ISSN 0181-7450 F : 15,50 €

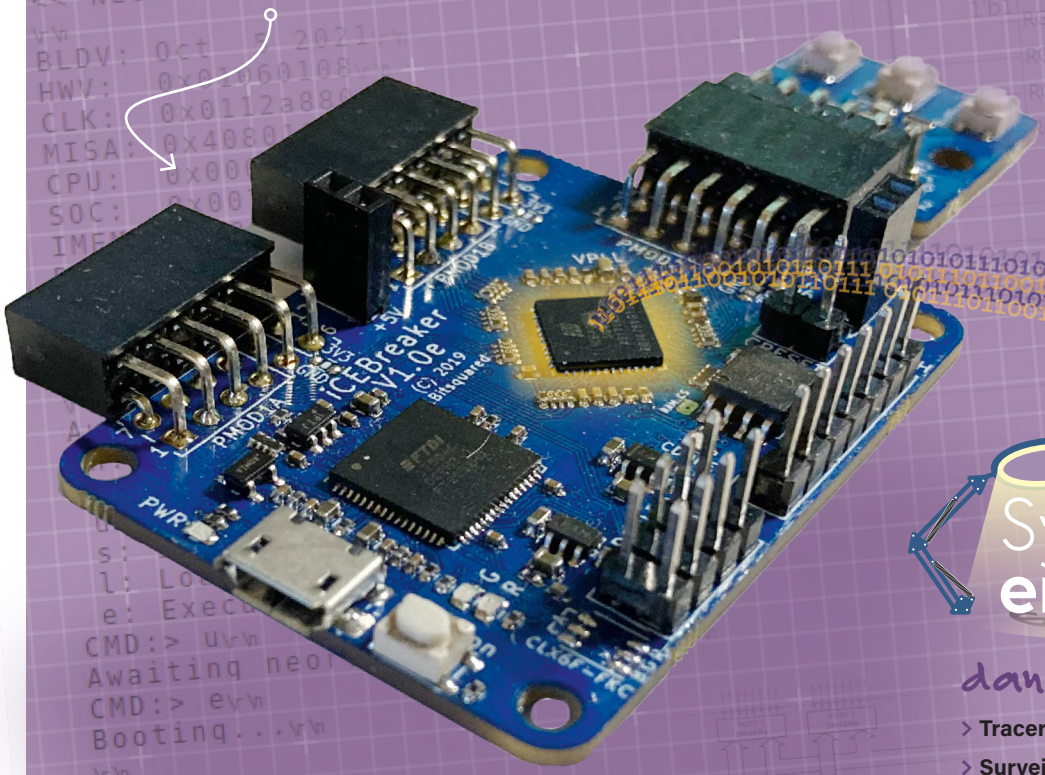
(B) 16,50 €
(CH) 29,00 FS
(CAN) 20.99 $ca
(And) 15,50 €
DOM surface 16,50 €
DOM avion 16,90 €
N Cal/S 2000 cfp

elektor 494

créer > partager > vendre

MARS & AVRIL 2022
ELEKTORMAGAZINE.FR

Construisez votre Processor *Open-Source* RISC-V

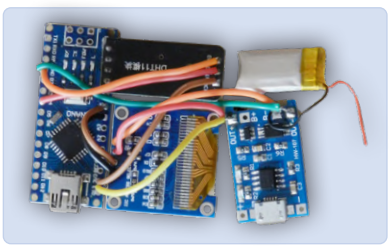


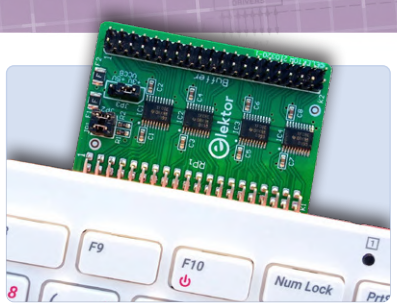
FOCUS SUR
Systèmes embarqués

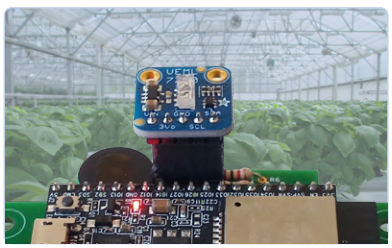
dans numéro :

- Tracer des graphiques avec Arduino
- Surveillance et débogage sans fil
- Identification des composants – trucs & astuces
- Réparation des batteries au lithium
- Approche DIY de la sécurité et de l'espionnage électronique
- Programmation d'automates avec le Raspberry Pi et le projet OpenPLC
- Lévitation magnétique : version compacte
- Questions d'éthique : trois questions fondamentales

et bien d'avantage !







L 19624 - 494 - F: 15,50 € - RD

elektor e-zine

Your dose of electronics

Chaque semaine où vous n'êtes pas abonné à l'e-zine d'Elektor est une semaine de grands articles et de projets électroniques qui vous manquent !

Alors, pourquoi attendre plus longtemps ? Abonnez-vous dès aujourd'hui à www.elektor.fr/ezine et recevez également le livre gratuit du projet Raspberry Pi !

À quoi pouvez-vous vous attendre ?

Éditorial
Chaque vendredi, vous recevrez les meilleurs articles et projets de la semaine. Nous couvrons les projets basés sur les MCU, l'IdO, la programmation, l'IA, et plus encore !

Promotionnel
Ne manquez pas les promotions de notre magasin, chaque mardi et jeudi nous avons une promotion spéciale pour vous.

Envoi des partenaires
Vous souhaitez rester informé des activités en cours dans le secteur ? Alors ce courriel vous donnera les meilleures informations. Non régulier, mais toujours le mercredi.

elektor
design > share > sell

45ème année
n° 494 – mars-avril 2022

ISSN 0181-7450
Dépôt légal : mars 2022
CPPAP 1125 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

Pour toutes vos questions :
service@elektor.fr

www.elektor.fr | www.elektormagazine.fr

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

Publicité :
Raoul Morreau
Tél. : +31 (0)6 4403 9907
Courriel : raoul.morreau@elektor.com

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

rédacteur en chef d'Elektor Magazine

Reporter, ce n'est pas annuler

Pour de nombreux professionnels de l'électronique, la fin de l'hiver est traditionnellement marquée par le salon *embedded world*, qui a eu lieu les années précédentes fin février ou début mars. Cet événement bien connu a tendance à être plus informel que l'énorme salon *electronica*, ce qui en fait un rendez-vous incontournable pour tous ceux qui manipulent à titre professionnel des micro-contrôleurs et du logiciel. (Et quels concepteurs ne s'occupent pas des deux de nos jours ?) J'aurais aimé pouvoir vous dire que vous pourrez nous rendre visite sur le stand Elektor en mars de cette année, mais le Coronavirus a encore contrecarré nos plans. Heureusement, les organisateurs du salon *embedded world* ont récemment annoncé que l'événement est « reporté mais pas annulé ». Avec de nouvelles dates (du 21 au 23 juin 2022), le parc d'exposition de Nuremberg fait une deuxième tentative. Touchons du bois !

En 2021, en pensant au salon *embedded world*, nous avions choisi le thème « systèmes embarqués » pour notre deuxième numéro de l'année 2022. Nous ne pouvions pas et ne voulions pas changer cela. Ainsi, pour le numéro de mai/juin, nous nous concentrerons sur l'Internet des Objets, dont la publication coïncidera probablement avec le salon *embedded world* désormais prévu en juin. Ce numéro sur l'IoT comprendra également des articles fouillés sur les nouveaux produits et les nouvelles tendances qui seront présentés à Nuremberg.

Que pouvez-vous attendre du numéro que vous avez entre les mains ? Nous vous présentons un assortiment de projets et d'articles de fond sur les petites puces informatiques. À la page 52, mon collègue Stuart Cording fait le point sur deux nouvelles tendances dans le monde de l'embarqué, le langage de programmation sécurisé Rust et la gestion des micrologiciels à la Toit. À partir de la page 6, Mathias Claussen, ingénieur au labo d'Elektor, explique comment « construire » et faire fonctionner votre propre processeur RISC-V. Pour les débutants et les makers, nous proposons un aperçu pratique des cartes bon marché basées sur la puce Raspberry Pi RP2040 (p. 28), ainsi qu'un petit atelier sur le traceur série de l'EDI Arduino (p. 15). Nous proposons également des projets tels qu'une carte tampon pour le Raspberry Pi (p. 24), un interrupteur d'éclairage sans contact (p. 42) et une interface sans fil/série (p.91). Bon travail !

notre équipe

Elektor est membre de la FIPP, une organisation qui « se développe depuis presque 100 ans pour réunir des propriétaires de médias et des créateurs de contenu du monde entier ».

Elektor est membre de VDZ (association d'éditeurs de magazines allemands) qui « représente les intérêts communs de 500 éditeurs allemands grand public et B2B. »

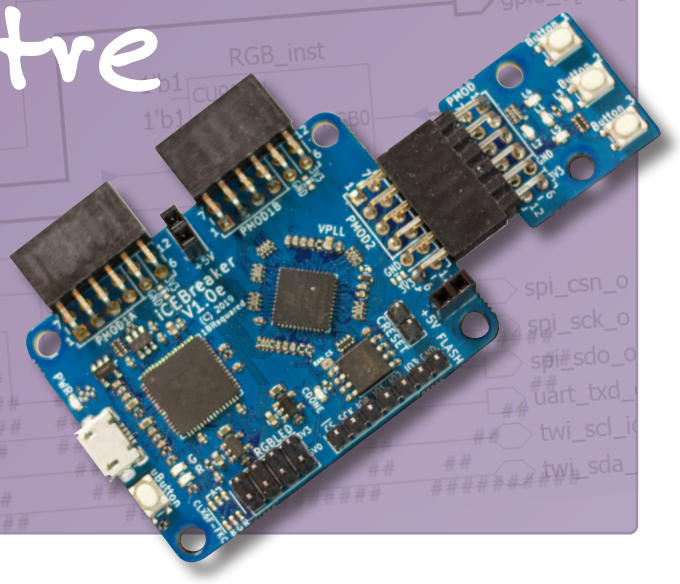
DANS CE NUMÉRO

Elektor n° 494
mars - avril 2022

Construisez votre contrôleur RISC-V

Débuter sur le noyau logiciel RISC-V NEORV32 pour FPGA à faible coût

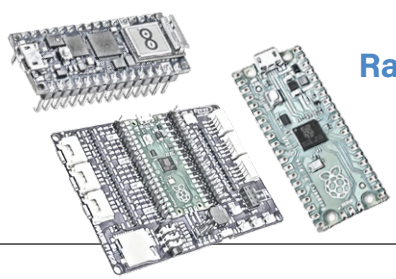
6



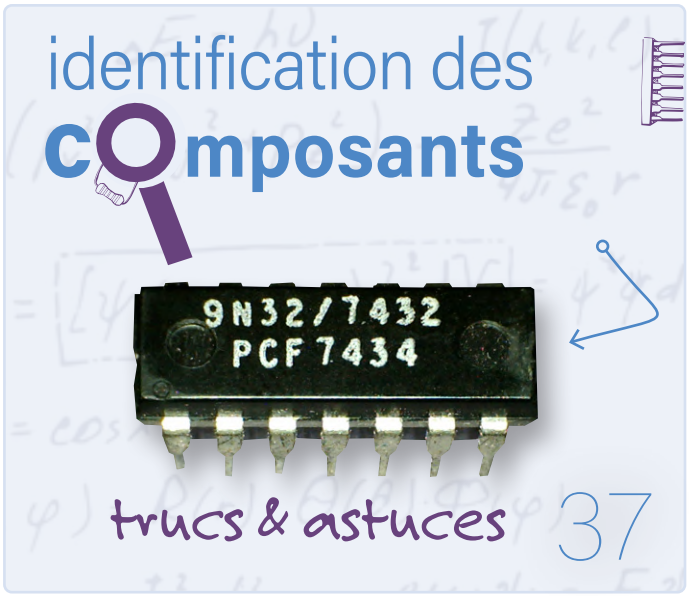
Rubriques

Articles de fond

Industrie



Cartes Raspberry Pi à gogo 28



Réalisations

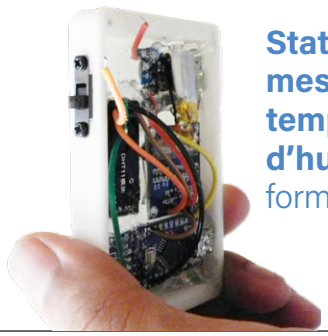
Bientôt dans ces pages

Le numéro de mai-juin 2022 d'Elektor
Vous retrouverez dans le prochain magazine Elektor l'habituel mélange stimulant de réalisations originales, de circuits soigneusement étudiés, d'articles de fond, de sujets nouveaux, de trucs et d'astuces pour les électroniciens actifs. Ce numéro s'intéressera tout particulièrement à l'Internet des Objets.

Quelques-uns des points forts :
- Premiers pas avec l'ESP32-C3 et l'Internet des Objets
- Double capteur de rayonnement à tube Geiger-Müller pour Arduino
- Instrument de mesure de CO_2 avec connexion au réseau
- Commutateur de lumière précis
- Cloud IoT à la sauce Arduino
- Outils de conception des filtres audio
- WinUI 3 : nouvelle bibliothèque d'interface utilisateur pour les applications Windows
- NB-IoT par la pratique

et bien d'autres choses encore !

Le numéro de mai-juin 2022 du magazine Elektor sera publié aux alentours du 5 mai 2022. La date d'arrivée du magazine papier chez les abonnés dépend des aléas d'acheminement. Le contenu et les titres des articles peuvent être modifiés.

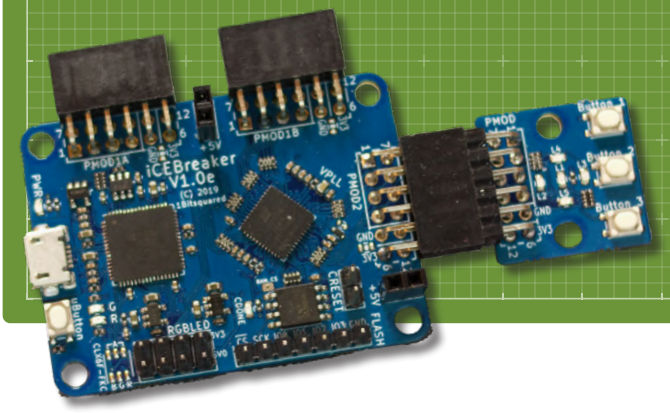


Station de mesure de température et d'humidité, au format de poche
96

FOCUS SUR
Systèmes embarqués

Construisez votre contrôleur RISC-V

Débuter sur le noyau logiciel RISC-V NEORV32 pour FPGA à faible coût



Mathias Claußen (Elektor)

Expérimenter le RISC-V vous tente ? C'est possible, sans puce câblée « en dur », avec le noyau logiciel RISC-V NEORV32 pour FPGA à faible coût.

Pour expérimenter avec un microcontrôleur *RISC-V*, il existe désormais des processeurs utilisant cette architecture de jeu d'instructions standard ouverte, par ex. le nouveau *ESP32-C3*. Mais il y a des alternatives à l'emploi d'une puce câblée « en dur » : *NEORV32* aborde par ex. le concept d'un *softcore* (noyau logiciel) RISC-V réalisable à l'aide d'un FPGA. Un tel processeur est un peu moins puissant qu'un processeur câblé, mais il est beaucoup plus souple et vous permet de tester différents projets et d'intégrer tout périphérique développé en interne dans votre matériel. Cette voie, riche d'enseignements, vous fera par ex. entrer dans les rouages intimes d'un processeur.

Une application pratique

Même ceux qui ont déjà expérimenté les FPGA feront vite face à des obstacles dès qu'il s'agira de configurer leur propre projet de petit processeur. L'ensemble du processus est assez complexe, même pour un ingénieur expérimenté, comme l'a montré notre série sur le projet SCCC de Martin Oßmann [1]. Heureusement, il n'est pas nécessaire de partir de zéro. On peut exploiter certaines solutions (quasiment clés en main) déjà développées par des experts qui les ont mises à disposition gratuitement.

L'une d'elles, également sous licence *open source*, sera utilisée ici. Cet article n'est pas du tout un cours complet sur le RISC-V ou les FPGA, mais il devrait accélérer le cycle d'apprentissage en montrant comment construire et faire fonctionner une 1ère application pratique aussi vite que possible.

FPGA, synthèse, *softcore*, RISC-V et compilateur

Si vous devez choisir un µcontrôleur polyvalent pour une application donnée, différents facteurs peuvent influencer votre décision. L'une des considérations de base est la gamme de périphériques intégrés proposée par le µcontrôleur. Celle-ci est figée dans le matériel par la version de la puce et ne peut être modifiée. Cette approche permet de fabriquer des puces peu coûteuses aux performances optimisées. Il en va différemment si vous basez votre propre contrôleur sur un FPGA. Un FPGA est constitué d'un ensemble de cellules logiques dites tables de consultation (*LUT* = *LookUp Table*) interconnectables à volonté via une matrice. La **figure 1** illustre les blocs existant dans une telle LUT. On a ici un élément LUT-4 avec 4 signaux d'entrée, une table de vérité, une bascule et un multiplexeur en sortie. La table de vérité permet de former toute porte élémentaire telle qu'un ET, un OU, un NON ou un OU EXCLUSIF. Associés à la matrice du FPGA, ces composants servent à créer des structures plus grandes comme des mémoires, additionneurs ou multiplexeurs, qui à leur tour peuvent être combinés pour former un système encore plus complexe tel qu'un processeur ou un système complet sur puce. Le FPGA peut être vu comme un jeu de briques de construction que l'on assemble à volonté pour construire par ex. un château, puis le démonter pour construire un pont ou toute autre structure, en réutilisant les mêmes briques.

Pour que le FPGA exécute une fonction spécifique, il doit être configuré de manière adéquate. Cependant, la fastidieuse tâche de création des LUT et de leur connexion à la matrice ne vous incombe pas. C'est celle

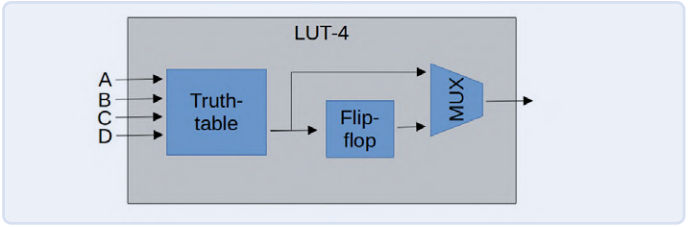


Figure 1. Flux de signaux dans une LUT-4.

des outils de synthèse FPGA. Il suffit de décrire la fonction souhaitée dans des langages tels que Verilog ou VHDL. Les outils de synthèse comprennent généralement ces deux langages de description du matériel. L'outil de synthèse connaît les caractéristiques du FPGA et, d'après le langage de description, il crée une suite de données binaires (ou flux de bits = *bitstream*) qui sert ensuite à configurer le FPGA. La **figure 2** montre la séquence de synthèse simplifiée d'un FPGA. La plupart des fabricants de FPGA offrent des outils gratuits fonctionnant sous Windows et Linux. Des solutions *open source* effectuent ce processus de synthèse pour certains FPGA. Elles s'exécutent en général sur système d'exploitation Unix, par ex. Linux ou macOS.
Un FPGA peut réaliser des fonctions logiques simples, des processeurs, voire des systèmes de processeurs… S'il abrite assez de LUT ! Verilog et VHDL savent aussi décrire ces derniers. Puisque le processeur n'est pas irréversiblement câblé dans le silicium du FPGA, on peut adapter sa fonction ou son comportement en modifiant la description matérielle. Un tel processeur est appelé *softcore* (noyau logiciel). Il en existe pour diverses architectures logicielles. Parfois, ces noyaux contiennent aussi des périphériques (interfaces de bus, etc.). L'architecture de processeur RISC-V à code source ouvert remporte un vif succès comme softcore. À l'heure où nous écrivons, le choix des MCU RISC-V câblés est encore gérable. Une première expérience de cette architecture peut ainsi être acquise en softcore en créant votre propre MCU RISC-V pour le tester et l'étudier.
Avec RISC-V, il n'y a ni frais de licence, ni accord de non-divulgation (NDA) ou autre accord de licence, tous généralement associés à l'exploitation des MCU des fabricants. RISC-V implique aussi la disponibilité de compilateurs de code source en C, notamment le compilateur GNU C (GCC). Et donc, les bibliothèques de base sont aussi en place.

NEORV32

Le projet NEORV32 de Stephan Nolting [4] montre qu'il n'y a pas besoin d'un FPGA coûteux pour construire son propre système sur puce (SoC = *System on Chip*). C'est un processeur compatible RISC-V avec tous les périphériques nécessaires pour fonctionner comme un SoC de type MCU sur un FPGA. Ce projet est entièrement implémenté en VHDL et n'est donc pas lié aux fabricants de FPGA. NEORV32 est entièrement *open source* et est aussi doté d'une documentation complète, d'un cadre logiciel et d'outils.
La **figure 3** montre les modules périphériques présents. Pêle-mêle on a des interfaces SPI, I²C et des UART, une interface WS2812, des GPIO, des unités MLI (PWM). Les utilisateurs débutants et avancés disposent ainsi d'un système complet avec environnement de développement intégral pour le NEORV32 incluant toutes les bibliothèques pour le matériel et les périphériques. De plus, des configurations types existent pour certaines cartes FPGA : vous êtes tout de suite opérationnel. Mais quelles sont ces cartes « prêtes à l'emploi » ? Est-ce difficile d'installer NEORV32 sur une carte FPGA non directement prise en charge ?

Choix du FPGA

En théorie, toute carte équipée d'un FPGA avec suffisamment de ressources convient, puisque NEORV32 n'utilise pas d'extensions

Figure 2. Étapes du processus de synthèse d'un FPGA.

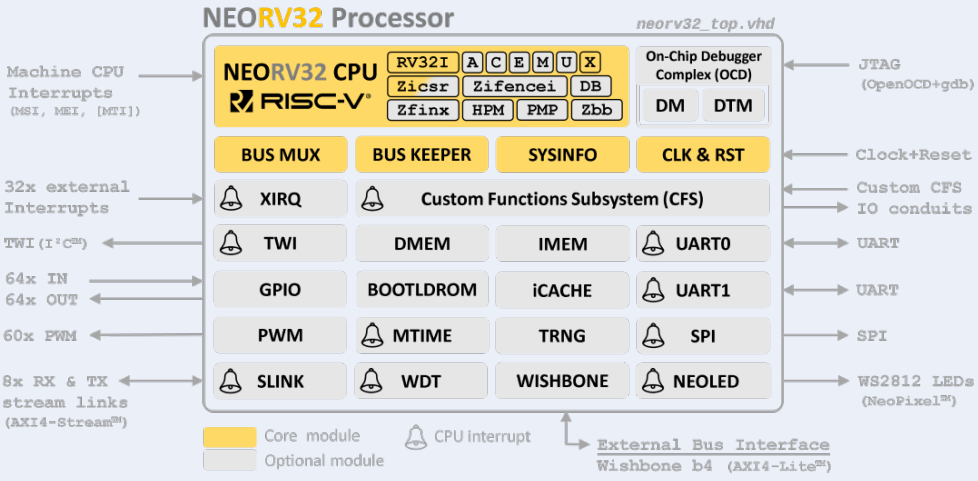


Figure 3. Schéma de principe des blocs fonctionnels de NEORV32 (source : Github / Nolting, S. [20]).

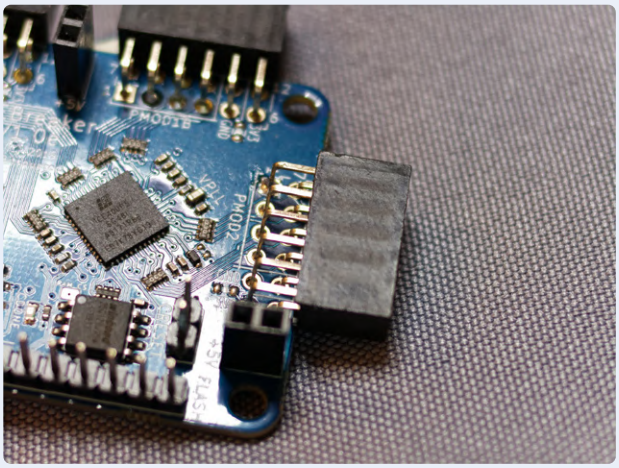


Figure 4. L'iCE40UP5K en format QFN mesurant 7×7 mm.

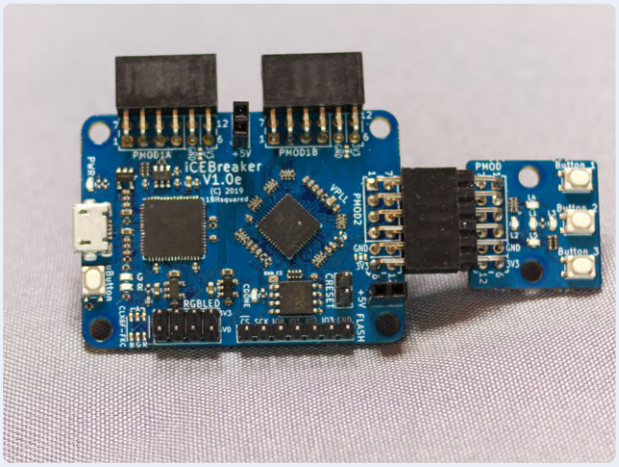


Figure 5. Carte iCEBreaker avec connecteur PMOD.

spécifiques au fabricant. Le FPGA iCE40UP5K de Lattice [5] équipe plusieurs cartes bon marché.

Le FPGA iCE40UP5K de Lattice est actuellement le plus puissant des iCE40 Ultra-Plus. Au total, il dispose de : 5280 LUT, 120 Kbits (15 Ko) de RAM EBR, de 1024 Kbits (128 Ko) de SPRAM et d'unités fonctionnelles câblées SPI et I²C. C'est une solide plate-forme pour construire vos propres projets. Ses caractéristiques, le type de boîtier de la puce et son faible coût contribuent à rendre ce FPGA intéressant. Le format QFN-48 de 7 × 7 mm (**fig. 4**) est beaucoup plus facile à exploiter que le BGA et son prix d'environ 5 € par puce le place dans une fourchette de prix intéressante. Aujourd'hui (octobre 2021), ce FPGA est proposé autour de 5 à 6 € l'unité chez tous les distributeurs, mais il est en rupture de stock et le délai de livraison peut atteindre 46 semaines.

Pour notre projet, vous n'aurez pas à concevoir un circuit imprimé pour le FPGA. Les FPGA iCEBreaker (**fig. 5**) et iCEBreaker Bitsy (**fig. 6**) de 1BitSquared [6] sont deux cartes de développement en matériel ouvert sur lesquelles le FPGA iCE40UP5K est déjà monté. En matériel ouvert, la carte UPduino V3.0 [7] de tinyVision.ai est une alternative (**fig. 7**).

Le projet NEORV32 est 100 % compatible avec l'UPduino V3.0 et comprend d'autres projets types. Les modifications requises sur la carte FPGA iCEBreaker sont très rapides à effectuer. Pour commencer, nous utiliserons l'UPduino V3.0, puis nous indiquerons comment adapter notre projet d'exemple pour qu'il fonctionne sur la carte iCEBreaker.

Avec ce FPGA, vous avez un SoC avec 64 Ko d'espace pour les applications, 64 Ko de RAM, une interface SPI, une I²C, une UART, 4 broches d'entrée, 4 de sortie, 3 MLI, ainsi que le cœur RV32IMAC fonctionnant à 18 MHz.

La chaîne d'outils

Vous pouvez prendre deux voies pour sélectionner la chaîne d'outils pour le Lattice iCE40up5k : soit les outils de Lattice [8], soit la suite OSS CAD entièrement basée sur des outils *open source* de YosysHQ [9]. Pour ce projet, nous choisirons la 2e voie, celle de l'*open source*. Cela signifie que notre système d'exploitation sera Ubuntu 20.04 LTS et que la suite OSS CAD sera utilisée pour synthétiser le NEORV32 pour l'iCE40UP5K.

Outre les outils du FPGA, un programme de démonstration du processeur RISC-V synthétisé sera aussi compilé ultérieurement : ce sera notre version du classique message « Hello World », envoyé à l'aide de l'UART. Là encore, des outils *open source* sont utilisés pour produire une chaîne d'outils adéquate (similaire à celle du Kendryte K210 [10]). Il existe un compilateur GNU C (GCC) et des bibliothèques complémentaires pour les périphériques NEORV32.

Mise en place

Comme mon collègue Clemens Valens l'expose dans sa vidéo [11], travailler avec des FPGA, c'est un peu comme cuisiner. Assurez-vous d'abord que tous les ingrédients et ustensiles (outils) nécessaires sont là. Pour éviter tout risque d'affecter l'installation principale de votre OS, vous pouvez travailler sur une machine virtuelle. Ici, nous supposons qu'une version 20.04 d'Ubuntu a été fraîchement installée sur un système AMD64. L'utilisation d'un Raspberry Pi devrait être possible, puisqu'Ubuntu 20.04 et l'OS du Raspberry Pi sont tous deux basés sur Debian, mais l'architecture peut engendrer de petites différences. Pour ce projet, je n'ai utilisé qu'Ubuntu 20.04 sur une machine AMD64.

Pour synthétiser le « matériel » du FPGA, la version à jour de la suite OSS CAD [12] doit être téléchargée dans le dossier de base. Ce fichier s'appelle *oss-cad-suite-linux-x64-xxxxxxxx.tgz*. Dans un terminal, décompressez ce fichier en utilisant `tar -xvzf oss-cad-suite-linux-x64-xxxxxxxx.tgz` et ensuite déplacez-le vers */opt* avec `sudo mv ~/oss-cad-suite /opt/`. Afin que le dossier soit accessible ultérieurement, définissez les droits avec `chmod 777 /opt/oss-cad-suite -R`. La bibliothèque *libgnat-9* est également requise, installez-la avec `sudo apt install libgnat-9`. Les outils FPGA sont alors en place.

Compilateur

Nous avons ensuite besoin des fichiers associés à NEORV32 [4] depuis le dépôt GitHub. À cet effet, nous allons installer *git* en entrant `sudo apt install git`. Puis : `git clone https://github.com/stnolting/neorv32.git ~/neorv32` clône le dépôt de NEORV32.

Pour communiquer plus tard avec le NEORV32, un programme de terminal sera nécessaire. En général, j'utilise un programme éprouvé : HTerm de Tobias Hammer. Il peut être téléchargé depuis sa page d'accueil [13] avec `wget www.der-hammer.info/terminal/hterm-linux-64.tgz`. Ensuite `mkdir ~/hterm && tar -xvf hterm-linux-64.tgz -C ~/hterm` extrait le contenu du fichier *tgz* dans le dossier *~/hterm*. Pour qu'un utilisateur puisse ensuite accéder aux interfaces série, il faut l'ajouter comme membre du groupe *dialout*. À cet effet, utilisez le terminal et entrez `sudo adduser $(whoami) dialout`.

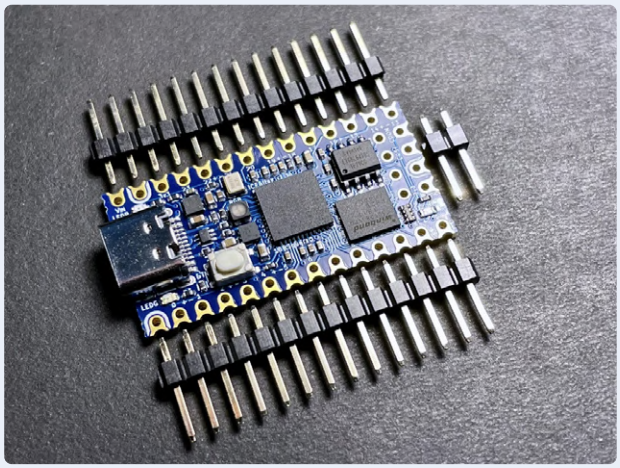


Figure 6. iCEBreaker Bitsy (source : https://cdn.shopify.com/s/files/1/1069/4424/products/IMG_3859_large.jpg / 1BitSquare).

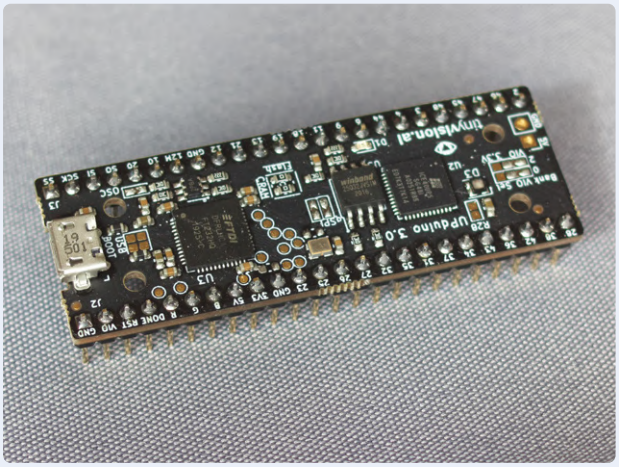


Figure 7. L'UPduino V3.0 avec les barrettes de broches installées.

Le compilateur C doit maintenant lui-même subir une compilation. Dans les exemples pour iCE40up5k, NEORV32 est configuré en RV32IMAC, les commandes de multiplication et de division d'entiers sont donc disponibles (cf. l'encadré **Convention de nomenclature RISC-V**). Le compilateur doit être compilé selon cette architecture particulière et les extensions de commandes ; la documentation de NEORV32 [14] indique pourquoi cette adaptation est importante.
Avec un terminal, tapez `cd ~` pour aller dans le dossier de base puis tapez `git clone https://github.com/riscv/riscv-gnu-toolchain --recursive` pour cloner la chaîne d'outils RISC-V. Il faudra encore installer quelques compléments logiciels comme suit :

```
sudo apt-get install autoconf automake autotools-
   dev curl python3 libmpc-dev libmpfr-dev libgmp-
   dev gawk build-essential bison flex texinfo gperf
   libtool patchutils bc zlib1g-dev libexpat-dev
```

Enfin, le compilateur et les bibliothèques peuvent être compilés.
Avec les processeurs RISC-V, les commandes prises en charge sont hiérarchisées. Cela signifie, par ex., que le code compilé pour un modèle RV32I sera exécutable sur un modèle RV32IMAC, mais pas l'inverse. C'est pourquoi il vaut mieux compiler d'abord la chaîne d'outils pour qu'elle soit compatible avec le modèle de plus petit dénominateur commun. À l'aide du terminal, tapez `cd ~/riscv-gnu-toolchain` pour aller au répertoire de la chaîne d'outils clonée puis `./configure --prefix=/opt/riscv --with-arch=rv32i --with-abi=ilp32` pour préconfigurer la chaîne d'outils de compilation. Dès lors, nous pouvons lancer la tâche de compilation avec `sudo make`. La chaîne d'outils sera sur */opt/riscv*. Pour que tout le monde accède au compilateur, il faut modifier les droits de */opt/riscv*. Accordez l'accès à tout le monde en tapant `chmod 777 /opt/riscv -R` sur un terminal.
Pour la suite OSS CAD et la chaîne d'outils RISC-V GCC, il faut effectuer une modification de la variable *path* du fichier */etc/environment*. Tapez `sudo nano /etc/environment` pour ouvrir le fichier et après `PATH=»` entrez la chaîne `/opt/oss-cad-suite/bin:/opt/riscv/bin:`, ensuite enregistrez le fichier.
La plupart des cartes FPGA gère l'interface de programmation avec une puce FT232H de FTDI. Pour qu'un utilisateur puisse y accéder sans droits root, il faut créer une règle *udev* appropriée pour la puce FTDI. Via le terminal, entrez `sudo nano /etc/udev/rules.d/53-lattice-ftdi.rules`. Un nouveau fichier s'ouvre en écriture ; il faut y entrer :

```
ACTION=="add", ATTR=="0403", ATTR=="6010",
   MODE:="666"
ACTION=="add", ATTR=="0403", ATTR=="6014",
   MODE:="666"
```

Puis enregistrez le fichier. Les préparatifs sont terminés et on peut commencer la synthèse puis le téléchargement de notre premier programme de test. Pour que tous les paramètres prennent effet, il faut redémarrer. Pour une utilisation avec VHDL et Verilog, il est préférable d'utiliser un éditeur avec coloration syntaxique.

NEORV32 pour le FPGA

À l'état hors tension, le FPGA n'est pas configuré. À la mise sous tension, le iCE40UP5K lit la description des connexions internes sur une flash SPI externe. Celle-ci doit donc d'abord être stockée dans la flash SPI de l'UPDuino V3.0 ou de l'iCEBreaker.
Ici, nous créons un projet type pour la carte UPduino V3.0 contenant le processeur et les périphériques. Le système créé devrait pouvoir fonctionner directement sur le FPGA.
Via un terminal, tapez `cd ~/neorv32/setups/osflow/` pour vous rendre dans le dossier d'exemples des outils *open source*. Pour démarrer la synthèse et donc la création du *bitstream* pour le FPGA, tapez `make BOARD=UPduino UP5KDemo`. C'est tout ! Mais il faut patienter, le processus de synthèse peut prendre du temps… Le dossier *~/neorv32/setups/osflow/* contient désormais un fichier nommé *neorv32_UPduino_v3_UP5KDemo.bit*. Le flux binaire interne du fichier décrit les interconnexions des blocs logiques de base à réaliser dans le FPGA. C'est ce flux binaire qu'il faut maintenant écrire dans la flash SPI de la carte FPGA.
Pour cela, nous pouvons relier la carte UPduino au PC par un câble USB. Dans le terminal, tapez `iceprog ~/neorv32/setups/osflow/neorv32_UPduino_v3_UP5KDemo.bit` pour lancer la programmation de la flash SPI externe, et la configuration est ensuite chargée dans le FPGA. Cela signifie que notre système RISC-V est maintenant configuré dans l'UPduino V3.0, et que nous pouvons lui confier des logiciels. Comme indiqué au début, 64 Ko de mémoire sont disponibles pour les applications et 64 Ko en RAM. Comme périphériques, nous avons trois

```
MEMORY
{
/* section base addresses and sizes have to be a multiple of 4 bytes */
/* ram section: first value of LENGTH => data memory used by bootloader (fixed!); second value of LENGTH => *physical* size of data memory */
/* adapt the right-most value to match the *total physical data memory size* of your setup */

  ram  (rwx) : ORIGIN = 0x80000000, LENGTH = DEFINED(make_bootloader) ? 512 : 8*1024

/* rom and iodev sections should NOT be modified by the user at all! */
/* rom section: first value of ORIGIN/LENGTH => bootloader ROM; second value of ORIGIN/LENGTH => maximum *logical* size of instruction memory */

  rom   (rx) : ORIGIN = DEFINED(make_bootloader) ? 0xFFFF0000 : 0x00000000, LENGTH = DEFINED(make_bootloader) ? 32K : 2048M
  iodev (rw) : ORIGIN = 0xFFFFFE00, LENGTH = 512

}
/* ************************************************************************** */
```

Figure 8. Modifications de configuration de la taille de la RAM.

```
user@user-VirtualBox:~/neorv32/sw/example/hello_world$ make exe
Memory utilization:
   text    data     bss     dec     hex filename
   5576       0     116    5692    163c main.elf
Executable (neorv32_exe.bin) size in bytes:
5588
user@user-VirtualBox:~/neorv32/sw/example/hello_world$
```

Figure 9. L'exécutable NEORV32_exe.bin est créé.

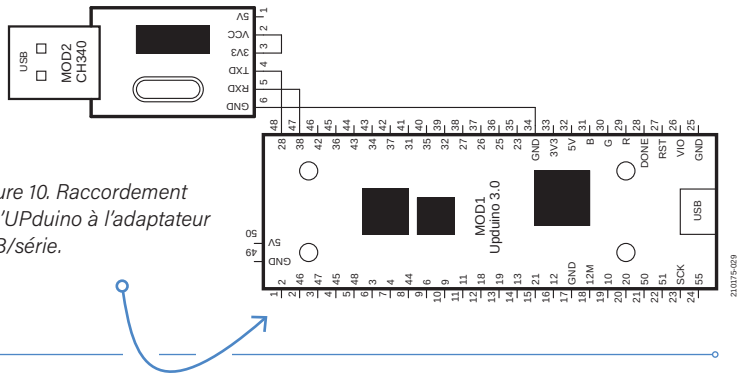


Figure 10. Raccordement de l'UPduino à l'adaptateur USB/série.

interfaces (SPI, I²C, UART), quatre entrées et quatre sorties, ainsi que trois sorties MLI. Le CPU synthétisé ici est un modèle RV32IMAC qui fonctionne à 18 MHz. Le SoC dans le FPGA a aussi un petit chargeur d'amorçage accessible via l'UART.

Hello World

Le FPGA est maintenant équipé du NEORV32 et la première démo *Hello World* peut être compilée et téléchargée sur le RISC-V. Le NEORV32 est configurable, la taille réelle de la RAM doit donc être définie dans le script du linker. Pour ce faire, entrez `nano ~/neorv32/sw/common/neorv32.ld` avec le terminal et à la ligne 62 :

```
ram (rwx) : ORIGIN = 0x80000000, LENGTH =
   DEFINED(make_bootloader) ? 512 : 8*1024
```

à la place de :

```
ram (rwx) : ORIGIN = 0x80000000, LENGTH =
   DEFINED(make_bootloader) ? 512 : 64*1024
```

(**fig. 8**)

Le compilateur RISC-V est maintenant prêt à l'emploi. Dans un terminal ouvert, vous pouvez aller dans le dossier du programme *Hello World* en tapant :

```
cd ~/neorv32/sw/example/hello_world
```

Pour produire un fichier exécutable pouvant être chargé dans le NEORV32, il suffit d'entrer `make exe`. Ce fichier s'appellera *neorv32_exe.bin* (**fig. 9**). Pour pouvoir exécuter NEORV32, il faut alors le charger sur la carte à l'aide du bootloader intégré. Il reçoit les données via l'UART (19200 bauds, 8 bits de données, 1 bit d'arrêt, sans parité ni contrôle de flux). Si une carte UPduino V3.0 est utilisée, un convertisseur USB-série externe est nécessaire, par ex. le convertisseur CH340 de la boutique d'Elektor. (Reportez-vous à l'encadré **Produits**.) Il doit être connecté comme indiqué à la **figure 10**.

HTerm est utilisé pour le téléchargement lui-même. On peut le lancer depuis un terminal avec `~/hterm/hterm` ; une fenêtre comme sur la **figure 11** devrait apparaître. Il faut sélectionner le convertisseur USB-série comme port, en général il s'appelle */dev/ttyUSB0* ; cela peut toutefois dépendre de la configuration matérielle et de l'adaptateur sélectionné.

Après raccordement de l'adaptateur série USB, l'UPduino peut être alimenté en tension et le message du bootloader doit apparaître (**fig. 12**). Si aucun caractère n'est envoyé à temps (en moins de 8 s) au bootloader, il tente de démarrer automatiquement à partir de la flash SPI qui ne contient encore aucun logiciel. Sinon, le bootloader passe en mode commande. Tapez `u` pour activer le téléchargement dans le bootloader et cliquez sur le bouton *Select File* dans HTerm. Comme on peut le voir sur la **figure 13**, il faut sélectionner le fichier *neorv32_exe.bin* dans le dossier *~/neorv32/sw/example/hello_world* puis le télécharger. Une fois tout terminé, le bootloader renvoie `OK` (cf. **fig. 14**) et il suffit d'entrer `e` pour exécuter le programme.

Le résultat apparaît à la **figure 15**. En fait, le programme a été stocké dans la « ROM » basée sur la RAM du NEORV32 et non dans la flash SPI. Cela prolonge la durée de vie de la flash SPI en y réduisant le nombre de cycles d'écriture. L'inconvénient c'est que le programme doit être rechargé à chaque redémarrage de NEORV32. Pour charger le programme automatiquement, il faut que le bootloader le mette

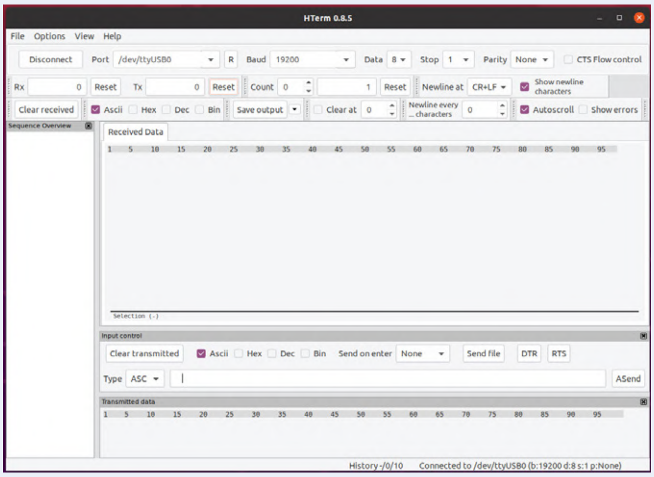


Figure 11. Aperçu d'une fenêtre HTerm.

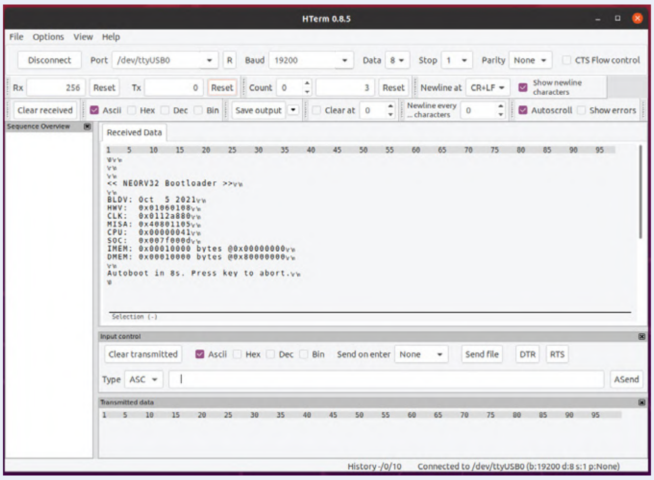


Figure 12. Chargeur de démarrage NEORV32.

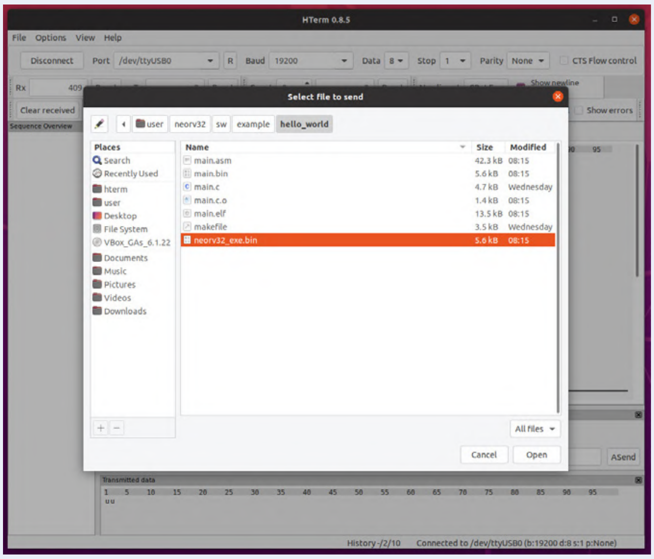


Figure 13. Fenêtre de téléchargement upload neorv32_exe.bin.

```
Available CMDs:\r\n
 h: Help\r\n
 r: Restart\r\n
 u: Upload\r\n
 s: Store to flash\r\n
 l: Load from flash\r\n
 e: Execute\r\n
CMD:> u\r\n
Awaiting neorv32 exe.bin... OK\r\n
CMD:>
```

Figure 14. Téléchargement réussi.

```
<< NEORV32 Bootloader >>\r\n
\r\n
BLDV: Oct  5 2021\r\n
HWV:  0x01060108\r\n
CLK:  0x0112a880\r\n
MISA: 0x40801105\r\n
CPU:  0x00000041\r\n
SOC:  0x007f000d\r\n
IMEM: 0x00010000 bytes @0x00000000\r\n
DMEM: 0x00010000 bytes @0x80000000\r\n
\r\n
Autoboot in 8s. Press key to abort.\r\n
Aborted.\r\n
\r\n
Available CMDs:\r\n
 h: Help\r\n
 r: Restart\r\n
 u: Upload\r\n
 s: Store to flash\r\n
 l: Load from flash\r\n
 e: Execute\r\n
CMD:> u\r\n
Awaiting neorv32 exe.bin... OK\r\n
CMD:> e\r\n
Booting...\r\n
\r\n
\r\n
                                                                                       ##                                       \r\n
                                                                                       ##         ##   ##   ##                  \r\n
 ##     ##   #########   ########    ########   ##      ##   ########    ########      ##       ################                \r\n
####    ##  ##          ##      ##  ##      ##  ##      ##  ##      ##  ##      ##     ##     ####            ####              \r\n
## ##   ##  ##          ##      ##  ##      ##  ##      ##          ##         ##      ##       ##   ######   ##                \r\n
##  ##  ##  #########   ##      ##  #########   ##      ##      #####        ##        ##     ####   ######   ####              \r\n
##   ## ##  ##          ##      ##  ##    ##     ##    ##           ##     ##          ##       ##   ######   ##                \r\n
##    ####  ##          ##      ##  ##     ##     ##  ##    ##      ##   ##            ##     ####            ####              \r\n
##     ##    #########   ########   ##      ##      ##       ########   ##########     ##       ################                \r\n
                                                                                       ##         ##   ##   ##                  \r\n
                                                                                       ##                                       \r\n
Hello world! :)\r\n
```

Figure 15. « Hello World » de NEORV32.

Convention de nomenclature RISC-V

RISC-V est un terme générique pour les diverses variantes de l'architecture. Notre collègue d'Elektor Stuart Cording a écrit un excellent article (« RISC-V quesaco », Elektor 7-8/2021, [2]) qui donne de plus amples détails. RISC-V décrit une architecture de jeu d'instructions (en anglais ISA = *Instruction Set Architecture*) qui classe les processeurs en versions 32, 64 ou 128 bits. Un processeur à 32 bits a un nom commençant par RV32 pour 32 bits, et donc RV64 indique un processeur à 64 bits. En outre, un certain nombre de lettres viennent s'ajouter pour indiquer les commandes et extensions que le processeur peut gérer. Ces lettres vont de A à Z ; elles sont explicitées dans la spécification RISC-V actuelle [3]. Les performances des processeurs sont fonction des commandes et extensions prises en charge.

dans la flash SPI. En plus de la démo « Hello World » de base, il y a d'autres exemples à découvrir, dont un FreeRTOS complet déjà présenté dans Elektor [15]. Cela vaut la peine d'aller voir le dossier *~/neorv32/sw/example*, où plusieurs autres exemples figurent dans des dossiers individuels.

Un nouvel environnement FPGA iCE40UP5K

L'utilisation de la carte iCEBreaker montrera comment adapter NEORV32 à d'autres cartes iCE40up5k. Les caractéristiques du SoC lui-même ne sont pas modifiées, seule l'affectation des broches sur le FPGA est adaptée de sorte qu'une carte iCEBreaker soit utilisable et qu'un bitstream adéquat soit produit.
Pour ce faire, il faut éditer le Makefile dans *~/neorv32/setup/osflow*. Le fichier s'ouvre avec l'éditeur de texte de votre choix et on ajoute la nouvelle carte comme cible. Pour la carte iCEBreaker, il faut écrire ce qui suit à la ligne 72 :

```
iCEBreaker:
$(MAKE) \
BITSTREAM=neorv32_$(BOARD)_$(DESIGN).bit \
NEORV32_MEM_SRC="devices/ice40/neorv32_imem.ice40up_
   spram.vhd devices/ice40/neorv32_dmem.ice40up_
   spram.vhd" \
run
```

Cela garantit que la carte est référencée dans le Makefile primaire. Dans *~/neorv32/setup/osflow/boards* il faut aussi un fichier *iCEBreaker.mk* avec le contenu suivant :

```
.PHONY: all

all: bit
echo "! Built $(IMPL) for $(BOARD)"
```

Les Makefiles sont alors prêts, mais il manque encore deux fichiers VHDL : *neorv32_iCEBreaker_BoardTop_UP5KDemo.vhd* et *neorv32_iCEBreaker_BoardTop_MinimalBoot.vhd* qui doivent être déposés sur *~/neorv32/setup/osflow/board_tops/*. Comme leur contenu est quasi identique à celui des fichiers *neorv32_UPduino_BoardTop_UP5KDemo.vhd* et *neorv32_UPduino_BoardTop_MinimalBoot.vhd*, copions d'abord ceux-ci en les renommant. Par ex. utilisons un terminal et entrons :

```
cp ~/neorv32/setups/osflow/board_tops/neorv32_
   UPduino_BoardTop_MinimalBoot.vhd
~/neorv32/setups/osflow/board_tops/neorv32_
   iCEBreaker_BoardTop_MinimalBoot.vhd
```

et :

```
cp ~/neorv32/setups/osflow/board_tops/neorv32_
   UPduino_BoardTop_UP5KDemo.vhd
~/neorv32/setups/osflow/board_tops/neorv32_
   iCEBreaker_BoardTop_UP5KDemo.vhd
```

Il faut apporter quelques modifications aux deux fichiers *neorv32_iCEBreaker_BoardTop_UP5KDemo.vhd* et *neorv32_iCEBreaker_BoardTop_MinimalBoot.vhd*. Dans le fichier *neorv32_iCEBreaker_BoardTop_UP5KDemo.vhd*, changez la ligne 42 en `entity neorv32_iCEBreaker_BoardTop_UP5KDemo is` et la ligne 68 en `architecture neorv32_iCEBreaker_BoardTop_UP5KDemo_rtl of neorv32_iCEBreaker_BoardTop_UP5KDemo is`. Dans le fichier *neorv32_iCEBreaker_BoardTop_MinimalBoot.vhd*, changez la ligne 42 en `entity neorv32_iCEBreaker_BoardTop_MinimalBoot is` et la ligne 54 en `architecture neorv32_iCEBreaker_BoardTop_MinimalBoot_rtl of neorv32_iCEBreaker_BoardTop_MinimalBoot is`.

En dernière étape, placez le fichier de contraintes *iCEBreaker.pcf* dans *~/neorv32/setups/osflow/constraints/*. Le **listage 1** donne le contenu de ce fichier. *ICEBreaker.pcf* affecte les fonctions à diverses broches du FPGA. Il englobe aussi directement le convertisseur USB/série présent sur la carte iCEBreaker, et achemine les boutons et les LED sur les broches d'E/S de NEORV32. Une chose manque : un bouton reset. Bien qu'il soit défini dans le fichier de contraintes, sa fonction n'est pas référencée ici.
Une fois les changements nécessaires effectués, il ne reste qu'à produire le bitstream comme avec la carte UPduino. Pour cela, tapez `cd ~/neorv32/setups/osflow` sur un terminal pour atteindre le dossier *osflow* ; ensuite, `make BOARD=iCEBeaker UP5KDemo` démarre le processus qui produit les bitstreams ; pour les télécharger vers l'iCEBreaker (comme avec l'UPduino), nous utilisons `iceprog ~/neorv32/setups/osflow/neorv32_iCEBreaker_UP5KDemo.bit`.
Le chargement du logiciel dans le NEORV32 est là aussi pris en charge par le bootloader intégré. La carte iCEBreaker évite le recours à un convertisseur USB/série externe, car nous utilisons la 2e voie du convertisseur intégré sur celle-ci.

Bouton de réinitialisation pour le NEORV32

Pour redémarrer le NEORV32, il faut couper brièvement l'alimentation électrique, puis la rallumer. Cela devient fastidieux à la longue, et l'iCEBreaker ayant assez de boutons, on peut utiliser l'un d'eux pour le réinitialiser (*reset*). Par ex., le *uButton* près de la prise micro-USB de la carte ; il est connecté à la broche 10 du FPGA.
Le NEORV32 possède une entrée interne *rstn_i* (reset) reliée à la sortie *lock* de la PLL (*Phase Lock Loop* = boucle à verrouillage de phase) de l'horloge du système. Si la PLL se déverrouille, elle envoie un reset au

Listage 1. iCEBreaker.pcf

```
#UART (uart0)
ldc_set_location -site {9} [get_ports uart_txd_o]
ldc_set_location -site {6} [get_ports uart_rxd_i]

#SPI - on-board flash
ldc_set_location -site {14} [get_ports flash_sdo_o]
ldc_set_location -site {15} [get_ports flash_sck_o]
ldc_set_location -site {16} [get_ports flash_csn_o]
ldc_set_location -site {17} [get_ports flash_sdi_i]

#SPI - user port
ldc_set_location -site {43} [get_ports spi_sdo_o]
ldc_set_location -site {38} [get_ports spi_sck_o]
ldc_set_location -site {34} [get_ports spi_csn_o]
ldc_set_location -site {31} [get_ports spi_sdi_i]

#TWI
ldc_set_location -site {2} [get_ports twi_sda_io]
ldc_set_location -site {4} [get_ports twi_scl_io]

#GPIO - input
ldc_set_location -site {18} [get_ports {gpio_i[0]}]
ldc_set_location -site {19} [get_ports {gpio_i[1]}]
ldc_set_location -site {20} [get_ports {gpio_i[2]}]
ldc_set_location -site {28} [get_ports {gpio_i[3]}]

#GPIO - output
ldc_set_location -site {25} [get_ports {gpio_o[0]}]
ldc_set_location -site {26} [get_ports {gpio_o[1]}]
ldc_set_location -site {27} [get_ports {gpio_o[2]}]
ldc_set_location -site {23} [get_ports {gpio_o[3]}]

#RGB power LED
ldc_set_location -site {39} [get_ports {pwm_o[0]}]
ldc_set_location -site {40} [get_ports {pwm_o[1]}]
ldc_set_location -site {41} [get_ports {pwm_o[2]}]

#Reset
ldc_set_location -site {10} [get_ports
   {user_reset_btn}]
```

NEORV32. La solution la plus propre serait d'envoyer le signal de réinitialisation de l'utilisateur au même point que le signal *lock* de la PLL de sorte le NEORV32 reçoive le reset de l'un ou de l'autre. La solution la plus simple et la plus rapide est d'utiliser l'entrée reset de la PLL. La connexion du uButton, *user_reset_btn* avec cette entrée est sur la **figure 16**. Dès lors, si on appuie sur uButton, la PLL est réinitialisée, le signal *lock_o* passe au niveau bas et le NEORV32 est réinitialisé. Afin d'incorporer cette modification dans le projet UP5K-demo, il faut insérer une ligne puis en modifier une autre dans le fichier *neorv32_iCEBreaker_BoardTop_UP5KDemo.vhd* se trouvant dans le dossier *~/neorv32/setups/board_tops*. Après la ligne 44, insérez la nouvelle ligne `user_reset_btn : in std_ulogic;`. Modifiez ensuite la ligne 130 : `RESETB => user_reset_btn,`. Attention, une erreur de syntaxe se produira si vous oubliez la virgule à la fin. Le uButton est maintenant configuré pour fonctionner comme un reset. Enfin, pour que les changements prennent effet, produisez un nouveau bitstream dans l'UP5K-Demo et chargez-le dans le FPGA iCEBreaker.

Et maintenant !

Il faudrait consacrer un livre entier à chacun de ces sujets : RISC-V, FPGA et NEORV32. Pour les débutants, l'iCE40UP5K est peu onéreux, mais la puce est bien souvent en rupture d'approvisionnement. Les deux cartes référencées dans l'article ne sont pas les seules à accueillir ce FPGA. Il en existe plusieurs autres : cela va du HAT pour Raspberry Pi à la console portable complète. Les différents outils et projets applicables à ce FPGA méritent également d'être étudiés. Citons par ex.

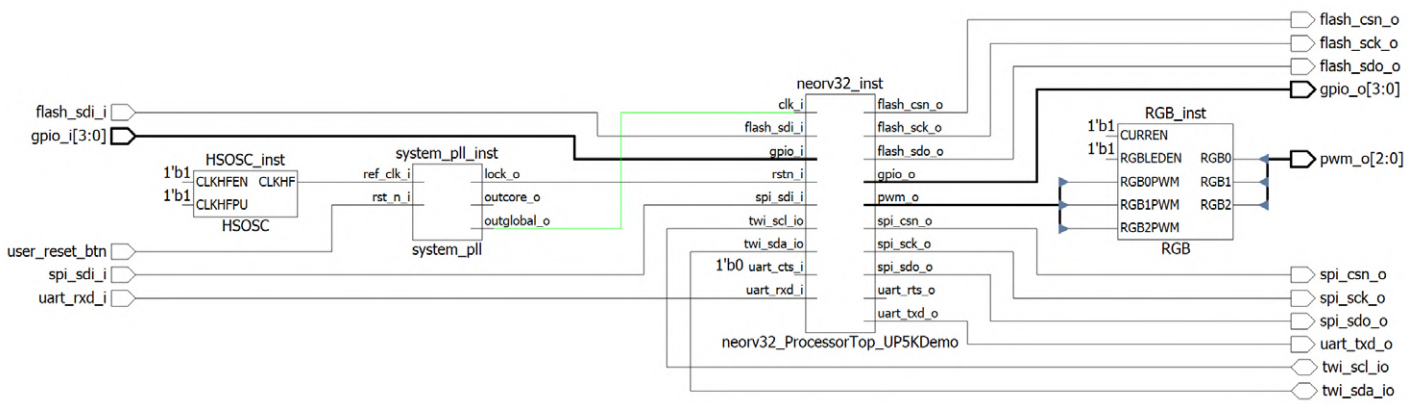


Figure 16. Raccordement du bouton de réinitialisation uButton à l'entrée rst de la PLL.

LiteX [16] qui permet d'assembler votre propre SoC comme dans un kit et n'est pas nécessairement limité à RISC-V, RISCboy [17] de Luke Wren qui fournit un système de type Gameboy dans le FPGA, et la console de jeu OK-ice40-PRO [18].
Si les difficultés d'approvisionnement disparaissent, d'autres projets et idées pour l'iCE40UP5K naîtront à coup sûr. Une petite console de jeux conçue à partir d'un iCE40UP5K et d'un Raspberry Pi RP2040 semble déjà être en préparation [19].

210175-04

Contributeurs

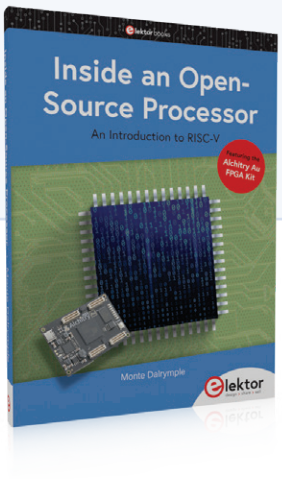
Texte et images : **Mathias Claußen**
Rédaction : **Jens Nickel**
Traduction : **Yves Georges**
Mise en page : **Harmen Heida**

Des questions, des commentaires ?

Envoyez un courriel à l'auteur (mathias.claussen@elektor.com) ou contactez Elektor (redaction@elektor.fr).

PRODUITS

- Carte de développement FPGA Alchitry Cu (Lattice iCE40 HX)
  www.elektor.fr/19640
- Convertisseur USB-TTL CH340, module UART CH340G (3,3 V/5,5 V)
  www.elektor.fr/19151
- Livre en anglais, « Inside an Open-Source Processor », M. Dalrymple (Elektor 2021)
  www.elektor.fr/19826



LIENS

[1] M. Ossmann, « projet SCCC (1) », Elektor 3-4/2019 : www.elektormagazine.fr/180394-04
[2] S. Cording, « RISC-V : quesaco ? », Elektor 7-8/2021 : www.elektormagazine.fr/210223-04
[3] Spécification du RISC-V : http://riscv.org/technical/specifications/
[4] Dépôt GitHub de NEORV32 : http://github.com/stnolting/neorv32/
[5] Page du produit Lattice iCE40UltraPlus : www.latticesemi.com/en/Products/FPGAandCPLD/iCE40UltraPlus
[6] Dépôt GitHub d'iCEBreaker : http://github.com/icebreaker-fpga/icebreaker
[7] Dépôt GitHub de l'UPduino : http://github.com/tinyvision-ai-inc/UPduino-v3.0
[8] Lattice Radiant : www.latticesemi.com/LatticeRadiant
[9] YosysHQ oss-cad-suite-build : http://github.com/YosysHQ/oss-cad-suite-build
[10] Configurer une chaîne d'outils pour le Kendryte K210, Elektor Labs : www.elektormagazine.com/labs/setup-a-toolchain-for-the-kendryte-k210-1
[11] Snickerdoodles au goût de Linux avec Zynq, ElektorTV : www.youtube.com/watch?v=EE4yYZ-FEoQ
[12] YosysHQ oss-cad-suite-build Releases : http://github.com/YosysHQ/oss-cad-suite-build/releases
[13] HTerm : www.der-hammer.info/
[14] NEORV32 - Construire la chaîne d'outils à partir de zéro : http://stnolting.github.io/neorv32/ug/#_building_the_toolchain_from_scratch
[15] W. Gay, « multitâche en pratique avec l'ESP32 », Elektor 1-2/2020 : www.elektormagazine.fr/190182-03
[16] LiteX : http://github.com/enjoy-digital/litex
[17] RISCBoy : http://github.com/Wren6991/RISCBoy
[18] OK-iCE40Pro Handheld : http://github.com/WiFiBoy/OK-iCE40Pro
[19] PicoStation3D : http://github.com/Wren6991/PicoStation3D
[20] Nolting S., The NEORV32 RISC-V-Processor, dépôt GitHub 2020 : http://raw.githubusercontent.com/stnolting/neorv32/master/docs/figures/neorv32_processor.png

tracer des graphiques avec Arduino

C'est facile avec le traceur sériel d'Arduino

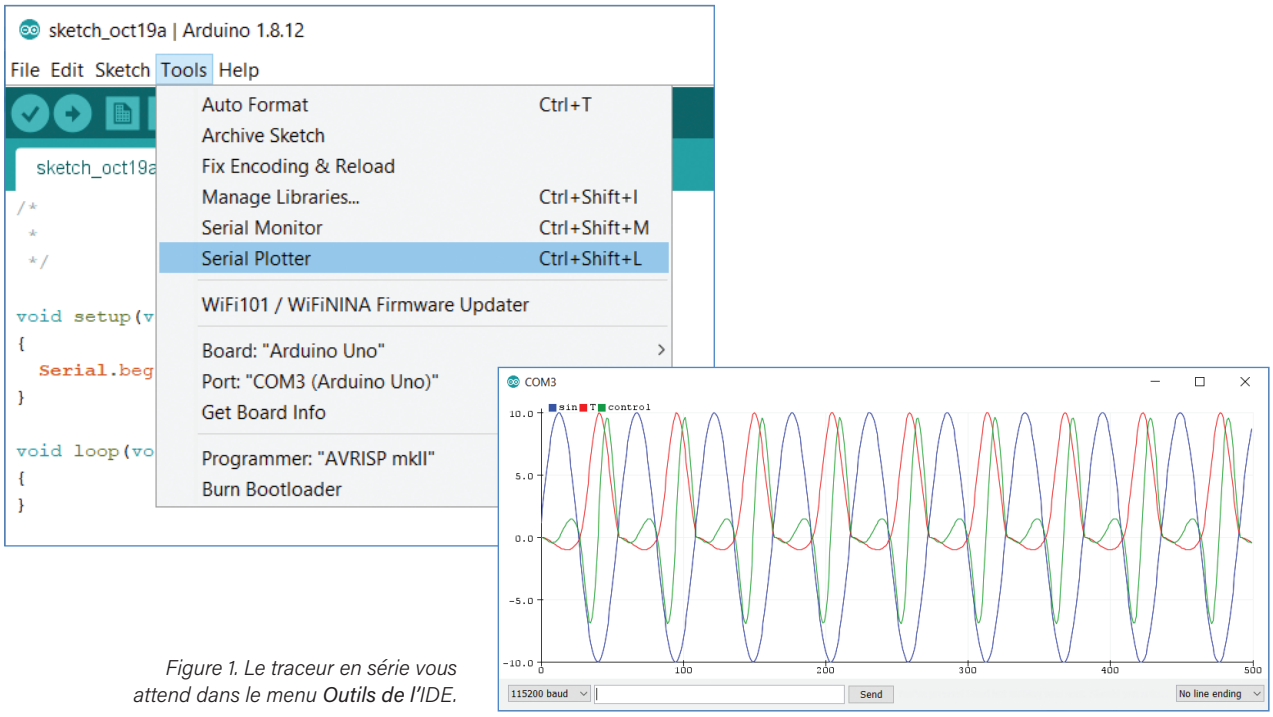


Figure 1. Le traceur en série vous attend dans le menu **Outils** *de l'IDE.*

Clemens Valens (Elektor)

L'un des points forts d'Arduino est son port série bien pratique pour exporter des données, recevoir des commandes ou pour déboguer. Il existe un moniteur série et un traceur série pour visualiser ces données. Le premier marche tout seul, et voici comment utiliser le Serial Plotter.

