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 490

créer > partager > vendre

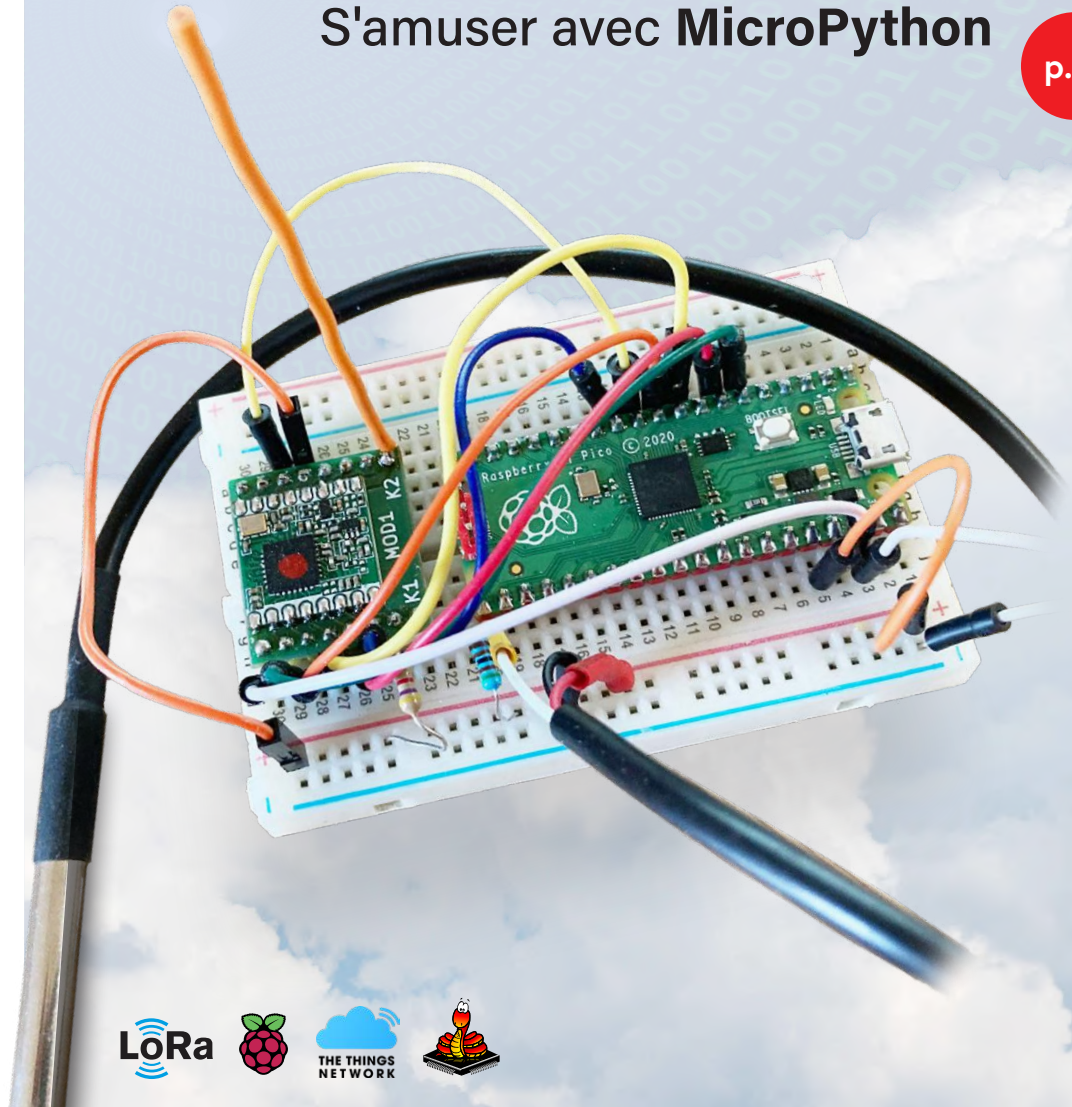
JUILLET & AOÛT 2021
ELEKTORMAGAZINE.FR

60 years young

LoRa avec le Raspberry Pi Pico

S'amuser avec MicroPython

p. 6



LoRa THE THINGS NETWORK

dans ce numéro :

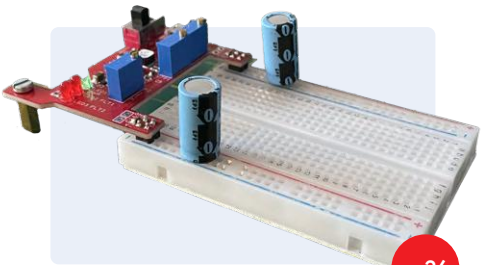
- **60 ans d'Elektor : le très attendu numéro double d'été**
- **Moniteur de la qualité de l'air pour particules de 2,5 µm**
- **MicroPython pour l'ESP32 et ses copains**
- **Extension du module convertisseur élévateur CC-CC MT3608**
- **Propeller 2 de Parallax (3) : faire clignoter une LED**
- **Java sur Raspberry Pi - Partie 2**
- **Raspberry Pi Compute Module 4**
- **Gadget Wi-Fi vestimentaire**
- **ESD – le destroyer fantôme**
- **Énergie solaire pour les robots de tonte**

et bien d'avantage !



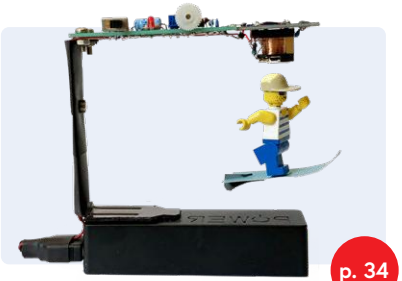
p. 17

RISC-V : quesaco ?
Pourquoi une nouvelle architecture de noyau enthousiasme-t-elle l'industrie ?



p. 26

Module d'alimentation polyvalent pour plaque d'expérimentation
Tensions positives et négatives grâce à un chargeur USB de 5 V



p. 34

Lévitation magnétique sans peine
Sans microcontrôleur !

L 19624 - 490 - F: 15,50 € - RD

Rejoignez la communauté Elektor

Devenez membre GOLD maintenant !



GOLD

- √ accès à l'archive d'Elektor
- √ 10% de remise dans l'e-choppe
- √ 6x magazine imprimé
- √ 6x magazine numérique
- √ des offres exclusives
- √ accès à plus de 1 000 fichiers Gerber
- √ le DVD annuel d'Elektor

Également disponible

abonnement « zéro papier » GREEN !



GREEN

- √ accès à l'archive d'Elektor
- √ 10% de remise dans l'e-choppe
- √ 6x magazine numérique
- √ des offres exclusives
- √ accès à plus de 1 000 fichiers Gerber

www.elektormagazine.fr/membres

44ème année
n° 490 – juillet-août 2021

ISSN 0181-7450
Dépôt légal : juillet 2021
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

DROITS D'AUTEUR :
© 2021 Elektor International Media B.V.

Toute reproduction ou représentation intégrale ou partielle, par quelque procédé que ce soit, des pages publiées dans la présente publication, faite sans l'autorisation de l'éditeur est illicite et constitue une contrefaçon. Seules sont autorisées, d'une part, les reproductions strictement réservées à l'usage privé du copiste et non destinées à une utilisation collective, et, d'autre part, les analyses et courtes citations justifiées par le caractère scientifique ou d'information de l'oeuvre dans laquelle elles sont incorporées (Loi du 11 mars 1957 -art. 40 et 41 et Code Pénal art. 425).
Certains circuits, dispositifs, composants, etc. décrits dans cette revue peuvent bénéficier de droits propres aux brevets; la Société éditrice n'accepte aucune responsabilité du fait de l'absence de mention à ce sujet. Conformément à l'art. 30 de la Loi sur les Brevets, les circuits et schémas publiés dans Elektor ne peuvent être réalisés que dans des buts privés ou scientifiques et non commerciaux. L'utilisation des schémas n'implique aucune responsabilité de la part de la Société éditrice. La Société éditrice n'est pas tenue de renvoyer des articles qui lui parviennent sans demande de sa part et qu'elle n'accepte pas pour publication. Si la Société éditrice accepte pour publication un article qui lui est envoyé, elle est en droit de l'amender et/ou de le faire amender à ses frais; la Société éditrice est de même en droit de traduire et/ou de faire traduire un article et de l'utiliser pour ses autres éditions et activités, contre la rémunération en usage chez elle.

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



rédacteur en chef d'Elektor Magazine

Cuisine à la sauce Elektor

Vous avez probablement vécu la même expérience : vous avez une excellente idée de montage, et vous découvrez que quelqu'un d'autre a déjà fait une grande partie du travail préparatoire. La plupart du temps, vous n'avez pas besoin de réinventer la roue. Au lieu de cela, vous pouvez vous rabattre sur des modules logiciels et matériels prêts à l'emploi, et les combiner (après une dose d'adaptation) pour donner vie à votre idée. Pour le premier projet décrit en détail dans ce numéro, mon collègue Mathias Claussen a suivi cette voie. Son nœud de capteurs LoRa peut collecter des données de mesure sur le terrain et les transmettre sur de longues distances, ce qui peut être utile aussi bien en extérieur pour surveiller l'environnement qu'à l'intérieur dans un système de domotique. Les ingrédients qu'il a utilisés sont les suivants : une carte Raspberry Pi Pico avec un capteur de température connecté, un module LoRa de SeeedStudio, une bibliothèque MicroPython LoRa, The Things Network comme plateforme dans le nuage et l'outil de développement Node-RED, qui peut être installé sur un Raspberry Pi ou un PC. Dans le style typique d'Elektor, nous ne présentons pas seulement le résultat final, mais nous proposons un cours de cuisine, afin que vous sachiez ce qui se passe et que vous puissiez adapter ou affiner la recette en fonction de vos préférences. Le tout est servi sur un platine d'expérimentation, mais nous vous fournissons aussi toutes les informations nécessaires pour réaliser un circuit imprimé. C'est aussi une habitude d'Elektor de proposer non seulement des menus opulents, mais aussi des plats plus légers. Si vous êtes encore novice en matière de MicroPython, je vous recommande l'article de la page 92. Une petite carte ESP32 est suffisante pour faire vos premiers pas. Et à partir de la page 30, vous apprendrez à utiliser la carte Pico susmentionnée pour allumer une LED à partir d'un smartphone.

Restez à l'écoute et en pleine forme !

Elektor a 60 ans : joignez-vous à la fête !



Elektor célébrera ses 60 ans de publications et d'innovations dans le monde de l'électronique avec quelques projets spéciaux passionnants. Nous travaillons sur un livre anniversaire, un film, un événement en direct (*World Ethical Electronics Forum*) et le « labo domestique mobile d'Elektor ». Nous sommes certains que cela vous intéresse. Restez à l'écoute, nous vous donnerons plus d'informations dans les semaines à venir. Vous avez des idées pour des projets spéciaux ? Envoyez-moi vos idées : denise.bodrone@elektor.com

Denise Bodrone, coordinatrice du comité « Elektor 60 »

notre équipe

Rédacteur en chef : Jens Nickel
Rédaction : Eric Bogers, Jan Buiting, Rolf Gerstendorf, Thomas Scherer, Clemens Valens, Mariline Thiebaut-Brodier (coordination)
Service aux lecteurs : Ralf Schmiedel
Correcteur technique : Malte Fischer
Laboratoire : Mathias Claussen, Ton Giesberts, Luc Lemmens, Clemens Valens (responsable)
Maquette : Giel Dols, Harmen Heida

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



Rubriques

3 **Édito :** cuisine à la sauce Elektor

52 **Zone D**
Extension du module convertisseur élévateur CC-CC MT3608

63 **Démarrer en électronique… (8)**
Les condensateurs : suite et fin

90 **Sur le vif**
Le futur était meilleur dans le passé

99 **Drôles de composant(s)**
Composants à couplage de charge dans les oscilloscopes

112 **Questions d'éthique**
L'Europe tente de dompter les GAFA

114 **Hexadoku**
casse-tête pour elektorniciens



Articles de fond

17 **RISC-V : quesaco ?**
Pourquoi une nouvelle architecture de noyau enthousiasme-t-elle l'industrie ?

22 **60 ans d'Elektor**
Le très attendu numéro double d'été

40 **Propeller 2 de Parallax (3)**
Faire clignoter une LED

55 **Banc d'essai : DT71 de Minware**
Brucelles de mesure numériques

66 **Petits circuits avec l'écosystème Qwiic**

70 **Java sur Raspberry Pi**
Partie 2 : commande des broches GPIO avec un service REST de Spring

80 **Raspberry Pi Compute Module 4**
Un Raspberry Pi industriel

100 **ESD – le destroyer fantôme**
Foudroiement spontané des composants

105 **Énergie solaire pour les robots de tonte**
Écologique, peu coûteux, simple !







Réalisations

6 **LoRa avec le Raspberry Pi Pico**
S'amuser avec MicroPython

26 **Module d'alimentation polyvalent pour plaque d'expérimentation**
Tensions positives et négatives grâce à un chargeur USB de 5 V

30 ***Raspberry Pi Pico Essentials***
Extrait : Wi-Fi avec le Raspberry Pi Pico

34 **Lévitation magnétique sans peine**
Sans microcontrôleur !

45 **Programmation des cartes Nucleo avec STM32CubeIDE**
Extrait : FreeRTOS pour le MCU STM32

58 **Gadget Wi-Fi vestimentaire**
ESPHome à nouveau à la manœuvre !

85 **Moniteur de la qualité de l'air, portable et autonome, pour particules de 2,5 µm**
Gardez un œil sur votre santé

92 **MicroPython pour l'ESP32 et ses copains**
Partie 1 : installation et premiers programmes



Bientôt dans ces pages

Le numéro de septembre-octobre 2021 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.

Quelques-uns des points forts :

- Charge électronique en CC et CA
- Système de caméra DIY pour Raspberry Pi
- Thermostat ESP32
- Lévitation magnétique sans peine, la solution numérique
- Boussole électronique
- MicroPython sur ESP32 : afficheur matriciel à LED
- Afficheurs dans les projets Raspberry Pi
- Traitement d'image pour les débutants avec Nvidia Jetson Image Processing

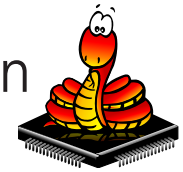
et bien d'avantage !

Ce numéro paraîtra le 3 septembre 2021.

elektor
créer > partager > vendre

LoRa avec le Raspberry Pi Pico

S'amuser avec MicroPython



Mathias Claußen (Elektor)

Se servir de MicroPython sur un Raspberry Pi Pico est une solution sympathique pour s'initier à la programmation. En y ajoutant un module LoRa RFM95 de SeeedStudio et un capteur de température DS18B20, on peut créer en un tournemain un nœud LoRaWAN avec MicroPython. Et comme le câblage de composants sur des plaques d'essai peut manquer de fiabilité, nous vous proposons un circuit imprimé.



Figure 1. Le Raspberry Pi Pico.



Figure 2. Module RFM95.

Le Raspberry Pi Pico (**fig. 1**) est une nouvelle carte à microcontrôleur, équipée du RP2040. Cette puce est conçue et développée en interne par la Fondation Raspberry Pi et il s'agit donc de leur premier silicium. Après tout le battage dont elle a fait l'objet – sans oublier les doutes et la fascination qu'elle suscite – je me suis dit qu'il était temps de lancer quelques projets. La carte ne dispose pas du Wi-Fi, mais peut fonctionner avec un jeu de piles pour des applications portables, alors pourquoi ne pas combiner le RPi Pico avec un modem LoRa ? Le RFM95 de HOPERF (**fig. 2**) est un module bon marché et largement répandu qui a été utilisé dans de précédents projets d'Elektor, comme « LoRaWAN : décollage facile » [1]. Alors que dans cet article, nous utilisions un module STM32 BluePill, cette fois-ci nous avons retenu le RPi Pico.

Nous construirons un dispositif simple de mesure de la température avec un capteur DS18B20, qui enverra les données à une passerelle LoRaWAN qui les transmettra au nuage du réseau *The Things Network*. De là, nous récupèrerons les données avec un RPi (classique) grâce à la plateforme domotique Node-RED.

Nous réaliserons le nœud LoRa de mesure de la température sur une plaque d'essai, mais si vous préférez un circuit imprimé, cet article vous en fournira une ébauche, réalisée avec KiCad.

La **figure 3** montre le flux de données entre le RPi Pico et le serveur de *The Things Network*, via le réseau LoRaWAN. Vous pouvez voir qu'une passerelle se trouve au milieu pour traduire les données hertziennes en quelque chose qui peut être acheminé sur l'internet vers *The Things Network*. Comme nous sommes intéressés par la température transmise, nous devons traiter les données déposées dans *The Things Network*. Pour ce faire, nous allons les récupérer et les présenter sur une petite page web. La **figure 4** montre le flux de données de *The Things Network* vers un RPi qui créera une page web avec la température transmise actuelle ainsi qu'un graphique montrant les dernières valeurs reçues.

Le logiciel Node-RED se chargera de récupérer les données du réseau *The Things Network* et de produire une page web. C'est un outil qui permet de construire sous forme graphique des flux de données et de les traiter. Cela peut se faire simplement avec le navigateur de votre choix – il vous suffit d'installer le serveur Node-RED et d'ouvrir un navigateur

pour commencer. Node-RED a déjà été utilisé dans des projets d'Elektor et permet d'accéder rapidement au traitement et à la manipulation de données. Un livre d'initiation à Node-RED est disponible dans la boutique d'Elektor (voir **Produits** pour plus d'informations).

Les ingrédients de ce projet sont simples et figurent dans l'encadré **Matériel requis**. Assurez-vous également que vous êtes à portée d'une passerelle LoRaWAN qui acheminera vos données vers *The Things Network*. Si vous n'en avez pas à proximité et que vous souhaitez démarrer avec votre propre passerelle pour avoir une meilleure couverture LoRaWAN, jetez un œil à [3]. Il s'agit d'une solution tout-en-un que vous pouvez acheter dans le commerce. Vous pouvez également construire votre propre passerelle basée sur un RPi avec [4].

D'abord la plaque d'essai

Pour mettre au point notre montage, il sera d'abord réalisé sur une plaque d'essai. Cela permet d'effectuer des tests rapides et, comme vous pouvez le voir, avec un câblage simple (fig. 5). Pour connecter le modem LoRa au RPi Pico, nous avons besoin de quatre connexions pour l'interface SPI de base : MOSI, MISO, SCK et CS (*data in*, *data out*, *clock* et *chip select*). Cela permettra l'échange de données avec le modem LoRa. RESET et DIO0 sont nécessaires en plus pour commander le modem. Les broches DIO1 à DIO3 sont connectées et requises par certaines autres bibliothèques LoRa, comme la bibliothèque LMIC (si nous programmons le RPi Pico en C/C++). Nous pouvons choisir l'un des ports SPI dont dispose le RPi Pico et le connecter au modem LoRa RFM95. Il suffit également de connecter les broches RESET et DIO0 du module à l'une des broches GPIO du Pico. Pour obtenir un modem LoRa pleinement opérationnel, il faut également fixer une antenne.

Pour l'antenne, nous utiliserons un simple fil. Un morceau de fil de cuivre de 1 mm de diamètre suffit parfaitement. Pour calculer la longueur de l'antenne quart d'onde pour 868 MHz (gamme dans laquelle le module LoRa fonctionne), on a la formule : $\lambda/4 = (c_0/868\ \text{MHz})/4 = (299.792.458\ \text{m/s})/(868.000.000\ 1/\text{s}*4) = 0{,}08635\ \text{m} = 8{,}635\ \text{cm}$. Cette longueur s'applique pour la propagation dans le vide, mais dans le cuivre la vitesse de l'onde est plus faible, ce qui implique un facteur de raccourcissement. Avec un coefficient de 0,95, on obtient finalement une longueur approximative de 8,2 cm pour le fil de cuivre.

Bien que facultative, la résistance de rappel sur la ligne de réinitialisation s'est avérée nécessaire dans le passé. L'ajouter permet d'avoir un fonctionnement plus stable, car le RESET peut capter du bruit par les fils volants et réinitialiser le module de manière inattendue. Si vous travaillez sur votre propre carte LoRa, il peut être plus sûr de la prévoir, sans la câbler, quitte à la mettre plus tard à la main sur votre carte.

Pour le capteur de température, on utilisera une connexion One-Wire. Comme le nom One-Wire l'indique, une seule broche GPIO suffit pour l'échange de données. Le protocole peut être implémenté de façon purement logicielle, donc on peut utiliser n'importe quelle broche GPIO. Le protocole One-Wire nécessite la présence d'une résistance de rappel. Selon l'UC utilisée, il peut s'agir d'une résistance interne,

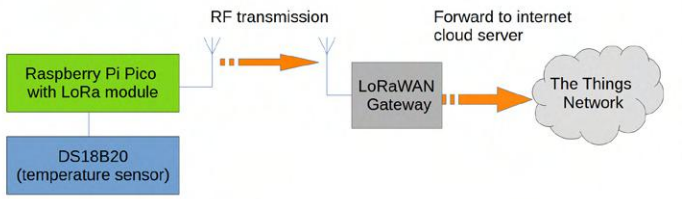


Figure 3. Flux de données vers The Thing Network.



Figure 4. Flux de données entre The Thing Network et la page web de l'utilisateur.

Matérie requis

- RFM95*
- Jeu de plaques d'essai*
- 9 fils de liaison mâle-mâle
- Carte BoB pour RFM95
- Connecteur à 2×20 broches
- Connecteur à 2×8 broches
- Capteur de température DS18B20
- Fil de cuivre de 10 cm (diamètre 1 mm

Les pièces marquées d'un * peuvent être commandées dans la boutique d'Elektor, voir **Produits**.

Pour la carte d'extension (BoB), des fichiers Gerber sont fournis [15] pour votre fabricant de circuits imprimés favori

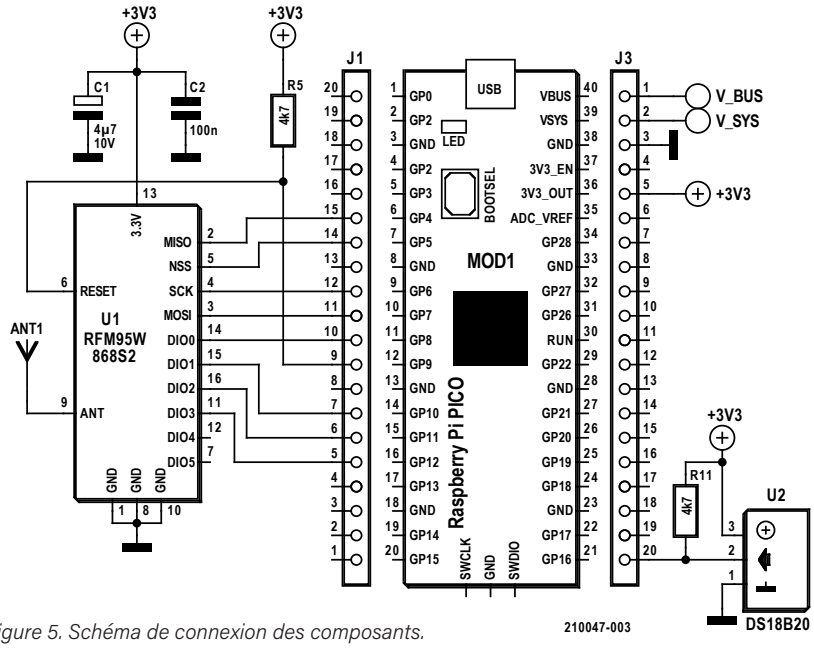


Figure 5. Schéma de connexion des composants.

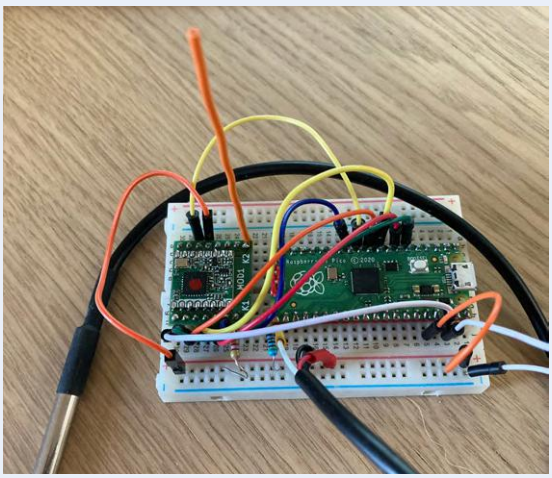


Figure 6. Installation sur une plaque d'essai.

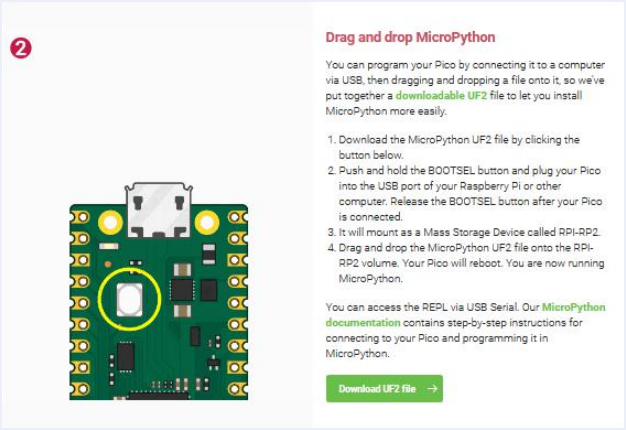


Figure 7. Téléchargement de MicroPython pour le Raspberry Pi Pico.

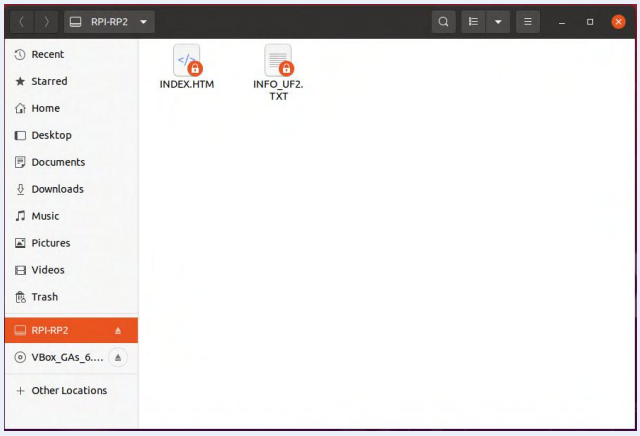


Figure 8. Raspberry Pi Pico en mode bootloader.

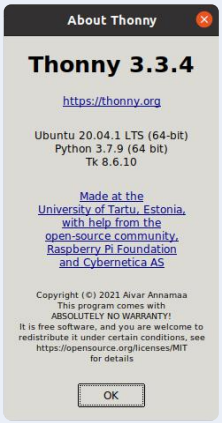


Figure 9. Thonny installé en version 3.3.4.

fournie par le bloc GPIO de votre UC, ou d'une résistance externe. Pour avoir un transfert de données fiable, une résistance de rappel externe est préférable.
Une fois tout le matériel installé, cela ressemblera à ce que vous voyez sur la **figure 6**, passons au codage.

MicroPython et LoRa

Nous avons déjà programmé en C/C++ pour d'autres projets LoRa, cette fois nous choisissons MicroPython. Le RPi Pico est une carte bien adaptée à l'enseignement, et l'utilisation de MicroPython facilitera encore un peu plus les choses. Si vous ne connaissez pas MicroPython sur le RPi Pico, consultez le livre de Gareth Halfacree et Ben Everard, *Get Started with MicroPython on Raspberry Pi Pico* (Raspberry Pi Foundation, 2021). Si vous souhaitez obtenir une copie papier, rendez-vous sur la boutique d'Elektor [5]. Si vous préférez un livre électronique, vous pouvez obtenir une copie gratuite en anglais sur [6].
La combinaison de LoRa et MicroPython a été réalisée sur d'autres plateformes, mais cela présente certaines difficultés. MicroPython est un langage interprété comme l'était le BASIC sur le C64 ou d'autres ordinateurs domestiques. L'accès s'en trouve facilité, mais au prix d'une surcharge de l'UC pendant l'exécution et donc certaines limitations.
La bibliothèque que nous utiliserons est une version légèrement modifiée de uLora par fantasticdonkey que l'on trouve sur GitHub à [7]. Cette bibliothèque est elle-même une variante de la bibliothèque TinyLoRa pour CircuitPython d'Adafruit. Les limitations mentionnent que nous ne pouvons que transmettre des données, sans rien recevoir en retour du LoRaWAN. Avec cette limitation, nous avons également dû utiliser l'APB (activation par personnalisation) pour l'authentification, ce qui signifie que la clé de session du réseau et la clé de session de l'application sont stockées dans notre code.
Comme ce n'est qu'un exemple pour montrer que l'on peut faire du LoRa avec MicroPython sur le Raspberry Pi Pico, ce n'est pas parfait, mais cela permet de débuter rapidement. Les fichiers modifiés nécessaires peuvent être téléchargés depuis la page GitHub d'Elektor [12]. Notez également que même si le logiciel semble fonctionner, la stabilité peut être un problème qui devra être résolu à un moment donné.

Installer MicroPython sur le RPi Pico

Le Raspberry Pi Pico est livré avec un chargeur de démarrage qui permet de changer facilement le micrologiciel en cours d'exécution. Comme nous aurons besoin de MicroPython, nous installons la dernière version disponible sur [8] (cf. fig. 7). Téléchargez le fichier UF2 depuis la page web et préparez votre RPi Pico à entrer en mode *bootloader*.
Pour entrer en mode *bootloader*, déconnectez le RPi Pico, appuyez sur le bouton BOOTSEL et reconnectez-le à votre ordinateur. Un nouveau périphérique de stockage de masse doit alors apparaître, comme sur la **figure 8**. Faites glisser le fichier UF2 téléchargé dans ce lecteur, ce qui installera la version courante de MicroPython.
Après un redémarrage, vous serez prêt à continuer, ou du moins votre RPi Pico le sera. Pour le développement, ce serait bien d'avoir un éditeur ou un EDI. L'étape suivante est la configuration de Thonny comme EDI Python.

Installer Thonny

L'installation de l'EDI Thonny sur une Ubuntu 20.04 récente ou même sur Windows ne prend que quelques clics. Pour Windows, vous pouvez récupérer un programme d'installation à l'adresse [9]. Pour Ubuntu, vous pouvez utiliser `wget -q -O - https://github.com/thonny/thonny/releases/download/v3.3.4/thonny-3.3.4.bash` pour obtenir le programme d'installation de la version 3.3.4 avec prise en charge du RPi Pico. Après avoir téléchargé le fichier, exécutez-le avec `bash thonny-3.3.4.bash`. Si l'installation réussit, vous aurez l'EDI Thonny en version 3.3.4 installé (cf. **fig 9**).

Après son premier démarrage, nous devons configurer le RPi Pico. Cela se fait à partir du menu avec *Outils → Options* qui ouvrira la boîte de dialogue de configuration (cf. **fig. 10**). Lorsque les outils et le matériel sont en place, il est temps de passer au code.

Mise en place du code

Pour LoRa, il est important de savoir dans quelle région du monde vous vous trouvez. La bande ISM utilisée peut être dans la gamme des 433 MHz, 868 MHz ou 915 MHz. Pour l'Europe, il s'agit principalement de 868 MHz alors qu'aux États-Unis c'est 915 MHz. Ce projet est conçu pour l'Europe, il utilisera donc le plan de fréquence 868 MHz, et si vous le reproduisez, assurez-vous de l'adapter à votre zone géographique. Comme la bibliothèque uLoRa est modifiée pour être utilisée sur le RPi Pico, vous devez ouvrir et enregistrer *ulora.py*, *ttn_eu.py*, *ulora_encryption.py* et *lora.py* sur votre RPi Pico dans l'EDI Thonny. Pour le fichier *ttn_eu.py*, s'il n'est pas applicable à votre zone, vous devez enregistrer le *fichier ttn_xx.py* correspondant sur votre carte. Cela fournira à la carte les bibliothèques requises, et un exemple simple que nous allons expliquer maintenant.

Dans le fichier *lora.py* se trouve la logique principale de ce projet. La **figure 11** détaille le déroulement du programme. Après l'initialisation, on recherche si un capteur de température est connecté au bus One-Wire. S'il n'y en a pas, après 120 s de sommeil, il y a une nouvelle tentative pour en trouver un. Si un capteur est présent, sa température est lue puis transmise. La façon dont la valeur est codée n'est pas très efficace en termes de charge utile, mais sera plus tard utile pour la démonstration. La **figure 12** montre les données transmises dans la console de The Things Network. Après une transmission, nous devons attendre 900 s pour respecter la politique d'utilisation équitable de The Things Network. Si vous souhaitez démarrer votre code automatiquement sur votre RPi Pico, enregistrez le fichier *lora.py* sous le nom de *main.py*.

