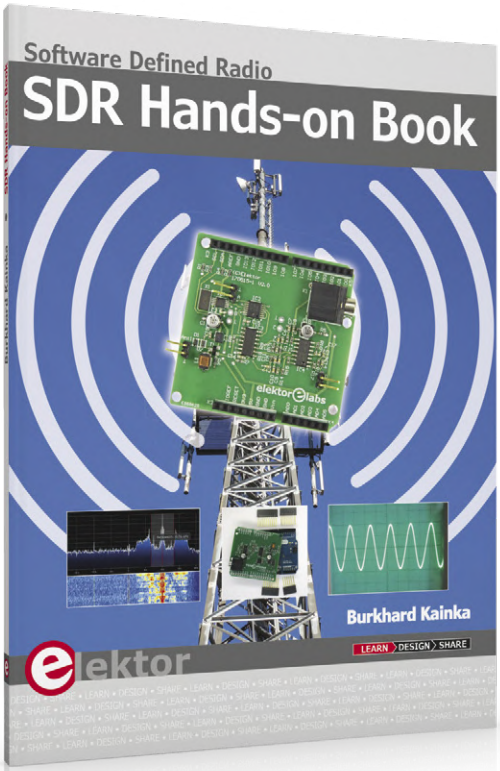


Le HiFiBerry DAC+ DSP est un convertisseur numérique-analogique à haute résolution pour le Raspberry Pi. Il combine un convertisseur N/A Burr-Brown à entrée et sortie numériques avec un puissant processeur de signal numérique.

Prix (membres) : 71,96 €

www.elektor.fr/hifiberry-dac-dsp

nouveau



SDR Hands-on Book

(livre en anglais)

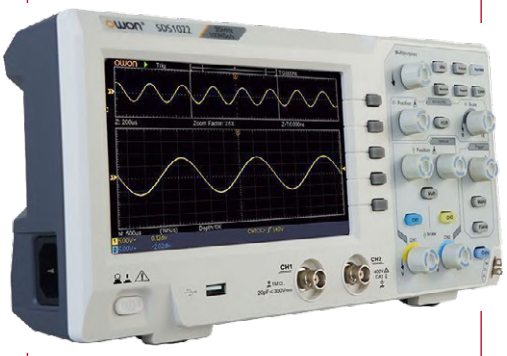
Le *shield* SDR d'Elektor (réf. 18515) permet de construire un récepteur polyvalent à ondes courtes jusqu'à 30 MHz. Avec une simple carte Arduino et le logiciel approprié, il est possible de recevoir des stations radio, des signaux en morse, des stations SSB et des signaux numériques.

Dans ce livre, notre auteur à succès et radioamateur enthousiaste, Burkhard Kainka décrit comment pratiquer la radio logicielle en utilisant le *shield* SDR d'Elektor. Il ne se contente pas de donner une formation théorique, il explique également comment utiliser de nombreux outils logiciels libres.

Membre : 26,96 €

www.elektor.fr/sdr-hands-on-book

OWON SDS1022



L'OWON SDS1022 est un oscilloscope à prix compétitif avec une fréquence d'échantillonnage de 1 Géch/s et une bande passante de 20 MHz. De plus, avec cet oscilloscope, vous disposez de deux canaux et de la possibilité de stocker 10.000 points de mesure. Il y a un connecteur USB à l'arrière pour l'exportation des résultats de mesure vers un PC, et un port USB à l'avant pour l'exportation vers une clé USB.

Prix (membres) : 170,96 €

www.elektor.fr/owon-sds1022

The Complete ESP32 Projects Guide

(livre en anglais)

L'objectif de ce livre en anglais est d'enseigner la programmation avec l'EDI Arduino et le langage MicroPython à l'aide de projets basés sur l'ESP32, en utilisant la très populaire carte de développement DevKitC ESP32. Les nombreux projets sont de niveau de difficulté croissant. Ils ont tous été testés et fonctionnent. Les explications sont complétées par des schémas de principe, les schémas des montages et les programmes.

Prix (membres) : 35,96 €

www.elektor.fr/esp32-projects-guide

Outil de mesure « mains libres » PCBite

PCBite est l'outil parfait pour manipuler un circuit imprimé pendant sa conception. Quatre puissants aimants posés sur un support en acier inoxydable permettent d'immobiliser des circuits imprimés de formes et de tailles différentes. Quatre bras également aimantés et flexibles sont terminés par une pointe de touche. Il suffit de les relier à un multimètre ou tout autre instrument de mesure pour garder les mains libres.