La plupart des utilisateurs d'Arduino connaissent le *Serial Monitor* intégré à l'IDE (dans le menu *Outils,* ou avec *Ctrl+Maj+M*). Ce modeste terminal série affiche toutes les données reçues sur le port série. Il permet d'envoyer des chaînes de caractères (p.ex. des commandes ou des données) de l'ordinateur à la carte Arduino. Pendant le débogage, utilisez le *Serial Monitor* pour faire envoyer par le sketch des messages texte sur ce qu'il fait ou ce qu'il est censé faire, c'est pratique.

Peloteur en série

Le *Serial Monitor* peut aussi afficher des valeurs numériques, bien sûr. Vous avez déjà essayé de suivre l'évolution d'un paramètre sous forme d'une liste de valeurs imprimées ? C'est plutôt soporifique…

L'EDI Arduino comporte un *Serial Plotter* (dans le menu *Outils,* ou avec *Ctrl+Maj+L*, **fig. 1**).
À partir des données numériques que reçoit ce traceur en série, il crée des graphiques et les affiche. Ce n'est pas la cybernétique avec plein d'options, mais c'est facile à utiliser. Seulement il n'y a pas de doc à son sujet. Seul le code source Java [1] peut vous donner une idée de ses capacités. Essayez et vous découvrirez les fonctions suivantes :

- nombre illimité de graphiques ;
- mise à l'échelle automatique de l'axe vertical ou Y ;
- 500 points sur l'axe horizontal ou X ;
- afficher les 500 points de données les plus récents.

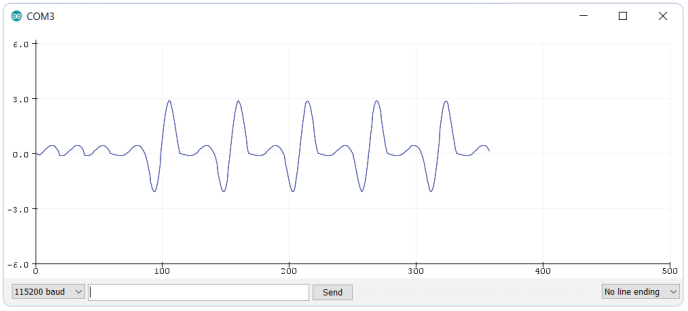


Figure 2. L'axe horizontal peut accueillir jusqu'à 500 points. Si vous en envoyez plus, il se mettra à défiler.

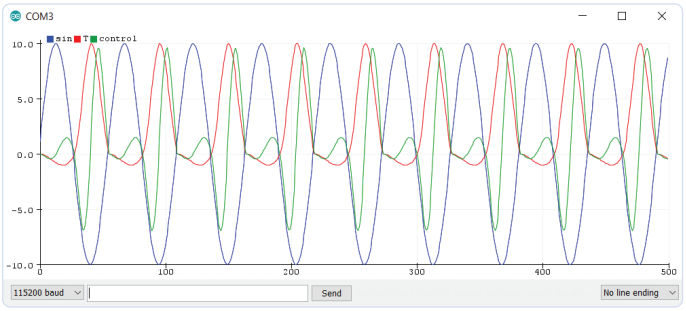


Figure 3. En théorie, le nombre de graphes est illimité. Utilisez des libellés pour savoir quel graphique représente quelles données.

Peloter un graphique

Après avoir ouvert le traceur en série, choisissez la même vitesse de transmission que celle qui est spécifiée dans Serial.begin dans le *sketch*. La case voisine de la boîte de sélection de la vitesse est destinée à renvoyer les données vers le *sketch*, exactement comme dans le *Serial Monitor*. Cela vous permet de contrôler le croquis dans une certaine mesure.

Vous n'aurez besoin d'apprendre ni le grec ni aucune commande spéciale pour utiliser le traceur série, ce sont les mêmes commandes de port série que le moniteur série. Peloter en série, c'est juste une autre façon de visualiser les données envoyées par le port série. C'est seulement le formatage des données qui importe. Pour tracer le graphique d'un seul paramètre (**fig. 2**), une valeur de capteur p.ex., il suffit de séparer les valeurs successives par un caractère de saut de ligne. Autrement dit, pour envoyer les valeurs, utilisez

```
Serial.println(sensor_value);
```

Les échelles s'adaptent à la plage

Le traceur utilisera ces valeurs pour l'axe vertical, l'axe Y ; il incrémente lui-même l'axe horizontal (X). Autrement dit, la première valeur envoyée sera considérée comme la valeur «*y*» pour $x = 0$, la deuxième valeur est la valeur «*y*» pour $x = 1$, et ainsi de suite. Lorsque *x* atteint 500, le graphique commence à défiler horizontalement ; à mesure que les points les plus anciens sont supprimés au début, de nouveaux points sont ajoutés à la fin .

Lorsque la *valeur y* sort des limites, l'échelle de l'axe vertical s'adapte automatiquement pour rester dans les limites verticales. Inversement, si toutes les valeurs tiennent entre dans des limites resserrées (lorsque le graphique défile et qu'une valeur élevée a été supprimée au début), l'échelle s'adapte à la nouvelle plage de valeurs.

Peloter plusieurs graphiques

Pour tracer deux paramètres ou plus, envoyez les différentes valeurs sous forme d'une liste de valeurs séparées par une virgule, un espace ou un caractère de tabulation (\t) et fermez la liste avec un caractère de retour à la ligne :

```
Serial.print(temperature);
Serial.print(',');
Serial.print(humidity);
Serial.print(',');
Serial.print(pressure);
Serial.println();
```

Le format qui sépare les données par une virgule est dit CSV, pour *comma-separated values* (valeurs séparées par des virgules). La plupart des tableurs reconnaissent ce format, certains vous permettent de choisir vous-même le caractère séparateur. Une fois que le résultat vous plaît dans le *Serial Plotter*, exportez les valeurs avec le *Serial Monitor* pour les copier-coller dans un tableur et obtenir de jolis graphiques et faire d'autres traitements.

La liste des valeurs des données est lue comme des valeurs y différentes pour une valeur x invariable, de sorte que tous les graphiques sont actualisés au même moment et à la même vitesse. Le traceur affectera une couleur différente à chaque graphique sans demander l'avis de l'utilisateur. La seule façon d'influer un peu sur ce choix est de jouer sur l'ordre des données.

Libellés

Ce peloteur ne manque pas de fantaisie, puisqu'il permet de spécifier un libellé pour chaque graphique (**fig. 3**). Il y a deux façons de le faire :

1. Avant d'envoyer des valeurs de données, envoyez une liste de libellés séparés par une virgule, un espace ou une tabulation et terminez avec un caractère de retour à la ligne. Ces caractères réservés ne peuvent donc pas figurer dans les libellés. Pour qu'elles correspondent, les données doivent être envoyées dans le même ordre que les libellés. Exemple :

   ```
   Serial.println(«temperature,humidity,pressure»)
   ```

2. Envoyez étiquettes et valeurs par paires et séparez-les par deux points (:), par exemple :

   ```
   Serial.println(«temperature:21,humidity:76,
   pressure:1007»)
   ```

   Envoyez ces paires toujours dans le même ordre pour que le traceur ne mélange pas les données des graphiques. Une liste de paires de valeurs de libellés doit être séparée à son tour soit par une virgule, un espace ou une tabulation et se terminer par un caractère de retour à la ligne.

Toute chaîne de texte sera considérée comme un libellé, et vous ne manquerez pas d'occasions de vous fourvoyer, par exemple en oubliant les délimiteurs, ce qui fait que les données se changeront en libellé.

Caprices et limites

- L'axe horizontal se prolonge à l'infini. Pas moyen de boucler le tracé par programmation. À moins de fermer le traceur et de le rouvrir. Parfois, il faut s'y reprendre à deux fois pour se débarrasser des libellés et/ou des données périmés.

- Vous ne pourrez pas tracer un point à une position (*x*,*y*) de votre choix sans que de nouveaux points soient ajoutés derrière.

- Que l'axe vertical se mette à l'échelle automatiquement, c'est bien, mais ça devient très ennuyeux lorsque des aberrations ou des pépins forcent l'échelle à augmenter et rendent difficile, voire impossible, la visualisation des données intéressantes beaucoup plus petites. Cela peut également se produire lorsqu'une série de données se situe dans la plage de [–1,+1] par exemple tandis qu'une autre est dans la plage [–100,+100]. Dans ce cas, l'échelle sera de –100 à +100 pour les deux séries.

  Des petits malins essaient de ruser en ajoutant quelque chose comme `min:-100,max:100` à la liste des valeurs de données pour tenter d'arrêter la mise à l'échelle automatique, mais cela ne fonctionne que si toutes les valeurs restent dans ces limites (**fig. 4**).

- Seul le papa du peloteur en série sait pourquoi l'échelle minimale est généralement de –6 à +6 mais parfois de –5 à +5.

Fonction sparadrap

Voici un petit sparadrap à insérer à vos sketches pour améliorer un peu la convivialité du traceur en série tout en gardant vos croquis lisibles :

```
void plot(String label, float value, bool last)
{
  Serial.print(label); // May be an empty string.
  if (label!=»») Serial.print(«:»);
  Serial.print(value);
  if (last==false) Serial.print(«,»);
  else Serial.println();
}
```

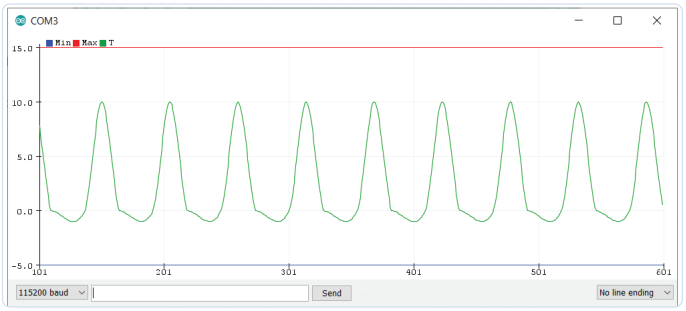


Figure 4. L'envoi de valeurs minimales et maximales constantes évite la mise à l'échelle automatique de l'axe vertical, mais seulement tant que toutes les autres valeurs restent dans ces limites.

Si le libellé de la chaîne est spécifié, cette fonction imprimera une paire `label:value`. Si *label* est une chaîne vide, alors elle n'imprimera que la valeur. Cela lui permet de sortir des données au format CSV pur dans le Serial Monitor, utilisable avec des tableurs.

Vous noterez que vous ne pouvez pas avoir le *Serial Monitor* et le *Serial Plotter* ouverts en même temps.

Après avoir ouvert le port série, appelez-le ainsi :

```
// Graph1, more graphs follow.
plot(«Temperature»,temperature_value,false);
// Graph2, more graphs follow.
plot(«Humidity»,humidity_value,false);
// Graph3, last graph.
plot(«Pressure»,pressure_value,true);
```

N'oubliez pas de fermer et rouvrir le traceur en série chaque fois que vous modifiez le croquis pour vous assurer que données et libellés périmés sont tous supprimés. En cas de doute, recommencez.

200540-03

Votre avis, s'il vous plaît ?
Questions et commentaires sur cet article bienvenus par courriel à l'auteur clemens.valens@elektor.com ou à la redaction@elektor.fr

Ont contribué
Auteur : **Clemens Valens**
Rédaction : **C.J. Abate**
Maquette : **Giel Dols**
Traduction : **Rémy Bécarre**

LIEN

[1] **Code source du traceur série de l'IDE Arduino :**
https://github.com/arduino/Arduino/blob/master/app/src/processing/app/SerialPlotter.java

carte CLUE d'Adafruit

Une solution intelligente pour les projets IoT

Tam Hanna (Slovakie)

La carte BBC micro:bit a connu un grand succès, bien au-delà du marché éducatif auquel elle était destinée. La carte CLUE d'Adafruit vient d'apparaître sur le marché, avec un écran à part entière et plus de mémoire. Dotée du Bluetooth LE et d'une multitude de capteurs intégrés, elle convient particulièrement bien aux petits projets IoT.

Après le succès des cartes Arduino et Raspberry Pi, il est devenu évident qu'il y a de l'argent à faire avec les différents types d'ordinateurs éducatifs. En 2016, la BBC est entrée dans la course avec micro:bit, un ordinateur monocarte doté d'un SoC Bluetooth de Nordic Semiconductor au lieu d'un processeur à part entière sous Linux.

Depuis lors, des entreprises ont vécu d'incroyables histoires, telles que *Spy oz* en Slovaquie qui a lancé des sociétés qui ne s'occupent que de la distribution de l'écosystème micro:bit [1].

L'adage selon lequel « à peine est-on servi qu'on regarde ce qu'il y a dans l'assiette du voisin » s'applique également au domaine de l'informatique embarquée. Les progrès continus dans le domaine des SoC Bluetooth ont laissé les 16 MHz et les 16 Ko de SRAM de micro:bit paraitre un peu dépassés. De plus, son afficheur à 5 × 5 LED

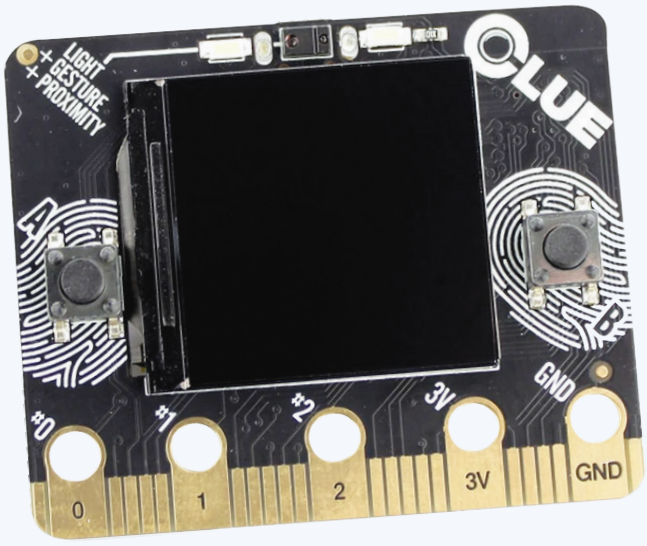


Figure 1. L'attaquant vu de face…

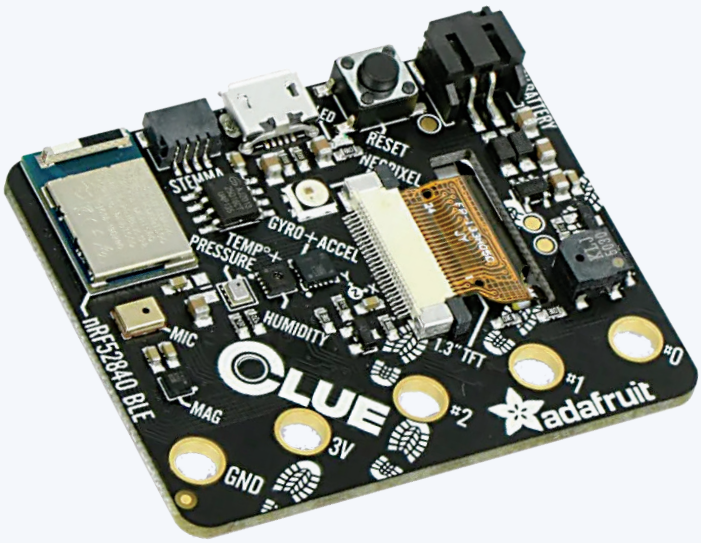


Figure 2. … et par l'arrière.

CARACTÉRISTIQUES

- Processeur Nordic nRF52840 Bluetooth LE processor : 1 Mo de flash, 256 Ko de RAM, cœur à 64 MHz Cortex M4
- Écran TFT IPS de 1,3», 240×240, couleur pour texte et graphiques
- Alimentation : toute source de 3,6 V (régulateur interne et diodes de protection)
- Deux boutons pour l'utilisateur et un bouton de RàZ
- Capteur de mouvement, accéléromètre/gyroscope et magnétomètre
- Capteurs de proximité, lumière, couleur et mouvement
- Capteur de son PDM, microphone
- Capteur d'humidité SHT
- Capteur BMP280 pour température et pression barométrique/altitude
- LED RVB NeoPixel de signalisation
- Mémoire flash interne de 2 Mo pour enregistrement de données, images, fonts ou code CircuitPython
- Buzzer/haut-parleur pour l'émission de sons et de bips
- Deux LED blanches brillantes à l'avant pour l'éclairage/la détection des couleurs
- Connecteur Qwiic/STEMMA QT pour ajouter des capteurs, pilotes de moteurs ou des écrans par I²C. Possibilité de connecter des capteurs GROVE I²C avec un câble d'adaptation.
- Programmable avec l'EDI Arduino ou le langage CircuitPython.

ne convient pas pour représenter autre chose que les plus simples des graphiques.

Adafruit attaque avec CLUE

Avec le lancement du nRF52840 de Nordic Semiconductor, un SoC Bluetooth monocœur dont le processeur ARM atteint 64 MHz et est doté de 256 Ko de RAM, il y avait de quoi proposer un nouveau plat.

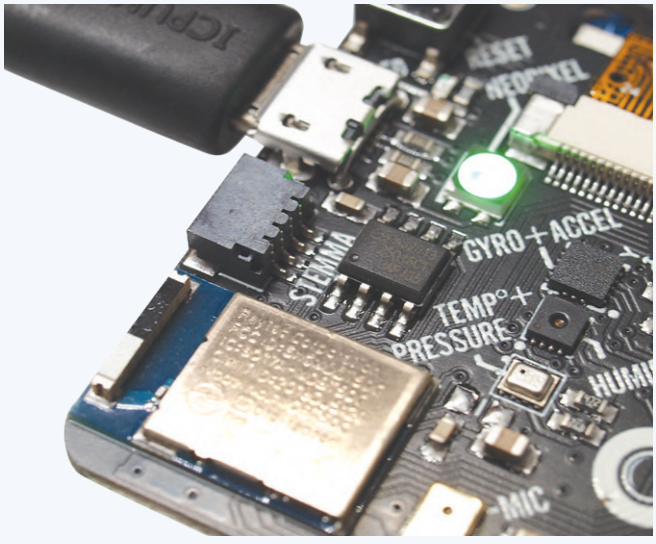


Figure 3. Le port STEMMA permet à la carte CLUE d'Adafruit d'être connectée à des cartes d'extension.

Les **figures 1** et **2** montrent le résultat : la carte CLUE d'Adafruit qui semble très similaire à la carte micro:bit de BBC. À part le SoC, qui n'est pas immédiatement visible sur ces photos, c'est l'écran beaucoup plus grand situé à l'avant qui se distingue vraiment. Les LED sont remplacées par un écran couleur de 240×240 pixels, basé sur la technologie classique IPS LCD plutôt que sur la technologie organique.
Un autre détail agréable de la carte est le connecteur placé à l'arrière, cf. **figure 3**. Elle dispose d'un bus I²C au format interne d'Adafruit sur lequel d'autres capteurs peuvent être facilement connectés. Il existe également un adaptateur pour le format Grove de Seeed, qui propose divers capteurs à prix raisonnable.
Notez que la carte CLUE ne semble être que partiellement compatible avec la micro:bit. Bien que le connecteur le long du bord inférieur soit d'une construction identique, l'utilisation d'un afficheur différent signifie qu'à première vue, la plupart des boîtiers disponibles pour le BBC micro:bit ne conviendront pas pour la carte CLUE d'Adafruit.
L'auteur a testé cette hypothèse avec un boîtier ThingiVerse de domw disponible sur [2]. La partie avant ne convenait pas, car l'écran de la CLUE était beaucoup plus grand que la matrice de LED de l'original de BBC. Compte tenu de cela, l'auteur a été particulièrement surpris par le fait que la face arrière du boîtier s'est correctement montée, malgré les connecteurs supplémentaires. Cependant, lorsqu'on l'examine de plus près, cela est probablement dû au fait que la conception du boîtier était relativement généreuse. Si le boîtier avait été conçu avec un ajustement plus serré, il aurait été probablement inutilisable.

Une question de programmation

Comme la micro:bit est un système éducatif, le développement ne ressemble pas à l'expérience que l'on peut avoir en utilisant les environnements de développement embarqués classiques tels que

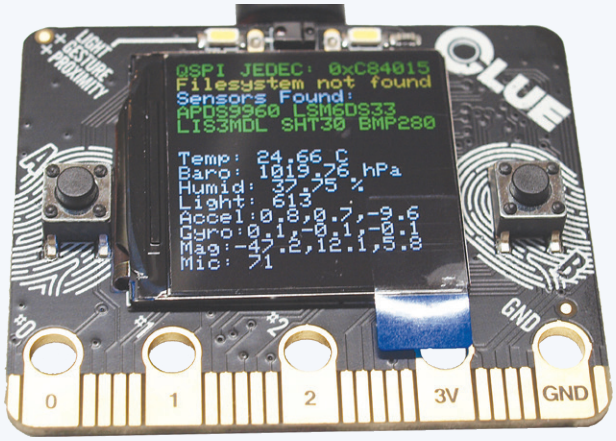


Figure 4. Cette appli pré-installée montre les informations renvoyées par les capteurs.

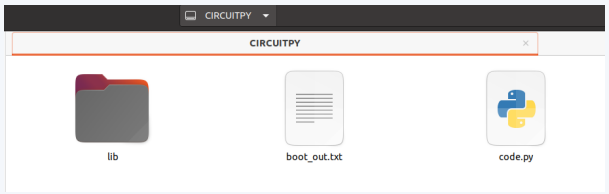


Figure 5. Le « runtime » Python dévoile également un lecteur virtuel.

ARM Keil. Cela peut être gênant pour les puristes de l'embarqué, mais en pratique il est indispensable, car de nombreuses universités n'ont pas assez de personnel compétent pour déboguer les programmes C++ (croyez l'auteur : les étudiants créent des erreurs de programmation vraiment complexes).
Au lieu de cela, on s'appuie généralement sur un quatuor composé de l'EDI Arduino, CircuitPython, MakeCode et Scratch. Cependant, pour la CLUE, seulement deux de ces environnements sont actuellement disponibles. MakeCode est en cours d'adaptation, sans date de disponibilité précise, et il n'y a aucune information pour Scratch. Un chargeur d'amorçage série pour déployer le code est inclus, un peu comme pour la carte Raspberry Pi Pico.
Pour une première petite expérience, faisons fonctionner CircuitPython. Si vous connectez une carte neuve à un ordinateur, via le connecteur micro-USB à l'arrière, l'écran affiche une page d'état (**fig. 4**) qui fournit des informations sur l'état fonctionnel.
En appuyant deux fois sur le bouton *reset* situé au dos de la carte, le tampon de trame du pilote de l'écran se fige. Le poste de travail connecté (l'auteur utilise Linux) voit alors une nouvelle clé USB où le code compilé peut être chargé.
Fait intéressant, la carte CLUE d'Adafruit est toujours visible par l'ordinateur. Si elle n'est pas en mode *bootloader*, *dmesg* la détecte comme suit :

```
tamhan@TAMHAN18:~$ dmesg
. . .
[28292.202193] usb 1-2.7: Manufacturer: Adafruit LLC
[28292.202195] usb 1-2.7: SerialNumber: 7687A137B6FDB874
[28292.204040] cdc_acm 1-2.7:1.0: ttyACM0: USB ACM
   device
```

Après une double pression sur le bouton *reset*, une clé USB apparaît, comme ici :

```
tamhan@TAMHAN18:~$ dmesg
. . . .
[28371.624193] sd 10:0:0:0: Attached scsi generic sg6
   type 0
```

Important : ce lecteur ne reste pas activé pour toujours. S'il demeure inutilisé pendant plus de 30 s environ, le micrologiciel se remet à tourner normalement.

Recherche de fichier

En visitant l'URL *https://circuitpython.org/board/clue_nrf52840_express/*, nous pouvons faire notre première étape et télécharger le fichier *adafruit-circuitpython-clue_nrf52840_express-en_US-6.1.0.uf2*. Il contient le *runtime* qui doit être placé sur la clé USB.
Notez que vous trouverez également un fichier nommé *CURRENT.UF2* sur le lecteur. Celui-ci vous permet de télécharger le micrologiciel qui se trouve actuellement dans la mémoire du système cible.
Étrangement, le *runtime* n'est pas fourni pas avec une bibliothèque complète spécifique à tous les capteurs disponibles. Nous devons plutôt aller dans l'URL *https://circuitpython.org/libraries* pour télécharger l'archive *adafruit-circuitpython-bundle-6.x-mpy-20210329.zip* puis l'extraire dans un dossier approprié du système de fichiers.
À ce stade, vous devriez jeter un autre coup d'œil à l'écran de la carte CLUE, car le *runtime* délivre en sortie et en permanence le contenu de la console. Un détail agréable est que le dispositif sur le PC (cf. **fig. 5**) présente la mémoire interne de l'environnement de travail Python.

Il est important de déplacer les dossiers suivants de l'archive vers le dossier *Libs* du dispositif :

adafruit_apds9960
adafruit_bus_device
adafruit_display_shapes
adafruit_display_text
adafruit_lsm6ds
adafruit_register

Comme si cela ne suffisait pas, Adafruit s'attend également à ce que vous rassembliez les fichiers individuels suivants. Je ne comprends pas pourquoi ils ne sont pas tous regroupés dans une seule archive :

adafruit_bmp280.mpy
adafruit_clue.mpy
adafruit_lis3mdl.mpy
adafruit_sht31d.mpy
adafruit_slideshow.mpy
neopixel.mpy

Exemple de code

Pour un premier essai simple avec l'environnement Python, vous pouvez utiliser l'exemple fourni sur *https://learn.adafruit.com/adafruit-clue/clue-spirit-level*. Il s'agit d'une application de niveau à bulle utilisant plusieurs idiomes spécifiques à CLUE.
La première étape du code consiste à inclure un ensemble de bibliothèques :

```
import board
import displayio
from adafruit_display_shapes.circle import Circle
from adafruit_clue import clue
```

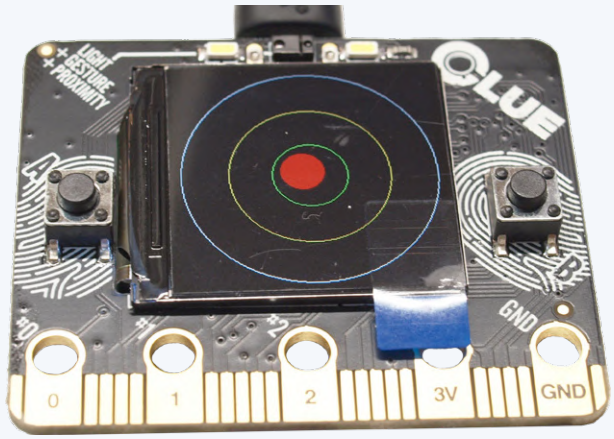


Figure 6. Un niveau à bulle utilisant les capteurs intégrés.

En plus de l'objet `clue`, qui assure diverses fonctions liées à la carte, importer la classe `Circle` est également intéressant ici. La pile GUI permet à la fois de dessiner directement dans un *frame buffer* et de travailler avec des objets, convertis par le micrologiciel, en éléments visibles à l'écran.
Dans la section suivante, le micrologiciel initialise une référence à l'écran et assemble un objet de type groupe pour l'écran :

```
display = board.DISPLAY
clue_group = displayio.Group(max_size=4)
```

L'objet `clue_group` est intéressant parce qu'il crée un élément parent qui rappelle un arbre DOM. Notre code crée donc plus ou moins des objets arbitraires dans cet arbre à afficher.
Regardez la photo de la **figure 6**, la prochaine action du programme est de créer les trois cercles qui représentent la déviation et à les enregistrer pour la sortie :

```
outer_circle = Circle(120, 120, 119, outline=clue.WHITE)
middle_circle = Circle(120, 120, 75, outline=clue.
   YELLOW)
inner_circle = Circle(120, 120, 35, outline=clue.GREEN)
clue_group.append(outer_circle)
clue_group.append(middle_circle)
clue_group.append(inner_circle)
```

On trouve ensuite quelques tâches de « ménage », dont le sens est plus facile à comprendre en examinant l'exemple de code ci-dessous :

```
x, y, _ = clue.acceleration
bubble_group = displayio.Group(max_size=1)
level_bubble = Circle(int(x + 120), int(y + 120), 20,
   fill=clue.RED, outline=clue.RED)
bubble_group.append(level_bubble)

clue_group.append(bubble_group)
display.show(clue_group)
```

Enfin et surtout, nous avons besoin d'une boucle qui analyse les valeurs de position fournies par la bibliothèque Adafruit via l'attribut `acceleration` et qui les écrit dans les propriétés de l'objet `bubble_group` :

```
while True:
    x, y, _ = clue.acceleration
    bubble_group.x = int(x * -10)
    bubble_group.y = int(y * -10)
```

La façon la plus pratique d'exécuter rapidement du code sur la carte CLUE est d'utiliser le fichier *code.py* de la figure 5. Le micrologiciel CircuitPython l'exécute automatiquement à chaque démarrage. La figure 6 montre ce que vous pouvez obtenir.
Comme la carte est dotée du Bluetooth, ce peut être un canal pour communiquer avec l'ordinateur hôte. Sur [3], Adafruit fournit un exemple amusant qui illustre l'utilisation de l'API web Bluetooth implémentée dans Google Chrome.

Et maintenant en C

Le langage Python permet d'obtenir rapidement des résultats « non bureaucratiques » avec un système embarqué. Cependant, la puissance maximale est atteinte avec le langage C. Sur un système monocœur avec liaison radio, la mise en œuvre de la communication est un défi si l'on veut que l'application ne souffre pas de problèmes de synchronisation. C'est pourquoi Adafruit oblige plus ou moins les développeurs à utiliser l'EDI Arduino. Un système d'exploitation en temps réel fonctionne alors en arrière-plan, allouant la puissance de calcul aux différentes tâches.
Sous Linux, la première étape nécessite un paquet d'extension qui permet à l'EDI Arduino (version 1.8.6 ou supérieure) de communiquer avec le *bootloader* non standard de CLUE :

```
tamhan@TAMHAN18:~$ pip3 install --user adafruit-nrfutil
Collecting adafruit-nrfutil
  . . .
Successfully installed adafruit-nrfutil-0.5.3.post13
```

Ensuite, il faut saisir l'URL *www.adafruit.com/package_adafruit_index.json* dans le *Gestionnaire de carte* pour rendre le package de la carte *Adafruit nRF52* disponible pour le téléchargement. Une fois ces étapes terminées, la carte sera disponible sous *Outils* > *Type de carte* > *Adafruit CLUE*.
Malheureusement, comme dans le cas de CircuitPython, l'installation de ces bibliothèques et des autres paramètres est un processus laborieux. Vous trouverez plus d'informations à ce sujet dans [4].

LIENS

[1] StreamIT, s. r. o. : https://edutronik.sk/v2/
[2] Boîtier : www.thingiverse.com/thing:1767446
[3] Appli de démonstration : https://learn.adafruit.com/bluefruit-dashboard-web-bluetooth-chrome
[4] Informations : https://learn.adafruit.com/adafruit-clue?view=all

Carte CLUE : vaut-elle le coup ?

Avec la carte CLUE, vous disposez d'une plateforme d'évaluation très attrayante, agréable à utiliser dans le domaine de l'interfaçage et en plus dotée d'un écran couleur. Toutefois son prix est relativement élevé par rapport à la carte BBC micro:bit (nettement moins chère). Notez également que la carte CLUE n'est pas compatible à 100% avec micro:bit. L'expérience nous apprend que ce détail « insignifiant » a un fort impact, surtout lorsque l'on se trouve dans une situation difficile, comme la migration d'un projet existant et fonctionnel entre les deux plateformes.

Pour ceux qui veulent travailler avec un microcontrôleur « pur », basé sur la technologie Nordic ou tout autre module radio, vous serez probablement mieux servis par une carte d'évaluation classique. En résumé, la carte CLUE est un produit attachant pour ceux qui apprécient la carte BBC micro:bit, mais qui ont besoin d'un peu plus de puissance ou d'un véritable écran.

210395-04

Des questions, des commentaires ?

Contactez Elektor (redaction@elektor.fr).

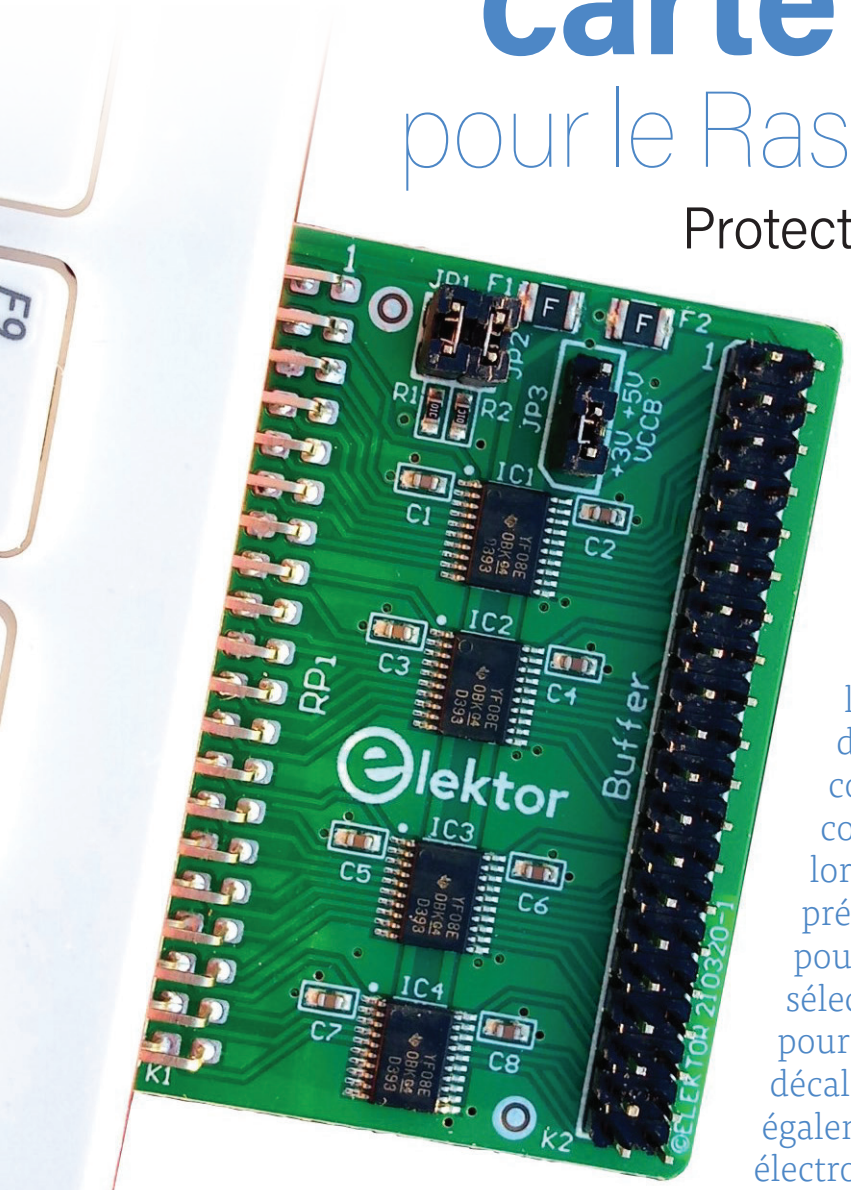
Contributeurs

Texte et illustrations : **Tam Hanna**
Rédaction : **Jens Nickel**
Traduction : **Asma Adhimi**
Mise en page : **Giel Dols**

PRODUITS

- Carte CLUE d'Adafruit
  www.elektor.fr/19512
- Livre « Initiation au langage CircuitPython et à la puce nRF52840 », Michaël Bottin, Elektor, ISBN 978-2-86661-211-5
  www.elektor.fr/19523

carte tampon
pour le Raspberry Pi 400
Protection des entrées/sorties



Ton Giesberts (Elektor)

Sorti en novembre 2020, le Raspberry Pi 400 est en fait un Raspberry Pi 4, mais logé dans un clavier, avec le connecteur d'extension GPIO à l'arrière du boîtier. La connexion de matériel supplémentaire comporte toujours des risques, notamment lors du prototypage. La carte tampon présentée ici est spécialement conçue pour le Raspberry Pi 400. Elle permet la sélection des niveaux hauts 3,3 V et 5 V pour les 26 broches GPIO, et les tampons/décaleurs de niveau utilisés à cet effet offrent également une protection contre les décharges électrostatiques.

Est-il besoin de présenter ici les cartes Raspberry Pi ? Depuis la sortie de la première version en 2012, elles ont fait l'objet (ou fait partie) de nombreux projets dans le magazine Elektor. Nous avons présenté un certain nombre de montages différents et de circuits imprimés d'extension pour ces cartes à processeurs. Pour ce projet, disons-le tout net : le circuit dont je vais parler n'est pas nouveau, puisque sa première version a été abordée en 2015 dans la page web du labo d'Elektor [1]. En 2018, la conception a été adaptée au connecteur d'extension à 40 broches d'E/S qui a était la norme pour connecter du matériel supplémentaire aux nano-ordinateurs depuis l'introduction du Raspberry Pi B+ [2]. Et maintenant, cette dernière version est adaptée à l'un des produits les plus récents de la famille Raspberry Pi : le Raspberry Pi 400.

Pour faire simple, le 400 est un Raspberry Pi 4 logé dans un clavier. Il ressemble aux ordinateurs, autrefois célèbres dans les années 1980, tels que le Commodore 64, le Sinclair Spectrum et l'Acorn BBC.

En termes de spécifications, le Raspberry Pi est bien évidemment beaucoup plus puissant que ses prédécesseurs désormais dépassés. Mais il y a une similitude frappante : le connecteur d'extension GPIO à l'arrière du boîtier, auquel l'utilisateur peut connecter du matériel externe (conçu par lui-même). La connexion de matériel supplémentaire comporte toujours des risques, en particulier lors du prototypage, et la carte tampon présentée ici évite d'endommager le Raspberry Pi au cours du processus. En outre, la carte tampon permet l'interfaçage avec les niveaux 3,3 V et 5 V, et les tampons/décaleurs de niveau utilisés à cet effet offrent également une protection contre les décharges électrostatiques.

Matériel

Le schéma électrique du projet (**fig. 1**) est exactement le même que celui présenté dans notre magazine en 2018. Les tampons/décaleurs de niveau à 8 bits TXS0108E utilisés dans ce projet sont

bidirectionnels. Et chaque broche de port A et de port B possède une résistance de rappel au niveau haut. La résistance de rappel au niveau bas d'une broche GPIO du Raspberry Pi est généralement de l'ordre de 40 à 60 kΩ. Cette valeur est trop élevée pour mettre l'entrée/sortie au niveau bas lorsque la carte tampon est connectée. Ainsi, il faut noter qu'avec cette configuration d'entrée, le niveau logique ne sera pas bas, le rappel ne fonctionnant pas comme prévu.

Les E/S ont des broches d'alimentation séparées pour le côté Raspberry Pi et le monde extérieur, respectivement VCCA et VCCB. Chaque broche d'E/S sur le port A du TXS0108E possède une résistance de rappel vers VCCA, connectée à l'alimentation +3,3 V du Raspberry Pi 400, et chaque broche d'E/S sur le port B possède une résistance de rappel vers VCCB. La tension VCCB – pour le niveau des E/S sur K2 – peut être fixée à +3,3 V ou +5 V par le cavalier du connecteur JP3. La valeur des résistances de rappel des tampons est de 40 kΩ pour les sorties à l'état bas et 4 kΩ pour les sorties à l'état haut. Ainsi, les sorties des tampons sont en fait des drains ouverts. Par exemple, si une LED est connectée entre la sortie et la masse, un diviseur de tension est créé lorsqu'une résistance série supplémentaire est utilisée. Une charge résistive sur la sortie fera chuter le niveau logique haut. C'est quelque chose dont il faut tenir compte !

Les connexions d'alimentation entre K1 et K2 sur la carte tampon possèdent deux fusibles réarmables à coefficient de température positif de 0,5 A (F1 et F2) pour protéger les alimentations +5 V et +3,3 V du Raspberry Pi 400.

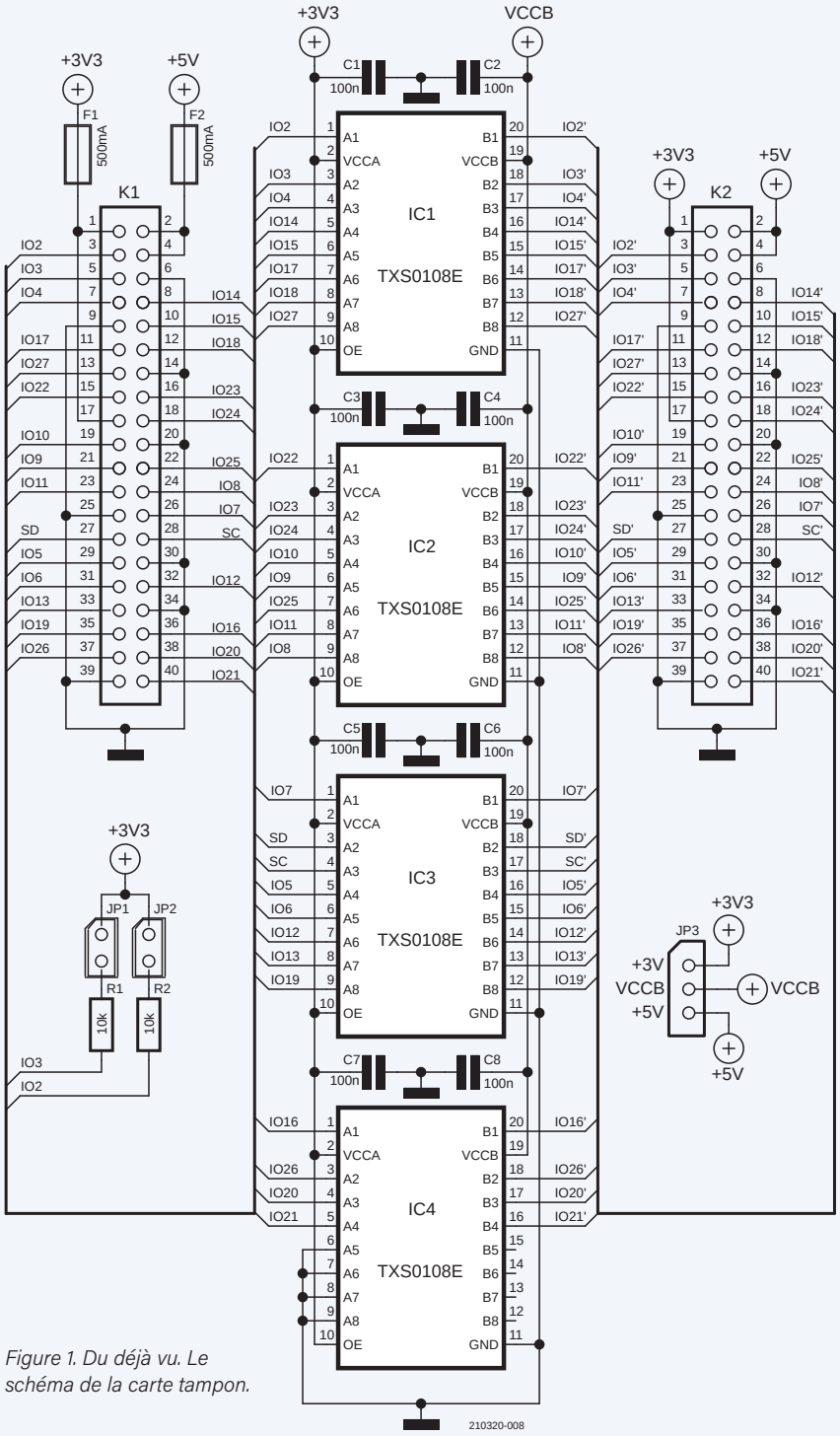


Figure 1. Du déjà vu. Le schéma de la carte tampon.

Pour l'implémentation d'un bus I²C servant à communiquer avec du matériel externe, GPIO2 correspond à la ligne de données série (SDA) et GPIO3 à la ligne d'horloge série (SCL). Les résistances de rappel supplémentaires R1 et R2 peuvent être activées avec les cavaliers JP1 et JP2.

Pendant le démarrage du Raspberry Pi, les bornes GPIO0 (ID_SD) et GPIO1 (ID_SC) sont utilisées pour lire l'EEPROM d'un HAT (*Hardware Attached on Top*, ou extension matérielle) I²C. Après le démarrage, ces broches GPIO peuvent être utilisées comme les 26 autres, mais il faut s'assurer de l'absence d'incidence sur le système lorsqu'on a monté un HAT I²C. Pour empêcher la lecture de GPIO0 et GPIO1 pendant le démarrage, ajoutez l'entrée suivante dans */boot/config.txt* :

```
force_eeprom_read=0
```

Pour plus d'informations, consultez la documentation Raspberry Pi sur le fichier *config.txt* [3].

Le circuit imprimé

Le schéma n'a peut-être rien de nouveau, mais le circuit imprimé (**fig. 2**) est spécialement adapté au Raspberry Pi 400. Il est un

LISTE DES COMPOSANTS

Résistances

R1,R2 = 10 kΩ, 100 mW, 1%, CMS 0603

Condensateurs

C1 à C8 = 100 nF, 50 V, 10%, X7R, CMS 0603

Semi-conducteurs

IC1 à IC4 = TXS0108EPWR, CMS TSSOP-20

Divers

K1 = réceptacle 2 × 20, angle droit, au pas de 2,54 mm
K2 = connecteur mâle à 2 × 20 broches , vertical, au pas de 2,54 mm
JP1,JP2 = connecteur mâle à 2 broches, vertical, au pas de 2,54 mm
JP3 = connecteur mâle à 3 broches, vertical, au pas de 2,54 mm
JP1,JP2,JP3 = cavalier, au pas de 2,54 mm
F1,F2 = fusible réarmable CTP, CMS, polyfuse, 1210L050YR Littelfuse
Circuit imprimé 210320-1

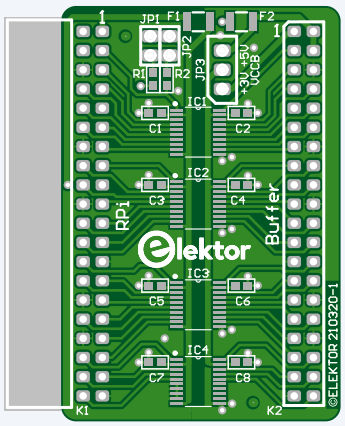


Figure 2. L'implantation du nouveau circuit imprimé de la carte tampon.

peu plus petit que la carte tampon originale présentée en 2018. Les fichiers Gerber de cette nouvelle carte sont disponibles en téléchargement, vous pouvez donc la commander chez le fabricant de circuits imprimés de votre choix. Mais il est, bien sûr, beaucoup plus pratique d'acheter la carte tampon entièrement assemblée dans l'e-choppe d'Elektor [4].

Un réceptacle à angle droit est utilisé pour le connecteur du côté du Raspberry Pi 400 de la carte tampon (K1) afin qu'il puisse être inséré dans le connecteur GPIO à l'arrière du clavier (**fig. 3**). Le connecteur pour les E/S tamponnées est un connecteur droit mâle standard à 40 broches (K2). La taille de la carte tampon est de 55×44 mm, y compris le connecteur K1 qui dépasse du bord du circuit imprimé. Par rapport au circuit imprimé original (réf. 150719-1), les deux rangées de broches de K1 sont interverties, car un réceptacle est utilisé ici. Placer un connecteur droit mâle standard à 40 broches pour K1 afin de connecter cette carte tampon à un Raspberry Pi 2, 3 ou 4 via un câble plat – comme avec l'ancienne version du carte tampon – ne fonctionnera pas ici. Pour autant, la **figure 4** montre que cette carte peut toujours être utilisée avec un Raspberry Pi 2, 3 ou 4.

Le connecteur de sortie K2 peut être connecté à des circuits externes à l'aide d'un câble en nappe, court, à 40 conducteurs, avec deux connecteurs 2×20 reliés, ou simplement un connecteur à réceptacle unique sur lequel sont soudés des fils courts, ou des supports simples pour fils. Cependant, faites attention en enfonçant un réceptacle à 40 conducteurs sur K2 ou en le débranchant de la carte. Ne le faites pas pendant que la carte tampon est encore insérée dans le Raspberry Pi 400, car une certaine force est nécessaire et le connecteur GPIO du Raspberry Pi 400 pourrait être endommagé.

Test de la carte tampon

Deux programmes Python très simples pour tester la carte tampon – empruntés à l'ancien projet – sont disponibles en téléchargement sur la page Elektor Labs de ce projet [5]. L'un sert à tester toutes les broches GPIO configurées en sortie, *Check_all_GPIOs_as_output.py*, et l'autre à tester toutes les broches GPIO configurées en entrée, *Check_all_GPIOs_as_input.py* (*210320-11.zip*). Sous Raspbian, il suffit de double-cliquer sur l'un des fichiers pour lancer l'EDI par défaut de Python, puis de sélectionner *RUN* pour lancer le test.

Lors du test des broches GPIO configurées en sortie, une seule LED basse consommation connectée entre une broche et GND est nécessaire pour vérifier le fonctionnement d'une sortie. Une résistance de 1,8 kΩ peut être utilisée comme résistance en série pour la LED, mais sa valeur n'est pas vraiment critique. Elle limitera le courant traversant la LED si elle est directement connectée à la tension d'alimentation positive. Les sorties sont testées séquentiellement en quatre groupes de huit broches maximum chacun, nommés IOA à IOD. Du fait de la sortie à drain ouvert, la tension aux bornes d'une LED (rouge) plus une résistance est d'environ 2,6 V, lorsque l'alimentation sélectionnée pour les sorties (JP3) est de 5 V. Connectez la résistance et la LED à l'une des sorties sélectionnées et elle s'allumera pendant 0,2 s. La fréquence de répétition de cette impulsion dépend de la taille du groupe : 1,6 s pour les groupes A à C (chacun de huit sorties) et seulement 0,4 s pour le groupe D (deux sorties). Modifiez la valeur « IOA » avec un autre groupe de la ligne :

```
for i in IOA:          # leds blink 0.2 s in IOx group
```

afin de tester les autres groupes de sorties. Bien sûr, GPIO0 et GPIO1 (ID_SD et ID_SC) peuvent aussi être ajoutés à l'un des groupes.

Le programme de test des broches GPIO configurées en entrée utilise une E/S comme sortie pour indiquer que l'entrée testée fonctionne ; c'est GPIO3 par défaut. Connectez une résistance de 1,8 kΩ et une LED entre la broche 5 (IO3') de K2 et GND. Une seule entrée à la fois est testée dans le code source, pour s'assurer que seule celle-ci fonctionne comme entrée. Changez le numéro dans la ligne suivante pour tester une autre broche GPIO comme entrée :

```
IN1 = 2  #selected GPIO to test as input
```

Le programme affiche également la broche GPIO sélectionnée et son niveau d'entrée. Les résistances de rappel des entrées sont

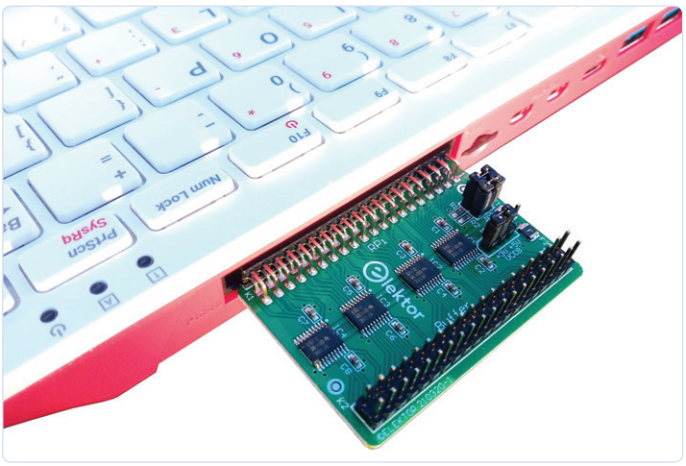


Figure 3. La carte tampon branchée sur le Raspberry Pi 400.

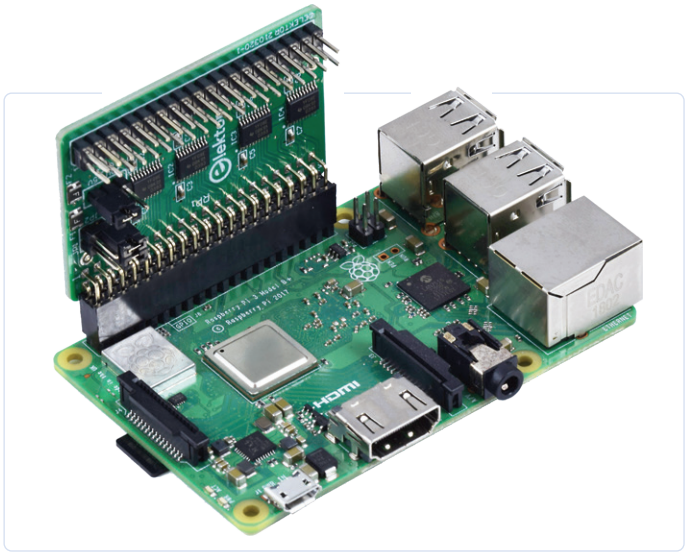


Figure 4. La carte peut également être utilisée avec des cartes Raspberry Pi « classiques ».

activées. Ainsi, pour que la LED connectée s'allume, la broche d'entrée actuelle doit être reliée à la masse. Ceci étant fait, la sortie change. Enfin, sélectionnez une autre broche GPIO pour la sortie afin de pouvoir également tester GPIO3 comme entrée. Bien entendu, il existe de nombreuses façons de tester les GPIO. Si vous avez une méthode plus efficace et/ou plus rapide, merci de nous en faire part.

Cette carte tampon vous permet de connecter en toute tranquillité un matériel nouveau au Raspberry Pi 400, en réduisant considérablement le risque qu'il soit endommagé en cours d'expérimentation. Mais réduire les risques grâce à cette carte tampon ne constitue pas une garantie. Bien souvent, le bon sens pourra aussi être très utile.

210320-04

Contributeurs

Conception, texte et illustration : **Ton Giesberts**
Rédaction : **Luc Lemmens**
Mise en page : **Giel Dols**
Traduction : **Asma Adhimi**

Des questions, des commentaires ?

Contactez Elektor (redaction@elektor.fr).

PRODUITS

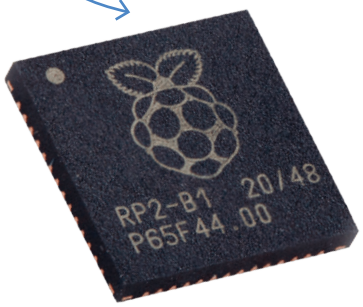
- **Carte tampon pour Raspberry Pi 400**
  www.elektor.fr/19884
- **Kit Raspberry Pi 400 – Raspberry Pi 4-based PC Kit (EU) + Free GPIO Header**
  www.elektor.fr/19431
- **Raspberry Pi 400 – Raspberry Pi 4-based PC Kit (US) + Free GPIO Header**
  www.elektor.fr/19429

LIENS

[1] Carte tampon pour Raspberry Pi, Elektor Labs : www.elektormagazine.fr/labs/raspi-buffer-board
[2] « Carte tampon pour Raspberry Pi », Guy Weiler, Elektor 11-12/2018 : www.elektormagazine.fr/180430-04
[3] Documentation Raspberry Pi sur le fichier *config.txt* : www.raspberrypi.com/documentation/computers/config_txt.html
[4] Carte tampon pour Raspberry Pi 400 dans la boutique Elektor : www.elektor.fr/raspberry-pi-400-buffer-board
[5] Page Elektor Labs de ce projet : https://bit.ly/3GdRJ4G

cartes **Raspberry Pi** RP2040 **à gogo**

Mathias Claußen (Elektor)



Le RP2040 est le premier microcontrôleur conçu par la Fondation Raspberry Pi. D'abord intégré sur la carte Raspberry Pi Pico, destinée aux électroniciens, il a également trouvé sa place, depuis son lancement, sur des cartes et des kits de fournisseurs tiers. Il est temps de découvrir ce que ces fabricants proposent !

Le RP2040 est certainement une alternative intéressante aux microcontrôleurs plus connus. Non seulement le rapport qualité-prix de cette puce est impressionnant, mais elle est en outre actuellement disponible. La documentation et le support technique de la Fondation Raspberry Pi sont certains de ses points forts, ce qui en fait un bon choix pour les débutants dans cet environnement. Les toutes dernières informations relatives à cette puce sont accessibles dans les articles [1] [2][3], le webinaire [4] ou les vidéos [5] d'Elektor.

La carte Raspberry Pi Pico – celle du fabricant, sur laquelle est incorporé le RP2040 – est équipée d'un minimum de matériel supplémentaire permettant de maintenir son prix à environ 5 €. Un an après sa sortie, le circuit intégré RP2040 a trouvé sa place sur un certain nombre de cartes tierces, dotées d'un large éventail de périphériques. Nous examinons ici de plus près quelques-unes de ces cartes pour vous aider à déterminer celle qui correspond à vos besoins.

Raspberry Pi Pico

La carte Raspberry Pi Pico (**fig. 1**) contient le minimum de matériel nécessaire au fonctionnement du RP2040. La carte comporte une LED verte pilotable par l'utilisateur et un convertisseur Buck-Boost CC/CC permettant d'alimenter la carte à partir d'une source de tension externe comprise entre 1,8 à 5,5 V, ou via la tension de 5 V fournie par le port USB. La carte Raspberry Pi Pico est toujours l'une des meilleures en ce qui concerne le rapport puissance/prix, surtout si vous souhaitez simplement acquérir de l'expérience avec le RP2040 et son environnement de développement. Peut-être avez-vous déjà une collection de modules, de composants ou de capteurs externes que vous pouvez interfacer avec la carte. Elle bénéficie de la publication d'un certain nombre de livres et il existe de nombreuses ressources sur l'internet permettant l'autoformation. Vous aurez besoin de matériel supplémentaire si vous souhaitez aller au-delà des éléments de base (voir le kit d'expérimentation Raspberry Pi Pico d'Elektor ci-dessous). Sans trop d'efforts, la carte Pico peut également servir de débogueur pour un autre RP2040. À 5 € l'unité, elle est très abordable, et si vous voulez une Pico avec des connecteurs pré-soudés et un câble micro-USB compatible (**fig. 2**), ne cherchez pas ailleurs que dans l'e-choppe Elektor, où vous trouverez également des cartes d'extension intéressantes pour la Raspberry Pi Pico.

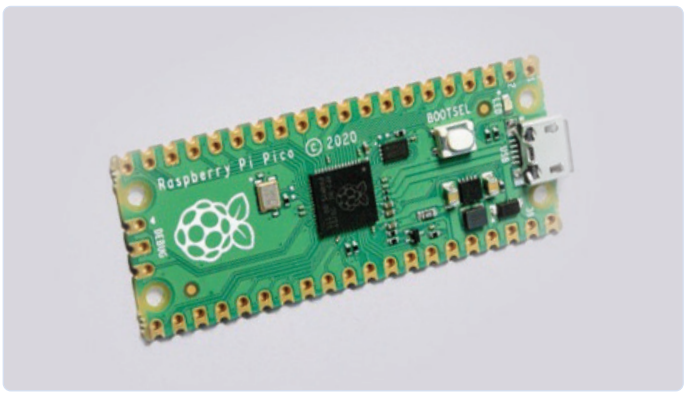


Figure 1. Raspberry Pi Pico. (Source : Raspberry Pi)

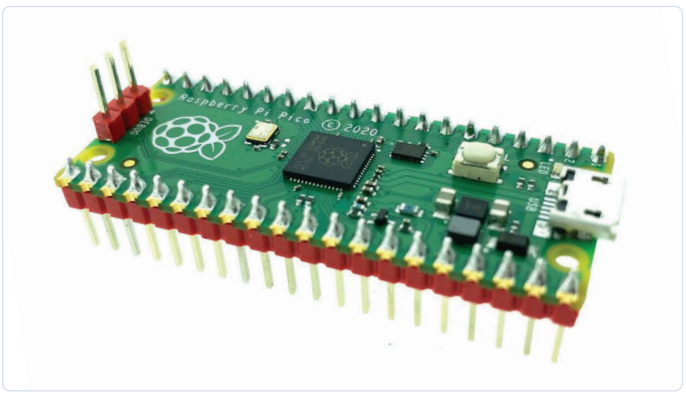


Figure 2. Raspberry Pi Pico avec barrettes. (Source : Elektor)

Feather RP2040 d'Adafruit

La carte Feather RP2040 d'Adafruit (**fig. 3**) est une carte au format Feather, basée sur un RP2040. La différence par rapport au Raspberry Pi Pico réside principalement dans le nombre de périphériques et la capacité de mémoire Flash intégrée à la carte. La Raspberry Pi Pico ne dispose que de 2 Mo, tandis que la carte Feather contient 8 Mo de mémoire Flash SPI, ce qui offre beaucoup plus d'espace pour vos propres logiciels. Une LED RVB est intégrée à la carte, ainsi qu'un port USB-C et un connecteur STEMMA QT pour brancher les périphériques Qwiic, STEMMA QT ou Grove I2C. Le connecteur de batterie JST PH permet d'alimenter la carte par une batterie LiPoly monocellulaire qui peut être chargée directement via le port USB.

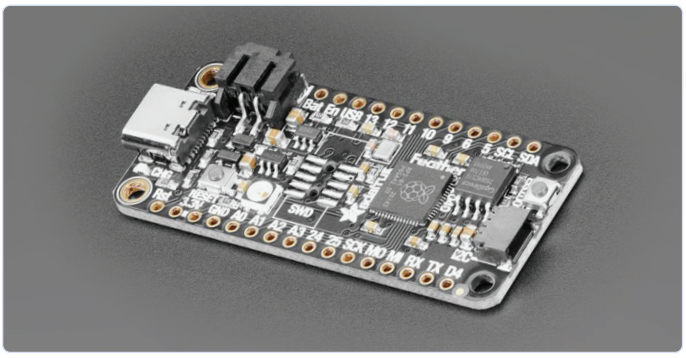


Figure 3. Feather RP2040 d'Adafruit. (Source : Adafruit)

Thing Plus - RP2040 de SparkFun

La carte Thing Plus - RP2040 de SparkFun (**fig. 4**) est très similaire à la carte Feather RP2040 d' Adafruit, même pour l'affectation des broches. Cette carte est équipée de 16 Mo de mémoire QSPI Flash (au-dessous de la carte), ce qui est le maximum adressable par le RP2040. Elle est également dotée d'une LED RVB et de trois LED d'état. Comme pour la Feather RP2040 d'Adafruit, elle comprend un circuit de charge pour une batterie unicellulaire au lithium et un connecteur Qwicc. Un logement pour carte micro-SD est également installé sous la carte et les cartes SD peuvent être adressées en tant que mémoire de masse en utilisant la fonction d'automate fini PIO du RP2040. Un webinaire [4] révèle comment la flexibilité des automates finis PIO a conduit à une séquence inhabituelle d'affectations GPIO pour l'interface du lecteur de carte SD de Sparkfun. Tous les possesseurs de périphériques utilisant le connecteur Qwiic ou des extensions avec un brochage de carte Feather devraient examiner de plus près cette carte !