Vous pouvez remarquer dans le code qu'une valeur est écrite dans un fichier appelé *current.txt*. Il s'agit du compteur de trame utilisé pour la transmission vers The Things Network. Si nous ne sauvegardions pas cette valeur, elle serait remise à zéro à chaque redémarrage du nœud,

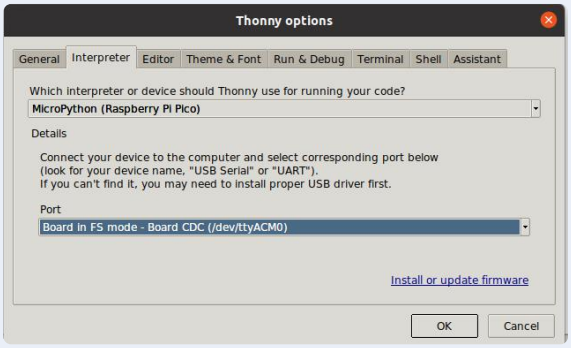


Figure 10. Paramètres de connexion pour le Raspberry Pi Pico.

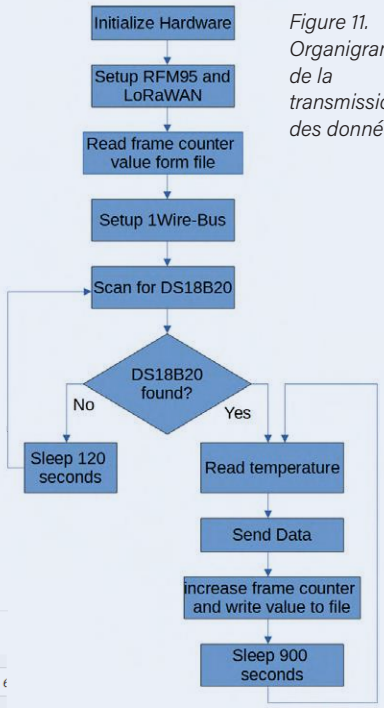


Figure 11. Organigramme de la transmission des données.

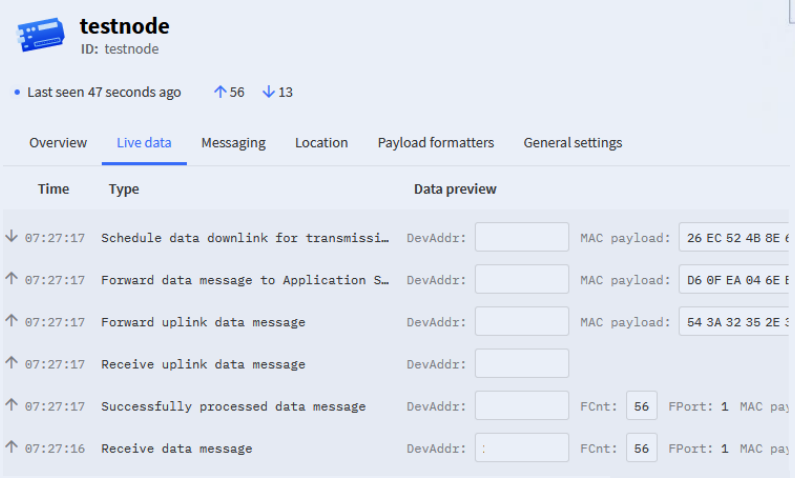


Figure 12. Données reçues à l'intérieur de la console TTN.

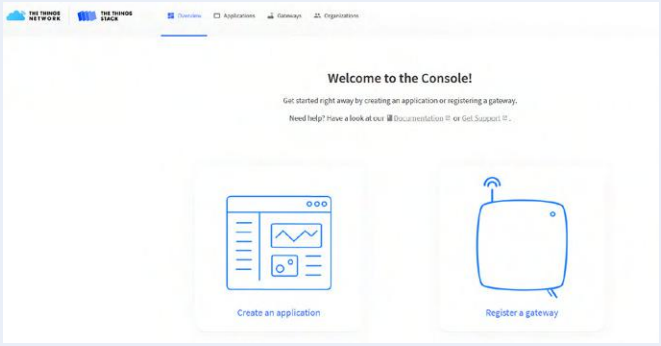


Figure 13. Créer une nouvelle application.

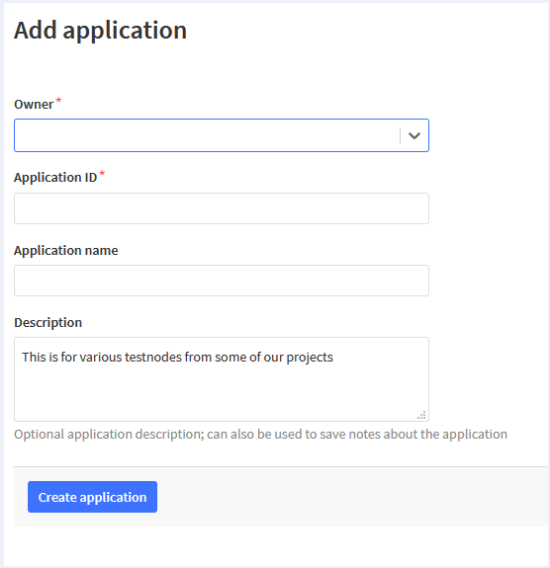


Figure 14. Assistant pour ajouter une nouvelle application.

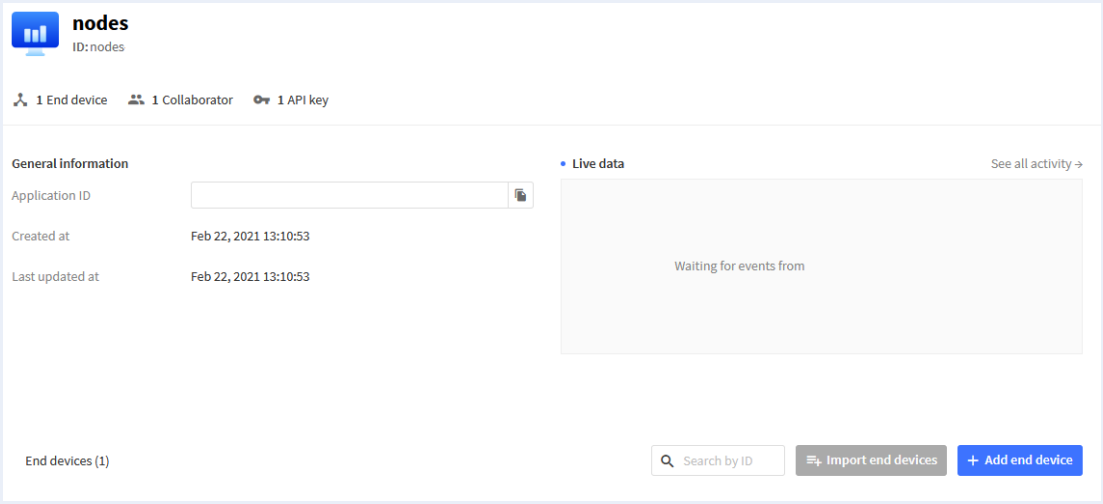


Figure 15. Ajouter un nouveau dispositif.

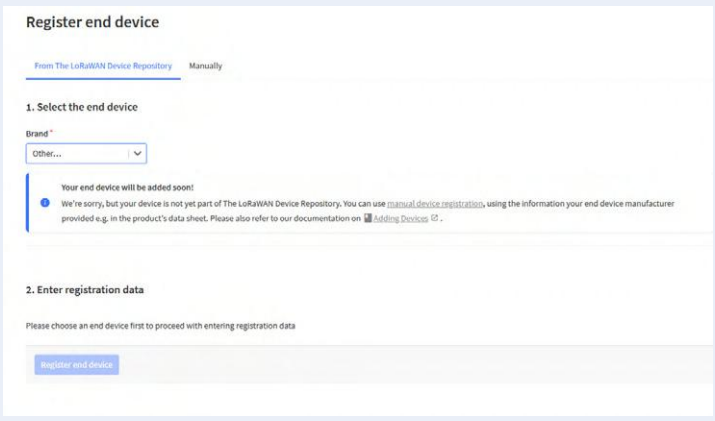


Figure 16. Assistant pour créer un nouveau nœud.

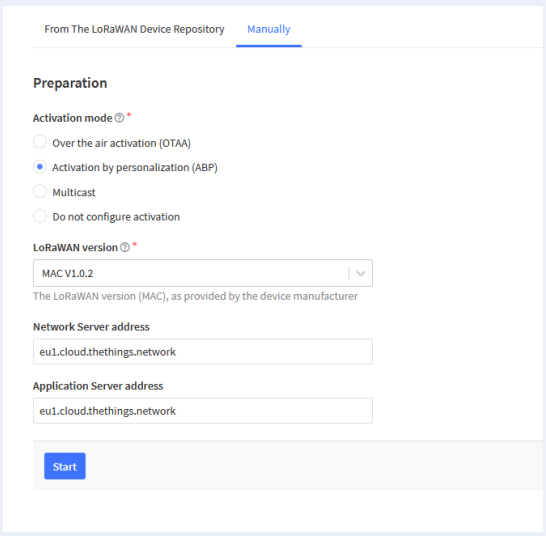


Figure 17. Paramètres du mode ABP.

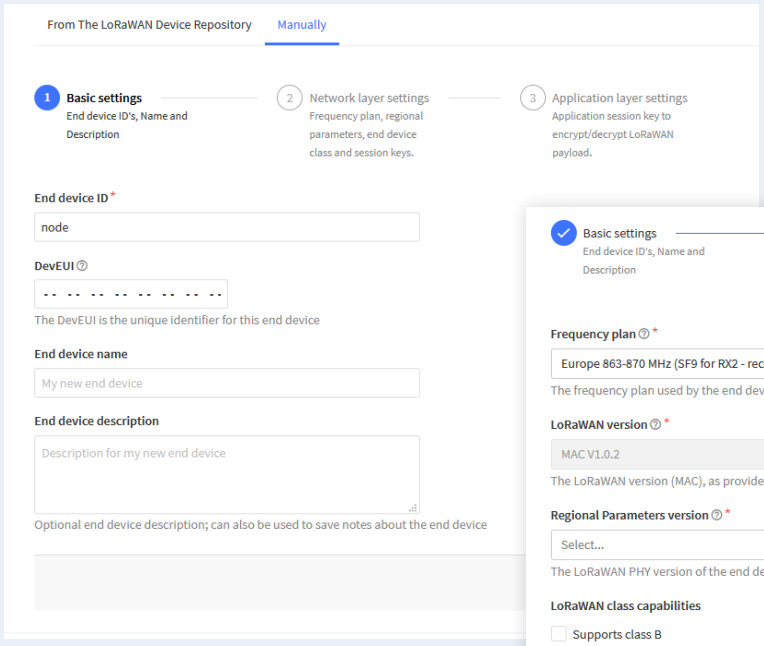


Figure 18. Paramétrage du nom et de l'EUI.

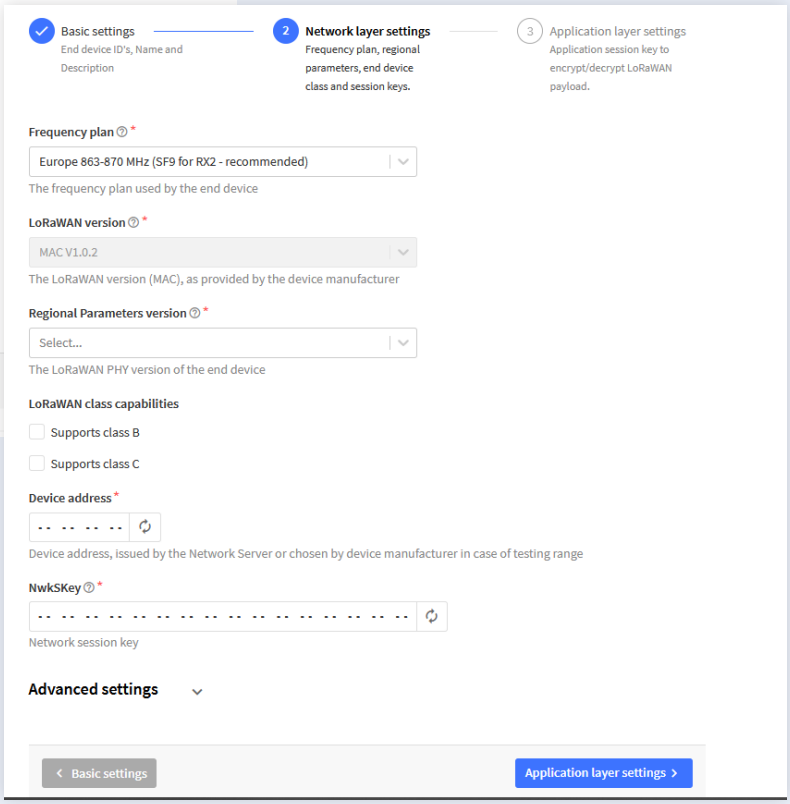


Figure 19. Paramètres du plan de fréquence.

par ex. après un changement de batterie. Si le compteur était remis à zéro à chaque redémarrage de l'appareil, les données transmises seraient rejetées par The Things Network pour des raisons de sécurité. Après chaque transmission, le fichier est mis à jour avec la valeur courante utilisée pour la recharger au redémarrage suivant. Ce n'est pas idéal et réduira la durée de vie de la mémoire flash. Si nous écrivons une nouvelle valeur toutes les 15 min, cela signifie 96 écritures par jour, ou 35.040 écritures par an, ce qui est loin d'être parfait et doit être amélioré.

Vers The Things Network et retour

De précédents articles expliquent comment configurer un nœud LoRa, mais je voudrais ajouter quelques mots à ce sujet. L'article précédent présentait la pile The Things Network dans sa version 2. La version 3 a récemment été annoncée, et dans certains endroits vous pouvez déjà migrer vos applications et passerelles dans cette nouvelle version et donc la console.

Pour que les données soient transmises, il faut ajouter quelques paramètres importants aux fichiers MicroPython fournis. Nous devons avoir un compte sur The Things Network pour utiliser l'infrastructure communautaire. Lorsqu'un compte a été créé, ou si vous souhaitez en créer un, vous pouvez vous rendre sur la pile de la version 3 en utilisant [10]. Après la connexion à notre compte, nous devons créer un nouveau nœud. Pour cela, nous devons d'abord configurer une application. Comme le montre la **figure 13**, cliquez sur *Create an application* et remplissez le formulaire de la **figure 14**. Sélectionnez le propriétaire de cette application, dans ce cas votre compte, et renseignez l'*Application ID*. Cliquez sur *Create application* pour terminer le processus.

Une nouvelle application ayant été configurée, nous pouvons y ajouter un nouveau nœud. Celui-ci fournira les informations d'identification que nous devons entrer dans le script MicroPython. Dans votre application nouvellement créée, sélectionnez *Add end device* pour lancer un assistant de création d'un nouveau nœud (**fig. 15**).

L'assistant (**fig. 16**) vous demandera de sélectionner un appareil. Comme le nôtre ne sera pas dans la liste, sélectionnez *Manually*. Comme sur la **figure 17**, choisissez *Activation by personalization* (ABP) et sélectionnez la version LoRaWAN pour notre MAC. Ici, vous pouvez choisir MAC V1.0.2. Sélectionnez *Start* pour accéder aux paramètres de base (**fig. 18**). Renseignez l'*End device ID*, un identifiant unique pour votre nœud. Les champs *End device name* et *End device description* sont utilisés pour vous permettre de distinguer ultérieurement les nœuds que vous avez créés. La page suivante (**fig. 19**) vous demande de sélectionner le plan de fréquences. Réglez-le en fonction de votre zone. Comme le nœud que nous réalisons se contente de faire du LoRaWAN pour transmettre des données, nous ne prenons pas en charge la classe B ou la classe C. Sélectionnez *Application layer settings* pour continuer. Générez une clé de session d'application (**fig. 20**) et terminez avec *Add end device*. Le nouveau dispositif créé apparaîtra (**fig. 21**). Nous avons besoin pour notre script MicroPython de la *Device address*, de la *NwkSKey* et de l'*AppSKey*. Celles-ci seront placées dans le fichier *lora.py*, où DEVADDR

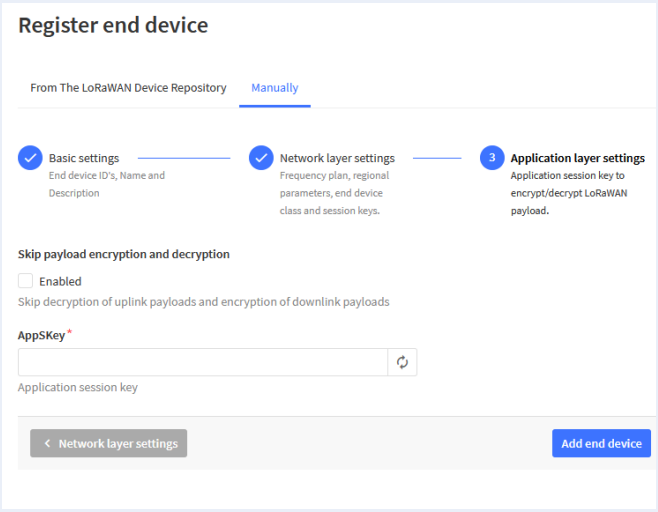


Figure 20. Génération de la clé d'application.

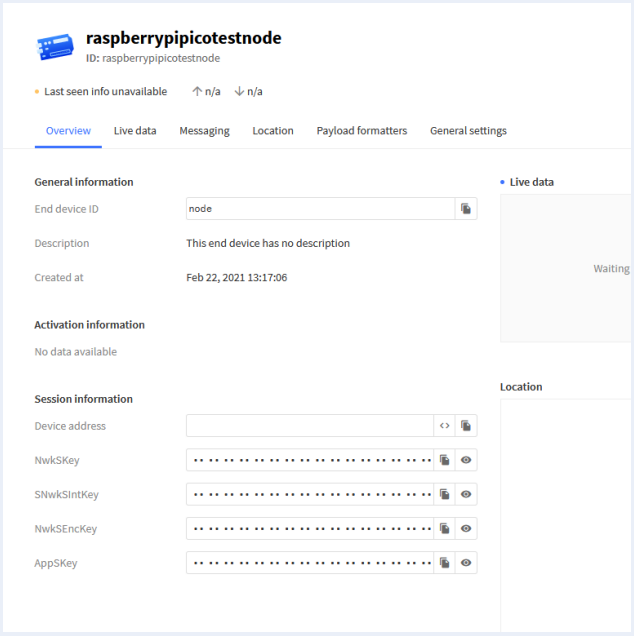


Figure 21. Informations sur le dispositif pour le nouveau nœud.

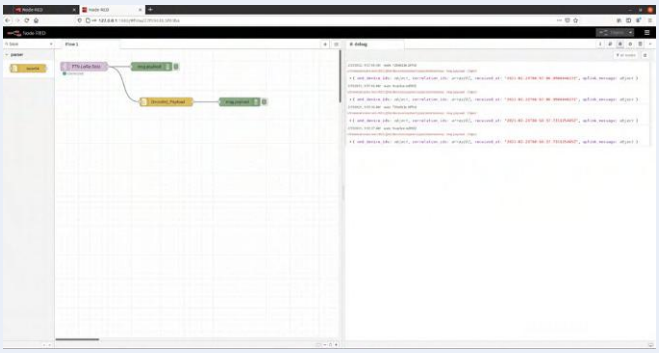


Figure 22. Flux de base Node-Red pour les données LoRaWAN.

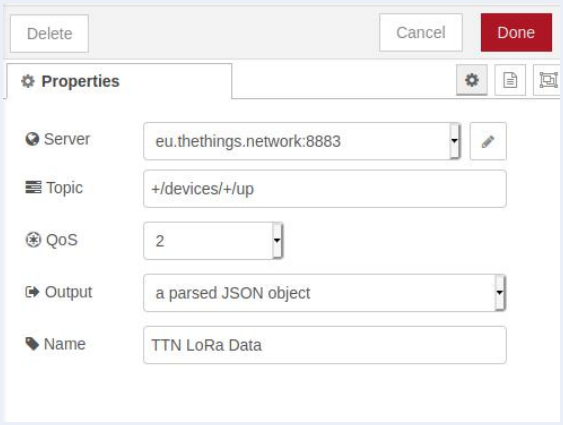


Figure 23. Paramétrage du serveur pour le nœud MQTT.

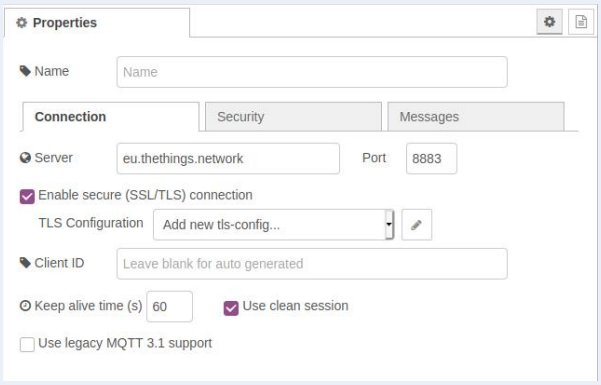


Figure 24. Paramètres du serveur et de SSL.

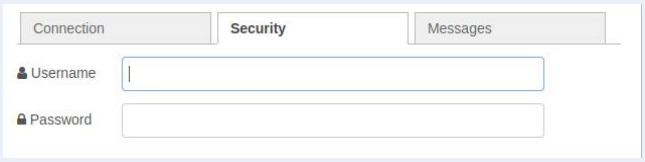


Figure 25. Paramétrage du nom d'utilisateur et du mot de passe.

contiendra l'adresse du dispositif, `NWKEY` contiendra NwkSKey et `APP` AppSKey. Si toutes les données sont entrées correctement, la transmission apparaîtra dans The Things Network.

Récupérer les données avec Node-RED

La commande suivante, dans un terminal ou via une connexion par SSH, permet d'installer Node-RED sur un Raspberry Pi :

```
bash <(curl -sL https://raw.githubusercontent.
com/node-red/linux-installers/master/deb/
update-nodejs-and-nodered
```

Pour que Node-RED fonctionne comme un service, vous devez également entrer et exécuter la commande suivante après son installation :

```
sudo systemctl enable nodered.service
```

Le guide d'installation complet est disponible ici [11]. Pour démarrer le service, exécutez la commande suivante dans un terminal : `sudo systemctl start nodered.service`.
Une fois que Node-RED est configuré et fonctionne, nous devons bâtir un flux comme celui de la **figure 22** pour récupérer nos données transmises. Bien que des nœuds pour The Things Network existent, sur un RPi vous devez utiliser la « connectivité » générique MQTT. L'utilisation des nœuds pour The Things Network sur un RPi peut entraîner des erreurs et des plantages de votre flux de données. Sans doute des problèmes de compatibilité avec les nœuds fournis, car ils fonctionnent bien avec du matériel à base de X86.
Pour le connecteur MQTT, nous sommes intéressés par les données que notre nœud va transmettre, appelées *payload*. La **figure 22** montre le flux complet de Node-RED pour récupérer les données que nous transmettons. La configuration requise commencera par le nœud MQTT. Ici, nous devons renseigner le serveur qui fournira nos données, et les informations d'identification.
La **figure 23** montre le *Serveur* et le *Topic* qui doivent être saisis. Un *Topic* peut être considéré comme un espace de conversation sur un sujet défini. De cette manière, seuls les messages pour lesquels il existe un intérêt sont fournis. Nous utilisons `+/devices/+/up` comme *Topic*. Cela signifie que nous sommes intéressés par tous les messages qui nous sont envoyés par les noeuds enregistrés dans notre application. Comme sortie, nous attendons un objet JSON décodé pour un traitement ultérieur. Dans l'étape suivante, il est nécessaire de définir les préférences pour les serveurs de The Things Network. Pour modifier les paramètres d'un serveur, cliquez sur le bouton crayon à droite du serveur. Une nouvelle boîte de dialogue s'affiche **(fig. 24**). Comme nous fonctionnons sur la nouvelle pile V3, utilisez *eu1.cloud.thethings.network* et le port *8883*. Assurez-vous que l'option *Enable secure (SSL/TLS) connection* est cochée. Sous l'onglet *Security* (cf. **fig. 25**), vous devez entrer un nom d'utilisateur et un mot de passe. Comme nom d'utilisateur, utilisez le nom de l'application que nous avons créée précédemment dans la console The Things Network.
Obtenir le mot de passe implique un peu plus de travail. Rendez-vous sur la console The Things Network et connectez-vous à votre application. Cliquez sur l'entrée de menu *API Keys* sur la gauche et commencez à ajouter une *API Key* avec *Add API key.* Un nouvel assistant (cf. **fig. 26**) apparaîtra. Choisissez les droits d'accès requis

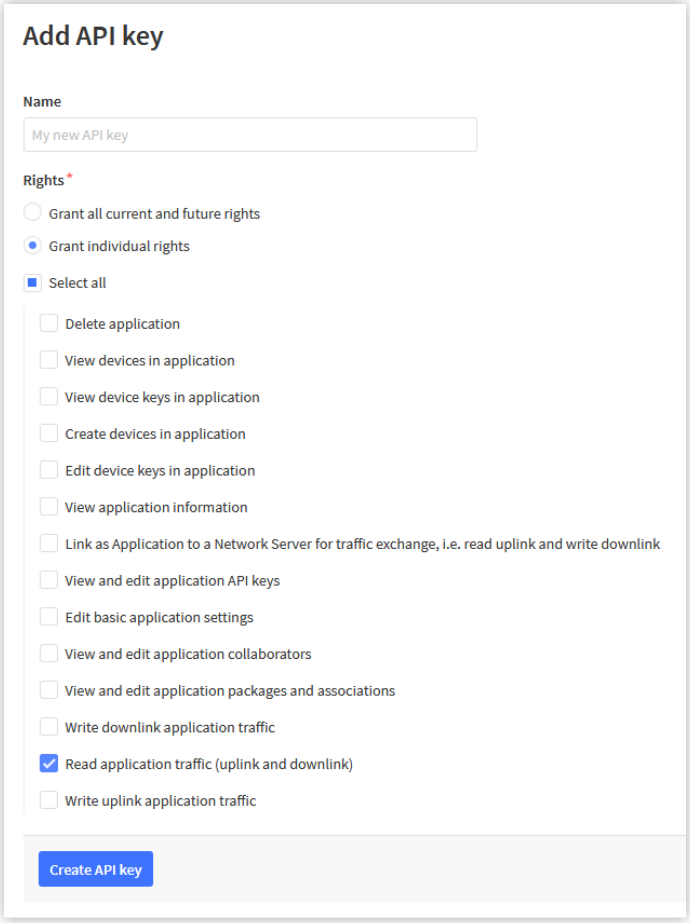


Figure 26. Assistant pour une nouvelle clé d'API.

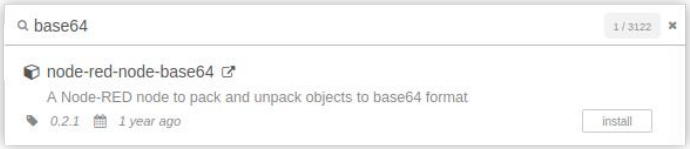


Figure 27. Paramètres du décodeur BASE64 dans Node-RED.

et terminez le dialogue avec *Create API Key*. L'étape suivante vous présentera une nouvelle clé API que vous pourrez utiliser. Veillez à la stocker dans un endroit sûr, car vous ne pourrez plus y accéder par la suite. Cette clé est le mot de passe que nous devons utiliser pour Node-RED.
Si tous les paramètres sont en place, les changements seront appliqués avec *Done*. Si maintenant vous utilisez ces paramètres, le nœud MQTT établira une connexion et de nouvelles données lui seront présentées. Seul inconvénient avec les données que nous recevons : la charge utile, les données que notre nœud envoie, est codée avec BASE64. Pour récupérer les octets bruts transmis, nous devons insérer un décodeur BASE64 qui fera la conversion. S'il est configuré comme dans la **figure 27**, les nœuds suivants pourront accéder aux données sous forme de tableau d'octets.
Pour plus de facilité, nous avons configuré un flux de démonstration comme celui de la **figure 28** pour afficher depuis un navigateur web les dernières données arrivées. La **figure 29** montre l'aspect final de la page web. Ici, il vous suffit d'entrer vos informations d'identification.

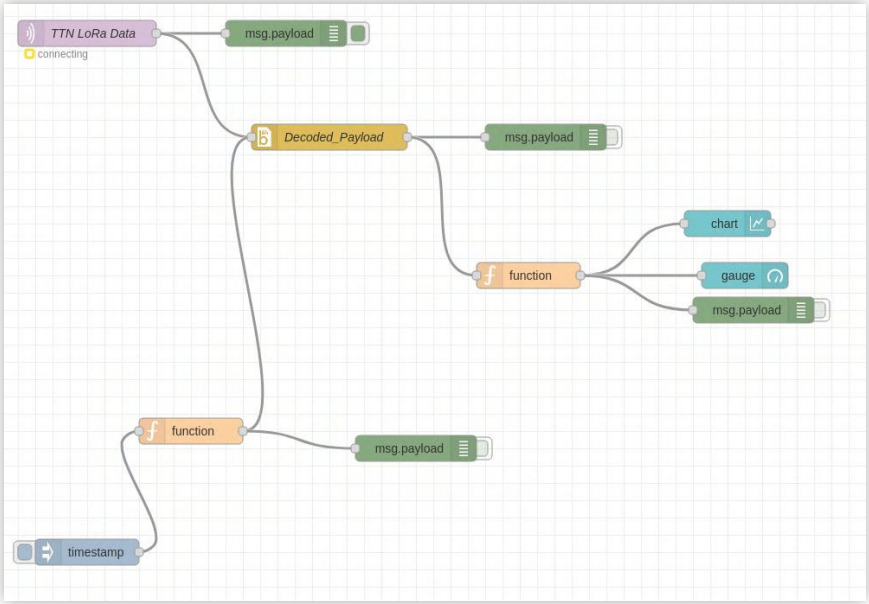


Figure 28. Exemple de flux de données dans Node-RED.

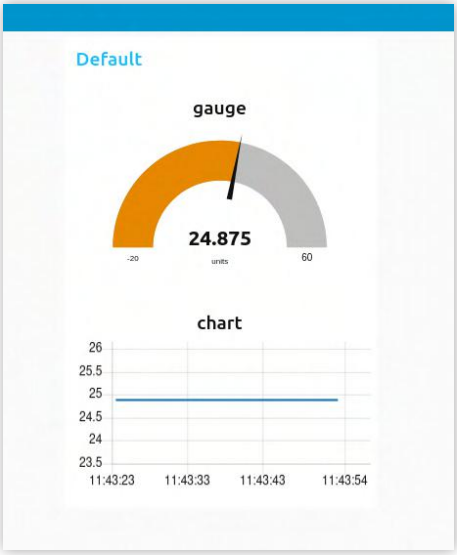


Figure 29. Données présentées sous forme de page web.

Simple et adapté aux débutants

Bien que ce ne soit pas le projet le plus complexe que vous puissiez réaliser, il vous permettra de commencer à bricoler avec le RPi Pico et LoRaWAN. L'utilisation de MicroPython peut être, surtout pour les débutants, une transition en douceur vers le monde des systèmes embarqués. Le temps nécessaire à la construction et à l'exécution de ce projet le rend adapté aux salles de classe (virtuelles) et à l'enseignement. Mais nous vous avions promis un circuit imprimé. Si vous ne touchez pas aux composants, vous pouvez vous arrêter là. Si vous jouez du fer à souder, continuez !

Un mot d'avertissement

Habituellement, lorsque nous présentons des schémas et des circuits imprimés dans Elektor, ils ont été câblés et testés (au moins pour vérifier qu'ils ne risquent pas de prendre feu ou de tuer de jolis chatons). Pour ce circuit imprimé et ces schémas, les choses sont différentes. Le circuit imprimé, et donc le schéma, est un travail en cours. Bien qu'il devrait fonctionner, il n'a pas encore été testé, donc soyez conscient qu'il peut y avoir quelques bugs. Vous pouvez télécharger tous les fichiers KiCad de ce montage depuis la page Elektor ou le dépôt GitHub d'Elektor [12] pour le modifier, l'adapter à vos propres besoins. Si vous avez vu des défauts de conception ou des choix de conception stupides (ce qui ne veut pas dire quelque chose comme l'utilisation d'un RPi Pico), n'hésitez pas à nous faire part de vos suggestions. Cela ne se limite pas aux défauts. Si pour vous, il manque quelque chose, n'hésitez pas à suggérer des changements. Comme je l'ai déjà mentionné, il s'agit d'un travail en cours. Vos commentaires sont les bienvenus.