Prix (membres) : 121,46 €

www.elektor.fr/pcbite-kit

Hexadoku casse-tête pour elektorniciens

Votre magazine se termine toujours et encore par une grille de sudoku particulière. Éteignez le fer à souder, oubliez les écrans de PC et d'oscilloscope, rangez les pinces et les fils. Faites travailler votre matière grise d'une autre façon : attrapez un crayon pour remplir la grille d'hexadoku.
N'oubliez pas d'envoyer votre réponse pour tenter de gagner un bon d'achat.

Une grille hexadoku est composée de chiffres du système hexadécimal, de 0 à F. Remplissez le diagramme de 16 x 16 cases de telle façon que **tous** les chiffres hexadécimaux de 0 à F (0 à 9 et A à F) n'apparaissent **qu'une seule et unique fois** dans chaque rangée, colonne et carré de 4 x 4 cases (délimités par un filet gras).

Certains chiffres, déjà placés dans la grille, en définissent la situation de départ.
Pour participer, inutile de nous envoyer toute la grille, il suffit de nous donner **la série de chiffres** sur fond grisé.

Participez et gagnez !
Nous tirons au sort **cinq** 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 !

Où envoyer votre réponse ?
Envoyez votre réponse (les chiffres sur fond grisé) avec vos coordonnées par courriel, avant le **20 juillet 2019** à l'adresse **hexadoku@elektor.fr**

Les gagnants

La solution de la grille du numéro de mai/juin 2019 est **46F7C**.
Les cinq bons Elektor d'une valeur de **50 €** vont à : Guillaume **Demierbe** (Belgique), Michalis **Fostiropoulos** (Grèce), Ruth **Hanselmann** (Allemagne), Sune **Johansson** (Suède), Alexandr **Papazya** (Russie).

Bravo à tous les participants et félicitations aux gagnants !

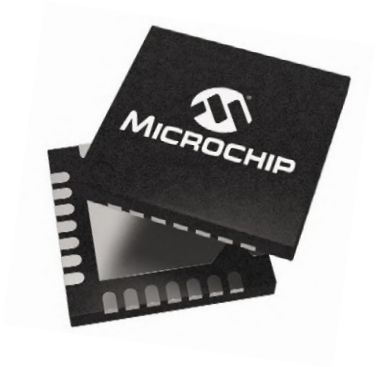
6			D	3		A		5						C	4
			C					E	3		9	0		5	6
		F			5	1	6						2		
5	1					9	B	6		C	F			3	
8						3		9	7		0	F		D	
		3			A	F	7	4		B		2			
A		D	7	2	6						E			4	
		4	E		B					2	3	6		8	0
1	6		F	4	2					0		C	8		
	2			5						3	4	1	0		D
			0		1		8	F	A	6			4		
	C		A	D		7	0		2						F
	8			A	C		F	D	6					0	2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

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

Différenciez votre application

Personnalisez votre projet embarqué grâce à nos briques intégrées très souples

Microchip sait que vos impératifs de conception sont uniques. C'est pourquoi nous vous proposons un ensemble robuste de périphériques intégrés pour un large éventail d'applications :



- Ajoutez des affichages percutants et des fonctions tactiles pour une interaction intuitive avec l'utilisateur
- Connectez votre application au monde, avec ou sans fil
- Commandez votre moteur ou votre système de conversion d'énergie
- Protégez les données de votre application

Personnalisez votre produit avec des périphériques intégrés, et réduisez les coûts et le temps de conception.

Personnalisez votre projet sur www.microchip.com/FlexibleFunctions

MICROCHIP

productronica
fast forward
the start-up platform
powered by Elektor

PARTICIPEZ AU CONCOURS POUR LANCER VOTRE STARTUP SUR LE SALON **PRODUCTRONICA** 2019

Participez à l'édition 2019!

12 au 15 novembre 2019 à Munich

informations détaillées :
www.elektormagazine.fr/p-ffwd

Platinum Sponsor:
DISTRELEC
Distribution with a difference

Productronica Fast Forward is brought to you by
productronica
elektor
INNOVATION STARTUP TRADE