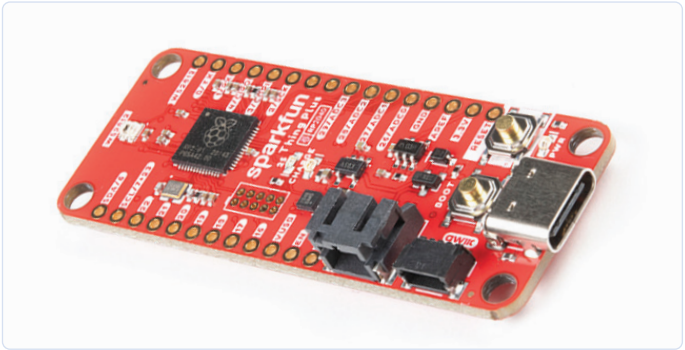


Figure 4. Thing Plus RP2040 de SparkFun. (Source : SparkFun)

Arduino Nano RP2040 Connect

La carte Arduino Nano RP2040 Connect (**fig. 5**) ajoute les fonctions de communication Wi-Fi et Bluetooth au microcontrôleur RP2040. Pour 27 €, ce n'est certainement pas la carte la moins chère, mais elle est bien équipée avec à peu près tous les périphériques embarqués que vous pouvez souhaiter. Elle est dotée de 264 Ko de mémoire RAM statique (SRAM) et de 16 Mo de mémoire flash et elle possède un module ublox NINA-W102 Wi-Fi et BLE v4.2, une centrale inertielle (IMU) à 6 axes (STM LSM6DSOXTR), un microphone (MP34DT05) et une puce de chiffrement ATECC608A de Microchip. Elle est bien sûr prise en charge par l'EDI Arduino et malgré tous les périphériques supplémentaires, avec ses dimensions de 43,18 mm × 17,78 mm, elle est considérablement plus petite que la carte Raspberry Pi Pico.

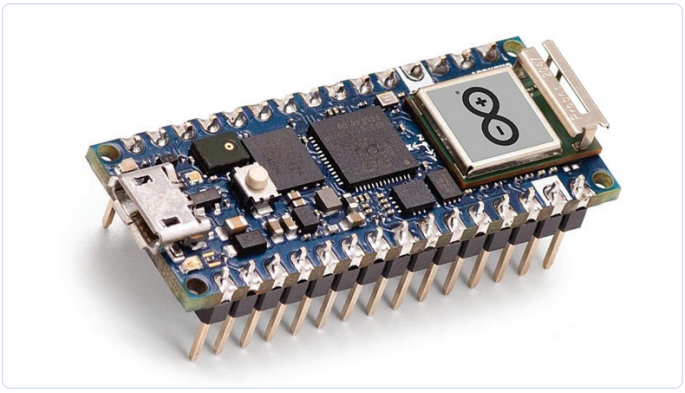


Figure 5. Arduino Nano RP2040 Connect. (Source : Arduino.cc)

Le matériel embarqué est plus que suffisant pour faire ses premiers pas avec un RP2040 et pour réaliser des projets encore plus ambitieux. La possibilité d'échanger des données via Bluetooth et Wi-Fi, ainsi que la présence du microphone intégré, laissent penser qu'il est possible de réaliser une application d'IA. En revanche, je ne suis pas sûr de pouvoir recommander l'Arduino Nano RP2040 Connect à un novice dans cet environnement. Si vous avez l'intention de commencer à développer des applications Wi-Fi et Internet des Objets (IdO), il est préférable de vous orienter vers la carte ESP32 ou la nouvelle ESP32-C3. La fiche technique du module NINA-W102 monté sur l'Arduino Nano RP2040 Connect indique qu'un SoC ESP32 se cache en fait à l'intérieur pour prendre en charge les communications Wi-Fi et Bluetooth. Ceux qui ont une certaine expérience du Wi-Fi et du Bluetooth, et qui souhaitent tester un RP2040 avec une connexion dans le nuage, pourraient s'intéresser à ce que cette carte a à offrir. Mon collègue Clemens Valens a déjà publié une vidéo succincte à ce sujet, à retrouver sur la chaîne YouTube d'Elektor [6].

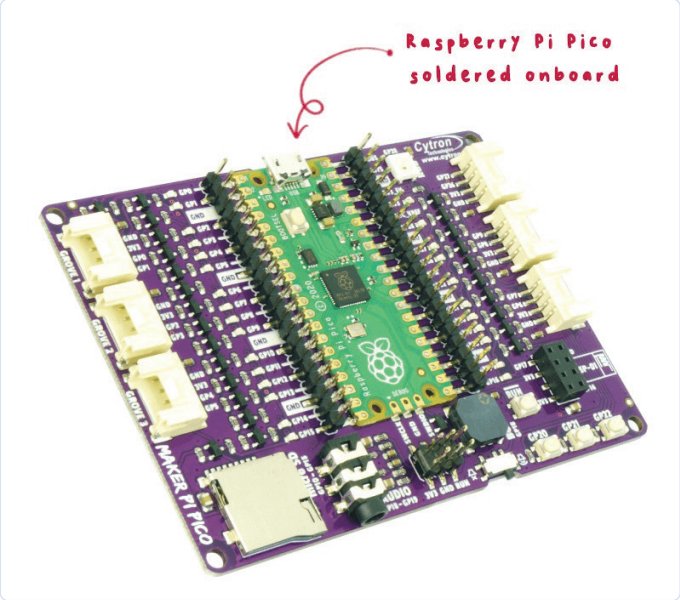


Figure 6. Maker Pi Pico de Cytron. (Source : cytron.io)

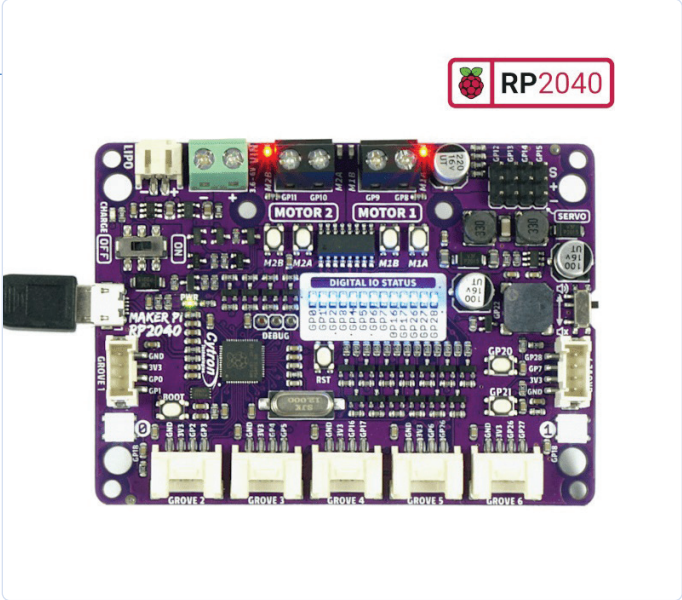


Figure 7. Maker Pi RP2040 de Cytron. (Source : cytron.io)

Maker Pi Pico de Cytron (avec un Raspberry Pi Pico soudé)

Si vous voulez commencer à faire des expérimentations avec le Raspberry Pi Pico et le RP2040, cette plateforme (**fig. 6**) constitue un bon point de départ. La version qui utilise une carte Raspberry Pi Pico soudée est proposée pour environ 18 €.

La principale différence entre cette carte et les autres mentionnées jusqu'à présent est son format. La Maker Pi Pico Base de Cytron est une plateforme expérimentale sur laquelle est monté un Raspberry Pi Pico. Toutes les broches du Raspberry Pi Pico sont ramenées sur deux rangées de connecteurs, de sorte qu'elles peuvent facilement être sondées avec un appareil de mesure. La carte offre également un certain nombre de connexions Grove pour connecter des périphériques compatibles. Chaque broche GPIO de la carte est associée à une LED, ce qui vous permet de vérifier rapidement son état. Il y a également un logement micro-SD et une prise pour une carte Wi-Fi ESP-0, ainsi qu'un certain nombre de boutons-poussoirs et un buzzer.

Cette plateforme est idéale si vous disposez déjà d'autres composants compatibles avec Grove ; il vous suffit de les brancher et de commencer à travailler avec la carte Raspberry Pi Pico. Les connecteurs permettent également de brancher rapidement, et en toute sécurité, d'autres modules. Les périphériques montés comprennent un buzzer (avec un commutateur de désactivation), une LED RVB WS2812, des boutons et un logement pour carte SD, avec lesquels vous pouvez vous lancer pas à pas dans des projets de plus en plus ambitieux avec la carte Pico. Le connecteur ESP-01 offre également la possibilité de configurer rapidement un projet Wi-Fi. L'interface de débogage à trois broches du Raspberry Pi Pico est également disponible sur les broches du connecteur de la carte.

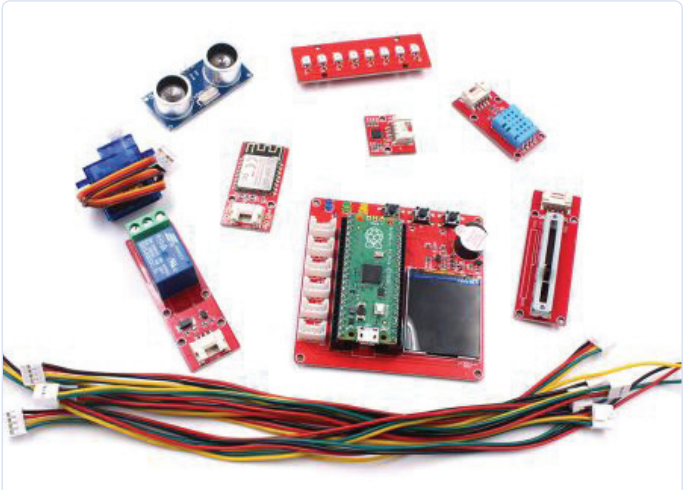


Figure 8. Kit d'expérimentation Raspberry Pi Pico d'Elektor. (Source : Makerfabs)

Maker Pi RP2040 de Cytron

Vous vous intéressez à la commande de moteurs et à la robotique ? Avec le Raspberry Pi PR2040 ? Il peut parfois être utile de pouvoir piloter de petits moteurs ou un moteur pas à pas avec le Raspberry Pi Pico. C'est exactement l'objectif de la carte Maker Pi RP2040 de Cytron (**fig. 7**). Elle contient un circuit intégré de commande de moteur (MX1508/TC1508) avec deux ponts en H pour piloter deux moteurs CC à basse tension ou un moteur pas à pas. Si vous n'êtes pas sûr de savoir comment utiliser un pont en H ou comment commander un moteur, jetez un coup d'œil à mon article au sujet des fondamentaux [7]. Il est possible d'utiliser de petits moteurs d'une puissance allant jusqu'à 6 V, et 1 A par canal, directement à partir de cette carte. Elle offre également la possibilité de connecter directement jusqu'à quatre servos. L'alimentation de la carte provient du port USB, d'une batterie LiPo ou d'une alimentation externe de 3,6 V à 6 V. Un circuit de charge LiPo est inclus pour la gestion de la batterie et l'alimentation de toutes ces sources peut être coupée avec un interrupteur embarqué.

La puce RP2040 est montée sur la carte Maker Pi RP2040 de Cytron et offre exactement la même capacité de mémoire Flash (2 Mo) que la carte Raspberry Pi Pico. La carte comprend également 13 LED d'état GPIO, deux LED WS2812, un buzzer, deux boutons et sept ports Grove pour une extension facile.

Kit d'expérimentation Raspberry Pi Pico d'Elektor

À 45 €, le kit d'expérimentation Raspberry Pi Pico d'Elektor (**fig. 8**) propose la carte la plus chère de la gamme. La base de ce kit est le Raspberry Pi Pico, qui se branche directement sur cette carte,

de sorte qu'il peut facilement être remplacé si nécessaire. La carte principale du kit elle-même comporte des boutons, des LED, des buzzers, un écran TFT et des connecteurs Grove. Avec des accessoires tels que les LED WS2812, le capteur de température et d'humidité DHT11, le relais, le potentiomètre, le capteur de distance à ultrasons, le servo, le gyroscope et le capteur d'accélération MPU6050 ainsi que l'ESP8266, vous obtenez un kit qui convient aussi bien aux débutants qu'aux utilisateurs avancés qui aiment l'expérimentation. Comme ces composants sont faciles à connecter sur la carte en tant que modules (via les connecteurs Grove), la carte Raspberry Pi Pico offre une réelle flexibilité d'utilisation. Grâce à l'écran de 1,44 pouce (avec un contrôleur ST7735), vous pouvez également développer vos premières applications graphiques en Python ou C/C++. Si vous n'avez pas encore de modules ou d'autres composants, ce kit représente un bon rapport qualité-prix et offre une introduction très complète au monde du RP2040.

Quelle est la meilleure carte ?

Chacun doit répondre par lui-même à cette question. La réponse dépend considérablement de ce que vous envisagez de faire. Chaque carte, chaque kit, a ses avantages et ses inconvénients. En fonction de votre budget et de vos besoins, vous devez déterminer quelles sont les caractéristiques les plus indispensables. Avez-vous besoin d'une carte avec tous les périphériques déjà installés, ou vaut-il mieux commencer par un kit de base et ajouter ensuite des modules spécifiques en fonction de vos nécessités ? Votre application doit-elle disposer d'une fonction de gestion/de charge de la batterie ? La carte sera-t-elle destinée à votre usage personnel, ou pensez-vous qu'elle sera un cadeau inspirant pour un membre de la nouvelle génération d'ingénieurs ou d'électroniciens ?

Il est clair que nous ne pouvons pas recommander une carte en particulier. À l'évidence, le microcontrôleur RP2040 offre de nombreuses possibilités pour les développeurs débutants et expérimentés et, surtout, il est actuellement en stock.

210629-04

Des questions, des commentaires ?

Envoyez un courriel à l'auteur (mathias.claussen@elektor.com) ou contactez Elektor (redaction@elektor.fr).

Contributeurs

Analyse et texte : **Mathias Claußen**
Rédaction : Jens Nickel
Mise en page : Harmen Heida et Giel Dols
Traduction : Asma Adhimi

PRODUITS

- **Kit d'expérimentation du Raspberry Pi Pico d'Elektor**
  www.elektor.fr/19834
- **Maker Pi RP2040 – carte pour la robotique avec Raspberry Pi RP2040 de Cytron**
  www.elektor.fr/19926
- **Maker Pi Pico de Cytron**
  www.elektor.fr/19706
- **Arduino Nano RP2040 Connect avec connecteurs**
  www.elektor.fr/19754
- **Thing Plus - RP2040 de SparkFun**
  www.elektor.fr/19628
- **Feather RP2040 d'Adafruit**
  www.elektor.fr/19689
- **Raspberry Pi Pico RP2040**
  www.elektor.fr/19562
- **Raspberry Pi Pico RP2040 (avec connecteurs soudés)**
  www.elektor.fr/19568

LIENS

[1] « Carte Raspberry Pi Pico à RP2040 », site Elektor : www.elektormagazine.fr/news/carte-raspberry-pi-pico-a-rp2040
[2] « Place au microcontrôleur Raspberry Pi RP2040. Hue Pico ! », site Elektor : www.elektormagazine.fr/news/microcontroleur-raspberry-pi-rp2040-pico-board
[3] Raspberry Pi Pico (avec connecteurs soudés) : www.elektormagazine.fr/news/raspberry-pi-pico-mcu-with-preinstalled-pin-headers-fr
[4] Eben Upton et Nathan Seidle à propos de la carte Raspberry Pi Pico et du RP2040 : www.elektormagazine.com/news/eben-upton-nathan-seidle-sparkfun-raspberry-pi-pico-rp2040
[5] Elektor TV sur YouTube : www.youtube.com/user/ElektorIM
[6] Clemens Valens, « Board Review: Arduino Nano RP2040 Connect », ElektorTV : www.youtube.com/watch?v=2EnCf64zZSA
[7] Mathias Claussen, « Commande de moteurs : les ponts en H », Elektor Janvier/Février2022 : www.elektormagazine.fr/210491-04

approche DIY
de la sécurité et de l'espionnage électronique

Chauffez ou refroidissez la SRAM

Luka Matic (Croatie)

Nous abordons ici le piratage des données en SRAM (RAM statique) et leur chiffrement par des méthodes non répertoriées qui étendent la capacité de rétention de ces SRAM. Une fois la théorie assimilée, vous pourrez imaginer vos propres variantes ou améliorer et automatiser ce qui est exposé ici. N'hésitez pas à essayer !

Note de l'éditeur : cet article est un extrait du livre intitulé *A Handbook on DIY Electronic Security and Espionage* (« Approche DIY de la sécurité et de l'espionnage électronique ») formaté et légèrement modifié pour correspondre aux normes éditoriales et à la mise en page du magazine *Elektor*. Puisque cet article est extrait d'une publication plus vaste, certains termes peuvent faire référence à des passages du livre d'origine situés ailleurs. L'auteur et l'éditeur ont fait de leur mieux pour l'éviter et seront heureux de répondre aux questions – Pour les contacter, voir l'encadré « Des questions, des commentaires ? ».

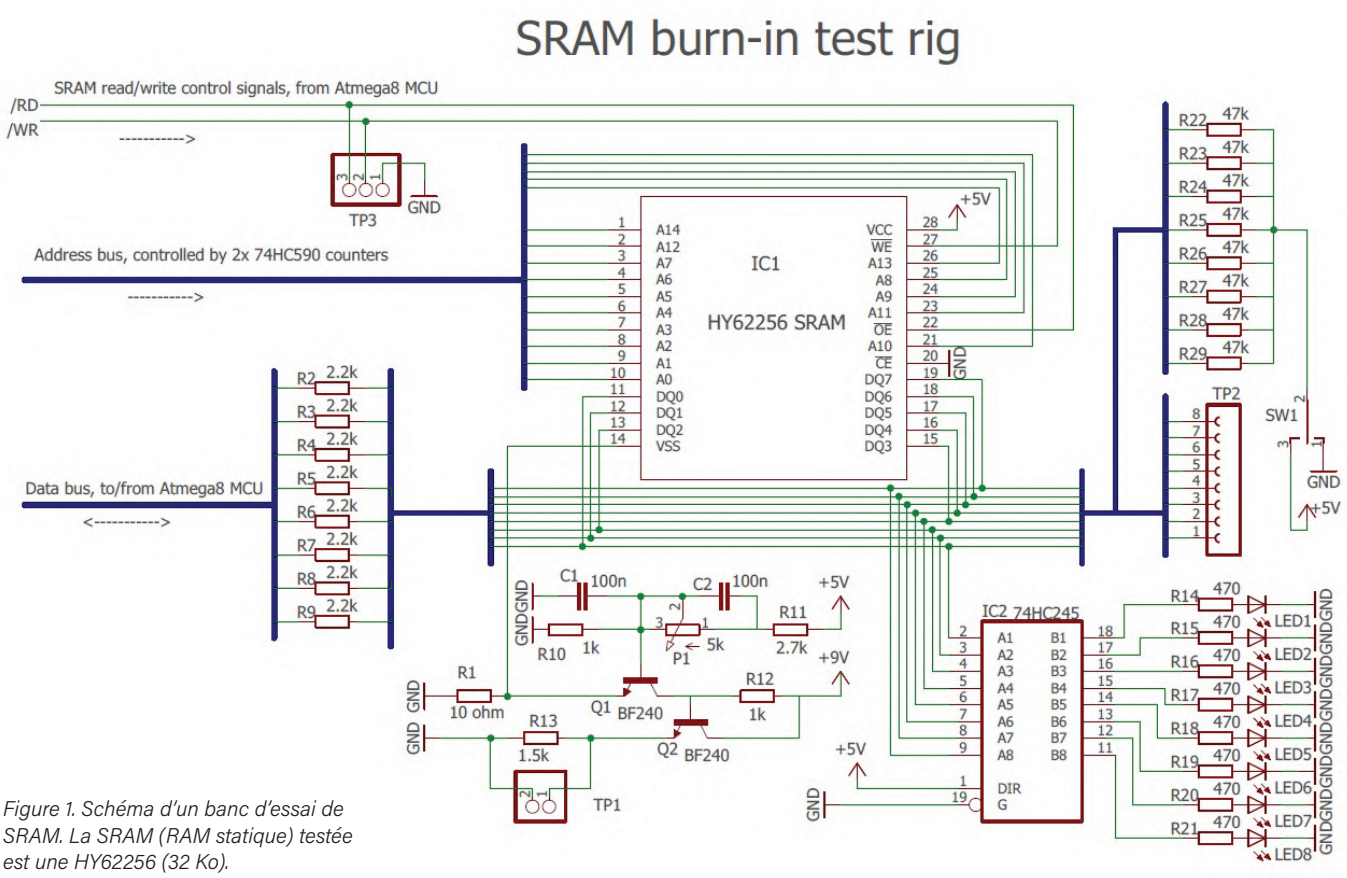


Figure 1. Schéma d'un banc d'essai de SRAM. La SRAM (RAM statique) testée est une HY62256 (32 Ko).

Récupération de données d'après leur empreinte en SRAM

La **figure 1** donne le schéma du banc d'essai utilisé. L'amplificateur analogique à deux étages (Q1 à base commune et Q2 à collecteur commun, pour obtenir la bande passante max.) amplifie la chute de tension sur R1 (image du courant d'alimentation de la SRAM) que l'on observe sur un oscilloscope via le point de test TP1. Un microcontrôleur (MCU) ATmega8 (non représenté sur les schémas) pilote le dispositif. On règle le bus d'adresse sur l'adresse de la mémoire à tester. C'est assez lent, car on utilise deux compteurs à 8 bits 74HC590. C'est acceptable, car il n'est pas nécessaire que l'adresse change vite. Le MCU lit/écrit les données sur le bus et contrôle les commandes /RD et /WR de la SRAM. Ces deux signaux sont utilisés pour déclencher les mesures de l'oscilloscope (point de test TP3). Les résistances R2 à R9 découplent le bus de données, pour le cas où il est activé à la fois par le MCU et la SRAM.

Les résistances R22 à R29 polarisent le bus d'adresses à $+V_{cc}$ ou à la masse (selon SW1) pour mesurer les temps d'accès en lecture et de montée/descente du bus de données, via TP2. Le tampon octal IC2 sert à piloter les huit LED pour faciliter la surveillance du bus de données.

Les variables à mesurer sont le courant consommé en lecture/écriture de la SRAM, et les tensions (c.-à-d. les temps de retard/montée/descente) sur le bus de données. Les variations des signaux mesurés révèlent les marques résiduelles (l'empreinte) des données stockées auparavant et permettent, on l'espère, d'extraire les octets écrits avant que la SRAM ne soit mise hors tension. Pour cela, les cellules SRAM doivent avoir conservé des données figées pour une période relativement longue. Ces durées peuvent varier selon la puce SRAM utilisée.

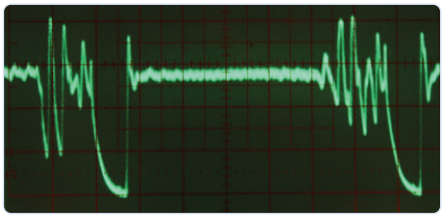


Figure 2. Courant I_{dd} capturé sur un oscilloscope Tek 466 pendant l'écriture des octets 0xDE (à gauche) et 0x01 (à droite) dans la SRAM (à 200 ns/div).

Une résistance de 100 Ω, 5 W plaquée sur la SRAM, avec une sonde de température en sandwich, permet de la chauffer et de la tester à une température élevée, jusqu'à 80-90 °C. Cf. la photo du banc d'essai (fig. 5). Les variations des temps de montée/descente de 1-2 ns doivent être mesurées de manière fiable, ce qui nécessite un oscilloscope à 100 MHz. J'ai utilisé un oscilloscope

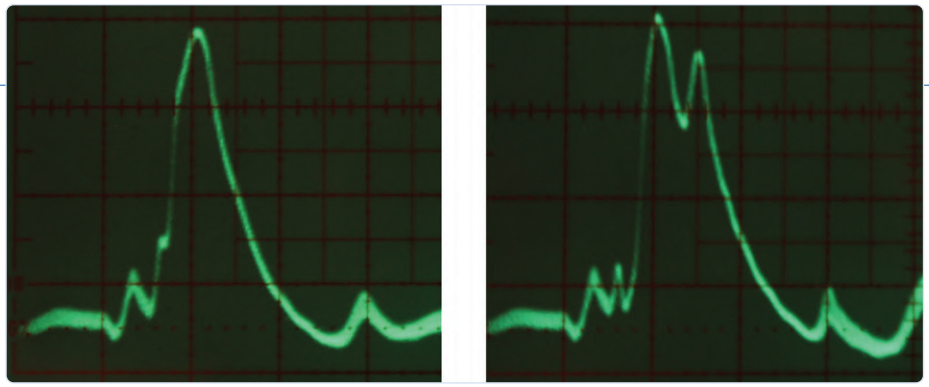


Figure 3. Courant I_{dd} pendant la lecture mémoire de 0xDE (à gauche) et 0x01 (à droite) (à 200 ns/div).

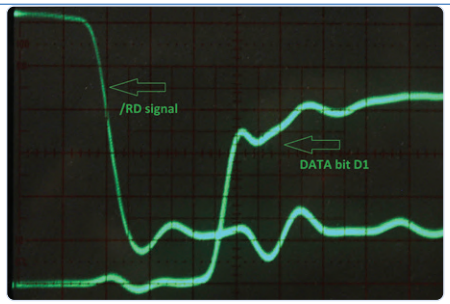


Figure 4. Formes du signal /RD et d'un bit de données lors de la lecture de la SRAM (à 10 ns/div).

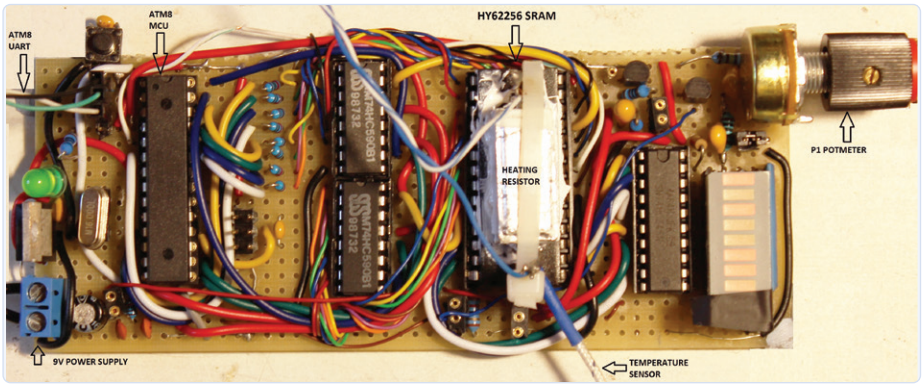


Figure 5. Le banc d'essai de « burn in » des SRAM de l'auteur.

analogique Tektronix 466 vintage à écran à persistance variable avec « flood gun » pour signaux rapides non répétitifs. Sous la rubrique *Rétronique*, *Elektor* a publié un brillant article sur ce type d'oscilloscope à mémoire [1]. La persistance n'est pas indispensable, car les séquences de test de lecture/écriture peuvent être exécutées en boucles répétitives contrôlées par le MCU. J'ai acquis l'oscilloscope Tek 466 à Ljubljana (Slovénie) pour 150 € au marché aux puces électronique.
Commençons par faire à l'oscilloscope quelques mesures de consommation de courant de la RAM et de la tension /RD, cf. **figures 2, 3** et **4**. Les mesures du courant lors de l'écriture et/ou de la lecture de SRAM dans des cellules marquées (ayant longtemps conservé le même octet) ou non (celles dont les bits ont changé sans cesse pendant la même durée), et la comparaison des résultats, peuvent être utilisées pour récupérer les données. La comparaison des temps d'accès en lecture et des formes d'onde (fig. 4) entre les bits sur le bus de données peut également aider à récupérer les données.
La **figure 5** est une photo de mon banc d'expérimentation de la récupération de données en SRAM.

Voici la procédure : j'écris une suite de constantes en SRAM que je laisse ainsi pour la marquer (*burn-in* en anglais). Je programme des compteurs incrémentés toutes les 10 ms dans d'autres cellules SRAM. Je chauffe la SRAM à 80 °C (cela augmente le nombre de porteurs chauds et laisse des traces plus profondes) et je maintiens la puce sous tension pendant 12 h tandis que le MCU continue à incrémenter les compteurs (cela évite de créer des traces dans les cellules concernées). 12 h plus tard, j'éteins la résistance chauffante et la SRAM revient à la température ambiante. Voici le résultat des tests effectués sur deux cellules SRAM :

- à l'adresse 0x204F était stockée une constante (0x66), et on espère qu'elle a pu laisser une trace ;
- à 0x7FF1 tournait un compteur, qui normalement n'a laissé aucune trace.

Les mesures des temps d'accès en lecture (cf. fig. 4) ont montré de très faibles différences (moins d'1 ns) entre les bits « 0 » et « 1 », et je ne les ai donc pas considérées comme pertinentes. Seuls les MOSFET des bistables (qui mémorisent « 0 » ou « 1 ») sont affectés par les effets du *burn-in*. Les MOSFET d'accès à la SRAM et de transfert au bus de données à la lecture ne le sont pas. Cependant, les mesures du courant d'alimentation I_{dd} lors de l'écriture de différents octets dans les cellules marquées ont donné de meilleurs résultats. Cette méthode est meilleure pour d'autres configurations, par ex. pour lire la SRAM depuis un MCU, car elle ne nécessite pas d'accès physique aux 8 bits du bus de données de la SRAM.
Cette fois encore, la cellule à l'adresse 0x204F qui contenait l'octet 0x66 avait subi un *burn-in* à 80 °C. La cellule à 0x7FF1 n'a pas subi de *burn-in* (tous ses bits étant inversés cycliquement). Les mesures (**fig. 6** et **7**) montrent que l'écriture d'un motif octal avec plus de zéros dans une cellule de mémoire marquée consomme plus d'énergie que l'écriture du même motif dans une cellule non marquée. La puissance requise par l'écriture de 0x66 (une valeur marquée) et de 0x99 (complément à 1 de cette valeur) dans une cellule brûlée est notablement plus élevée que dans la cellule non marquée. Mais ces mesures ne suffisent pas à établir l'octet qui a été marqué dans la cellule. Il faut un post-traitement mathématique (filtrage, corrélation avec des motifs connus, etc.).
La SRAM 62256 est un composant ancien assez insensible au marquage par *burn-in*, loin des composants à haute intégration actuels.
Il faut beaucoup d'autres expériences avec d'autres types de SRAM, dans des conditions de marquage variées. Certains auteurs rapportent des résultats contradictoires, ce qui indique donc que les effets résiduels du marquage ne sont ni assez étudiés ni bien compris. Cela ouvre un nouveau et vaste champ de recherche.

Réduction de la tension d'alimentation

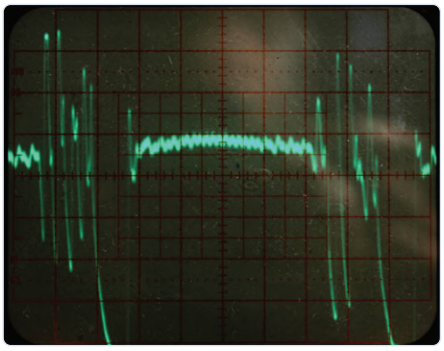
Outre ce que nous avons vu pour extraire les traces de données, il y a une autre approche que je n'ai pas encore essayée. Il s'agit de réduire progressivement la tension d'alimentation jusqu'à ce qu'une cellule de mémoire produise une erreur de lecture ou bien d'écriture. Les cellules marquées et non marquées sont constamment comparées. En général, l'erreur, par ex. lors de l'écriture de 0x00 ou 0xFF, n'affecte que certains bits d'une cellule de mémoire marquée selon l'état de marquage de ces bits (0 ou 1). C'est parce que les tensions de seuil des grilles MOSFET des cellules marquées sont légèrement augmentées ou diminuées par le *burn-in*.

Application

La méthode du « burn-in » pourrait être exploitée pour des communications secrètes améliorées par stéganographie avancée avec de nouveaux types de SRAM à haute intégration plus sensibles au marquage que la « vieille » 62256 testée ici. Une puce SRAM étudiée dans ce but sera utilisable si elle présente un marquage significatif après quelques heures de chauffage à 80 °C (sous tension et avec un message secret chiffré stocké) et si ce marquage persiste pendant 10 à 15 jours une fois la SRAM à température ambiante puis éteinte (retirée du circuit). La méthode de communication fonctionnerait comme suit :

1. L'« expéditeur » Alice chiffre le message secret pour le « récepteur » Bob, et le stocke dans la SRAM.
2. Alice chauffe la SRAM (sous tension et avec le message secret chiffré) à 80 °C pendant 6 à 12 heures.
3. Alice laisse la SRAM refroidir à température ambiante (sous tension).
4. Alice retire du circuit la SRAM refroidie.
5. Alice met la puce SRAM sous enveloppe et l'envoie à Bob.
6. Si l'« espionne » Ève intercepte le courrier, il a simplement l'air de contenir un composant provenant de RS. Même si elle tente de récupérer les données marquées, elle ne verra rien de suspect, car le message est chiffré. Et n'en conclura pas qu'Alice y a caché des données. Peut-être que la puce vient d'être retirée d'un serveur où elle a conservé des constantes pendant très longtemps, par ex. quelques mois à 30 °C.
7. Quelques jours plus tard, Bob reçoit le courrier et exécute la procédure de récupération des données comme ci-dessus.
8. Bob déchiffre le message.

Saurez-vous imaginer d'autres façons dont Alice, Bob, Eve et d'autres pourraient exploiter les effets de persistance de la mémoire SRAM ? Il y en a beaucoup !



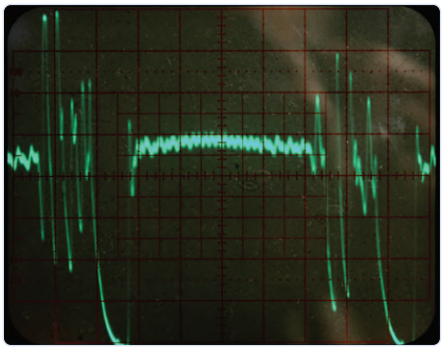
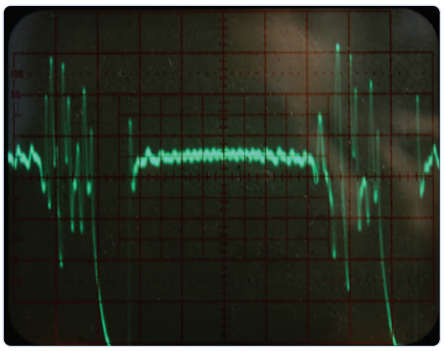


Figure 6. I_{dd} pendant l'écriture des octets 0x00, puis 0xFF à 0x7FF1 (à gauche) et 0x204F (à droite).



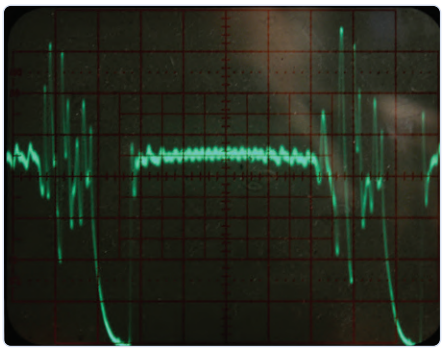


Figure 7. I_{dd} pendant l'écriture des octets 0x66, puis 0x99 à 0x7FF1 (à gauche) et 0x204F (à droite).

Démonstration de l'approche à froid

La 2e démonstration est bien plus simple que la précédente. Le type d'approche à suivre fonctionne probablement pour beaucoup de RAM de types anciens et récents.
Le refroidissement d'une puce jusqu'à –50 °C ne réduit pas de manière significative la conductivité des microcircuits en silicium modérément dopé, et la puce continue de fonctionner normalement, mais hors tension, elle gardera les données, car à –50 °C, les capacités de grille des MOSFET se déchargent très lentement. Le même banc d'expérimentation est utilisé, mais sans résistance chauffante. Des données judicieusement choisies et stockées en SRAM produiront des effets lumineux sur les huit LED du bargraphe. Si ces mêmes effets continuent après la remise sous tension, cela signifie que le contenu de la SRAM est intact. Si des effets irréguliers ou aléatoires apparaissent, c'est que le contenu de la SRAM est perdu. J'ai mesuré le temps maximal de rétention de données à environ –30 °C. C'est facile à atteindre avec un spray réfrigérant (**fig. 8**).
À la température ambiante, les données ne persistent que 0,1 s, mais cela peut atteindre 10 s à –30 °C.
Je refroidis d'abord la SRAM à –30 °C environ et j'augmente progressivement le temps de remise sous tension tout en maintenant la

Figure 8. Spray cryogénique utilisé dans la démo de l'approche à froid.

avec un matériel très bon marché (par ex. l'approche à froid) tandis que d'autres, comme la récupération de traces de données dans la SRAM, peuvent nécessiter un matériel plus sophistiqué et un post-traitement (bien qu'à la portée de l'espion à petit budget !) pour, au-delà de la démonstration du principe, être exploitable en pratique.

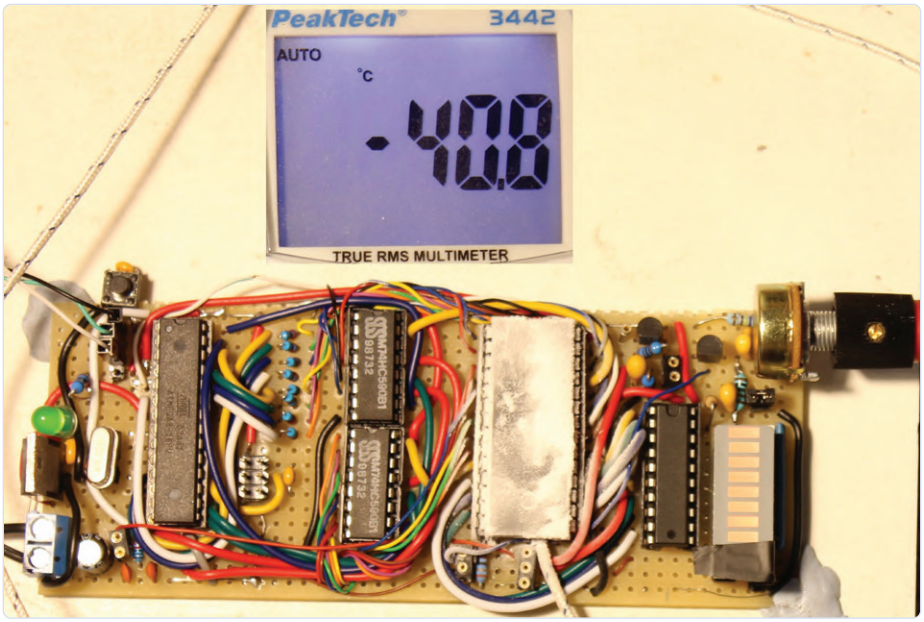


Figure 9. Le circuit intégré SRAM a été refroidi pour obtenir une rétention des données pendant 10 s.

température avec le spray. J'ai atteint un temps de rétention de 10 s à –40 °C (**fig. 9**). Ces résultats étaient attendus, car les broches d'alimentation restent « court-circuitées » par quelques kΩ pendant la mise hors tension, de sorte que la broche V_{dd} peut être considérée comme « à la terre ».

Simplicité surprenante

Comme vu ci-dessus, certaines approches pratiques sont réalisables à la maison

Il ne faut pas oublier que les CI récents à haute intégration sont plus sensibles aux approches présentées. En améliorant les idées évoquées ici, vous pourriez imaginer vos propres dispositifs et procédures d'approche. Il reste beaucoup à faire et donc aucune excuse à l'apathie et à la dépression typiques des ingénieurs du 21e siècle ! Je suis sûr qu'un bon chiffrement à petit budget est possible de nos jours. À mesure que la technologie progresse et que les prix baissent, les choses tournent en faveur d'Alice et de Bob.

210628-04

LIEN

[1] « Tektronix 564 oscilloscope à mémoire (1963) », rubrique Rétronique, Elektor 07-08/2011 : www.elektormagazine.fr/100920

Des questions, des commentaires ?

Envoyez un courriel à l'auteur (luka.matic@gmail.com) ou contactez Elektor (redaction@elektor.fr).

Contributeurs

Texte : **Luka Matic**
Rédaction : **Jan Buiting**
Mise en page : **Giel Dols**
Traduction : **Yves Georges**

identification des cOmposants

Trucs et astuces, bonnes pratiques et autres informations utiles

David Ashton (Australie)

Pour dépanner un circuit imprimé, il est primordial de savoir identifier ses composants. Le développement de vos compétences dans ce domaine vous servira aussi à constituer un stock de pièces de rechange qui pourrait s'avérer utile un jour. Cependant, il n'est pas toujours facile de déterminer qui est quoi. Voici quelques conseils pour vous aider.

De nombreux lecteurs sont, comme moi, des bricoleurs qui récupèrent des composants sur de vieux appareils électroniques. J'ai commencé à m'intéresser à l'électronique à l'adolescence – il y a environ 50 ans – et mon père, qui était vendeur dans une entreprise de machines comptables, récupérait pour moi des cartes de rebut auprès de leurs techniciens. Ces cartes contenaient des transistors, des résistances, des diodes et quelques condensateurs. Les fils étaient pliés sur la face inférieure de la carte et même avec le gros pistolet à souder que mon père m'avait offert, j'avais du mal à enlever certains composants. Mais ça se passait en Rhodésie (aujourd'hui le Zimbabwe), loin du centre de l'univers de l'électronique. Les composants étaient chers et souvent difficiles à se procurer, alors ces cartes étaient de l'or pour moi.

Le boîtier des transistors était très bizarre, mais avec un simple testeur que j'avais construit à partir d'un article de revue, je les ai identifiés comme étant de types NPN. Ils étaient marqués « B686 » et avaient un point de couleur – brun, rouge, orange, jaune ou vert – sur une bosse sur le dessus du boîtier. À cette époque, l'internet n'existait pas et les catalogues de composants étaient rares et chers. Je savais que les transistors marqués « Bxxx » appartenaient souvent à la série japonaise 2SB, mais mon exemplaire du vénérable *Towers International Transistor Selector* m'apprit que le 2SB686 était un transistor de puissance PNP et pas de signal NPN comme les miens. L'identification du véritable numéro de type de ces transistors était donc un défi. Et de taille !

La solution m'a été apportée par un tableau de caractéristiques de transistors publié par la revue *Practical Electronics* que je recevais à l'époque. Il y avait le 2N2926, avec le même boîtier, avec une note indiquant que le point de peinture sur le dessus du boîtier indiquait la plage de gain. Le 2N2926 (**fig. 1**) est un transistor à l'aspect étrange et ce devait être celui-là, même si les miens étaient marqués différemment. J'ai vérifié que les gains de mes transistors correspondaient à ceux indiqués dans le tableau, et cela a conclu l'affaire. Le 2N2926, avec son point de couleur, est un transistor tellement exotique que ce devait être le même, même avec un marquage différent.

J'ai travaillé dans le domaine de l'électronique et des télécoms la majeure partie de ma vie et je continue à démanteler de vieilles cartes pour récupérer des composants – les appareils professionnels fournissent souvent des composants de haute qualité. Depuis ma jeunesse, j'ai relevé de nombreux défis pour identifier des composants de récupération encore plus étranges, pas toujours avec succès. Mais voici quelques-unes des techniques auxquelles j'ai eu recours au fil des ans, y compris un quiz d'identification de composants vous permettant de tester vos propres connaissances. Il y en a de toutes sortes, des plus anciens aux plus récents de type CMS, ainsi que des exemples de mes problèmes et techniques.

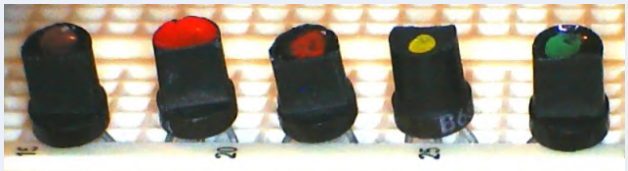


Figure 1. Transistors 2N2926. La couleur indique le gain H_{fe} du transistor : brun 35-70 ; rouge 55-110 ; orange 90-180 ; jaune 150-300 ; vert 235-470.

Trouver des données sur les composants

Lorsque j'ai commencé, les catalogues de composants étaient difficiles à trouver et valaient leur pesant d'or. C'est plus facile aujourd'hui avec l'internet, mais si vous trouvez des informations utiles – codes de couleur, informations des fabricants, fiches techniques, etc. – copiez-les ou imprimez-les. J'ai encore un tableau des données sur les transistors des années passées.

De nos jours, apprenez comment chercher sur l'internet. Il existe une multitude de bons sites de fiches techniques. Google est votre ami ! Mais aidez Google à vous aider. Si vous cherchez une fiche technique, tapez « datasheet » dans le champ de recherche. Et utilisez un numéro de référence aussi court que vous pouvez. J'ai eu récemment plusieurs circuits intégrés avec la même immatriculation « 2026-1SM », probablement leur identifiant de composant. Mais j'ai entré « 2026 datasheet » dans le moteur de recherche qui m'a fourni la fiche technique du MIC2026 de Micrel (aujourd'hui partie de Microchip) – un commutateur de distribution de puissance à deux canaux.

Avec votre identifiant de composant, certains sites de fiches techniques retournent une liste de fiches qui lui correspondent exactement ou partiellement, ou qui le contiennent – cela peut être utile pour affiner votre recherche. J'enregistre la plupart des fiches techniques sur mon disque dur pour ne pas avoir à refaire la recherche, mais c'est une préférence personnelle. J'ai indiqué le lien vers mon site de fiches techniques préféré [1], et je garde quelques sites dans mes favoris, ainsi que certains sites de fabricants.

Comment lire les fiches techniques

La plupart des fiches techniques commencent par une description du composant, suivie des valeurs à ne pas dépasser. Viennent ensuite des détails sur l'utilisation, les brochages, etc. Les informations sur le boîtier se trouvent généralement à la fin, et il est souvent nécessaire de les consulter pour s'assurer que ce que vous avez en main correspond bien à ce qui est décrit – le même nombre de broches et le même boîtier.

Tous les fabricants d'un même composant n'utilisent pas forcément le même boîtier, ce qui peut compliquer la recherche. Un composant produit par plusieurs fabricants aura souvent les mêmes caractéristiques, mais pas toujours, donc trouvez, si possible, une fiche technique du fabricant de votre composant.

Connaissez vos composants

Certains composants ressemblent beaucoup à d'autres. Maintenant, quand je vois un composant qui ressemble à une résistance, en me basant sur sa forme et sa couleur, je peux dire que c'est un condensateur ou une inductance avec un faible taux d'erreur.

La plupart des gens savent que si c'est marqué « 2Nxxxx », c'est un transistor, et si ça ressemble à un circuit intégré avec quatre ou six broches, c'est sans doute un optocoupleur. Et je sais que si ça a l'air d'un transistor marqué *xxNyy* (par ex. « 35N60 ») ou marqué *IRFxxx* (par ex. « IRF540 »), c'est un FET. Ça vient avec l'expérience.

Pendant mes recherches pour cet article, j'ai vu qu'un « V » sur un circuit intégré signifie souvent qu'il est fabriqué par Vishay. Bon à savoir, peut-être cela me fera-t-il gagner du temps à l'avenir.

Comment tester les composants

À peu près au moment où j'ai rencontré ces 2N2926, je me suis construit un simple testeur de transistors avec un vieux multimètre, un interrupteur, un potentiomètre de 500 kΩ et une pile de 1,5 V. Je l'ai toujours, mais maintenant, la plupart des multimètres numériques ont un testeur de transistors intégré, et certains un testeur de condensateurs. Tester des transistors à effet de champ est un peu plus difficile, mais possible dans un labo d'amateur moyen. J'ai un LCR-mètre de base que j'utilise beaucoup, mais j'aimerais en avoir un meilleur.

Si vous pouvez identifier la nature d'un composant (transistor, FET, condensateur, etc.), vous avez fait la moitié du chemin, et vous pourrez peut-être même l'utiliser sans en savoir davantage. Certains composants CMS, en particulier les condensateurs, n'ont aucun marquage et vous devez donc les mesurer pour pouvoir les utiliser. Une « sonde-pincette » pour votre appareil de mesure peut faciliter énormément le test des CMS, et la boutique Elektor propose des brucelles de test très intéressantes qui mesurent à peu près tout, voir l'encadré **Produits**.

Les circuits intégrés, bien sûr, nécessitent des testeurs spécialisés, mais vous pouvez fabriquer vous-même un testeur d'ampli-op et acheter des testeurs de CI logiques si vous en utilisez beaucoup. Avoir un numéro de composant et une fiche technique est un début pour les CI et les transistors, mais de nombreux composants, surtout les passifs, sont utilisables si leur seule valeur est connue. Le test des composants est un art en soi.

Tirez le meilleur parti du marquage des composants

Il y a généralement plusieurs numéros sur un transistor ou un CI et cela vaut la peine de faire un effort pour savoir lequel est le numéro du type. Les préfixes sont souvent omis sur les petits CMS. Familiarisez-vous avec les logos des fabricants. Aller directement sur le site web du fabricant peut vous faire gagner beaucoup de temps. La plupart

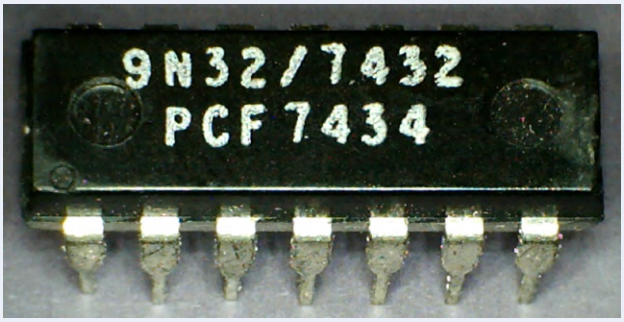


Figure 2. Est-ce un 7432 ou un 7434 ? Supposer que PCF est un préfixe du fabricant est raisonnable, mais ce n'est pas le cas. (Ça ne l'était pas à l'époque, en tout cas !) Il n'y a pas de TTL 7434. C'est un 7432 fabriqué au cours de la 34e semaine de 1974. Le 9N32 est un autre indice, de nombreux circuits intégrés 74xx étaient également marqués 9Nxx, bien qu'il soit très difficile de trouver des informations à ce sujet.

des composants ont un code de date qui indiquait autrefois l'année et la semaine (par ex. 8634), mais de nos jours, il peut s'agir d'un code de lot énigmatique. (Autrefois, un circuit logique TTL série 74 fabriqué en 1974, avec un code de date 74xx (**fig. 2**) pouvait être un casse-tête). Si vous avez plusieurs composants d'un même type, cherchez le code identique sur tous : il s'agit du numéro de composant, les autres sont des codes de date ou de lot sans intérêt.

Les valeurs des composants passifs sont soit indiquées en clair (comme 47 pF), soit sous forme de chiffres ou d'un code couleur de résistance au format *chiffre1, chiffre2, multiplicateur* (le nombre de zéros) sur le composant. Les inductances CMS ont souvent une valeur indiquée en microhenrys de cette façon, ainsi *3R3* correspond à 3,3 µH et *333* à 33 mH (33 000 µH). Les condensateurs peuvent être marqués en picofarads. Un condensateur au tantale marqué *227* vaut 22 × 10^7 pF = 220 µF. Certains composants peuvent avoir cinq ou six anneaux de code couleur, alors l'internet est d'une grande aide pour les décoder. La plupart des petits condensateurs CMS ne sont pas marqués du tout, alors utilisez vos compétences et votre appareil de test de composants pour les vérifier. Et procurez-vous une loupe ou un microscope USB (voilà comment j'ai pris la plupart des photos de cet article) pour une lecture bien plus facile des inscriptions minuscules sur les petits composants.

L'internet met de nombreuses ressources à votre disposition – cherchez « IC Manufacturers logos » ou « SMD codes » si vous avez besoin de plus d'informations. Et cherchez « EIA-96 » pour décoder les résistances CMS avec ce qui ressemble à un code bizarre de deux chiffres et une lettre.

Tenez compte du contexte

Si vous savez de quel genre d'appareil provient le composant, cela peut vous donner un indice sur sa nature. Une alimentation a des chances d'avoir un circuit intégré MLI à découpage, tandis qu'une carte audio est plus susceptible d'avoir des ampli-op.

Ne croyez pas que vous allez tout identifier !

J'ai un sac de transistors marqués *0V8F* qui ont obstinément refusé d'être identifiés. Les composants CMS peuvent être difficiles, voire impossibles à identifier, car ils portent souvent des numéros de référence abrégés. Même les ressources considérables de l'internet peuvent ne pas suffire.

Faites le tri

J'ai parlé des cartes que j'ai eues quand j'étais enfant, avec des fils de composants pliés. Je les avais tous dessoudés religieusement. Aujourd'hui, je les ignore, sauf s'ils sont vraiment spéciaux : ils n'en valent pas la peine.

Les condensateurs électrolytiques doivent toujours être testés, en particulier les gros modèles pour alimentations, et s'ils ont le dessus bombé, c'est un signe évident qu'ils sont secs ou qu'ils fuient.

Les anciens composants tels que les résistances au carbone n'ont aucun intérêt, et les anciens condensateurs électrolytiques sont souvent bien plus gros, à capacité égale, que les types modernes.

De nombreux CMS modernes ont des pattes très serrées ou sont de type BGA (*ball-grid array*), ce qui nécessite un équipement spécial pour les installer et les enlever sur les cartes. Si vous n'en avez pas l'utilité, jetez-les. Sinon, identifiez-les pour être sûr que ça vaut le coup de les dessouder.

Apprendre à identifier et à utiliser les composants de vieilles cartes peut valoir le temps passé. Vous pouvez récupérer des composants de haute qualité qui, si vous les stockez systématiquement, vous épargneront souvent d'avoir à en acheter pour un projet. Ces compétences peuvent aussi vous être utiles pour réparer des circuits imprimés.

210024-04

Contributeurs

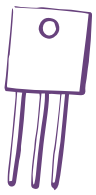
Idée, texte et illustrations : **David Ashton**
Rédaction : **Clemens Valens**
Traduction : **Helmut Müller**
Mise en page : **Harmen Heida**

Des questions, des commentaires ?

Envoyez un courriel à l'auteur (stn564@yahoo.com) ou contactez Elektor (redaction@elektor.fr)

LIEN

[1] Site de fiches techniques avec de nombreuses options : www.alldatasheet.com/

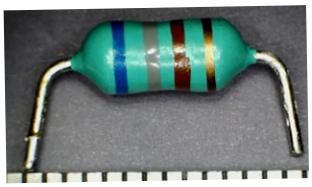


Faites le quiz d'identification des composants

FAITES LE QUIZ D'IDENTIFICATION DES COMPOSANTS

Les échelles sont en millimètres. Voyez comment vous vous en sortez !

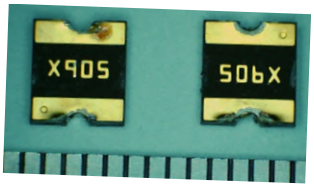
A - Pouvez-vous dire ce que c'est juste en le regardant ?



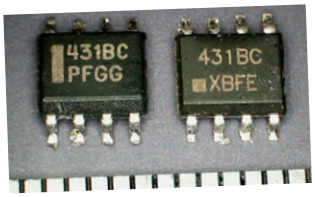
B - Et celui-ci ?

C - Qu'est-ce que c'est et quelle est sa valeur ?

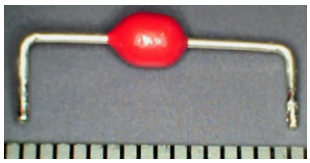
D - Quel est ce circuit intégré à l'aspect étrange ? Boîtier bizarre, mais bon numéro de composant (ou est-ce 431) ?



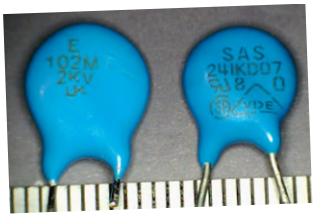
E - 506X ou X905 ? Ces CMS présentent une très faible résistance (quelques ohms). De quoi s'agit-il ?



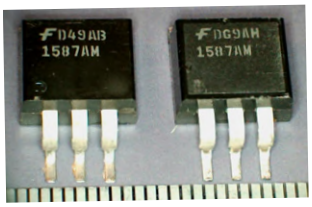
F - Que serait ce CI ?



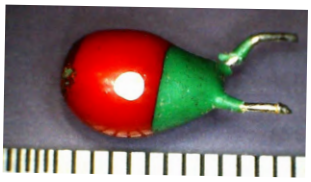
G - Celui-ci est bizarre. Le marquage 431 est à peine visible..



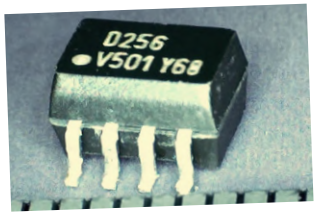
H - S'agit-il de composants du même type, ou pas ?



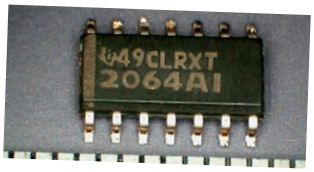
I - Et ceux-ci ? Pouvez-vous en trouver une fiche technique ?



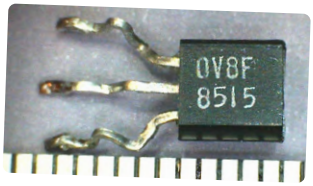
J - Est-ce un condensateur au tantale ? Ou autre chose ?



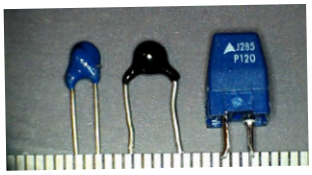
K – Marqué D256. Indice : l'épaisseur du circuit intégré.



L - Celui-ci est assez facile. Indice : qui est le fabricant ?

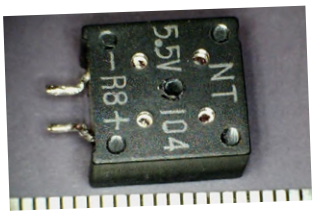


M - Souvenez-vous, celui-ci, je vous en ai parlé. Des idées ?



N - Ces composants se ressemblent, mais de quoi s'agit-il ? Des condensateurs au tantale ?

O - Pas de marquage. Mais à quoi cela sert-il ? Et les « poignées » : des radiateurs ?



P – Ce boîtier est percé de quatre trous (entre les inscriptions). Est-ce un buzzer ou un capteur de température ou d'humidité ?

Réponses au Quiz

A - C'est un condensateur céramique à tube – 22 nF. Pas tout à fait la forme d'une résistance et pas de la bonne couleur. Présenté ici avec – dans le sens des aiguilles d'une montre – 47 pF, 1 nF, et… une résistance de 486 kΩ 1%, qui, dans le lot, ressemble le moins à une résistance – beaucoup plus longue qu'une résistance normale ! Ça m'a trompé la première fois que je l'ai vue. Alors, méfiance, testez toujours !

B - Comme le condensateur en A était trop long pour une résistance, ceci est trop court et trop gros, et la couleur vert-bleu serait inhabituelle pour une résistance. C'est une inductance de 680 nH. La voici (en haut à gauche) avec (dans le sens des aiguilles d'une montre) 470 nH, 47 nH et 820 nH. Le 47 nH est de la couleur crème habituelle pour une résistance, mais un peu trop courte pour une résistance.

C - Il s'agit d'une très vieille résistance, utilisant le code couleur à point. Rouge-Noir-Jaune = 200 kΩ et pas de tolérance, donc probablement 20%. En fait, elle vaut 225 kΩ, un écart de 12,5% seulement ! Sans doute pas la peine de la conserver, sauf pour sa valeur historique : elle doit avoir plus de 80 ans !

D - Ceux-ci ont nécessité beaucoup de travail. Vous pouvez facilement trouver le LH1540 – un optocoupleur, dans différents boîtiers mais pas celui-ci. Il y a aussi 431 dessus – était-ce peut-être une version bizarre du TL431 ? Finalement, au bas d'une fiche technique de Vishay, j'ai vu une image de ce boîtier (appelé 8 PowerSOIC) sans autre information qu'un lien vers une fiche technique de dimensions. Cela m'a conduit à la fiche technique LH1540ACD qui fait référence à ce boîtier particulier. Parfois, il faut creuser.

E - Faible résistance, hein ? Suite à une intuition, j'ai branché sur mon alimentation et j'ai augmenté lentement le courant. À environ 700 mA, il s'est échauffé et passé à une résistance assez élevée. C'est un Polyfuse – un fusible à réarmement automatique très pratique. Les codes qui y figurent n'ont été d'aucune aide.

F - D'expérience, tout ce qui porte la mention 431 est généralement une référence de tension TL431 de Texas Instruments, ou d'une seconde source. Largement utilisée, elle peut aussi servir de capteur tension dans les alimentations. Il s'agit de la version CMS SOIC-8 ; le boîtier habituel est le TO-92.

G - Celui-ci m'a longtemps intrigué. Le chiffre 431 est à peine visible sur le fond rouge. Il s'agit d'une VDR (résistance dépendant de la tension) 430 V, une protection de surtension. Elle présente un circuit ouvert et une très faible capacité. Vu sa petite taille, je doute qu'elle puisse absorber un pic de courant élevé.

H - Ils sont presque identiques en taille et en couleur. À gauche, un condensateur, à droite, une VDR. Le condensateur a la valeur indiquée (102 = 1 nF), et supporte 2 kV. Le 241 sur la VDR signifie 240 V (24 et un zéro). Il est peu probable que vous voyiez une VDR de 1000 V, et il n'y aurait pas d'autre tension dessus ! Encore une fois, fiez-vous à l'expérience et ne faites jamais de suppositions !

*I - Le 1587AM est commun aux deux pièces. Cherchez « 1587 datasheet » et vous serez dirigé vers Alldatasheet.com. Sélectionnez les références se terminant par 1587 (*1587) et vous obtiendrez le régulateur de tension LDO LMS1587 de chez Texas. Si vous avez reconnu le logo Fairchild et que vous cherchez « Fairchild 1587 » sur Google, vous trouverez le Fairchild RC1587, qui est le même composant.*

J - C'est un condensateur au tantale de 20 µF, de l'époque où ils étaient codés par couleurs. En voici quelques autres : 2,2 µF, 3,3 µF et 5 µF. Seulement deux dans les valeurs standard, et des codes couleurs très bizarres ! Méfiez-vous de ces vieux « tantys » ils ont tendance à se mettre en court-circuit. Mais ils sont plus beaux que les nouveaux !

K - L'épaisseur du circuit intégré est un indice : on a généralement affaire à un optocoupleur. Un autre indice est le « V », qui indique souvent un produit de chez Vishay (voir D ci-dessus). Ça peut parfois vous faire gagner pas mal de temps. Son nom complet est ILD2561T et c'est un optocoupleur à deux canaux, à entrée CA, avec des rapports de transfert de courant (CTR) bien appariés.

L - Vous voyez le logo Texas Instruments juste avant le 49 sur la première ligne ? Allez sur le site web de TI, tapez 2064, et vous obtenez trois possibilités : une carte d'alimentation, un commutateur de distribution d'alimentation à huit broches et ceci – le TLE2064 – une version améliorée de l'ampli-op à quatre JFET d'entrée TL064. Comme d'habitude, pas de préfixe sur les composants CMS.

M - Voici mon transistor 0V8F non identifié – en boîtier TO-92 – dont il est question dans le texte. Dix points au premier lecteur qui pourra l'identifier avec le fabricant et la fiche technique. J'en ai plusieurs et le numéro de type est 0V8F, l'autre nombre est un code de date et varie. Oui, j'ai essayé OV8F !

N - Les petits bleu et noir sont des thermistances NTC. Lorsque la température augmente, la résistance diminue. Elles proviennent de capteurs de climatisation. Le grand est une thermistance PTC – plus utilisée pour la protection. Traversée par un courant excessif, elle chauffe et sa résistance augmente pour limiter le courant – comme les Polyfuses en E ci-dessus.

O - Je n'ai pas pu résister au plaisir d'inclure celui-ci. Vous pensez que c'est un capteur de courant ? Gagné ! L'image provient de la fiche technique de la sonde à effet Hall ACS756 d'Allegro. Le courant passe par les « poignées », et les broches sont le capteur à effet Hall. Isolement jusqu'à 3 kV, idéal pour les alimentations.

P - Il s'agit d'un supercondensateur : 104 = 100 000 µF = 0,1 F. Notez que la valeur est en µF, et non en pF comme sur d'autres condensateurs. 5,5 V est une tension courante pour ces condensateurs. Ils servent en général d'alimentation de secours pour la mémoire ou l'horloge en temps réel. Ici, on a affaire à l'un des modèles de 0,1 F rectangulaires les plus courants. Pourquoi les trous dans le boîtier ? Aucune idée !

interrupteur sans contact DIY

Mathias Claußen (Elektor)

La pandémie m'a donné l'idée de réaliser un interrupteur qui interprète des gestes de la main. L'absence de contact physique réduit en effet la transmission de virus et bactéries d'une personne à l'autre. C'est un atout pour les espaces de travail très fréquentés. Cet article décrit le principe de cette idée et sa mise en œuvre sur un *ESP32* (avec carte d'extension) qui évalue les gestes de la main.

Début 2020, lors de la 1ère vague de COVID-19, pour contenir la propagation, des mesures furent prises dans les bureaux. Nettoyage et désinfection des surfaces devinrent le rituel quotidien. S'il est facile d'essuyer machines à café et bouilloires avant et après utilisation, s'assurer que d'autres surfaces, par ex. les interrupteurs des bureaux paysagers et des couloirs, ne puissent devenir des sites de prolifération du virus est plus difficile.

En y réfléchissant, j'ai réalisé que le problème serait résolu à l'aide d'interrupteurs ne nécessitant aucun contact physique. Dans la

Avertissement

Tous les circuits présentés dans cet article fonctionnent sur le secteur. Ni les circuits ni les circuits imprimés présentés ici n'ont été testés du point de vue fonctionnalité et sécurité. Il est déconseillé de fabriquer ou utiliser les CI sans les faire vérifier au préalable. Pour réaliser un projet à partir de cet article, il est vital d'être conscient des dangers. Vous travaillerez entièrement à vos risques et périls. Seuls des ingénieurs ou des concepteurs qualifiés peuvent le faire !

boutique d'Elektor [1], j'ai trouvé le HAT *3D Gesture & Tracking* de *Seeed Studio* pour *Raspberry Pi* (**fig. 1**).

Figure 1. Le 3D Gesture & Tracking Shield (carte d'extension pour suivi et gestuelle 3D).

Le MGC3130

Début 2019, *James Rowley* et *Mark Omo* ont présenté un interrupteur d'éclairage sans contact *The Open Smart Switch* [2] sur la plateforme *hackster.io*. Ce projet utilise le *MGC3130* de *Microchip Technology*. Le HAT 3D Gesture & Tracking pour Raspberry Pi envisagé utilise la même puce. La chaîne *ElektorTV* [3] a abordé en détail les caractéristiques de cette puce.