Schéma : partie 1

Les schémas sont divisés en deux parties, même si tous les composants sont sur une seule feuille KiCad. La première partie concerne la connexion du RFM95, notre émetteur-récepteur LoRa, et du capteur de température DS18B20 au RPi Pico. Pour le RFM95, nous avons besoin d'une liaison SPI, composée de *MISO*, *MOSI*, *SCK* et *nCS*. En outre, nous devons ajouter RESET et DIO0 pour un fonctionnement minimal. La **figure 5** montre le schéma de connexion du RFM95 au RPi Pico. Notez également que nous avons ajouté deux condensateurs, 100 nF (C2) et 4,7 µF (C1), à côté du RFM95. Les condensateurs doivent fournir les pics de courant si le RFM95 est en mode transmission ou réception, comme recommandé par la fiche technique.

Vous pouvez également voir R5, une résistance de 4,7 kΩ sur la ligne de réinitialisation. Normalement, elle n'est pas nécessaire, car le RFM95 offre une résistance de rappel interne. L'utilisation du module dans plusieurs projets a montré qu'une résistance de rappel externe évite les réinitialisations indésirables qui se produisent de temps à autre sans elle. Les éléments qui ne sont pas utilisés par le logiciel MicroPython, mais qui peuvent l'être ultérieurement par les bibliothèques C/C++ sont *DIO0*, *DIO1*, *DIO2* et *DIO3*. Alors que DIO0 est nécessaire pour le fonctionnement général de la plupart des bibliothèques LoRa, les autres DIO fournissent des fonctions optionnelles. D'autres bibliothèques peuvent utiliser ces fonctions.

Pour le DS18B20 (fig. 3), nous alimentons « volontairement » la broche VCC. Le protocole One Wire permet d'avoir une alimentation passive en utilisant seulement la broche de données et la masse (alimentation parasite). Certaines cartes non authentiques se comportent bizarrement lorsqu'elles ne sont pas alimentées via VCC. Par conséquent, les trois broches, y compris VCC, sont connectées, en cas d'installation involontaire d'un de ces capteurs contrefaits. Soyez donc prudent : si vous achetez par erreur des capteurs DS18B20 contrefaits, les performances peuvent en souffrir, ou les alimenter avec la seule broche de données ne fonctionnera pas du tout. Le protocole One Wire impose une résistance de rappel de 4,7 kΩ sur la ligne de données.

Le RPi Pico est un module, vous n'avez donc pas besoin de connecter de nombreux composants. Ce qui est différent, ce sont les étiquettes V_BUS et V_SYS attachées aux broches 40 et 39. Elles seront utilisées pour connecter une batterie rechargeable au dispositif, ce qui nous amène à la deuxième partie du schéma.

Schéma : partie 2

La deuxième partie du schéma (**fig. 30**) concerne la section pour une batterie rechargeable. Elle consiste en un chargeur au lithium MCP73871-1CC avec trajet de courant. Ainsi, si nous fournissons de l'énergie à son entrée, il la transmettra à sa sortie, et n'utilisera pas du tout la batterie. Si l'alimentation externe est déconnectée ou insuffisante, une commutation automatique vers la batterie se produira. Le chargeur est actuellement configuré pour charger des batteries de 1000 mA.

Pour protéger la batterie attachée, un circuit XB8089D, déjà utilisé dans le kit du *superchargeur* [14], est inclus. Il empêchera la surcharge, la

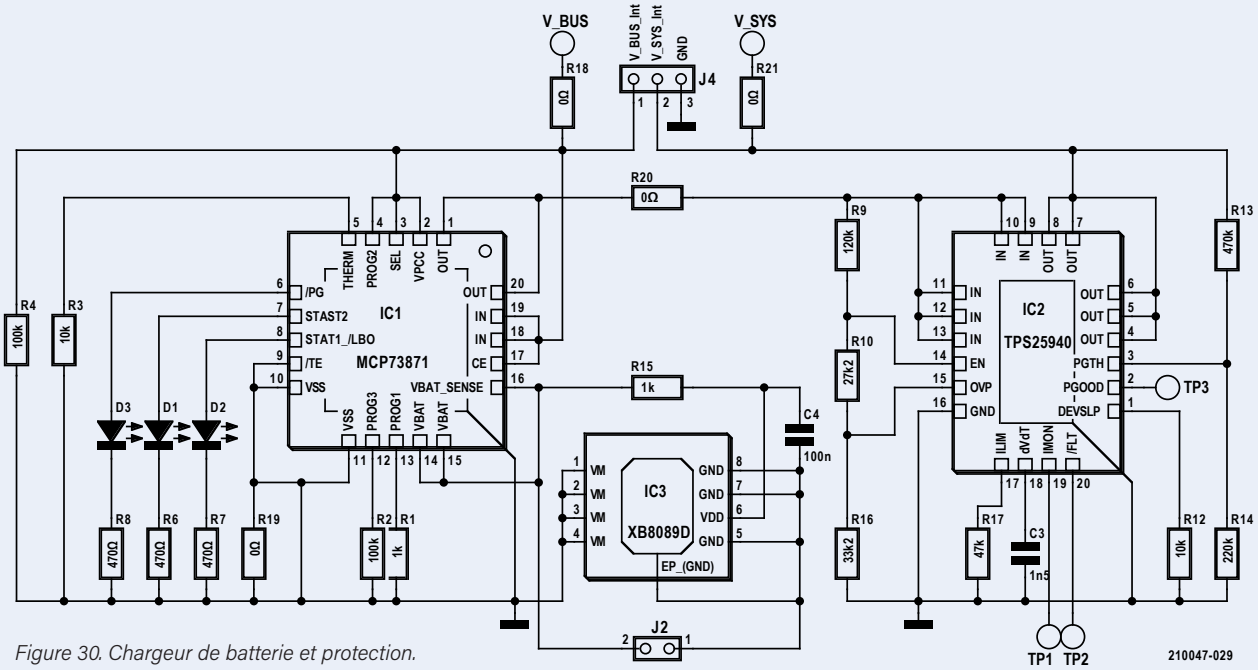


Figure 30. Chargeur de batterie et protection.

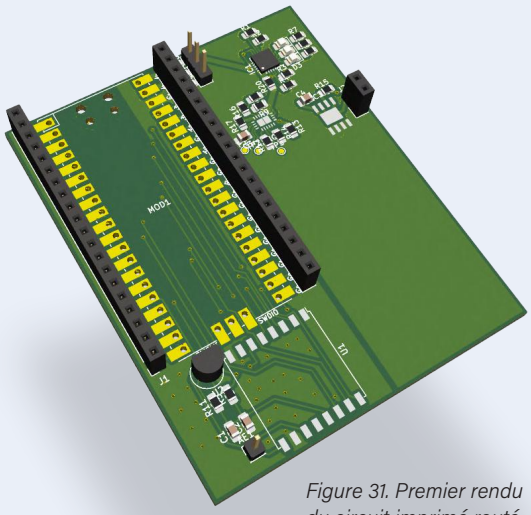


Figure 31. Premier rendu du circuit imprimé routé.

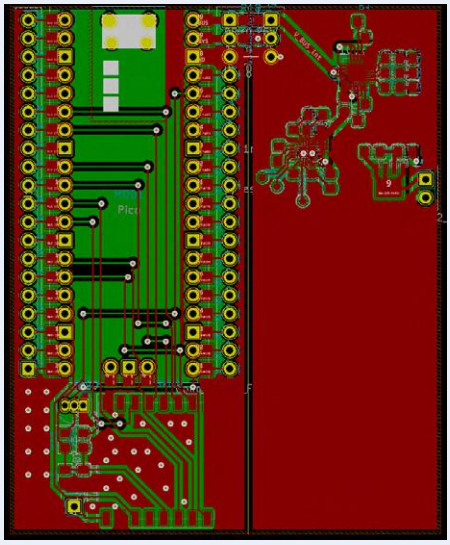


Figure 32. Aperçu de la disposition réalisée.

décharge profonde, la surintensité et l'inversion de polarité. Le dernier élément est un *eFuse* (fusible électronique) TPS25940 de Texas Instruments. Celui-ci agit comme une diode idéale, empêchant le retour du courant vers le chargeur de batterie au lithium. Il servira d'élément de protection contre les surintensités et les décharges profondes. La surtension et la sous-tension sont déterminées par R9, R10 et R16. La fiche technique fournit les formules nécessaires pour calculer les valeurs souhaitées. Nous utilisons 120 kΩ pour R9, 27,2 kΩ pour R10 et 33,2 kΩ pour R16 pour obtenir 5,37 V de seuil de surtension et 2,957 V de seuil de sous-tension. Cela devrait être dans la plage de sécurité, pour la batterie et le convertisseur DC/DC du RPi Pico. Nous utilisons R17 (47 kΩ) pour limiter le courant à 1,89 A, car c'est le maximum que le chargeur au lithium sera capable de fournir. Pourquoi est-ce une mauvaise idée d'utiliser un fusible réarmable (*polyfuse*) ? Lisez l'article d'Elektor [13]. L'utilisation d'une diode introduirait une chute de tension d'au moins 0,3 V à travers elle, ce qui signifie que nous convertissons de l'énergie en chaleur.

Quelque chose que l'on ne voit pas souvent dans les circuits imprimés sont R20, R21 et R18. Ce sont des résistances de 0 Ω qui vous permettront de déconnecter des parties du chargeur de batterie. Si certaines parties doivent être examinées de plus près ou ne fonctionnent pas du tout, elles peuvent être séparées et contournées sans couper les pistes sur le circuit imprimé. Il suffit de retirer quelques composants CMS.

Placement rapide sur un circuit imprimé

Le schéma étant terminé, tous les composants ont été placés sur un circuit imprimé et routés. Ce n'est pas très joli pour l'instant, mais pour un premier essai, cela fera l'affaire. La **figure 31** montre le premier rendu. La **figure 32** donne un aperçu du routage. Vous pouvez également voir que le circuit imprimé est conçu pour que la partie chargeur puisse être complètement retirée ou utilisée à d'autres fins.



Si elle ne fonctionne pas, vous pouvez également la retirer du circuit imprimé. C'est comme un deux-en-un.

Que faire ensuite ?

Les étapes suivantes dépendront des réactions de nos lecteurs, c'est-à-dire vous. Si vous aimez ce projet et souhaitez que nous le poursuivions, que nous y ajoutions des composants ou bien si vous avez des suggestions pour l'améliorer, n'hésitez pas à laisser un commentaire ou à nous envoyer un message. De même, si vous avez des propositions pour différentes parties, nous sommes intéressés. Vous pouvez également nous laisser un message sur d'autres projets potentiellement intéressants.

Pour finir

Utiliser MicroPython sur le RPi Pico et travailler sur un projet LoRaWAN est assez simple et peut être amusant même pour un débutant. Mais la stabilité du logiciel peut être un problème. Pour des installations plus fiables, il est préférable de programmer en C/C++. Si les scripts MicroPython fonctionnent, c'est un moyen rapide et pratique de se lancer. Ce que vous mesurez et transmettez dépend de vous, et le code utilisé est suffisamment simple pour être testé par des enfants. Pourquoi transmettre uniquement des mesures de température ? Vous pouvez faire un système d'alarme LoRaWAN avec quelques détecteurs PIR. Ou vous pouvez surveiller si vos plantes ont besoin d'eau. Laissez libre cours à votre imagination.
Bien que ce soit le premier projet avec un RPi Pico, ce ne sera pas le dernier. Nous en préparons d'autres, mais notez qu'ils ne sont peut-être pas terminés. Néanmoins, nous espérons qu'ils vous inspireront pour vos propres projets ou qu'ils vous donneront des informations qui pourront vous être utiles à l'avenir.

(210047-04)

PRODUITS

- **Carte à microcontrôleur Raspberry Pi Pico**
  www.elektor.fr/raspberry-pi-pico-microcontroller-board
- **Plaque d'essai (830 points)**
  www.elektor.fr/breadboard-830-tie-points
- **Mini plaque d'essai et fils de liaison**
  www.elektor.fr/mini-breadboards-jumper-wires
- **Module émetteur-récepteur LoRa ultra-long (868 MHz) RFM95 de SeeedStudio**
  www.elektor.fr/rfm95-lora
- **Livre en anglais « KiCad Like a Pro » (2e édition)**
  www.elektor.fr/kicad-like-a-pro
- **Passerelle intérieure LoRaWAN LPS8 de Dragino**
  www.elektor.fr/dragino-lps8-indoor-lorawan-gateway
- **Concentrateur GPS LoRaWAN pour Raspberry Pi (868 MHz) PG1301 de Dragino**
  www.elektor.fr/dragino-pg1301
- **Livre électronique en anglais « Programming with Node-RED »**
  www.elektor.fr/programming-with-node-red-e-book

Contributeurs
Conception et texte : Mathias Claußen
Rédaction : Jens Nickel
Mise en page : Harmen Heida
Traduction : Denis Lafourcade

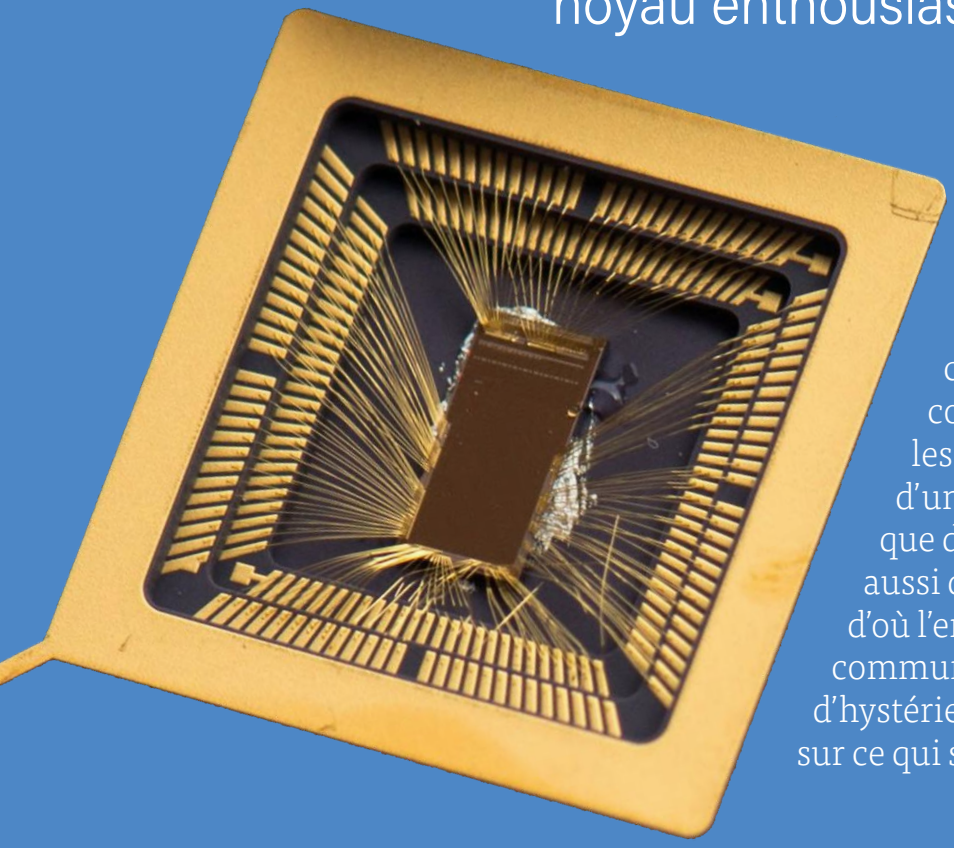
Des questions, des commentaires ?
Envoyez un courriel à l'auteur (mathias.claussen@elektor.com) ou contactez Elektor (redaction@elektor.fr).

LIENS

[1] **« LoRaWAN : décollage facile », Elektor 03-04/2020** : www.elektormagazine.fr/191065-04
[2] **Fichiers du projet sur GitHub** : https://github.com/ElektorLabs/210047-LoRa-with-the-Raspberry-Pi-Pico
[3] **Passerelle intérieure LoRaWAN LPS8 de Dragino** : www.elektor.fr/dragino-lps8-indoor-lorawan-gateway
[4] **Concentrateur GPS LoRaWAN pour Raspberry Pi (868 MHz) PG1301 de Dragino** : www.elektor.fr/dragino-pg1301
[5] **Livre en anglais « Get Started with MicroPython on Raspberry Pi Pico »** : www.elektor.fr/get-started-with-micropython-on-raspberry-pi-pico
[6] **Document PDF en anglais, gratuit, « Get Started with MicroPython on Raspberry Pi Pico »** : https://hackspace.raspberrypi.org/books/micropython-pico/pdf/download
[7] **Dépôt GitHub de uLoRa** : https://github.com/fantasticdonkey/uLoRa
[8] **MicroPython sur Raspberry Pi Pico** : www.raspberrypi.org/documentation/pico/getting-started/
[9] **EDI Thonny** : https://thonny.org/
[10] **Pile The Things Network v3** : https://eu1.cloud.thethings.network/console/
[11] **Instructions d'installation de Node-Red** : https://nodered.org/docs/getting-started/raspberrypi
[12] **GitHub d'Elektor** : https://github.com/ElektorLabs
[13] « **superchargeur LiPo DIY », Elektor 05-06/2021** : www.elektormagazine.fr/191188-B-03
[14] **« superchargeur LiPo en kit », Elektor 01-02/2021** : www.elektormagazine.fr/191188-04
[15] **Fichiers Gerber de la carte du RFM95** : https://github.com/ElektorLabs/191069-RFM95-BOB/

RISC-V : quesaco ?

Pourquoi une nouvelle architecture de noyau enthousiasme-t-elle l'industrie ?



Stuart Cording (Elektor)

Le monde électronique semble être devenu fou de *RISC-V*. Pourquoi ? Qu'est-ce ? Et comment pouvez-vous contribuer ? Si vous parcourez les brèves, vous savez qu'il s'agit d'une architecture de processeur et que des puces l'utilisent. Vous savez aussi que RISC-V est *libre et ouvert*, d'où l'engouement suscité et l'énorme communauté de fans. Au-delà du parfum d'hystérie de ces réactions, penchons-nous sur ce qui se cache derrière ce sigle.

Il faut savoir que RISC-V définit une architecture de jeu d'instructions, ou *ISA* [1], et non un processeur. Cela signifie que les concepteurs du RISC-V décrivent comment faire pour concevoir le fonctionnement d'un processeur qui serait basé sur leur ISA. Par *concevoir*, nous voulons vraiment dire création du processeur avec tous ses registres, accumulateurs, opérations mathématiques, bus mémoire et tout le reste.

L'ISA (*Instruction Set Architecture*) documente par ex. les opérations prises en charge, les capacités d'adressage de la mémoire, le fonctionnement de la pile et le déroulement des interruptions, etc. Pour les opérations prises en charge, elle définit combien de bits sont utilisés pour coder l'instruction et quels bits sont utilisés pour coder la source de tout opérande nécessaire.

La raison de cet engouement est que l'ISA est libre et ouvert. Ouvert signifie que quiconque peut aider à son développement, et libre implique une utilisation libre de tout droit. Toutefois, le design ouvert et libre des cartes *Arduino* ne signifie pas que celles-ci sont gratuites et il en ira de même pour la réalisation du processeur RISC-V de vos rêves.

Qui RISC-V concurrence-t-il ?

Chaque processeur possède une ISA qui appartient à son concepteur et peut faire l'objet d'une licence. *Microchip* produit des appareils basés sur des processeurs *PIC* à 8 et 16 bits et, quelque part, il existe une ISA pour les décrire. Ce sont des noyaux appartenant à Microchip et qui sont vendus avec ses microcontrôleurs. Si vous voulez développer votre propre microcontrôleur, vous opterez sans doute pour *Arm* et *MIPS*. Ces noyaux sont de la propriété intellectuelle (*PI*) qui peut être acquise sous licence. Les entreprises qui les ont conçus ont œuvré pour convertir l'ISA en un processeur matériel, développé des outils de prise en charge, créé les infrastructures connexes et vous font payer pour utiliser le tout. Un problème se pose si ces options ne répondent pas *tout à fait* à vos attentes.

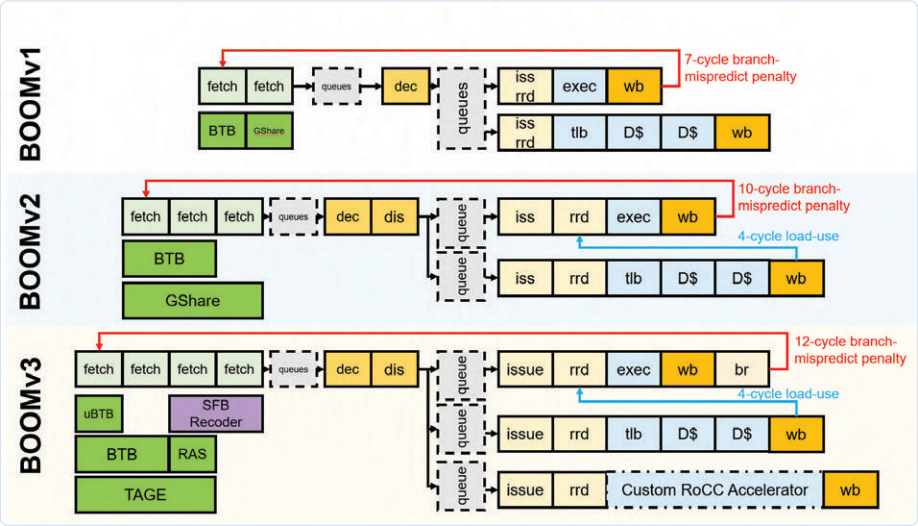


Figure 1. Processus de développement utilisé par le projet BOOM [27] pour implémenter le pipeline RISC-V. (Source : Regents of the University of California)

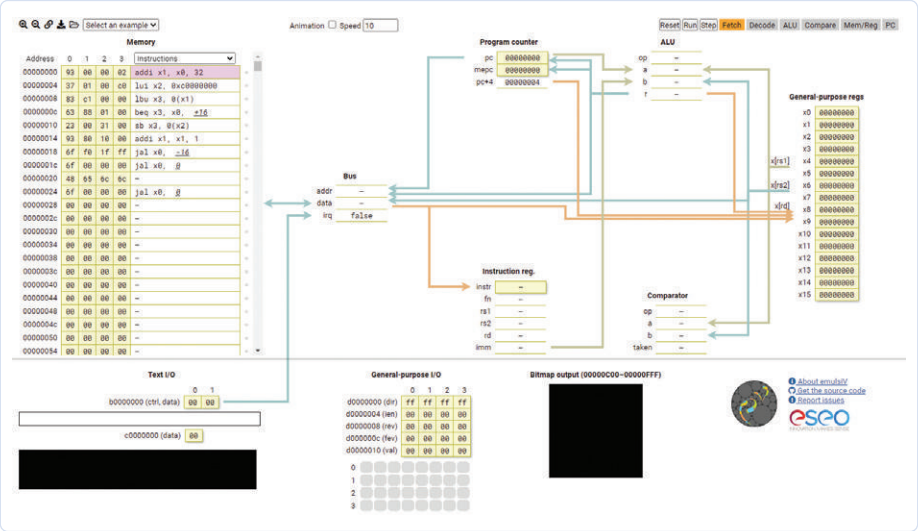


Figure 2. Le simulateur emulsiV permet à tous d'essayer le RISC-V dans un navigateur web.

Par ex., votre application doit exécuter très rapidement du chiffrement, mais en consommant le moins d'énergie possible ; un possible processeur sous licence peut exécuter votre tâche en cent instructions. Pour réduire l'énergie nécessaire, il faut trouver un fondeur de puces (*fab*) spécialisé dans la faible consommation d'énergie. Par rapport à un processus standard de fabrication, les coûts peuvent alors s'envoler et votre super-produit sera trop cher pour votre marché cible. Certains de vos ingénieurs sont peut-être capables d'optimiser le temps d'exécution du code en créant de nouvelles instructions pour le processeur, mais comme l'ISA ne vous appartient pas, vous ne pouvez pas la modifier. Vous avez donc un problème de performance du processeur qu'il faut résoudre par une approche de fabrication. Nous y reviendrons plus tard…

Que fait RISC-V tel quel ?

En deux mots : « peu, mais assez ». Fondamentalement, il faut d'abord choisir l'architecture précise que vous souhaitez. Des ISA à 32 et 64 bits sont définies, et une ISA à 128 bits est également en cours de développement. Ces définitions de base sont baptisées RV32I et RV64I. Avec la RV32I, vous aurez 49 instructions [2] à votre disposition. Le « I » signifie ici *Integer* (entier). Ce sont les instructions arithmétiques et logiques de base sur les entiers (ADD, SUB, AND, OR, XOR), les décalages, comparaisons, sauts et liaisons, ainsi que certaines instructions système [3]. Si vous recherchez un code compact, l'option 'C' peut vous intéresser. Les instructions sont codées sur 16 bits, façon mode *Thumb* d'Arm. On peut y ajouter des instructions multiplication et division (M), atomiques (A) et à virgule flottante (F, D et Q).

Ensuite, il faut concevoir le cœur du processeur d'après les caractéristiques des options choisies dans un *langage de description du matériel* [4] (*Hardware Description Language = HDL*), tel que VHDL ou Verilog. Comme cette étape est difficile, c'est là que la communauté intervient. Concevoir des processeurs exige de multiples compétences, c'est pourquoi nombre de spécialistes et d'entreprises proposent des designs prêts à l'emploi. Pour emprunter la voie *libre*, ne cherchez pas plus loin que la plateforme *PULP* [5], créée par l'*ETH Zurich* et l'*Università di Bologna*. Si vous souhaitez voir comment ce genre de chose est réalisé, *GitHub* présente l'implémentation CV32E40P RV32IM[F]C [6], et le décodeur d'instructions [7]. Le projet *BOOM* est une autre implémentation : un noyau paramétrable à haute performance pour la recherche en architecture, développé à Berkeley par l'*Université de Californie* (**fig. 1**).

Si vous êtes pressé et avez besoin d'assistance, il faudra payer pour obtenir une licence pour l'implémentation d'un tiers comme *SiFive* [8]. Ils disposent d'une gamme de designs à 32 et 64 bits [9] qui peuvent être personnalisés.

Comment essayer RISC-V ?

Bien que RISC-V existe depuis quelque temps, peu de puces sont disponibles pour le tester. Industriellement parlant, RISC-V est encore assez nouveau. Si les microcontrôleurs vous intéressent, vous savez que la plupart des acteurs du secteur ont adopté Arm, abandonnant leurs propres noyaux. C'est un investissement stratégique à long terme. À part l'économie des redevances versées à Arm, passer à RISC-V aurait peu d'avantages pour les utilisateurs. Il faudrait aussi que le service R&D pense RISC-V, l'intègre à toutes

ses autres *PI* (analogiques, temporisateurs, bus, interfaces, mémoires), mette à jour l'*EDI* de développement, le compilateur, le débogueur, etc.

Ceux qui ont un disque dur *Seagate* ou *Western Digital*, utilisent peut-être déjà RISC-V [10] [11]. Mais au-delà de posséder un produit qui l'utilise, vous voulez sans doute exécuter du code sur ce noyau. Le moyen le plus rapide d'y parvenir est un simulateur tel qu'*emulsiV* d'*ESEO* [12], basé sur leur implémentation du noyau RISC-V *Virgule* (**fig. 2**). En plus du processeur, le simulateur offre des E/S de texte, une sortie bitmap et des E/S à usage général (GPIO). Sept exemples couvrent les bases, de l'ajout/sortie de texte ASCII au contrôle des GPIO. L'intéressante option *animation* (case à cocher centrale du haut) montre d'où viennent et où vont les données au fur et à mesure de l'exécution du code. Vous pouvez essayer le code du **listage 1** (le copier dans un éditeur de texte, enregistrer le fichier sous `program.hex` et le télécharger dans le simulateur).

Pour découvrir le RISC-V au format Arduino, le *HiFive1 Rev B* est disponible sur *CrowdSupply* [13]. Il utilise le microcontrôleur SiFive *FE310-G002* [14]. C'est un dispositif élémentaire : périphériques numériques seulement (I2C, UART, SPI, PWM, GPIO), un peu de SRAM, appuyée par une flash QSPI hors puce pour le stockage non-volatile. La carte comprend un module Wi-Fi et Bluetooth ainsi qu'un *J-Link* de *Segger* pour le débogage USB.

De l'autre côté du spectre des performances, il y a le *SoC* PolarFire [15] de Microchip avec quatre cœurs RISC-V à 64 bits secondés par un *FPGA*. C'est une plateforme configurable à loisir qui peut exécuter *Linux* tout en prenant en charge des applications en temps réel.

Listage 1. Code hexa brut pour simulateur emulsiV à sauvegarder et télécharger comme program.hex.

```
:10000000930000023701 00c083c100006388010033
:10001000230031009380 10006ff01fff6f0000007d
:1000200048656c6c6f20456c656b746f72210000c5
:00000001FF
```

Listage 2. Code C pour implémenter le chiffrement ChaCha.

```
unsigned int processLine(unsigned int res, unsigned int word) {
  res = qr1_c(res, word);
  res = qr2_c(res, word);
  res = qr3_c(res, word);
  res = qr4_c(res, word);
  res = qr1_c(res, word);
  res = qr2_c(res, word);
  res = qr3_c(res, word);
  res = qr4_c(res, word);
  return res;
}
```

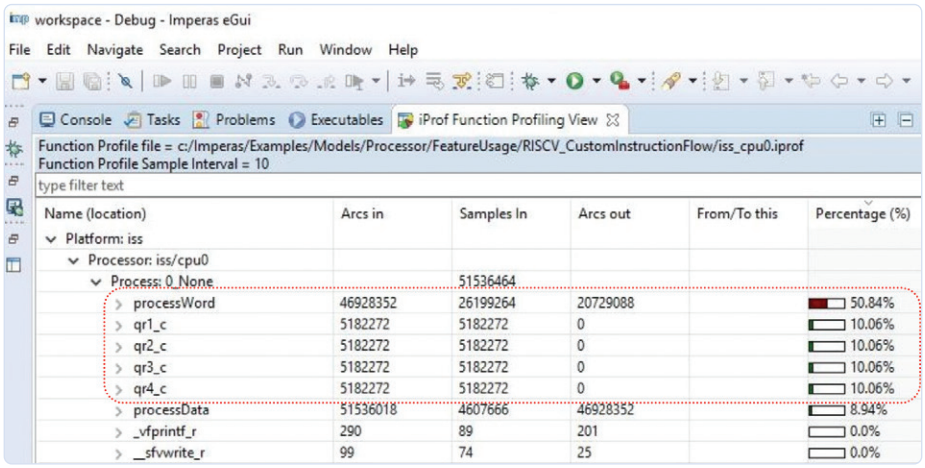


Figure 3. Le chiffrement ChaCha occupe 55 % du temps processeur avec le code compilé en C standard. (Source : Imperas Software Limited)

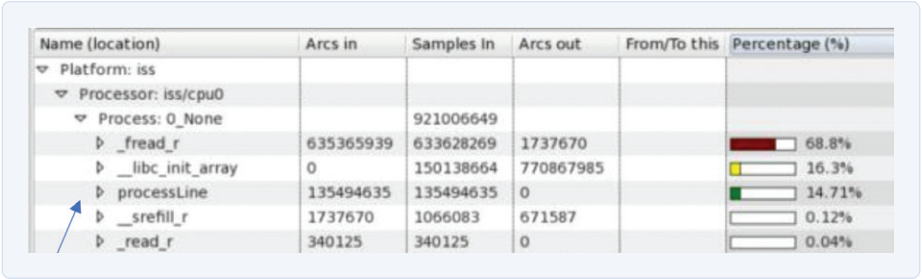
Comment personnaliser mon RISC-V ?

Nous mentionnions plus haut que l'avantage de RISC-V était de pouvoir adapter le jeu d'instructions aux besoins de chaque application. C.-à-d. que si l'implémentation de processeur choisie répond à 95 % des besoins, on peut y ajouter des fonctions astucieuses pour couvrir les 5 % manquants. Supposons par ex. que l'application fasse un usage intensif du chiffrement de flux *ChaCha* [16], tel que décrit dans une note d'application d'*Imperas* [18], un autre acteur RISC-V qui fournit des outils de vérification, d'analyse et de profilage.

Cependant, vous observez que votre implémentation ChaCha sur un noyau RISC-V nécessite un temps de traitement élevé. Alors vous voudrez réduire le temps d'exécution, mais aussi profiter de la baisse de consommation résultant de cette amélioration, par ex. en entrant dans un mode de veille à faible énergie.

Le code (**listage 2**) fait un usage intensif des instructions *XOR* et *rotate* dans une étape dite des *quarter rounds* pour laquelle quatre fonctions C `qrX_c()` ont été écrites. La fonction `processLine()` les appelle pour effectuer le chiffrement. L'analyse des temps d'exécution montre que cette tâche occupe 55 % du temps du processeur, dont environ 32 % sont répartis sur les fonctions *quarter round* (**fig. 3**).

Avec RISC-V, il suffit d'implémenter quatre instructions *quarter round* spécifiques qui s'exécutent en un seul cycle au lieu de laisser le code produit par le compilateur C s'en charger. Dans l'ISA, il existe en effet une section réservée aux instructions personnalisées. Au départ, les instructions peuvent être ajoutées à une conception



Name (location)	Arcs in	Samples in	Arcs out	From/To this	Percentage (%)
Platform: iss					
Processor: iss/cpu0					
Process: 0_None		921006649			
_fread_r	635365939	633628269	1737670		68.8%
__libc_init_array	0	150138664	770867985		16.3%
processLine	135494635	135494635	0		14.71%
__srefill_r	1737670	1066083	671587		0.12%
_read_r	340125	340125	0		0.04%

Figure 4. Avec les dernières instructions spécialisées développées, la charge processeur du chiffrement ChaCha tombe à moins de 15 %. (Source : Imperas Software Limited)

RISC-V et implémentées sous forme codée en C. Cela permet de simuler les nouvelles instructions afin de les tester et vérifier si une amélioration des performances est réalisable. Ici, avec des instructions spécialisées *quarter round* disponibles sur le cœur RISC-V personnalisé, la fonction `processLine()` mobilise désormais moins de 15 % du potentiel du processeur (**fig. 4**) . Si c'est considéré comme correct, l'équipe de développement peut alors réaliser l'implémentation matérielle des instructions en Verilog.

Hélas, l'utilisation des instructions nouvelles ne se borne pas à recompiler le code C (**listage 3**). Modifier un compilateur RISC-V pour utiliser de nouvelles instructions représente un effort énorme. À la place, les instructions codées en hexa sont appelées par l'assembleur en ligne comme pour un code optimisé à la main.

Comment contribuer à RISC-V ?

Si contribuer au développement permanent de RISC-V vous tente, c'est le moment. RISC-V International [19] est l'organisation chargée de développer et promouvoir l'idée RISC-V (**fig. 5**). Chacun peut adhérer comme membre de la communauté [20] ou y faire carrière, de nombreuses entreprises et universités [21] y participent activement.

Si vous pensez voir arriver sur le marché de nombreux microcontrôleurs RISC-V, vous risquez d'être déçu. *GigaDevice* en propose quelques-uns [22] et un fournisseur russe un autre pour le marché des compteurs intelligents [23]. Arm est trop ancré auprès des grands acteurs, et les jeunes pousses auront du mal à lutter sur ce marché saturé, même avec l'avantage financier d'un processeur sans royalties à débourser.

RISC-V sera plus probablement mis à profit dans des niches où sa capacité à personnaliser le noyau apporte de gros avantages, tels qu'une ultra-faible consommation [24]. Compte tenu des obstacles à l'octroi de licences technologiques à la Chine, RISC-V est une alternative tentante d'acquisition de PI américaine. *Alibaba* a annoncé un RISC-V à 64 bits à 16 cœurs, 2 GHz, en gravure 12 nm [25] et indique que le noyau est destiné à une infrastructure de serveur. Enfin, la *European Processor Initiative* [26] s'est penchée sur les architectures hétérogènes dans lesquelles Arm et RISC-V (ou d'autres



Figure 5. Logo officiel de RISC-V International qui promeut et soutient le développement de l'ISA.

Listage 3. Utilisation des instructions du RISC-V.

Pour utiliser les instructions RISC-V, il faut un assembleur en ligne. Les instructions sont codées en hexa car l'assembleur ne connaît pas le nom de ces nouvelles instructions.

```
unsigned int processLine(unsigned int res, unsigned int word) {
  asm __volatile__("mv x10, %0" :: "r"(res));
  asm __volatile__("mv x11, %0" :: "r"(word));
  asm __volatile__(".word 0x00B5050B\n" ::: "x10"); // equivalent to qr1_c()
  asm __volatile__(".word 0x00B5150B\n" ::: "x10"); // equivalent to qr2_c()
  asm __volatile__(".word 0x00B5250B\n" ::: "x10"); // equivalent to qr3_c()
  asm __volatile__(".word 0x00B5350B\n" ::: "x10"); // equivalent to qr4_c()
  asm __volatile__(".word 0x00B5050B\n" ::: "x10"); // equivalent to qr1_c()
  asm __volatile__(".word 0x00B5150B\n" ::: "x10"); // equivalent to qr2_c()
  asm __volatile__(".word 0x00B5250B\n" ::: "x10"); // equivalent to qr3_c()
  asm __volatile__(".word 0x00B5350B\n" ::: "x10"); // equivalent to qr4_c()
  return res;
}
```

noyaux) pourraient cohabiter. Le but est d'obtenir un effet de synergie en exploitant le meilleur processeur pour chaque tâche dans les designs multicœurs.
RISC-V n'est pas la première tentative de PI libre et ouverte pour les processeurs, mais c'est la plus aboutie à ce jour. Considérant son héritage, ses qualités (souplesse, approche ouverte), l'intérêt qu'il suscite dans les universités et le soutien important de l'industrie, RISC-V accompagnera la carrière d'ingénieurs durant une, voire deux générations.

(210223-04)

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**
Traduction : **Yves Georges**
Mise en page : **Giel Dols**



PRODUITS

- **Carte de développement RED-V Thing Plus de SparkFun – SoC SiFive RISC-V FE310**
  www.elektor.fr/sparkfun-red-v-thing-plus-sifive-risc-v-fe310-soc
- **Carte de développement RISC-V TTGO T-Display-GD32 de LilyGo**
  www.elektor.fr/lilygo-ttgo-t-display-gd32-risc-v-development-board

LIENS

[1] **Architecture des jeux d'instructions et microarchitecture, GeeksforGeeks, octobre 2019** : http://bit.ly/3bBKAy2
[2] **RISC-V, Wikipédia** : https://fr.wikipedia.org/wiki/RISC-V
[3] **J. Zhu, « RISC-V Reference »** : http://bit.ly/30IlCXj
[4] **Langage de description du matériel, Wikipédia** : https://fr.wikipedia.org/wiki/Langage_de_description_de_mat%C3%A9riel#:
[5] **Site de la plate-forme PULP** : https://pulp-platform.org/
[6] **Dépôt GitHub – CV32E40P RISC-V core** : https://github.com/openhwgroup/cv32e40p
[7] **Implémentation SystemVerilog du décodeur d'instructions CV32E40P** : https://github.com/openhwgroup/cv32e40p/blob/master/rtl/cv32e40p_id_stage.sv
[8] **Site de SiFive** : www.sifive.com
[9] **Core Designer de SiFive** : http://bit.ly/3rKnkU7
[10] **Page RISC-V de Seagate** : http://bit.ly/3etS0oU
[11] **Page RISC-V de Western Digital** : http://bit.ly/3eyldyP
[12] **Simulateur emulsiV pour processeur RISC-V Virgule** : http://bit.ly/30AzqmG
[13] **Page CrowdSupply pour la carte Arduino HiFive1 Rev B** : http://bit.ly/3eww7Fj
[14] **Fiche technique du microcontrôleur SiFive FE310-G002** : https://bit.ly/3bFANak
[15] **Page produit PolarFire SoC** : http://bit.ly/3bFAU5K
[16] **Variante ChaCha [du chiffrement de flux Salsa20], Wikipédia** : http://bit.ly/3rHfZES
[17] **Chiffrement de flux, Wikipédia** : https://fr.wikipedia.org/wiki/Chiffrement_de_flux
[18] **« Imperas RISCV Custom Instruction Flow Application Note », Imperas Software Limited, octobre 2019** : http://bit.ly/3vguDVK
[19] **À propos de RISC-V** : https://riscv.org/about/
[20] **Adhésion à RISC-V** : https://riscv.org/membership/
[21] **Membres RISC-V** : https://riscv.org/members/
[22] **Microcontrôleurs RISC-V GigaDevice GD32** : http://bit.ly/2NbSgO9
[23] **Сергей, microcontrôleur russe K1986BK025 basé sur le noyau processeur RISC-V pour les compteurs électriques intelligents, Habr, décembre 2020** : http://bit.ly/3qGPY7o
[24] **Site de MicroMagic** : www.micromagic.com
[25] **J. Aufranc, « Infos détaillées sur le Cœur 64 bits RISC-V XT910 d'Alibaba », CNX Software, août 2020** : http://bit.ly/3eync6f
[26] **Site Initiative Européenne sur les Processeurs** : www.european-processor-initiative.eu/
[27] **Dépôt GitHub –cCœur BOOM RISC-V** : https://github.com/riscv-boom/riscv-boom

le très attendu numéro double d'été

60 ans d'Elektor

Thomas Scherer

Soixante ans de magazine Elektor ! Depuis 1961, vous suivez l'évolution des techniques, des appareils, cartes et composants dans nos colonnes – notre marque de fabrique c'est l'équilibre entre théorie et pratique. Pendant des décennies, le numéro double de l'été a été très attendu, avec ses « circuits pour les vacances » ; il rassemblait souvent plus de cent mini-projets dans ce seul numéro. Presque tous étaient faciles à construire, avec une électronique plutôt simple, mais toujours très, très utiles… En voici un petit aperçu.

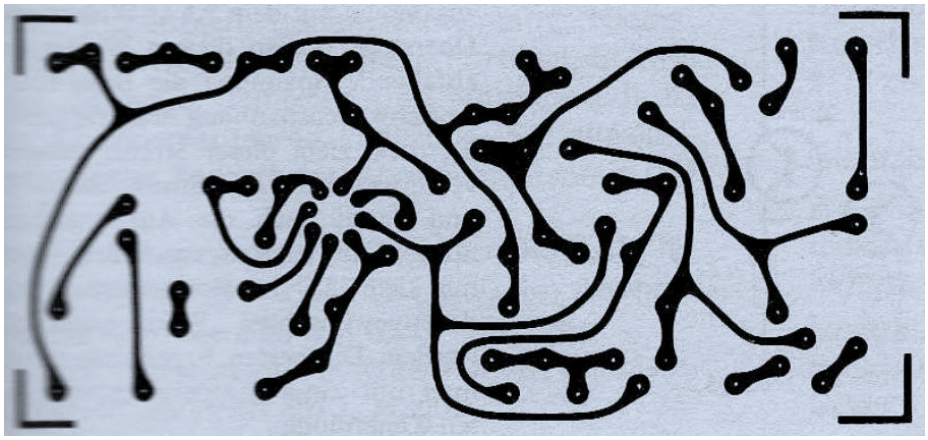


Figure 1. Le circuit imprimé tracé à la main de l'ampli de puissance de 100 W (40411) (paru dans l'édition allemande). www.elektormagazine.de/magazine/elektor-197011/54723

Je suis un lecteur assidu d'Elektor depuis le tout début. Dès les années 1970, j'ai appris à tenir un fer à souder par le bon bout, et je suis resté accro. Je ne me rappelle pas combien de montages j'ai reproduits chez moi, mais j'ai toujours aimé le numéro d'été qui fourmillait d'*idées de mini-circuits*. C'était une véritable mine de trouvailles. De nombreux électroniciens – même les professionnels – ont été inspirés par ces solutions, souvent originales. Devise : Elektor, le journal qui capte l'intérêt de ses lecteurs.

Amplificateurs…

Elektor et Edwin – pour les vétérans de l'électronique audio, ces deux noms sont indissociables. Le 1er numéro en allemand de ce magazine proposait de réaliser un amplificateur. Sa « particularité » était l'absence de courant (dérivant) de repos. C'est un fait : un nouveau concept d'ampli n'est jamais de trop. Le 1er numéro en allemand avec la collection de *Circuits d'été* de 1970 rassemblait huit sortes d'amplis parmi les 104 circuits [1] publiés. Chez Elektor, les circuits imprimés étaient alors dessinés à la main. La **figure 1**, avec « l'amplificateur de puissance de 100 W (40411) », en est l'illustration. En 1970, 2 × 6 W était une puissance d'installation Hi-Fi classique, mais 100 W, c'était un véritable coup de force. Le chiffre entre parenthèses est le type des transistors de puissance NPN de l'étage de sortie (en boîtier TO3, le plus moderne). Comme attendu, le schéma était assez avancé et utilisait un ampli-op 709 (prédécesseur du 741) en entrée. La version anglaise du magazine n'est apparue qu'en décembre 1974. En 1975, le numéro double de juillet/août [2] inclut pour la 1ère fois les *Circuits d'été*. Parmi les 18 circuits où le terme « amplificateur » apparaît, j'ai été particulièrement intéressé par le projet n°16, le « Bat Detector » (**fig. 2**). Ici, le bavardage ultrasonore des chauves-souris est transposé en fréquences audibles par mélange avec le signal d'un oscillateur. Comme c'était le cas à l'époque, le projet utilisait des transistors discrets (la version anglaise du magazine n'incluait pas le circuit). En 1980, j'intégrai l'équipe de rédaction. J'étais le plus jeune employé d'Elektor et je fus émerveillé par la couverture psychédélique du numéro *Circuits d'été* de 1980 [3] (**fig. 3**). Comme les autres, ce numéro faisait la part belle aux amplificateurs. À l'époque, nous étions focalisés sur l'intérêt de particularités du schéma, par ex. transistors configurés en cascode. Mais je fus étonné par une autre idée de circuit de ce numéro. Voilà bien une vedette de l'époque : le chargeur de piles NiCd (il y avait peu d'alternatives en piles rechargeables, en ce temps-là). La particularité du circuit (**fig. 4**) est qu'il n'utilise pas de transformateur d'alimentation. Il est *relié au secteur*, en fait, couplé par condensateur sans transfo. Bob van der Horst, rédacteur en chef à l'époque, s'appesantit sur cette technique sans transfo : la tête de mort sur le schéma indique que ce circuit est potentiellement mortel si vous ne faites pas attention !

Quoi de neuf dans les années 80 ?

L'idée de la domotique ne date pas de l'arrivée d'internet. Dans le numéro double de 1985, parmi les *Circuits d'été*, figure un gradateur sensitif double. Avec ses deux CI et un circuit imprimé rond (**fig. 5**), cette commande d'éclairage trouvait place dans une boîte de dérivation standard. À cette époque, Elektor concevait ses circuits imprimés avec des symboles adhésifs.

Le modélisme figurait en bonne place dans les circuits d'été. En mars 1989, nous publiâmes un générateur sonore qui imitait le bruit produit par un moteur diesel marin. Le circuit de la **fig. 6** montre qu'il suffit d'un ampli-op standard, de deux transistors, d'une LDR et d'une ampoule pour produire le bruit caractéristique du martèlement du moteur diesel marin d'un remorqueur. Ce bruit variait même avec la vitesse.

Voyages mystères magiques

Les circuits qui ne fonctionneront jamais ou dont l'efficacité est nulle sont un thème très attendu. En dépit de preuves scientifiques du contraire, de nombreux utilisateurs défendent l'utilité de pseudo-produits. Des dispositifs antibuée à placer dans les bouilloires et les chauffe-eaux sont encore en vente. Un adoucisseur d'eau électronique fut publié dans le numéro de *Circuits d'été* de 1994 (**fig. 7**).

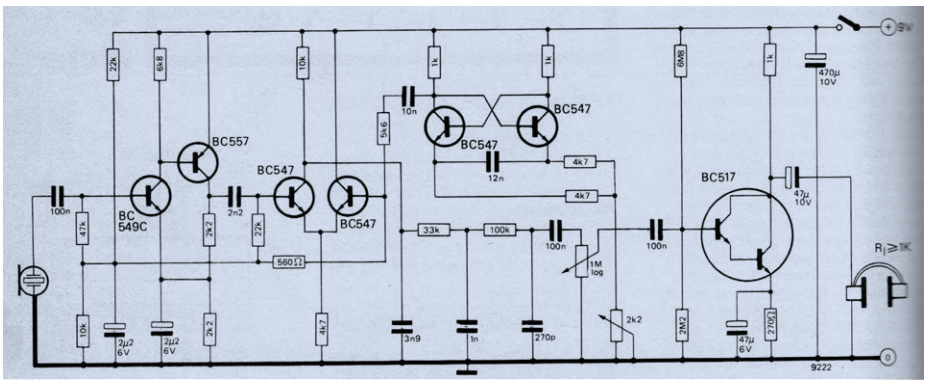


Figure 2. Circuit d'écoute des ultrasons d'écholocalisation des chauves-souris (paru dans l'édition allemande). www.elektormagazine.de/magazine/elektor-197507/55831

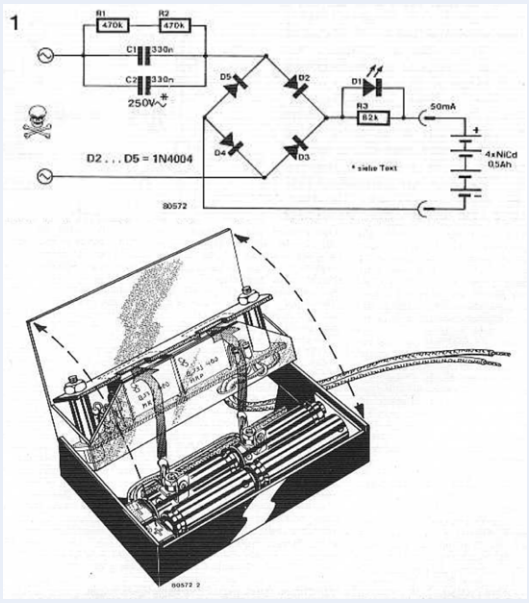


Figure 4. Chargeur NiCd sans transformateur. Le boîtier entièrement isolé est illustré ci-dessus.
www.elektormagazine.com/magazine/elektor-197999/44465

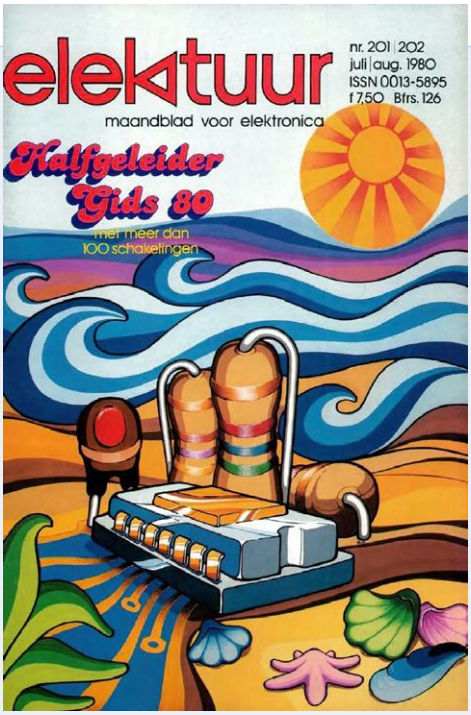


Figure 3. Circuits d'été 1980… Ça pulse !
www.elektormagazine.com/magazine/elektor-197999

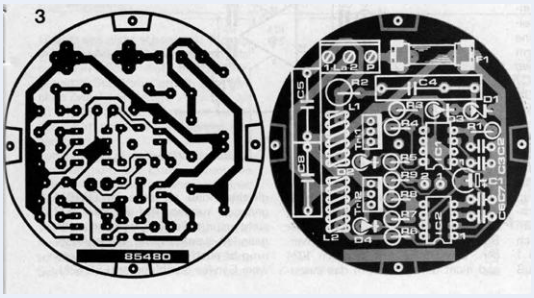


Figure 5. Circuit imprimé de gradateur double.
www.elektormagazine.fr/magazine/elektor-198507/52602

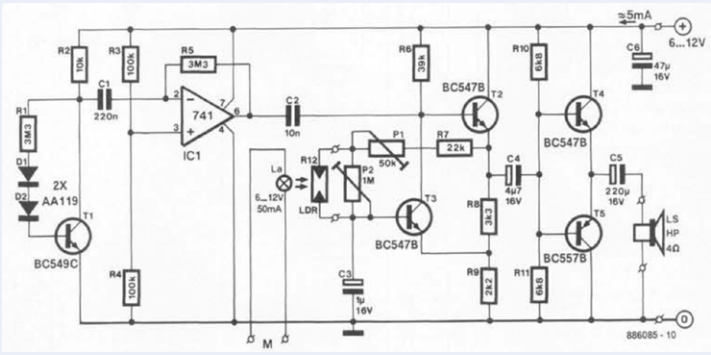


Figure 6. Quelques composants simulent le son d'un diesel marin.
www.elektormagazine.com/magazine/elektor-198903/47528
Ce montage a été publié en français par le « petit frère » d'Elektor, le magazine Elex (n°31, 03/1991)

Cet appareil appliquait un champ alternatif de 400 $V_{c\text{-}à\text{-}c}$, 700 Hz au flux d'eau entrant. L'article affirmait que cela remplaçait les adoucisseurs d'eau utilisant une technique d'extraction des minéraux scientifiquement prouvée (et beaucoup plus coûteuse). Ce n'était pas le 1er détartrant du genre à être présenté dans Elektor. Il doit en rester beaucoup montés sur les conduites d'eau dans le monde entier. On trouve même des modèles similaires dans les quincailleries. Certains se prétendent efficaces malgré le blindage formé par les tuyaux de cuivre mis à la terre !

PC et autres

Le tournant du millénaire : dans la série *Circuits presque sans composant*, on trouve le *Chargeur de batterie alimenté par PC* (**fig. 8**), de Burkhard Kainka, collaborateur de longue date d'Elektor. À l'heure actuelle, tous les électroniciens ont un PC sur leur bureau, et beaucoup s'interrogent : quelle est son utilité pratique ? La plupart étaient équipés d'un port série ou plus, avant que le port USB n'ait raison d'eux. L'idée était de cumuler le courant des signaux de données via une porte OU à trois entrées, constituée de diodes pour recharger en continu trois accus CdNi…

La maison et le jardin n'ont pas été oubliés non plus. Le *Circuit d'été* n°63 de 2006 décrivait un répulsif à limaces et comment il avait été conçu, testé et ajusté pour protéger les fruits mous de l'auteur contre les limaces et les escargots en maraude (**fig. 9**). La réalisation utilise des CI numériques et un ampli de puissance de sortie alimentant une clôture électrique qui repousse ces gastéropodes indésirables.

S'y ajoutent de nombreux circuits et outils RF destinés aux radioamateurs et aux ingénieurs RF. La **figure 10** présente le prototype de la radio DSP d'Elektor publiée en 2010. Ce récepteur mondial, avant-gardiste et doté d'une interface USB, sans aucun réglage (redouté), fut plébiscité.

Juste pour le plaisir…

Chez Elektor nous publions traditionnellement de temps en temps un schéma peu sérieux, sans réelle utilité, qui ne fonctionne pas ou pas comme décrit. Pire encore, ce circuit peut avoir délibérément été conçu pour nuire ou faire une farce à un collègue. L'un

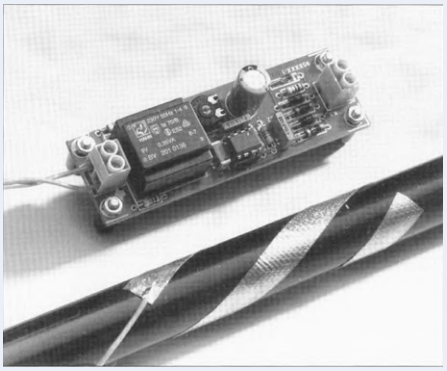


Figure 7. Détartreur électronique.
www.elektormagazine.fr/magazine/elektor-199407/35757

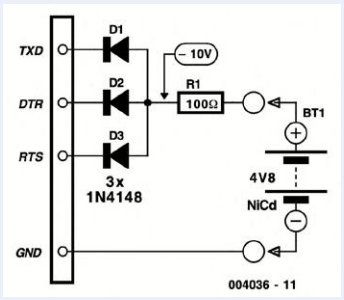


Figure 8. PC chargeur de batterie.
www.elektormagazine.com/magazine/elektor-200007/16844

Le téléchargement de ces articles est gratuit !

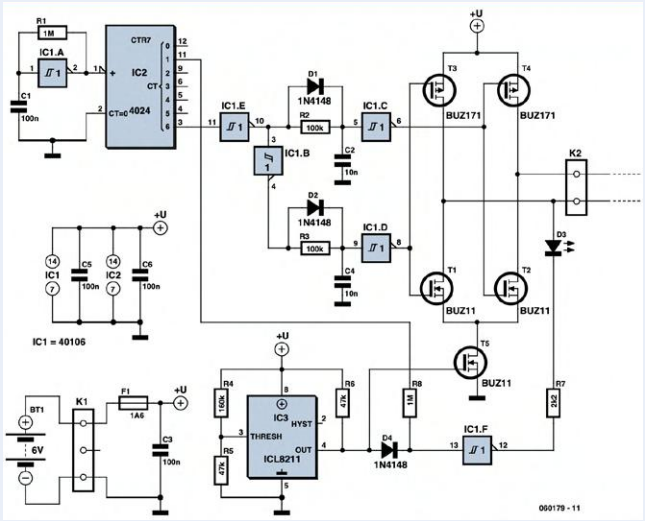


Figure 9. Chasse-limaces.
www.elektormagazine.fr/magazine/elektor-200607/10458

d'eux me vient à l'esprit : c'est le « moustique électronique » [4] des *Circuits d'été* de 1990. Si quelqu'un l'avait caché dans ma chambre, le côté amusant m'aurait sûrement échappé… Cette idée était méchante !

(210256-04)

Addendum

Le dernier numéro double d'été avec plus de 100 mini-circuits est paru en 2012. Malheureusement, depuis la création d'Elektor soit six décennies, l'électronique est devenue plus complexe et plus industrielle. Un circuit réalisé avec une poignée de composants ne peut plus justifier la publication d'un article. Au fil des ans, Elektor a décrit des applications comme la charge, les alimentations, la gradation, les LED, ainsi que la transmission de données. Désormais, les composants peuvent souvent être remplacés par un µcontrôleur polyvalent dont les actions découlent d'un logiciel. Inévitablement, un espace plus précieux est occupé par l'installation d'un EDI, les trucs et astuces du langage de programmation, etc. Cela ne signifie pas que notre amour des petits circuits s'est éteint ; au contraire, les débutants peuvent en particulier beaucoup apprendre ici ! Pour vous mettre en appétit, jetez un coup d'œil à notre nouvelle série de mini-circuits très récente sur notre site : www.elektormagazine.com/tags/circuit-shorts. Sur elektormagazine.fr, cherchez avec « petits circuits ».



Figure 10. La radio DSP d'Elektor.
www.elektormagazine.fr/magazine/elektor-201007/11614

Contributeurs
Auteur : **Thomas Scherer**
Rédaction : **Jens Nickel, Stuart Cording**
Traduction : **Yves Georges**
Mise en page : **Harmen Heida**

LIENS

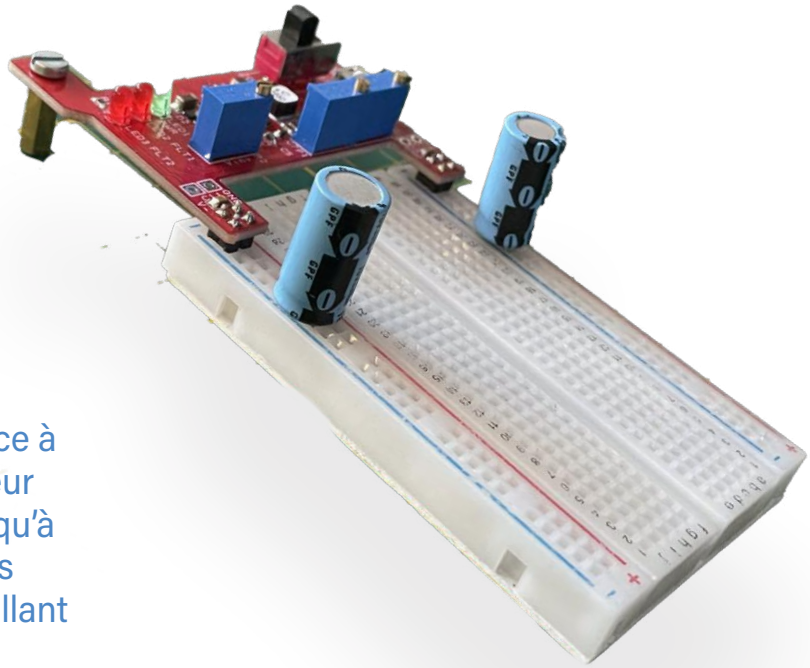
[1] Elektor, numéro allemand, 04/1970 : www.elektormagazine.de/magazine/elektor-197011
[2] Elektor, numéro anglais, 07-08/1975 : www.elektormagazine.com/magazine/elektor-197507
[3] Elektor, numéro anglais, 07-08/1980 : www.elektormagazine.com/magazine/elektor-197999
[4] Moustique électronique : www.elektormagazine.fr/magazine/elektor-199007/34659

module d'alimentation polyvalent pour plaque d'expérimentation

Tensions positives et négatives grâce à un chargeur USB de 5 V

Fons Janssen (Pays-Bas)

Sans alimentation de labo sous la main, il est toujours utile de posséder un module d'alimentation pour plaque d'expérimentation. Sur la plupart des modules proposés sur le marché, la tension de sortie est toujours positive et sensiblement inférieure à la tension d'entrée. Notre projet est différent. Grâce à deux circuits intégrés Maxim, le chargeur micro-USB de 5 V à l'entrée délivre jusqu'à quatre tensions de sortie, trois positives allant de 0,6 V à 20 V, et une négative allant de –1,8 V à –11 V.



INFOS SUR LE PROJET

Mots clés
Plaque d'expérimentation, module d'alimentation, prototype

Niveau
Expert

Temps
1 h

Outils
Station de soudage à air chaud ou four de refusion, pâte à souder

Coût
20 €

La plaque d'expérimentation est un accessoire utile pour le test rapide d'un prototype. Elle est utilisée par tous : électroniciens expérimentés, mais aussi débutants ou étudiants qui prennent encore leurs marques dans le monde de l'électronique. Pour construire un circuit, pas besoin de soudure, et il est très facile de changer les fils ou les composants. Et puisque tout circuit a besoin d'électricité, nous avons conçu un module d'alimentation facile à utiliser sur une plaque d'expérimentation.

Une alimentation de labo est idéale en cas d'expérimentation avec des composants électroniques ou des prototypes, mais elle n'est pas toujours disponible. Par contre, on peut toujours trouver un chargeur micro-USB de 5 V. Nous en avons tous autour de nous, parfois plus qu'il n'en faudrait, pour alimenter ou charger nos appareils, téléphones portables et autres gadgets. Comme la plupart de ces adaptateurs sont protégés contre les courts-circuits, notre projet est parfait et abordable pour les circuits montés sur plaque d'expérimentation. Sa puissance est limitée, mais comme *chacun* sait, les plaques d'expérimentation ne sont pas adaptées aux circuits de forte puissance.

On trouve de nombreux circuits imprimés pour l'alimentation de plaques d'expérimentation, mais sur la plupart d'entre eux, contrairement à notre montage, la tension de sortie est bien inférieure à la tension d'entrée. Les sorties négatives ou symétriques sont encore plus rares.

Le circuit présenté ici constitue une nouveauté. Il permet de créer facilement une source d'alimentation négative ou symétrique (par ex. +9 V/–9 V) pour un circuit composé d'amplificateurs opérationnels, en se servant par exemple d'un adaptateur standard de 5 V. La tension de sortie positive peut aller jusqu'à 20 V ! Mais on peut aussi créer une petite tension de 3,3 V pour un microcontrôleur. Toutes les combinaisons sont possibles. Doté de deux circuits intégrés, le module d'alimentation pour plaque d'expérimentation propose quatre tensions de sortie : une négative et trois positives.