Le MGC3130 capte le champ électrique. La puce envoie un signal basé sur la mesure du champ électrique produit à la surface d'une tablette tactile. Cinq électrodes disposées autour de la tablette servent à détecter le champ et tout changement subtil de ce dernier quand un objet comme une main s'approche de la tablette et le perturbe. La **figure 2** montre le principe de fonctionnement du capteur et le champ produit. Le doigt peut être reconnu et sa position calculée. En outre des gestes 3D, par ex. « roue à air » ou « pichenette », peuvent être évalués dans les quatre directions. Une pichenette consiste à passer devant le capteur avec un doigt en l'air. La puce peut être calibrée et configurée à l'aide des outils fournis par Microchip. La page [4] fournit toutes les informations nécessaires pour utiliser la puce dans une réalisation personnelle.

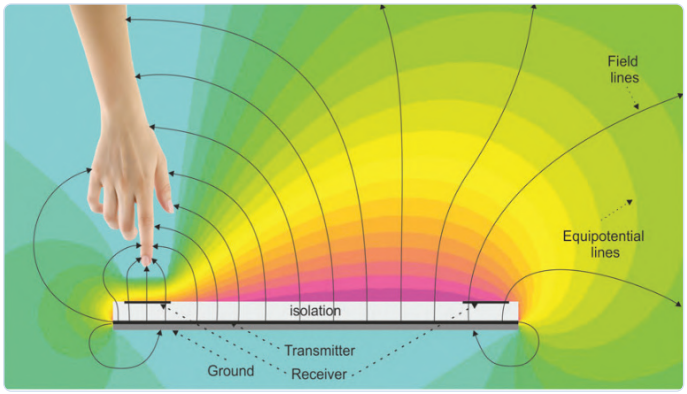


Figure 2. Le champ du capteur MGC3130. (Source : Microchip)

Elle communique avec le monde extérieur par une interface I²C et nécessite les deux lignes usuelles (données + horloge) plus deux autres (réinitialisation et interruption bidirectionnelle). Le document *DS40001718E* [5] de Microchip fournit les détails de l'interface. Pour cette puce, on peut concevoir un *pad* personnalisé, mais il faut noter que la carte devra comporter au moins quatre couches. Et dans ce cas, il sera nécessaire de paramétrer et configurer la puce pour correspondre au nouveau dessin du pad.

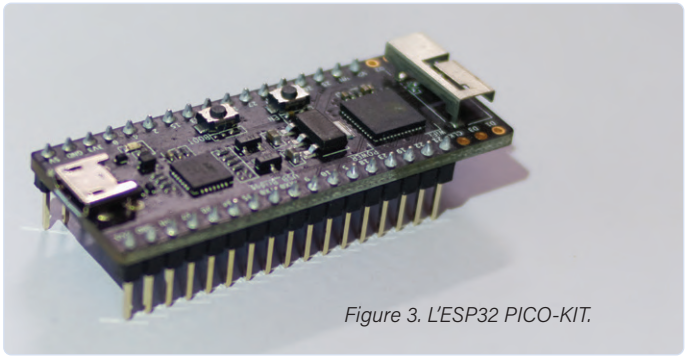


Figure 3. L'ESP32 PICO-KIT.

Quel microcontrôleur ?

En ce qui concerne le microcontrôleur, nous avons vraiment l'embarras du choix de nos jours. Pour n'en citer que quelques-uns, cela va du *Raspberry Pi Pico* à l'*ESP8266* ou à l'*ESP32*, en passant par le *STM32 Blue Pill* et l'éprouvé *ATmega328*. L'application ne nécessite pas de calculs compliqués et ne produit qu'un seul signal pour commuter le relais.

Il nous faut aussi réfléchir au mode de fonctionnement de l'interrupteur : doit-il être autonome pour allumer une seule lampe ou intégré à une installation domotique existante ? La 2e option, l'intégration dans un système de commande tel qu'*ESPHome*, aurait l'avantage de centraliser la commande de la lampe ou de la charge. Il convient également de déterminer comment effectuer les mises à jour du micrologiciel sur le système. Avec l'ESP8266 ou l'ESP32, une liaison Wi-Fi permet ces mises à jour sans fil (en anglais *OTA = Over The Air*).

Nous avons opté pour une variante de l'ESP32 qui dispose du Wi-Fi, du Bluetooth, d'une mémoire Flash embarquée suffisante et des mises à jour via Wi-Fi : l'*ESP32 PICO-KIT* (photo **fig. 3**).

Figure 4. Boîtier à encastrer. (Source : Würth [9])

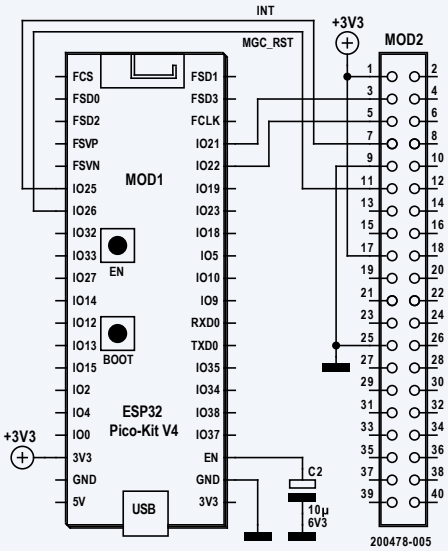


Figure 5. Schéma du prototype.

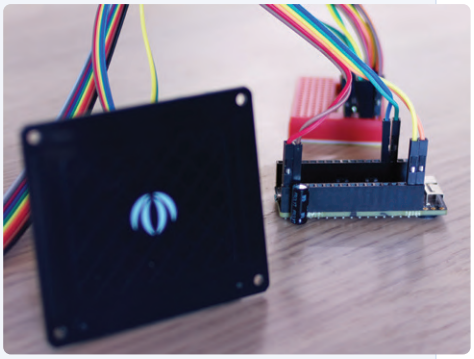


Figure 6. Interconnexion du tout.

Concept de base

Comme pour tout projet, il faut définir le concept de base et vérifier tous les composants potentiellement disponibles pour accomplir la tâche. Ici, nous possédons déjà un capteur, le 3D Gesture & Tracking Shield pour le Raspberry Pi et un ESP32, en version PICO-KIT. Il nous faut aussi un relais pour allumer et éteindre la lumière et enfin une alimentation électrique. Il suffit d'un micrologiciel et le tour sera joué…

Mais, tout n'est pas si simple. Souvent, en cours de développement, nous nous heurtons à des problèmes imprévus qui doivent être résolus. Le processus de développement peut être divisé en trois domaines qui concourent à élaborer le produit fini. Il y a d'abord le logiciel qui évalue les données des capteurs et déclenche les actions adéquates. Ensuite le matériel électronique et la carte sur laquelle les composants sont montés. Un schéma de connexion des composants est produit. Idéalement, le développeur du logiciel a déjà demandé un retour d'information sur l'affectation des broches et les exigences particulières. Il arrive souvent que des erreurs de disposition du matériel puissent être corrigées par de petites modifications du logiciel, mais cela peut être une source d'ennui pour le développeur du logiciel.

Enfin, il y a la conception mécanique de l'ensemble complet et du boîtier qui doit tout accueillir. Souvent, la taille du boîtier n'est pas critique et nous pouvons en choisir (ou fabriquer) un de dimension adéquate. Cependant, dans certains cas, nous aurons besoin de tout faire entrer dans un boîtier standard, par ex. une boîte d'encastrement électrique ordinaire, pour l'intégrer dans l'installation électrique domestique. Pour ce projet, notre plan initial était de faire tenir l'ensemble dans un volume de 65 mm × 55 mm × 25 mm. Toutefois, cela ne correspond pas à certaines boîtes d'encastrement courantes (**fig. 4**) utilisées dans certains pays européens. Pour conférer un caractère plus universel à cette réalisation, il serait souhaitable que l'ensemble soit assez petit pour être monté dans ce type de boîtier également.

```
26      mgx3130_rst_pin =resetPin;
27      xTaskCreate( Touchinput_Task, "MGC3130Drv", 4096, NULL, tskIDLE_PRIORITY, &x
28      configASSERT( xHandle );
29  }

PROBLEMS  OUTPUT  TERMINAL  DEBUG CONSOLE

x: 48540, y: 65534, z: 34456
x: 47465, y: 65534, z: 34252
x: 46156, y: 65534, z: 33950
x: 44656, y: 65534, z: 33498
x: 43026, y: 65534, z: 32867
x: 41337, y: 65534, z: 32204
x: 39667, y: 65534, z: 31760
x: 38103, y: 65534, z: 31560
x: 36732, y: 65534, z: 31502
x: 35621, y: 65534, z: 31525
x: 34798, y: 65534, z: 31599
x: 34243, y: 65534, z: 31708
x: 33899, y: 65534, z: 31841
x: 33696, y: 65534, z: 31986
x: 33569, y: 65534, z: 32119
x: 33467, y: 65534, z: 32215
x: 33360, y: 65534, z: 32265
x: 33237, y: 65534, z: 32278
x: 33103, y: 65534, z: 32277
x: 32972, y: 65534, z: 32275
x: 32857, y: 65534, z: 32284
x: 32764, y: 65534, z: 32304
Gesture: FLICK_SOUTH_TO_NORTH, class: FLICK_GESTURE, edge flick: no, in progress: no
x: 32687, y: 65534, z: 32331
x: 32611, y: 65534, z: 32358
```

Figure 7. Sortie de données du capteur de gestes.

Mise en œuvre du concept initial

Le schéma de la **figure 5** montre la connexion initiale du 3D Gesture & Tracking Shield pour Raspberry Pi à l'ESP32. La carte est installée sur une plaque d'essai (**fig. 6**) pour tester sa fonctionnalité. Plusieurs bibliothèques sont disponibles pour le MGC3130. La bibliothèque utilisée dans ce projet provient de Seeed Studio [6], elle est prévue pour un Raspberry Pi.

Pour l'ESP32, l'interface I²C et l'affectation des broches d'E/S doivent être adaptées. Une fois ces modifications effectuées, les tests peuvent démarrer avec cette bibliothèque. La **figure 7** montre la suite des coordonnées positionnelles du mouvement pour un « geste de pichenette » reconnu.

Alimentation et relais

Lors de la conception et du test du prototype, nous ne nous sommes pas préoccupés de l'alimentation électrique ni du relais à inclure dans la réalisation finale. Le contrôleur final fonctionnera à partir de la tension du secteur. À cet effet, nous devrons inclure une alimentation de 5 V ou 3,3 V suffisamment puissante pour l'ESP32, le 3D Gesture & Tracking Shield et le relais.

Les *HLK-PM03* et *HLK-PM01* de *Hi-Link* (**fig. 8**) sont parmi les modules d'alimentation les plus compacts. Ils fournissent une

Figure 8. Convertisseur CA/CC Hi-Link.

puissance de 3 W, soit 600 mA en 5 V ou environ 900 mA en 3,3 V. L'ESP32 nécessite environ 500 mA (2,5 W sous 5 V, 1,65 W sous 3,3 V). Nous avons choisi un relais *Panasonic* de type *ADW12*. Il nécessite une impulsion d'environ 67 mA en 3,3 V ou 40 mA pour la version 5 V (220 mW). Le 3D Gesture & Tracking Shield lui-même nécessite encore 20 mA en 3,3 V (66 mW). Tout compris, nous arrivons à un minimum de 2,8 W si l'ESP32 PICO-KIT est alimenté en 5 V. Cela nous rapproche de la puissance de sortie maximale de ce module 5 V.

L'ESP32 PICO-KIT consomme moins s'il est alimenté en 3,3 V, car le régulateur *LDO* (*Low DropOut*) intégré n'est plus nécessaire et ne sera pas alimenté. C'est pourquoi nous utiliserons finalement un module 3,3 V tel que le HLK-PM03. Il peut fournir 900 mA maximum. Le relais choisi est un Panasonic *ADW1203HLW*. Les contacts de ce relais unipolaire bistable acceptent 16 A sous 277 VAC max.. Il est assez compact : 24 mm × 10 mm × 16 mm. Il suffit d'une courte impulsion pour l'activer ou le désactiver et la bobine ne consomme aucun courant permanent.

Vers une version finale du schéma

Le schéma préliminaire du circuit est présenté à la **figure 9**. Nous n'avons pas encore inclus de protection autour du module d'alimentation HLK-PM03. Habituellement on installe, côté entrée CA, un fusible thermique et une varistance et, côté sortie, un fusible et des condensateurs de lissage de la tension. Sans ces mesures de protection du côté de l'entrée, une défaillance interne du module peut conduire à un incendie. En sortie, une régulation et un filtrage insuffisants de la tension peuvent réduire considérablement la portée Wi-Fi ou déclencher un crash logiciel. L'alimentation finale comprendra ces modifications.

Un condensateur de 10 µF placé entre les broches GND et EN de l'ESP32 évite tout défaut de synchronisation qui empêcherait de passer en mode de programmation à la réinitialisation. C'est également utile si, à l'occasion, l'ESP32 refuse d'être programmé depuis le port micro-USB.

Comme déjà indiqué ci-dessus, la carte ESP32 PICO-KIT comprend un régulateur AM1117-3.3 censé fournir 3,3 V à l'ESP32 à partir de 5 V. Ce régulateur supporte 3,3 V sur sa broche de sortie s'il n'est pas alimenté. Cependant, dans notre schéma du circuit, une diode Schottky (D1) est connectée entre l'alimentation 3,3 V et l'alimentation 5 V afin qu'un régulateur, monté le cas échéant sur un autre type de carte, ne puisse pas être endommagé. En l'incluant, il circule un courant de l'alimentation 3,3 V à travers le régulateur. Le courant minimal de commande du relais 3,3 V est de 67 mA. C'est beaucoup plus que ce que les broches de l'ESP32 peuvent fournir. Un étage de commande à transistors (T1 et T2) est donc utilisé. Les diodes de roue libre D2 et D3 sont essentielles pour éviter de détruire les transistors par les pics de tension inverse à la coupure de la bobine du relais.

Le 3D Gesture & Tracking Shield est connecté à l'ESP32 via le

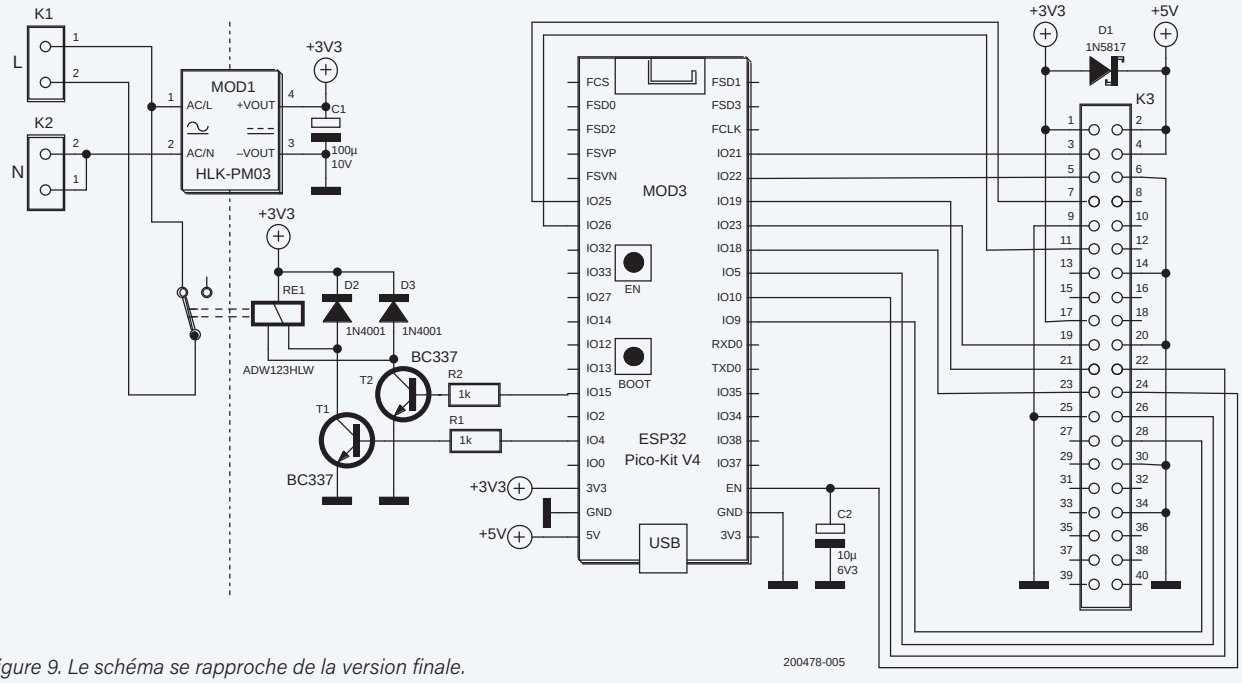


Figure 9. Le schéma se rapproche de la version finale.

Figure 10. Blocs de base du microprogramme.

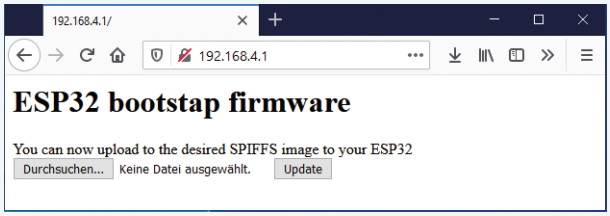


Figure 11. Téléchargement de l'image SPIFF via un navigateur web.

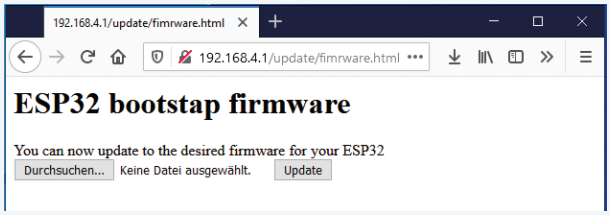


Figure 12. Mise à jour du micrologiciel OTA via un navigateur web.

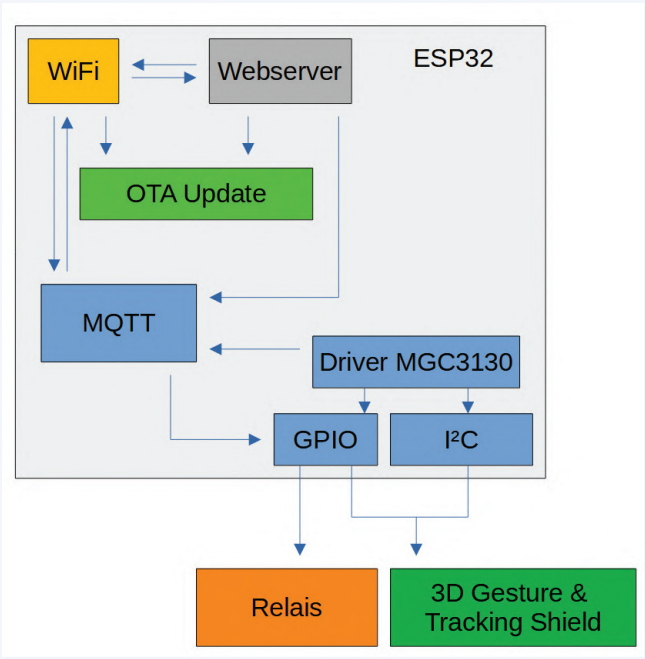


Figure 13. Liaisons du module de micrologiciel.

connecteur à deux rangées de broches K3. Les 3,3 V, les signaux I²C et les deux broches d'E/S sont les seules connexions requises pour le HAT. Il comprend déjà les résistances de rappel au positif du bus I²C. La carte a été conçue pour fournir des signaux SPI et autres broches d'E/S nécessaires pour l'ajout d'un moniteur TFT de 3,5 pouces.

La flexibilité de la matrice d'E/S de l'ESP32 facilite la connexion des périphériques. Pour le 3D Gesture & Tracking Shield, les broches 21 et 22 sont affectées au bus I²C, la broche 25 à l'interruption et la broche 26 à la réinitialisation du MGC3130. S'il n'est pas nécessaire d'intégrer un écran TFT, nous pourrions utiliser une variante ESP32 nettement plus petite ou même un module ESP32-C3.

Micrologiciel

Après avoir affecté les broches de l'ESP32, il faut compiler le micrologiciel pour l'utiliser. Les routines de la bibliothèque permettent de créer un serveur web qui produit deux pages web de base et d'effectuer des mises à jour sans fil du micrologiciel. La **figure 10** donne les modules du microprogramme de base. Cette approche évite de réécrire et de déboguer certaines des fonctions de base utilisées dans de nombreuses applications similaires.

Le microprogramme produit deux pages web de base. L'une sert à écrire une nouvelle image *SPIFFS* (*Serial Peripheral Interface Flash File System*) dans l'ESP32 (**fig. 11**) et l'autre à fournir une mise à jour du micrologiciel de l'ESP32 (**fig. 12**).

Aujourd'hui, le micrologiciel n'exécute que les fonctions de commutation de base. Un autre composant permet d'évaluer les informations du MGC3130. Une page web minimale permet de configurer la messagerie *MQTT* (*Message Queuing Telemetry Transport*) et une autre d'activer/désactiver la charge via un navigateur web. La **figure 13** illustre l'interaction des modules. La commande de l'interrupteur d'éclairage est donc logicielle ou manuelle.

Casse-tête…

Le micrologiciel et le schéma du circuit étant prêts, il faut s'assurer que tout le matériel tient dans l'espace disponible. En matière d'outils de conception de circuit imprimé (CI), chacun a ses préférences et utilise celui qui lui est familier. De mon côté, je préfère KiCad qui crée des schémas et des mises en page éditables librement sans grand investissement financier. Elektor a d'ailleurs publié le livre *KiCad Like a Pro* pour KiCad [7].

La **figure 14** illustre le CI réalisé et, en effet, il n'est pas très joli. Des découpes dans la carte assurent l'isolement nécessaire entre les zones sous tension du secteur et l'alimentation 3,3 V. Tout cuivre superflu à l'interconnexion des composants est retiré. La **figure 15** montre le routage obtenu. Les découpes et les fentes sur le pourtour de la carte sont réalisées à la fraiseuse, on ne peut donc pas s'attendre à des angles vifs à 90°. Si vous souhaitez faire fabriquer la carte, vérifiez les caractéristiques du fraisage pour avoir des angles internes de rayon minimal.

Des modèles 3D existent pour presque tous les composants. Ils serviront à étudier l'implantation des composants et à utiliser

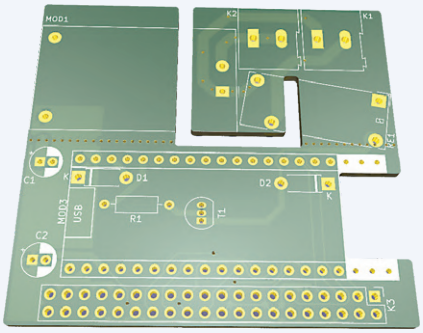


Figure 14. Circuit imprimé terminé.

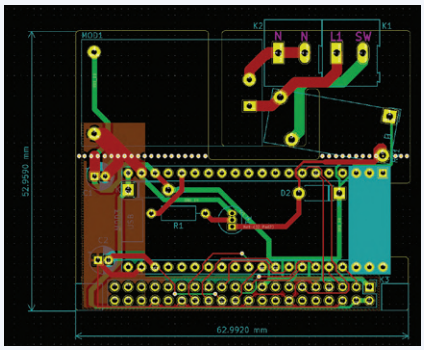


Figure 15. Schéma d'implantation terminé.

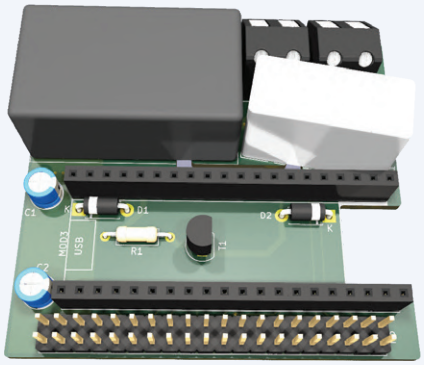


Figure 16. Disposition des composants sur le circuit imprimé.

au mieux l'espace disponible. Pour tout faire tenir, vous serez peut-être amené à utiliser des cartes filles, perpendiculaires à la carte principale. La **figure 16** montre une vue 3D de notre CI.

Chausse-pieds SVP !

KiCad a l'avantage de pouvoir exporter un fichier STEP utilisable dans un programme de conception 3D. Dans FreeCAD, nous chargeons et plaçons correctement le modèle de notre CI et son boîtier (impression 3D de la **figure 17**). Sur la **figure 18**, on voit que le CI ne tient pas ; son contour déborde de l'espace disponible. Un conflit apparaît entre les composants et les surfaces internes du boîtier et le bord de la carte (en vert) dépasse à l'avant. Est-ce une erreur sur les modèles 3D, un problème de mise à l'échelle, ou bien une simple erreur de mesure ? La dernière hypothèse est la bonne.

Après avoir revu les dimensions de la carte, tout semble un peu à l'étroit sur le CI. La vue de côté des composants de la **figure 19** montre qu'il n'y a pas assez de place pour tous ceux-ci. Il faut garder une distance sûre entre pistes, il n'est donc pas possible de rapprocher les composants. La **figure 20** montre qu'une fois tout assemblé sous FreeCAD, le contour du module d'alimentation déborde encore sur un pilier d'angle. Il nous faut revenir en arrière et revoir la conception avec d'autres composants.

Et maintenant ?

Du point de vue matériel et micrologiciel, la conception de base de l'interrupteur sans contact fonctionne comme prévu. Pour que l'unité puisse être utilisée avec une autre gamme de normes électriques, la carte doit aussi pouvoir tenir dans un boîtier circulaire. La figure 4 montre une épure de boîtier pour interrupteurs et prises à encastrer dans des cloisons de plaques de plâtre ; ce modèle est courant dans certains pays européens. Pour résoudre cet écueil, il faudra étudier la compacité du module d'alimentation et la taille des modules ESP32 du marché pour

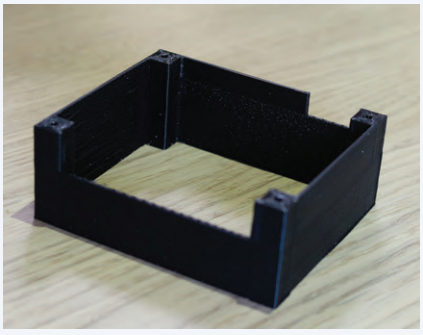


Figure 17. Le boîtier imprimé en 3D.

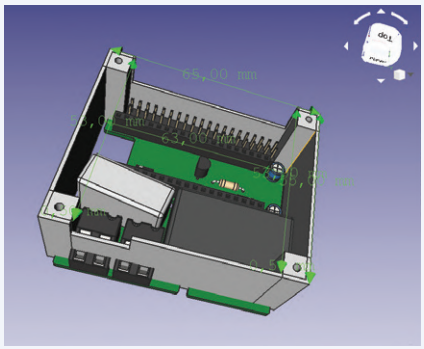


Figure 18. Le circuit imprimé ne rentre pas.

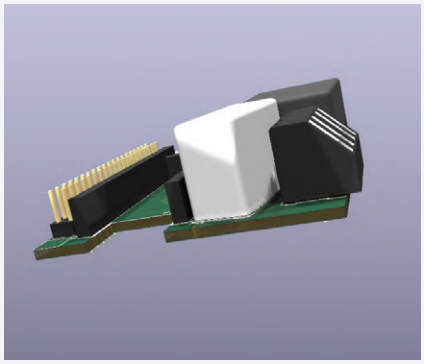


Figure 19. Vue de côté montrant les composants placés les uns contre les autres.

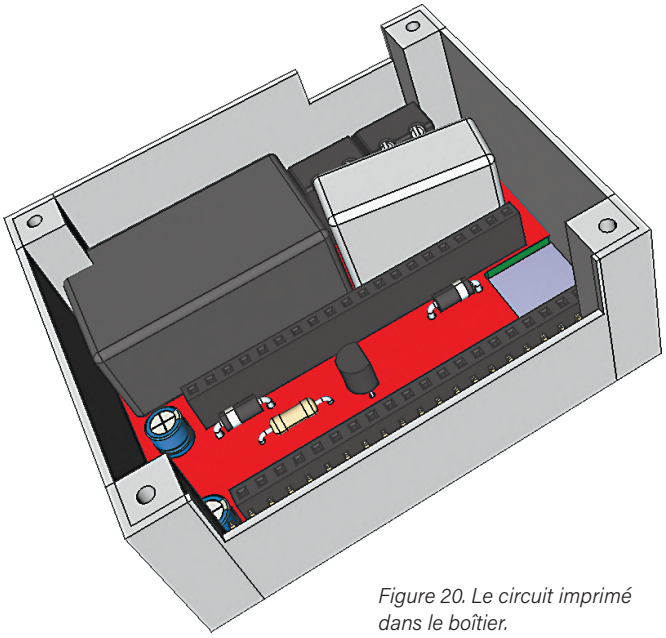


Figure 20. Le circuit imprimé dans le boîtier.

prendre moins de place que le PICO-KIT ESP32.
Avec un peu de chance, des modules ESP32-C3 devraient arriver sous peu sur la paillasse du laboratoire. Plus compacts, ces modules sont moins chers, moins énergivores et équipés de Wi-Fi & Bluetooth. Vu leur faible consommation, un module d'alimentation plus petit devrait suffire et faire gagner un peu plus de place.
À ce stade de développement, nous avons décidé de mettre ce projet en veille. L'expertise de nos lecteurs est la bienvenue ainsi que toute suggestion sur des composants alternatifs plus compacts ou l'optimisation de la disposition et du routage pour remplir l'espace disponible. Ce projet n'est pas achevé et nous pensons que les lecteurs trouveront bien des erreurs dans le circuit actuel. Des conseils et des astuces sur l'interopérabilité KiCad/FreeCAD seraient également les bienvenus. Si vous souhaitez jeter un coup d'œil au code source, au schéma, au circuit imprimé ou aux données du boîtier, tout est téléchargeable depuis le dépôt GitHub d'Elektor [8].

200478-04

Contributeurs

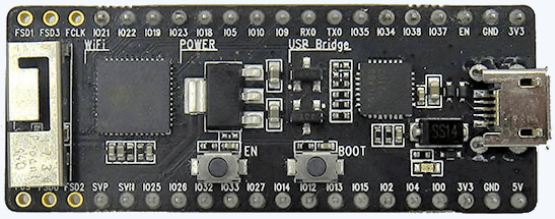
Texte et images : **Mathias Claußen**
Rédaction : **Jens Nickel, C. J. Abate**
Traduction : **Yves Georges**
Mise en page : **Giel Dols**

Des questions, des commentaires ?

Envoyez un courriel à l'auteur (mathias.claussen@elektor.com) ou contactez Elektor (redaction@elektor.fr).

PRODUITS

- **ESP32-PICO-Kit V4**
  www.elektor.fr/18423



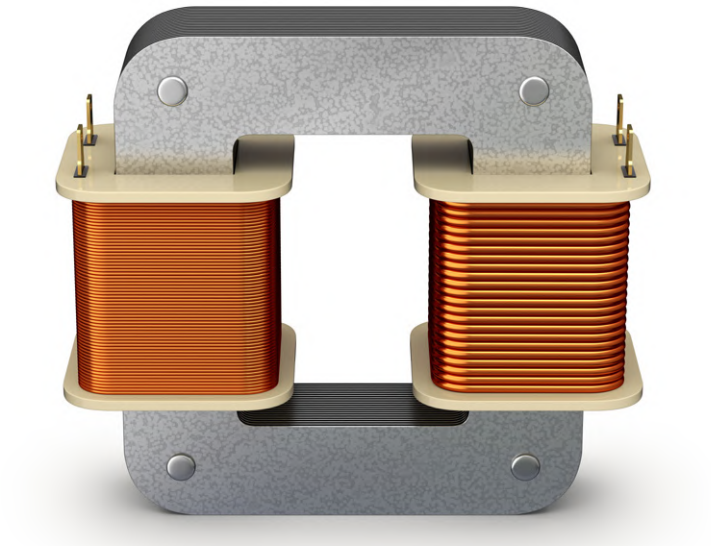
- **Modèle 3445 de PeakTech , multimètre numérique, mesure de la valeur efficace vraie, avec Bluetooth (6000 points)**
  www.elektor.fr/18774
- **WT 1013, station de soudage numérique (95 W) de Weller**
  www.elektor.fr/19338

LIENS

[1] Boutique d'Elektor : www.elektor.fr
[2] Interrupteur intelligent « open source » : www.hackster.io/133854/open-smart-switch-44fa54#toc-future-developments-8
[3] Vidéo Youtube de démonstration du Gesture Pad : www.youtube.com/watch?v=WVSVhEeMi_4
[4] Page produit du MGC3130 : www.microchip.com/wwwproducts/en/MGC3130
[5] Fiche technique du MGC3130 : https://ww1.microchip.com/downloads/en/DeviceDoc/40001718E.pdf
[6] Bibliothèque Seeed Studio MGC3130 : https://github.com/Seeed-Studio/Seeed_Linux_mgc3x30
[7] Peter Dalmaris, « KiCAD like a Pro », Elektor : www.elektor.fr/kicad-like-a-pro
[8] Dépôt Github : https://github.com/ElektorLabs/200478-Touchless-Lightswitch
[9] Boîte d'encastrement pour mur creux de Würth : https://eshop.wurth.fr/Categories-produits/Boite-d-encastrement-pour-mur-creux/310755020304.cyid/3107.cgid/fr/FR/EUR/

démarrer en électronique… (12)

Adaptation d'impédance et transformateurs



Eric Bogers (Elektor)

Avec cet épisode, nous arrivons au bout du sujet des « bobines ». Nous jetterons un dernier coup d'œil aux filtres de bande et aborderons les aspects les plus importants des transformateurs.

Adaptation d'impédance

Revenons brièvement au filtre passe-bande utilisé dans la conclusion de l'article précédent [1] (**fig. 1**). Vous remarquerez que, comme charge, nous avons remplacé le haut-parleur par une résistance ; en effet, ce n'est qu'avec une résistance de terminaison de valeur constante que le comportement du filtre est conforme à la théorie – or l'impédance d'un haut-parleur est tout sauf constante. Deux facteurs viennent compliquer les choses : premièrement, l'impédance atteint un pic à la fréquence de résonance et, deuxièmement, l'impédance augmente avec la fréquence (en raison de l'auto-inductance de la bobine mobile).

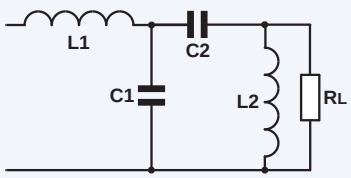


Figure 1. Filtre passe-bande.

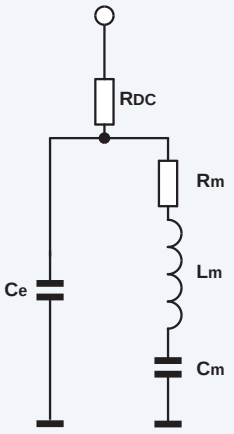


Figure 2. Compensation de l'impédance.

Lorsque ces perturbations se produisent à plus de deux octaves de la fréquence de coupure du filtre (ce qui correspond à la moitié ou au double de cette fréquence), on peut les négliger. Dans le cas contraire, les hypothèses initiales faites lors du calcul du filtre ne sont pas valables et le filtre ne fonctionnera pas comme prévu. Heureusement, il est possible de compenser ce problème. La **figure 2** montre un réseau de compensation réalisable.

Ce réseau se compose de deux parties : à gauche, le condensateur C_e, qui doit compenser l'auto-inductance de la bobine mobile, et à droite, un circuit résonant en série qui compense le pic de résonance. Lorsqu'un facteur de perturbation est suffisamment éloigné de la fréquence de coupure, il n'est pas nécessaire de le compenser et la partie correspondante du réseau peut être omise. Les valeurs des composants sont déterminées comme suit :

R_{DC} = résistance en courant continu du haut-parleur

$$C_e = \frac{1000 \cdot L_e}{R_{DC}^2} \qquad [\mu\text{F, mH}]$$

$$R_m = \frac{R_{DC} \cdot Q_e}{Q_m} - R_{loss}$$

$$L_m = \frac{159 \cdot R_{DC} \cdot Q_e}{f_s} \qquad [\text{mH}]$$

$$C_m = \frac{159000}{f_s \cdot R_{DC} \cdot Q_e} \qquad [\mu\text{F}]$$

Les valeurs des paramètres R_{DC}, Q_m, Q_e et f_s peuvent être évaluées à l'aide de logiciels de mesure spéciaux tels que Kirchner ATB PC Pro. R_{DC} est la résistance en courant continu du haut-parleur et f_s la fréquence centrale du pic de résonance. R_{loss} est la résistance équivalente de la bobine et L_e son auto-inductance.

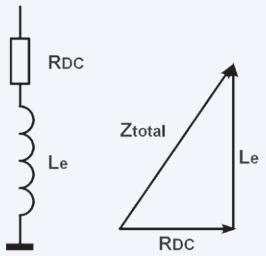


Figure 3. Auto-inductance et résistance en courant continu.

Mais comment déterminer L_e ? Aux fréquences élevées, le haut-parleur peut être considéré comme un circuit en série composé d'une auto-inductance et d'une résistance en courant continu, comme représenté sur la **figure 3**. À une fréquence suffisamment élevée, on peut lire la valeur de l'impédance totale Z_{total} sur le diagramme d'impédance. Pour l'auto-inductance L_e, on applique la formule :

$$L_e = \frac{\sqrt{Z_{tot}^2 - R_{DC}^2}}{2 \cdot \pi \cdot f}$$

Dans l'exemple de la figure 3, l'impédance totale à 10 kHz est de 42 Ω, tandis que R_{DC} est égale à 7 Ω. Le calcul indique qu'une auto-inductance de 0,66 mH doit être compensée par un condensateur d'environ 10 µF.

Transformateurs

Les transformateurs peuvent être utilisés pour élever ou abaisser une tension alternative. Lorsque c'est la tension du secteur qui est abaissée, on parle habituellement d'un transformateur d'alimentation ; lorsqu'il s'agit du signal, on parle (vous l'avez déjà deviné) d'un transformateur de signal. Le fonctionnement de ces deux « espèces » de transformateurs est cependant exactement le même.

Les transformateurs de signal ne sont habituellement pas utilisés pour élever ou abaisser la tension des signaux, mais plutôt pour séparer galvaniquement les (sous-)circuits les uns des autres, par exemple pour empêcher la création d'une boucle de masse. Dans ce cas, on parle d'un couplage par « transformateur symétriseur ».

La **figure 4** montre quelques représentants de l'espèce « transformateur ». À droite, on peut voir une partie d'un transformateur toroïdal : les enroulements sont bobinés autour d'un noyau de fer en forme d'anneau. Ce type de transformateur se distingue par un flux magnétique de fuite très faible, raison pour laquelle il est populaire dans les amplificateurs de puissance audio.

En haut à gauche, vous pouvez voir un transformateur qui peut être soudé directement sur un circuit imprimé. Pour l'isolement et une bonne tenue mécanique, ce transformateur est moulé dans un bloc en plastique.

Enfin, en bas à gauche, nous avons deux transformateurs de signal audio pour circuits imprimés. Les transformateurs audio sont souvent entourés d'un blindage en mu-métal pour éviter qu'ils ne captent des champs magnétiques parasites. Ce n'est toutefois pas le cas dans les exemples de la figure 4.

La **figure 5** présente le symbole schématique d'un transformateur. Si cela vous fait penser à deux bobines, vous avez raison : un transformateur est constitué de deux bobines (ou plus) enroulées sur un noyau commun. En principe, ce noyau pourrait être une pièce en fer massif, mais cela entraînerait des pertes inacceptables par courants de Foucault. C'est pourquoi, dans les transformateurs 50 Hz (c'est-à-dire les transformateurs d'alimentation), le noyau est constitué d'un empilement de fines tôles d'acier au silicium, isolées les unes des autres. Pour les fréquences plus élevées, de la poudre de fer est mélangée à un liant isolant et pressée dans la forme désirée. C'est ainsi que l'on obtient des noyaux de ferrite. À très haute fréquence, un noyau n'est plus nécessaire et vous pouvez utiliser deux bobines à air.

Un transformateur fonctionne de la manière suivante : une tension alternative est appliquée à l'enroulement primaire qui est alors parcouru par un courant alternatif produisant un champ magnétique variable qui induit une tension alternative dans l'enroulement secondaire. Il n'y a pas de liaison conductrice entre les enroulements primaire et secondaire : ils sont séparés galvaniquement.

Les désignations « primaire » et « secondaire » peuvent donner l'impression qu'un transformateur ne fonctionne que dans un seul sens, mais c'est faux : il est tout à fait possible d'appliquer une tension à l'enroulement secondaire et d'utiliser ensuite la tension disponible sur l'enroulement primaire. Toutefois, il faut veiller à ce que le noyau de fer ne soit pas amené à saturation magnétique par une tension trop élevée. Si, par exemple, un transformateur de 12 V (c'est-à-dire qui convertit la tension du secteur de 230 V en 12 V) est branché à l'envers, il y aura à coup sûr un fusible grillé (ou un enroulement de transformateur brûlé). En raison de la saturation magnétique du noyau, un courant beaucoup trop élevé circule. Toutefois il est possible d'appliquer une tension de 12 V sur l'enroulement secondaire afin de disposer de 230 V sur l'enroulement primaire. Les onduleurs fonctionnent selon ce principe.

Le rapport entre le nombre de spires au primaire et le nombre de spires au secondaire est appelé le rapport de transformation. La formule suivante s'applique :

$$T = \frac{W_1}{W_2} = \frac{U_1}{U_2} = \frac{I_2}{I_1}$$

Le rapport entre les tensions d'entrée et de sortie est égal au rapport du nombre de spires, le rapport entre les courants d'entrée et de sortie est exactement l'inverse. Cela découle de l'égalité théorique des puissances apparentes au primaire et au secondaire :

$P_1 = P_2$

Un transformateur n'étant pas un composant idéal, il y a toujours des pertes, et ces puissances présentent une différence, mais en général assez faible pour qu'on puisse la négliger dans la plupart des calculs.
Pour les transformateurs d'alimentation, plutôt que le rapport de transformation, on indique la tension disponible au secondaire pour une tension du secteur de 230 V.
De nombreux transformateurs d'alimentation ont plusieurs enroulements secondaires, fournissant parfois des tensions différentes. Il est possible de connecter des enroulements secondaires en série, mais il faut tenir compte de leur phase et de leur courant maximal. Pour connecter plusieurs enroulements en parallèle, ils doivent fournir la même tension et avoir une capacité de puissance (presque) identique. De plus, ils doivent être connectés en phase.
En outre, les transformateurs d'alimentation ont souvent plusieurs enroulements primaires pour pouvoir adapter les appareils qui les utilisent aux différentes tensions du secteur dans les différents pays. Il n'est cependant jamais possible d'appliquer une tension de 400 V à un transformateur d'alimentation de 230 V, car, comme on l'a vu, le transformateur entrera en saturation magnétique, ce qui fera sauter un fusible ou détruira son enroulement primaire. C'est pourquoi, sur les sites alimentés en triphasé, on craint les erreurs de câblage des prises délivrant alors du 400 V au lieu du 230 V attendu. L'inverse ne présente pas de danger : un appareil équipé d'un transformateur 400 V et branché sur le 230 V ne fonctionnera sans doute pas parce que sous-alimenté, mais ne risquera pas de flamber.
Voilà qui conclut le sujet des « bobines ». La prochaine fois, nous commencerons (enfin !) à nous intéresser aux semi-conducteurs.

210626-04

La série d'articles « démarrer en électronique » est basée sur le livre « Basic Electronics Course » de Michael Ebner, publié par Elektor.

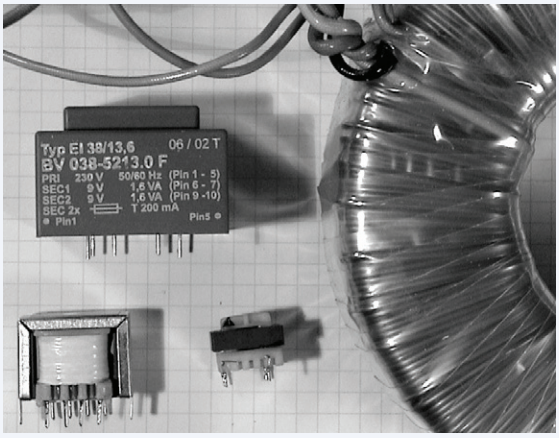


Figure 4. Divers transformateurs d'alimentation et de signaux.

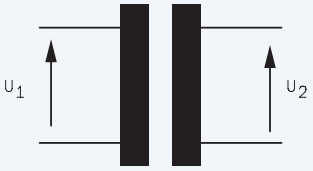


Figure 5. Symbole schématique d'un transformateur.

Contributeurs

Idée et illustrations : **Michel Ebner**
Texte et rédaction : **Eric Bogers**
Traduction : **Helmut Müller**
Mise en page : **Giel Dols**

Des questions, des commentaires ?

Contactez Elektor (redaction@elektor.fr).

PRODUITS

- **B. Kainka, *Initiation à l'électronique et programmation de montages pour débutants***
  www.elektor.fr/19339
- **R. Mallard, *L'électronique pour les débutants***
  www.elektor.fr/15662

LIEN

[1] « Démarrer en électronique… », Elektor, 01-02/2022 : www.elektormagazine.fr/210564-04

quoi de neuf dans le développement de l'embarqué ?

Rust et mises à jour des déploiements IoT

Embedded Development

Stuart Cording (Elektor)

Alors que les annonces de commercialisation de systèmes embarqués se succèdent, donnant l'impression que la technologie avance rapidement, le secteur lui-même paraît lent. C'est pourquoi ce fut un choc quand la fondation Raspberry Pi, célèbre créateur d'ordinateurs monocartes, sortit le microcontrôleur (MCU) RP2040 doté de deux cœurs Cortex-M0+, sans flash intégrée : un double cœur dans cette catégorie, c'était du jamais vu. Mais l'excitation est vite retombée. Tout compte fait, les progrès des systèmes embarqués sont mesurés, sensés et réfléchis. Mais, nous le verrons, les évolutions en cours pourraient changer le développement de l'embarqué dans la décennie à venir.

Le développement de logiciels embarqués sans C est difficile à imaginer. Le langage C a supplanté l'assembleur devenu trop lourd pour développer des applications entières, sauf si la seule option est l'assembleur hautement optimisé et codé à la main. Le langage, développé par Dennis Ritchie [1] des Bell Labs, offre assez de souplesse pour développer des applications complexes, tout en permettant un accès aisé aux registres. C'est essentiel pour écrire un code de MCU compact qui gère cet accès pour les routines d'interruption. L'écriture de tâches telles que la manipulation des bits des registres est aussi aisée. Et, contrairement à l'assembleur, le code résultant est facile à lire. Le C occupe aussi une place de choix en se classant parmi les trois premiers langages de programmation dans les enquêtes et les analyses de marché (**fig. 1**) [2] [3].

Si c'est du C, c'est daté !

Le langage C remonte à 1972, soit 50 ans. Ses limitations bien connues sont, pour beaucoup, liées à l'utilisation de pointeurs. Si ces pointeurs facilitent l'accès aux registres aux développeurs de systèmes embarqués, ils peuvent aussi causer de dangereux accès mémoire hors limites. En outre, loin des langages plus modernes, les compilateurs C effectuent peu de vérifications de code. Ils ignorent les variables inutilisées, elles sont pourtant un indice d'erreur de codage.

Une norme de codage telle que MISRA C [4] permet de s'assurer que seul un code C sécurisé est intégré dans un système embarqué. Cette norme est née de l'importance croissante du C pour programmer les systèmes embarqués dans l'industrie automobile. Le C++ résout certains problèmes des pointeurs du C par des *références* [5] non modifiables pour référencer un autre objet, elles ne peuvent pas être *NULL* et doivent être initialisées à la création. Malgré cela, AUTOSAR, un partenariat de développeurs de systèmes automobiles, a élaboré un guide de directives [6] de plusieurs centaines de pages sur l'utilisation du C++ dans les applications liées à la sécurité. Ainsi, bien que la maîtrise de ces langages établis soit essentielle pour les développeurs, il apparaît que chaque langage a assez de défauts pour qu'il faille écrire des directives pour éviter les erreurs courantes.

Présentation de Rust

Se proclamant *très adapté* au développement de systèmes sûrs, Rust apparaît comme un concurrent potentiel. Au début (2006), Rust est un projet privé de Graydon Hoare, puis vers 2010, son employeur, Mozilla Research, le sponsorise. La Rust Foundation [7] est créée en 2021, après une restructuration de l'entreprise qui toucha l'équipe de développement de Rust.

Rust est différent car, au moment de la compilation, de nombreux problèmes sont détectés et signalés, là où souvent les compilateurs C/C++ les ignoreraient. Pour les déclarations de variables, un système de *propriété* existe. Pour éviter l'utilisation abusive de variables, il utilise un *vérificateur d'emprunt* qui s'applique à la compilation. Il faut aussi explicitement déclarer l'accès en lecture/écriture des variables passées par référence. La syntaxe de Rust ressemble beaucoup à celle du C/C++, avec l'usage familier de crochets autour des fonctions et mots-clés de contrôle.

La priorité à la sécurité du code justifie les efforts déployés pour fournir Rust aux développeurs de logiciels embarqués sur matériel dédié. Cependant, la nature de la programmation embarquée fait que la vérification statique du compilateur peut créer des problèmes avec certains types de code. Pour contourner ces problèmes, le code peut, par endroits être marqué comme *non-sûr* et, par ex., permettre de « déréférencer » des pointeurs. Définir explicitement des sections de code comme non-sûres clarifie le contournement des règles de Rust.

Faites un tour avec Rust

L'une des meilleures façons d'aborder Rust est d'expérimenter avec un Raspberry Pi. En suivant les instructions du site rustup.rs [8], l'installation est simple. Depuis la ligne de commande, il suffit de saisir la chaîne ci-après et de suivre les instructions :

```
curl --proto '=https' --tlsv1.2 -sSf https://sh.rustup.
   rs | sh
```

Contrairement au C/C++ qui sort des binaires, RUST produit un *crate* (littéralement une caisse, la déclinaison Rust du paquet). Le gestionnaire de paquets Cargo simplifie le processus de compilation en ligne de commande. Il stocke les données nécessaires pour produire le crate et permet au développeur de définir les logiciels requis pour le construire. En invoquant Cargo depuis la ligne de commande, un nouveau projet Rust est produit comme suit :

```
cargo new rust_test_project
```

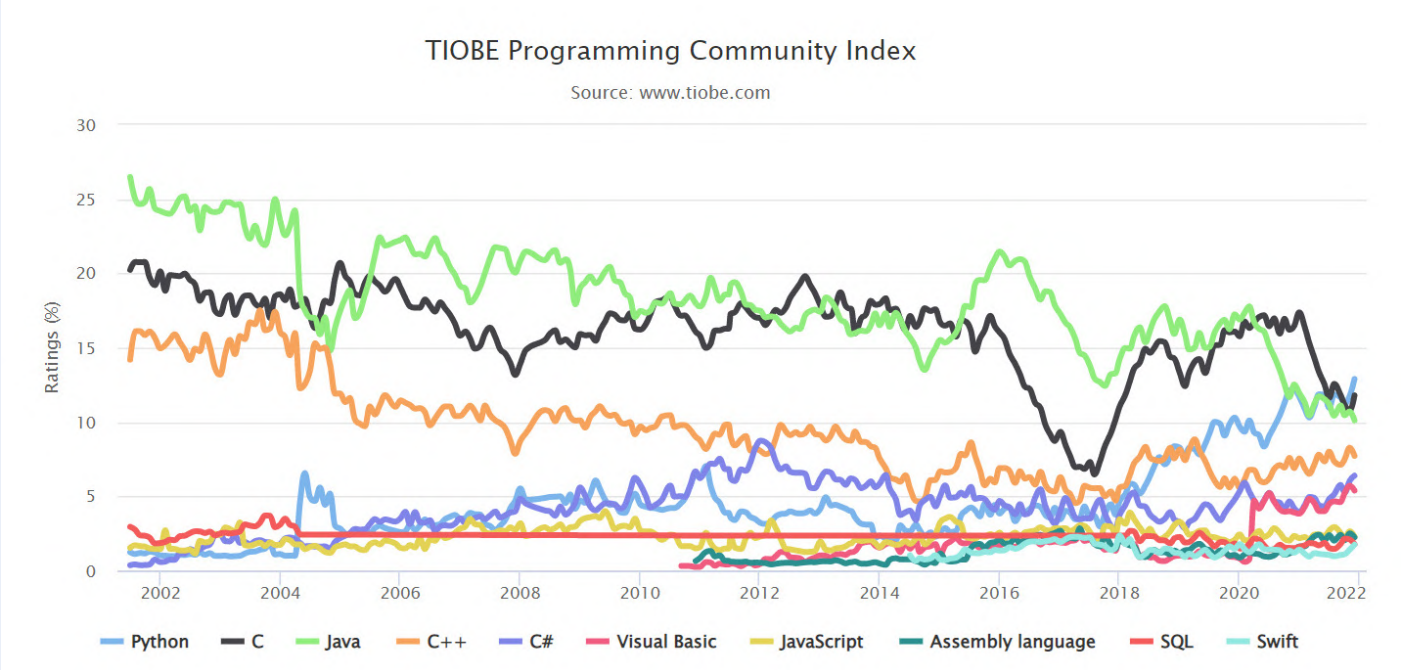


Figure 1. Le langage C reste un langage de programmation apprécié, se plaçant régulièrement dans le trio de tête des enquêtes et analyses de marché. (Source : www.tiobe.com)

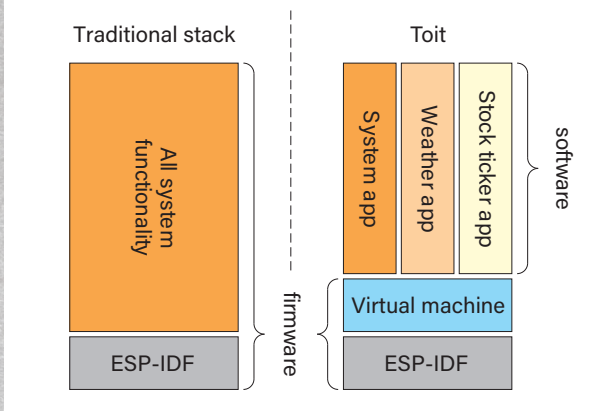


Figure 2. Toit exécute le code IoT sous forme d'applis au-dessus d'une machine virtuelle tournant sur un ESP32. (Source : Toit)

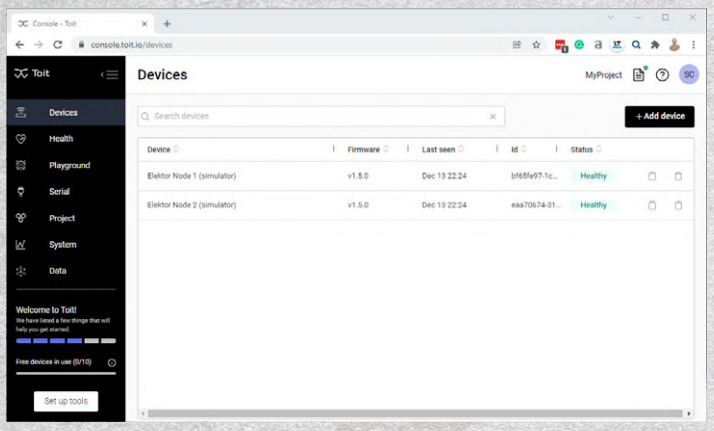


Figure 3. La console Toit affiche dans un navigateur deux nœuds ESP32 simulés.

Le dossier du nouveau projet contient un fichier nommé `Cargo.toml`. Il faut modifier ce fichier selon les besoins. Dans une simple application Raspberry Pi faisant clignoter une LED reliée à une broche GPIO, il faut définir une « dépendance de crate » idoine pour accéder aux lignes GPIO. Des crates sont partagés sur crates.io [9], la page créée à cet effet par la communauté Rust. La fonction de recherche permet de trouver un crate approprié : *rppal*. La plateforme indique si le crate peut être construit ou non, fournit le n° de version, une documentation et des exemples de code. Le nom et le n° de version du crate sont ensuite ajoutés comme *dépendances* dans `Cargo.toml` (**list. 1**).

Dans le dossier `src`, le développeur trouvera le fichier de code source Rust `main.rs`, un exemple de projet simple, *Hello World*. Remplacer ce code par le **list. 2** permet de faire clignoter dix fois une LED connectée à la borche GPIO 23 (br. 16 du connecteur Raspberry Pi). La compilation s'effectue en ligne de commande à l'aide de `cargo build`, et le crate s'exécute avec `cargo run`.

Un obstacle à l'utilisation de Rust sur certains µcontrôleurs est la nécessité d'utiliser une chaîne d'outils LLVM (*Low Level Virtual Machine*). Si celle-ci est disponible, l'utilisation d'un tutoriel en ligne peut commencer. Un exemple pour la carte BBC micro:bit [10] est fourni. Dès la syntaxe de Rust assimilée, il se montre très similaire à l'écriture de code en C/C++ (**list. 3** [11]). Un autre exemple pour STM32 [12] est fourni.

Listage 1. Ajout de la dépendance rppal à Cargo.toml dans un projet Rust.

```
[package]
name = "rust_test_project"
version = "0.1.0"
edition = "2021"
 [dependencies]
rppal = "0.13.1"
[dependencies]
rppal = "0.13.1"
```

Listage 2. Faire clignoter une LED en Rust.

Le code Rust pour faire clignoter une LED est similaire au code C. Cet exemple se base sur le code inclus dans le crate *rppal*.

```
use std::error::Error;
 use std::thread;
 use std::time::Duration;
use rppal::gpio::Gpio;
 use rppal::system::DeviceInfo;
// La GPIO utilise les n° de br. du BCM. BCM GPIO 23 est lié à la br. physique 16.
const GPIO_LED: u8 = 23;
fn main() -> Result<(), Box<dyn Error>> {
     let mut n = 1;
    println!("Blinking an LED on a {}.", DeviceInfo::new()?.model());
    let mut pin = Gpio::new()?.get(GPIO_LED)?.into_output();
    while n < 11 {         pin.set_
        // Faire clignoter la LED en mettant la br. à 1 durant 500 ms.
high();         thread::sleep(Duration::
from_millis(500));
pin.set_low();         thread::sleep(Duration::
from_millis(500));         n += 1;
     }
     Ok(())
 }
```

Listage 3. Extrait de code Rust simplifié qui vérifie l'état d'une broche d'entrée sur la carte BBC micro:bit.

```
#![no_std]
 #![no_main]

extern crate panic_abort;
extern crate cortex_m_rt as rt;
 extern crate microbit;

use rt::entry;
use microbit::hal::prelude::*;

#[entry]
 fn main() -> ! {
     if let Some(p) = microbit::Peripherals::take() {
         gpio         let mut
p.GPIO.split();// Split GPIO
 =

        // Configure le bouton GPIO en entrée
         let button_a = gpio.pin17.into_floating_input();

        // variable de boucle
         let mut state_a_low = false;

        loop {
             let // Obtenir l'état du bouton
button_a_low = button_a.is_low();

            if button_a_low && !state_a_low {
             }// Message de sortie

 {            if !button_a_low && state_a_low
                 // Message de sortie
             }

            // Mémoriser l'état du bouton
;             state_a_low = button_a_low
  }
     }
     panic!("End");
 }
```

Rust, est-ce l'avenir ?

Si oui, qu'est-ce qui freine l'adoption de Rust pour l'embarqué ? D'une part, il existe déjà Ada [13], un langage de programmation robuste adapté aux systèmes de sécurité critique. Malgré ses 40 ans, il n'a pas réussi à supplanter le C, bien qu'il fût développé spécifiquement pour les systèmes embarqués en temps réel. D'autre part, le C est partout : développeurs innombrables, outils et bibliothèques de code disponibles.

Toutefois, le gestionnaire de paquets Crate pourrait contribuer à accélérer l'adoption de Rust. Garder la trace des bibliothèques périphériques des fournisseurs écrites en C/C++ peut être difficile et opaque, aussi la définition explicite de la version des crates utilisée dans `Cargo.toml` pourrait susciter une large adhésion. Cela peut aussi simplifier la vie des fabricants de MCU qui peinent à prendre en charge leurs énormes portefeuilles de produits. Cependant, Ada a déjà réagi en sortant en 2020 son gestionnaire de paquets *Alire* [14]. Et, à l'instar de Rust, Ada inclut déjà le support de la carte BBC micro:bit. Pour comparer les deux langages avec un MCU, voir ce lien [15].

Maintenir les dispositifs IoT à jour

Avec un nombre toujours croissant de dispositifs embarqués connectés aux réseaux, le plus grand défi est de maintenir leur micrologiciel à jour. Ces m. à j. se font classiquement par téléchargement sans fil, en déposant le code binaire en mémoire flash à l'aide d'un chargeur d'amorçage embarqué et résidant en zone protégée. Toutefois, une mise à jour du µcode risque de ne pas se déployer correctement, et de rendre le produit inutilisable. Il peut aussi arriver qu'un bogue du nouveau code empêche les m. à j. ultérieures de se déployer, laissant les dispositifs touchés vulnérables. Ainsi, bien que l'application fonctionne correctement, l'appareil devient inutilisable, parfois à cause d'une erreur dans une seule ligne de code.

Depuis des années, les machines virtuelles (VM) déploient de façon standard plusieurs applications ou systèmes d'exploitation (OS) sur des serveurs. La VM exécute un OS, en faisant croire qu'il se trouve sur un matériel spécifique. En cas de défaillance catastrophique de cet OS, seule la VM concernée est touchée, et non les autres systèmes fonctionnant sur le serveur. Les utilisateurs de postes de travail connaissent les logiciels de virtualisation tels que VirtualBox, VMware et Parallels, leur permettant de tester des logiciels ou d'autres OS sans mettre en péril leur machine principale.

Mise à jour du µlogiciel des nœuds IoT : Toit sur place

L'équipe de Toit, entreprise danoise créée par d'anciens ingénieurs de Google, s'est demandé pourquoi la virtualisation était absente des MCU exécutant des applications IoT. En effet, elles nécessitent des m. à j. régulières pour corriger les bugs et prendre en charge les évolutions des services cloud avec lesquels elles communiquent. Il va de soi qu'aucune firme ne flashera manuellement les nœuds IoT sur place si des appareils sont bloqués après une m. à j.

Pour commencer, ils ont opté pour l'ESP32 d'Espressif. Ils recommandent l'ESP32-WROOM-32E avec microprocesseur Xtensa LX6 à 32 bits à double cœur, 520 Ko de SRAM et 4 Mo de flash. La plateforme a déjà son ESP-IDF (*Espressif IoT Development Framework*). Elle est donc prête à l'emploi comme nœud IoT. Toit a ensuite conçu une VM (**fig. 2**) qui permet aux applications de s'exécuter par-dessus en toute sécurité. Les applis sont écrites en Toit, un langage de haut niveau, orienté objet et sûr.

Test de Toit avec des nœuds ESP32 simulés

La gestion et le déploiement d'applications sur un ensemble de nœuds IoT s'effectuent via la console Toit couplée à Visual Studio Code de Microsoft. Pour qui ne souhaite que tester la plateforme, la console Toit permet d'utiliser des dispositifs ESP32 simulés. Après installation de l'extension Toit pour Visual Studio Code, les applications peuvent être écrites et simulées localement ou déployées sur les appareils connectés à la console.

Un fichier YAML accompagne les applis Toit. Ce fichier de langage de balisage sert à déclarer le nom de l'appli Toit, le fichier de code source et à définir les déclencheurs. Ces derniers peuvent lancer l'appli après le démarrage, l'installation et à intervalles de temps définis.

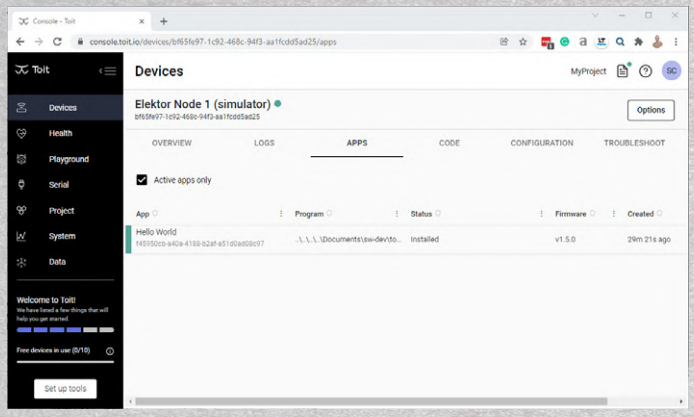


Figure 4. L'application Hello World installée avec succès sur un nœud simulé.

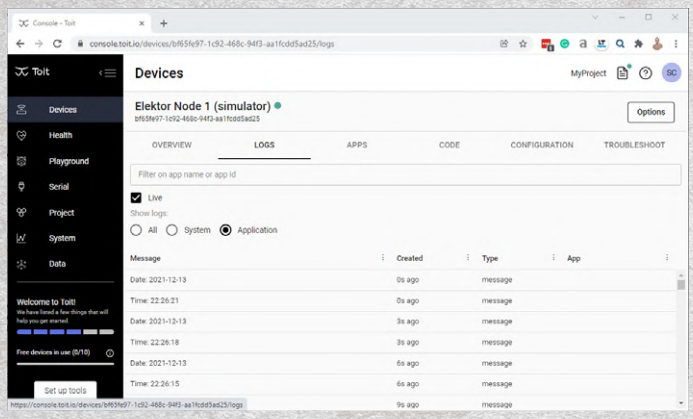


Figure 5. La sortie LOGS montre l'heure et la date émises par l'application Toit toutes les trois secondes, comme défini dans le fichier YAML.

Le déploiement d'applis ou de m. à j. ne nécessite ni la désactivation ni la mise hors tension du dispositif ESP32 cible. À la place, la machine virtuelle met à jour l'appli in situ et la lance selon les paramètres fournis par le fichier YAML. Si l'appli échoue, par ex. à cause d'un débordement de pile, les autres applis continuent de s'exécuter [16]. L'examen du journal de la Console permet de diagnostiquer de telles erreurs.