Le schéma

La **figure 1** présente le schéma du module d'alimentation pour plaque d'expérimentation. Le cœur du circuit se compose d'un Maxim MAX8614B [1]. Il est équipé d'un convertisseur élévateur pouvant délivrer une tension (V_{BST}) supérieure à la tension d'entrée, et d'un convertisseur abaisseur-élévateur inverseur pour obtenir une tension négative (V_{INV}). Le potentiomètre P1, dans la boucle de réaction du convertisseur élévateur positif, permet un réglage de la tension de sortie compris entre :

$$\left(\frac{120k}{33k}+1\right)\times 1,01V \approx +5V \quad et \quad \left(\frac{620k}{33k}+1\right)\times 1,01V \approx +20V$$

Le convertisseur abaisseur-élévateur inverseur possède aussi un potentiomètre (P2) dans son circuit de réaction, ce qui permet de régler sa tension de sortie entre :

$$-\frac{39k}{27k}\times 1,25V \approx\sim -1,8V \quad et \quad -\frac{239k}{27k}\times 1,25V \approx\sim -11V$$

La puissance de sortie maximale délivrée par les convertisseurs est d'environ 2 W pour le convertisseur élévateur et 1 W pour le convertisseur abaisseur-élévateur. Ainsi, plus la tension de sortie est élevée, plus la puissance de sortie maximale est faible. Par ailleurs, l'ondulation résiduelle augmente en fonction de la charge. C'est pourquoi il vaut mieux placer un condensateur électrolytique, de quelques centaines de microfarads, sur les rails d'alimentation de la plaque d'expérimentation. Pensez à vérifier la polarité et la tension nominale du condensateur !

Le second circuit intégré du schéma est le MAX38903 (de la marque Maxim également, voir [2]). Il s'agit d'un régulateur de tension linéaire (LDO) qui permet, grâce à P3, un réglage de sa tension de sortie entre :

$$\left(\frac{0k}{68k}+1\right)\times 0,6V \approx +0,6V \quad et \quad \left(\frac{500k}{68k}+1\right)\times 0,6V \approx +5V$$

Avec un LDO, la tension de sortie ne peut être supérieure à la tension d'entrée. La tension de sortie maximale dépend de la tension du chargeur USB (ici, 5 V), des pertes dans les fils/raccords et de la tension nominale du MAX38903. En pratique, cette tension maximale est d'environ 4,5 V. Le LDO peut supporter jusqu'à 1 A. La plupart des adaptateurs peuvent fournir un tel courant, mais la dissipation de puissance dans le circuit régulateur sera d'autant plus forte que la tension de sortie demandée sera plus faible :

$$P_{diss} = (5V - V_{out}) \times I_{out}$$

Cette dissipation ne devant pas excéder 2 W, un courant de 1 A ne peut être délivré qu'à une tension de sortie supérieure à 3 V. Heureusement, le MAX38903 possède plusieurs circuits de protection, pour éviter au module d'alimentation de fumer sous l'excès de puissance. Mais il sera de votre responsabilité d'assurer la protection du circuit monté sur la plaque d'expérimentation.

Montage

Les fichiers Gerber pour fabriquer le circuit imprimé sont présentés en **figure 2**. Les fichiers sources DesignSpark sont disponibles au lien indiqué en [3].

Le circuit imprimé a été conçu dans un format aussi compact que possible. Il faut d'abord placer les composants CMS, en commençant par les circuits intégrés (IC). Le soudage de ces petites pièces en boîtiers TDFN peut s'avérer compliqué, et n'est pas conseillé pour les électroniciens inexpérimentés. Certains experts peuvent souder les broches sur les côtés des boîtiers avec un fer à souder à

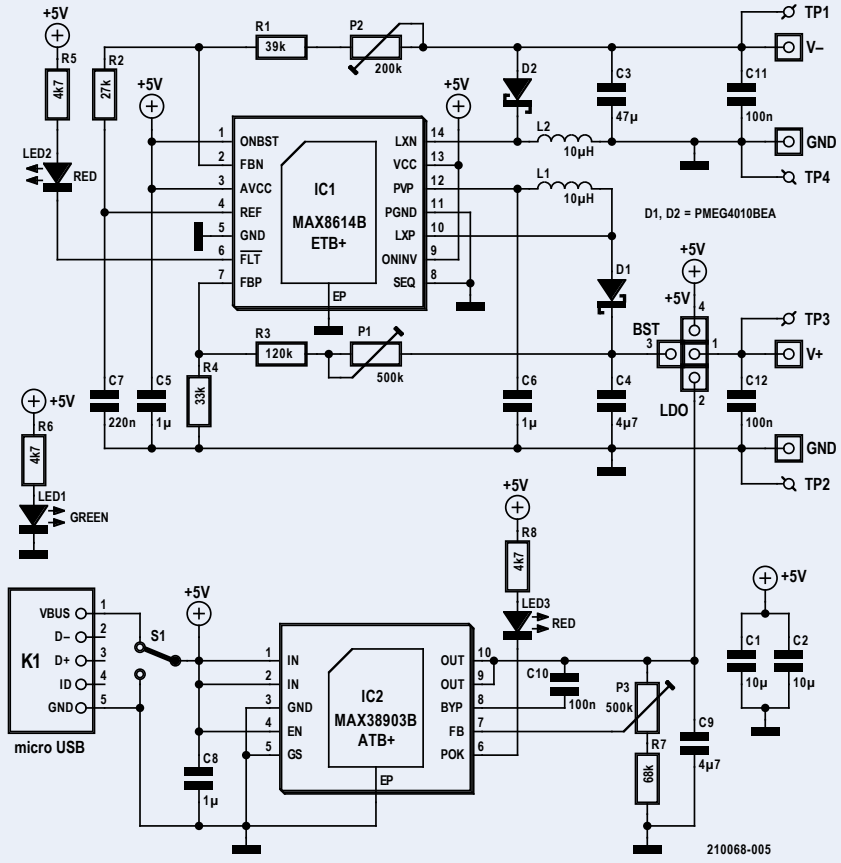


Figure 1. Schéma de l'alimentation électrique de la plaque d'expérimentation.

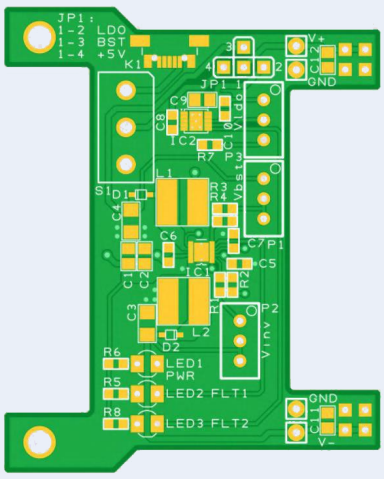


Figure 2. Dessin du circuit imprimé (fichiers Gerber et fichiers sources disponibles au téléchargement).

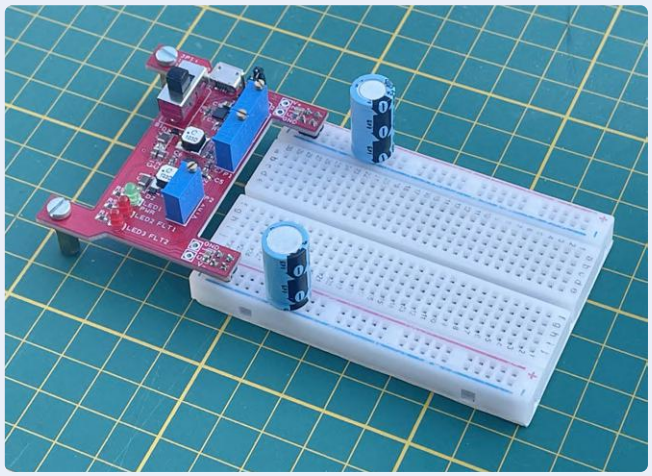


Figure 3. Le module d'alimentation sur une plaque d'expérimentation MB102 avec condensateurs électrolytiques supplémentaires.

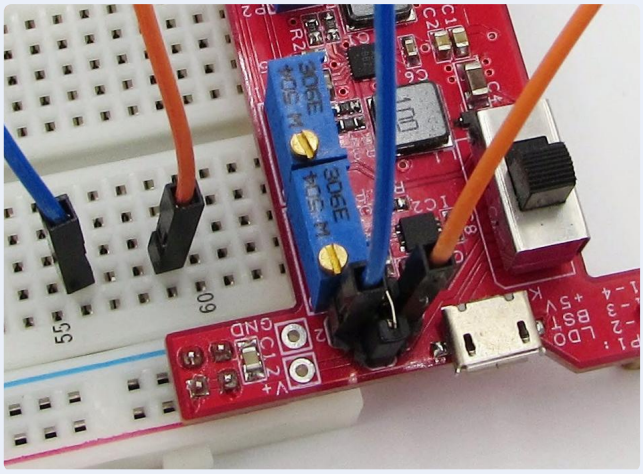


Figure 4. Rail positif branché à V_{BST}, fils de raccordement à V_{LDO} et +5 V à la plaque d'expérimentation.

pointe fine, mais le soudage des broches apparentes sur le dessous nécessite une station à air chaud ou un four de refusion. La pâte à souder reste la meilleure option, même si l'on peut construire un prototype par soudage des broches au fer à souder standard, en appliquant un surplus de flux et en utilisant une station à air chaud pour la refusion. On peut aussi utiliser un verre grossissant, un microscope numérique ou un stéréomicroscope pour inspecter les joints de soudure avant de passer aux autres composants. Le soudage du connecteur micro-USB peut sembler moins difficile, mais il vaut mieux s'en occuper dans un troisième temps. On peut ensuite monter les bobines, diodes, résistances et condensateurs environnants, puis les composants traversants comme l'interrupteur, les LED et les embases, dont le soudage est aisé au fer à souder standard. Pour JP1, on coupe deux broches extérieures sur l'une des deux rangées d'une embase à 3×2 broches pour obtenir une configuration triangulaire de cavalier à 4 broches.

Utilisation

Le module d'alimentation se branche à la plaque d'expérimentation comme indiqué en **figure 3**. Les connecteurs J1 et J2 sont branchés sur les rails d'alimentation horizontaux de la plaque d'expérimentation. Le rail supérieur est raccordé à V+, le rail inférieur à V– et les deux rails intérieurs à la masse (GND) (toutes les sources d'alimentation du circuit partagent une masse commune).

On peut placer le circuit imprimé à gauche du plan de travail avec deux boulons ou entretoises M3 de 12 mm. Deux trous de fixation sont percés à cet effet. Ceci permet d'améliorer la stabilité mécanique de la connexion entre le module d'alimentation et la plaque d'expérimentation, mais aussi de prévenir toute chute lors du réglage des tensions de sortie.

L'interrupteur S1 permet d'allumer et d'éteindre l'intégralité du module d'alimentation. Lorsque l'interrupteur est fermé, la LED1 verte indique que le montage est sous tension.

Puisque le circuit propose trois tensions d'alimentation positives, il faut sélectionner laquelle connecter au rail d'alimentation supérieur de la plaque d'expérimentation (V+). Pour cela, on utilise le cavalier JP1. Placez le cavalier en position 1-2 pour la faible tension ajustable (V_{LDO}), en position 1-3 pour la forte tension ajustable (V_{BST}) et en position 1-4 pour la tension fixe de +5 V. Les positions du cavalier sont indiquées sur le schéma d'implantation du circuit. Les deux tensions restantes peuvent être connectées au circuit de la plaque d'expérimentation à l'aide de fils mâle-femelle, comme indiqué en **figure 4**. Sur cette image, la tension V_{BST} est branchée au rail positif, tandis que les fils bleu et orange sont branchés à la tension +5 V, et la tension V_{LDO} à la plaque d'expérimentation. La tension de sortie négative V_{INV} est directement raccordée au rail d'alimentation inférieur (V–).

Le réglage des tensions est effectué grâce aux potentiomètres multi-tour P1 à P3. Le schéma d'implantation du circuit indique que le potentiomètre est associé à chaque sortie. Plusieurs points d'essai permettent au multimètre de mesurer V+ (TP3) et V– (TP1) au cours du réglage. TP2 et TP4 sont raccordés à GND.

Les deux LED rouges indiquent une surcharge ou un court-circuit de l'une des sources d'alimentation : LED2 (ou FLT1 sur le circuit) pour V_{BST} et/ou V_{INV}, et LED3 (FLT2) pour V_{LDO}. Le convertisseur élévateur et le LDO limitent le courant de sortie en cas de surcharge, et les sorties s'éteignent en cas de surchauffe du circuit régulateur. Dans les deux cas, les LED s'allumeront. Si le convertisseur abaisseur-élévateur est en surcharge, les deux sorties de IC1 seront coupées et la LED2 s'allumera. Pour remettre l'alimentation en fonctionnement normal, on l'éteint avec S1 avant de rallumer, après avoir remédié au court-circuit.

LIENS

[1] **Maxim Integrated, « MAX8614A/MAX8614B: Dual-Output (+ and -) DC-DC Converters for CCD », 12/2019** : https://datasheets.maximintegrated.com/en/ds/MAX8614.pdf

[2] **Maxim Integrated, « MAX38903A/MAX38903B/MAX38903C/MAX38903 », 4/2020** : https://datasheets.maximintegrated.com/en/ds/MAX38903A-MAX38903D.pdf

[3] **Page de ce projet** : www.elektormagazine.fr/210068-04

Les tensions de sortie seront restaurées et la LED rouge s'éteindra, sous réserve qu'il n'y ait pas de surcharge. Mais rien ne permet d'indiquer la surcharge de la tension fixe de +5 V. Dans ce cas, on estime que le module d'alimentation est protégé contre les courts-circuits. Si ce projet est conçu pour les plaques d'expérimentation, il peut aussi être utile pour d'autres applications d'alimentation simple à faible puissance. Pour cela, il suffit de raccorder les fils entre les broches de sortie à d'autres circuits électroniques (prototypes). Enfin, vous pouvez utiliser le chargeur micro-USB de 5 V sur votre plan de travail ! Ce projet figure également sur le site web d'Elektor Labs, au lien suivant : www.elektormagazine.com/labs/breadboard-power-supply

(210068-04)

Des questions, des commentaires ?
Envoyez un courriel au rédacteur (luc.lemmens@elektor.com) ou contactez Elektor (redaction@elektor.fr).

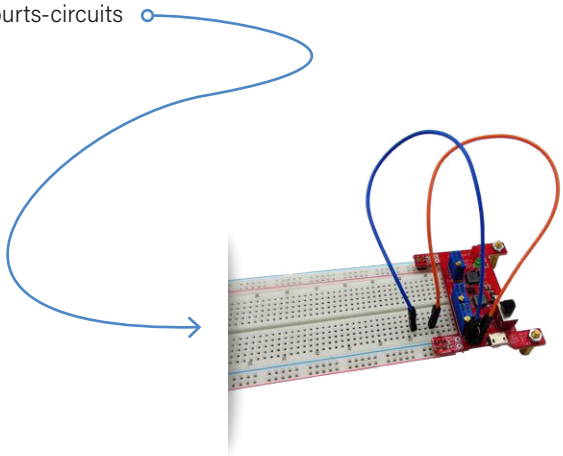
Contributeurs
Conception et texte : **Fons Janssen**
Rédaction : **Luc Lemmens**
Illustrations : **Patrick Wielders, Fons Janssen, Luc Lemmens**
Mise en page : **Harmen Heida**

Produits

- **Plaque d'expérimentation (830 points) :**
www.elektor.fr/breadboard-830-tie-points
- **Microscope numérique Andonstar AD407 HDMI avec écran LCD de 7 pouces :**
www.elektor.fr/andonstar-ad407-hdmi-digital-microscope-with-7-lcd-screen

Spécifications

- Parfaitement adapté aux plaques d'expérimentation MB102
- Tension ajustable négative de –1,8 V à –11 V (VINV)
- Tension ajustable positive de +0,6 V à +5 V (VLDO)
- Tension ajustable positive entre +5 V et +20 V (VBST)
- Tension positive fixe de +5 V (du module d'alimentation)
- Fonctionne grâce à un chargeur de téléphone standard de 5 V avec connecteur micro-USB
- Sorties (sauf +5 V) protégées contre la surcharge et les courts-circuits



LISTE DES COMPOSANTS

Résistances (taille 0603, sauf indication contraire)
R1 = 39 kΩ
R2 = 27 kΩ
R3 = 120 kΩ
R4 = 33 kΩ
R5, R6, R8 = 4,7 kΩ
R7 = 68 kΩ
P1, P3 = potentiomètres multi-tour 500 kΩ (style Vishay 24 W)
P2 = potentiomètre multi-tour 200 kΩ (style Vishay 24 W)

Inductances
L1, L2 = 10 µH 1 A (Fastron 242408FPS)

Condensateurs
C1, C2 = 10 µF, 16 V X5R 0805
C3, C4 = 4,7 µF, 50 V X7R 1206
C5, C6, C8 = 1 µF, 16 V X7R 0603
C7 = 220 nF, 16 V X7R 0603
C9 = 4,7 µF, 25 V X5R 0805
C10 = 100 nF, 16 V X7R 0603
C11, C12 = 100 nF, 50 V X7R 0805

Semi-conducteurs
D1, D2 = diode Schottky 40 V/1 A PMEG4010, SOD-323
LED1 = LED, faible puissance, 3 mm, verte
LED2, LED3 = LED, faible puissance, 3 mm, rouge
IC1 = convertisseur DC-DC à double sortie (+ et –) MAX8614BETD+
IC2 = LDO ajustable 1,7 à 5,5 VIN, 1 A, MAX38903BATB+

Divers
J1, J2 = embase à 2×2 broches (pas de 2,54 mm)
JP1 = embase à 2×3 broches (pas de 2,54 mml, voir texte)
K1 = connecteur micro-USB, Amtek MIUSB-F5M-AGB-U
S1 = interrupteur à glissière SDPT, NKK Switches CS12ANW03
Cavalier pour JP1



Raspberry Pi Pico *Essentials*

Extrait : Wi-Fi avec le Raspberry Pi Pico

Dogan Ibrahim (Royaume-Uni)

Cet article reproduit presque la moitié d'un chapitre du livre en anglais, *Raspberry Pi Pico Essentials*, de Dogan Ibrahim, publié par Elektor en mars 2021 et devenu un best-seller. Il s'agissait alors du premier ouvrage indépendant présentant des aspects concrets et éprouvés du microcontrôleur RP2040 embarqué sur le module Raspberry Pi Pico, ainsi que les logiciels correspondants. Le Pico est-il aussi accessible qu'il le prétend et est-il à la hauteur de ce que l'on attend d'une carte à contrôleur « prête à l'emploi » ? C'est ce que nous allons découvrir dans cet article.

Note de l'éditeur : cet article est un extrait du livre *Raspberry Pi Pico Essentials* 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 ? ».

Dans cet article (à l'origine la moitié du chapitre 10 du livre, *Ndlr*), nous réaliserons un projet qui utilise une liaison Wi-Fi pour établir la communication entre le RPi Pico et un smartphone.

Commande d'une LED à partir d'un smartphone en utilisant le Wi-Fi

Description : dans ce projet, nous piloterons une LED connectée au RPi Pico à partir d'un téléphone portable via la liaison Wi-Fi (pour commander un appareil, la LED peut être remplacée par ex. par un relais). Les commandes doivent être terminées par un retour chariot (CR/LF ou « nouvelle ligne »). Les commandes valides sont les suivantes :

LON	Allumer la LED
LOFF	Éteindre la LED

Objectif : présenter l'utilisation de la connexion Wi-Fi sur le RPi Pico.

Connexion Wi-Fi du Pico : le RPi Pico n'a pas de module Wi-Fi intégré. Il ne peut donc pas être connecté à un réseau Wi-Fi sans être relié à un module Wi-Fi externe. Le moyen le plus simple et le plus économique de le doter d'une fonction Wi-Fi est d'utiliser une carte processeur ESP-01. Il s'agit d'une carte minuscule (voir **fig. 1**), mesurant seulement 2,7 cm × 1,2 cm, animée par un processeur ESP8266 et coûtant environ 4 à 5 €. Voici les caractéristiques prometteuses de l'ESP-01 :

- Tension de fonctionnement : +3,3 V
- Interface : utilisation de commandes AT simples sur le port série/UART
- Pile de protocoles TCP/IP intégrée
- 802.11 b/g/n
- Aucun composant externe requis

L'ESP-01 communique avec le processeur hôte par l'intermédiaire de ses broches de port série TX et RX. Il s'agit d'une carte avec les huit broches suivantes :

VCC :	broche d'alimentation +3,3 V
GND :	masse de l'alimentation électrique
GPIO0 :	broche d'entrée/sortie. Cette broche doit être connectée à +3,3 V pour un fonctionnement normal, et à GND pour télécharger le micrologiciel sur la puce
GPIO2 :	broche d'E/S à usage général
RST :	broche de réinitialisation. Doit être connectée à +3,3 V pour un fonctionnement normal.

CH_PD : broche d'activation. Doit être connectée à +3,3 V pour un fonctionnement normal.
TX : broche de sortie série
RX : broche d'entrée série

Les broches de l'ESP-01 ne sont pas compatibles avec une platine d'essai standard, un adaptateur est donc nécessaire pour installer la carte sur une platine d'essai (**fig. 2**).
Schéma fonctionnel : la **figure 3** montre le schéma fonctionnel du projet.
Schéma de circuit : la **figure 4** montre le schéma du circuit du projet. On utilise les broches UART TX et RX du RPi Pico pour communiquer avec l'ESP-01.
Listage des programmes : le **listage 1** montre le programme (*Picowifi*). Il est inclus dans la compilation de fichiers située dans la section *Téléchargements* de la page web Elektor du livre [1]. Au début du programme, la vitesse de transmission série est réglée sur 115200, c'est le débit en bauds par défaut pour l'ESP-01, et la LED est configurée en sortie et est éteinte. La fonction `ConnectToWiFi` est appelée pour se connecter au routeur Wi-Fi local. Des commandes de type AT sont utilisées pour configurer l'ESP-01 afin qu'il se connecte au routeur Wi-Fi.

Le reste du programme s'exécute dans une boucle sans fin formée à l'aide d'une instruction `while`. À l'intérieur de cette boucle, on reçoit des données du smartphone et on commande la LED en conséquence. Les commandes `LON` et `LOFF` allument et éteignent la LED, respectivement. Les paquets de données sont reçus du smartphone à l'aide de la fonction `readline`. La fonction `find` recherche une sous-chaîne dans une chaîne de caractères et renvoie une valeur non nulle si la sous-chaîne est trouvée. On utilise la fonction `find` car les données reçues de l'appareil mobile ont le format suivant : `+ID0,n:données` (par ex. `+ID0,3:LON`) où 0 est l'ID de la liaison et *n* est le nombre de caractères reçus. En utilisant la fonction `find`, nous pouvons facilement rechercher les chaînes de caractères `LON` ou `LOFF` dans le paquet de données reçu.
La fonction `ConnectToWiFi` envoie les commandes suivantes à l'ESP-01 pour se connecter au réseau Wi-Fi :

AT+RST réinitialise l'ESP-01
AT+CWMODE définit le mode de l'ESP-01 (ici, il est réglé sur le mode Station).
AT+CWJAP définit le SSID et le mot de passe du Wi-Fi.
AT+CPIMUX définit le mode de connexion (ici, il s'agit de connexions multiples).
AT+CIFSR renvoie l'adresse IP (inutilisé ici).
AT+CIPSTART définit le mode de connexion TCP ou UDP, l'adresse IP de destination et le numéro de port (on utilise ici UDP avec le numéro de port 5000). L'adresse IP de destination est réglée sur « 0.0.0.0 » afin que n'importe quel appareil puisse envoyer des données du moment que le port 5000 est utilisé (vous pouvez remplacer cette valeur par l'adresse IP de votre smartphone pour ne recevoir des données que de votre téléphone).

Notez que chaque commande est suivie d'un petit délai. La commande `AT+CWJAP` nécessite un délai plus long. On peut facilement modifier le programme pour supprimer les délais en vérifiant les

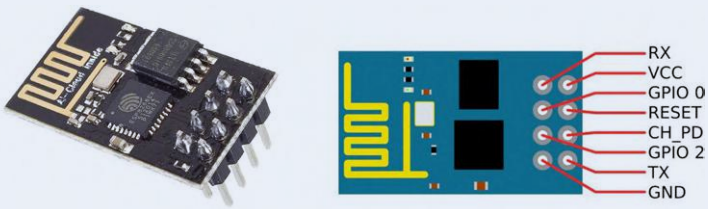


Figure 1 : Carte à processeur ESP-01.



Figure 2 : Adaptateur ESP-01 pour platine d'essai.

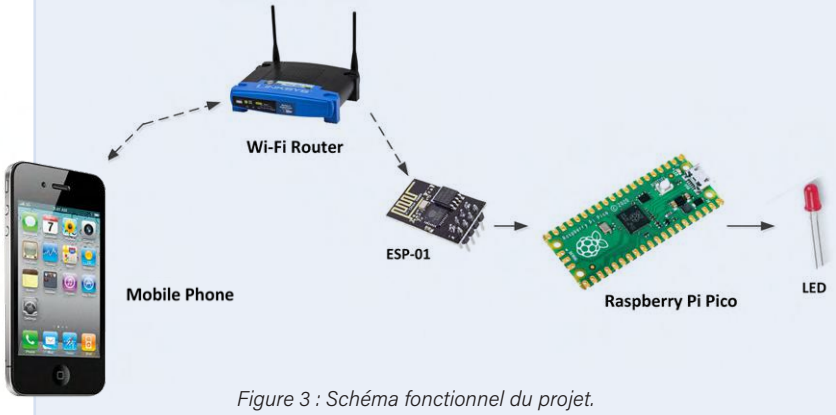


Figure 3 : Schéma fonctionnel du projet.

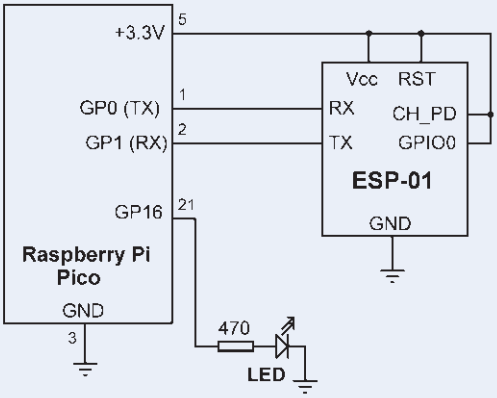


Figure 4 : Schéma du circuit du projet.

Listage 1. Programme Python pour commander un ESP-01 comme module Wi-fi.

```
# ----------------------------------------------------------
#          UTILISER LE WI-FI
#          ===============
#
# Dans ce projet, une puce ESP-01 est connectée
# au Raspberry Pi Pico. Cette puce est utilisée
# pour connecter le Pico au réseau Wi-Fi.

# Auteur : Dogan Ibrahim
# Fichier : Picowifi.py
# Date : février 2021
# ----------------------------------------------------------
from machine import Pin, UART
import utime
uart = UART(0, baudrate=115200,rx=Pin(1),tx=Pin(0))

LED = Pin(16, Pin.OUT)
LED.value(0)

#
# Envoi des commandes AT pour la connexion de l'ESP01 au Wi-Fi local
#
def ConnectToWiFi():
    uart.write("AT+RST\r\n")
    utime.sleep(5)

    uart.write("AT+CWMODE=1\r\n")
    utime.sleep(1)

    uart.write('''AT+CWJAP="BTHomeSpot-XNH","49345xyzpq"\r\n''')
    utime.sleep(5)

    uart.write("AT+CPIMUX=0\r\n")
    utime.sleep(3)

    uart.write('''AT+CIPSTART="UDP","0.0.0.0",5000,5000,2\r\n''')
    utime.sleep(3)

ConnectToWiFi()

#
# Boucle principale
#
while True:
    buf = uart.readline()          # Lecture des données
    dat = buf.decode('UTF-8')      # Décodage
    n = dat.find("LON")            # Comporte LON?
    if n > 0:
        LED.value(1)               # LED ON
    n = dat.find("LOFF")           # Comporte LOFF?
    if n > 0:
        LED.value(0)               # LED OFF
```

réponses de l'ESP-01. De cette façon, dès que la réponse correcte est reçue, le programme peut continuer. Il se peut que vous deviez réinitialiser matériellement l'ESP-01 en l'éteignant et en le rallumant avant de lancer le programme.

Test du programme

L'utilitaire *PacketSender* (**fig. 5**) sur PC ou smartphone (après installation d'une appli UDP) permet de tester sans peine le programme.
Vous devez installer une appli de serveur UDP sur votre téléphone mobile Android avant de commencer le test avec le smartphone. Il existe de nombreuses applis UDP disponibles gratuitement dans le *Play Store*. Pour ce projet, nous avons installé et utilisé *UDP/TCP Widget* de K.J.M, comme le montre la **figure 6**.
Les étapes pour tester le programme sont les suivantes :

- Construisez le circuit.
- Téléchargez le programme sur votre RPi Pico.
- Lancez l'application *UDP/TCP Widget* sur votre téléphone portable.
- Cliquez sur le symbole de l'engrenage et réglez le protocole sur *UDP*, l'adresse IP sur l'*adresse IP* de votre RPi Pico (*192.168.1.160* sur le Pico de l'auteur) et le *port* sur *5000*, comme illustré à la **figure 7**.
- Cliquez sur l'élément de menu *MESSAGE* et sélectionnez *Texte (UTF-8)* comme *Format*, et entrez la commande `LON` pour allumer la LED. Sélectionnez *LF\n* comme *Terminator* et cliquez sur le symbole OK (symbole de vérification), comme le montre la **figure 8**.
- Maintenant, cliquez sur le bouton *SEND* (**fig. 9**) pour envoyer la commande au RPi Pico. Vous devriez voir le message « *Packet Sent* » s'afficher temporairement en haut de votre écran Android.

On peut obtenir l'adresse IP de l'ESP-01 en scannant tous les appareils connectés au routeur Wi-Fi local. Vous pouvez vous servir par ex. de l'appli Android *Who Uses My WiFi - Network Scanner* de Phuongpn pour voir les adresses IP de tous les appareils connectés à votre routeur. L'ESP-01 est listé comme indiqué dans la **figure 10** (IP : 192.168.1.160), avec le nom *Espressif*.

(210198-04)

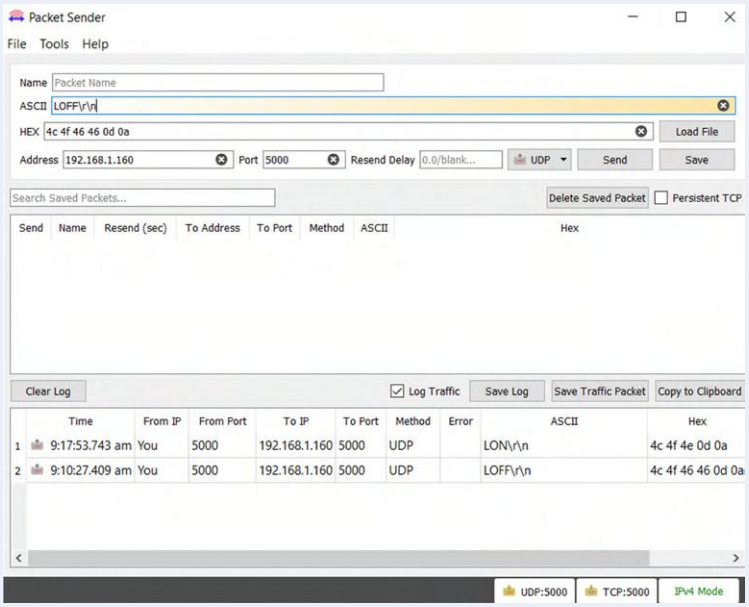


Figure 5 : Utilisation de PacketSender pour tester le programme.

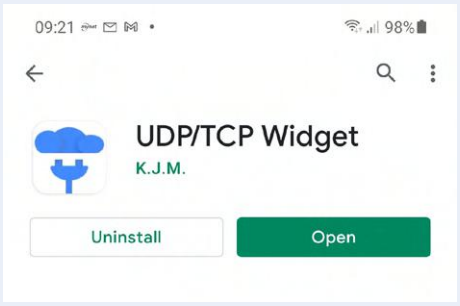


Figure 6 : Appli UDP/TCP Widget pour Android.

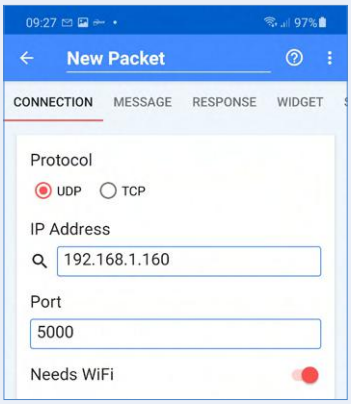


Figure 7 : Configuration de l'appli.

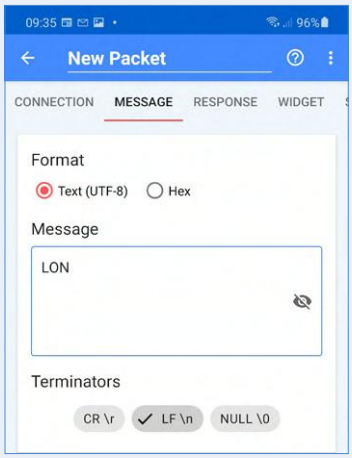


Figure 8 : Commande pour allumer la LED.

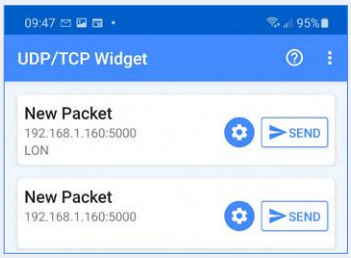


Figure 9 : Cliquez sur SEND pour transmettre la commande.

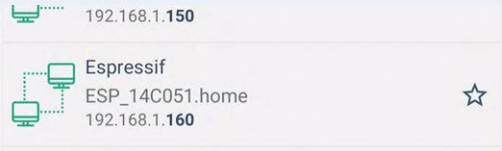


Figure 10 : Trouver l'adresse IP de l'ESP-01.

Produits



- **Livre en anglais « Raspberry Pi Pico Essentials »**
  Version papier : www.elektor.fr/raspberry-pi-pico-essentials
  Version numérique : www.elektor.fr/raspberry-pi-pico-essentials-e-book



- **Carte à microcontrôleur Raspberry Pi Pico**
  www.elektor.fr/raspberry-pi-pico-microcontroller-board

LIEN

[1] **Fichiers du programme d'exemple** : www.elektor.fr/raspberry-pi-pico-essentials

Des questions, des commentaires ?
Envoyez un courriel à l'auteur (d.ibrahim@btinternet.com) ou contactez Elektor (redaction@elektor.fr).

Contributions
Auteur : Dogan Ibrahim
Rédaction : Jan Buiting
Mise en page : Harmen Heida
Traduction : Denis Lafourcade

Lévitation magnétique sans peine

Peter Neufeld (Allemagne) et
Luc Lemmens (Elektor)



Les pages web d'*Elektor Labs* présentent deux projets de lévitation magnétique DIY similaires du designer Peter Neufeld. Dans les deux cas, une figurine Lego « flotte » dans l'air. Les deux appliquent le même principe : un capteur à effet Hall mesure l'intensité du champ magnétique entre un électroaimant et un aimant en néodyme. La valeur mesurée asservit l'électroaimant. Le circuit de commande est spécifique à chaque projet. Le premier, *The Easy Way* [2] purement analogique, s'appuie sur un comparateur, tandis que le second, *The Digital Way* [3], utilise un microcontrôleur *M5Stack Atom ESP32 Pico*. Cet article s'intéresse à la première méthode.

Sur Wikipédia, la lévitation est décrite comme le « *fait, pour un objet, de se déplacer ou de rester en suspension au-dessus du sol, sous l'effet d'une force connue* » [1]. En effet, l'objet est soumis à une force ascendante égale à la force gravitationnelle terrestre. La force électromagnétique est l'une des options de conception d'un projet de lévitation électronique. En théorie, une architecture à deux aimants *permanents* ferait également l'affaire, mais cet équilibre est *instable* et en réalité toutes sortes de facteurs le perturbent et le rompent, on utilise donc des électroaimants. Un circuit de commande peut alors réguler le champ magnétique pour maintenir l'objet flottant en place. Cela requiert un asservissement basé sur un signal représentatif de la position de l'objet flottant. Pour les deux projets décrits dans les pages d'Elektor Labs, un capteur à effet Hall mesure l'intensité du champ magnétique entre un électroaimant immobile, fixé à un cadre et un aimant permanent qui « flotte » au-dessous. La tension de sortie du capteur fournit un signal représentatif de la distance entre les deux. Le terme d'*asservissement* évoque souvent des souvenirs de calculs et de circuits élaborés et complexes nécessaires pour réaliser un système de commande stable, mais Peter Neufeld montre que cela peut être résolu de façon simple.

Les projets de lévitation attirent toujours le regard : il est toujours fascinant de voir qu'un objet flotte tout seul dans l'air. Cela semble magique, comme si les lois de la gravité ne s'appliquaient plus à cet objet, mais il n'en est rien. J'ai trouvé extraordinaire que cela puisse être fait avec si peu d'électronique. Les projets de lévitation magnétique que j'avais vus avant ceux de Peter semblaient bien plus difficiles à réaliser, souvent avec des bobines faites maison, et des courants élevés. Quoi qu'il en soit, les vidéos que Peter a postées prouvent que c'est possible, même s'il reconnait qu'il faut une certaine précision et de la dextérité pour obtenir un bon calibrage. Pour savoir si cela fonctionne, il suffit d'essayer !

Cet article couvre le projet dit *easy way* [2], c.-à-d. celui qui utilise un comparateur analogique pour asservir l'électroaimant. L'article s'intéresse surtout au fonctionnement de la lévitation. Dans un prochain article, la version avec M5Stack Atom ESP32 Pico sera exposée [3] en soignant l'esthétique pour créer un gadget de lévitation agréable à observer.

Matériel pour la version analogique

Le schéma de la **figure 1** montre que seul un matériel très simple est nécessaire pour réaliser ce projet. L1 représente la bobine d'un relais, sans mécanisme de commutation. Elle appartient à une carte relais standard (voir ci-dessous) fixée à un cadre avec le noyau dirigé vers le bas, l'objet en lévitation (avec un aimant permanent) reste en suspension sous cette bobine. R6 et la LED3 remplacent la diode *flyback* D4 sur cette carte et servent d'indicateur optique pour l'étalonnage du circuit de lévitation. Le capteur Hall U1, monté sur le noyau, mesure le champ magnétique entre L1 et l'objet flottant ; sa sortie est connectée à l'entrée inverseuse du comparateur U2, un LM311. L'entrée non-inverseuse d'U2 est connectée au curseur de RV1, trimmer de réglage du circuit d'asservissement et R3 fournit une

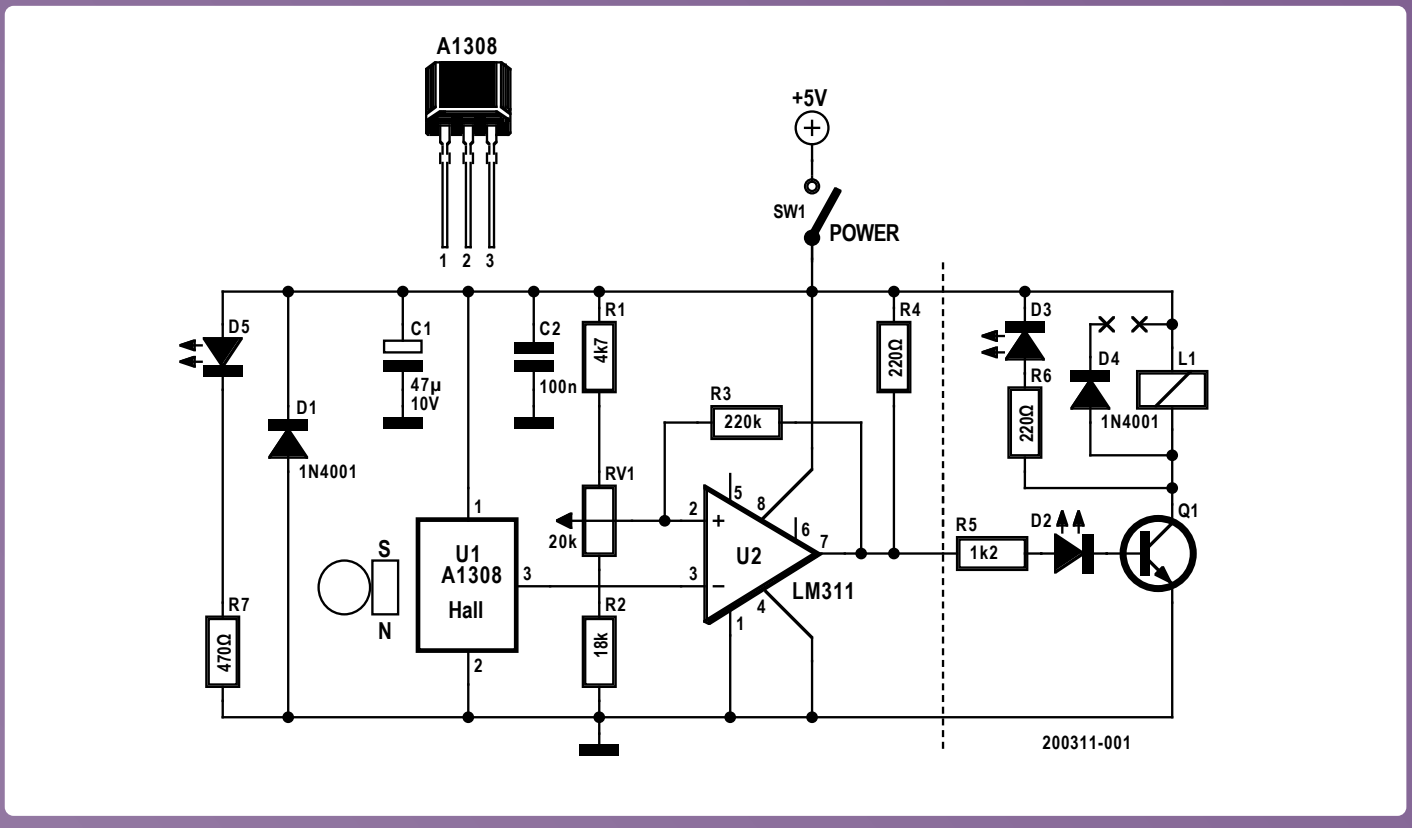


Figure 1. Schéma de ce projet de lévitation.

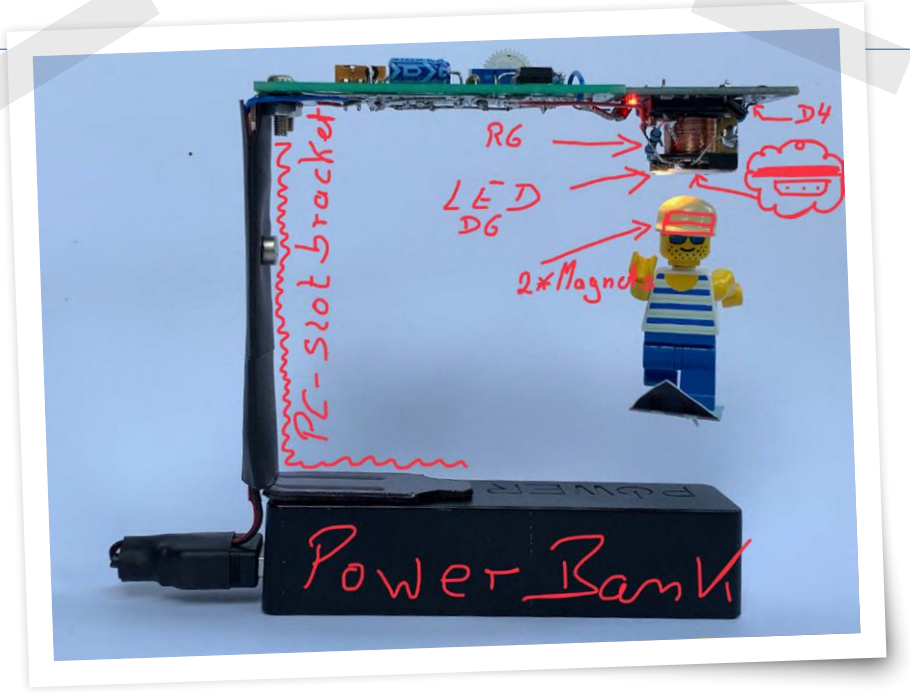


Figure 2. Vue de l'ensemble terminé.

petite hystérésis au comparateur. La sortie du comparateur active la bobine lorsque la tension à la sortie du capteur à effet Hall est inférieure au niveau défini par le curseur de RV1.
La LED D5 indique la mise sous tension et SW1 est l'interrupteur M/A. D1 est une diode de protection contre l'inversion accidentelle de la polarité d'alimentation (montage *crowbar)*.

Obtenir les (bons) composants

La plupart des composants du schéma et de la liste correspondante sont standard, seuls les capteurs à effet Hall A1302 ou A1308 d'Allegro sont un peu plus difficiles à trouver. Les capteurs moins chers et plus largement disponibles comme le SS49 n'ont pas fonctionné dans les prototypes



Figure 3a. Bornier retiré, commencez à ôter le couvercle du relais.

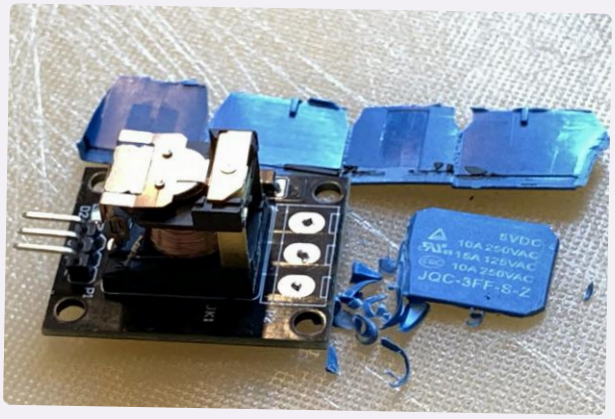


Figure 3b. Relais ouvert.

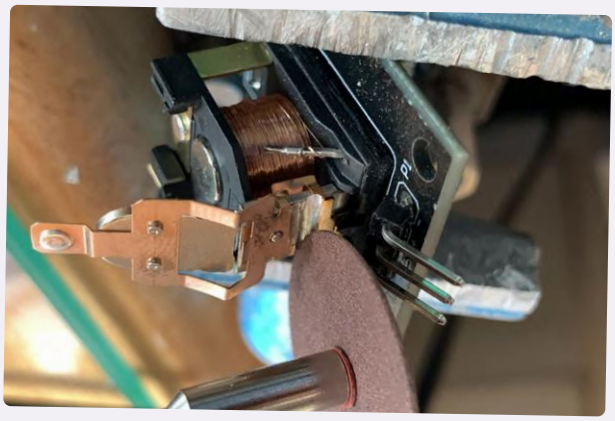


Figure 3c. Retirez le contact de l'interrupteur et coupez le noyau.

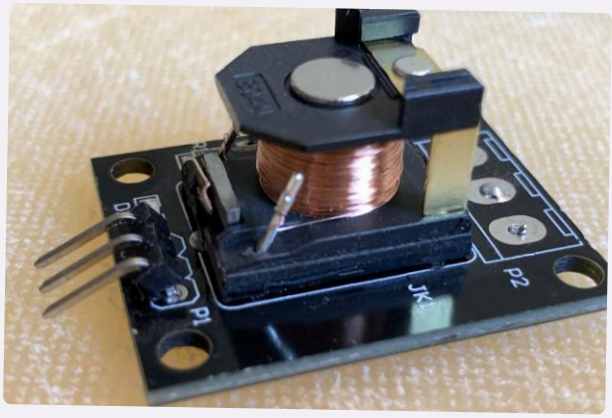


Figure 3d. Relais transformé en électroaimant.

Figure 3. Modification du relais (module).

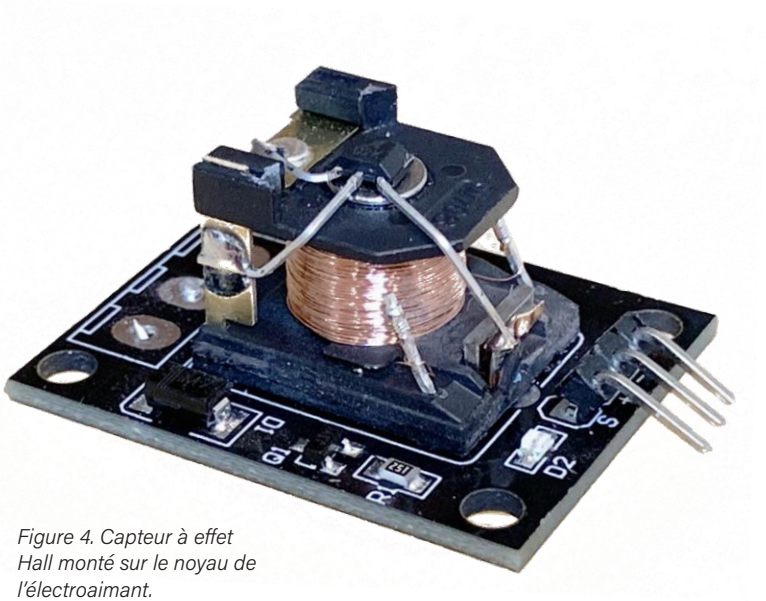


Figure 4. Capteur à effet Hall monté sur le noyau de l'électroaimant.

LISTE DES COMPOSANTS

Résistances

R1 = 4,7 kΩ
R2 = 18 kΩ
R3 = 220 kΩ
R4,R6 = 220 Ω
R5* = 1,2 kΩ
R7 = 470 Ω
RV1 = trimmer multitour 20 kΩ

Condensateurs

C1 = 47 µF, 10 V radial
C2 = 100 nF

Semi-conducteurs

D1 = 1N4001
D2*,D5 = LED rouge
D3 = LED blanche
D4* = non montée (ou à retirer de la carte de relais)
Q1* = BC550
U1 = capteur Hall A1302 ou A1308 (Allegro)
U2 = comparateur LM311

Divers

Carte relais 5 V*
SW1 = interrupteur à glissière
Disques aimantés au néodyme*

* voir le texte

des projets de Peter, sans doute parce que le signal de sortie ne réagit pas assez vite lorsque le champ magnétique varie, mais cela n'a pas été étudié de près.

Étonnamment, la pièce la plus difficile à trouver fut la carte à relais utilisée pour l'électroaimant. Ces modules sont disponibles sur internet dans beaucoup de boutiques en ligne (pour makers et Arduino) et vous devriez pouvoir en acheter sans peine. Peter avait testé des cartes marquées HW-482 avec des relais de type JQC-3FF-S-Z et JQC3F-05VDC. J'ai (ré)appris une leçon très importante quand on fait des achats sur internet : *ne jamais* se fier aux photos publiées ! Sur Amazon, j'ai trouvé exactement la même carte que celle dont Peter a posté la photo. J'ai reçu des modules dont la carte était marquée HW-307 et le relais identifié sous FL-3FF-S-Z. Ils marchent comme décrit et promis en ligne, c'est-à-dire relais SPDT de 5 V et transistor de commande, diode flyback et LED de signalisation, mais le transistor de cette carte est un PNP, contrairement au NPN des cartes relais testées par Peter. J'ai quand même essayé, en dehors du type et de la marque, le relais était semblable à celui utilisé dans le projet original ; j'ai vu plus tard que l'intérieur aussi. Pour ce projet, la bobine et le noyau du relais sont tout de même cruciaux. J'ai empilé deux ou trois aimants au néodyme en forme de disque d'un diamètre de 8 à 12 mm et d'une épaisseur de 2 à 3 mm. La taille et le nombre d'aimants nécessaires dépendent bien sûr du poids à faire léviter (figurine Lego ou autre). J'ai commencé par une simple pile de deux aimants (Ø10 mm, hauteur de 2 mm) et je vous recommande d'expérimenter avec un objet tout aussi simple pour appréhender le réglage du circuit d'asservissement. Mieux vaut marquer le dessus de la pile avec un marqueur permanent ou du ruban adhésif pour noter la bonne orientation. Une fois que ça marche, continuez avec d'autres aimants et ajoutez



un objet (par ex. figurine Lego). Il va de soi que la taille et le poids de la charge pouvant être mise en lévitation avec ce matériel sont limités. Dans cet article, le terme *aimant(s)* fait référence à la charge complète, c.-à-d. pile d'aimants plus charge optionnelle.

Montage

Aucun circuit imprimé pour ce projet, mais un morceau de Veroboard ou même une plaque d'essai fait l'affaire. La **figure 2** montre un aperçu du prototype original. Si vous construisez le matériel de ce projet, commencez par retirer le bornier à trois voies de la carte à relais afin d'avoir plus d'espace pour accéder au relais. Étudiez les photos de Peter (**fig. 3**) pour voir ce qu'il faut faire avec le relais, à savoir : retirer le couvercle et le mécanisme de commutation, seuls la bobine et le noyau sont utilisés. Il faut raccourcir le circuit magnétique (de U en J) pour ne pas court-circuiter le champ. Une meule sur miniperceuse (Dremel ou autre) convient pour couper les pièces métalliques.

Quel que soit le type de carte à relais, la diode écrêteuse D4 doit être ôtée et remplacée en ajoutant R6 et la LED blanche D3. Avec la carte à relais achetée par Peter, la partie la plus à droite du schéma de la figure 1 est alors terminée, y compris Q1, R5 et D2 qui sont présents d'origine. Dans mon cas, je n'ai gardé que la bobine du relais, j'ai enlevé les autres pièces et reconstitué cette partie du circuit avec des composants traversants. Pour Q1, tout transistor NPN standard (par ex. BC550) convient pour commander la bobine du relais. Rétrospectivement, vu les cartes que j'ai reçues, il aurait été plus logique de n'acheter que le relais, mais ces petits modules produits en série sont probablement moins chers qu'un relais séparé.

*Notez que les références des pièces de la carte de relais du schéma (L1, R5, D2 Q1 et D4) **ne** correspondent **pas** aux références des composants sur la carte. Il n'est pas difficile*

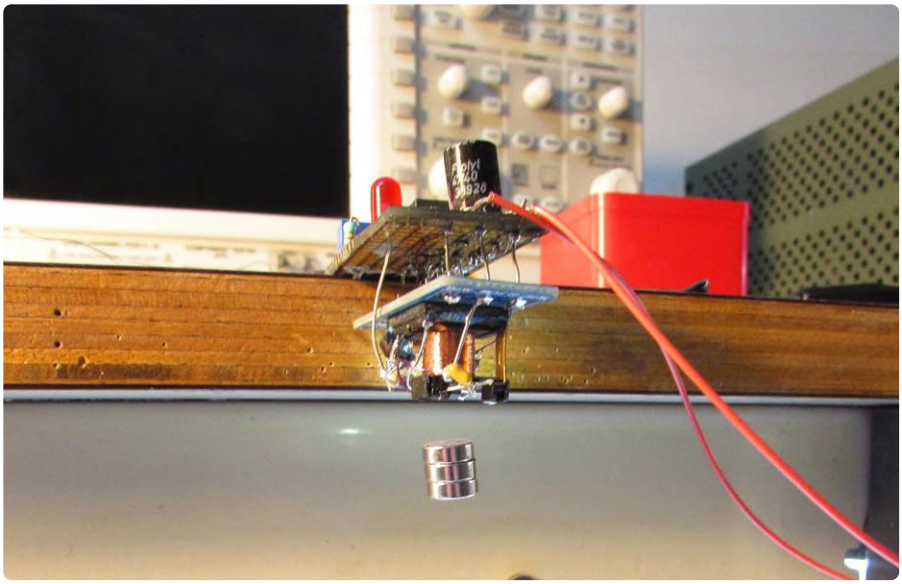


Figure 5. Version finale de mon prototype. J'aurais pu faire mieux, mais ça marche.

d'identifier la diode écrêteuse D4 qui doit être retirée.

Le capteur à effet Hall est posé directement sur le noyau de la bobine, alignez-le au centre de la surface du noyau comme indiqué sur la **figure 4**. Collez un mince morceau de plastique sur le capteur et ses broches pour éviter de court-circuiter l'alimentation si l'aimant vient se coller au noyau.

Construire le reste du circuit en utilisant la carte à relais modifiée et un morceau supplémentaire de Veroboard n'est pas difficile. Cependant, le câblage de la bobine et l'orientation de l'aimant permanent et du capteur à effet Hall sont très cruciaux pour que la lévitation magnétique fonctionne :

- quand l'aimant permanent se rapproche de la bobine et du capteur, la tension de sortie de ce dernier doit augmenter ;
- quand la bobine est alimentée, son champ magnétique doit attirer l'aimant.

Cela ne fonctionne que si ces deux conditions sont remplies. La 1ère condition est facile à vérifier en mesurant la tension de sortie du capteur Hall à l'aide d'un multimètre ; il suffit de retourner l'aimant si la tension diminue en l'approchant du capteur. Pour la seconde… je pouvais sentir l'attraction entre la bobine alimentée et l'aimant, tout semblait correct. Le circuit de commande que j'avais construit semblait bien fonctionner puisque les deux LED s'allumaient quand il le fallait : D2 avec la bobine allumée, D3 clignotait avec la bobine éteinte alors que l'aimant se rapprochait de la bobine. Il devrait suffire d'ajuster RV1 pour obtenir la tension correcte sur l'entrée non-inverseuse du comparateur pour que l'aimant attaché à une figurine Lego (ou un objet léger quelconque) lévite comme dans la vidéo. Mais rien à faire : à tout coup l'aimant s'écrasait sur la bobine ou tombait sur la table. Puis je me suis rappelé la seconde condition pour que la lévitation fonctionne. Je sentais



ALIMENTATION PAR BATTERIE

En l'absence d'aimant permanent près du noyau et du capteur Hall, la bobine est allumée en permanence et la consommation totale de courant de ce circuit est principalement déterminée par le courant dans l'électroaimant soit 75 mA env. Quand un aimant lévite sous la bobine, le courant moyen se réduit à 50 mA pour une tension d'alimentation de 5 V.

C'est étonnamment faible pour un projet de lévitation magnétique. Peter Neufeld utilise une batterie externe USB comme alimentation, cela permet de présenter le projet sur une table ou un bureau sans avoir à mettre de câble d'alimentation puisque la batterie participe au support qui soutient le matériel. La plupart des batteries externes s'éteignent automatiquement au bout de quelques secondes en cas de faible charge. Selon le type et la marque, 50 mA peuvent être inférieurs à ce seuil. Donc dans certains cas, ce type d'alimentation ne fonctionne pas, mais on peut y remédier en augmentant la consommation du circuit, par ex. en ajoutant une ou plusieurs LED de puissance.





bien que l'aimant - lorsque je le tenais entre mes doigts - était attiré par la bobine si elle était alimentée. Cela semblait aussi correct. Ou non ?

Si tout échoue, lisez le manuel, réfléchissez, ou relisez

J'ai d'abord vérifié l'intensité du champ magnétique de la bobine. J'ai été surpris de constater la faiblesse de sa force électromagnétique : alimentée directement en 5 V, elle pouvait à peine saisir le plus petit objet en fer. Il ne semble guère possible de soulever quoi que ce soit avec, et on voit encore moins comment elle pourrait faire flotter une charge relativement lourde comme un aimant auquel on aurait attaché un objet. Mais j'ai oublié quelque chose de très important : lorsque la bobine n'est pas alimentée, il existe déjà une force magnétique statique entre l'aimant permanent de la charge et le noyau métallique de la bobine. Cette dernière est bien plus forte que la force électromagnétique de la bobine. C'était donc ça l'astuce : quand l'aimant permanent s'approche du noyau, il y a un point où la force statique n'est *tout simplement* pas assez forte pour tirer l'aimant vers le noyau. Et là, le champ électromagnétique supplémentaire entre en jeu : la bobine n'ajoute qu'une petite force au champ statique, juste assez pour tirer l'aimant vers le haut. Le champ magnétique mesuré par le capteur à effet Hall augmente (et donc sa tension de sortie) à mesure que l'aimant se rapproche du noyau, et la bobine s'éteint (avec un réglage correct de RV1) bien avant que l'aimant n'atteigne le noyau. La gravité reprend le dessus et l'aimant repart vers le bas, le champ mesuré par le capteur baisse, ce qui réactive la bobine, etc.

Le champ électromagnétique étant bien plus faible que le champ magnétique statique, il est difficile de mesurer si la bobine alimentée attire ou repousse l'aimant. J'ai utilisé une bonne vieille boussole pour voir si l'orientation du champ électromagnétique était correcte pour remplir la 2e condition de lévitation. En relisant la documentation de Peter, j'ai vu qu'il indiquait une solution simple pour trouver la bonne orientation de la bobine : la tension de sortie du capteur doit augmenter si l'aimant se rapproche. Elle doit *aussi* augmenter quand on alimente la bobine (par ex. en reliant le collecteur de Q1 à GND). Il faudra peut-être intervertir les connexions de la bobine et de la LED D3 pour y arriver.

Réglage

Comme indiqué ci-dessus, il faut de la précision et de la dextérité pour trouver le point d'équilibre de la charge. Il est quelque part entre 10 et 15 mm de distance entre la bobine et la charge. Peter décrit une procédure utilisant une pile de *Post-It* pour obtenir la bonne distance pour l'étalonnage, mais sans doute par manque de précision et de dextérité, cela n'a pas fonctionné pour moi. Mon astuce fut de poser l'aimant sur ma main, et de le soulever très lentement vers la bobine jusqu'à sentir la force magnétique l'attirer. Si la bobine s'éteint avant d'atteindre ce point, il faut régler RV1 pour un seuil plus élevé sur l'entrée non-inverseuse du comparateur, ou à un niveau plus bas si elle ne s'éteint pas avant que l'aimant ne soit tiré vers le noyau. La LED blanche s'allume brièvement lorsque la bobine s'éteint, réglez le potentiomètre jusqu'à ce que cette LED semble allumée en continu (en réalité elle clignote à la fréquence de commande de l'aimant). Au début, on se retrouve avec l'aimant collé au noyau, mais une fois le coup de main pris, il devient de plus en plus facile d'obtenir un bon calibrage pour d'autres charges. Avec ma réalisation, je pouvais même entendre la commutation du circuit d'asservissement avec le réglage (presque) correct. Eh oui : y parvenir peut prendre du temps, mais vous réussirez !

Ma première intention était de construire un prototype au moins aussi joli que celui de l'auteur, mais avec toutes les modifications que j'ai dû apporter pour que ça marche, j'ai totalement échoué à cet égard : voyez la **figure 5**. Mais mon objectif principal était atteint : j'ai réussi à faire de la lévitation magnétique ! Et peut-être qu'un jour, quand je ne saurai plus quoi faire de mon temps libre... Mais je ferai plus d'efforts sur l'esthétique pour la future seconde partie, avec l'autre projet de lévitation conçu par Peter Neufeld, la *solution numérique*. Cette *simple* installation analogique servira de leçon pour éviter la plupart des pièges rencontrés et le second projet devrait être plus facile à réaliser, laissant plus de temps pour soigner l'allure du matériel !

(200311-04)

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

Des questions, des commentaires ?
Envoyez un courriel au rédacteur (luc.lemmens@elektor.com)

Produits

- **Sphère magnétique en kit**
  www.elektor.fr/magnetsphere-kit

LIENS

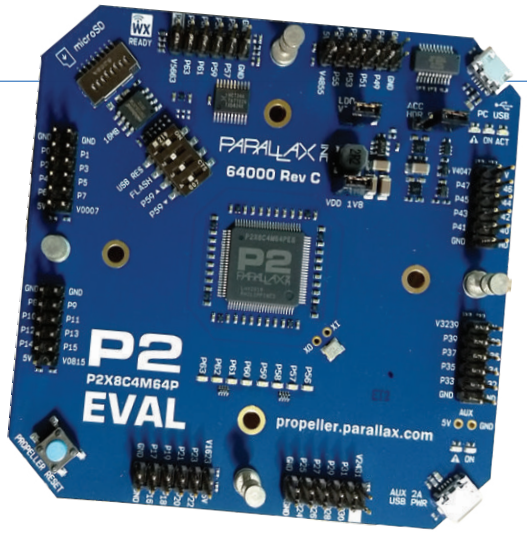
[1] **Page Wikipédia sur la lévitation** : https://fr.wikipedia.org/wiki/L%C3%A9vitation

[2] **P. Neufeld, « Magnetic Levitation - The Easy Way », site Elektor Labs, juin 2020** : www.elektormagazine.fr/labs/magnetic-levitation-the-easy-way

[3] **P. Neufeld, « Magnetic Levitation - The Digital Way », site Elektor Labs, juillet 2020** : www.elektormagazine.fr/labs/magnetic-levitation-the-digital-way

Propeller 2 de Parallax (3)

Faire clignoter une LED



Mathias Claussen (Elektor)

Dans les articles précédents, nous avons allumé une LED. Passons au clignotement. L'idée est de poursuivre la découverte des broches « intelligentes » pour créer un UART (émetteur-récepteur série universel asynchrone) capable de transmettre des caractères. Commençons par établir la communication avec le Propeller 2 de Parallax !

Dans les deux premiers articles de cette série, nous avons commencé à écrire du code pour le microcontrôleur Propeller 2 et allumé une LED avec un programme écrit en Spin2. Maintenant faisons clignoter cette LED et continuons à découvrir les broches d'E/S.