Le **listage 4** présente une appli simple *Hello World* affichant l'heure et la date, accompagnée de son fichier YAML (**list. 5**). La **figure 3** montre deux nœuds simulés dans la Console, tandis que la **figure 4** montre l'appli *Hello World* installée avec succès. Enfin, la **figure 5** montre la sortie de la date et de l'heure à intervalles de trois secondes, comme défini à l'aide du déclencheur `on_interval: "3s"` dans le fichier YAML. Visual Studio Code crée le code du projet et l'extension Toit le déploie sur le nœud choisi attaché au compte Console de l'utilisateur (**fig. 6**).

À première vue, l'approche Toit peut sembler un peu restrictive. L'environnement de virtualisation ne permet que le contrôle des GPIO, de l'UART, du SPI ou de l'I²C. Cela dit, vu que les nœuds IoT se contentent en général de collecter des données de capteurs pour les envoyer dans le cloud, cela fournit assez de flexibilité pour les applis. Toit exploite également son propre gestionnaire de paquets (registre) [17], fournissant des pilotes pour divers capteurs, périphériques d'entrée, écrans LCD et d'autres utilitaires. L'environnement prend aussi en charge le mode basse consommation de l'ESP32, et selon eux, deux piles AA font fonctionner le dispositif pendant des années [18]. Si une m. à j. d'appli arrive lorsque le nœud IoT est en mode sommeil profond, la Console la déploie au prochain réveil du nœud.

Rust et Toit – l'avenir de l'embarqué ?

Rust et Toit font tous deux ressortir deux choses : ils sont simples de mise en œuvre et gèrent les pilotes de bas niveau via des gestionnaires de logiciels. Ainsi les développeurs se concentrent sur leur travail : créer des applis. Les applis devenant de plus en plus complexes, cette réutilisation du code est vitale pour réduire le temps de développement et de commercialisation des produits.

Rust a une énorme montagne à gravir. Le C/C++ est si bien ancré dans le développement embarqué, qu'il est difficile de trouver son talon d'Achille et d'offrir un avantage assez important pour attirer les développeurs. De plus, déjà établi comme langage pour les systèmes critiques de sécurité, Ada fait écran aux avantages de Rust.

Toit, en revanche, résout un casse-tête majeur : maintenir les nœuds IoT distants à jour, déployer de nouvelles fonctions auprès de la base d'uti-

Listage 4. Application simple écrite en Toit qui envoie des chaînes date et heure formatées au journal de la Console.

```
main:
  time := Time.now.local
  print "Time: $(%02d time.h):$(%02d time.m):$(%02d time.s)"
  print "Date: $(%04d time.year)-$(%02d time.month)-$(%02d time.day)"
```

Listage 5. Fichier auxiliaire YAML indiquant à la Console Toit comment déployer l'appli.

```
name: Hello World
entrypoint: hello_world.toit
triggers:
  on_boot: true
  on_install: true
  on_interval: "3s"
  on_interval: "3s"
```

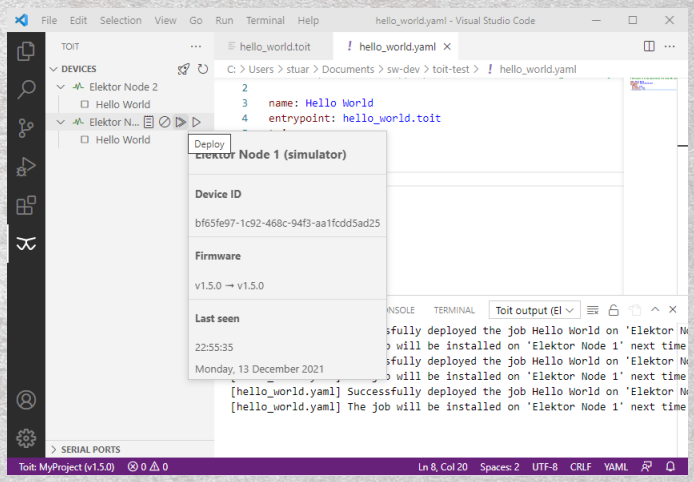


Figure 6. Grâce à l'extension Toit de Visual Studio Code, les m. à j. d'applis peuvent être déployées sans délai à distance sur les nœuds IoT sans crainte de les mettre en panne.

lisateurs, le tout sans aller sur place. Certains développeurs s'inquiéteront du minimum exigé de 4 Mo de flash et de la seule prise en charge actuelle d'ESP32. Toutefois, si une demande du marché pour d'autres MCU apparaît, il ne semble pas y avoir de raison technique qui empêcherait la prise en charge d'autres plateformes.

210652-04

Des questions, des commentaires ?

Envoyez un courriel à l'auteur (stuart.cording@elektor.com) ou contactez Elektor (redaction@elektor.fr).

Contributeurs

Texte et illustrations : **Stuart Cording**
Rédaction : **Jens Nickel, C. J. Abate**
Mise en page : **Harmen Heida**
Traduction : **Yves Georges**

PRODUITS

- Livre en anglais « BBC micro:bit » de D. Ibrahim, Elektor, 2016
  www.elektor.fr/17872
- BBC micro:bit Go Set de Joy-IT
  www.elektor.fr/18930

LIENS (TOUS EN ANGLAIS)

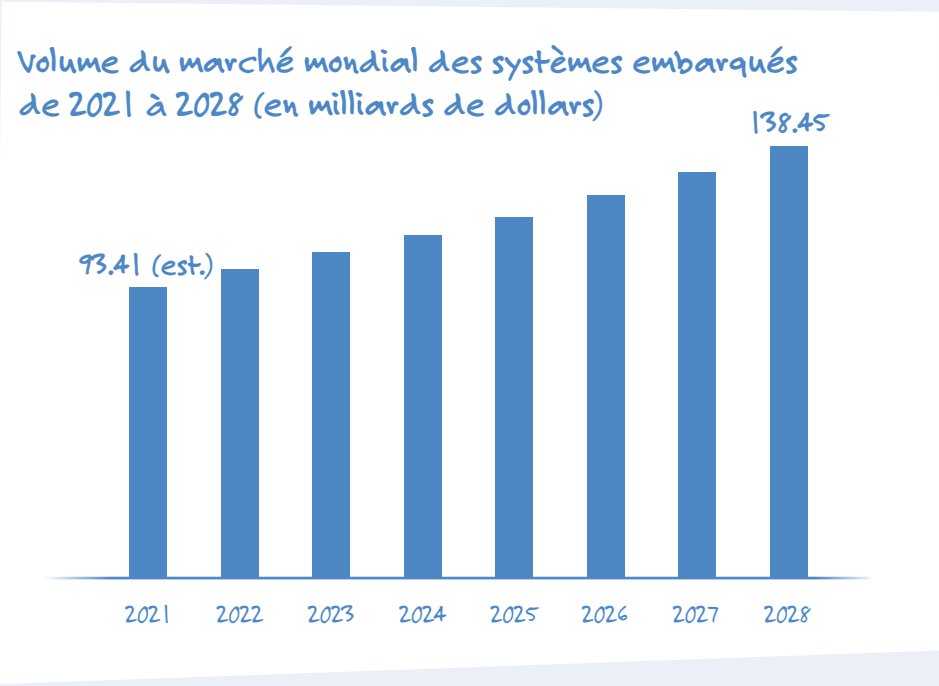
[1] Dennis Ritchie, Musée de l'histoire de l'informatique : https://bit.ly/3oOXG1A
[2] P. Jansen, « TIOBE Index for December 2021 », TIOBE Software BV, 12/2021 : https://bit.ly/3EXx8Ri.
[3] S. Cass, « Top Programming Languages 2021 », IEEE Spectrum, 2021 : https://bit.ly/3oPJq8z.
[4] Site de la MISRA : https://bit.ly/3s1Mv7D
[5] « C++ References », Tutorials Point : https://bit.ly/31Y6k53.
[6] « Guidelines for the use of the C++14 language in critical and safety-related systems », AUTOSAR, 10/2018 : https://bit.ly/3dO3zWl.
[7] Site de la Fondation Rust : https://bit.ly/3DNgO45
[8] Site rustup : https://bit.ly/3EUTiDR
[9] Site Rust, registre Crate : https://bit.ly/3dLKmor
[10] droogmic, « MicroRust », 04/2020 : https://bit.ly/3EUWFdM.
[11] droogmic, « MicroRust: Buttons », 04/2020 : https://bit.ly/3DWes30
[12] bors et NitinSaxenait, « Discovery », 12/2021 : http://bit.ly/3GERDT7.
[13] Site Get Ada Now : https://bit.ly/3GHyikw
[14] F. Chouteau, « First beta release of Alire, the package manager for Ada/SPARK », AdaCore, 10/2020 : https://bit.ly/3EUXOSC.
[15] F. Chouteau, « Microbit_examples », 12/2021 : https://bit.ly/3mnH7bx.
[16] « Watch us turn an ESP32 into a full computer! », Toit, 03/2021 : https://bit.ly/3IM0WT8.
[17] Page du registre des logiciels Toit : https://bit.ly/3yokpo7
[18] Docs Toit, prérequis, site : https://bit.ly/3oNSECq

Robert van der Zwan (Elektor)

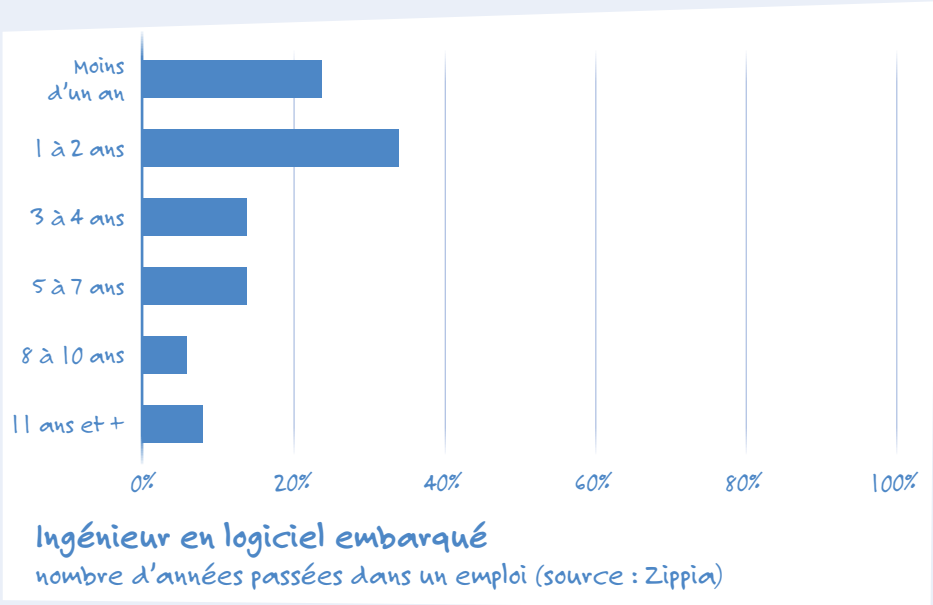
Embarqué : toujours en croissance

Selon la société d'études de marché The Brainy Insights, le marché mondial des systèmes embarqués semble toujours très prometteur, comme si la pandémie de Covid-19 n'avait jamais existé. Ses recherches concluent à un taux de croissance annuel composé (*CAGR*) de 5,73% pour les années 2021 à 2028. Bien que cette estimation soit plutôt prudente par rapport à d'autres sociétés d'études de marché, le rapport de la société d'études indienne a une note d'optimisme. L'automobile (part de marché de 5%), la santé (10%) et les communications (45%) sont des secteurs prometteurs dans le monde de l'embarqué et représentent déjà 60% de ses revenus.

(Sources : The Brainy Insights, The Insight Partners)

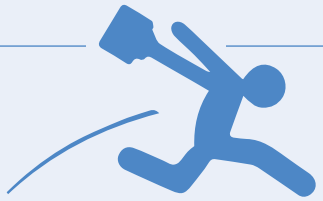


Ingénieur en logiciel embarqué ? C'est vous qui décidez.



Y a-t-il des facteurs qui freinent une partie du potentiel du marché de l'embarqué ? Tout à fait. L'un d'eux est le manque d'ingénieurs en logiciel qualifiés. Alors qu'en 2010, environ 4,5% des ingénieurs en logiciel embarqué étaient au chômage (chiffres américains), ce chiffre est aujourd'hui inférieur à 2%. Selon le cabinet de recrutement Built In, la demande d'ingénieurs en logiciel sera 21% plus élevée en 2028 qu'en 2021/2022. La croissance moyenne pour toutes les professions sera d'environ 5% pour la même période. Les ingénieurs en logiciel embarqué peuvent choisir ce qu'ils veulent, ce qui conduit 60% d'entre eux à changer d'emploi.

(Sources : Built In, site de recherche sur les carrières Zippia)



Passer d'un emploi à un autre

Scenario	Coût	Durée
Services de **conception de circuits imprimés** avec analyse de l'intégrité du signal et optimisation en option	~5K$ à 35K$	~1 à 8 semaines
Conception de schémas, **dessin de circuits imprimés** et développement de micrologiciels pour des microcontrôleurs de complexité faible à moyenne	~25K$ à 50K$	~6 à 12 semaines
Développement FPGA VHDL	~15K$ à 125K$	~1 à 6 mois et +
Développement de micrologiciels/logiciels	~10K$ à 125K$	~2 semaines à 6 mois et +

Pourquoi les ingénieurs en logiciel embarqué peuvent-ils passer d'un emploi à un autre sans se sentir mal à l'aise ? C'est parce que le travail de conception d'un système embarqué typique prend environ six mois. Jetez un coup d'œil au tableau établi par le développeur de systèmes embarqués AppliedLogix. Cette société américaine compte des clients dans un large éventail de secteurs et est qualifiée pour donner quelques moyennes. Ces moyennes montrent clairement qu'un ingénieur en logiciel embarqué peut terminer un travail, en être satisfait et passer à la mission suivante au sein ou en dehors de la même entreprise.

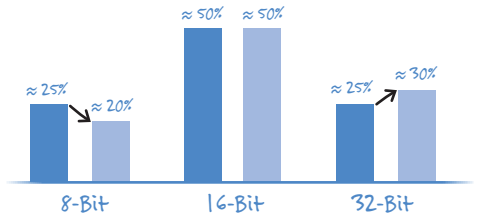
(Sources : AppliedLogix, Zippia)

Les **MCU à 8 bits** accusent un certain retard

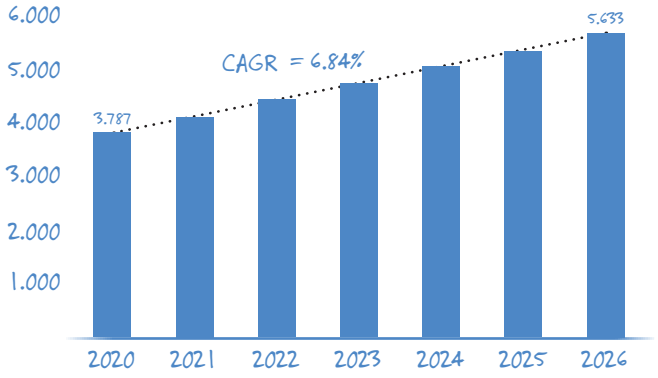
Bien que les microcontrôleurs à 8 bits soient toujours là, même à la fin de cette décennie, il est également vrai que les MCU à 32 bits font plus de progrès que prévu. Le fait que les contrôleurs à 32 bits gagnent du terrain est très probablement lié à la croissance rapide des systèmes embarqués en temps réel. Alors que les systèmes embarqués dans leur ensemble connaîtront une croissance annuelle de 5,7% d'ici à 2028, les systèmes embarqués en temps réel connaîtront une croissance supérieure d'un point de pourcentage (6,9%). En outre, le prix unitaire des MCU à 32 bits a diminué de façon constante. La répartition « 25% 8 bits, 25% 32 bits, 50% 16 bits » appartient au passé.

(Sources : Allied Market Research, The Brainy Insights, Grand View Research)

Part de marché des microcontrôleurs, 2021 vs. 2027



Marché mondial de la sécurité embarquée, 2020-2026 (milliards de dollars)
(Source : Knowledge Sourcing Intelligence)



La sécurité : où en est-on ?

La dernière fois que nous avons réalisé une infographie sur les solutions de sécurité pour les systèmes embarqués, il y a deux ans, il est apparu que les ingénieurs avaient du retard dans la mise en œuvre de ces solutions. Est-ce toujours le cas ? Non, ce n'est plus le cas. Le marché des solutions de sécurité pour les systèmes embarqués croît plus rapidement que le marché des systèmes embarqués lui-même. La différence de taux de croissance n'est pas spectaculaire – 5,73% contre 6,84% dans les années à venir – mais elle est suffisante pour ne plus s'alarmer. Non seulement la protection logicielle est mise en œuvre, mais aussi l'identification matérielle.

(Sources : The Brainy Insights, Knowledge Sourcing Intelligence)

Les bénéfices de la **5G** pour **l'industrie** et **l'automobile**

Mark Patrick (Mouser Electronics)

Les avantages de la 5G ne se limitent pas à la téléphonie mobile. Si l'augmentation des vitesses de téléchargement améliorera certainement l'expérience de navigation sur les smartphones, c'est dans des applications qui n'ont pas encore vu le jour que la 5G aura probablement davantage de retentissement. Voyons comment la technologie 5G risque d'impacter les secteurs de l'industrie et de l'automobile, et comment intégrer la 5G dans votre prochain projet de conception.

Pourquoi la 5G dans la production industrielle ?

À l'heure actuelle, un grand nombre d'usines intelligentes sont freinées dans leur développement par les limites des architectures câblées existantes, qui utilisent des réseaux éprouvés tels que l'Ethernet industriel, Profinet et CANbus pour connecter les divers capteurs, actionneurs et contrôleurs présents dans les équipements automatisés. Ces connexions câblées font de la moindre modification des installations de production un processus long et coûteux.

Les générations précédentes de réseaux sans fil, y compris la technologie 4G/LTE plus rapide, n'ont pas apporté la réactivité en temps réel et la faible latence nécessaires à l'autonomie. De plus, l'usine est un environnement opérationnel difficile, où des niveaux élevés de bruit électrique et d'interférences nuisent aux performances de nombreuses technologies antérieures de communication sans fil. Les capacités réseau améliorées de la 5G peuvent résoudre certains de ces problèmes et augmenter de ce fait l'efficacité et la flexibilité des systèmes.

La surveillance est l'une des fonctions clés de toute usine automatisée. La 5G apporte une capacité mMTC (Massive Machine-Type Communications) qui répond aux besoins des réseaux de capteurs sans fil (WSN) étendus. La 5G est également plus économe en énergie que ses prédécesseurs, un facteur essentiel pour prolonger l'autonomie de ces dispositifs connectés sur batterie et réduire ainsi la maintenance.

Pour le contrôle de mouvement et la robotique industrielle, qui nécessitent une précision et une sensibilité en temps réel, l'association du TSN (Time-Sensitive Networking) et de l'Ethernet industriel câblé s'est imposée comme la technologie réseau privilégiée. Avec sa communication ultra-fiable à faible latence (URLLC), la 5G est une alternative sans fil viable qui permet en outre la robotique dans le cloud.

La réalité virtuelle, la réalité augmentée et l'intelligence artificielle (RV/RA/IA) sont trois technologies connexes qui font leur apparition dans l'environnement des usines. Alliant vitesse élevée et URLLC, la 5G permet le traitement à la périphérie. Les calculs gourmands en énergie peuvent y être effectués dans le cloud, ce qui permet l'utilisation d'appareils moins complexes et moins coûteux sur le terrain.

La mise en œuvre de la 5G s'accompagne autant de défis que d'opportunités

Pour protéger les investissements antérieurs dans les technologies de réseaux câblés et sans fil, les projets 5G doivent s'intégrer de manière transparente dans l'infrastructure existante. L'une des principales difficultés rencontrées jusqu'à présent tient au fait que la couverture intérieur n'a jamais été une priorité pour les opérateurs de reseaux mobiles. Or, les avancées des technologies Open-RAN réduisent le coût de possession des réseaux d'accès radio 5G (5G RAN), faisant des déploiements de la 5G privée, également connus sous le nom de réseaux non publics (NPN), une réelle possibilité. Pour les entreprises qui préfèrent cette option, les régulateurs du monde entier mettent à disposition un spectre dédié et économique pour la 5G privée. En outre, selon les besoins opérationnels de l'usine, la 5G privée peut être soit totalement isolée du réseau public, soit partagée.

La 5G et l'ère de la voiture connectée

Le secteur automobile devrait lui aussi être à l'avant-garde du déploiement de la 5G, même si plusieurs années sont encore vraisemblablement nécessaires pour faire de l'autonomie de niveau 5

une réalité commerciale. On peut toutefois parier que votre prochaine voiture sera connectée à Internet pour gérer la télématique, le système de communication C-V2X (Cellular Vehicle-to-Everything) et l'info-divertissement.

La voiture connectée d'aujourd'hui peut générer jusqu'à quatre téraoctets de données par jour, soit l'équivalent d'environ 500 films. Les derniers progrès de la technologie de communication C-V2X utilisent déjà ces données de multiples façons. Les données issues des systèmes de gestion du moteur, par exemple, sont désormais envoyées à des centres de service à distance à des fins de maintenance prédictive. Les informations sur les conditions de circulation locales et la météo peuvent également contribuer aux systèmes de sécurité publique. Même le comportement du conducteur et le kilométrage du véhicule peuvent alimenter les bases de données des régimes d'assurance basés sur l'utilisation.
Au cours des cinq dernières années, le 3GPP (3rd Generation Partnership Project), un organisme de normalisation mondial pour les technologies de télécommunications cellulaires, y compris l'accès radio, le réseau cœur et les capacités de services, qui fournissent une description complète du système pour les télécommunications mobiles, a étendu les fonctionnalités C-V2X en lien avec l'évolution de la technologie des réseaux cellulaires. Les capacités de la version 16 ouvrent la voie aux systèmes avancés d'assistance au conducteur (ADAS).

Bien que la généralisation des voitures autonomes semble encore loin, on assiste déjà à des essais très médiatisés. Tesla, Google et BMW font la une des journaux, ce qui renforce les attentes du grand public et crée une dynamique. De nombreux véhicules haut de gamme possèdent déjà un certain niveau d'autonomie, certains atteignant le niveau 3, qui repose également sur les technologies C-V2X.
Bien que les réseaux 4G/LTE prennent en charge un grand nombre des applications mentionnées ci-avant, le volume croissant de données partagées exerce une pression de plus en plus forte sur la bande passante disponible. En outre, la faible latence s'impose comme une nécessité à mesure que les systèmes embarqués critiques de sécurité et de gestion de l'énergie deviennent de plus en plus sophistiqués. Les vitesses réseau et les capacités de traitement dans le cloud/en périphérie doivent prendre en charge des niveaux de latence dignes des réflexes humains pour atteindre de plus hauts niveaux d'autonomie. De même, pour les systèmes ADAS plus sophistiqués, la voiture connectée doit réagir aux événements environnants en temps réel. Le réseau sans fil actuel atteint ses limites et apparaît de plus en plus comme un obstacle ; sans la 5G, il n'y aura pas de voiture autonome.

En conclusion

Le déploiement du réseau 5G s'est en grande partie focalisé sur la mise à niveau du réseau 4G/LTE à partir des spécifications 5G NR NSA (New Radio Non-Standalone) du 3GPP, version 15, ce qui a permis le lancement d'un ensemble limité de services 5G. Or, le véritable potentiel de la 5G repose sur le déploiement de la version 16 du 3GPP et, plus tard, de la version 17. Les applications telles que la voiture autonome et l'usine autonome ne deviendront une réalité que lorsqu'elles auront facilement accès à ce niveau supérieur de performances réseau. Le déploiement initial de la 5G a joué la carte de la prudence et a été freiné par les répercussions de la pandémie. La deuxième vague de déploiement du réseau 5G va certainement accélérer la demande vis-à-vis d'un large éventail d'applications qui restent à découvrir.

220061-04

Pour plus d'informations sur la 5G, visitez le site « Empowering Innovation Together » de Mouser : **www.mouser.com/empowering-innovation/5G**.

À propos de l'auteur

En tant que directeur du marketing technique chez Mouser Electronics dans la région EMEA, Mark Patrick est responsable de la création et de la diffusion du contenu technique dans la zone, un contenu qui est essentiel à la stratégie de Mouser visant à soutenir, informer et inspirer son public d'ingénieurs. Avant de diriger l'équipe de marketing technique, Patrick faisait partie de l'équipe de marketing achat de la région EMEA et jouait un rôle essentiel dans l'établissement et le développement des relations avec les principaux partenaires de fabrication. En plus d'avoir occupé divers postes dans les domaines de la technique et du marketing, Patrick a travaillé pendant huit ans chez Texas Instruments, dans les secteurs de la gestion des applications et de la commercialisation des technologies. Ingénieur passionné par la pratique, amateur de synthétiseurs vintage et de motos, il n'hésite pas à les réparer. Patrick est titulaire d'un diplôme d'ingénieur en électronique (honneurs de première classe) de l'université de Coventry.

relais à bobine mobile

David Ashton (Australie)

Le composant auquel nous nous intéressons ici est vraiment spécial, au point qu'au-delà de ses propriétés physiques, nous n'avons pas réussi à en savoir beaucoup sur lui. Mais, même s'il est à bobine mobile, il ne s'agit sûrement pas d'un appareil de mesure !

Dites bobine (ou plutôt cadre) mobile, et la plupart d'entre vous penseront immédiatement aux bons vieux galvanomètres de mesure. Bien qu'ils soient aujourd'hui obsolètes, vous en avez sans doute vu ou utilisé à un moment ou un autre. Mais je parle ici de relais à bobine mobile. Imaginez que vous connectiez un fil à l'aiguille d'un voltmètre et un autre à sa butée. Lorsque l'appareil atteint sa pleine déviation, il établit le contact. Voilà en gros comment ces relais fonctionnent.

Il y a quelques années, j'ai trouvé ces relais dans un tableau de distribution et il m'a fallu du temps pour comprendre à quoi j'avais affaire. Ils sont d'une grande beauté mais, en plus, ils font partie des relais les plus sensibles que j'aie jamais rencontrés.

Ils ont été fabriqués par BBC Goerz Electro, société sur laquelle il est difficile de trouver des informations. Nous savons que Goerz était une société autrichienne d'optique. BBC (Brown Boveri Corporation) était et est toujours une entreprise suisse d'électricité qui a connu pas mal d'avatars pour devenir ABB (Asea Brown Boveri), son nom actuel. BBC Goerz a fabriqué de nombreux instruments de test professionnels sous la marque Metrawatt, et Gossen Metrawatt fabrique aujourd'hui des multimètres et autres appareils de test. C'est une carrière en dents de scie, en quelque sorte.

Même avec ces données sur le fabricant, je n'ai pu trouver aucune information sur ces relais, à part quelqu'un qui en vend sur eBay pour environ 100 $ pièce. Ils sont bâtis sur un culot octal. Quand je les ai reçus, je me suis demandé s'il s'agissait d'un genre de lampe. En fait, ils sont logés dans un tube en plastique transparent qui se dévisse, donnant accès à leur délicat mécanisme. Le numéro de type, 91041-2, est clairement imprimé sur le tube (**fig. 1**), avec un numéro de série ainsi qu'un schéma de connexion qui facilite leur test (**fig. 2**).

J'ai monté un banc de test pour ces relais sur un châssis avec trois embases et quelques trous pour monter d'autres composants. J'ai utilisé un potentiomètre de 470 kΩ en série avec la bobine du relais, ainsi qu'une alimentation de 12 V, et j'ai branché une LED en série avec le contact pour le voir se fermer. Ils fonctionnent avec un courant de seulement 260 µA pour une tension de 113 mV – trouvez-moi un autre relais aussi sensible !

Figure 1. Les relais, vus de dos, avec leur numéro de type imprimé sur leur tube en plastique.

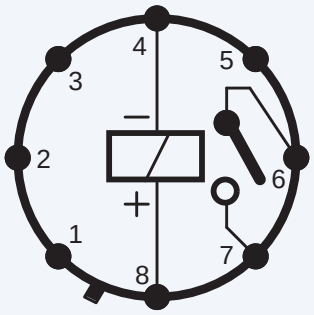


Figure 2. Ce schéma est imprimé sur le côté du relais.

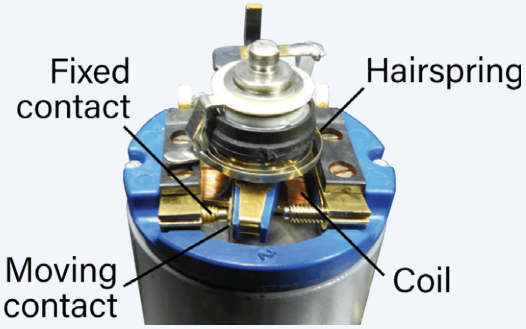


Figure 3. Une photo annotée identifiant le ressort spiral, la bobine et les contacts.

Les contacts sont petits (**fig. 3**) et il faudra les utiliser pour actionner un autre relais si l'on veut commuter une puissance significative. De nos jours, bien sûr, on utiliserait un optocoupleur pour faire la même chose, car il faudrait quand même ajouter un peu d'électronique à ces relais pour obtenir de telles spécifications, ce qui confirme qu'ils sont d'un autre âge. Ces relais sont vraiment spéciaux !

210557-04 - VF : Helmut Müller

Des questions, des commentaires ?
Contactez Elektor (redaction@elektor.fr).

elektorstore

www.elektor.fr

e-choppe Elektor

des produits et des prix surprenants

L'e-choppe Elektor a évolué. D'abord simple commerce de proximité pour nos productions (livres, magazines, kits et modules), c'est désormais une boutique en ligne bien rodée qui propose des produits surprenants à des prix très étudiés. Ce sont les produits que nous aimons et testons nous-mêmes. Si vous avez une suggestion, n'hésitez pas : sale@elektor.com. Seule exigence :
jamais cher, toujours surprenant !

Archives 1978-2021 d'Elektor sur clé USB

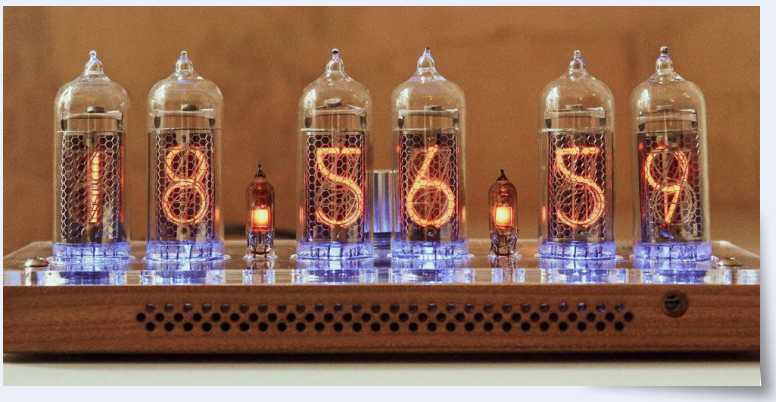
Prix : 199,95 €
Prix (membres) : 99,95 €

www.elektor.fr/20073

Horloge Nixie à 6 chiffres avec tubes IN-14

Prix : ~~299,00 €~~
Prix spécial : 269,00 €

www.elektor.fr/20044



Great Scott Gadgets Opera Cake – Commutateur d'antenne for HackRF One

Prix : 169,95 €

Prix (membres) : 152,96 €

www.elektor.fr/20083

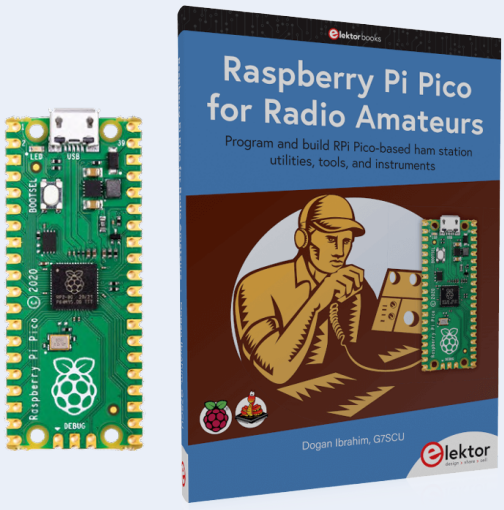
Velleman VTSS210 – Station de réparation pour CMS multifonctions

Prix : 84,95 €

Prix (membres) : 76,46 €

www.elektor.fr/19948

Raspberry Pi Pico for Radio Amateurs + Raspberry Pi Pico RP2040 GRATUIT

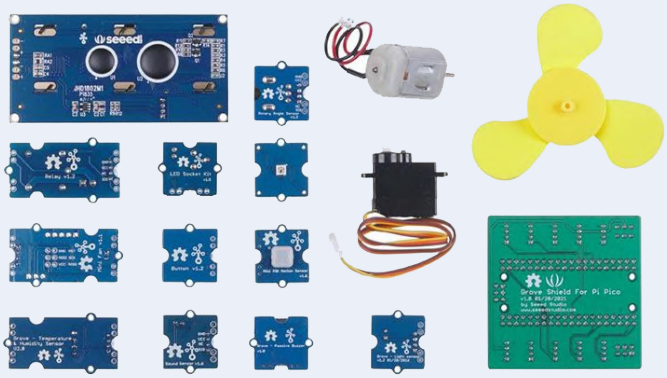


Prix : 34,95 €

Prix (membres) : 31,46 €

www.elektor.fr/20041

Grove Starter Kit de Seeed Studio pour Raspberry Pi Pico



Prix : 54,95 €

Prix (membres) : 49,46 €

www.elektor.fr/20016

visite à domicile

Chacun son tour

Joachim Schröder (Allemagne) et **Eric Bogers** (Elektor)

Tout électronicien devra un jour ou l'autre réaliser de ses propres mains un objet indispensable à son circuit, par exemple un boîtier embellissant un projet, ou à tout le moins le protégeant des affronts du monde. Que l'objet en question soit à travailler avec du bois ou du métal, une panoplie d'outils sera nécessaire. Avec, peut-être, un tour comme celui-ci.

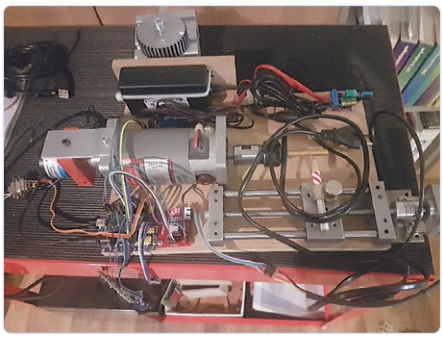


Figure 1. Un premier prototype avec un moteur CC pour la broche principale, un moteur pas-à-pas Nema 17 pour la vis-mère, et une commande par simple module PWM.

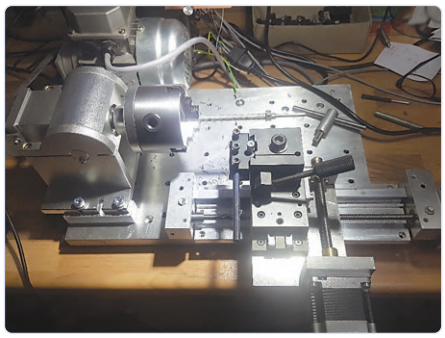


Figure 2. Prototype plus complet avec moteur CA, variateur de vitesse et moteurs pas à pas X et Z.

Percer quelques trous dans un boîtier pour y passer des boutons n'exige guère plus qu'un outillage et une dextérité de bricoleur ordinaire – une scie sauteuse et une perceuse tenues par une main sûre. Il n'en va plus de même des projets mécatroniques complexes nécessitant des mécanismes de guidage précis. Le recours à un tour s'avère dès lors vite indispensable – à moins bien sûr que vous ne soyez prêt à payer (relativement cher) les compétences d'un professionnel pour la réalisation de vos travaux mécaniques.

Joachim Schröder se sert professionnellement d'un tour manuel pour effectuer des réparations et des travaux d'usinage. Il possède ce tour depuis de nombreuses années, mais songeait depuis quelque temps à le transformer en tour à commande numérique dont la vis-mère serait synchronisée de façon électronique.

Dans ce type de tour, le chariot porte-outils est déplacé par un servomoteur ou un moteur pas-à-pas synchrones avec la rotation de la broche principale (p. ex. 0,1 mm par tour), la broche étant équipée à cet effet d'un codeur rotatif. L'avantage de cette commande numérique est qu'un déplacement plus grand ou plus petit par tour n'implique aucun changement d'engrenages.

Parce qu'il ne pouvait se permettre d'abandonner l'usage de son tour manuel, J. Schröder décida finalement de ne pas le modifier mais de partir de zéro pour construire un modèle à commande numérique. Ses premiers prototypes (**figures 1** et **2**) lui apprirent qu'un des plus gros problèmes à résoudre allait être de nature logicielle, à savoir le calcul de l'instant où le moteur devrait avancer d'un pas – en raison du temps de traitement requis, ce calcul était impossible à réaliser en virgule flottante (de prime abord le format numérique évident) : avec par exemple une broche principale effectuant 1200 tours par minute et un codeur produisant 1024 impulsions par rotation, il aurait fallu 20 480 calculs en virgule flottante par seconde. Un des grands défis fut donc de convertir ces calculs en opérations sur des entiers.

Un petit tour (de Chine) et puis s'en va

J. Schröder commanda donc un petit tour (manuel) de provenance chinoise, que le déballage révéla doté d'un robuste bâti en fonte (**figures 3** et **4**). Un bon point de départ donc, même si le plastique des engrenages était quant à lui aussi visiblement que complètement inadapté (mais peu importait, ces engrenages ne serviraient pas).

J. Schröder récupéra un engrenage sur la transmission d'origine, en équipa le codeur rotatif destiné à la synchronisation de la vis-mère, puis monta celui-ci à l'aide d'un adaptateur (**figures 5** et **6**).

Mais après seulement cinq jours d'utilisation, le variateur de vitesse d'origine (chinoise) du moteur CC de 230 V rendit l'âme, avec pour dernier soupir une forte détonation. J. Schröder le remplaça par un moteur CA de 0,37 kW et un variateur de qualité du fabricant allemand Lenze (**fig. 7**)

La **figure 8** montre le résultat final (provisoire ?) de tous ces déboires et efforts. La place réservée à cet article vous a hélas privé de bien des détails, mais si le projet vous intéresse vous pouvez contacter Joachim Schröder par courriel (voir **encadré**).

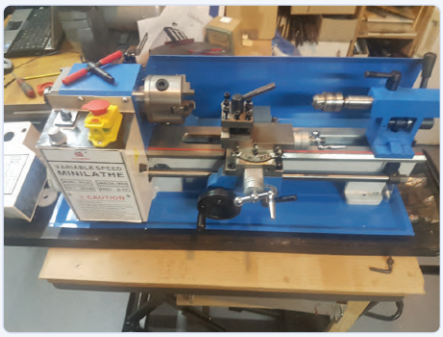


Figure 3. Le tour est (enfin) arrivé.

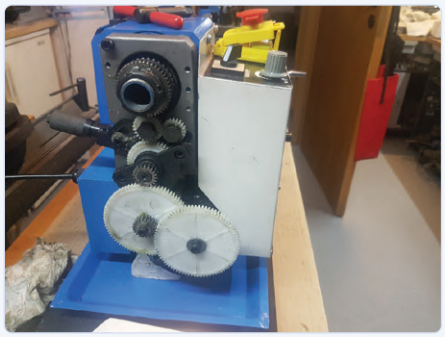


Figure 4. Le châssis est de qualité, mais les engrenages (en plastique) clairement bas de gamme.

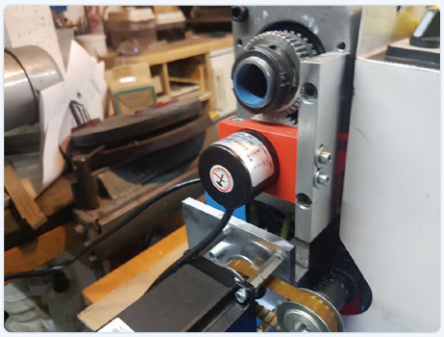


Figure 5. Le codeur pour la synchronisation est monté à l'aide d'un adaptateur...

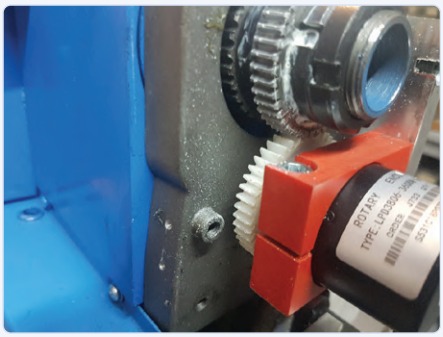


Figure 6. ...et entraîné par un engrenage récupéré sur la boîte de vitesses.

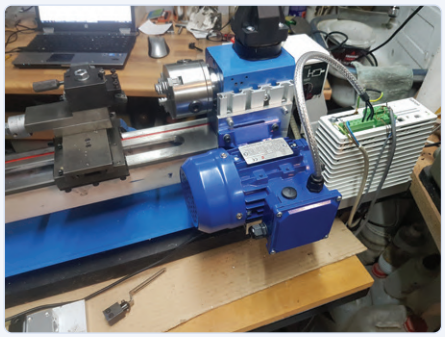


Figure 7. Montage du moteur de 0,37 kW et du variateur.

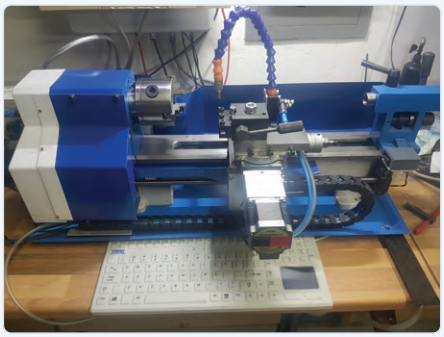


Figure 8. Version finale du tour.

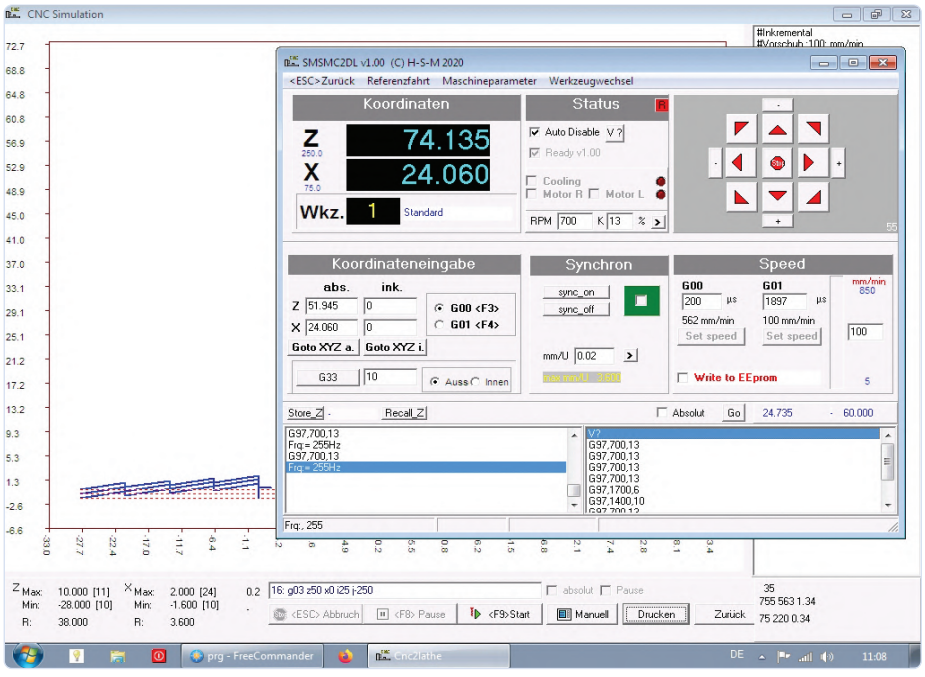


Figure 9. Capture d'écran du logiciel.

Un mot sur le logiciel

Laissons le mot de la fin à J. Schröder : « J'ai construit ma première fraiseuse numérique il y a plus de 30 ans et en ai écrit moi-même le logiciel. C'était sous DOS, à une époque où le matériel était commandé par le port LPT. J'ai ensuite construit des machines de plus en plus performantes, avec cette fois-ci Windows comme support du logiciel. Comme Windows n'offrait pas de capacités de traitement en temps réel, j'ai recouru à un microcontrôleur pour le contrôle des axes. J'ai d'abord utilisé des contrôleurs *CCBasic* de Conrad (c'était au début du siècle), mais l'impossibilité d'effectuer des calculs en virgule flottante constituait un véritable défi. J'utilise aujourd'hui des cartes Arduino qui fonctionnent parfaitement – elles peuvent commander des mouvements 3D jusqu'à 1200 tr/min. Utiliser le logiciel Arduino de ma fraiseuse 3D comme point de départ du logiciel de mon tour 2D s'imposait. Aussi facile à dire qu'à faire. La **figure 9** donne un aperçu de son interface. »

210605-04

Contributeurs

Texte et photos : **Joachim Schröder**
Rédaction : **Eric Bogers**
Mise en page : **Giel Dols**
Traduction : **Hervé Moreau**

Des questions, des commentaires ?

Envoyez un courriel à l'auteur du projet (js@hsm-ebs.de) ou contactez Elektor (redaction@elektor.fr).

voyage dans les réseaux neuronaux (4e partie)

Les neurones embarqués

Stuart Cording (Elektor)

Notre réseau neuronal fonctionne parfaitement sur les PC et les ordinateurs portables et nous pouvons mettre en œuvre en toute confiance notre perceptron multicouche. Nombre d'applications ont besoin des capacités de faible consommation et de latence minimale qu'offre un microcontrôleur. Certaines ne souhaitent tout simplement pas partager leurs données privées avec des services d'IA basés sur des systèmes tiers dans le nuage. Ici, nous rendrons notre réseau neuronal compatible avec Arduino et transférerons notre code de classification des feux de signalisation dans l'univers des systèmes embarqués.

Les outils et les plateformes d'apprentissage automatique (ML) semblent aujourd'hui pousser comme des champignons. Quelle que soit la complexité de votre tâche ou encore le volume de vos données, il existe un service dans le nuage (*cloud*) pour les traiter. Cependant, dans de nombreux cas, la mise en œuvre de l'apprentissage automatique dans le cloud est inadaptée. Si vous traitez des données sensibles ou personnelles, vous ne souhaitez peut-être pas les transférer via l'internet pour les faire analyser par un outil de ML. De même par ex. pour les applications automobiles ou d'autres applications embarquées en temps réel. Ces systèmes exigent une décision immédiate. Une solution de ML locale est donc nécessaire du fait de la latence d'une connexion internet ou de l'impossibilité éventuelle d'une connexion. Cette approche est également nommée « apprentissage automatique en périphérie de réseau » [1].

Avec un réseau neuronal fonctionnant en périphérie, même de simples microcontrôleurs peuvent exécuter des algorithmes de ML. Cependant, en raison des exigences relatives au processeur pour l'entraînement, le réseau est souvent entraîné dans le nuage ou à l'aide d'un PC puissant. Les poids résultants sont ensuite téléchargés sur le microcontrôleur pour un fonctionnement déconnecté d'internet et à faible latence.

Ce dernier article de cette série concerne le portage de notre réseau neuronal de perceptron multicouche sur un Arduino. Couplé à un capteur RVB, nous lui apprenons à classer les couleurs des feux de signalisation comme nous l'avons fait avec Processing sur PC ou ordinateur portable. En utilisant un Arduino à 32 bits (DUE ou Mo Pro), nous pouvons entreprendre la phase d'apprentissage sur le microcontrôleur. Cependant, nous allons constater qu'il est assez lent. Ainsi, nous scinderons également la tâche de manière à ce que le réseau soit entraîné en s'appuyant sur la puissance d'un PC, puis exécuté dans l'environnement à faible latence et basse consommation d'une application de microcontrôleur embarqué.

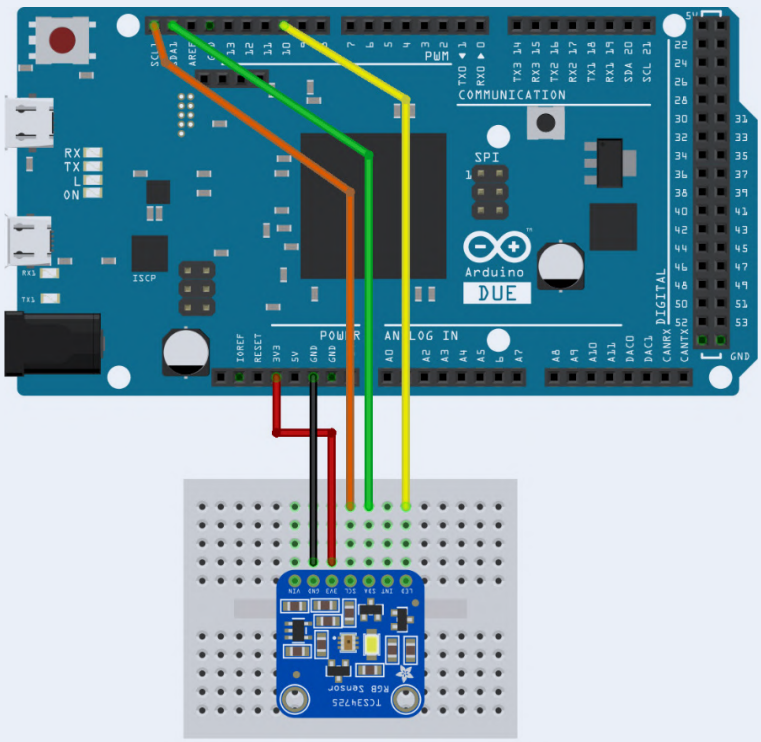
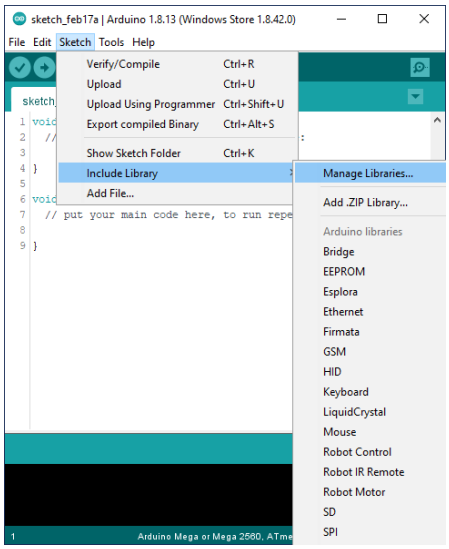


Figure 1. Schéma de câblage du capteur RVB TCS34725 avec un Arduino DUE.

Choix d'un capteur

Pour identifier les couleurs des feux de signalisation, l'Arduino aura besoin d'un capteur approprié en entrée. Le TCS34725 d'Adafruit est une solution de détection RVB appropriée, disponible et prête à l'emploi sous la forme d'une carte au format 2 × 2 cm [2]. Elle permet d'acquérir des données RVB, comme nous l'avons fait précédemment avec la caméra, et de les appliquer à la même configuration MLP utilisée pour les exemples donnés sur PC. La carte peut être utilisée avec les Arduino 5 V et 3,3 V, et les données du capteur sont



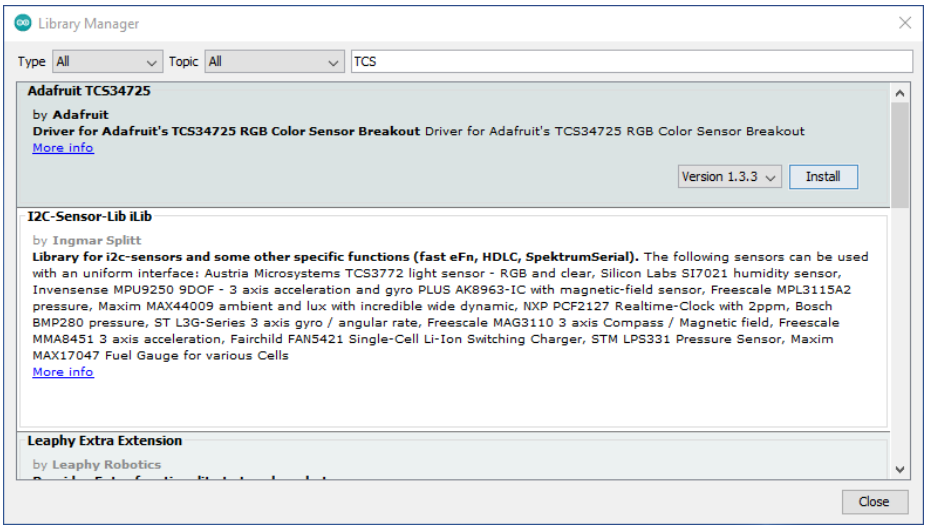


Figure 2. Installation de la bibliothèque logicielle du capteur RVB Adafruit TCS34725 dans l'EDI Arduino.

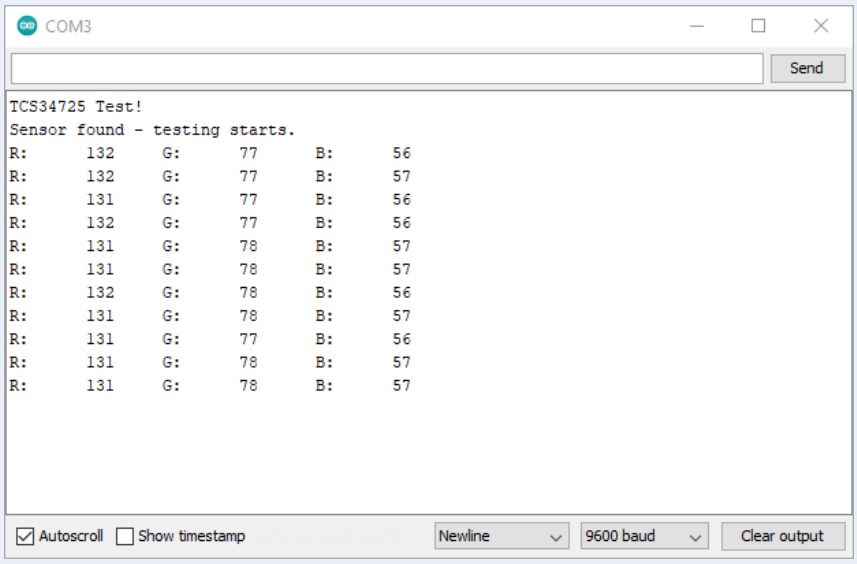


Figure 3. Sorties de valeurs RVB fournies par le croquis tcsrgbsensor.ino Arduino.

acquises à l'aide de l'interface I2C, ce qui nécessite la bibliothèque Wire.

Pour assurer un éclairage constant de l'échantillon à analyser, la carte comprend également une LED blanche commandée par une sortie numérique Arduino ; ici c'est la broche 10 (**fig. 1**). Fort heureusement, la carte est associée à une bibliothèque bien construite, ce qui permet une mise en place et un fonctionnement simples et directs. Comme précédemment, nous utiliserons le code du référentiel GitHub préparé pour cette série d'articles [3].

Avant d'exécuter du code, nous devons installer la bibliothèque pour le capteur RVB. Cela devrait être facile, car elle est accessible via le gestionnaire de bibliothèque dans l'EDI Arduino. Dans la barre de menu, il suffit de sélectionner *Croquis -> Inclure une bibliothèque -> Gérer les bibliothèques* puis d'entrer « tcs » dans le champ de recherche du Gestionnaire de bibliothèque. La bibliothèque « Adafruit TCS34725 » devrait apparaître. Cliquez simplement sur « Install » pour l'installer (**fig. 2**). En cas de difficultés, le code source et le fichier d'en-tête peuvent être téléchargés depuis le référentiel Adafruit GitHub [4] et ajoutés manuellement aux projets.

Pour s'assurer que le capteur fonctionne et disposer de la méthode d'acquisition des valeurs RVB dont nous avons besoin pour entraîner le dispositif de ML, nous allons commencer par le croquis *arduino/tcsrgbsensor/tcsrgbsensor.ino*. Après avoir initialisé le capteur RVB, le code allume la LED et commence à renvoyer les valeurs RVB via la sortie série (**fig. 3**). Ouvrez *Outils -> Moniteur série* pour les visualiser. Le réglage du débit est de 9600 bauds. Pour améliorer la qualité des relevés et réduire l'impact des autres sources de lumière, il est utile de fabriquer un écran de protection autour de la carte du capteur. Une bande de carton noir d'environ 3 cm de haut et 10 cm de long est idéale (**fig. 4**). L'écran permet également de maintenir l'image des feux de signalisation à une distance constante du capteur RVB.

Le capteur RVB produisant des résultats fiables, nous pouvons maintenant obtenir l'évaluation des couleurs rouge, orange et vert à l'aide de notre image de feux de circulation imprimée (à partir de *trafficlight/resources*). Les résultats obtenus par l'auteur sont présentés dans le **tableau 1**.

Tableau 1. Valeurs RVB capturées à l'aide du capteur TCS34725 pour les trois couleurs du feu de signalisation.

Feu tricolore	R	V	B
Rouge	149	56	61
Orange	123	77	61
Vert	67	100	90

Une bibliothèque MLP pour Arduino

Les valeurs RVB étant déterminées, nous avons besoin de l'implémentation du réseau neuronal pour la plateforme Arduino. Comme Processing utilise Java, nous ne pouvons pas simplement prendre le code de la classe Neural et l'ajouter à un projet Arduino. Nous avons donc légèrement modifié la classe Neural Java pour la transformer en classe C++. Dans l'univers Arduino, de tels objets de code réutilisables peuvent être transformés en « bibliothèques ». Celles-ci se composent d'une classe C++ et d'un fichier supplémentaire pour mettre en évidence les mots-clés de la classe. Si vous souhaitez écrire votre propre bibliothèque ou mieux comprendre le fonctionnement des classes C++ sur Arduino, il existe un excellent tutoriel sur le site web Arduino [5].

Le code de notre bibliothèque Neural se trouve dans *arduino/neural*. Les projets Arduino suivants incluent simplement le code source de la classe Neural dans le croquis pour simplifier l'écriture. Le fichier d'en-tête neural.h doit également être ajouté au dossier dans lequel est enregistré le projet.

L'utilisation de la classe Neural sur un Arduino est pour l'essentiel la même que dans Processing. La création de notre objet `network` à titre de variable globale est légèrement différente et réalisée comme suit :

```
Neural network;
```

Pour construire l'objet avec le nombre souhaité de nœuds d'entrée, cachés et de sortie, nous entrons les lignes suivantes :

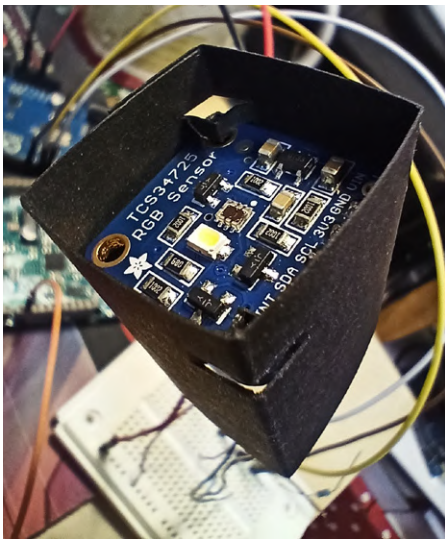


Figure 4. Capteur RVB TCS34725 avec son écran de protection noir.

```
network = new Neural(3,6,4);
```

La configuration de base des valeurs de biais et du taux d'apprentissage est ensuite codée comme précédemment dans Processing :

```
network.setLearningRate(0.5);
network.
setBiasInputToHidden(0.35);
network.
setBiasHiddenToOutput(0.60);
```

À partir de là, nous pouvons continuer à utiliser les méthodes appliquées précédemment dans Processing.

Détection des feux de circulation avec la carte Arduino

Nous pouvons maintenant commencer à explorer le MLP sur Arduino. Le croquis */arduino/tlight_detect/tlight_detect.ino* suit la même structure que le projet *Processing tlight_detect.pde*. Le réseau neuronal (3/6/4) est configuré à partir de la ligne 40, et il est entraîné avec les données RVB dans une boucle à partir de la ligne 51. Avant d'exécuter le code, les valeurs RVB pour « Rouge », « Orange » et « Vert » acquises précédemment doivent être saisies à partir de la ligne 56 :

```
teachRed(220, 56, 8);
teachAmber(216, 130, 11);
teachGreen(123, 150, 128);
```

Téléchargez le code et ouvrez le moniteur série pour visualiser la sortie (**fig. 5**). Comme précédemment, la vitesse de transmission doit être réglée sur de 9600 bauds. Le projet vérifie que le capteur RVB est fonctionnel avant d'éteindre la LED blanche pendant l'apprentissage. Une fois le MLP configuré, il effectue le cycle d'apprentissage 30.000 fois, améliorant à chaque fois sa capacité à classer chacune des trois couleurs.

Pour assurer un certain retour d'information pendant l'apprentissage, un point est émis tous les 1.000 cycles de boucle (3.000 époques d'apprentissage). Comparé au PC, l'apprentissage est un processus lent. Chaque appel à une fonction d'apprentissage (`learnRed()`, etc.) nécessite environ 5,55 ms. L'apprentissage des trois couleurs nécessite environ 8,5 min sur un Arduino M0 Pro doté d'un microcontrôleur SAMD21. Si vous souhaitez examiner le temps d'exécution de votre plateforme, cochez la case « Show Timestamp » dans le moniteur série et remplacez la ligne 66 par :

```
Serial.println(".");
```

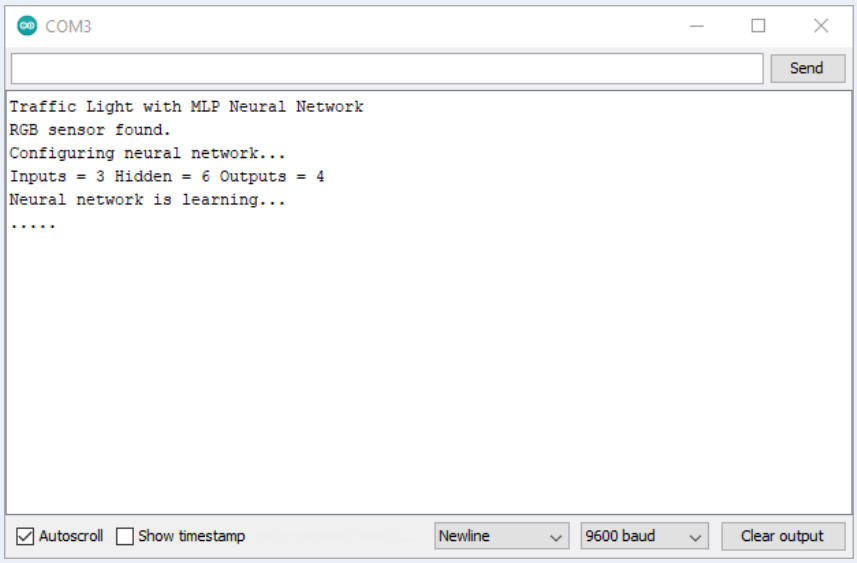


Figure 5. Sortie de tlight_detect.ino pendant l'apprentissage.

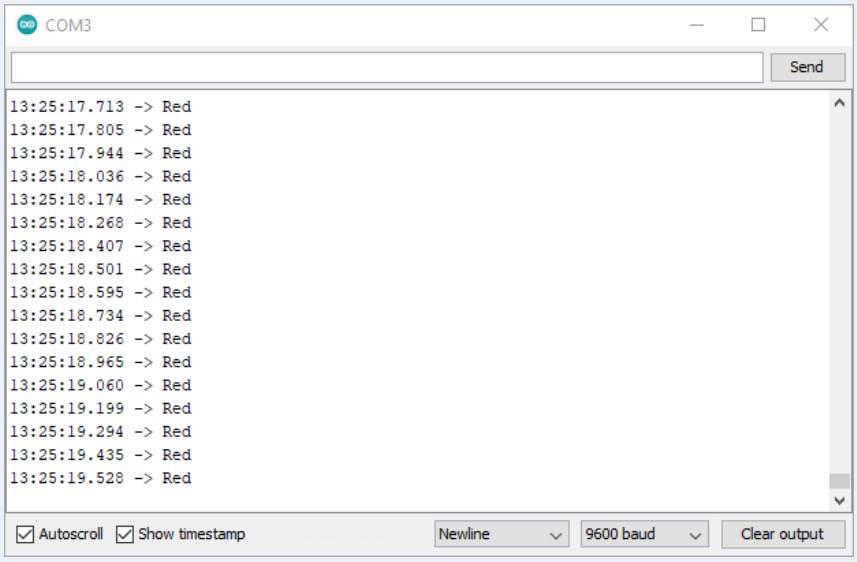


Figure 6. Résultat de tlight_detect.ino après apprentissage et détection des couleurs.

Vous ajoutez ainsi le caractère retour chariot et un horodatage est généré pour chaque caractère « . » produit.

Une fois l'apprentissage terminé, l'Arduino est capable de montrer immédiatement sa nouvelle compétence. Chaque fois qu'une couleur de feu de circulation est détectée, elle est envoyée au moniteur série (**fig. 6**). Au cours de l'expérimentation de l'auteur, le réseau neuronal a également détecté la couleur « orange » même si rien n'était placé devant le capteur. Bien que ce phénomène semble être lié à l'éclairage ambiant, il met en évidence une faiblesse de fonctionnement.

Pour améliorer le code, nous pouvons faire connaître au MLP les couleurs « autres », comme nous l'avons fait précédemment dans Processing. L'approche peut également servir à écarter la classification « orange » en l'absence d'image. Le croquis *tcsrgbsensor.ino* peut être utilisé pour acquérir les lectures du capteur pour l'environnement du feu de signalisation, le cadre et l'arrière-plan de l'image. Ces valeurs peuvent ensuite être entrées aux lignes 60 et suivantes du croquis *tlight_detect.ino* dans les appels de fonction `teachOther()`.

Les valeurs indiquées dans le **tableau 2** ont été atteintes et testées avec le croquis *tlight_detect.ino*. La classification s'est améliorée, mais la classification erronée « orange » en l'absence d'image n'a pas été totalement résolue. Comme toujours, des améliorations sont possibles !