Une solution simple

Vous pouvez suivre un schéma simple : « allumer la LED, puis l'éteindre ». Pour ce faire, vous avez besoin des fonctions suivantes :

- `pinwrite()`
- `repeat()`
- `waitms()`

Les étapes successives sont :

- Allumer la LED
- Attendre 500 ms
- Éteindre la LED
- Attendre 500 ms
- Répéter

La **figure 1** montre le code. Vous pouvez le télécharger dans la page web de l'article [1].
Après avoir utilisé les broches en tant que sorties, l'étape suivante serait de montrer comment les utiliser en entrée. Cependant, nous n'allons pas le faire à ce stade et nous reviendrons sur ce sujet plus tard. Le fait de disposer d'une sortie de données en série pour communiquer avec un PC fait de l'affichage d'un état lu sur une broche d'E/S quelque chose de beaucoup plus intéressant que le simple clignotement d'une LED. De plus, nous pouvons utiliser cette sortie pour envoyer des données d'état et faire un peu de débogage du code. La prochaine étape de cette série va concerner les broches « intelligentes ».

Broches intelligentes

Aujourd'hui, les équipes de marketing adorent qualifier les produits d'« intelligents » (par ex. « ville intelligente », « données intelligentes », « contrats intelligents »). Avec les broches dites intelligentes, c'est une autre histoire, car elles vont bien au-delà des broches d'E/S habituelles à usage général. Sur d'autres microcontrôleurs, vous avez parfois la possibilité de sélectionner plusieurs fonctions pour une broche spécifique. Certains, comme l'ESP32, possèdent une matrice d'E/S qui peut acheminer n'importe quel signal d'E/S des périphériques internes vers n'importe quelle broche. Dans ce cas, une broche d'E/S ne sera toujours qu'une broche d'E/S et les fonctions UART, SPI ou CA/N se trouveront dans un bloc de matériel spécialisé, simplement connecté à la broche elle-même. Avec les broches intelligentes du Propeller 2, c'est différent car les périphériques spécialisés ne sont plus des blocs de fonctions spécifiques à l'intérieur du microcontrôleur, acheminés par une matrice d'E/S, mais ils sont plutôt intégrés, au moins partiellement, à chaque broche d'E/S. D'où le terme de *broche intelligente*.

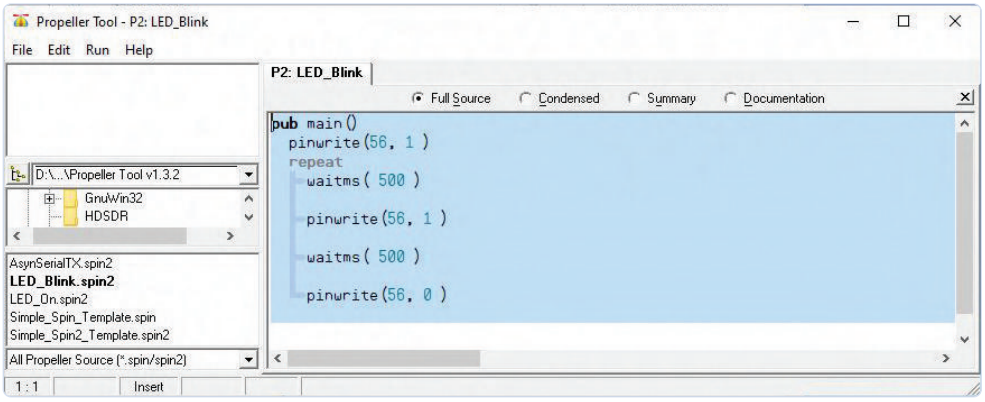


Figure 1. Code pour faire clignoter une LED.

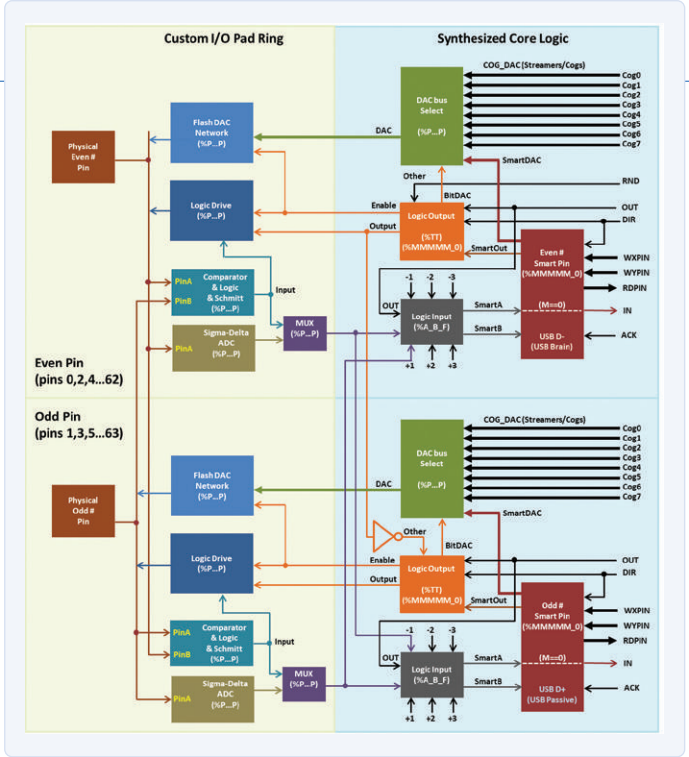


Figure 2. Présentation des broches intelligentes. (Source : Rayman/Parallax, http://bit.ly/parallax-smartpin)

L'utilisateur 'rayman' a fort à propos fait un excellent travail sur le forum Parallax [2] en apportant à la communauté un aperçu du fonctionnement interne des broches intelligentes. Vous pouvez trouver sa publication sur le Propeller 2 et l'image en [3], ou voir la **figure 2**. Dans l'encadré « Vue d'ensemble des fonctions des broches intelligentes », j'ai incorporé un extrait de la fiche technique du Propeller 2. Comme vous pouvez le voir, une broche peut servir à plusieurs fonctions. Pour le moment, c'est la configuration *11110** = *émission série asynchrone* pour envoyer des données à des fins de débogage ultérieur qui nous intéresse. La première étape consiste à déterminer comment définir le mode approprié de la broche, et si c'est possible avec Spin2 ou si nous devons y ajouter un peu de langage assembleur.

Configuration de l'UART

C'est là que le mal de tête peut arriver si c'est la première fois que travaillez avec Spin2 et le Propeller 2. La fiche technique actuelle est relativement complète. Mais tout lire et comprendre peut nécessiter beaucoup de temps. Notre objectif est d'obtenir une simple fonction, `tx()`, qui émet un caractère sur une broche, comme un UART. Nous choisissons les paramètres suivants : 115200 bauds, 8 bits de données, pas de parité et un seul bit d'arrêt. La première étape consiste à configurer la broche.

Pour ce faire, procédez comme suit :

- Configurer la broche en émission série asynchrone
- Définir le débit en bauds et les bits de données
- Activer la broche intelligente

Ces trois étapes simples permettent de paramétrer une broche sous la forme d'un UART en mode émission. Comme par la suite nous modifierons le code et le réutiliserons, nous le plaçons dans une fonction. Une fonction contient simplement du code ou des fragments de code souvent utilisés dans un programme. Vous évitez ainsi un copier-coller dans le code et vous limitez par ailleurs la maintenance à cet emplacement unique. Nous allons donc utiliser une fonction et l'appeler `serial_start`. Celle-ci ne comporte aucun argument et se résume aux trois étapes indiquées ci-dessus. La broche utilisée est actuellement codée « en dur » comme broche 57 (l'une des broches de la LED également accessible sur l'un des connecteurs périphériques, comme le montre la **figure 3**). La fonction commence par le préfixe `PUB` suivi de son nom. L'accolade finale est vide, car il n'y a pas d'arguments.

```
PUB serial_start()
WRPIN( 57, %01_11110_0  ) 'définit le mode émission
   asynchrone pour la broche tx
```

Vue d'ensemble des fonctions des broches intelligentes

00000	broche intelligente désactivée (par défaut)
00001	référentiel long (P[12:10] != %101)
00010	référentiel long (P[12:10] != %101)
00011	référentiel long (P[12:10] != %101)
00001	bruit CN/A (P[12:10]= %101)
00010	bruit de dispersion 16 bits CN/A, bruit (P[12:10]= %101)
00011	bruit de dispersion 16 bits CN/A, PWM (P[12:10]= %101)
00100*	sortie impulsion/cycle
00101*	sortie transition
00110*	fréquence oscillateur à commande numérique (NCO)
00111*	service oscillateur à commande numérique (NCO)
01000*	triangle PWM
01001*	dent de scie PWM
01010*	PWM alimentation à découpage, retour V et I
01011	périodique/continu : codeur à quadrature A-B
01100	périodique/continu : aug. sur montée A & B haut
01101	périodique/continu : aug. sur montée A & B haut / dim. sur montée A & B bas
01110	périodique/continu : aug. sur A haut {/ dim. sur montée B}
01111	périodique/continu : aug. sur A haut {/ dim. sur B haut}
10000	durée des états continus sur entrée A
10001	durée des états hauts continus sur entrée A
10010	durée de X états hauts/montées/fronts sur entrée A ou temporisation sur X états hauts/montées/fronts sur entrée A
10011	pendant X périodes, mesure du temps
10100	pendant X périodes, comptage des états
10101	pour des périodes de X cycles d'horloge minimum, mesure du temps
10110	pour des périodes de X cycles d'horloge minimum, comptage des états
10111	pour des périodes de X cycles d'horloge minimum, comptage des périodes
11000	échantillonnage/filtrage/capture CA/N, horloge interne
11001	échantillonnage/filtrage/capture CA/N, horloge externe
11010	oscillo CA/N avec déclencheur
11011*	hôte/dispositif USB (paire de broches paire/impaire = DM/DP)
11100*	émission série synchrone (données A, horloge B)
11101	réception série synchrone(données A, horloge B)
11110*	émission série asynchrone (débit en bauds)
11111	réception série asynchrone (débit en bauds)
* signal OUT forcé	
PWM = modulation de largeur d'impulsion	

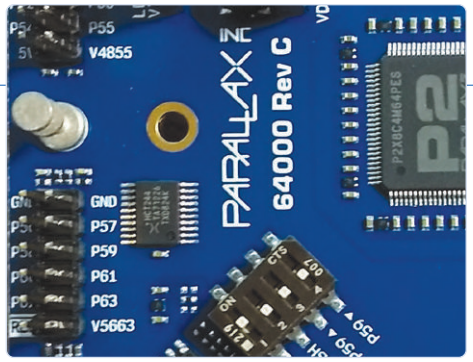


Figure 3. Emplacement de l'embase pour avoir accès aux broches de la LED.

```
PUB serial_start()
  WRPIN( 57, %01_11110_0  )              'set async tx mode for txpin
  WXPIN( 57, ((217<<16) + (8-1 )) )      'set baud rate to sysclock/115200 and word size to 8
  org
   dirh    #57
  end
```

Figure 4. Code de la fonction `serial_start()`.

```
PUB tx(val)

      WYPIN(57,val) 'load output word
org

      WAITX   #1              'wait 2+1 clocks before polling busy
      wait
      RDPIN   val,#57  WC     'get busy flag into C
      IF_C    JMP     #wait           'loop until C = 0

end
```

Figure 5. Code de la fonction `tx`.

```
WXPIN( 57, ((217<<16) + (8-1 )) )
'débit en bauds = sysclock/115200 et taille du mot = 8 bits
org' début de la partie en assembleur
dirh #57
end   'fin de la partie en assembleur
```

À partir de la ligne 1, nous avons la fonction, comme mentionné plus haut – et plus précisément, l'*en-tête de fonction*. La ligne suivante règle la broche 57 en *émission série asynchrone*, ce qu'indique la valeur 11110. Le premier bit est toujours à zéro ; les deux bits les plus significatifs, ici '01', indiquent que la broche doit être pilotée par la fonction GPIO ou Smart Pin (broche intelligente). Nous utilisons ici la fonction Spin2

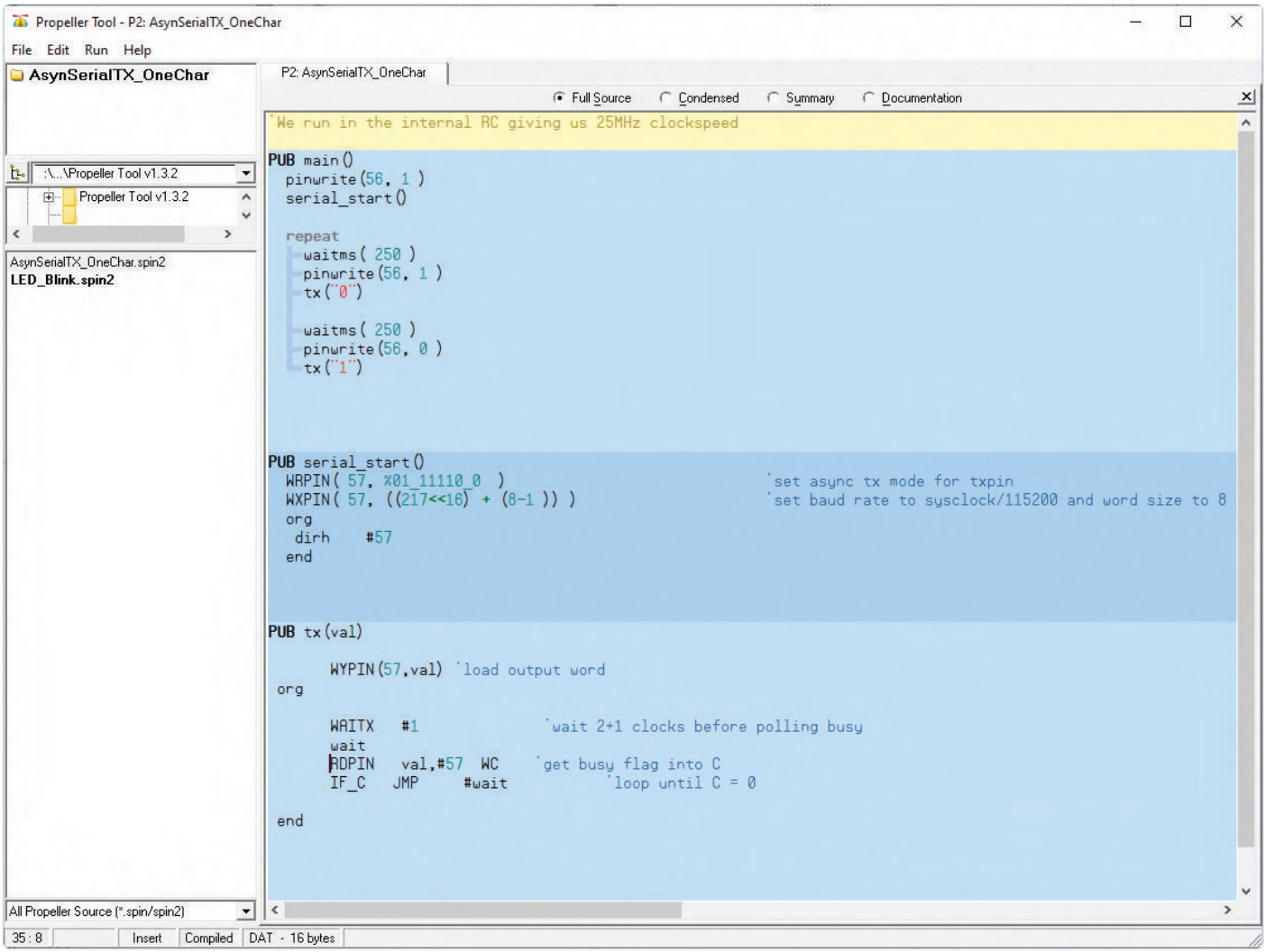


Figure 6. Code complet.

`WRPIN` pour effectuer notre première étape. La fonction suivante, `WXPIN`, définit le diviseur d'horloge et les bits de données à utiliser pour une broche intelligente en mode série asynchrone. Pour simplifier, nous ne tenons pas compte pour l'instant de la partie concernant le diviseur de débit d'émission. Il est possible de calculer la valeur de ce débit en appliquant la formule horloge système / débit en bauds – ici 25 MHz/115200 bauds – ce qui donne env. 217. Ce résultat doit être décalé de 16 vers la droite. Pour le nombre de bits à transférer, nous utilisons la formule (bits souhaités – 1), ce qui nous laisse (8 – 1) bits. Toute la magie de la configuration du débit d'émission et des bits de données est là. Les trois lignes suivantes sont différentes du code précédent. Comme il s'agit de montrer une petite section d'assembleur, quelques mots d'explications supplémentaires sont nécessaires. Avec l'indication `org`, vous pouvez incorporer une section d'instructions en assembleur, nécessaires ici. La commande `dirh` activera les fonctions de broche intelligente comme prévu pour notre étape 3. Ce qui est différent, c'est la façon d'indiquer le numéro de la broche sur laquelle nous travaillons, car il doit commencer par un '#'.

La dernière ligne ferme la section en assembleur. Dans ce cas particulier, elle correspond également à la fin de la fonction elle-même. Il aurait été préférable d'éviter l'assembleur, mais il n'y a actuellement aucun équivalent Spin2 pour l'instruction `dirh`. La **figure 4** montre le code formaté.

La broche étant dans le bon mode, nous pouvons continuer et mettre en place une fonction pour émettre un caractère et attendre jusqu'à ce que ce soit fait. Il est possible de récupérer cette fonction dans la fiche technique [4], page 91. Nous avons vu comment créer des fonctions sans arguments, ce qui signifie qu'aucune information ne leur est transmise. Pour émettre un caractère, il serait utile de pouvoir le passer à la fonction. Comme nous essayons d'éviter autant que possible l'assembleur, nous utiliserons, le cas échéant, les fonctions Spin2 plutôt que l'assembleur intégré.

Émission

Pour l'émission, nous créons une fonction `tx` presque identique à la fonction `serial_start()` comme le montre la **figure 5**.

La différence visible, outre le nom, est l'argument `val` entre parenthèses. `val` contiendra le caractère à imprimer. À l'intérieur de la fonction, nous allons d'abord écrire la valeur dans le registre de transmission de la broche 57 avec la commande `WYPIN`. La section suivante contient à nouveau quelques lignes de code en assembleur. Nous devons attendre que le drapeau 'occupé' de l'émetteur ne soit plus levé et que l'émission soit terminée. D'après la fiche technique, il faut patienter pendant trois cycles de l'unité centrale pour lire le drapeau en toute fiabilité. Cette attente est réalisée par l'instruction `WAITX` avec le paramètre `#1`, car son exécution prend deux cycles + la valeur spécifiée pour la fonction (ici un cycle). La ligne suivante est une étiquette appelée `wait`, en langage assembleur. Nous pourrons y accéder ultérieurement. L'instruction `RDPIN` en assembleur, comme écrit ici, sert à lire l'état de la broche avec report. Ce report est signalé par l'indication `WC` à la fin de l'instruction. Le bit de report, qui sert ici de drapeau d'occupation, est important car il indique si l'émission est terminée.

`RDPIN val, #57 WC` lit l'état, y compris le bit de report dans notre valeur `val`. Alors que le contenu est en cours d'émission, nous pouvons réutiliser la mémoire de `val` pour y lire l'état de la broche intelligente. L'ultime commande `IF_C JMP #wait` est un saut conditionnel ; en BASIC, ce serait l'équivalent du fameux `GOTO` combiné à une

Publicité



instruction `IF`. En clair, cela signifie : *le bit de report est-il levé (ici le drapeau 'occupé') ? Revenir à l'éqtiquette 'wait' et recommencer à partir de là, sinon continuer.* Notre émission est considérée comme terminée si le bit de report n'est plus levé. La fonction s'exécutera donc jusqu'à sa fin et reviendra là où elle a été appelée.

Assemblage des différentes parties

Nous pouvons maintenant assembler le code et insérer, après chaque fonction `pinwrite()`, l'émission d'un « 0 » ou d'un « 1 » en incorporant les fonctions `tx("0")` ou `tx("1")` dans notre code, comme le montre la **figure 6**.

Pour capturer la sortie, connectez un convertisseur série USB. À cet effet, nous avons utilisé notre fidèle Logic 16 et enregistré la sortie de la LED et l'émission en série. Le résultat apparaît dans la **figure 7** et la **figure 8**.

Et pour transmettre une chaîne de caractères ? Peut-on envoyer un simple `print("Hello World")` sur la liaison série comme nous en avons l'habitude dans l'univers Arduino ? Oui, c'est possible et nous le ferons dansle prochain article.

(200479-C-04)

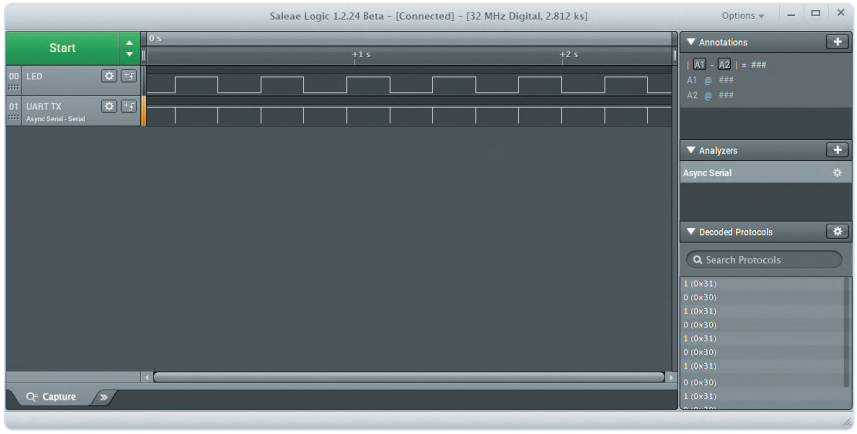


Figure 7. Tracé de l'analyseur logique avec un caractère transmis toutes les 500 ms.

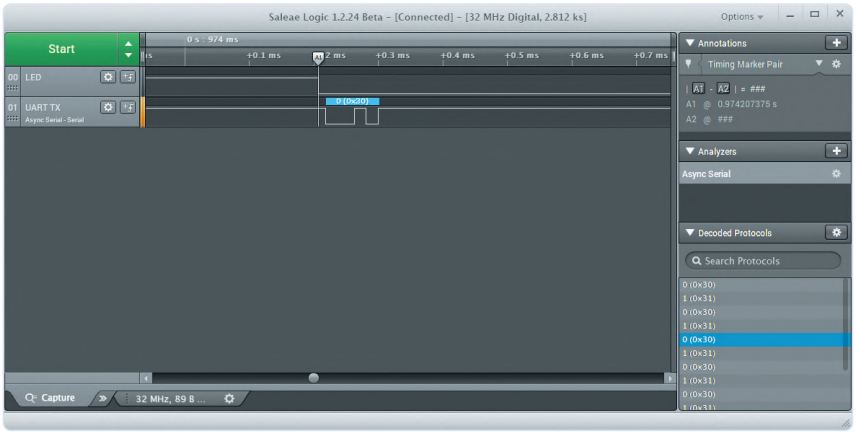
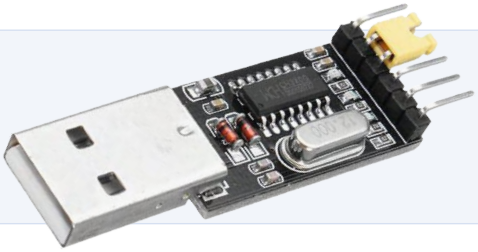


Figure 8. Tracé agrandi montrant le caractère émis.

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

Contributeurs
Conception et texte : **Mathias Claußen**
Rédaction : **Jens Nickel et C. J. Abate**
Traduction : **Pascal Godart**
Mise en page : **Giel Dols**

PRODUITS



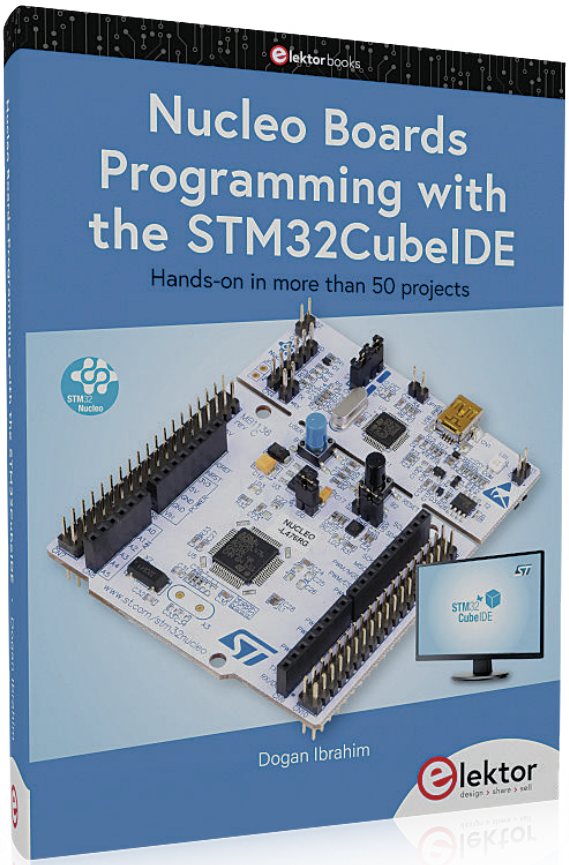
› **Convertisseur USB-TTL CH340G (3,3 V/5,5 V)**
www.elektor.fr/ch340-usb-to-ttl-converter-uart-module-ch340g-3-3-v-5-5-v

LIENS

[1] **Page de l'article** : www.elektormagazine.fr/200479-C-04
[2] **Forum Parallax** : https://forums.parallax.com
[3] **Vue d'ensemble des broches intelligentes, rayman** : https://forums.parallax.com/discussion/171420/smartpin-diagram-now-with-p-p-bit-mode-table/p8
[4] **Feuille de caractéristiques du Propeller 2 (préliminaire)** : https://docs.google.com/document/d/1gn6oaT5Ib7CytvlZHacmrSbVBJsD9t_-kmvjd7nUR6o/edit#heading=h.1h0sz9w9bl25

programmation des cartes Nucleo avec STM32CubeIDE

Extrait : FreeRTOS pour le MCU STM32



Dogan Ibrahim (Royaume-Uni)

Cet article présente une annexe extraite du livre *Nucleo Boards Programming with the STM32CubeIDE* de Dogan Ibrahim, publié par Elektor. Au cœur des projets décrits dans le livre se trouvent la carte de développement *Nucleo-L476RG*, bon marché, et le logiciel gratuit *STM32CubeIDE*. Ils forment un couple idéal pour des applications embarquées assez avancées. Pour cette courte initiation, le *STM32*, *CubeIDE* et *FreeRTOS* unissent leurs forces.

Note de l'éditeur : cet article est un extrait du livre *Nucleo Boards Programming with the STM32CubeIDE – Hands-on in More Than 50 Projects* 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 ?** ».

La plupart des systèmes en temps réel (RT) complexes traitent quasi simultanément un certain nombre de tâches (ou programmes). Prenons l'exemple d'un système RT tout simple qui doit faire clignoter une LED à intervalle fixe, et en même temps, surveiller les touches d'un clavier. Une solution serait d'écrire une tâche qui à la fois scrute le clavier en boucle à intervalle fixe et fait clignoter la LED. Bien que cette approche puisse fonctionner pour un cas simple, dans la plupart des systèmes RT complexes, il faut mettre en œuvre une approche *multitâche*.

Le multitâche consiste à traiter plusieurs tâches (ou programmes) en parallèle avec la même unité centrale (UC). Cependant, il n'est pas possible que plusieurs tâches s'exécutent en même temps sur une seule UC. On procède en fait à une commutation des tâches en partageant le temps d'utilisation de l'UC. Dans une application, les tâches sont souvent interdépendantes : il faut instaurer une forme de coopération. Par ex., l'exécution d'une tâche peut dépendre de l'achèvement d'une autre, ou une tâche peut avoir besoin de données d'une autre. Si c'est le cas, les tâches concernées doivent être synchronisées à l'aide d'une méthode de communication entre tâches.

Les systèmes RT sont sensibles au temps et leur UC n'est jamais surchargée. Dans ces systèmes, en général, les tâches obéissent strictement à des priorités. Une tâche ayant une priorité plus élevée peut s'emparer de l'UC utilisée par une tâche moins prioritaire et en avoir l'usage exclusif jusqu'à ce qu'elle la libère. Si la tâche de priorité supérieure a terminé son traitement ou attend qu'une ressource se libère, la tâche de priorité inférieure peut s'emparer de l'UC et reprendre le traitement au point où il a été interrompu.

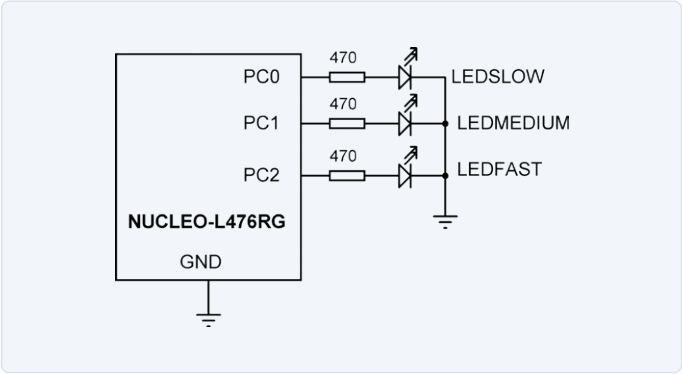


Figure 1. Schéma du circuit d'exemple pour exécuter Free RTOS sur MCU STM32.

Les systèmes en temps réel doivent aussi réagir aux événements le plus vite possible. Les événements externes sont en général traités à l'aide d'interruptions externes et la latence d'interruption doit être très courte afin que la routine de service d'interruption soit exécutée dès qu'une interruption se produit.
Dresser une liste des avantages attribués au noyau multitâche est aisé :

- Sans noyau multitâche, plusieurs tâches peuvent être exécutées en boucle, mais cette approche donne lieu à des performances en temps réel très mal contrôlées où les temps d'exécution des tâches ne peuvent être maîtrisés.
- On peut coder les différentes tâches comme des routines de service d'interruption. En pratique, cela peut fonctionner, mais si l'application comporte de nombreuses tâches, le nombre d'interruptions augmente et le code est moins gérable.
- Un noyau multitâche permet d'ajouter de nouvelles tâches ou de supprimer certaines des tâches existantes sans aucune difficulté.
- Le test et le débogage d'un système multitâche à noyau multitâche sont plus faciles à réaliser qu'avec un système multitâche sans noyau.
- La mémoire est mieux gérée avec un noyau multitâche.
- Un noyau multitâche facilite la gestion de la communication intertâches.
- Un noyau multitâche facilite le contrôle de la synchronisation des tâches.
- Un noyau multitâche facilite la gestion du temps de l'UC.
- En général, un noyau multitâche protège la mémoire en empêchant une tâche d'accéder à l'espace mémoire d'une autre.
- En général, un noyau multitâche offre la possibilité de gérer la priorité des tâches, les plus prioritaires peuvent accaparer l'UC et arrêter l'exécution des moins prioritaires. Les tâches importantes s'exécutent quand c'est nécessaire.

La nécessité d'un RTOS

Un RTOS (*Real-Time Operating System* ou système d'exploitation en temps réel) est un programme de gestion des ressources du système, il ordonnance l'exécution des tâches en son sein et en synchronise l'exécution, gère l'allocation des ressources et assure la communication et la messagerie entre les tâches. Tout RTOS comprend un noyau qui fournit les fonctions de bas niveau, principalement l'ordonnancement et la création de tâches, la communication intertâches, la gestion des ressources, etc.
Les RTOS complexes fournissent aussi d'autres services : gestion de fichiers, opérations de lecture-écriture sur disque, interruptions, gestion de réseau, gestion des utilisateurs, etc.

Dans un RTOS, une tâche est un fil d'exécution indépendant, avec son jeu local de données. Un RTOS comprend plusieurs tâches, chacune exécutant son propre code. Elles communiquent et se synchronisent entre elles pour avoir accès aux ressources partagées. L'exemple le plus simple d'un RTOS est celui où il y a par ex. trois LED et où chaque LED clignote à son propre rythme. La programmation d'un système aussi simple sans noyau multitâche pourrait être ardue. Nous verrons ici comment un RTOS tel que FreeRTOS peut être utilisé pour partager les ressources de l'UC.
Le RTOS le plus simple est un ordonnanceur qui détermine l'ordre d'exécution des tâches en son sein. Chaque tâche possède son contexte propre, constitué de l'état de l'UC et de ses registres associés. L'ordonnanceur passe d'une tâche à une autre en commutant le contexte : celui de la tâche en cours d'exécution est stocké et le contexte de la tâche suivante est chargé de sorte que l'exécution de celle-ci puisse reprendre correctement là où elle en était. Le temps que met l'UC pour changer de contexte s'appelle *temps de commutation de contexte* et est en général négligeable par rapport au temps d'exécution réel des tâches.