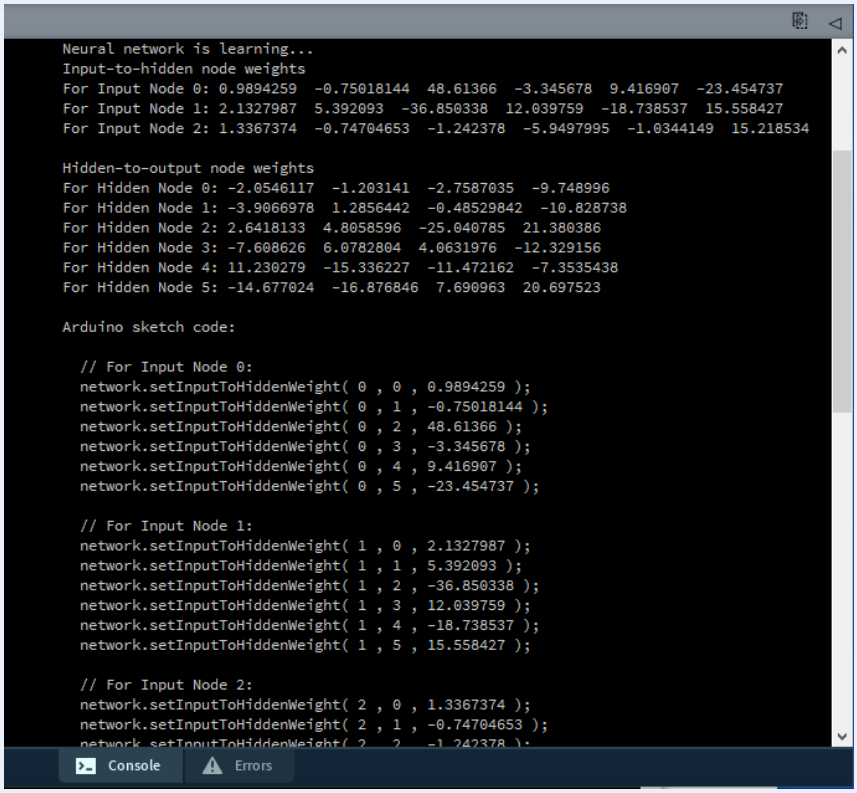


Figure 7. Un PC permet de calculer rapidement les poids en utilisant learnfast.pde dans Processing. La sortie de la console texte peut ensuite être collée dans tlight_weights.ino.

Tableau 2. Valeurs RVB capturées à l'aide du capteur TCS34725 pour les couleurs « indésirables ».

Couleur	R	V	B
Feu tricolore dans une ambiance sombre	92	90	82
Cadre blanc	92	90	75
Fond bleu	73	93	89

Apprentissage accéléré

Si vous ajoutez l'apprentissage des « autres » couleurs au croquis Arduino, l'attente sera d'environ 18 min pour qu'un Mo Pro assimile les classifications souhaitées. Il y a donc certainement la possibilité d'optimiser le processus. Puisque nous disposons d'un PC puissant capable de calculer les poids en moins d'une seconde, il serait logique de faire l'apprentissage avec, puis de transférer les résultats vers l'Arduino. Avec ces poids, nous avons également un moyen de programmer plusieurs microcontrôleurs avec les mêmes « connaissances ». Si les capteurs RVB fonctionnent tous de manière similaire par rapport à notre entrée, chaque microcontrôleur devrait classer correctement l'image du feu de signalisation. C'est donc ce que nous allons réaliser ensuite.

Nous revenons brièvement à Processing pour ouvrir le projet */arduino/learnfast/learnfast.pde*. L'ensemble de l'application s'exécute dans la fonction `setup()`. Le réseau neuronal est configuré avec les mêmes nœuds d'entrée, cachés et de sortie que ceux utilisés sur la carte Arduino (3/6/4). Dans la boucle d'apprentissage (ligne 36), les valeurs utilisées sont celles acquises par l'Arduino à l'aide du capteur RVB et du croquis *tcsrgbsensor.ino*. Lorsque le code est exécuté, il envoie du texte sur sa console. La dernière section contient le code qui configure tous les poids entrée-caché et caché-sortie (**fig. 7**). Il suffit de copier le code généré à partir de la ligne `// For Input Node =0` jusqu'à la fin de la sortie de texte.

De retour dans l'IDE Arduino, nous pouvons maintenant ouvrir le croquis */arduino/tlight_weights/tlight_weights.ino*. Il s'agit de la même chose que le croquis *tlight_detect.ino* mais, avec une préprogrammation des poids au lieu d'une phase d'entraînement du réseau neuronal. C'est le cas à la ligne 51 avec la fonction `importWeights()`. Collez simplement le code de la sortie de *learnfast.pde* dans la fonction `importWeights()` à la ligne 86 dans *tlight_weights.ino*. Programmez la carte Arduino, et elle devrait détecter avec précision les couleurs des feux de signalisation comme auparavant.

En fait, maintenant que nous avons ce processus d'apprentissage accéléré, nous pouvons aussi programmer le même croquis *tlight_weights.ino* dans un Arduino UNO. Il suffit de brancher le capteur RVB à la carte, d'ouvrir le moniteur série, et le fonctionnement est tout aussi précis qu'avec un Arduino Mo Pro ou DUE. À titre de comparaison, vous pouvez scruter la broche 9 pour voir combien de temps il faut à la méthode `calculateOutput()` pour effectuer les calculs.

Et ensuite ? D'autres systèmes embarqués ?

Alors, que pouvons-nous faire à partir de là ? Eh bien, nous disposons d'un réseau neuronal MLP opérationnel qui fonctionne à la fois sur PC et sur microcontrôleur. Nous disposons également d'un processus d'apprentissage accéléré pour générer les poids nécessaires aux applications sur microcontrôleur. Vous pouvez essayer d'appliquer ce MLP à des tâches trop difficiles à résoudre à l'aide d'instructions *if-else* et de limites fixes. Il pourrait même être possible de mettre en œuvre un projet simple de reconnaissance vocale pour identifier quelques mots. Vous pourriez également explorer les éléments suivants :

- La classe Neural utilise le type de données double. Peut-elle être accélérée en utilisant le type *float* tout en conservant sa précision ? Quelle vitesse d'exécution peut-elle atteindre ?
- La fonction sigmoïde utilise la fonction mathématique *exp()*, qui calcule l'exponentielle élevée à l'argument transmis. La fonction d'activation peut-elle être simplifiée par une approximation tout en assurant une classification précise ?
- Si vous vouliez tenter la reconnaissance vocale, comment prépareriez-vous un échantillon de voix pour le présenter à ce dispositif MLP ?
- Et si vous apportiez des modifications importantes ? Comment implémenter une deuxième couche de nœuds

cachés ? Pouvez-vous implémenter une fonction d'activation différente ?

Dans cette série d'articles, nous avons couvert un large domaine et découvert les premières recherches sur les neurones artificiels. À partir de là, nous avons examiné comment les MLP utilisent la rétropropagation pour apprendre, puis étudié leur fonctionnement en utilisant les puissantes capacités de traitement d'un PC. Nous avons ensuite porté le MLP sur un microcontrôleur à titre d'exemple de dispositif d'apprentissage automatique en périphérie de réseau. Si vous décidez de développer ces exemples plus avant, n'hésitez pas à partager vos résultats avec nous, chez Elektor.

210160-D-04

Des questions, des commentaires ?

Envoyez un courriel à l'auteur (stuart.cording@elektor.com).

Contributeurs

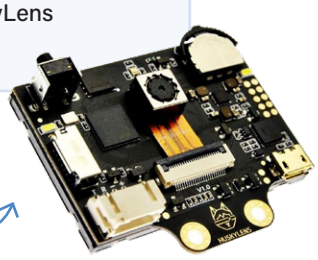
Idée, texte et images : **Stuart Cording**
Rédaction : **Jens Nickel, C. J. Abate**
Illustrations: **Patrick Wielders**
Mise en page : **Harmen Heida**
Traduction : **Pascal Godart**

Une agriculture plus performante

L'agriculture s'est toujours appuyée sur la nature et les saisons afin de déterminer le meilleur moment pour récolter et semer. La tradition a toujours dicté la date optimale pour planter en bénéficiant de la mousson. Cependant, avec l'évolution des conditions météorologiques, les rendements des cultures ont souffert. Tout cela est en train de changer grâce à l'IA. Des agriculteurs indiens participant à un projet pilote ont effectué leurs plantations d'arachide fin juin, soit trois semaines plus tard que la normale. C'est la suite Microsoft Cortana Intelligence qui a produit ces conseils. Grâce aux historiques de données climatiques, les recommandations reçues par les agriculteurs ont permis d'obtenir un rendement supérieur de 30 % en moyenne [6].

PRODUITS

- Livre en anglais, « Artificial Intelligence » de B. van Dam
  www.elektor.fr/18090
- Kit AIY Vision pour Raspberry Pi de Google
  www.elektor.fr/19365
- Caméra AI avec boîtier en silicone de HuskyLens
  www.elektor.fr/19248



LIENS

[1] M. Patrick, « ML at the Edge: a Practical Example », codemotion, 09/2020 : https://bit.ly/2ZQYipU
[2] Capteur de couleur RVB avec filtre IR et LED blanche – TCS34725 : http://bit.ly/2NKFS7T
[3] Dépôt GitHub – simple-neural-network : https://bit.ly/2ZHLv9p
[4] Dépôt GitHub – Adafruit_TCS34725 : http://bit.ly/37RJg81
[5] « Writing a Library for Arduino », Arduino : http://bit.ly/3aWeNaP
[6] « Digital Agriculture: Farmers in India are using AI to increase crop yields », Microsoft News Center, 11/2017 : http://bit.ly/2PacsQZ

lévitation magnétique
encore plus simple

Troisième version : la plus compacte

Peter Neufeld (Allemagne)

Voyons un circuit analogique simple permettant de faire léviter de petits objets, qui a suscité de nombreuses discussions et expériences. Le matériel ressemble à celui déjà publié dans *Elektor*, mais avec encore moins de composants. L'électronique tient même dans le boîtier d'un relais industriel standard modifié !

J'ai partagé mes expériences sur les circuits de lévitation magnétique sur la page *Elektor Labs*. Les deux premières versions ont été publiées en ligne et dans *Elektor* : la 1ère était entièrement analogique avec un comparateur *LM311* [1], et la 2e numérique [2], pilotée par un module à microcontrôleur, basé sur *ESP32*. Je ferai ici référence à ces articles par **Partie I** et **Partie II**. En discutant de la lévitation magnétique avec Luc Lemmens, ingénieur du labo d'Elektor, certaines idées sont apparues. Je les ai suivies pour améliorer et optimiser les projets que j'avais déjà réalisés. Certaines modifications avaient déjà été mises en œuvre dans les circuits des dites publications, mais nos discussions ont aussi débouché sur une optimisation supplémentaire. Le résultat, objet de cet article, a permis de simplifier notablement « l'ancien » circuit analogique de lévitation pour petits objets (magnétiques). Ceci est la 3e et, du moins pour l'instant, dernière partie de mes projets de lévitation magnétique. Même si la conception est plus simple que pour les deux autres projets, cela fonctionne très bien.

Le principe, en bref

L'idée-force de ces trois projets de lévitation magnétique est que le champ magnétique de la bobine de relais modifiée ne fournit qu'une faible partie de la force de lévitation nécessaire. La majeure partie est celle qu'un aimant permanent placé dans l'objet en lévitation exerce sur le noyau de fer du solénoïde. Un capteur à effet Hall mesure l'intensité du champ magnétique total (champ statique de l'aimant + champ de la bobine). La tension sur sa broche de sortie est une mesure de la distance entre la bobine et l'objet en lévitation. Son retour en entrée du circuit de commande du courant de l'électroaimant forme une boucle de contre-réaction et donc régule la distance entre le noyau et l'objet en lévitation.

Un potentiomètre permet le préréglage fin du circuit de commande pour l'ajuster à l'objet magnétique qui doit léviter sous la bobine. Si vous êtes novice sur cette expérimentation, ce réglage peut être vite décourageant. En cas de problème, lisez la **Partie I** [1] pour ses conseils et astuces. Le circuit marche bien, mais il est bien sûr limité pour ce qui concerne la taille et le poids des objets qu'il peut soulever.

Circuit simplifié

La **figure 1** donne le schéma de cette 3e version, une version simplifiée de la **Partie I**. Le comparateur LM311 [3] fournit ici le courant de commande (assez faible) directement à la bobine, donc sans transistor externe. Peut-être le savez-vous : la sortie à collecteur ouvert du LM311 peut piloter une petite charge. Cela simplifie notablement le circuit de la 1ère solution 100 % analogique, conçue pour la **Partie I**, dans laquelle un transistor *BC550 NPN* ajouté à la sortie du comparateur alimentait l'électroaimant.

Pour réduire encore plus le nombre de composants et gagner un peu de place, le diviseur de tension réglable de la référence sur l'entrée inverseuse du comparateur (P1 et R4) peut aussi être modifié. On peut remplacer ces deux composants par un seul ajustable multitour ; il n'est pas critique, toute valeur entre 1 kΩ et 1 MΩ fera l'affaire. Comme la **figure 2** le prouve, l'ajustable de préréglage à un tour et la résistance fixe R4 s'insèrent bien dans le boîtier du relais, cette dernière modification n'est donc pas indispensable.

Les composants utilisés ici sont très répandus, disponibles et abordables. Dans la **Partie I**, nous avions écrit que *seuls* les capteurs à effet Hall *A1302* ou *A1308* d'*Allegro Microsystems* étaient utilisables bien que par ex. les SS49E fussent moins chers et plus facilement disponibles. Cependant, dans la **Partie II** *Lévitation magnétique, version numérique*, nous constations avec satisfaction que cette assertion était erronée. D'autres tests ont prouvé que le type et la marque d'IC2 ne sont pas vraiment critiques, ce qui facilite notablement la recherche des composants nécessaires à ces projets. Les capteurs Allegro sont plus chers et difficiles à trouver.

Comment trouver le bon relais et le modifier ?

L'électroaimant (L1) est désormais le solénoïde d'un relais 5 V standard, pour être plus précis, de marque Finder (réf. 40.52.7.005.0000). Il est facilement disponible et bien plus facile à modifier que les relais 5 V suggérés auparavant. Les photos de la **figure 3** illustrent le processus complet de modification du relais.

Le boîtier transparent d'un relais Finder est souvent collé à sa base. Il est cependant facile de l'ouvrir en chauffant modérément le joint adhésif avec un sèche-cheveux, ou une station de soudage à air chaud. Un couteau Exacto, légèrement chauffé, peut faciliter l'ouverture sans faire de dégâts. C'est justement ce que nous voulons ici : garder le capot intact et le réutiliser comme boîtier pour l'électronique complète du dispositif de lévitation.

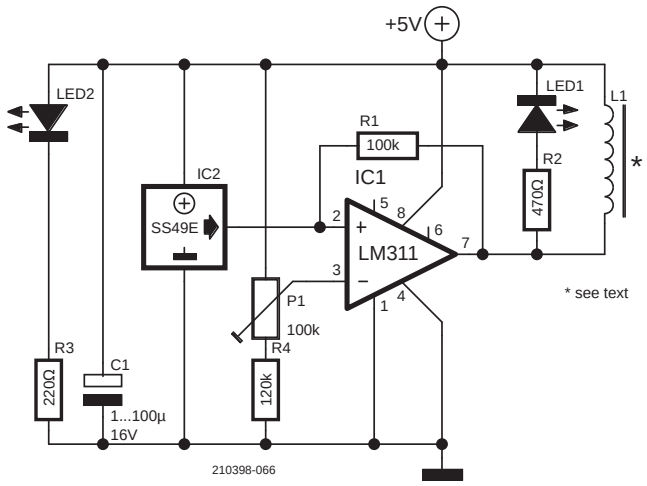


Figure 1. Schéma de cette version de la lévitation magnétique.

Figure 2. L'ajustable à un tour et la résistance en série tiennent dans le boîtier.

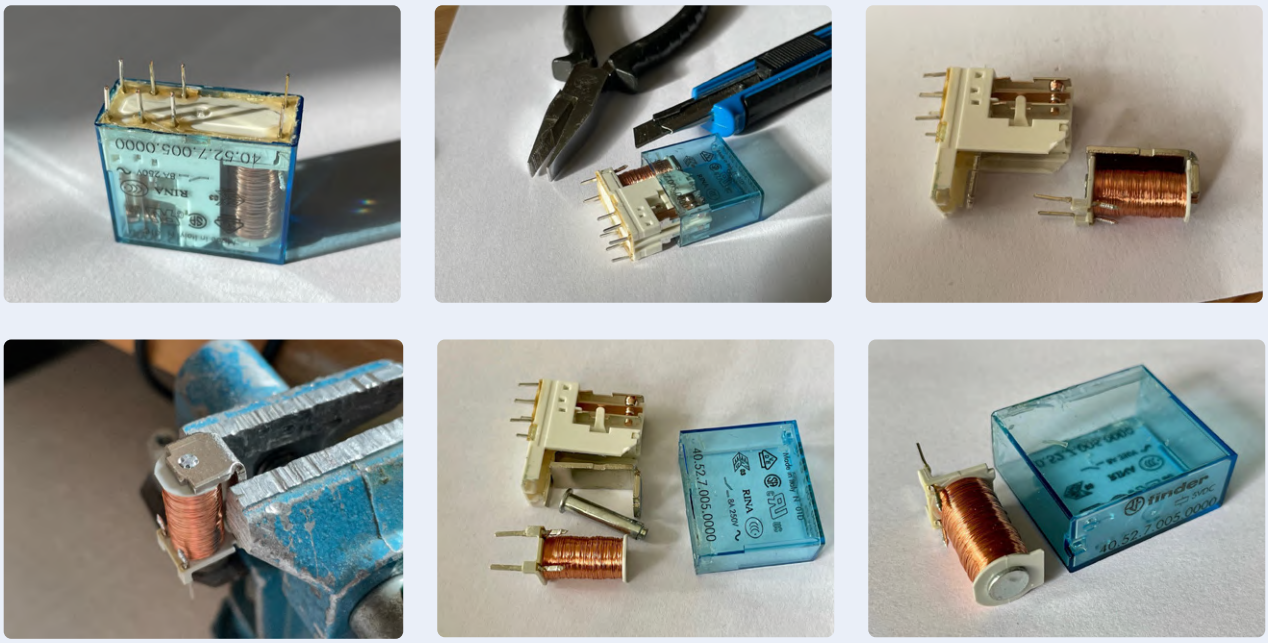


Figure 3. Démontage du relais et préparation du solénoïde.

Comme pour les relais utilisés précédemment, le circuit magnétique doit être ouvert pour éviter un court-circuit magnétique : ce doit être un I et non un U. Cette opération est assez aisée, car le noyau rond et la partie plate rectangulaire ne sont liés que par martelage. On peut rompre cette liaison avec une petite disqueuse, un fraisage ou perçage superficiel.
Le noyau de fer peut alors être retiré (s'il ne tombe pas tout seul) et retourné pour avoir une surface polaire un peu plus grande et également plate et y coller le capteur à effet Hall, (cf. **fig. 4**). Cela permet de placer l'électroaimant et le capteur à effet Hall dans le boîtier du relais. Ils y seront bien à l'abri des chocs de l'aimant.
Avec le montage alimenté en 5 V, le courant consommé avec un objet magnétique lévitant sous la bobine n'est que de 50 mA et au maximum de 90 mA, si la bobine est alimentée en continu (par ex., s'il n'y a pas d'aimant près du noyau).
Avec ce relais Finder modifié, l'électroaimant marche très bien et son capot procure même une bonne protection mécanique de l'ensemble du circuit vis-à-vis des chocs des aimants permanents sur le noyau ou le capteur. Si vous voulez vraiment le construire dans l'ancien boîtier de relais – comme je l'ai fait – c'est possible moyennant un effort mécanique acceptable et un peu de réflexion pour déterminer comment les composants peuvent être disposés dans ce volume plus petit. Il va de soi que le circuit peut aussi très facilement être construit sur une plaque d'essai ou une carte pastillée (cf. **fig. 5**), et il suffit de quatre fils pour connecter l'ensemble bobine, LED, résistance et capteur à effet Hall au reste du circuit. J'ai utilisé un morceau de câble plat de 20 cm de long dans un prototype.

Bon ingénieur ou bricoleur courageux, mais téméraire ?

Luc du labo d'Elektor a analysé de manière critique et rigoureuse que le LM311 peut absorber 50 mA à sa sortie. Mais qu'en est-il de 90 mA ? Eh bien, cela justifie le titre du paragraphe. Cela dit, mon but était de garder le circuit aussi simple que possible et de réduire le nombre de composants à un minimum absolu. Et tout simplement, le bricoleur qui m'habite refusait le transistor de commande supplémentaire pour prévenir la mort prématurée de mon circuit.
Un examen plus poussé du circuit interne du LM311 (**fig. 6**) et des diagrammes de la fiche technique révèle que la puce possède un circuit de protection interne qui limite le courant de sortie à un niveau inoffensif, même en cas de court-circuit de sortie, et assure une dissipation de puissance maximale de 350 mW sous une tension de 5 V. Cela rend la sortie presque immortelle, ou du moins, insensible à une charge de faible impédance. Pour finir, j'ai laissé mon circuit sous tension 48 h et, comme je l'attendais et l'espérais, il a survécu !
OK, prenons un peu de recul. Je respecte les limitations de vitesse, je paie mes impôts, je lis et je suis les fiches techniques… Je connais de nombreuses raisons de rester dans les limites de sécurité des spécifications des semi-conducteurs. Même si les circuits intégrés disposent de toutes sortes de circuits de protection, ce n'est pas bon de les garder aux limites. Cela réduit certainement la durée de vie des puces. On peut arguer que, dans ce cas, il durera encore assez longtemps et que changer le LM311 ne coûtera pas une fortune, mais avec cet assemblage compact, réparer le circuit est difficile.
Mieux vaut prévenir que guérir, mais je voulais aussi relever le défi de voir si le circuit peut fonctionner avec un courant de sortie plus faible. Un moyen facile et évident d'y parvenir est de remplacer le relais 5 V avec bobine de 50 Ω par un relais 6 V et bobine de 75 Ω (Finder, réf. 40.52.7.006.0000), mais je n'avais pas un tel relais sous la main.
J'ai donc opté pour une meilleure solution : tout en conservant l'alimentation de 5 V, j'ai réduit la tension aux bornes de la bobine en insérant deux diodes 1N4148 en série entre l'alimentation et le solénoïde

(cf. **fig. 7**). Le courant de la bobine sans charge utile fut réduit à 50 mA et avec charge utile à 39 mA de moyenne. Une simple résistance de 22 Ω aura le même effet. Et cela fonctionne bien aussi ! Cela permet au comparateur de rester dans ses spécifications électriques. Malheureusement, cette puissance inférieure réduit aussi la distance entre l'électroaimant et l'objet en lévitation. Pire encore, le système de commande a moins de marge pour absorber les perturbations. En d'autres termes, à ce niveau de puissance réduite, l'objet tombera ou heurtera l'électroaimant même s'il ne s'écarte que peu de la distance d'équilibre. Le circuit est moins capable de corriger l'écart.

De nouvelles idées de conception ?

Finalement, j'aime mieux cette petite version que la version analogique construite sur Veroboard pour la **Partie I**. Elle présente mieux et l'électronique est mieux protégée par le boîtier du relais. La version de la **Partie II** qui utilise un *M5Stack Atom* basé sur ESP32 a montré que le rôle du comparateur analogique pouvait être tenu par un µcontrôleur et un algorithme simple. Selon moi, les différentes variantes de ce

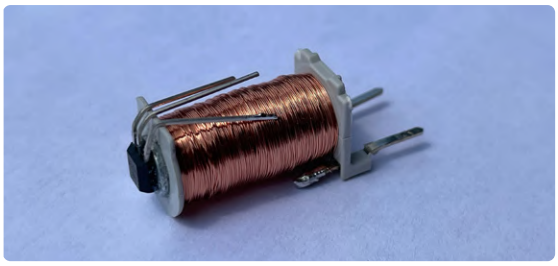


Figure 4. Le capteur à effet Hall collé au noyau de fer de la bobine.

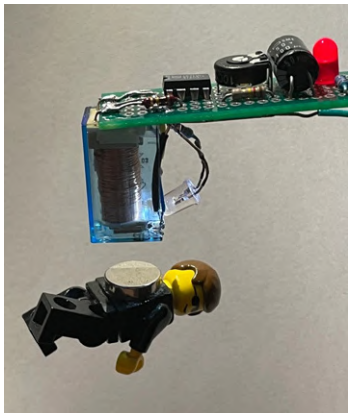


Figure 5. Un morceau de Veroboard convient aussi pour ce circuit.

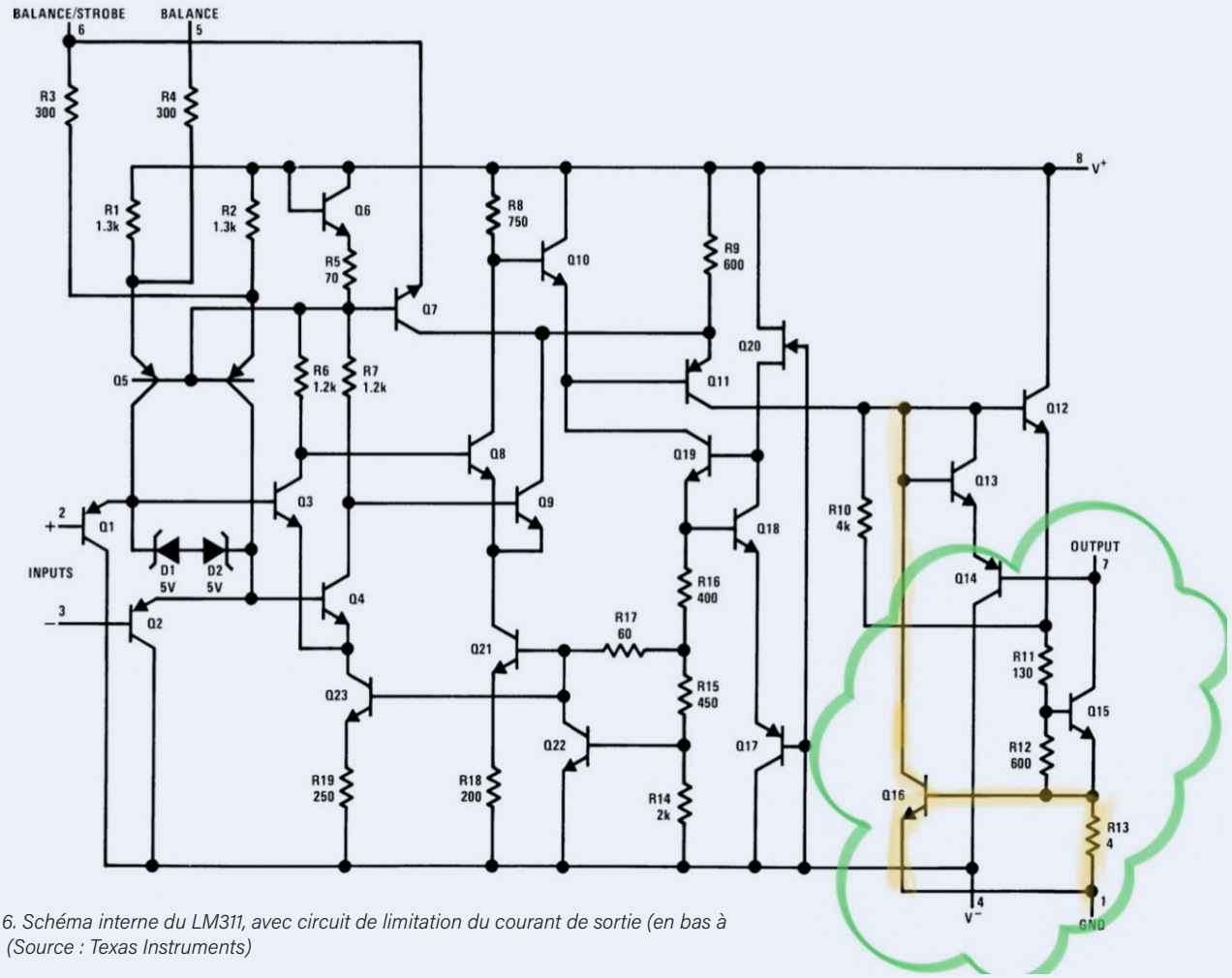


Figure 6. Schéma interne du LM311, avec circuit de limitation du courant de sortie (en bas à droite). (Source : Texas Instruments)

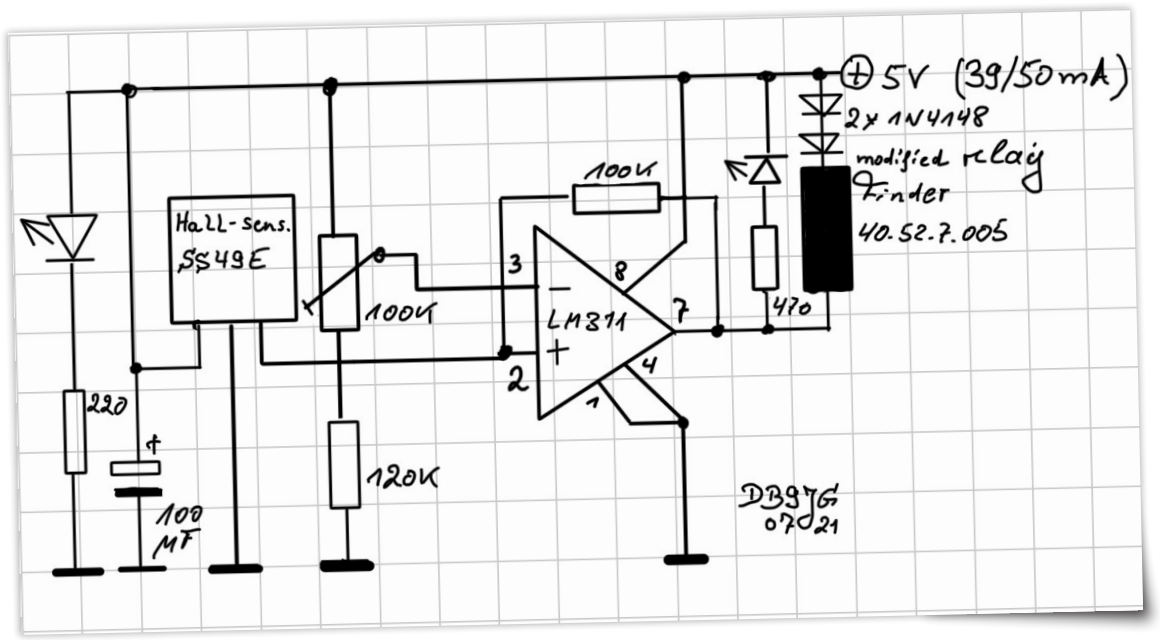


Figure 7. Le circuit de lévitation magnétique avec deux diodes 1N4148 pour limiter le courant dans la bobine.

petit projet, avec leur très classique problème de conception de l'asservissement, ont montré que l'analogique a toujours ses avantages, même à l'époque des µcontrôleurs rois !
Comme annoncé en début d'article, c'est la dernière et ultime version de mes projets de lévitation magnétique, à moins que… Peut-être avez-vous des idées ou des suggestions pour de nouvelles améliorations ?

210398-04

Des questions, des commentaires ?

Envoyez un courriel à l'auteur (luc.lemmens@elektor.com).

Contributeurs

Conception et texte : **Peter Neufeld**
Montage : **Luc Lemmens**
Illustrations : **Peter Neufeld, Patrick Wielders, Luc Lemmens**
Mise en page : **Harmen Heida**
Traduction : **Yves Georges**

LISTE DES COMPOSANTS

Résistances
R1 = 100 kΩ
R2 = 470 Ω
R3 = 220 Ω
R4 = 120 kΩ
P1 = ajustable 100 kΩ

Condensateurs
C1 = 1 µF à 100 µF, 16 V, radial

Semi-conducteurs
LED1,LED2 = LED verte
IC1 = capteur à effet Hall SS49E
IC2 = comparateur LM311

Divers
L1 = relais 5 V, Finder, réf. 40.52.7.005.0000 (voir texte)

PRODUITS

- 3e main avec loupe, éclairage LED et support de fer à souder de Velleman
  www.elektor.fr/19864
- Station de soudage à régulation de température VTSS220 de Velleman
  www.elektor.fr/19865

LIENS

[1] « Lévitation magnétique sans peine », Elektor 7-8/2021 : www.elektormagazine.fr/200311-04
[2] « Lévitation magnétique, version numérique », Elektor 9-10/2021 : www.elektormagazine.fr/200278-04
[3] Fiche technique du comparateur LM311 : www.ti.com/lit/ds/symlink/lm311-n.pdf

programmation d'automates

avec le Raspberry Pi et le projet OpenPLC

Visualisation des programmes PLC avec AdvancedHMI

Josef Bernhardt (Allemagne)

Dans les systèmes de commande industriels, ainsi qu'en domotique, il est courant de commander et d'observer le processus en cours sur des écrans. Dans ce chapitre, extrait du dernier livre de Josef Bernardt sur la programmation des automates programmables (PLC), vous vous familiariserez avec le logiciel AdvancedHMI (« Interface Homme-Machine avancée »). Vous accéderez aux variables de l'automate et essaierez de les visualiser. Il est également possible d'accéder au processus et de simuler des interrupteurs avec lui.

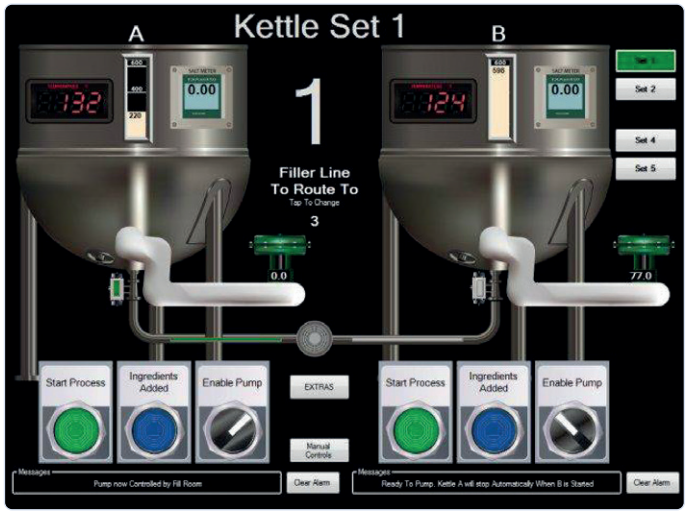


Figure 1. Exemple d'utilisation d'AdvancedHMI.

AdvancedHMI est un programme .NET écrit en Visual Basic dans l'EDI Visual Studio Community de Microsoft. Si vous installez le *Mono Framework* sur votre Raspberry Pi, vous pouvez également exécuter le logiciel sur celui-ci et avoir accès au programme PLC en cours d'exécution. Le logiciel AdvancedHMI et l'EDI Visual Studio Community peuvent tous deux être téléchargés gratuitement. La **figure 1** illustre un exemple d'AdvancedHMI en action.
Tout d'abord, téléchargez le logiciel sur le site web de l'éditeur et écrivez un programme de test pour votre Raspberry Pi utilisé comme automate. Pour le télécharger, rendez-vous sur le site web de l'éditeur, **www.advancedhmi.com**, et ajoutez le « AdvancedHMI Base Package » à votre panier, comme le montre la **figure 2**. Ensuite, cliquez sur la case *Check out*.

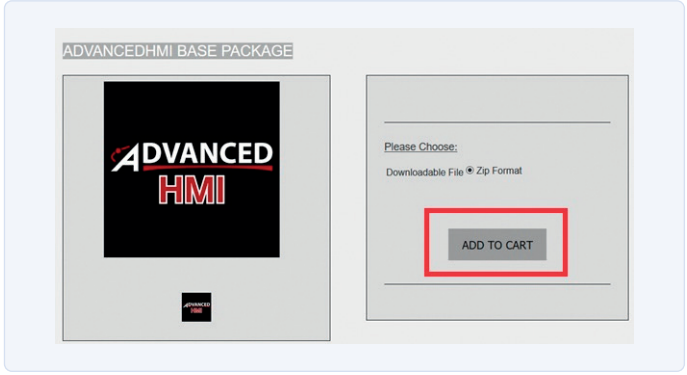


Figure 2. Téléchargement d'AdvancedHMI.

Note de l'éditeur : cet article est un extrait du livre *PLC Programming and the OpenPLC Project - ModbusRTU and ModbusTCP examples using the Arduino Uno and the ESP8266* formaté et légèrement modifié pour correspondre aux normes éditoriales et à la mise en page du magazine *Elektor*. Puisque cet article est extrait d'une publication plus vaste, certains termes peuvent faire référence à des passages du livre d'origine situés ailleurs. L'auteur et l'éditeur ont fait de leur mieux pour l'éviter et seront heureux de répondre aux questions – pour les contacter, voir l'encadré « Des questions, des commentaires ? ».

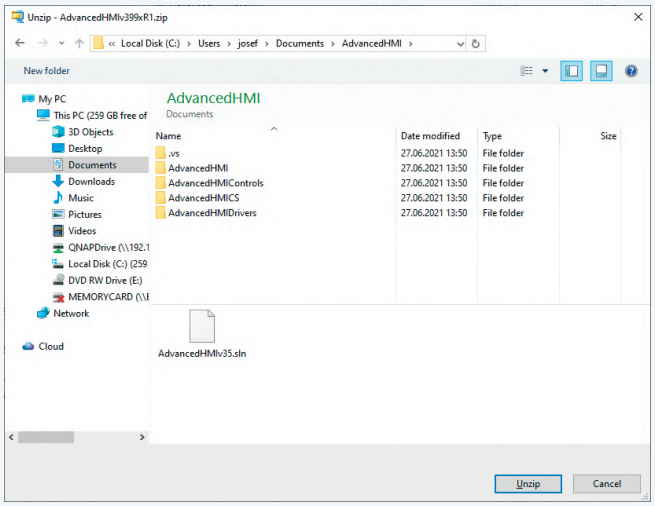


Figure 3. Décompression d'AdvancedHMI.

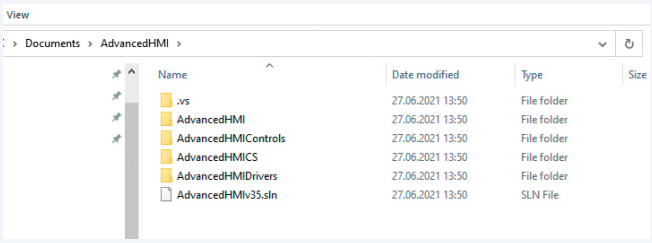


Figure 4. Le répertoire du projet AdvancedHMI.

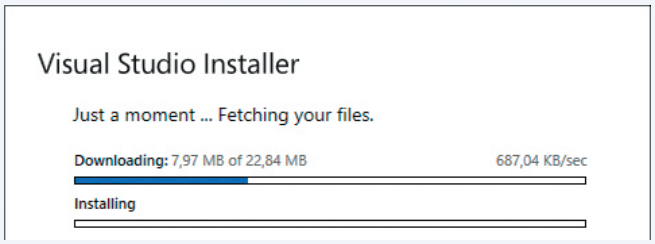


Figure 5. Le programme d'installation de Visual Studio.

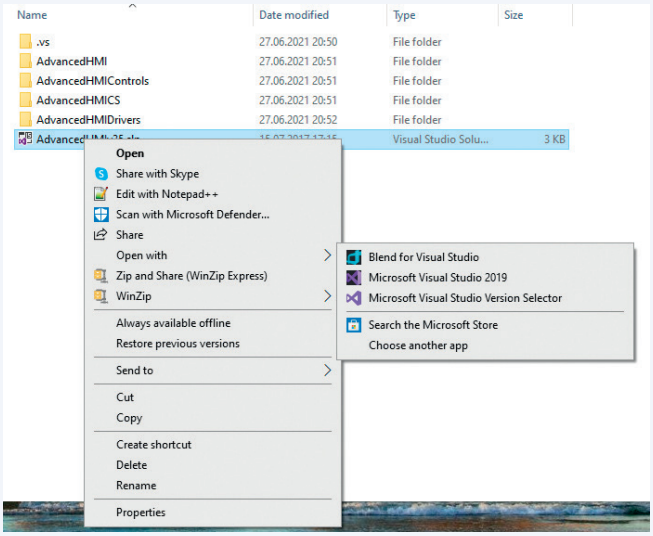


Figure 6. Ouvrez votre projet AdvancedHMI.

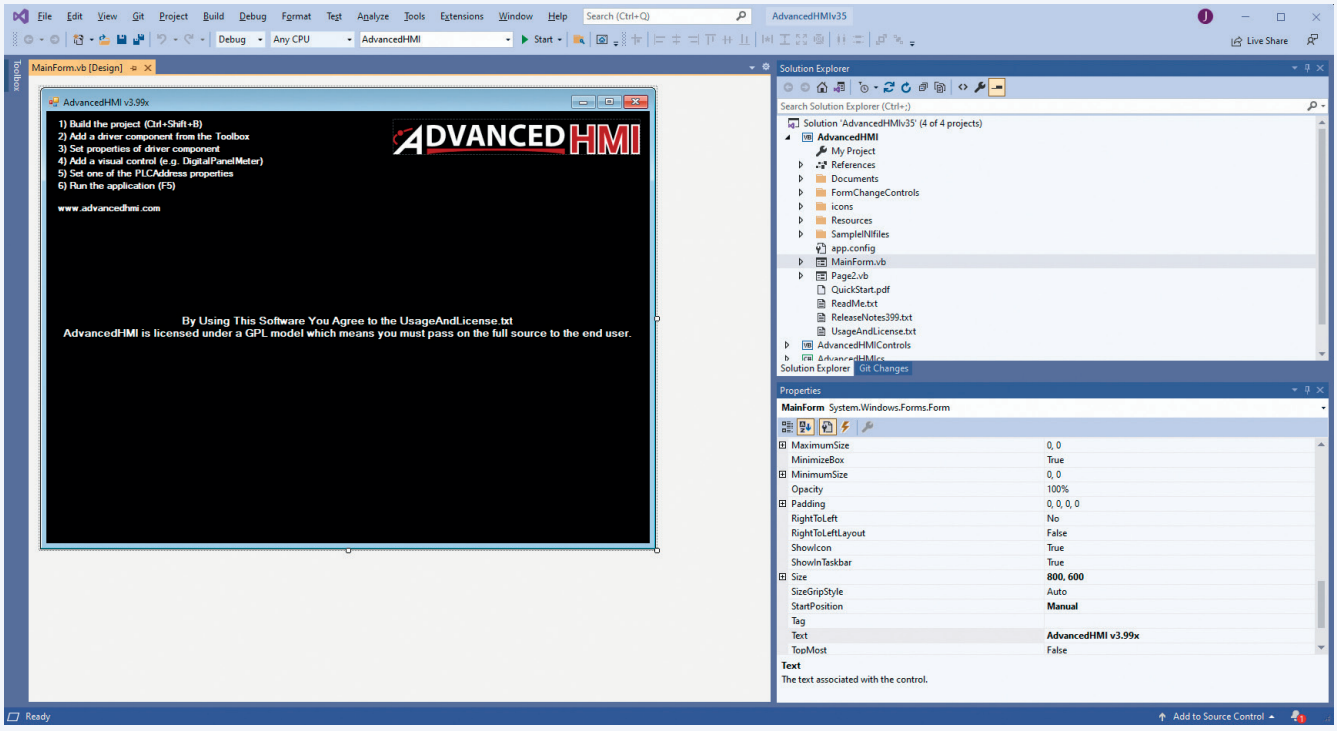


Figure 7. L'interface utilisateur de Visual Studio.

Préface du livre

Ce livre a pour but de fournir aux lecteurs une introduction pratique à l'utilisation de l'ordinateur Raspberry Pi comme automate programmable (PLC) dans leurs projets. Le projet doit beaucoup aux programmeurs Edouard Tisserant et Mario de Sousa, qui ont lancé le « projet Matiec » après la publication de la norme CEI 61131-3 en 2003, ce qui a rendu possible la traduction des langages de programmation définis dans cette norme en programmes en C. Plus tard, lorsque le Raspberry Pi est devenu de plus en plus populaire, Thiago Alves a lancé le projet « openplcproject ». Il a étendu l'éditeur du projet « Beremiz » et a écrit une bibliothèque d'exécution et une interface web pour le Raspberry Pi et le PC. Dès lors, il devint possible d'écrire des programmes sur le PC et de les installer sur le Raspberry Pi.
De nombreux utilisateurs de Raspberry Pi sont désormais en mesure de réaliser leurs propres systèmes de commande et de régulation en utilisant leur propre matériel. Le matériel et le logiciel sont également excellents à des fins de formation, car ils respectent la norme CEI.
Les débutants apprendront également tout sur l'installation et la programmation dans les cinq langages de programmation afin de construire leurs propres systèmes de commande. Un chapitre aborde le suivi des processus à l'écran avec AdvancedHMI. Sont également expliqués les circuits avec l'Arduino et l'ESP8266, nécessaires pour le Modbus.

Après vous être enregistré, téléchargez le programme gratuitement et décompressez-le dans un répertoire (**fig. 3**). Enregistrez le fichier ZIP comme modèle.
Ensuite, créez un répertoire *AdvancedHMI* et copiez-y le contenu du fichier *AdvancedHMIv399xR1.ZIP* (**fig. 4**). Pour ouvrir le projet, un environnement de développement est nécessaire, alors installez le logiciel *Visual Studio Community* à partir du site web de Microsoft, **https://visualstudio.microsoft.com/vs/community/**, comme le montre la **figure 5**.
Une fois le processus d'installation terminé, démarrons le premier projet de démonstration. Avec un clic droit de la souris, nous ouvrons la solution *AdvancedHMI* avec Microsoft Visual Studio 2019 (**figures 6** et **7**). En double-cliquant sur *MainForm.vb*, vous obtenez la vue HMI encore vide de votre projet de démonstration (WinForm). Vous pouvez maintenant exécuter le programme en cliquant sur *Start* (**fig. 8**). À ce stade, le projet a été compilé et enregistré dans le répertoire indiqué à la **figure 9**.
Ces fichiers de projet sont nécessaires pour utiliser un programme sur un autre ordinateur. Vous pouvez également exécuter ces fichiers sur le Raspberry Pi avec le Mono Framework.
Maintenant, faites glisser et déposez le composant *ModbusTCPCom* de la boîte à outils de gauche (**fig. 10**) dans votre espace de travail. Dans la fenêtre *Properties*, vous pouvez maintenant saisir l'adresse IP de votre Raspberry Pi (**fig. 11**).
Pour tester la communication, créez un nouveau programme PLC avec l'éditeur Open PLC (**fig. 12**), chargez-le sur le Raspberry Pi et démarrez-le. Il s'agit d'un simple circuit On/Off utilisant les boutons *On* et *Off* (**fig. 13**).
La situation devient intéressante avec le bouton *BTN_HMI*, qui se trouve à l'adresse *%QX2.0*. Cette adresse, située en dehors de la plage d'adresses des GPIO du Raspberry Pi, est nécessaire pour que l'interface HMI déclenche une pression sur un bouton. Avec les temporisations *TON0* et *TOF0*, vous pouvez réaliser un circuit de LED clignotante pour le programme *LEDBlink*.
Revenez maintenant à votre projet HMI et faites glisser un interrupteur (*MomentaryButton*) et trois voyants lumineux (*PilotLight*) depuis la boîte à outils, et étiquetez-les comme vous pouvez le voir dans le résultat final, **figure 14**. Pour ce faire, sélectionnez l'objet et modifiez le texte dans la fenêtre *Properties* comme indiqué.
Définissez les champs *PLCAdressClick* comme suit :

- **BTN_HMI** est à l'adresse 17, parce que vous avez utilisé *%QX2.0* pour le *BTN_HMI* (8+8+1).
- La **LED** va à **1**
- **LEDModbus** va à **2**
- **LEDBlink** va à **3**

Le *MomentaryButton* peut être changé de bouton en interrupteur dans les propriétés de *OutputType* comme le montre la **figure 15**. Cela fait, vous disposez des adresses correctes pour avoir accès aux variables de votre programme PLC. Vous pouvez maintenant lancer votre programme et effectuer un test.
Les trois LED sont toujours dans le même état que les LED de la carte de test. Le commutateur *BTN_HMI* vous permet d'allumer la LED 2 de la carte de test.
Le **tableau 1** montre le mappage des adresses Modbus pour le Raspberry Pi avec OpenPLCproject et AdvancedHMI. OpenPLC fournit également un espace d'adressage séparé pour les variables en mémoire avec un support pour les variables de 16, 32 et 64 bits, comme résumé dans le **tableau 2**.
Pour installer Framework Mono sur le Raspberry PI, il faut saisir les commandes suivantes :

```
sudo su
apt-get update
apt-get install mono-complete
apt-get install mono-vbnc
```

Vous pouvez maintenant créer un répertoire sur le Raspberry Pi et utiliser *WinSCP* pour copier votre projet du PC vers le Raspberry Pi et le tester.

Function Code	Usage	PLC Address	Modbus Register	Register Size	Value Range	Access	Advanced HMI	Example
FC01 Read Coil	Digital Outputs	%QX0.0 - %QX99.7	0-799 (1-800)	1 Bit	0 or 1	Read / Write	"00001" – "00800"	LED
FC02 Discrete Input	Digital Inputs	%IX0.0 – %IX99.7	0-799 (1-800)	1 Bit	0 or 1	Read Only	"100001" – "100800"	BTN
FC04 Input Register	Analog Inputs	%IW0 – %IW99	0-1023 (1-1024)	16 Bit	0-65535	Read Only	"30001" – "31024"	ADC
FC03 Holding Register	Analog Outputs	%QW0 – %QW99	0-1023 (1-1024)	16 Bit	0-65535	Read / Write	"40001" – "41024"	DAC

Tableau 1

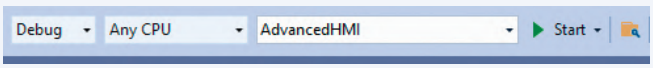


Figure 8. La barre de menu de Visual Studio.

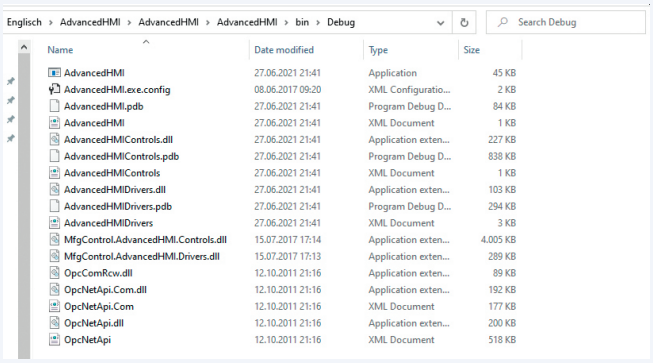


Figure 9. Le répertoire de débogage d'AdvancedHMI.

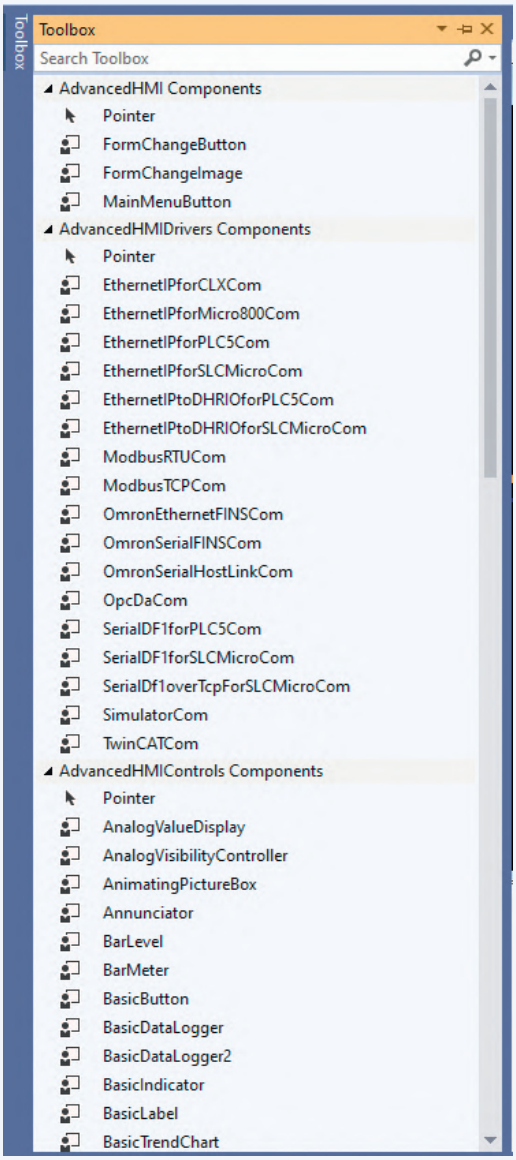


Figure 10. La boîte à outils de Visual Studio.

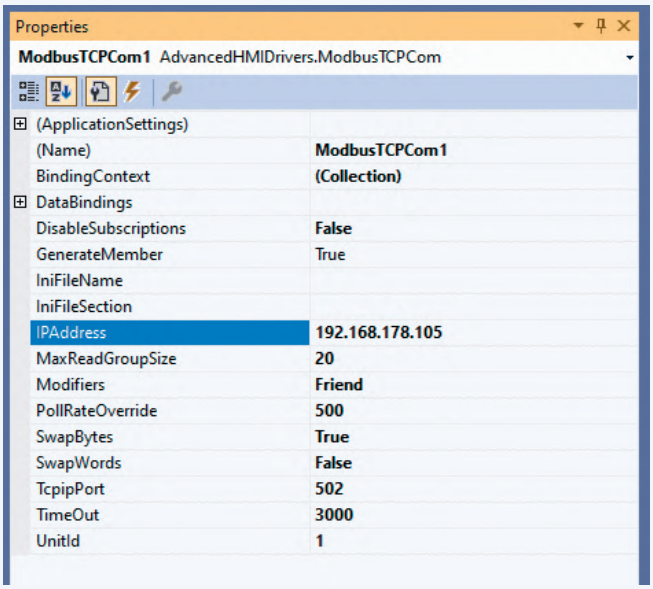


Figure 11. Paramètres Modbus (IP et port).

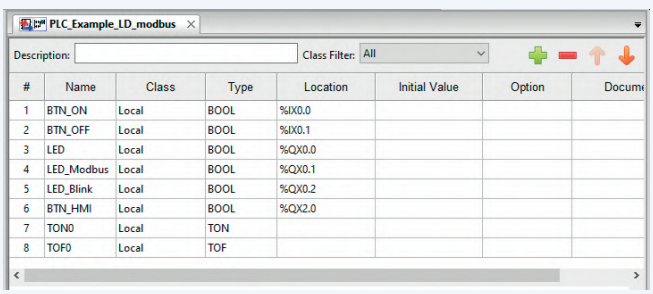


Figure 12. Exemple de liste de variables de l'API.

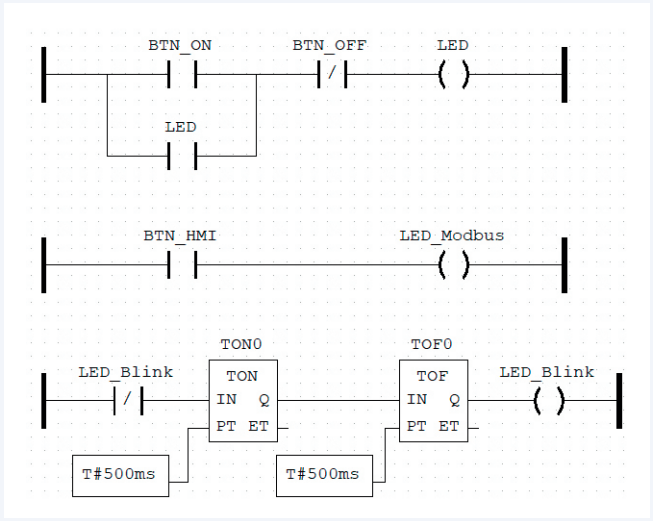


Figure 13. Exemple de programme de LED.

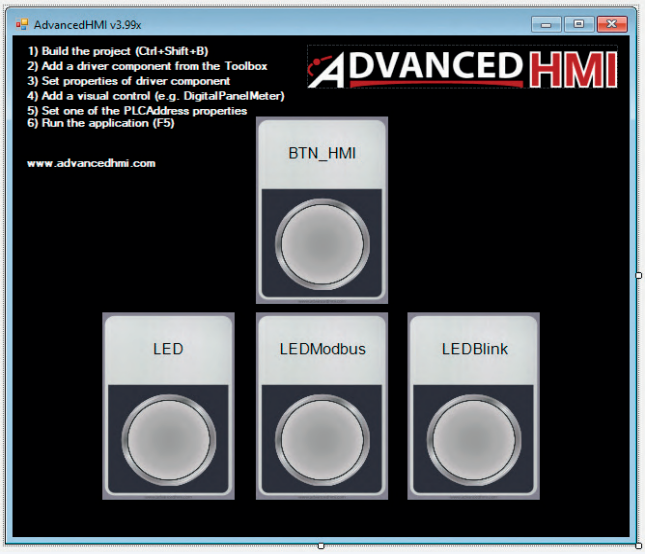


Figure 14. Le programme HMI.

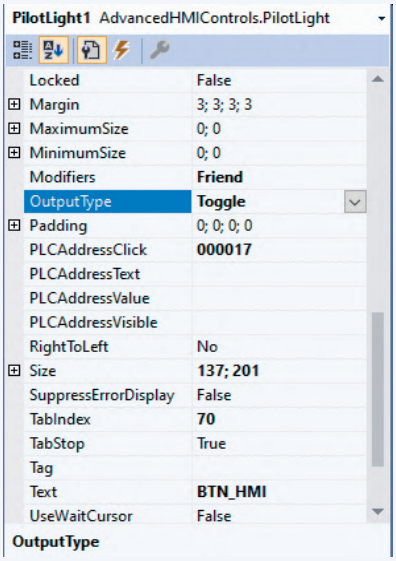


Figure 15. Réglage des paramètres de l'HMI.

L'exemple de programme est aussi disponible dans le répertoire de téléchargement sous *AdvancedHMI*. Les fichiers *.EXE* des exemples traités dans le livre se trouvent dans le sous-répertoire suivant :

`PLC-Book-Download\AdvancedHMI\AdvancedHMI\bin\Debug`

Ce sous-répertoire est contenu dans le paquet de logiciels publié par l'auteur comme ressource de son livre. Le logiciel peut être téléchargé gratuitement. Rendez-vous sur [1], allez à *Téléchargements* et cliquez sur le nom du fichier : *Software_PLC Programming with the Raspberry Pi and the OpenPLC Project*. Enregistrez le fichier d'archive ZIP (128,94 Mo) en local, puis décompressez-le.

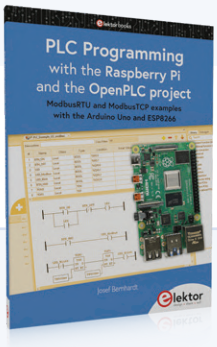
210642-04

Des questions, des commentaires ?

Envoyez un courriel à l'auteur (josef@bernhardt.de) ou contactez Elektor (redaction@elektor.fr).

Contributeurs

Texte : **Josef Bernhardt**
Rédaction : **Jan Buiting**
Mise en page : **Giel Dols**
Traduction : **Helmut Müller**



PRODUITS

- **Livre en anglais, « PLC Programming with the Raspberry Pi and the OpenPLC Project », J. Bernhard**

  Version papier : www.elektor.fr/19966

  Version numérique : www.elektor.fr/19967

Register Type	Usage	PLC Address Range	Modbus Address	Register Size	Value Range	Access	Advanced HMI
Holding Register	General 16Bit Register	%MW0 - %MW1023	1024..2047 (1025-2048)	16 Bit	0-65535	Read / Write	"41025" – "42048"
Holding Register	General 32Bit Register	%MD0 - %MD1023	2048..4095 (2049-4096)	32 Bit	0-4.294.967.295	Read / Write	"42049" – "44096"
Holding Register	General 64Bit Register	%ML0 - %ML1023	4096..8191 (4097-8192)	64 Bit	0-huge	Read / Write	"44097" – "48192"

Tableau 2

LIEN

[1] Ressources du livre/page d'information : www.elektor.fr/plc-programming-with-the-raspberry-pi-and-the-openplc-project

sur le vif

Emballé, c'est pesé

Ilse Joostens (Belgium)

Ainsi va la vie, il ne se passe pas une seule journée sans que nous ne soyons enquiquinés par la plus insignifiante des choses. Prenez par exemple ces produits vendus sous blister, autrement dit sous une coque en plastique fixée sur un support en carton. Pour ouvrir ces satanés trucs, il vous faut des ciseaux ou un couteau, et quoi qu'il en soit un certain doigté. Car si ce n'est pas votre jour, il y a de fortes chances que durant l'opération, non seulement les bords acérés du plastique vous cisaillent un doigt, mais qu'en plus vous coupiez ou abîmiez le produit. Et je ne parle pas de ces emballages assurément faciles à ouvrir, mais qui semblent avoir été spécialement conçus pour vous empêcher de remettre le produit à l'intérieur, surtout le jour où vous souhaitez le retourner. Bienvenue dans le monde merveilleux de l'emballage !

Tétrismatique inverse

Je ne sais pas si vous vous en souvenez, mais dans l'article sur l'*Horloge de sable* paru en janvier 2017 [1], j'avais expliqué le principe de la cinématique inverse. Peu importe les détails de ce principe à vrai dire, même si y jeter un œil ne nuira pas à votre élévation intellectuelle. À cette époque, donc, je faisais mes courses dans le supermarché de notre quartier. J'étais avec ma meilleure moitié, et nous nous amusions à organiser nos achats sur le tapis de caisse de façon à pouvoir les glisser au plus vite dans nos sacs, les plus lourds au fond. Comme nous aimons les jeux de mots improbables, nous avions alors appelé cette tactique la « tétrismatique inverse ».

Mais revenons à l'électronique. Ce que nous préférons avec elle, ce sont les schémas, les composants, les cartes, les modules, les instruments de mesure, etc. L'aspect mécanique d'un projet, par exemple la fabrication d'un boîtier ou d'un panneau de commande, nous laisse souvent beaucoup moins enthousiastes, voire entraîne chez nous des sueurs froides. Car aux yeux de l'électronicien, passer des heures à limer un morceau de métal n'est pas de prime abord l'activité la plus intéressante du monde. Ceci dit, la mécanique aussi peut nous apporter son lot de menus plaisirs. Mais l'emballage d'un produit ? Qui diable a déjà frémi d'excitation devant une boîte en carton et un rouleau d'adhésif ? Sans doute certains Youtubeurs, comme Marco Reps [2] ou Dave Jones d'EEVBlog [3] qui, de leur propre aveu, gagnent leur vie en ouvrant des emballages à l'aide d'un couteau ridiculement long. Je dois d'ailleurs reconnaître que la vidéo de Marco Reps ouvrant un kit d'horloge Nixie m'a profondément touchée – j'avais mis énormément de soin et d'amour dans l'emballage de ces kits.

Emballer un produit fini semble simple, pourtant rien n'est plus trompeur. Choisir l'emballage d'un produit nécessite une bonne dose de réflexion. Doit-il être attrayant, ou peut-on négliger son aspect visuel ? Quel type de boîte utiliser ? De quelles dimensions ? En quelle matière ? Quel coût est acceptable ? De toute évidence, la boîte – notamment dans le cas d'un produit fragile – ne doit pas être trop petite, mais pas trop grande non plus pour éviter un surplus de matériau de remplissage ou de protection. Le produit ne doit pas être ballotté dans la boîte lorsque celle-ci est manipulée, car bien sûr cela pourrait l'endommager, mais aussi parce qu'il est hors de question que le client puisse deviner le contenu de la boîte en la secouant – ce que les enfants adorent faire, mais aussi certains adultes à Noël [4]. Pour ce qui est des produits électroniques, le recours à un matériau d'emballage antistatique peut être nécessaire. Lorsque le produit est composé

de plusieurs pièces, une bonne idée est de l'emballer de telle sorte qu'à l'ouverture le client trouve les pièces dans le bon ordre de montage – autre illustration du principe de la tétrismatique inverse.

Le choix du matériau d'emballage peut être dicté par des facteurs moins évidents. Lorsque j'ai demandé à ce que le sable de l'*Horloge de sable* soit mis dans un sachet antistatique, certains sourcils se sont dressés : personne n'avait songé qu'en raison de l'attraction électrostatique le sable fin et sec se collerait aux parois d'un sachet en plastique ordinaire, ce qui allait rendre son scellage difficile.

Un problème de poids

L'emballage des kits électroniques comprenant un grand nombre d'éléments de petite taille nécessite une attention particulière. Une bonne approche est de dresser une liste du contenu du kit, d'en trier les pièces par type et par fonction, de mettre ces différentes catégories dans des sachets, et enfin de mettre le tout dans un plus grand sachet. Cette méthode offre une meilleure vue d'ensemble, et surtout réduit considérablement les risques d'erreurs auxquels vous vous exposeriez si vous jetiez tout en vrac dans la boîte. Pour le contrôle final, une pesée des sachets à l'aide d'une balance de précision montre immédiatement s'il manque un élément ou s'il y en a un en trop.

La panoplie de balances et sachets dont nous disposons ferait pâlir d'envie n'importe quel Panoramix de la fumette, sans parler de la table d'emballage, des dévidoirs de ruban adhésif, du matériel de scellage, des films tubulaires, des récipients vides et de nos rayonnages industriels, toutes choses bien pratiques par ailleurs.

Ce bon vieux et fidèle cheval qu'est notre service postal est aussi une source de préoccupations. Car allez savoir pourquoi, à partir d'un certain poids le coût d'envoi d'un colis double soudainement. Tout l'art de l'emballeur professionnel consiste donc à ne pas dépasser cette limite satanique, et pour ma part, malgré une expérience somme toute conséquente en ce domaine, je n'entre jamais tout à fait sereine dans un bureau de poste. On peut dire que là encore je pratique la tétrismatique inverse, mais ici sa pratique s'avère parfois épuisante.

Depuis quelque temps nous pouvons heureusement compter sur les services d'Alibaba et de ses consorts d'Extrême-Orient. Grâce à leurs pratiques concurrentielles effrénées et autres activités économiques douteuses, nous n'avons plus besoin d'emballer nos kits. Disons que cela m'épargne pas mal de boulot. Et rien que pour ça, j'ai envie de leur dire merci.

210625-04

Contributeurs

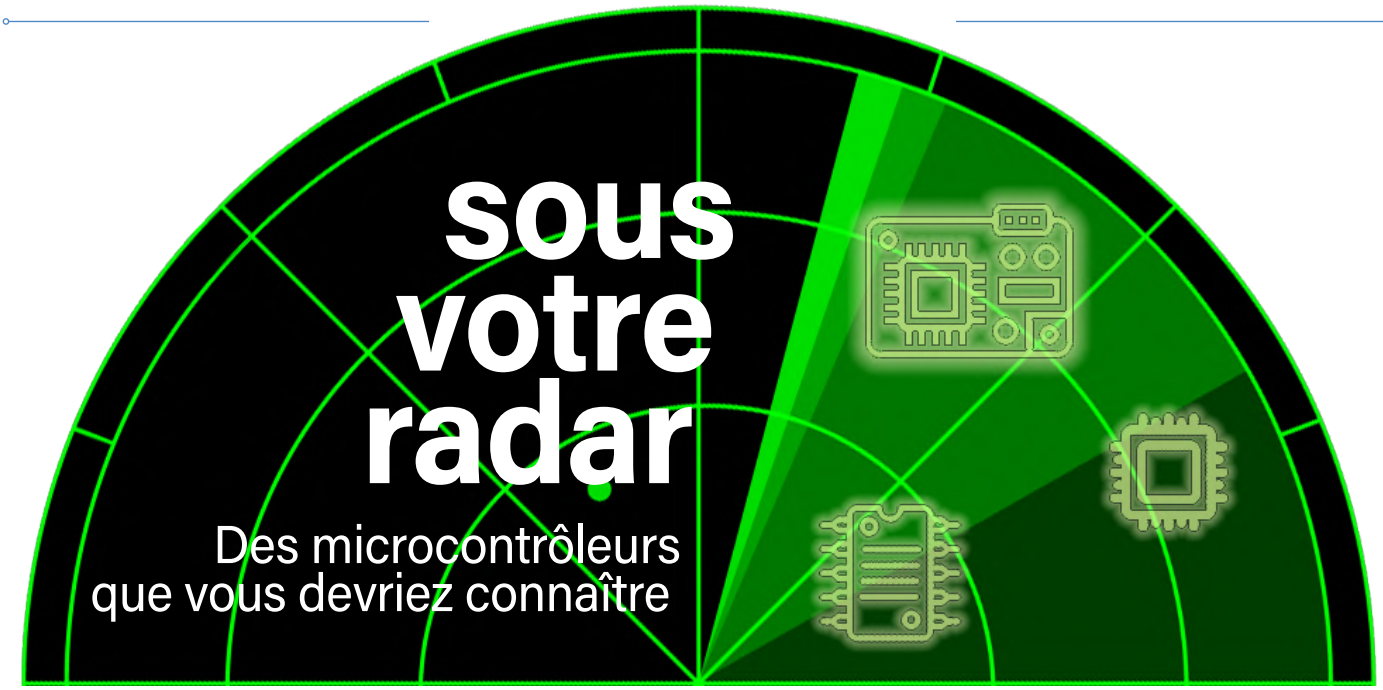
Texte : **Ilse Joostens**
Rédaction : **Eric Bogers**
Mise en page : **Harmen Heida**
Traduction : **Hervé Moreau**

Des questions, des commentaires ?

Contactez Elektor (redaction@elektor.fr).

LIENS

[1] Ilse Joostens, « Horloge de sable », Elektor, 01-02/2017 : www.elektormagazine.fr/magazine/elektor-201701/40071
[2] Marco Reps, horloge Elektor à tubes Nixie : https://youtu.be/ge_9CNiZZ_A?t=25
[3] Dave Jones, EEVBlog, horloge Elektor IV-22 à VFD : https://youtu.be/SyXiWNZs7l4?t=1733
[4] Une farce à l'intention de ceux qui secouent leurs paquets-cadeaux : www.youtube.com/watch?v=ZeCjiEiPqAM



sous votre radar

Des microcontrôleurs que vous devriez connaître

Clemens Valens (Elektor)

Le marché mondial des microcontrôleurs est plus diversifié qu'on ne le pense. Passons en revue quelques-uns des microcontrôleurs et fabricants pas souvent apparus dans *Elektor*. Vous pourriez en trouver un ou plusieurs utiles pour un futur projet.

Pour un projet Elektor, le choix d'un microcontrôleur dépend surtout de la disponibilité à bas prix d'outils de développement du logiciel et de programmation de la puce, et de la possibilité pour les particuliers de les acquérir. Pour de nombreux amateurs d'électronique, cela restreint leur vision du marché mondial des microcontrôleurs. Le monde des microprocesseurs va bien au-delà des PIC, AVR, ARM ou ESP, si souvent au cœur de nos projets maison. Jetons un coup d'œil à quelques-uns de ces MCU qui échappent à notre perception.

Tout a commencé avec quatre bits

Introduit en 1971, l'Intel 4004 est considéré comme le premier microprocesseur produit commercialement (vous trouverez plus d'informations à son sujet dans le numéro spécial d'*Elektor Industry* consacré à 60 ans d'électronique [1]). Il s'agissait d'un composant à 4 bits. Avec ses puces périphériques, il formait la famille MCS-4. Elle a été suivie par la famille MCS-40 avec l'unité centrale 4040. Le premier microcontrôleur (pas un microprocesseur) de Texas Instruments, le TMS1000 de 1974, avait également une architecture à 4 bits qui, comme le 4040, a animé de nombreuses calculatrices de poche. Dans un monde où les fabricants de microcontrôleurs semblent vouloir allonger autant que possible les mots de données (64 bits est devenu banal), vous seriez surpris du nombre de microcontrôleurs à 4 bits qui sont encore utilisés aujourd'hui. Mais pourquoi ? La réponse est probablement un mélange d'héritage, de consommation d'énergie et de coût.

Un MCU à 4 bits comprend moins de transistors que les puces avec des mots plus longs. Il consomme donc comparativement moins d'énergie, un bonus pour la durée de vie des piles. Moins de transistors signifie également moins d'espace, ce qui permet de caser un cœur de 4 bits dans un coin de la puce, là où un cœur plus grand ne tiendrait pas. La taille de la puce elle-même est plus petite, ce qui permet de réduire les coûts (dans quelle mesure peut-on se demander). Les applications en grande série telles que les calculatrices, les minuteries, les horloges et les montres, les ordinateurs de bicyclette, les jouets et les télécommandes, etc. utilisent des MCU à 4 bits et ce depuis de nombreuses années. Les fabricants évitent généralement de modifier les produits qui marchent, ce qui explique la persistance sur le marché de ces composants.

Si vous désirez essayer un microcontrôleur à 4 bits, jetez un coup d'œil aux familles NY... de la société taiwanaise Nyquest. Les outils de développement peuvent être téléchargés gratuitement (**fig. 1**). Autres fabricants : le Suisse EM Microelectronic, le Chinois CR Micro, le Taiwanais Tenx Technology.

8051

Avant qu'ARM ne devienne le principal fournisseur de cœurs de microcontrôleurs pour presque tous les fabricants de semi-conducteurs de la planète, il y avait le 8051 à 8 bits. Créé par Intel en 1980 sous le nom de MCS-51, son cœur (**fig. 2**) a été cédé sous licence à plusieurs concurrents et a été intégré dans une foule de produits, dont beaucoup, avec de nombreuses variantes, sont encore fabriqués aujourd'hui. Quarante ans après l'introduction du 8051, de nouveaux produits sont toujours conçus à partir de ses dérivés.

De plus, les utilisateurs du 8051 ont développé une quantité de logiciels et de savoir-faire au cours de cette période, que l'arrivée d'un meilleur microcontrôleur n'a pas rendus caducs.

Enfin, le cœur du 8051 est devenu quasiment gratuit, ce qui en fait une option intéressante pour les fabricants de semi-conducteurs tentés par la production de composants à très bas coût. Ils ne le mentionnent pas toujours dans la fiche technique, mais s'il est indiqué quelque chose comme « 1T instruction cycle », il y a fort à parier qu'il s'agit d'un dérivé du 8051. Le 8051 original consommait 12 cycles d'horloge (appelés « 12T ») pour la plupart des instructions, alors que les plus modernes n'en nécessitent qu'un seul (d'où le « 1T »). En plus de demander moins de cycles d'horloge par instruction, l'exécution du programme est également (beaucoup) plus rapide car certains de ces composants modernes fonctionnent à des fréquences allant jusqu'à 450 MHz au lieu des 12 MHz de l'original. Les MCU 8051 modernes sont donc à la fois puissants et bon marché.

Aujourd'hui, le principal inconvénient du 8051 est sans doute son incompatibilité avec les langages de programmation modernes comme le C/C++, en raison de son étrange structure de mémoire. Pour en tirer le meilleur parti, il faut une chaîne d'outils commerciale Keil ou IAR, ou équivalente. La populaire chaîne d'outils GCC a été portée sur toutes sortes de microcontrôleurs, mais pas sur le 8051. La chaîne d'outils libre SDCC fait une bonne partie du travail, mais c'est loin d'être parfait. Bien sûr, on peut toujours utiliser l'assembleur. Vous préférez le Pascal ? Voyez le Turbo51 [2].

Vous pouvez trouver des MCU basés sur le 8051 dans des produits bon marché de grande série tels que des jouets, des claviers et des souris de PC, des brosses à dents, des appareils ménagers, des télécommandes, etc., surtout originaires d'Asie où le 8051 semble très populaire. Silan, SiGma Micro, SinoWealth, Silicon Laboratories, Sonix, STC et SyncMOS (pour ne citer que ceux avec un « S » initial) en fabriquent tous.

Si vous êtes tenté d'expérimenter avec une variante moderne du 8051, jetez un coup d'œil à la famille CH55x de WCH. Pas chers, mais seulement documentés en chinois, ils

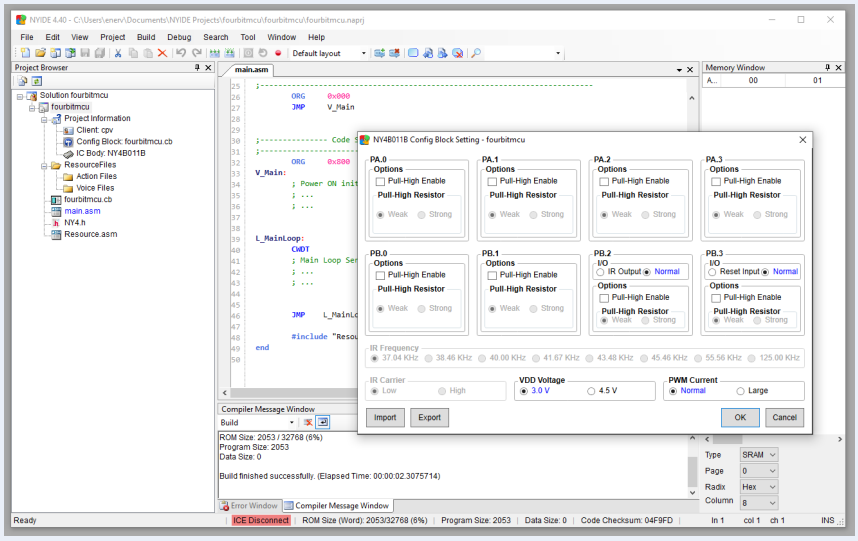


Figure 1. Le NYIDE 4.40 de Nyquest montre que les microcontrôleurs à 4 bits peuvent aussi avoir des EDI modernes.

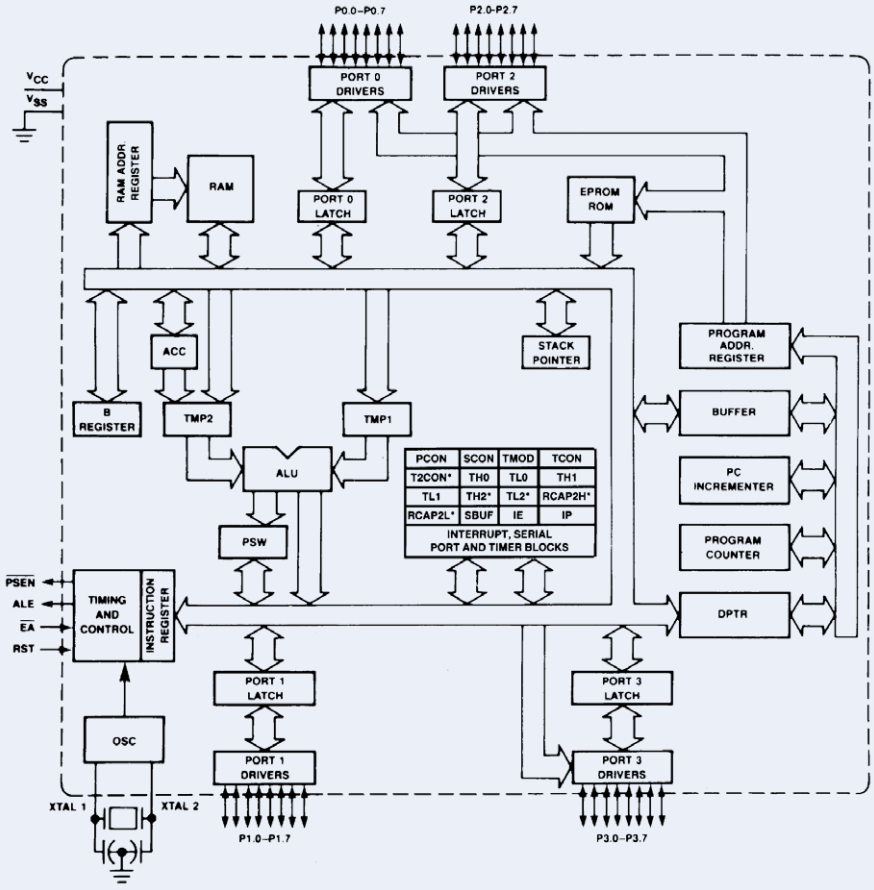


Figure 2. Voici l'architecture de base de ce qui pourrait bien être le cœur de microcontrôleur le plus utilisé au monde, le 8051. (Source : Intel)

Figure 3. Le Commodore 64, l'un des célèbres ordinateurs de salon des années 1980, contient un 6502. (Source : ralfsfotoseite @ Pixabay)

disposent d'une interface USB qui facilite leur programmation et sont utilisés dans des projets ouverts. En faire un Arduino ? Voir par exemple [3].

Vieux de la vieille et survivants

À côté du 8051 de 1980, il existe encore quelques autres cœurs de processeurs de cette période, notamment le 6502 et le Z80, mais aussi le 68000 de Motorola (maintenant NXP), un peu plus récent. Le Z80 et la version CMOS du 6502, le W65C02, sont toujours fabriqués et activement supportés par leurs créateurs Zilog et WDC (WDC a créé le W65C02, pas le 6502). La survie du 68000 semble surtout due aux besoins de maintenance de vieilles applications (mais qui sait combien de sociétés en ont pris une licence ?).

6502

Le 6502 a été utilisé dans plusieurs des premiers ordinateurs célèbres comme le Commodore 64, l'Apple II et le BBC Micro, et a connu un grand succès (**fig. 3**). Sa version CMOS améliorée et à faible consommation est toujours d'actualité, bien qu'assez chère. La raison en est que la plupart des utilisateurs ne prennent une licence que pour l'utiliser dans des FPGA, des ASIC et similaires. Comme tout est confidentiel, il est difficile de savoir de quels composants il s'agit.
Cependant, WDC produit des puces embarquées que vous pouvez essayer. Un exemple est le microcontrôleur W65C265S avec un CPU W65C816S à 16 bits qui est entièrement compatible avec le W65C02S à 8 bits et qui fonctionne à partir de 1,8 V seulement. Il existe aussi des modules à contrôleur, et même une carte sœur avec des connecteurs Seeed Studio Grove, Sparkfun QWIIC et MikroE Click.

Z80

Le Z80 est un autre processeur à succès de la fin des années 70 et du début des années 80. Zilog en a développé le cœur en 1975 et il est toujours en production. Sa licence a été accordée à de nombreux fabricants qui l'ont copié et cloné dans le monde entier, ce qui a donné naissance à une énorme base d'utilisateurs. Plusieurs familles ont vu le jour, comme l'eZ80 qui fonctionne avec des horloges allant jusqu'à 50 MHz, ou les Z8 et eZ8 Encore!, qu'on retrouve notamment dans la ligne de produits ZMOTION, une famille de MCU optimisés pour la détection de mouvement PIR.
Si vous voulez mettre les mains dans le cambouis, essayez le Z8FS040BSB avec ses 4 Ko de mémoire flash et ses cinq broches GPIO dans un boîtier SOIC à 8 pattes. Notez que le compilateur gratuit SDCC supporte plusieurs MCU basés sur le Z80, comme ceux de Rabbit (maintenant Digi) et de la Nintendo Gameboy.

Composants à très faible coût

De nombreux produits électroniques contenant des microcontrôleurs sont fabriqués en très grande série. Pensez aux appareils électroménagers, aux horloges, aux brosses à dents électriques, aux e-cigarettes, aux testeurs de coronavirus, aux cartes à puce, aux détecteurs de fumée, aux jouets, etc. (**fig. 4**). Pour avoir une idée des chiffres : les consoles de jeu comme la Gameboy, la Wii et la Switch de Nintendo ont toutes dépassé les 100 millions d'unités vendues. Imaginez ce que sont les chiffres pour les cartes à puce, par exemple. Économiser un centime sur le coût d'un tel produit est énorme et il existe donc un grand marché pour les microcontrôleurs ultras bon marché.
Parmi leurs fabricants, certains ont attiré l'attention des amateurs d'électronique, dont Padauk, MDT et Holtek.

Padauk

Padauk fabrique des MCU à trois centimes, à programmation unique (OTP) et à base de flash, dont la particularité est l'architecture

Figure 4. Quelques exemples d'applications rendues possibles par les microcontrôleurs à très bas coût. (Source : Holtek.com)

basée FPP (*Field-Programmable Processing units*). Il s'agit de bancs de registres avec un compteur de programme, un pointeur de pile, un accumulateur et un registre de drapeaux, qui permettent un changement de contexte rapide, bien utile, par exemple pour le traitement des interruptions et le multitâche. Cela rappelle un peu le banc de quatre registres du 8051. Cependant, comme la plupart de leurs produits n'ont qu'un seul FPP (certains en ont deux, le PFC460, quatre et le MCS11, huit), ce ne sont que des MCU de base.
Le PMS150 est un bon point de départ. On trouvera une excellente présentation de ces composants sous [4]. SDCC supporte les PDK14 et PDK15. Pour le PDK13, c'est en cours.

MDT

Comme déjà mentionné, MDT, alias Micon Design Technology, fabrique des clones ou des dérivés des PIC de Microchip. La popularité des PIC chez les faiseurs a attiré une certaine attention sur les produits MDT. Cependant, au cours de mes recherches pour cet article, leur site web a soudain disparu. Une recherche sur l'internet concernant les composants MDT produit plusieurs résultats pour le craquage de MCU et les services de rétro-ingénierie des produits MDT, entre autres, ce qui suggère qu'ils sont largement utilisés.

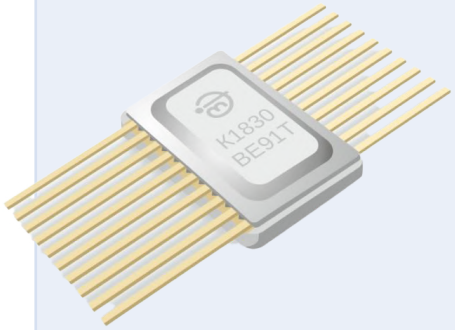


Figure 6. Le K1830BE91T de NIIET possède un cœur 8051 et est fonctionnellement équivalent au AT89C2051 de Microchip. (Source : https://niiet.ru)

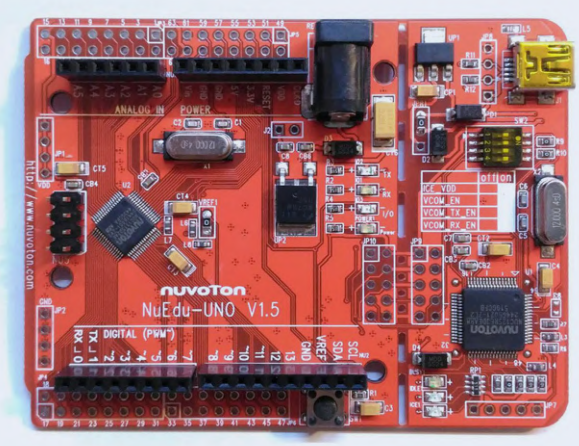


Figure 5. La carte NuMaker Uno de Nuvoton possède un module de programmation/débogage NuLink détachable. (Source : https://danchouzhou.blogspot.com)

Holtek

Quelques produits Holtek ont été utilisés par le passé dans des projets Elektor. Il ne s'agissait pas de microcontrôleurs, mais de décodeurs de clavier et de canaux RC. Cependant, Holtek fabrique également des MCU, dont beaucoup sont basés sur le 8051 ou les ARM Cortex-M0 et M3 ainsi que sur leur propre cœur. Comme chez beaucoup de fournisseurs de MCU à bas prix, la gamme des produits Holtek est divisée en MCU d'usage général (« type I/O ») et spécifiques. La documentation est bonne et l'EDI HT-IDE3000 (assembleur et C) est gratuit (bien que difficile à trouver), mais un programmateur Holtek est nécessaire. Un composant que j'ai trouvé intéressant est le MCU analogique HT66F4550 qui intègre deux amplis ops et possède une sortie audio.

Et les géants d'Asie ?

Elektor a publié des centaines de projets basés le plus souvent sur des microcontrôleurs PIC ou AVR de Microchip (anciennement Atmel), ou ESP d'Espressif, ou un MCU avec un cœur ARM de chez NXP ou ST. Le MSP430 de Texas Instruments a également fait quelques apparitions. À l'exception d'Espressif, tous ces fabricants sont européens ou américains. Cela témoigne d'un biais, car il y a en Asie beaucoup de grands fabricants de MCU.

Renesas

L'un des plus grands, sinon le plus grand fabricant asiatique de semi-conducteurs est Renesas, formé à partir de divisions de NEC, Hitachi et Mitsubishi. Selon certains, c'est même le premier fournisseur mondial de MCU. Les lecteurs de longue date d'Elektor se souviennent peut-être des séries d'articles sur les R8C et R32C/111 d'il y a une quinzaine d'années [5]. La récente famille de microcontrôleurs à 32 bits RX671 est spécialisée dans le pilotage en temps réel et l'interface homme-machine (HMI) sans contact par détecteurs de proximité et reconnaissance vocale, l'idéal pour les interfaces HMI hygiéniques modernes. Renesas fournit également de nombreuses cartes de développement et d'évaluation que je vous encourage à essayer. N'hésitez pas à nous faire part de vos trouvailles.

Nuvoton

Issu de Winbond en 2008, Nuvoton a acquis la division des puces moribonde de Panasonic en 2020. L'entreprise propose une large gamme de MCU à cœur 8051 et ARM, ainsi que quelques composants à cœur propriétaire. Contrairement à de nombreux concurrents, Nuvoton ne dispose pas de sa propre chaîne d'outils. Pour les dispositifs 8051, ils proposent Keil et IAR tandis que pour ARM, ils utilisent Eclipse. Sur GitHub, vous pouvez trouver de l'aide pour utiliser SDCC avec certains composants Nuvoton. La carte NuMaker Uno (**fig. 5**) est un bon début pour utiliser Nuvoton. Avec son contrôleur NuMicro NUC131 ARM Cortex-M0, elle est compatible Arduino. Elle comprend également un module débogueur / programmeur Nu-Link détachable, utilisable avec d'autres modules. Le support logiciel est disponible sur GitHub (OpenNuvoton). Consultez le dépôt NuMaker pour le support de mbed Arduino, MicroPython et autres.

Des MCU russes ?

Pour compléter cet article, j'ai voulu ajouter quelques informations sur les microcontrôleurs russes. Malheureusement, je ne lis pas le russe, et la plupart des sites web sont en russe, ce qui complique la recherche d'informations utiles. Je suis tombé sur Milandr, Mikron et le fabless Syntacore qui font tous des contrôleurs basés RISC-V. Selon [6], Milandr a pris une licence ARM, mais n'en mentionne pas d'applications sur son site.

NIIET fabrique le K1921VK01T destiné aux applications de commande de moteurs et de mesure intelligente. Il est construit autour d'un noyau ARM Cortex-M4F. OpenOCD propose un support pour ce MCU. En octobre 2021, NIIET a annoncé un contrôleur basé sur RISC-V pour remplacer les STM32- et MSP430 qui sont actuellement utilisés dans les « équipements civils » (comme ils les appellent) en Russie. Ils ont également des MCU RISC à 8 et 16 bits et quelques MCS-51 (Intel 8051) et MCS-96 (Intel 80196) (**fig. 6**).

Un monde d'options

Dans cet article, nous avons passé en revue quelques microcontrôleurs et fabricants rarement utilisés dans les projets Elektor, mais qui constituent pourtant une part importante du marché mondial des MCU. Bien sûr, cet article est loin d'être exhaustif et certains composants ou fabricants intéressants ont pu être oubliés. Pendant ma recherche pour cet article, j'ai compilé une liste de plus de 50 fabricants de microcontrôleurs en activité, et je suis sûr qu'il y en a beaucoup d'autres. Si vous en connaissez d'autres, ignorés, mais intéressants, que vous aimeriez partager avec les autres lecteurs, faites-le-moi savoir.

210630-04

Contributeurs

Idée et texte : **Clemens Valens**
Rédaction : **Jens Nickel, C. J. Abate**
Mise en page : **Harmen Heida**
Traduction : **Helmut Müller**

QUIZ

Le 4004 et le 4040 ont tous deux été conçus par Federico Faggin. Quels autres processeurs célèbres a-t-il également conçus ?

Intel 8080, Zilog Z8 & Z80

Des questions, des commentaires ?

Envoyez un courriel à l'auteur (clemens.valens@elektor.com) ou contactez Elektor (redaction@elektor.fr).

PRODUITS

- Livre « Maîtrisez les microcontrôleurs à l'aide d'Arduino (3e édition) », Clemens Valens, Elektor
  www.elektor.fr/18064
- Microcontrôleur Raspberry Pi RP2040
  www.elektor.fr/19742
- Livre en anglais, « ARM Microcontroller Projects », Elektor
  www.elektor.fr/17620

LIENS

[1] Stuart Cording, « The Birth of the Microprocessor », Elektor Industry, 03/2021 : www.elektormagazine.com/magazine/elektor-241/60042
[2] Pascal pour 8051 : https://turbo51.com/
[3] CH55xduino : https://github.com/DeqingSun/ch55xduino
[4] Padouk PMS150 : https://jaycarlson.net/2019/09/06/whats-up-with-these-3-cent-microcontrollers/
[5] Gunther Ewald, « R8C et compagnie », Elektor, 01/2006 : www.elektormagazine.fr/magazine/elektor-200601/10319
[6] Microcontrôleurs russes : https://geek-info.imtqy.com/articles/M4836/index.html

surveillance et débogage sans fil

Une solution pour Arduino, ESP32 et Cie

Peter Tschulik (Autriche)

Les cartes à microcontrôleur modernes, équipées d'un ESP32 ou d'un ESP8266 par exemple, sont, pour un prix modique, très performantes et d'un grand confort de programmation, notamment avec l'EDI Arduino. La fonction *Over-The-Air* (OTA) est particulièrement intéressante. Elle permet de télécharger facilement des mises à jour de programmes et des données via un chargeur d'amorçage qui utilise la communication par Wi-Fi, ce qui permet de se passer de la connexion physique de la carte au port USB d'un PC ou d'un ordinateur portable. Cependant, pour transmettre sans fil des données série (telles que les informations de débogage), il nous faut une autre solution.

Kit « Long Distance Wireless UART-RS232 ».

Les mises à jour logicielles d'appareils de l'Internet des Objets, donc qui fonctionnent à distance « sur le terrain », peuvent s'effectuer par voie hertzienne (OTA). Un bon tutoriel sur le sujet avec l'EDI Arduino est disponible à l'adresse [1]. Cette méthode présente toutefois un inconvénient : le moniteur série n'est pas pris en charge, de sorte que les informations de sortie et de débogage ne peuvent pas être transférées.
Sur mon dernier projet, un système d'arrosage de jardin basé sur l'ESP32, j'ai eu un problème avec un capteur défectueux. Au lieu de tirer un câble USB à travers la terrasse jusqu'à l'ordinateur dans mon salon, j'ai commencé à chercher une solution sans fil. Malgré des recherches intensives, je n'ai pas pu trouver de module standard qui réponde à mes exigences et j'ai donc entrepris de développer ma propre solution. Je me souvenais vaguement de systèmes qui permettaient de transmettre sans fil des données d'interface série ; en fait, j'ai rapidement trouvé ce que je cherchais auprès d'une entreprise d'Extrême-Orient. Le kit *Long Distance Wireless UART-RS232* [2] offre des taux de transfert allant jusqu'à 115 200 bauds avec des options de configuration et une portée allant jusqu'à 1 km. Pour 75 € la paire émetteur/récepteur, j'ai pensé que cela représentait un bon rapport prix/performance pour ce projet.

Adaptateur hôte-USB

Le choix d'un convertisseur USB/série s'est avéré beaucoup plus difficile. Bien qu'il soit possible d'appliquer le signal de données série du système d'arrosage distant directement au

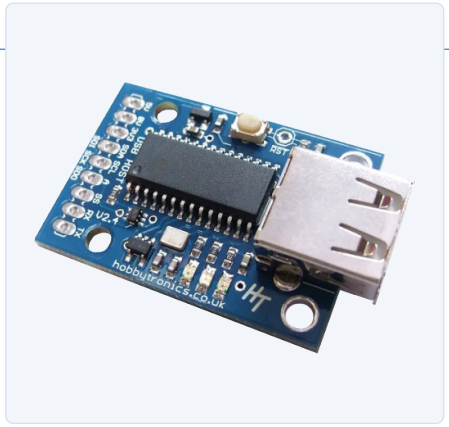


Figure 1. Petite par la taille, grande par le talent : la carte hôte-USB.
(Source : www.hobbytronics.co.uk)

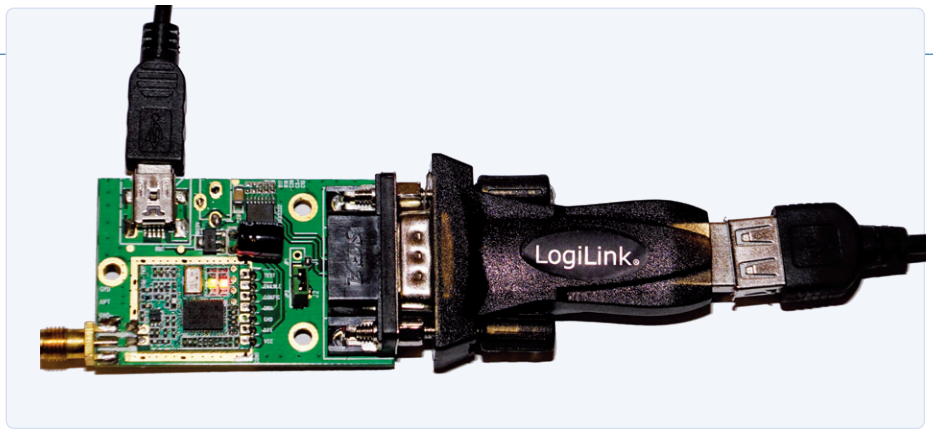


Figure 2. Il faut configurer les modules « Long Distance Wireless UART-RS232 ». Sur la photo, vous pouvez voir le cavalier en place et les LED verte et rouge.

module de transmission série, je souhaitais une solution plus universelle utilisant une connexion par port USB afin que la liaison puisse servir aussi facilement pour d'autres types de cartes de développement. Il faut pour cela un « adaptateur hôte-USB » qui se branche sur le port USB du site distant et communique avec la carte de développement comme le ferait un PC. Par chance, après quelques recherches intenses sur l'internet ([3] et [4]), j'ai repéré l'*USB-Host Shield* pour Arduino. Un fournisseur local avait l'article en stock, j'ai donc passé commande et quelques jours plus tard, j'étais occupé à le déballer sur mon bureau. Il ne m'a pas fallu longtemps pour découvrir un problème. Il y a de nombreuses plates-formes de développement différentes dans le monde des *makers* et plusieurs marques différentes de puces d'interface USB. Les cartes ESP32 par exemple utilisent les puces CP210x ou FTDI, les cartes Arduino utilisent le contrôleur Atmel ou les puces FTDI. Malheureusement, les bibliothèques pour Arduino de l'*USB Host Shield* ne prennent en charge qu'un seul type de puce à la fois. Cela signifie que vous devez rendre le logiciel de l'*USB-Host Shield* configurable et trouver le type de puce associé à la plate-forme de développement que vous utilisez. Ce n'est pas vraiment une solution prête à l'emploi ! J'ai fini par trouver une carte d'interface hôte-USB en provenance du Royaume-Uni qui s'adaptait beaucoup mieux aux différentes marques de puce d'interface (**fig. 1**) [5].

L'intérêt majeur de cette carte, c'est qu'elle peut fonctionner avec une large gamme de périphériques tels que des clés de mémoire flash, des claviers, joysticks et souris USB, des manettes Dualshock PS3/PS4, et dispose de pilotes série pour FTDI, CP210X, PL2303, CH340/1 et CDC, des périphériques MIDI et des modems USB. Leur site web fournit la liste de tous les micrologiciels gratuits. Si vous connaissez exactement l'application pour laquelle vous voulez utiliser la carte, vous pouvez la spécifier lors de la commande et elle sera fournie avec le micrologiciel installé. Pour cette application, j'avais besoin du micrologiciel *USB Host - Serial Driver for FTDI, CP210X, PL2303, CH340/1 and CDC* [6] ; il prend en charge toutes les puces USB utilisées par les différentes cartes de développement.

Pour programmer et configurer les modules UART RS232 sans fil ou la carte hôte-USB, il faut au minimum un convertisseur USB vers série, que l'on peut obtenir chez le même fabricant que pour les modules UART RS232 sans fil [8] ou par ex. chez Logilink [9]. Pour mon projet, j'avais besoin de deux convertisseurs de niveau RS232. J'ai choisi la bonne vieille solution du MAX232 [7].

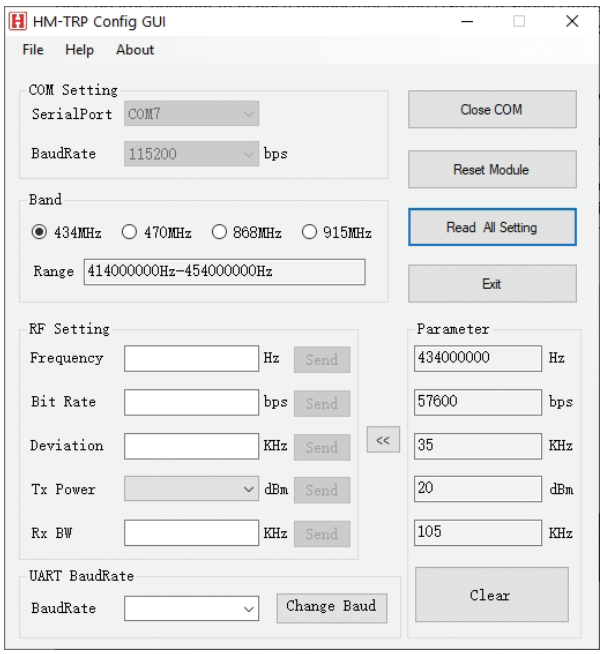


Figure 3. La configuration recommandée pour le module « Long Distance Wireless UART-RS232 ».

Configuration de la liaison

Lorsque j'ai reçu le kit *Long Distance Wireless UART-RS232*, la façon dont je devais configurer les modules n'était pas claire pour moi. J'ai contacté le vendeur et reçu le lendemain un courriel contenant le logiciel, un manuel et un lien vers une vidéo de configuration [10]. Tout d'abord, j'ai dû sortir les modules de leur boîtier en plastique et souder une barrette à l'emplacement du connecteur J2 (une rangée de plots juste devant le connecteur RS232). J'ai ensuite placé des cavaliers sur les broches selon les instructions afin que le module entre en mode configuration. Ensuite, j'ai connecté le module au PC via l'adaptateur USB/série. Le module est alimenté en 5 V (par ex. une alimentation USB enfichable) via le connecteur mini-USB (**fig. 2**).

Si les deux LED s'allument, vous pouvez lancer le programme **HY-TRP Setting GUI.exe** et régler le bon port COM et la vitesse de transmission par défaut sur 9 600. Cliquez ensuite sur le bouton *Open COM* et lisez toutes les valeurs avec *Read All Settings*. Les valeurs sont définies comme indiqué à la **figure 3**.

On peut trouver plus de détails dans le manuel [10]. Certains problèmes de fiabilité de la liaison ont été résolus en réduisant le débit en bauds de la liaison de transmission

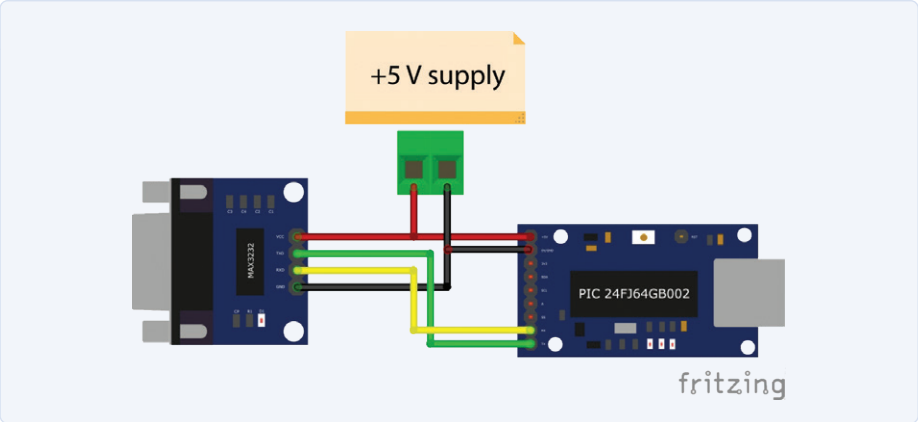


Figure 4. Câblage de la carte hôte-USB pour la configuration.

de 115 200 à 57 600. La carte hôte-USB possède un tampon qui tolère différentes vitesses de transfert. Les deux modules doivent être configurés de manière identique. Enfin, une fois la configuration terminée, n'oubliez pas de retirer les cavaliers.

Configuration de la carte hôte-USB

Pour programmer la carte hôte-USB avec le bon micrologiciel, référez-vous à la description donnée en [5]. Beaucoup de programmes disponibles en téléchargement chez Hobbytronics peuvent être chargés sur la carte. Le micrologiciel adéquat peut être chargé sur la carte en utilisant les instructions données en [6], et vous y trouverez également une description des paramètres de l'interface de commande. Pour la configuration, la carte hôte-USB doit être connectée à la carte convertisseur de niveau MAX3232 comme indiqué dans la **figure 4**.

Le convertisseur RS232-TTL se connecte à l'adaptateur RS232-USB, qui à son tour se connecte au PC via un câble USB. Le circuit n'a plus besoin que d'être alimenté par une alimentation stabilisée de 5 V.

Vous pouvez maintenant lancer un programme de terminal (ici Hterm), sélectionner le bon port USB et régler la vitesse de transmission sur la valeur par défaut de 9 600 bauds. En cliquant sur le bouton *Connect* en haut à gauche, vous vous connecterez à la carte hôte-USB. En haut à droite, les options *Newline at* et sous la fenêtre principale *Send on Enter* sont changées en *CR + LF*. Si vous entrez maintenant *HELP* dans le champ de saisie de la zone *Input Control* sous la fenêtre principale, les paramètres à jour devraient être listés comme indiqué dans la **figure 5**.

Nous modifions maintenant les paramètres de notre application à l'aide du champ de saisie *Input Control*. Tout d'abord, nous pouvons entrer la commande *BAUD 57600* pour modifier le débit en bauds qui doit

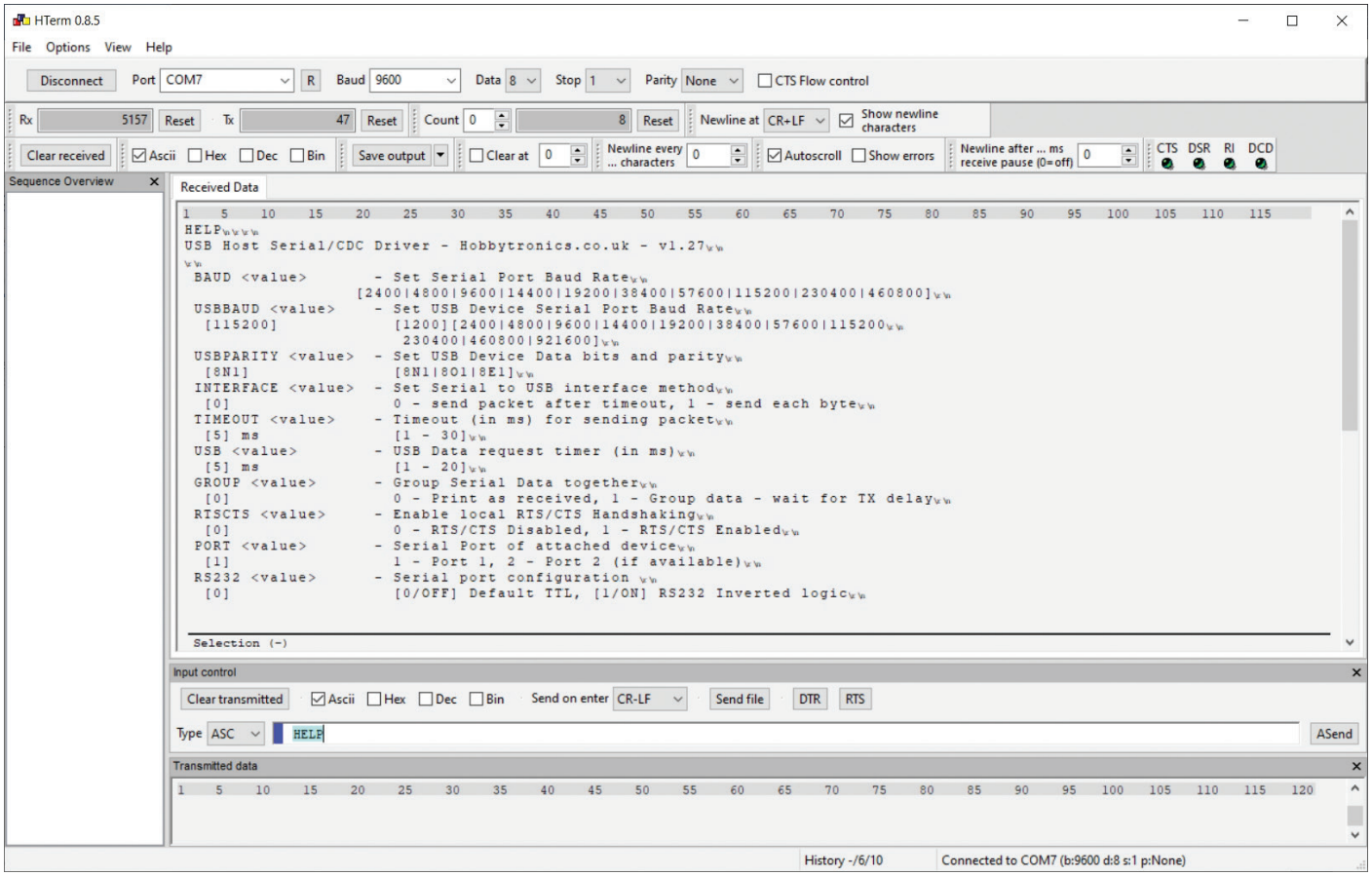


Figure 5. Configuration de la carte hôte-USB à l'aide de Hterm.

Tableau 1. Paramétrage dans HiTerm

BAUD	57600
USBBAUD	115200
USBPARITY	8N1
INTERFACE	0
TIMEOUT	5ms
USB	5ms
GROUP	0
RTSCTS	0
PORT	1
RS232	0

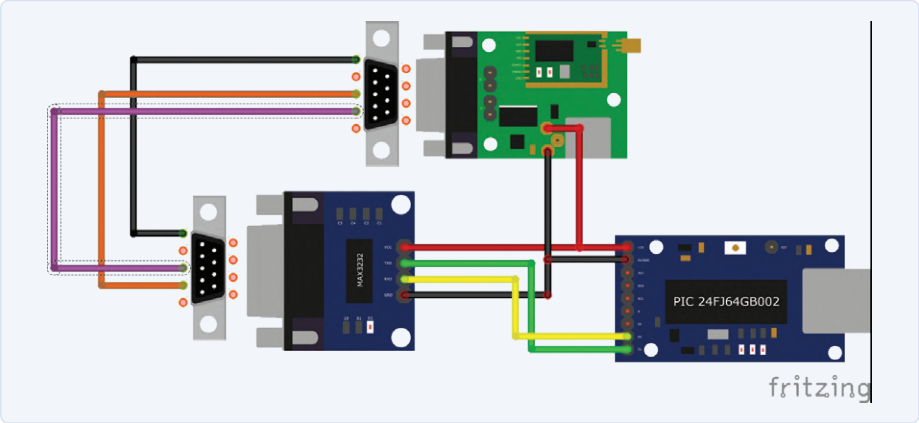


Figure 6. Câblage de l'émetteur à distance (emplacement de la carte IdO). La carte verte est le module sans fil. La carte IdO fournissant les données se connecte au port USB en bas à droite.

correspondre au débit en bauds défini pour les systèmes *Wireless UART*. Utilisez *Disconnect* puis *Connect* pour déconnecter et reconnecter le terminal au système, puis la commande *HELP* qui devrait vous permettre de vérifier la liaison avec la carte hôte-USB.

La commande *USBBAUD 115200* règle le débit en bauds de l'USB sur 115 200. Cela signifie que l'interface série de la carte Arduino/ESP32/ESP8266 connectée dans l'EDI Arduino doit toujours commencer par la commande *Serial.begin (115200)*.

On utilise les paramètres par défaut pour *USBPARITY, INTERFACE, TIMEOUT, USB, RTSCTS, PORT* et *RS232*. La valeur par défaut de l'option de regroupement des données série est OFF. La commande *GROUP 0* définit également ces propriétés à leur valeur par défaut. Le **tableau 1** donne un résumé des paramètres corrects. La configuration est ainsi terminée.

Câblage des modules

Avant de pouvoir câbler l'émetteur (du côté de la carte IdO sur le terrain) et le récepteur (du côté du PC), il est nécessaire de retirer du circuit imprimé le connecteur rond d'alimentation (à côté de la prise USB) sur les modules *Long Distance Wireless UART-RS232* afin de dégager les plots.

Du côté de l'émetteur, la prise mini-USB de la carte hôte-USB alimentera l'unité, voir le câblage à la **figure 6**. Pour connecter le module *Long Distance Wireless UART-RS232* à l'adaptateur RS232/TTL, vous devez utiliser un câble court terminé par des connecteurs SUBD-9

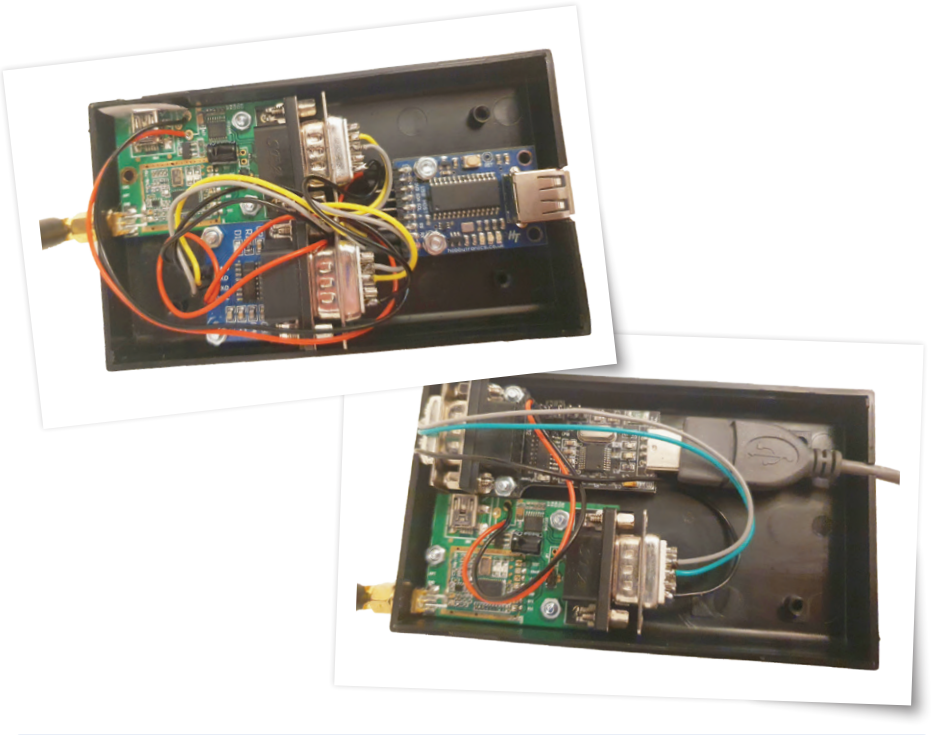


Figure 7. Câblage de la station réceptrice. Le PC est connecté en bas à droite par un câble d'extension USB.

mâles. Les fils des broches 2 et 3 sont croisés dans le câble comme le montre l'image.
Le module compact de [8] est utilisé comme convertisseur RS232/USB pour le récepteur et est câblé selon la **figure 7**. J'ai utilisé un câble d'extension USB entre le PC et l'unité de réception pour donner une certaine flexibilité dans le positionnement de l'unité à l'intérieur. Enfin, j'ai installé les unités d'émission et de réception dans deux boîtiers assortis. Les tests de mon système OTA ont montré la fiabilité de la transmission à travers deux étages avec des plafonds en béton armé. Dans l'ensemble, le système procure une installation de « câble USB virtuel » pratique, fiable et à longue portée, utile pour le développement et le débogage à distance d'applications IdO.

200549-01

Mode d'emploi

- Réglez le débit de l'interface de communication série de la carte utilisée sur 115 200 bauds en utilisant `Serial.begin(115200);` dans l'EDI Arduino.
- Une fois le micrologiciel transféré sur la carte distante, alimentez l'émetteur en 5 V via la prise mini-USB du module *Long Distance Wireless UART-RS232*.
- Puis connectez-le à la carte distante à l'aide d'un câble USB court.
- Branchez l'unité de réception sur un port USB libre du PC. Le récepteur est alimenté par le PC.
- Démarrez l'EDI Arduino, choisissez n'importe quelle carte, sélectionnez l'interface USB de la carte du récepteur et démarrez le moniteur série.
- Toutes les données envoyées par la carte devraient maintenant s'afficher.

Des questions, des commentaires ?

Envoyez un courriel à l'auteur (peter.tschulik@chello.at) ou contactez Elektor (redaction@elektor.fr).

Contributeurs

Idée, conception et texte : **Peter Tschulik**
Rédaction : **Rolf Gerstendorf**
Mise en page : **Giel Dols**
Traduction : **Denis Lafourcade**

PRODUITS

- **Offre groupée : livre « The Complete ESP32 Projects Guide » + carte ESP32-DevKitC-32D**
  www.elektor.fr/19897

- **Compilation d'articles sur l'ESP32 et l'ESP8266 (PDF)**
  www.elektor.fr/18516

LIENS

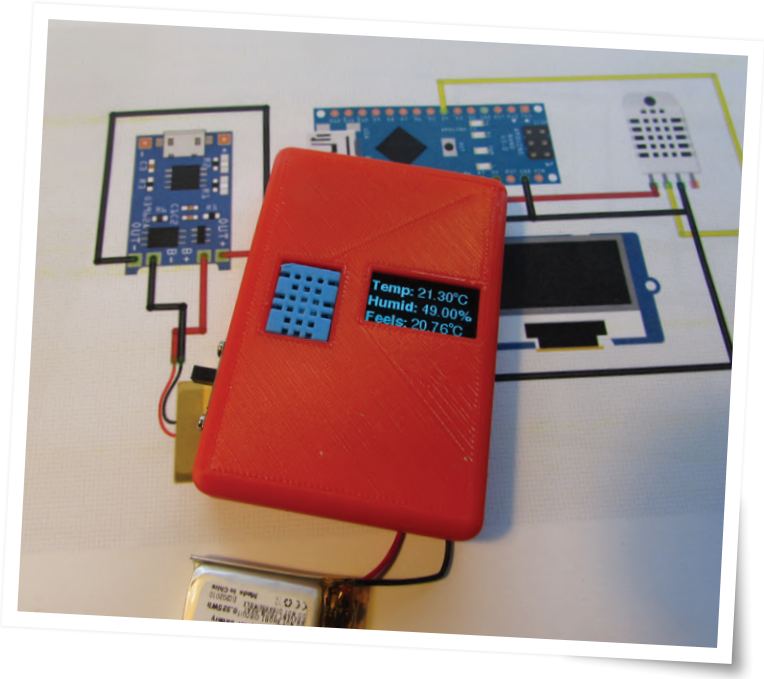
[1] Les bases de la programmation Over The Air (OTA) de l'ESP32 avec l'EDI Arduino : https://lastminuteengineers.com/esp32-ota-updates-arduino-ide/
[2] Module « Long Distance Wireless UART-RS232 » : http://www.aliexpress.com/i/32242647632.html
[3] USB Host Shield de SparkFun : http://www.sparkfun.com/products/9947
[4] USB-Host-Shield pour Arduino : https://bit.ly/3fJQrTO
[5] Carte hôte-USB v2.4 : https://www.hobbytronics.co.uk/usb-host-board-v24
[6] Hôte-USB – Pilote série pour FTDI, CP210X, PL2303, CH340/1 et CDC : https://www.hobbytronics.co.uk/usb-host-serial
[7] Adaptateur RS232-TTL : https://bit.ly/3peJH3r
[8] Câble adaptateur 2 fonctions USB COM Port DB9 TTL232 RS232 TTL 232 CH340T USB2.0 : https://bit.ly/3fHUGQ3
[9] Adaptateur USB-2.0 vers série LogiLink : https://bit.ly/3ma21Kg
[10] Instructions de configuration pour le module « Long Distance Wireless UART-RS232 », fichier HY035_HM-TRP-RS232.rar : https://1drv.ms/u/s!AjrAGEbCBLlsugxiqmd1FRRCJjic

station de mesure de température et d'humidité, au format de poche

Utilisation de modules prêts à l'emploi

Aarav Garg (Inde)

La réalisation de projets électroniques modernes a été considérablement facilitée par l'utilisation de modules prêts à l'emploi. Il suffit d'un peu de câblage et de micrologiciel pour créer un nouvel appareil comme celui présenté ici.



Luc Lemmens, labo d'Elektor : le concepteur de ce montage est un jeune innovateur indien de 15 ans, Aarav Garg, qui est plus qu'heureux de présenter sa « station météo de poche » dans notre magazine. Elle contient un Arduino Nano, un écran OLED de 0,96" et un capteur d'humidité et de température DHT11. Elle est alimentée par une batterie LiPo qui peut être rechargée via un module de charge USB. Un boîtier imprimé en 3D et fabriqué sur mesure complète cet appareil portable et pratique. Nous allons laisser la parole à Aarav pour qu'il nous explique comment il est parvenu à ce montage. Je l'ai reproduit et testé avec succès au laboratoire d'Elektor. On trouvera quelques instructions et commentaires supplémentaires dans les encadrés.

Dans cet article, vous apprendrez à construire une station météo de poche à base d'une carte Arduino Nano. Il s'agit d'un appareil compact que vous pouvez emporter n'importe où, dans votre poche, et qui vous donnera la température et l'humidité en temps réel sur son écran OLED. Ainsi, vous saurez toujours s'il faut prendre un parapluie ou une ombrelle ! L'appareil est équipé d'une batterie LiPo rechargeable de 160 mAh. Il s'agit d'un excellent projet d'apprentissage et amusant à réaliser.

Étape 1 : rassembler les composants

La première chose à faire au début de tout projet est de rassembler les composants nécessaires, comme le montre la **figure 1**. Les composants requis pour ce projet sont les suivants :

- Arduino Nano avec câble
- Module à capteur de température DHT11
- Écran OLED de 0,96 pouce
- TP4056, module de charge de batterie
- Petite batterie (j'ai utilisé une batterie LiPo de 160 mAh)
- Interrupteur à glissière

Outils :

- Fer à souder
- Fils de câblage
- Pistolet à colle chaude
- Imprimante 3D pour le boîtier (facultatif)

Rassemblez et/ou achetez tout ce qui est nécessaire et passez à l'étape suivante.

Étape 2 : disposition des composants

Il nous faut maintenant planifier la disposition de tous les composants à l'intérieur d'un boîtier. Je voulais que l'appareil soit aussi fin que possible, afin qu'il tienne aisément dans une poche. J'ai donc étalé tous les composants en évitant une structure à plusieurs couches qui aurait certes diminué la longueur et la largeur de l'appareil, mais aux dépens de son épaisseur.

Reportez-vous à la **figure 2** pour voir comment j'ai disposé les composants à l'inté-

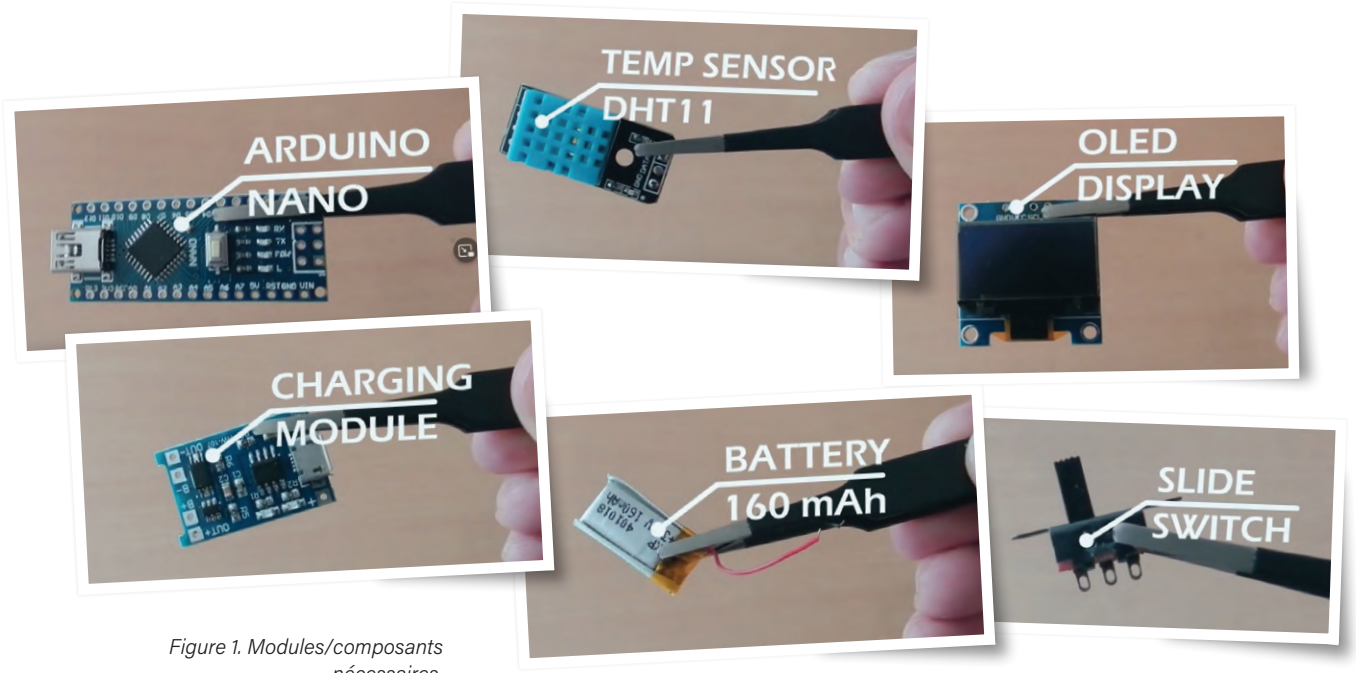


Figure 1. Modules/composants nécessaires.

rieur de ma station météo de poche. Vous pouvez aussi le faire à votre idée.

Étape 3 : schéma

À cette étape, il s'agit de concevoir un schéma pour notre station météo de poche. Il est en fait très simple, car il n'y a que très peu de modules à interconnecter, et aucun module à modifier. La **figure 3** donne le schéma de principe, auquel vous pouvez vous référer si nécessaire. Nous devons connecter la batterie à son module de charge et la sortie de ce module à la carte Arduino Nano. J'ai utilisé une carte Arduino Nano en raison de sa taille qui est parfaite pour ce projet ! Ensuite, connectez le module de capteur de température et l'écran OLED à la carte Arduino. Après cette découverte du schéma, passez à l'étape suivante.

Étape 4 : soudage / réalisation des connexions

Maintenant, il ne nous reste plus qu'à souder les fils de liaison entre les modules selon le schéma que nous avons dessiné précédemment. Coupez les fils à la bonne longueur, cela évitera un fouillis de fils plus tard. Efforcez-vous de réaliser des soudures précises pour éviter tout court-circuit. Tout ce proces-

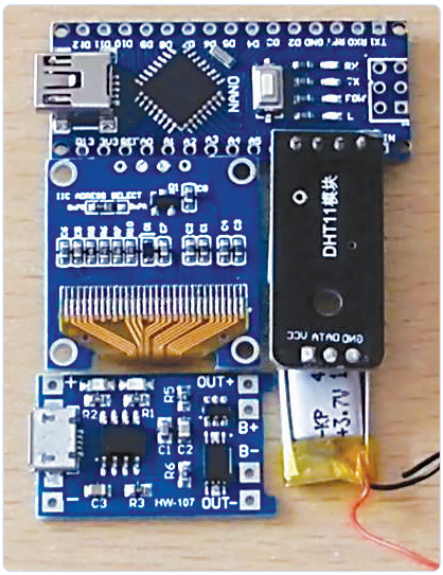


Figure 2. Disposition des modules.

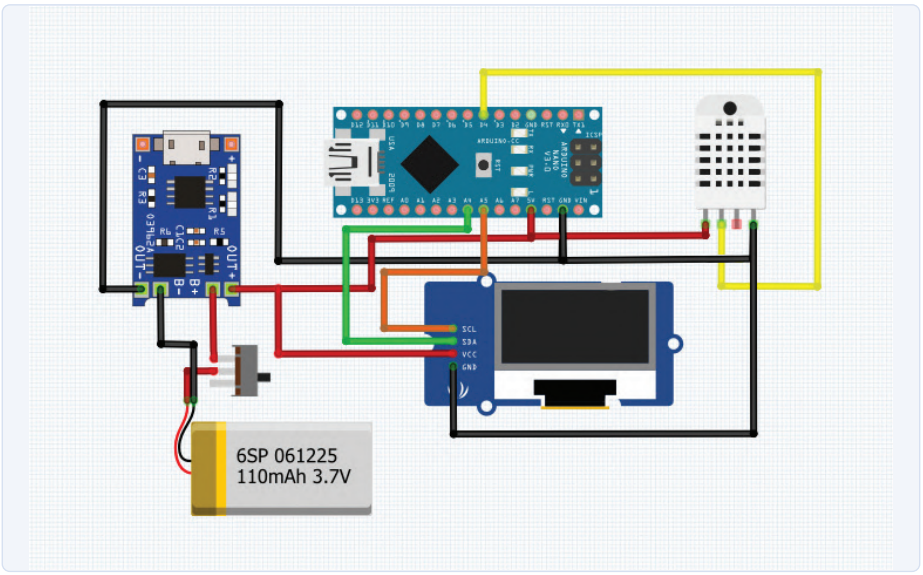


Figure 3. Dessin Fritzing montrant les interconnexions.

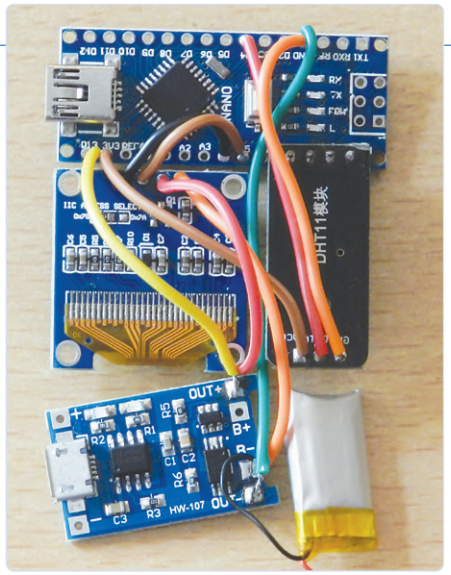


Figure 4. Après la réalisation de la plupart des interconnexions.

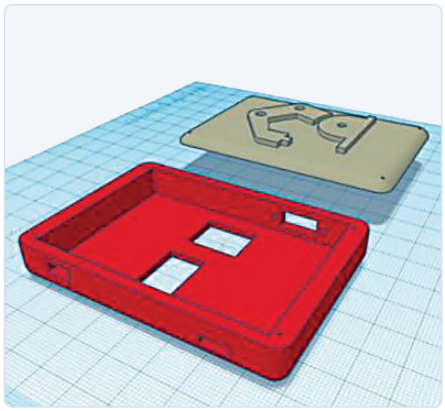


Figure 5. Conception du boîtier, réalisée avec Tinkercad.

sus peut paraître fastidieux, mais vous en serez récompensé plus tard, croyez-moi ! Une fois que vous avez terminé les soudures, ça doit ressembler à ce que montre la **figure 4.** Passez alors à l'étape suivante.

Étape 5 : après le soudage

Tout ce paquet de modules et de fils ne peut certes pas rester tel quel : il faut le loger dans un boîtier pour qu'il prenne un aspect professionnel. Et la meilleure option qui s'offre à nous est l'impression 3D. Alors, passez à l'étape suivante pour concevoir le boîtier et l'imprimer !

Étape 6 : fabrication du boîtier

J'ai conçu le boîtier de la station météo de poche (**fig. 5**) avec Tinkercad, un logiciel de CAO extraordinaire. Il est conçu pour des utilisateurs de tout niveau de compétence en CAO, même débutants. Je n'ai pas d'imprimante 3D, je me suis donc rendu sur le site d'IAmRapid, j'ai téléchargé mes fichiers STL pour obtenir un devis instantané et j'ai commandé les pièces dans la foulée. Le boîtier s'est avéré d'une grande qualité. Heureusement, toutes les découpes que j'avais prévues lors de la conception pour les différents modules et ports se sont avérées parfaites à 100%. *Les fichiers 3D pour l'impression du boîtier sont disponibles en téléchargement sur la page Elektor Labs de ce projet [1].* Passons maintenant à l'insertion de l'ensemble du circuit à l'intérieur du boîtier et à sa fixation.

Étape 7 : mise en place du circuit dans le boîtier

Il s'agit maintenant de placer l'ensemble du circuit à l'intérieur du boîtier fabriqué à l'étape précédente. Pour donner à l'appareil l'aspect professionnel requis, il est très important que tous les ports soient placés avec précision dans leurs découpes respectives. Pour assurer le bon fonctionnement de l'appareil, tous les composants doivent être solidement fixés à leur place et ne pas bouger à l'intérieur du boîtier. Pour cela, j'ai utilisé de la colle chaude. La **figure 6** montre le résultat. Une fois que vous avez terminé, passez à l'étape suivante.

Étape 8 : ajout de l'interrupteur

L'opération précédente terminée, il est temps d'ajouter l'interrupteur à glissière dans son emplacement dédié. Nous n'avions pas encore connecté l'interrupteur au circuit, car il doit être inséré dans le boîtier depuis l'extérieur (**fig. 7**).
Après avoir inséré l'interrupteur dans son emplacement, utilisez deux petites vis pour le maintenir en place. Connectez ensuite les deux fils, l'un provenant du VCC de la carte Arduino et l'autre de la sortie positive du module de charge de la batterie. De cette façon, la manœuvre de l'interrupteur allumera le circuit complet.

Étape 9 : fermeture du boîtier

Maintenant, nous devons fermer le boîtier. J'ai utilisé quelques vis pour fixer le couvercle du boîtier. J'avais prévu des trous de vis dès la conception du boîtier, prévenant ainsi tout problème ultérieur.
Assurez-vous que le couvercle est en place pour que l'appareil ait un aspect professionnel et soit pratique à transporter ! J'ai personnalisé l'appareil en mettant mon logo sur le couvercle.

Étape 10 : bataille de codage !

Nous en arrivons à ce qui distingue notre appareil d'une simple boîte en plastique sans intérêt : le micrologiciel de l'Arduino. Nous allons donc programmer notre station météo de poche pour qu'elle fasse son travail avec

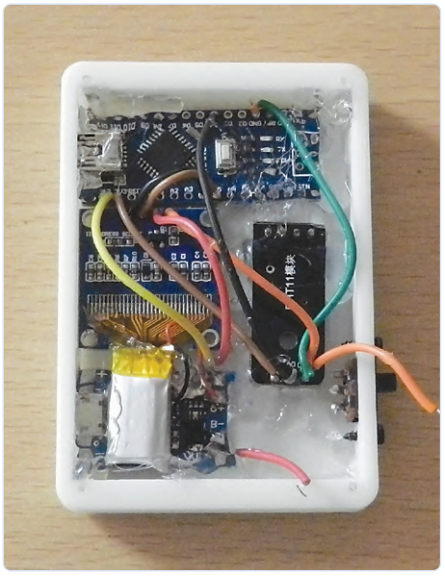


Figure 6. L'électronique collée à chaud à l'intérieur du boîtier.

Figure 7. L'interrupteur est fixé à l'extérieur du boîtier.

Données du capteur

Le DHT11 est un capteur d'humidité et de température bien connu dans le milieu des faiseurs. Ce n'est pas l'appareil le plus précis, mais il est bon marché et largement disponible. Les relevés de température et d'humidité du capteur sont affichés et utilisés pour calculer l'indice de chaleur [2], la troisième valeur affichée en bas de l'écran LCD (intitulée « Feels »). Il s'agit de la température perçue par une personne pour une combinaison particulière de température et d'humidité relative ; elle est calculée dans la bibliothèque Arduino DHT11. Il ne faut pas le confondre avec la *température ressentie*, qui tient compte de l'influence de la vitesse du vent.

Le croquis Arduino

Avec des projets comme celui-ci, rien ne fonctionne sans micrologiciel, en l'occurrence le croquis Arduino. Comme pour de nombreux modules, circuits intégrés et capteurs ordinaires, il existe des bibliothèques Arduino prêtes à l'emploi qui facilitent grandement la vie du programmeur. Le croquis d'Aarav utilise – entre autres – des bibliothèques graphiques, de contrôle de l'écran OLED, de lecture et de traitement des données du capteur DHT. Le listage ci-dessous contient les fonctions `setup()` et `loop()` du croquis, montrant que seule une quantité minimale de code est nécessaire pour faire fonctionner la station météo, le gros du traitement étant effectué par les fonctions des bibliothèques. Deux instructions sont nécessaires pour lire la température et l'humidité du DHT11, et une troisième pour calculer l'indice de chaleur à partir des données du capteur. Le reste de `loop()` affiche les valeurs obtenues sur l'écran. La plus grande partie du croquis, qui n'est pas reproduite ici, sert à afficher le logo de l'auteur à l'écran lorsque l'appareil est mis sous tension ou réinitialisé. Les données du capteur sont également envoyées au moniteur série de l'EDI Arduino à des fins de débogage.

```
void setup() {
   Serial.begin(9600);
   dht.begin();
  // SSD1306_SWITCHCAPVCC = generate display voltage from 3.3V internally
   if(!display.begin(SSD1306_SWITCHCAPVCC, 0x3C)) { // Address 0x3C for 128x64
     Serial.println(F("SSD1306 allocation failed"));
     for(;;); // don't proceed, loop forever
   }
  testdrawbitmap();    // draw the required image
   delay(1000);
 }
void loop() {
   float h = dht.readHumidity();                    // read humidity
   float t = dht.readTemperature();                 // read temperature
   float hic = dht.computeHeatIndex(t, h, false);   // compute heat index for temp in °C
  // Printing the results on the serial monitor
   Serial.print("Temperature = ");
  ............
}
```

Que faire si ça ne marche pas ?

Si vous lisez ceci, c'est que vous avez sûrement réalisé le montage. S'il fonctionne : bravo ! Sinon, vous aurez appris ce qu'il faut faire pour construire quelque chose, mais aussi ce qu'il ne faut pas faire, ce qui est au moins aussi important.

L'écran LED ne s'allume pas : soit vous l'avez fusillé, soit il y a une erreur dans votre code, par exemple un oubli d'initialiser l'écran ou une assignation d'adresse I²C incorrecte. Revérifiez votre code et testez si l'écran lui-même fonctionne ou non. Si oui, essayez de modifier le code. Toutes les lectures affichent « NA » : cela se produit si votre capteur de température a un problème. Soit vous avez fait une erreur de connexion à la carte Arduino : revérifiez le câblage. Soit le capteur lui-même a un défaut : essayez de le remplacer !

Ça fonctionne avec le câble USB mais pas avec la batterie : c'est le signe d'un problème avec votre batterie ou peut-être avec les connexions de la batterie !

Notes du laboratoire d'Elektor sur le matériel

Pour une réalisation compacte de la station météo de poche d'Aarav, il faut avoir un Arduino Nano sans broches soudées, mais malheureusement les cartes Nano dans mon tiroir en avaient toutes. Cela les rend trop hautes pour prendre place dans un boîtier plat et il est moins facile d'y souder des fils de connexion. Bien sûr, vous pouvez acheter un module sans broches, mais avec un peu de précautions, les broches peuvent être enlevées sans endommager la carte. Coupez les broches à ras de la base en plastique à l'aide d'une pince diagonale, puis découpez le plastique. Enfin, dessoudez les restes des broches et enlevez la soudure avec une pompe ou une tresse à dessouder. Le connecteur ISP à 6 broches est un peu plus difficile à enlever, mais on y arrive par la même méthode.

L'écran de 0,96" que j'ai commandé avait aussi des broches que j'ai enlevées pour gagner de la place dans le boîtier.

Mais avec ces écrans, il peut y avoir un autre problème : la plupart d'entre eux peuvent être configurés pour une interface SPI ou I2C, le projet utilisant cette dernière. J'ai fait l'erreur de commander la version SPI, et, même si la reconfiguration est possible, je ne la recommande pas pour ce projet. Non seulement parce que cela implique une soudure de CMS, mais aussi parce que quelques composants et câblages supplémentaires sont nécessaires.

À la différence du dessin Fritzing (**fig. 3**), Aarav a utilisé un module DHT11 : une carte qui ne contient pas seulement le DHT11 lui-même, mais aussi (entre autres) une résistance de rappel sur la broche de sortie de données. La résistance de rappel interne de l'entrée d'un Arduino Nano peut être trop élevée pour assurer le bon fonctionnement du capteur, donc une résistance supplémentaire de 10 kΩ peut être nécessaire si vous - comme je l'ai fait - utilisez un capteur DHT11 séparé. Mais avec mon prototype, ce fut inutile, le signal de données apparaissait beau et propre sur un oscilloscope.

Même si la réalisation matérielle est relativement simple, je vous conseille de programmer l'Arduino Nano et de tester la station météo avant de coller les modules dans le boîtier, quand il sera encore facile de corriger les éventuelles erreurs de soudage.

une efficacité maximale. Le croquis Arduino peut être téléchargé à partir de la page Elektor Labs de ce projet [1], mais si vous le souhaitez, vous pouvez aussi mettre les mains dans le cambouis et écrire le code vous-même !

Étape 11 : et c'est parti !

Et voici notre station météo de poche entièrement opérationnelle. Elle dispose d'un écran OLED pour vous permettre de profiter de la météo de manière optimale. Elle comprend également une batterie rechargeable avec un port de charge USB, et honnêtement, la batterie dure assez longtemps pour qu'elle n'ait que rarement besoin d'être rechargée. Elle dispose également d'un port Arduino Nano pour télécharger par la suite du code modifié. L'interrupteur présent à l'extérieur est également d'usage très pratique. En outre, la réalisation compacte de l'appareil permet de le glisser même dans la plus petite des poches ! Vous serez fier d'arborer cet appareil partout où vous allez et de l'utiliser comme station météo.

210394-04

Contributeurs

Idée, conception et texte : **Aarav Garg**
Illustrations : **Aarav Garg, Patrick Wielders, Luc Lemmens**
Traduction : **Helmut Müller**
Rédaction : **Luc Lemmens**
Mise en page : **Giel Dols**

Des questions, des commentaires ?

Envoyez un courriel à l'auteur (gargaarav79@gmail.com) ou contactez Elektor (redaction@elektor.fr).

LIENS

[1] Page Elektor Labs de ce projet : www.elektormagazine.fr/labs/pocket-weather-station
[2] Wikipédia sur l'indice de chaleur : https://en.wikipedia.org/wiki/Heat_index

réparation des batteries au lithium

Économisez de l'argent et augmentez la puissance !

Thomas Scherer (Allemagne)

C'est une histoire banale. Après quelques années d'utilisation, le tournevis sans fil doit être rechargé plus souvent et l'aspirateur sans fil n'a plus assez d'énergie pour ramasser vos miettes. Dans mon cas, c'est le robot tondeuse à gazon qui a rendu l'âme au milieu de la tonte, refusant obstinément de retourner à sa station de charge. La solution la plus simple et la plus coûteuse consiste à acheter une batterie de rechange. Mais avez-vous pensé à tout simplement remplacer les cellules de la batterie ? Cette solution est plus économique et réduit les déchets. En outre, vous pouvez choisir des cellules de remplacement d'une capacité supérieure à celle des cellules d'origine. Il ne s'agit plus d'une simple réparation, mais d'une mise à niveau !

Les appareils alimentés par des batteries Li-ion conservent leur puissance certainement beaucoup plus longtemps que ceux du passé alimentés par des cellules NiMH. Cependant, après de nombreux cycles de charge/décharge, il arrive un moment où la meilleure batterie au lithium a tellement perdu de sa capacité de stockage d'énergie qu'elle doit être remplacée. J'en ai fait l'expérience avec mes propres appareils et mes amis et collègues me demandent souvent conseil à ce propos. La solution la plus simple consiste à consulter le site web du fabricant de l'appareil pour voir si une batterie de rechange est disponible. Parfois, il n'y en a pas et, quand il y en a, son prix peut être dissuasif. Dans mon cas, l'appareil fonctionnait parfaitement et semblait bien parti pour quelques années encore – avec juste une batterie neuve. Dans ce cas, il peut être intéressant de démonter la batterie et de remplacer les cellules individuelles le moment venu, ce qui est souvent moins cher. On peut même envisager d'améliorer les performances en remplaçant les cellules d'origine par des cellules de puissance supérieure. Si vous avez retenu cette solution, préparez-vous à vous armer de tournevis et du fer à souder.

Tout est devenu silencieux…

Dans mon cas, j'avais remarqué que ma tondeuse à gazon Robbi [1] s'arrêtait de tondre au bout d'une demi-heure pour rallier sa station de charge pour une recharge d'une heure et demie. Auparavant, ça prenait une heure de recharge pour une heure de tonte. Ce changement de routine était-il un signe ? J'avais la tondeuse Robbi depuis quatre ans et je savais qu'elle était alimentée par des cellules au lithium, le moment était sans doute venu de les remplacer.

Plus tard dans l'après-midi, j'ai remarqué que le silence régnait dehors depuis un moment. Là, au milieu de la pelouse, Robbi s'était éteinte. Impossible de la réveiller en appuyant sur ses boutons. Je l'ai traînée jusqu'à la station de recharge et l'ai mise en charge. Robbi a fini par s'animer et son menu utilisateur m'a indiqué que son temps de fonctionnement cumulé était de 2 938 heures. Comme cela correspond à près de 1 500 cycles de charge, il était clair que la batterie n'en avait plus pour longtemps.

Et ce fut pire que je le craignais. Après une charge complète, Robbi a terminé son travail, mais elle ne s'est pas réveillée le lendemain

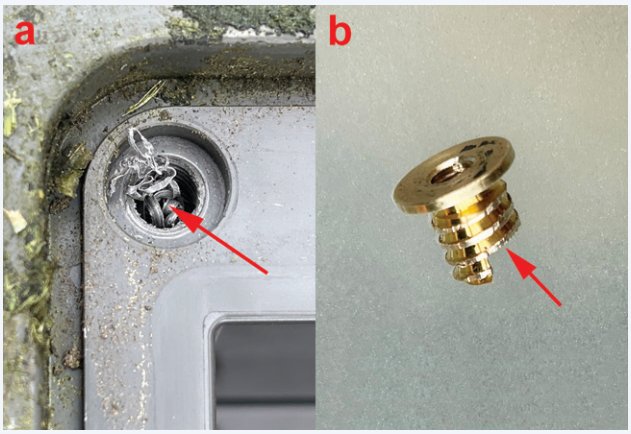


Figure 1. L'une des vis avait été vissée de travers à l'assemblage. Elle a fini par sortir, mais en emportant l'écrou à visser avec elle. L'écrou a été remis en place et collé.

matin. La remettre en charge n'a rien changé. J'avais récemment traité la pelouse avec de l'engrais et il avait plu abondamment. Il me fallait donc agir rapidement. Je pouvais presque entendre l'herbe pousser…

Désosser un pack ?

Chez le fabricant, le prix de la batterie de rechange était d'environ 100 €, alors que chez un autre fournisseur elle en coûtait la moitié. Pas mal ! D'après la spécification, il me fallait un pack de rechange de 18 V avec une capacité de 2,1 Ah. Cela signifiait cinq cellules, probablement dans le format standard 18650. Pour en avoir le cœur net, j'entrepris de dévisser le couvercle du logement de la batterie.

Évidemment, ce fut plus facile à dire qu'à faire. Trois des vis sont venues facilement, mais la quatrième était complètement bloquée. Finalement, elle est sortie, mais en entraînant son écrou à visser, arraché dans le processus et bloqué sur son filetage. Il semble que lors de l'assemblage en usine cette vis ait été vissée de travers et forcée. La **figure 1a** montre l'état de l'emplacement de l'écrou après son retrait, avec quelques copeaux restants. J'ai fini par séparer l'écrou de la vis et, bien qu'un bout de l'écrou en laiton ait été cassé (**fig. 1b**), il semblait récupérable. Le logement des piles étant conçu pour être étanche, l'écrou devait être remonté pour que les quatre vis puissent être suffisamment serrées pour assurer une bonne étanchéité. J'ai utilisé une colle époxy pour fixer l'écrou, qui a l'air de n'avoir jamais bougé (en haut à gauche sur **figure 2b)**. La partie mécanique réparée, l'attention s'est portée sur la batterie que la **figure 2b** montre dans son logement. Le contour des cinq cellules est clairement visible, et une règle graduée a confirmé qu'il s'agissait de cellules 18650. Il y a également beaucoup d'espace libre (**figure 2c**), ce qui m'a fait réfléchir : pouvais-je mettre cet espace à profit en utilisant des cellules de rechange plus nombreuses ou plus grandes ? Ma décision était maintenant prise ; un simple échange standard de batterie n'était plus une option.

Échange de cellules

La **figure 2a** montre que deux évidements dans le couvercle de la batterie empiètent sur l'espace disponible, ce qui exclut l'installation de deux rangées de cinq éléments pour doubler la capacité. Il existe cependant des cellules plus chères au format 18650 avec une capacité supérieure. Certaines marques parmi les plus réputées proposent des cellules de capacité jusqu'à 3500 mAh, mais au prix de 10 € pièce. AliExpress, eBay et d'autres sites similaires en proposent avec des caractéristiques encore supérieures, des offres à considérer avec prudence !

J'étais sur le point de passer une commande de cinq cellules quand je suis tombé sur d'autres cellules au format 21700, un peu plus inhabituel. Bien que légèrement plus grandes, elles offraient une capacité nettement supérieure pour le même prix. J'étais sûr de pouvoir les installer d'une manière ou d'une autre (je pensais à une rangée de trois et une rangée de deux pour former un profil en W). Avec une remise sur la quantité et les frais de port, les cinq cellules de 4 000 mAh me coûtaient 26 € au total. Elles atterrirent dans ma boîte aux lettres deux jours plus tard, ce qui n'était pas trop tôt : la pelouse avait sérieusement besoin d'être tondue.

Considérations

L'assemblage d'un pack de batteries à partir de cellules individuelles requiert de l'habileté manuelle, des connaissances en électricité et

Figure 2. Couvercle du logement de la batterie (2a), avec la vieille batterie en place (2b) et le logement vide (2c).

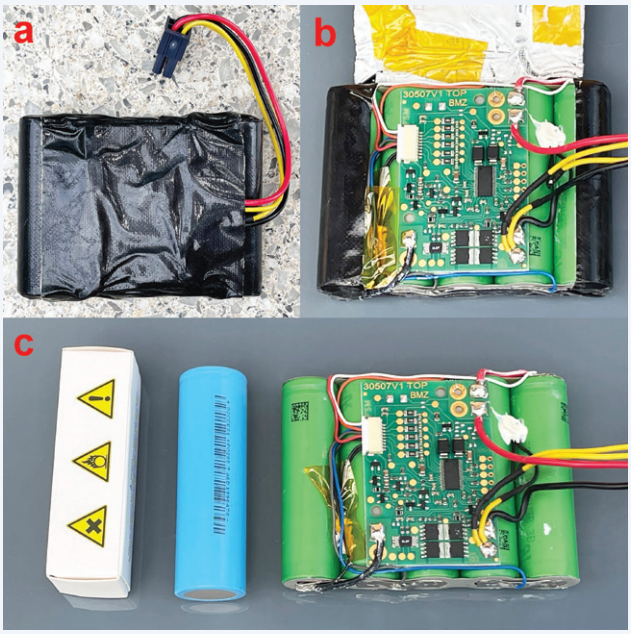


Figure 3. L'ancienne batterie enveloppée dans du ruban adhésif noir (3a). Les cellules et la carte BMS sont visibles après enlèvement de la partie supérieure du ruban adhésif (3b). Une cellule 21700 dans son carton et déballée, comparée à l'ancien pack de cinq cellules 18650 (3c).

Fonction bascule du BMS

Lorsque j'ai retiré la batterie représentée sur la figure 3c, j'ai été surpris de mesurer 19,2 V directement à ses bornes. Avais-je été trop hâtif en commandant les cellules de rechange ? En revanche, je n'ai pu mesurer qu'environ 18,5 V sur le connecteur du pack, au niveau de la connexion au moteur de la tondeuse (à droite de la carte BMS), qui présentait une impédance de source très élevée. Avec mes doigts, j'ai pu décharger cette source à la terre, ce qui a fait chuter la tension à quelques volts seulement. Le BMS était-il défectueux ?

J'ai branché une résistance de charge de 24 Ω directement aux bornes du pack et j'ai mesuré un courant de 0,75 A et une tension de batterie de 18,2 V. J'ai ensuite débranché la charge et mis le pack en charge. Après seulement quelques secondes de charge à 0,5 A, la sortie de tension du BMS est passée en mode basse impédance pour tirer un certain courant de la batterie via le BMS. Il semble que le BMS ait détecté que la tension de la batterie était passée sous le seuil bas de tension lors de la dernière utilisation de la tondeuse et qu'il ait bloqué le FET pour déconnecter la batterie. Cet état « bloqué » avait été verrouillé dans le BMS. Pour le tester, j'ai connecté une charge de 12 Ω à la batterie. Au bout de cinq minutes, le BMS a bloqué juste au-dessus de 13 V pour retourner à l'état normal à la charge suivante.

Ouf… Avec le BMS prêt à fonctionner, j'ai procédé à l'échange de cellules.

une soudeuse par points. Sur l'ancien pack (déballé sur la f**igure 3)**, les pôles positifs et négatifs des cinq cellules sont reliés en série par de fines bandes de nickel soudées par points. Cette méthode de connexion permanente n'endommage pas la cellule. Bien que la chaleur produite lors du soudage par points soit intense, elle est fugace et localisée, de sorte que les cellules s'échauffent à peine. Sans doute n'a-t-on jamais trop d'outils, mais là, je ne vois pas à quoi me servirait une soudeuse par points, je ne peux donc pas justifier une telle dépense. Alors, j'ai choisi de souder les fils directement sur les pôles de la cellule. Cette méthode n'est pas recommandée, sauf si vous suivez certaines règles et êtes conscient des dangers.

Le pack utilisé (**fig. 3)** est typique de ce qu'on trouve dans de nombreux autres appareils fonctionnant sur batteries. Il se compose de plusieurs cellules connectées en série et d'un module de gestion des batteries (BMS, *Battery Management System*), illustré sur les **figures 3b** et **3c**. Cette dernière montre également une comparaison de taille entre les nouvelles cellules et celles de l'ancien pack. Le BMS remplit trois fonctions :

1. Il équilibre les cellules, c'est-à-dire qu'il les maintient toutes à la même tension ou au même état de charge.
2. Il empêche que les cellules soient surchargées.
3. Il déconnecte la charge en cas de sous-tension pour éviter les décharges profondes.

La puce de la carte BMS qui a le plus de pattes gère toutes ces tâches. Il s'agit d'un microcontrôleur spécialisé qui surveille les tensions des cellules (via le connecteur de gauche). En cas de surtension ou de sous-tension, il déconnecte les cellules à l'aide de deux paires de MOSFET (en bas). Pour en savoir plus sur l'équilibrage des batteries au lithium, consultez [2] et [3].

Un pack neuf comprend un BMS, rendant inutile l'ancien BMS qui fonctionne toujours. Mais si vous ne changez que les cellules, vous pouvez le réutiliser. Une caractéristique importante du BMS que vous devez connaître est décrite dans l'encadré **Fonction bascule du BMS**.

LIENS

[1] « énergie solaire pour les robots de tonte », T. Scherer, Elektor, 07-08/2021 : www.elektormagazine.fr/200553-04
[2] « LiPo Auto Balancer », T. Scherer, Elektor, 06/2010 : https://bit.ly/3AhZa7e
[3] « Battery Management System Tutorial », Elektor Business Magazine, 02/2017 : https://bit.ly/3ly9cND

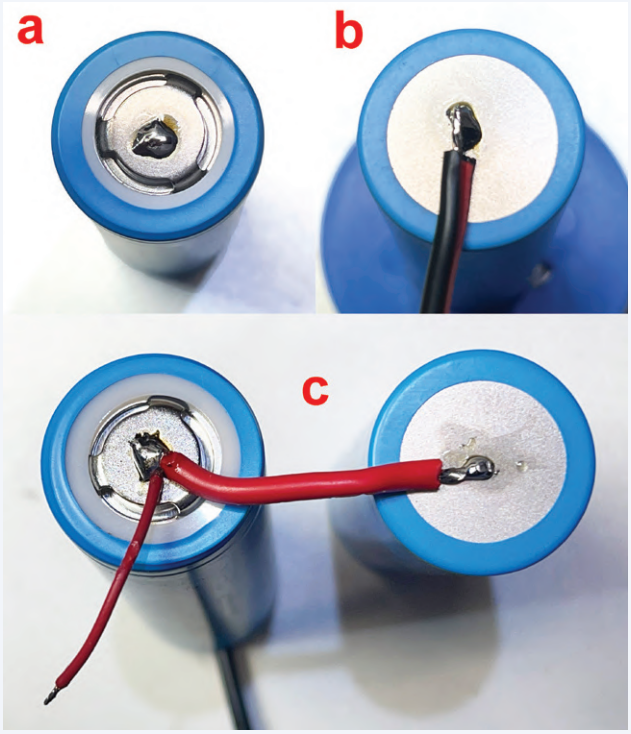


Figure 4. Le pôle positif est plus tolérant à la procédure de soudure (4a). Des fils courts sont soudés au pôle négatif (4b). Les cellules sont finalement connectées en série, et des fils plus fins de mesure de tension sont connectés au BMS (4c).

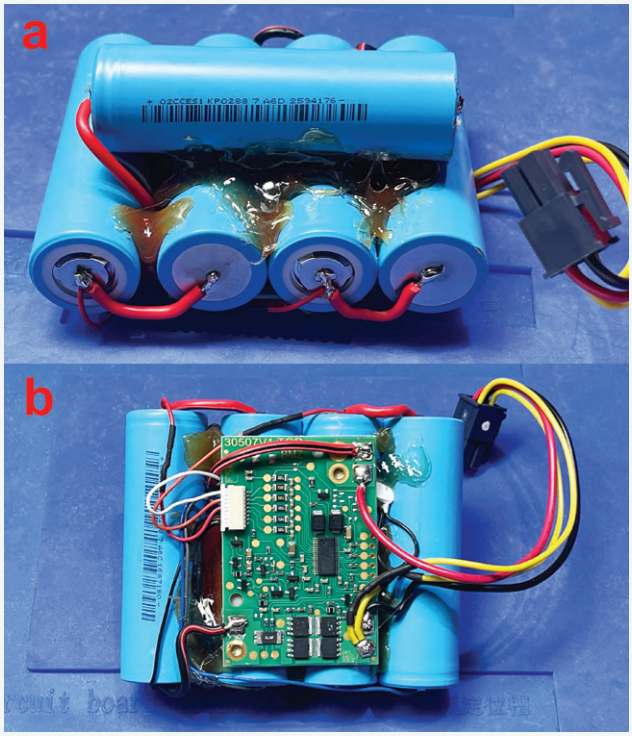


Figure 5. Vue de dessous du pack terminé (5a) et vue de dessus avec la carte BMS (5b).

Comment souder ?

N'ayant pas de poste à souder sous la main, j'ai décidé de souder le fil directement sur les pôles des cellules. En procédant avec prudence, vous pouvez le faire sans endommager les cellules. Le dommage thermique subi par les matériaux constitutifs des cellules est approximativement proportionnel à l'intégrale du temps et de la température. En d'autres termes, vous devez agir rapidement ! Trois choses sont importantes ici. La première : le fer à souder doit être suffisamment puissant pour que la pointe garde sa température pendant la soudure. Ainsi, le pôle de la cellule atteint rapidement la température de soudure. J'ai utilisé un fer à souder de 90 W possédant un régulateur qui permet de régler la température de la panne à plus de 400 °C. Ensuite, il est conseillé d'utiliser une soudure qui fond à une température plus basse. Enfin, il est préférable d'éviter la soudure sans plomb, de température de fusion plus élevée et qui ne mouille pas la surface aussi bien que la bonne vieille SnPb 60/40 que je préfère. D'après mon expérience, les surfaces de contact métalliques des cellules au lithium prennent facilement la soudure. Avec la température de la panne réglée sur 385 °C et l'utilisation de fil de soudure de 1 mm avec flux, j'ai réalisé chaque soudure en une seconde environ – suffisamment vite pour ne pas endommager la cellule.

Si vous gardez la panne sur les pôles de la cellule nettement plus longtemps (parce que le fer à souder n'est pas assez puissant, que la température est trop basse ou que vous utilisez une soudure sans plomb), vous risquez de surchauffer et d'endommager la cellule, ce qui aura un impact sur sa capacité électrique et réduira sans doute son nombre de cycles de charge/décharge. Tandis que si le contact ne dure qu'une seconde environ, il ne devrait pas causer de dommages. Sinon, vous pouvez également acheter des cellules un peu plus chères, munies d'une courte bande de nickel déjà soudée par points sur les pôles de la cellule. Ces bandes peuvent ensuite être soudées ensemble pour constituer le pack sans risque de surchauffe du contenu des cellules. En principe, c'est plus sûr, mais vous devrez isoler les bandes nues pour éviter tout court-circuit qui, en fonctionnement normal, causerait bien plus de dégâts que le bref coup de chaleur nécessaire pour souder les fils directement.

Un gros plan de ma méthode de soudure est visible sur la **figure 4**. Le pôle positif est constitué d'un capuchon métallique fixé à l'électrode de la pile en trois points. Cette disposition augmente la résistance thermique entre la surface de contact extérieure et la structure interne de la cellule, ce qui la rend moins sensible à l'opération de soudure. Commencez par appliquer de petites gouttes de soudure pour étamer rapidement tous les pôles positifs (**fig. 4a**). Ensuite, coupez des bouts de 3 cm d'un fil à brins de 1,5 mm^2, longueur qui laissera suffisamment de mou pour le positionnement des cellules dans le pack. Ces bouts sont soudés directement sur les pôles négatifs (**fig. 4b**) en deux étapes. Tout d'abord, étamez le centre de tous les pôles négatifs avec des petites gouttes de soudure. Ensuite, dénudez et étamez les extrémités des fils de connexion. Une fois les cellules refroidies, soudez rapidement un fil à chacun des pôles négatifs. Dans la **figure 4c**, vous pouvez voir la liaison terminée au pôle positif de la cellule suivante pour réaliser la connexion en série. Le fil rouge plus fin est la connexion de mesure de tension de la cellule à la carte BMS.

Assemblage et test

Le compartiment à piles ne pouvant pas accueillir les cinq cellules disposées sur deux rangées de trois et deux pour former une

configuration en W, j'ai donc dû en trouver une autre. La solution **(fig. 5**) : quatre des cellules sont placées les unes à côté des autres et solidarisées avec de la colle chaude. La cinquième cellule est collée en travers des quatre autres. La colle chaude rend l'assemblage fini très robuste. Le mastic silicone ferait aussi l'affaire.

Comme il adhérait correctement sur le nouveau pack, le ruban adhésif double face à l'arrière de la carte BMS n'a pas eu besoin d'être remplacé. Il ne restait plus qu'à connecter les six fils du connecteur blanc (**fig. 5b**) aux pôles des cellules correspondantes, ainsi que les bornes plus et moins de l'ensemble, à la carte BMS. Cette opération doit être effectuée avant de pouvoir tester la nouvelle batterie. Revérifiez tout le câblage pour vous assurer que vous n'avez pas fait d'erreur. Avoir pris une photo de l'ancienne batterie vous sera utile à ce stade pour tout vérifier. La batterie a fonctionné comme prévu et a pu être chargée et déchargée sans problème.

Avant d'être installé dans le logement de la batterie de la tondeuse, le pack terminé a été enveloppé dans du ruban adhésif pour rendre l'ensemble plus robuste et assurer une isolation et une protection contre l'humidité (**fig. 6**). Après avoir refermé le couvercle et allumé la tondeuse, celle-ci s'est mise à se calibrer avec les signaux du câble de guidage et à commencer à tondre. J'ai annulé cette dernière opération et mis la tondeuse dans sa station de charge. Il lui a fallu trois heures entières pour se charger, ce qui indique que la batterie a maintenant une capacité presque double de celle d'origine.

Le robot tond aussi bien qu'il l'a toujours fait – une heure de tonte suivie d'une heure de charge. La batterie n'est que partiellement déchargée au cours de ces cycles, je suppose donc que ces cellules de plus grande capacité supporteront beaucoup plus de cycles de charge avant de devoir être remplacées. Avec un peu de chance, je pense que le nouveau pack devrait durer deux fois plus longtemps que l'original. Si c'est le cas, ça valait la peine d'y passer trois heures de bricolage. Le salaire horaire pour le temps passé n'est sans doute pas terrible, mais j'ai maintenant une batterie pas facile à trouver dans le commerce !

Cette méthode de remplacement des cellules de la batterie n'est bien entendu pas limitée aux tondeuses à gazon. Vous pouvez l'appliquer pour prolonger la durée de vie des aspirateurs sans fil et d'autres appareils, même si l'espace disponible pour la batterie n'est pas aussi généreux. Le dernier aspirateur où je l'ai utilisée (avec des batteries 18650 à haute capacité) fonctionne depuis trois ans sans rechigner.

210368-04

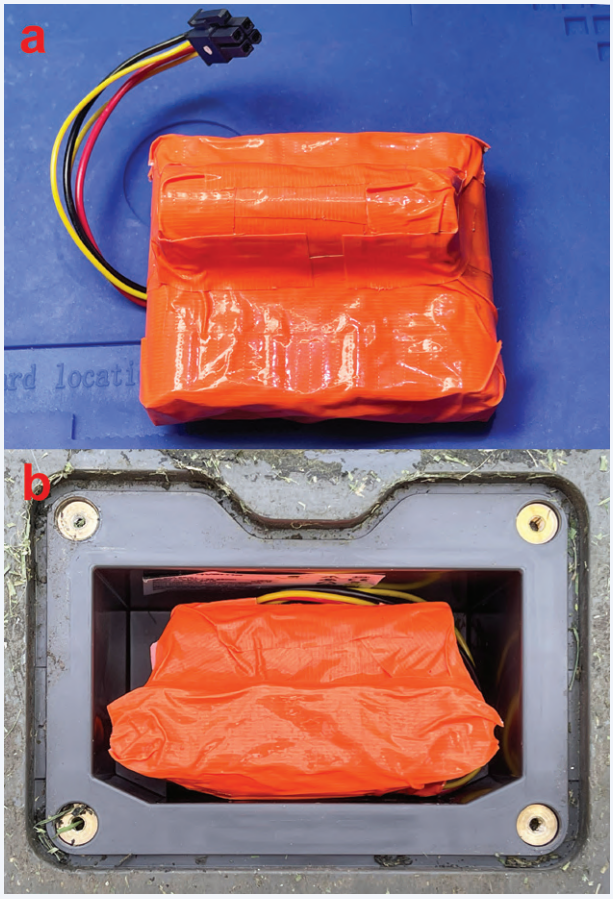


Figure 6. Le nouveau pack enveloppé de ruban adhésif orange (6a) s'insère parfaitement dans son logement (6b).

Des questions, des commentaires ?

Contactez Elektor (redaction@elektor.fr).

Contributeurs

Idée, conception et texte : **Thomas Scherer**
Rédaction : Jens Nickel, Stuart Cording
Traduction : Helmut Müller
Maquette : Giel Dols

PRODUITS

- **OW16B - multimètre numérique avec Bluetooth d'OWON**
  www.elektor.fr/18780
- **Pince ampèremétrique 4350 de PeakTech**
  www.elektor.fr/18161
- **RD6006 - alimentation DC (360 W) de JOY-iT**
  www.elektor.fr/19564

3e partie

Création d'interfaces graphiques en Python
Générateur de mèmes

Créez des mèmes internet à la volée avec la bibliothèque guizero.

MAKER
Laura Sach

Laura dirige l'équipe *A Level* de la Fondation Raspberry Pi chargée des ressources pédagogiques en informatique à destination des étudiants.

@CodeBoom

MAKER
Martin O'Hanlon

Martin crée des cours, des projets et des ressources en ligne au sein de l'équipe *Learning* de la Fondation Raspberry Pi.

@martinohanlon

Partant des bases acquises précédemment, nous allons écrire un programme de création de mèmes. L'idée est simple : nous fournissons en entrée du texte et une image, et le code les combinera à la façon des mèmes internet.

Commençons par créer la fenêtre principale. Elle contiendra deux widgets **TextBox** pour la saisie du texte à afficher en haut et en bas de l'image. Importez d'abord les widgets nécessaires :

```
from guizero import App, TextBox, Drawing
```

Créez ensuite les deux zones de saisie :

```
app = App("meme")

top_text = TextBox(app, "top text")
bottom_text = TextBox(app, "bottom text")

app.display()
```

Ceci étant posé, passons au widget **Drawing** qui contiendra l'image et le texte du mème.

Création du mème

Ajoutez la ligne ci-dessous juste avant `app.display()`. La valeur `fill` donnée à la largeur et à la hauteur du widget **Drawing** le fera occuper toute la fenêtre.

```
meme = Drawing(app, width="fill",
height="fill")
```

Le mème sera créé et affiché dès que l'utilisateur entrera du texte dans une zone de saisie. Confions cette tâche à une fonction appelée `draw_meme()`. Son rôle sera d'effacer le contenu précédent, d'afficher une image (ici la photo d'un pic noir), et d'insérer en haut et en bas de cette image le texte entré.

Comme nous l'avions fait lors de la deuxième partie, nous utilisons la propriété **value** pour récupérer la valeur du widget **Text**. Autrement dit `top_text.value` signifie : « STP Python, retourne la chaîne de texte contenue dans l'objet `top_text` ».

```
def draw_meme():
    meme.clear()
    meme.image(0, 0, "woodpecker.png")
    meme.text(20, 20, top_text.value)
    meme.text(20, 320, bottom_text.value)
```

Dans `meme.image()` et `meme.text()`, les deux premiers nombres passés en argument sont les coordonnées `x` et `y` de l'objet à afficher. L'objet `image` sera ainsi placé au point `(0,0)`, soit au coin supérieur gauche de la fenêtre.

Ajoutez la définition de la fonction `draw_meme()` à votre code, et appelez-la juste avant la ligne `app.display()` :

```
draw_meme()
```

Votre code devrait maintenant être celui du listage **meme1.py**.

Exécutez-le, et modifiez le texte des deux zones

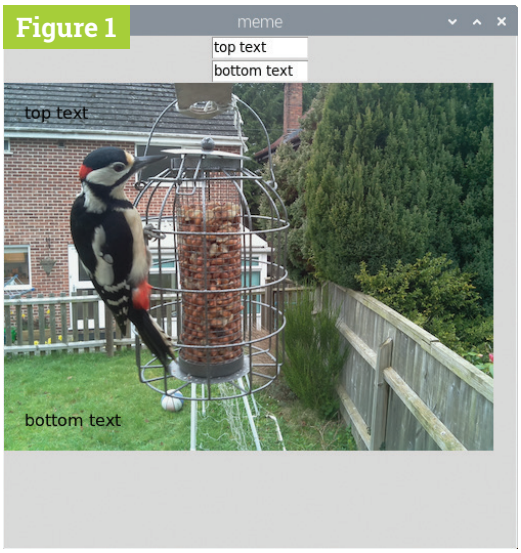


Figure 1 Notre mème avec un texte non stylisé.

de saisie : le mème n'est pas mis à jour (**fig. 1**). Pour qu'il le soit, il faudrait que la fonction `draw_meme()` soit appelée dès que du texte est saisi. C'est justement ce que permet l'instruction `command`, que nous utilisons donc dans nos deux widgets **TextBox**, comme ceci :

```
  top_text = TextBox(app, "top text",
command=draw_meme)
  bottom_text = TextBox(app, "bottom text",
command=draw_meme)
```

> « Le texte affiché est plutôt triste. Modifions son style. »

Le nouveau code est celui du listage **meme2.py**. Lancez-le pour vérifier qu'il fonctionne comme attendu. Le texte affiché est plutôt triste. Modifions sa couleur, sa taille et sa police :

```
  meme.text(
      20, 20, top_text.value,
      color="orange",
      size=40,
      font="courier")
  meme.text(
```

meme1.py

TÉLÉCHARGEZ LE CODE COMPLET :
magpi.cc/guizerocode

Langage : **Python 3**

```
001.  # Imports ---------------
002.
003.  from guizero import App, TextBox, Drawing
004.
005.
006.  # Functions -------------
007.
008.  def draw_meme():
009.      meme.clear()
010.      meme.image(0, 0, "woodpecker.png")
011.      meme.text(20, 20, top_text.value)
012.      meme.text(20, 320, bottom_text.value)
013.
014.
015.  # App -------------------
016.
017.  app = App("meme")
018.
019.  top_text = TextBox(app, "top text")
020.  bottom_text = TextBox(app, "bottom text")
021.
022.  meme = Drawing(app, width="fill", height="fill")
023.
024.  draw_meme()
025.
026.  app.display()
```

meme2.py

Langage : **Python 3**

```
001.  # Imports ---------------
002.
003.  from guizero import App, TextBox, Drawing
004.
005.
006.  # Functions -------------
007.
008.  def draw_meme():
009.      meme.clear()
010.      meme.image(0, 0, "woodpecker.png")
011.      meme.text(20, 20, top_text.value)
012.      meme.text(20, 320, bottom_text.value)
013.
014.
015.  # App -------------------
016.
017.  app = App("meme")
018.
019.  top_text = TextBox(app, "top text", command=draw_meme)
020.  bottom_text = TextBox(app, "bottom text", command=draw_meme)
021.
022.  meme = Drawing(app, width="fill", height="fill")
023.
024.  draw_meme()
025.
026.  app.display()
```

Astuce

Nous avons coupé au niveau des virgules certaines instructions pour faciliter leur lecture, mais pour le compilateur il s'agira bel et bien de la même instruction !

```
    20, 320, bottom_text.value,
    color="blue",
    size=28,
    font="times new roman",
    )
```

Jouez avec les paramètres de votre nouveau code (**meme3.py**) pour trouver un style de texte qui vous plaise (**fig. 2**).

Personnalisation du mème

Profitons maintenant d'autres widgets de la bibliothèque *guizero* pour permettre à l'utilisateur de modifier à sa guise la police, la taille et la couleur du texte. Pour la couleur et la police, nous utiliserons le widget **Combo** de liste déroulante. Pour la taille, le widget **Slider** (glissière, ou curseur) sera parfait. Commençons donc par les importer :

```
from guizero import App, TextBox, Drawing, Combo, Slider
```

Passons ensuite à la création d'un widget **Combo** pour la couleur. Nous l'appelons `color`, et plaçons son code juste après la définition des widgets **TextBox** :

```
bottom_text = TextBox(app, "bottom text", command=draw_meme)
color = Combo(app,
    options=["black", "white", "red", "green", "blue", "orange"],
    command=draw_meme)
```

meme3.py

Langage : **Python 3**

```
# Imports ---------------

from guizero import App, TextBox, Drawing


# Functions -------------

def draw_meme():
    meme.clear()
    meme.image(0, 0, "woodpecker.png")
    meme.text(
        20, 20, top_text.value,
        color="orange",
        size=40,
        font="courier")
    meme.text(
        20, 320, bottom_text.value,
        color="blue",
        size=28,
        font="times new roman",
        )


# App -------------------

app = App("meme")

top_text = TextBox(app, "top text", command=draw_meme)
bottom_text = TextBox(app, "bottom text", command=draw_meme)

meme = Drawing(app, width="fill", height="fill")

draw_meme()

app.display()
```

> « L'ordre d'apparition à l'écran sera le même que celui de la liste. »

Le paramètre `options` fournit les couleurs possibles sous forme de liste. Vous pouvez en ajouter d'autres (cf. la documentation pour leurs noms anglais). Leur ordre d'apparition à l'écran sera le même que celui de la liste. La valeur par défaut sera celle du premier élément, mais vous pouvez aussi la spécifier avec le paramètre `selected` :

```
color = Combo(app,
    options=["black", "white", "red", "green", "blue", "orange"],
    command=draw_meme,
    selected="blue")
```

Une fois que l'utilisateur a sélectionné une couleur, il faut la récupérer et la transmettre au paramètre `color` des deux instructions `meme.text()`. Pour cela nous recourons à nouveau à la méthode `value` :

```
meme.text(
    20, 20, top_text.value,
    color=color.value,
    size=40,
```

Figure 2

▲ **Figure 2** Stylisation du texte.

```
    font="courier")
```

Modifiez aussi l'instruction **meme.text** du widget **bottom_text** (code **meme4.py**), puis procédez de la même façon pour ajouter un deuxième widget **Combo** de sélection de polices. Voici leur liste : `["times new roman", "verdana", "courier", "impact"]`. Utilisez ensuite la méthode **font.value** dans la fonction `draw_meme()` pour que la police affichée soit celle sélectionnée.

Créez de même un widget **Slider** afin que l'utilisateur puisse choisir la taille du texte :

```
size = Slider(app, start=20, end=40,
command=draw_meme)
```

Les paramètres `start` et `end` définissent les bornes de l'intervalle des valeurs possibles (ici `20` et `40`). Là encore, appliquez la méthode **value** à l'objet **size** pour récupérer la valeur de la taille sélectionnée :

```
meme.text(
    20, 20, top_text.value,
    color=color.value,
    size=size.value,
    font=font.value)
```

Votre code devrait ressembler au listage **04-meme-generator.py** et à la **figure 3** une fois lancé.

L'interactivité serait encore plus grande si

meme4.py

Langage : **Python 3**

```
001.  # Imports ---------------
002.
003.  from guizero import App, TextBox, Drawing, Combo, Slider
004.
005.
006.  # Functions -------------
007.
008.  def draw_meme():
009.      meme.clear()
010.      meme.image(0, 0, "woodpecker.png")
011.      meme.text(
012.          20, 20, top_text.value,
013.          color=color.value,
014.          size=40,
015.          font="courier")
016.      meme.text(
017.          20, 320, bottom_text.value,
018.          color=color.value,
019.          size=28,
020.          font="times new roman",
021.          )
022.
023.
024.  # App -------------------
025.
026.  app = App("meme")
027.
028.  top_text = TextBox(app, "top text", command=draw_meme)
029.  bottom_text = TextBox(app, "bottom text", command=draw_meme)
030.
031.  color = Combo(app,
032.                options=["black", "white", "red", "green",
      "blue", "orange"],
033.                command=draw_meme, selected="blue")
034.
035.  meme = Drawing(app, width="fill", height="fill")
036.
037.  draw_meme()
038.
039.  app.display()
```

Widget Drawing

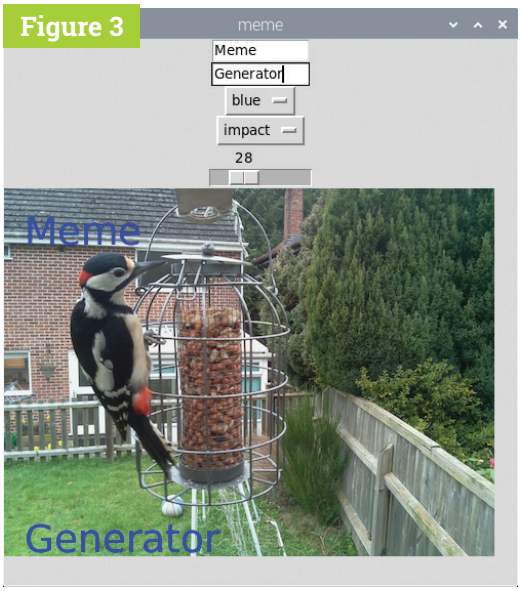
Outre l'affichage d'images et de textes, le widget **Drawing** permet de créer différentes formes géométriques et de les colorer. Pour en savoir plus, consultez l'appendice C du livre **magpi.cc/pythongui** ou la documentation en ligne : **awsie.github.io/guizero/drawing**.

04-meme-generator.py

Langage : **Python 3**

```
001.  # Imports ---------------
002.
003.  from guizero import App, TextBox, Drawing, Combo, Slider
004.
005.
006.  # Functions -------------
007.
008.  def draw_meme():
009.      meme.clear()
010.      meme.image(0, 0, "woodpecker.png")
011.      meme.text(
012.          20, 20, top_text.value,
013.          color=color.value,
014.          size=size.value,
015.          font=font.value)
016.      meme.text(
017.          20, 320, bottom_text.value,
018.          color=color.value,
019.          size=size.value,
020.          font=font.value,
021.          )
022.
023.
024.  # App -------------------
025.
026.  app = App("meme")
027.
028.  top_text = TextBox(app, "top text", command=draw_meme)
029.  bottom_text = TextBox(app, "bottom text", command=draw_meme)
030.
031.  color = Combo(app,
032.                options=["black", "white", "red", "green",
      "blue", "orange"],
033.                command=draw_meme, selected="blue")
034.
035.  font = Combo(app,
036.               options=["times new roman", "verdana", "courier",
      "impact"],
037.               command=draw_meme)
038.
039.  size = Slider(app, start=20, end=50, command=draw_meme)
040.
041.  meme = Drawing(app, width="fill", height="fill")
042.
043.  draw_meme()
044.
045.  app.display()
```

Figure 3



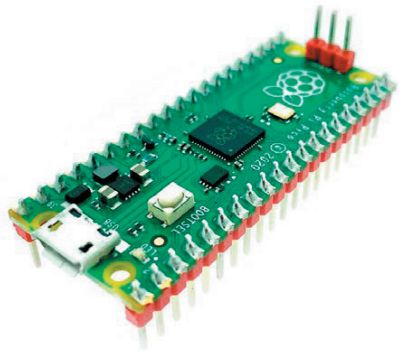
▲ **Figure 3** Le mème et ses widgets de sélection.

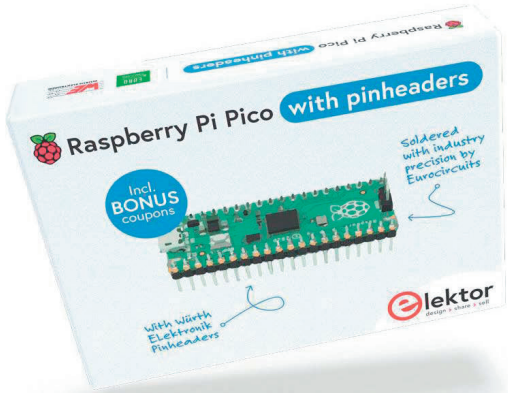
l'utilisateur pouvait choisir une image en la sélectionnant dans une liste (widget **Combo**) ou en saisissant son nom (widget **TextBox**). Sauriez-vous modifier le code en conséquence ? **(VF : Hervé Moreau)**

Note de l'éditeur : cet article est paru initialement dans le magazine MagPi (le magazine officiel du Raspberry Pi).
La maison d'édition Elektor publie ce magazine en néerlandais, allemand et français (www.magpi.fr).

ABONNEZ-VOUS ET RECEVEZ

Raspberry Pi + Headers GRATUIT





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

- Six numéros du magazine MagPi
- Une carte Raspberry avec headers

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

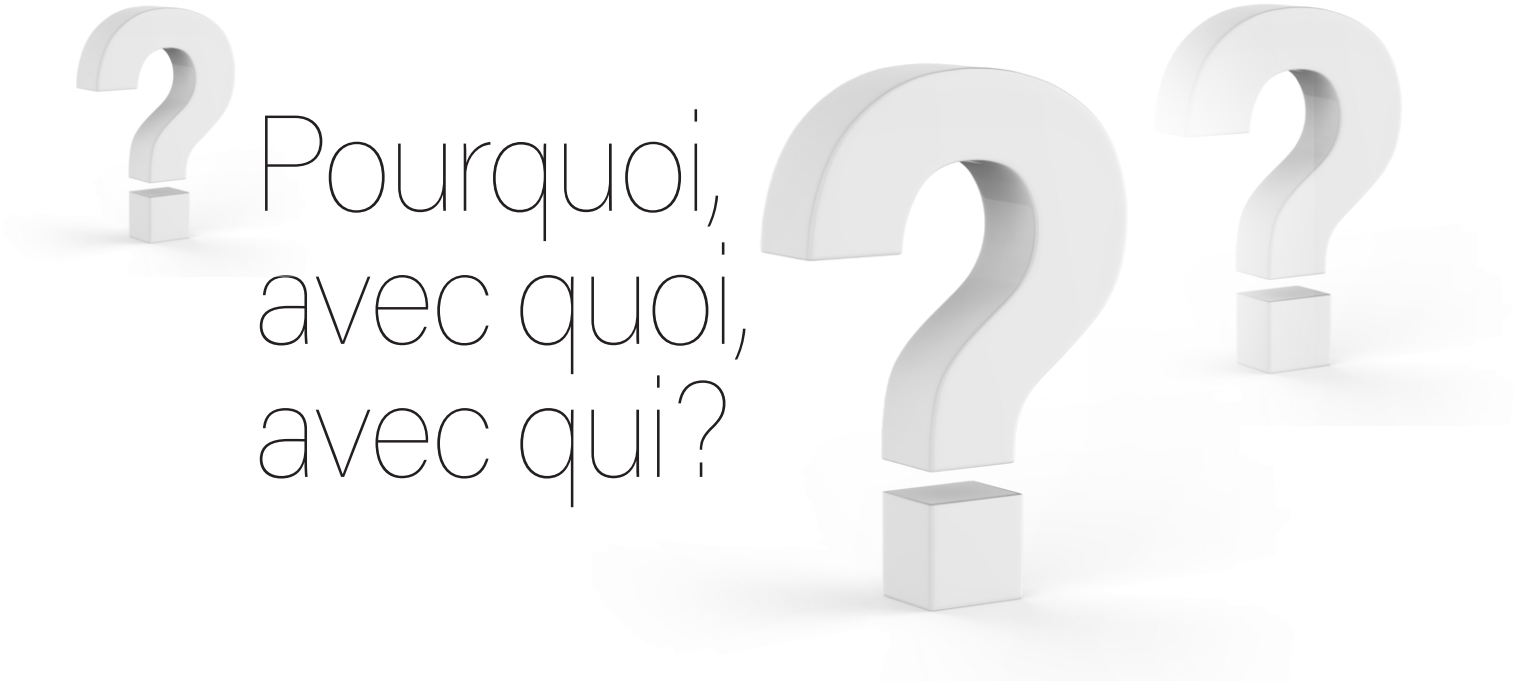
SEULEMENT **54,95 €** PAR AN (6 NUMÉROS)

Vos avantages :

- Prix au numéro réduit
- Chaque numéro directement dans votre boîte aux lettres
- Tous les numéros disponibles sous forme numérique (PDF)
- Découverte de chaque nouveau numéro avant sa sortie en kiosque

ABONNEZ-VOUS : WWW.MAGPI.FR

Trois questions **fondamentales**



Pourquoi, avec quoi, avec qui?

Priscilla Haring-Kuipers (Pays-Bas)

Je pense que les réponses à ces trois questions devraient servir de fondations à l'élaboration de tout produit, qu'il relève du domaine médical, musical, ou encore de l'Internet des Objets.

Pourquoi construisez-vous ce produit ? Pouvez-vous justifier son existence ? Quelles seraient les conséquences s'il n'était pas fabriqué (par vous) ?
Quels matériaux et procédés impliquent sa fabrication ? Que deviendra-t-il lorsqu'il ne servira plus ?
Qui sont les personnes qui le fabriquent ? Comment sont-elles traitées ? Quelle sera la signification de votre produit aux yeux de ceux qui le possèderont ? Quelle valeur humaine vous apporte sa fabrication ?

Permettez-moi de détailler l'origine et le rôle de ces questions avec mon regard de fabricante de synthétiseurs.

Pourquoi

Selon moi, toute personne qui construit quelque chose répond à une pulsion artistique plus ou moins prononcée. Dans notre PME, tout part généralement du besoin de créer un séquenceur musical novateur ou de concevoir un oscillateur à basse fréquence d'un nouveau genre. Si vous jugez qu'art et créativité sont l'expression d'une certaine valeur humaine, alors ce besoin exprimera lui aussi une certaine valeur. Il justifiera à lui seul la fabrication du produit en question (un synthétiseur *Thing* dans notre cas), mais il ne saurait être une raison suffisante pour en construire cent. Je me demande ensuite si le produit né de ce *besoin* ajoutera quelque chose au monde. Je réponds de manière très subjective en regardant ce qui existe déjà sur le marché des synthétiseurs (modulaires), en partant du principe que mon modèle apportera une nouveauté significative. Mon équipe et moi nous demandons ensuite si nos compétences techniques nous permettront d'atteindre cet objectif de nouveauté, mais puisqu'il a été élaboré à partir de nos connaissances et savoir-faire, la réponse est généralement « Oui ».

Avec quoi

Notre équipe s'interroge ensuite sur les conséquences qu'entraînera la fabrication de ce nouveau modèle. Nous examinons les différents choix possibles quant aux

matériaux et procédés nécessaires. Ce faisant nous dessinons – de façon presque imperceptible – la frontière de ce que nous sommes moralement prêts à accepter. Nous sommes conscients de ne pas faire tout ce que nous pourrions faire : nos choix éthiques dépendent d'informations que nous ne possédons pas toujours, mais nous l'acceptons et travaillons avec ce qui est disponible.

La durabilité des produits est pour nous un souci premier. Comme nous prévoyons pour nos synthétiseurs *Thing* une durée de vie comprise entre 20 et 100 ans, tout ce qui les compose doit vivre aussi longtemps, ou au moins pouvoir être remplacé. Outre qu'il doit être utilisable durant des décennies, un instrument de musique électronique se doit d'être réparable. Pour cela nous partageons nos schémas de conception, aidons clients et magasins de musique à réparer nos instruments, ou les invitons à nous les renvoyer pour réparation.

Nous avons choisi d'utiliser de l'aluminium et du bambou partout où nous le pouvions, deux matériaux durables et esthétiques. Le recyclage de l'aluminium est désormais très répandu. L'usine qui fabrique nos boîtiers et panneaux de commande utilise un mélange d'aluminium recyclé à 50 % qui est lui-même recyclable. Le bambou est quant à lui un bois dur dont la culture n'entraîne pas l'abattage de forêts.

Nos synthétiseurs *Thing* ne sont pas assez vieux pour que la question de leur recyclage se pose, mais nous déposons les déchets issus de nos prototypes dans des lieux de recyclage appropriés.

Avec qui et pour qui

La fabrication de nos synthétiseurs *Thing* implique de nombreuses personnes. Celles que nous connaissons le moins sont celles qui sont le plus éloignées de nous. Nous ne savons pas qui extrait nos matériaux, ni qui les transforme en composants. Nous achetons certains de ces composants à des usines de Taïwan, et supposons donc qu'ils y sont fabriqués, mais vraiment nous n'en savons rien. Nos circuits imprimés sont fabriqués à Shenzhen, où nous avons visité plusieurs

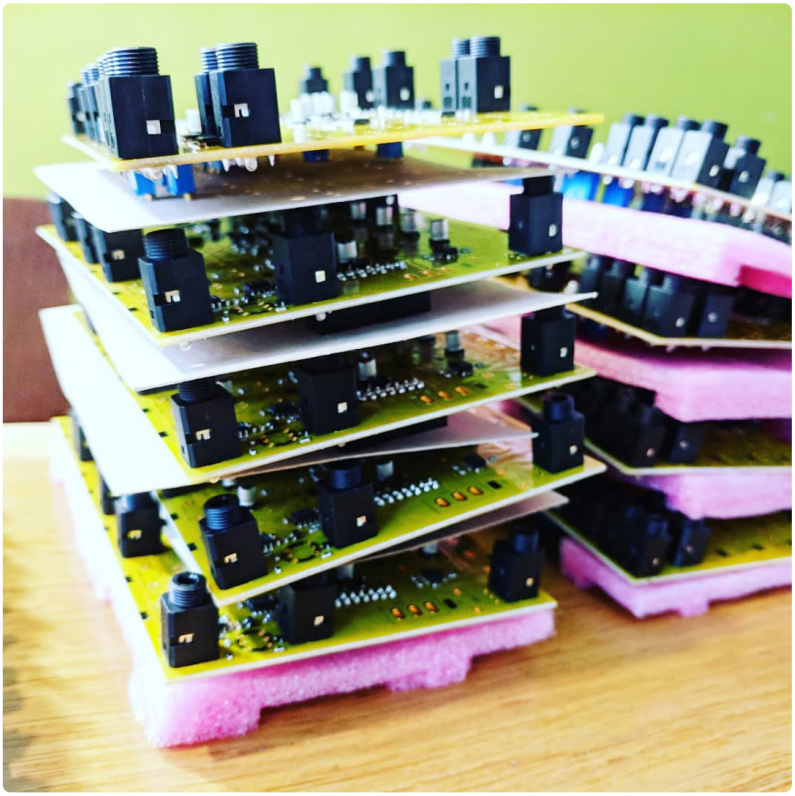


Figure 1. Les cartes de nos modules 'RectangularThing'.

usines et ateliers d'assemblage (dont l'un nous a semblé factice). Les conditions de travail s'améliorent, mais restent loin de ce que nous jugerions acceptable en Europe. Nous avons donc décidé de confier la plupart de nos activités d'assemblage à des entreprises néerlandaises, en partie parce que les conditions de travail y sont éthiques selon nos critères. Nous avons visité ces usines et fait connaissance avec les femmes qui assemblent nos produits.

Nous réfléchissons aussi à l'usage et au rôle de nos synthétiseurs *Thing*. Créer des sons, jouer et écouter de la musique, danser au son de la musique sont d'après nous des activités qui rendent le monde meilleur. Savoir que nous y contribuons est pour nous gratifiant, et nous nous efforçons de faciliter la créativité musicale avec nos produits. C'est pour cela que nous les concevons. Et nous voulons que cet engagement auprès de nos utilisateurs enrichisse également nos vies.

Tout ce puzzle de questionnements doit former un tout économiquement viable. Si nous fabriquons un modèle *Thing* que nous sommes certains de vouloir fabriquer, avec les matériaux que nous voulons et avec les entreprises et personnes que nous voulons, nos clients accepteront-ils le prix auquel nous sommes arrivés ? Ce n'est que si la réponse est « Oui » que nous démarrons une production, généralement de 100 unités.

Et vous, avez-vous déjà abordé la conception d'un produit selon cet angle ? Vos réponses aussi nous intéressent !

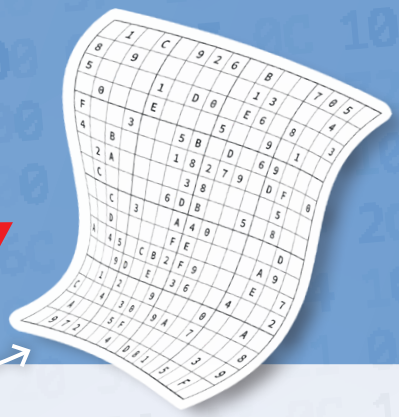
210663-04 – VF : Hervé Moreau

Forum mondial de l'électronique éthique

L'objectif du *Forum mondial de l'électronique éthique* (WEEF, *World Ethical Electronics Forum*) est de réunir les concepteurs du monde entier autour d'un débat public sur l'éthique et les objectifs de développement durable. La première édition s'est tenue le 18 novembre 2021 à Munich. Pour en savoir plus sur le WEEF et la prochaine édition 2022, visitez la page www.elektormagazine.com/weef.

hexadoku

casse-tête pour elektorniciens



La dernière page de votre magazine propose toujours une grille de sudoku particulière. Éteignez le fer à souder, oubliez les écrans, 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 €.

Où envoyer votre réponse ?

Envoyez votre réponse (les chiffres sur fond grisé) avec vos coordonnées par courriel, avant le **15 avril 2022** à l'adresse **hexadoku@elektor.fr**

LES GAGNANTS

La solution de la grille du numéro de janvier/février 2022 est **FA028.**

La liste des gagnants est publiée ici : www.elektormagazine.fr/hexadoku

Bravo à tous les participants et félicitations aux gagnants !

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

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

Tout recours est exclu, de même que le sont, de ce jeu, les personnels d'Elektor International Media et leur famille. Un seul gagnant par foyer.

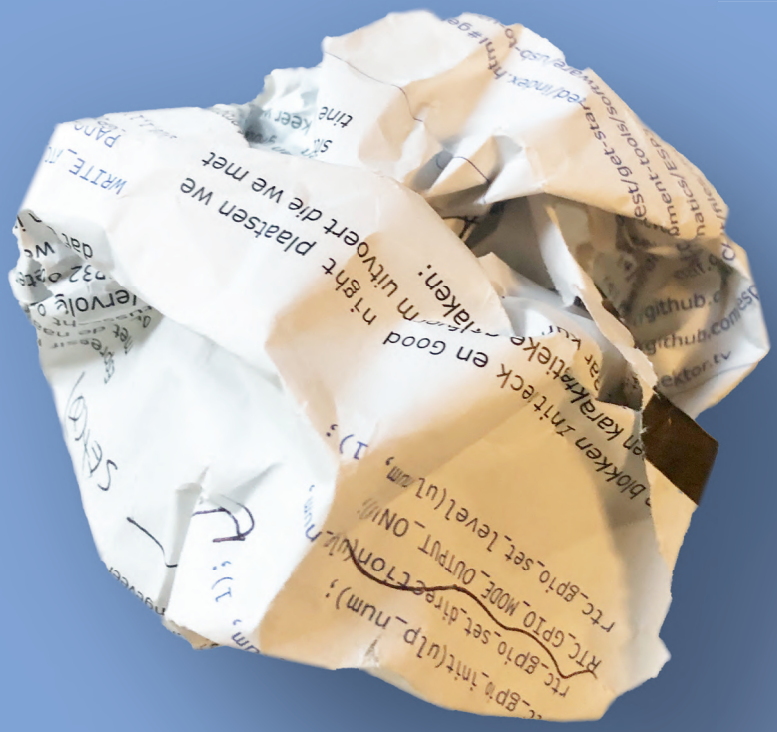
De nombreux outils de développement en un seul endroit

Provenant de centaines de fabricants fiables

Choisissez parmi une vaste sélection de produits sur **mouser.fr/dev-tools**

MOUSER ELECTRONICS

Grrr. Vous êtes resté coincé ?



Vous aimez nos projets d'impression Elektor mais vous avez besoin d'aide ou vous avez une idée, une question ou un commentaire sur un article ?

Pas de soucis. Les ingénieurs d'Elektor, les rédacteurs et les membres de la communauté sont également actifs sur les **médias sociaux**.

Vous pouvez nous trouver ici :

www.elektor.fr/FB

www.elektor.fr/TW

www.elektor.fr/YT

www.elektor.fr/insta

www.elektor.fr/LI

elektor
design > share > sell