FreeRTOS

FreeRTOS est un RTOS qui fonctionne sur de nombreux microcontrôleurs haut de gamme, dont la famille STM32. STM32CubeIDE inclut le logiciel FreeRTOS, mais nous ne ferons pas d'appel à FreeRTOS directement. ARM a créé la bibliothèque CMSIS-RTOS, qui permet de faire des appels à un RTOS sous-jacent tel que FreeRTOS, c.-à-d. que nous ferons des appels à CMSIS-RTOS (v. 2) pour commander le FreeRTOS sous-jacent.
Vaste sujet que FreeRTOS et le multitâche : il faudrait plusieurs livres pour en expliquer toutes les fonctions. Les lecteurs qui ne connaissent pas les principes du multitâche trouveront sur l'internet des livres, tutoriels et notes d'application. Le livre *ARM-Based Microcontroller Multitasking Projects - Using the FreeRTOS Multitasking Kernel* facilitera l'assimilation des notions de multitâche et l'apprentissage de FreeRTOS dans les MCU basés sur ARM [1].

Projet FreeRTOS avec STM32MCubeIDE

Nous allons maintenant créer une application simple avec trois LED connectées à la carte de développement NUCLEO-L476RG. Les LED sont baptisées *LEDSLOW*, *LEDMEDIUM* et *LEDFAST*. Voici les connexions de ces LED et leur cadence de clignotement (voir aussi la fig. 1) :

LED	Cadence	Broche GPIO
LEDSLOW	1 s	PC0
LEDMEDIUM	500 ms	PC1
LEDFAST	250 ms	PC2

Un premier programme

Voici les étapes :

- Démarrer STM32CubeIDE.
- Choisir de démarrer une nouvelle application.
- Créer un nouvel espace de travail.
- Sélectionner le STM32L476RG comme processeur.
- Nommer le programme : *FREE*.
- Configurer *PC0*, *PC1*, et *PC2* comme sorties *GPIO_Output*. Faire un clic droit sur les broches et définir trois étiquettes utilisateur *User Labels* : *LEDSLOW*, *LEDMEDIUM* et *LEDFAST* (**fig. 2**).
- Configurer l'horloge du MCU à 80 MHz.
- Sur le côté gauche, cliquer sur *Middleware* et sélectionner *FreeRTOS*.
- Définir l'interface : *CMSIS_V2*.
- De nombreux paramètres peuvent être configurés sous l'onglet *Configuration*. Leur utilisation nécessite une bonne connaissance de FreeRTOS. Dans ce projet de démonstration, toutes les valeurs par défaut conviennent (voir **fig. 3**).
- Cliquer sur *File*, sur *Save* et enfin sur *YES* pour générer le code.
- Cliquer sur *Core*, *Src*, et double-cliquer sur *main c* pour afficher le programme principal.

Ce programme comporte trois tâches, aussi appelées fils (*threads*). Les fonctions de base du logiciel FreeRTOS sont les suivantes :

- définir les ID des fils ;
- définir les attributs des fils ;
- initialiser l'ordonnanceur (*scheduler*) ;
- créer les fils ;
- démarrer l'ordonnanceur.

`osThreadId_t` est utilisé pour définir les ID des fils. Les ID des fils de ce programme sont :

```
osThreadId_t LEDSTaskHandle; // LED lente
osThreadId_t LEDMTaskHandle; // LED moyenne
osThreadId_t LEDFTaskHandle; // LED rapide
```

Les attributs de fil définissent divers paramètres d'un fil, tels que nom, priorité, taille de pile, etc. Par ex., les attributs de fil pour la tâche lente sont :

```
const osThreadAttr_t LEDSTask_attributes =
{
.name = "LEDSTask",
.priority = (osPriority_t) osPriorityNormal,
.stack_size = 128 * 4
};
```

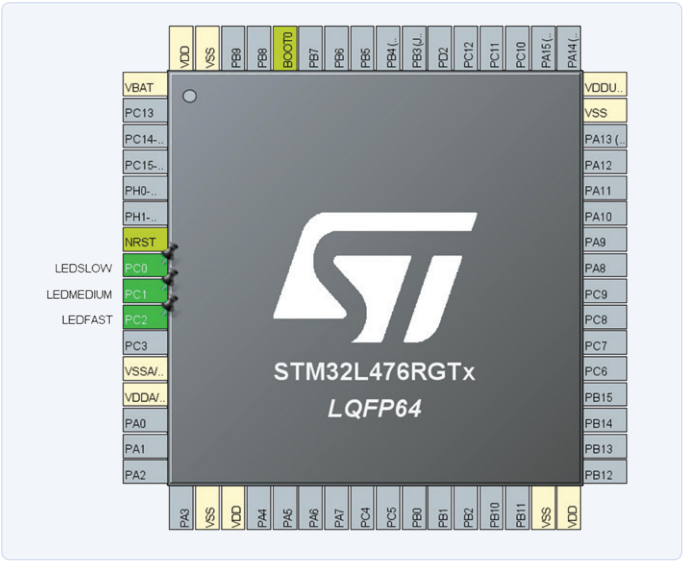


Figure 2. Configuration des sorties de l'UC STM32 dans STMCubeIDE.

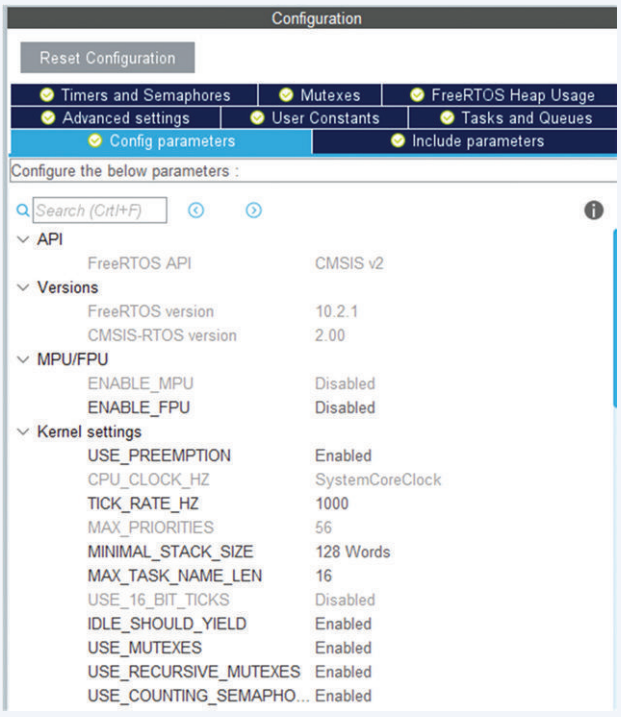


Figure 3. Pour les paramètres de configuration, accepter les valeurs par défaut.

LIENS

[1] **Livre en anglais de Dogan Ibrahim, « ARM-Based-Microcontroller-Multitasking-Projects - Using the FreeRTOS », Elektor, 2020**

[2] **Page du livre** : www.elektor.fr/nucleo-boards-programming-with-the-stm32cubeide

L'ordonnanceur est initialisé et commence par les appels de fonctions :

```
osKernelInitialize();
osKernelStart();
```

Les fils sont créés par la fonction `call osThreadNew()`. Par exemple, le fil pour la LED lente est créé comme suit :

```
LEDSTaskHandle = osThreadNew(StartLEDSTask, NULL,
   &LEDSTask_attributes);
```

Où `StartLEDSTask` est le nom de la fonction appelée par l'ordonnanceur pour réaliser le clignotement lent de la LED. Son contenu est :

```
void StartLEDSTask(void *argument)
{
for(;;)
{
HAL_GPIO_TogglePin(GPIOC, LEDSLOW_Pin);
osDelay(1000);
}
}
```

Noter qu'au lieu de `HAL_Delay()`, il faut utiliser `osDelay()`. Dans une tâche à haute priorité, `HAL_Delay()` pourrait empêcher un changement de contexte. Mais `osDelay()` indique à l'ordonnanceur de passer à une autre tâche pendant l'attente.
Compiler le programme en mode *Release* et glisser-déposer le fichier binaire *FREE.bin* sur le périphérique *NUCLEO-L476RG*. À ce stade, les trois fils sont exécutés simultanément dans leur propre boucle `sans fin`. L'ordonnanceur va commuter les fils pour donner l'impression qu'ils s'exécutent tous trois en même temps. Les trois LED doivent clignoter en même temps à des rythmes différents.

Le programme !

Le listage 1 montre le programme appelé *FREE*. Le programme peut être extrait de l'archive du logiciel libre (.zip) disponible sur la page web du livre [2]. Sur cette page, faire défiler vers le bas jusqu'à *Téléchargements* et prendre le fichier appelé *Software_Nucleo Boards Programming with the STM32CubeIDE*. Le décompresser sur un disque local, puis aller dans le dossier *FREE*.

(210236-04)

Des questions, des commentaires ?
Envoyez un courriel à l'auteur (d.ibrahim@btinternet.com) ou contactez Elektor (redaction@elektor.fr).

Contributions
Texte : **Dogan Ibrahim**
Rédaction : **Jan Buiting**
Mise en page : **Giel Dols**
Traduction : **Yves Georges**

Listage 1 : programme FREE.

```
/* USER CODE BEGIN Header */
/**
******************************************************************************
* @file : main.c
* @brief : Main program body
******************************************************************************
* @attention
*
* Copyright (c) 2020 STMicroelectronics.
* All rights reserved.
*
* This software component is licensed by ST under Ultimate Liberty license
* SLA0044, the «License»; You may not use this file except in compliance with
* the License. You may obtain a copy of the License at:
* www.st.com/SLA0044
*
******************************************************************************
*/
#include «main.h»
#include «cmsis_os.h»
//
// Define Thread IDs
//
osThreadId_t LEDSTaskHandle;          // Slow LED
osThreadId_t LEDMTaskHandle;          // Medium LED
osThreadId_t LEDFTaskHandle;          // Fast LED
```

```
//
// Slow LED task. Flash every second
//
void StartLEDSTask(void *argument)
{
   for(;;)
   {
     HAL_GPIO_TogglePin(GPIOC, LEDSLOW_Pin);
     osDelay(1000);
   }
}
//
// Medium LED task. Flash every 500ms
//
void StartLEDMTask(void *argument)
{
   for(;;)
   {
     HAL_GPIO_TogglePin(GPIOC, LEDMEDIUM_Pin);
     osDelay(500);
   }
}

//
// Fast LED task. Flash every 250ms
//
void StartLEDFTask(void *argument)
{
   for(;;)
   {
     HAL_GPIO_TogglePin(GPIOC, LEDFAST_Pin);
     osDelay(250);
   }
}

void SystemClock_Config(void);
static void MX_GPIO_Init(void);
void StartDefaultTask(void *argument);
//
// Start of main program
//
int main(void)
{
   HAL_Init();
   SystemClock_Config();
   MX_GPIO_Init();
   //
   // Slow LED Task attributes
   //
   const osThreadAttr_t LEDSTask_attributes =
   {
      .name = «LEDSTask»,
      .priority = (osPriority_t) osPriorityNormal,
      .stack_size = 128 * 4
   };
   //
   // Medium LED Task attributes
   //
   const osThreadAttr_t LEDMTask_attributes =
   {
      .name = «LEDMTask»,
```

```
        .priority = (osPriority_t) osPriorityNormal,
        .stack_size = 128 * 4
    };
    //
    // Fast LED Task attributes
    //
    const osThreadAttr_t LEDFTask_attributes =
    {
        .name = «LEDFTask»,
        .priority = (osPriority_t) osPriorityNormal,
        .stack_size = 128 * 4
    };

    /* Init scheduler */
    osKernelInitialize();

    /* creation of Tasks */
    LEDSTaskHandle = osThreadNew(StartLEDSTask, NULL, &LEDSTask_attributes);
    LEDMTaskHandle = osThreadNew(StartLEDMTask, NULL, &LEDMTask_attributes);
    LEDFTaskHandle = osThreadNew(StartLEDFTask, NULL, &LEDFTask_attributes);

    /* Start scheduler */
    osKernelStart();

    while (1)
    {
    }
}

void SystemClock_Config(void)
{
    RCC_OscInitTypeDef RCC_OscInitStruct = ;
    RCC_ClkInitTypeDef RCC_ClkInitStruct = ;

    RCC_OscInitStruct.OscillatorType = RCC_OSCILLATORTYPE_HSI;
    RCC_OscInitStruct.HSIState = RCC_HSI_ON;
    RCC_OscInitStruct.HSICalibrationValue = RCC_HSICALIBRATION_DEFAULT;
    RCC_OscInitStruct.PLL.PLLState = RCC_PLL_ON;
    RCC_OscInitStruct.PLL.PLLSource = RCC_PLLSOURCE_HSI;
    RCC_OscInitStruct.PLL.PLLM = 2;
    RCC_OscInitStruct.PLL.PLLN = 20;
    RCC_OscInitStruct.PLL.PLLP = RCC_PLLP_DIV7;
    RCC_OscInitStruct.PLL.PLLQ = RCC_PLLQ_DIV2;
    RCC_OscInitStruct.PLL.PLLR = RCC_PLLR_DIV2;
    if (HAL_RCC_OscConfig(&RCC_OscInitStruct) != HAL_OK)
    {
      Error_Handler();
    }

    RCC_ClkInitStruct.ClockType = RCC_CLOCKTYPE_HCLK|RCC_CLOCKTYPE_SYSCLK
                                  |RCC_CLOCKTYPE_PCLK1|RCC_CLOCKTYPE_PCLK2;
    RCC_ClkInitStruct.SYSCLKSource = RCC_SYSCLKSOURCE_PLLCLK;
    RCC_ClkInitStruct.AHBCLKDivider = RCC_SYSCLK_DIV1;
    RCC_ClkInitStruct.APB1CLKDivider = RCC_HCLK_DIV1;
    RCC_ClkInitStruct.APB2CLKDivider = RCC_HCLK_DIV1;

  if (HAL_RCC_ClockConfig(&RCC_ClkInitStruct, FLASH_LATENCY_4) != HAL_OK)
    {
      Error_Handler();
    }
```

```
  if (HAL_PWREx_ControlVoltageScaling(PWR_REGULATOR_VOLTAGE_SCALE1) != HAL_OK)
   {
     Error_Handler();
   }
}

static void MX_GPIO_Init(void)
{
   GPIO_InitTypeDef GPIO_InitStruct = ;

  /* GPIO Ports Clock Enable */
   __HAL_RCC_GPIOC_CLK_ENABLE();
   /*Configure GPIO pin Output Level */
   HAL_GPIO_WritePin(GPIOC, LEDSLOW_Pin|LEDMEDIUM_Pin|LEDFAST_Pin, GPIO_PIN_RESET);

  /*Configure GPIO pins : LEDSLOW_Pin LEDMEDIUM_Pin LEDFAST_Pin */
   GPIO_InitStruct.Pin = LEDSLOW_Pin|LEDMEDIUM_Pin|LEDFAST_Pin;
   GPIO_InitStruct.Mode = GPIO_MODE_OUTPUT_PP;
   GPIO_InitStruct.Pull = GPIO_NOPULL;
   GPIO_InitStruct.Speed = GPIO_SPEED_FREQ_LOW;
   HAL_GPIO_Init(GPIOC, &GPIO_InitStruct);
}

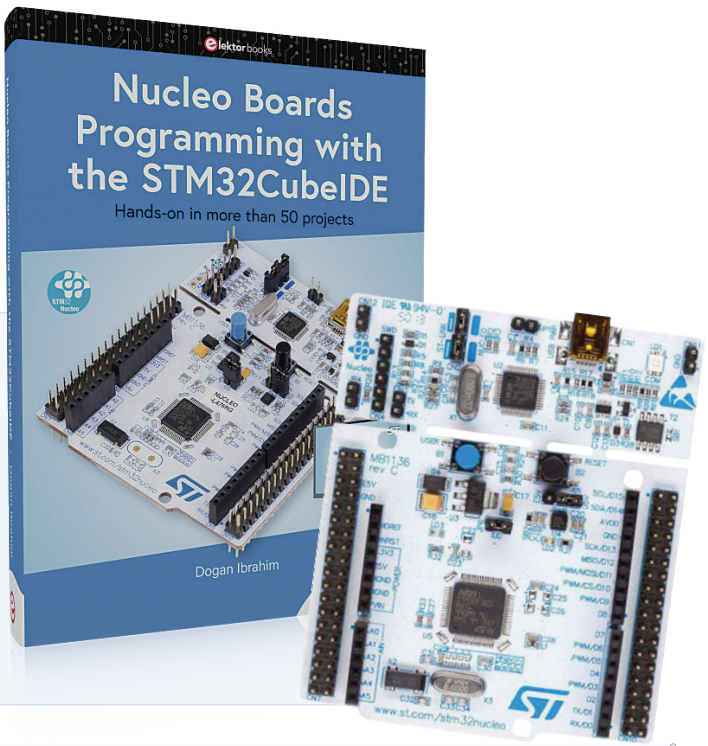
void Error_Handler(void)
{

}

#ifdef USE_FULL_ASSERT

void assert_failed(uint8_t *file, uint32_t line)
{

}
#endif
/************************ (C) COPYRIGHT STMicroelectronics *****END OF FILE****/
```



PRODUITS

- **Livre en anglais**
  « Nucleo Boards Programming with the STM32CubeIDE »

Version papier :
www.elektor.fr/nucleo-boards-programming-with-the-stm32cubeide
Version numérique :
www.elektor.fr/nucleo-boards-programming-with-the-stm32cubeide-e-book

- **Carte STM32 Nucleo L476RG**

www.elektor.fr/stm32-nucleo-l476rg-board

Extension du module convertisseur élévateur CC-CC MT3608

Johannes Sturz (Allemagne)

Sur l'internet, on trouve toutes sortes de modules électroniques pour un prix très modique. Ces modules couvrent à peu près tout : capteurs, alimentations, amplificateurs audio, cartes à microcontrôleurs, etc.

L'un d'eux est un convertisseur élévateur CC-CC basé sur le circuit intégré MT3608 d'Aerosemi (**fig. 1**). Il convertit une tension d'entrée de 2 à 24 V en une tension supérieure d'au moins 0,6 V, réglable et de 28 V max. Un trimmer permet de régler sa tension de sortie. Le CI débite jusqu'à 2 A (avec un refroidissement idoine).
Pour simplifier le circuit du module élévateur (**fig. 2**), certaines caractéristiques du CI ont été désactivées, mais pas de manière irréversible.



Figure 1. Bon marché, le module convertisseur élévateur MT3608 peut fournir jusqu'à 28 V à partir d'une tension d'entrée ≥ 2 V.

Ajouter une entrée d'activation

Le MT3608 a une entrée pour activer la puce et la mettre en veille. Lorsque la tension sur la broche *EN* d'activation est ≥ 1,5 V, le CI s'ébroue ; si elle est inférieure, on aura en sortie V_{IN} moins la chute directe de la diode Schottky D1 (0,3 V env.).
La broche *EN* permet d'activer et désactiver la puce à volonté et donc de réaliser une commande MLI. Comme nous le verrons plus loin, c'est pratique, par ex. pour commander la luminosité d'une ou plusieurs LED.

Sur le module élévateur, la broche *EN* (br. 4) n'est pas utilisable, car reliée à l'entrée du CI (br. 5). On peut lever l'obstacle en coupant la piste entre ces broches avec un scalpel (ou en soulevant la br. 4). Mieux vaut s'aider d'une loupe ou d'un microscope.

Sortie à courant constant ou driver de LED

La broche de rétroaction *FB* du MT3608 est reliée à la tension de sortie par un diviseur de

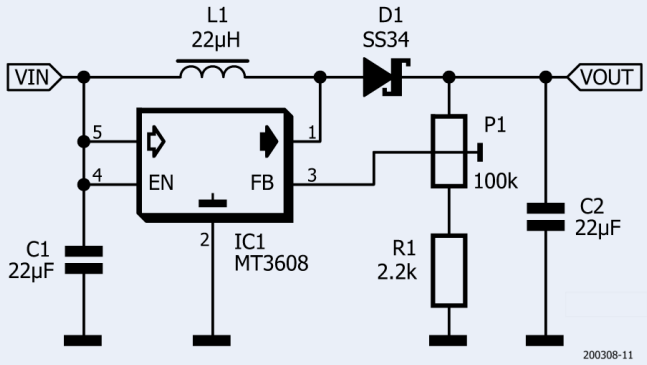


Figure 2. Schéma du module convertisseur élévateur de tension MT3608.



Figure 4. On peut récupérer des LED en bon état dans les lampes à LED hors service, car la panne vient en général du circuit de commande.

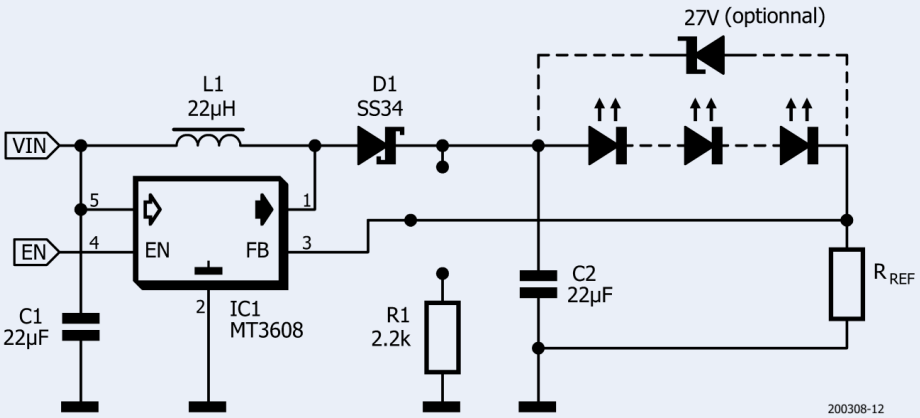


Figure 3. Le module convertisseur élévateur MT3608 transformé en gradateur de LED. Pour cela, il faut déconnecter la br. 4 du CI de V_{IN+} et retirer le trimmer.

tension (P1 et R1). Le CI règle alors sa tension de sortie de sorte à maintenir une tension constante de 0,6 V sur sa br. *FB*.

Si la branche supérieure du diviseur est remplacée par une ou plusieurs LED, le convertisseur élévateur devient une source à courant constant, lequel est déterminé par la branche inférieure du diviseur. Cela marche parce que la chute de tension d'une LED est quasi indépendante du courant qui la parcourt, de sorte que seule la tension aux bornes de la branche inférieure du diviseur agira sur la rétroaction *FB* et donc la tension de sortie du module. Celle-ci, à son tour, détermine le courant qui la traverse et qui, selon la première loi de Kirchhoff, est aussi le courant qui traverse la ou les LED.

En ôtant le trimmer du module, la broche *FB* du circuit intégré est libre et peut être reliée à une résistance externe de réglage du courant (R_{ref}) en série avec une chaîne de LED, cf. **figure 3**. De la loi d'Ohm, on déduit le courant :

$$I_{LED} = 0{,}6 / R_{ref}$$

La somme des chutes de tension sur les LED + 0,6 V ne doit pas dépasser 28 V, sous peine de détruire le MT3608.

Exemple : Arduino en variateur pour LED

Nous avons récupéré les LED encore intactes d'une lampe de 9 W en panne (**fig. 4**). Cette lampe comportait sept LED. Cependant, comme chacune contient en fait trois puces LED connectées en série, la chaîne compte 21 LED au total. À ~2,4 V par LED, la chute de tension totale est d'environ 50 V ce qui donne un courant de LED de 9 W/50 V = 180 mA env.

Une branche de trois éléments LED constitue une chaîne de neuf LED qui présente donc une chute de tension de 22 V env. C'est dans la plage de sortie du MT3608. Pour un courant de 180 mA à travers les LED, R_{ref} = 0,6/0,180 = 3,3 Ω et elle dissipera 110 mW. Les LED doivent être bien refroidies pour survivre.

Un sketch Arduino simple [1] permet alors de contrôler la luminosité des LED en envoyant des valeurs de 0 à 255 depuis le moniteur série. La br. 9 est configurée comme sortie MLI.

Dans ces expériences, le module convertisseur élévateur était alimenté par le port USB et pour ne pas surcharger le fusible de la carte Arduino, le courant de la LED a été réduit à 20 mA. À un courant plus élevé, une alimentation externe est nécessaire. Il faut aussi refroidir le MT3608, car son minuscule boîtier ne lui permet pas de dissiper plus de 0,6 W.
Cet exemple montre qu'une simple modification d'un module convertisseur élévateur de tension bon marché peut le transformer en driver de LED universel.

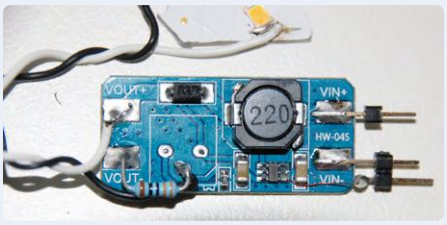
Conseils :

- Il n'est pas nécessaire de retirer le trimmer, mais cela améliore un peu l'efficacité. Si le trimmer reste en place, il faut régler V_{OUT} à sa valeur maximale.
- Avec 100 à 300 mA de courant de sortie, le rendement est > 90 %.
- La tension d'entrée doit dépasser la chute de tension dans la chaîne de LED, sinon il est impossible de l'éteindre complètement.
- Quand la broche EN est basse, la consommation de courant est réduite à quelques nA (si le trimmer a été retiré).
- Utiliser le convertisseur élévateur modifié sans les LED (par ex. en raison d'une connexion défectueuse) peut endommager le MT3608 (panne). Une diode Zener 27 V de puissance convenable entre V_{OUT+} et FB peut y remédier.
- Si la chute de tension dans la chaîne de LED dépasse 27 V, il faut la diviser en plusieurs segments. On peut alors soit piloter chaque segment par des modules séparés tout en connectant leurs entrées *EN* en parallèle, soit piloter tous les segments en parallèle en ajoutant une résistance série à chaque segment de chaîne de LED, avec bien sûr une moindre efficacité.
- Le refroidissement adéquat des LED est indispensable. La plupart des lampes LED meurent de surchauffe. Le module MT3608 nécessite aussi un refroidissement.

(200308-04)

Des questions, des commentaires ?
Envoyez un courriel au rédacteur (clemens.valens@elektor.com)

Contributeurs
Contributeurs : **Johannes Sturz, Hesam Moshiri**
Rédaction : **Clemens Valens**
Mise en page : **Harmen Heida**
Traduction : **Yves Georges**



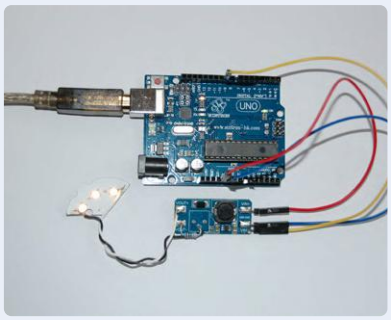


Figure 5. Un Arduino Uno et un module MT3608 modifié contrôlent ensemble la luminosité d'une chaîne de LED.

LIEN

[1] Téléchargements pour cet article : www.elektormagazine.fr/200308-04

MyVanitar - Tutoriels pour les passionnés d'électronique



Hesam Moshiri, utilisateur d'Elektor Labs, publie des tutoriels vidéo et des projets destinés aux amateurs d'électronique sur sa chaîne YouTube MyVanitar.

Testeur de composants

La plupart des oscilloscopes ont un mode XY qui affiche la tension d'une voie en fonction de la tension d'une autre voie et non en fonction du temps. Ce mode permet de créer les fameuses figures de Lissajous, mais, avec un peu d'ingéniosité, vous pouvez aussi l'utiliser comme traceur de courbes pour vérifier des composants.
Cette vidéo montre comment le mode XY associé à un générateur de forme d'onde et une résistance de 10 Ω permet de réaliser un testeur de composants ou un traceur de courbe V-I. Vous pourrez par ex. déterminer la tension de coude d'une diode Zener ou la tension directe d'une LED inconnue, comme celles utilisées ailleurs dans ces pages.
www.elektor-labs.com/4243

Décodage I²C

Une autre vidéo d'Hesam montre comment utiliser un oscilloscope pour décoder les signaux d'un bus I²C. Vous aurez besoin d'un oscilloscope capable de décoder les protocoles série, mais cette option est aujourd'hui devenue assez courante. Il vous faudra sans doute jongler un peu avec la configuration, mais une fois que ce sera fait, tout sera simple.
www.elektor-labs.com/4281

DT71 de Minware

brucelles de mesure numériques

Harry Baggen (Pays-Bas)

Les pincettes de mesure sont un outil très pratique pour identifier et mesurer certains composants, en particulier ceux de type CMS. Les brucelles DT71 de Miniware peuvent reconnaître et analyser automatiquement tous les types courants de composants passifs et offrent également quelques fonctions supplémentaires. En outre, l'instrument est d'une conception novatrice avec un afficheur rotatif.



Il existe un bon nombre de modèles de « brucelles » pour l'identification et la mesure des composants passifs. Les meilleures versions sont assez chères, entre 200 et 300 €. L'entrée de gamme se situe entre 20 et 30 €, mais ces pincettes sont souvent incapables de mesurer les inductances et leur tenue mécanique est médiocre. La lecture de la liste des possibilités offertes par la pincette DT71 de Miniware m'a convaincu qu'il s'agissait là d'un excellent modèle offrant vraiment tout ce que l'on pouvait attendre d'un tel outil. Et même si elle est un peu plus chère que ces modèles bon marché, son prix reste abordable.

Conception

La première chose qu'on remarque en déballant la DT71, ce sont ses dimensions. Elle mesure 14 cm de long et pèse moins de 25 g. Elle comprend deux parties : la partie pincette et la partie afficheur avec un petit écran OLED. Ces parties se connectent l'une à l'autre par une prise jack de 3,5 mm à 4 contacts. Au bout de l'afficheur se trouve un capteur tactile qui donne accès à toutes les fonctions. L'afficheur peut tourner sur son axe par rapport à la pincette. En outre, un capteur d'inclinaison intégré détecte si vous tenez la pincette dans votre main gauche ou droite et ajuste l'orientation de l'écran en conséquence. L'ensemble de l'appareil est réalisé en plastique avec une finition soignée.

Les branches de la pincette comportent des indicateurs de polarité rouge et bleu. Les pointes de mesure en métal sont plaquées or et sont remplaçables. Les ressorts d'écartement des branches constituent une innovation remarquable. Au lieu de maintenir les branches écartées au moyen d'un ressort mécanique, on utilise deux paires d'aimants, l'une où les aimants s'attirent et l'autre où ils se repoussent. Cette disposition des paires d'aimants permet d'obtenir un effet de ressort très doux.

La DT71 est fournie dans une petite boîte en plastique (**fig. 1**) qui



Figure 1. La DT71 se compose de deux parties. Elle est fournie avec un jeu de pointes de mesure de rechange et un câble pour la charge des batteries et la connexion à un PC.

contient, outre les parties pincette et afficheur, un jeu de pointes de mesure de rechange et un câble adaptateur avec une prise USB-C qui est utilisé pour charger les piles au lithium intégrées (**fig. 2**) et logées dans la partie pincette et pour connecter la partie afficheur à un ordinateur par un câble USB (pour modifier les paramètres et mettre à jour le micrologiciel). Le câble USB et l'adaptateur secteur ne sont pas fournis, mais la plupart d'entre nous en ont déjà sous la main.

Fonctions de mesure

Le fabricant s'est efforcé de doter la DT71 du plus grand nombre de fonctions de mesure possible. Pour commencer, il y a les mesures sur les composants : résistances, diodes, condensateurs et inductances. En réglage automatique, l'instrument recherche le type de composant le plus probable et affiche sa valeur à l'écran.

Ensuite, la DT71 peut mesurer des fréquences jusqu'à 20 MHz et des tensions continues jusqu'à 40 V.

De plus, la DT71 possède un générateur de signaux simple qui peut produire des sinusoïdes, du bruit et des impulsions avec une valeur crête à crête d'environ 3 V. L'utilisateur peut même programmer n'importe quelle forme d'onde (100 points maximum). Pour cela, il faut connecter l'afficheur à un PC et modifier le fichier *CAL.INI* dans la mémoire de la pincette (**fig. 3**). La saisie doit se faire en hexadécimal. C'est bien, mais produire rapidement une forme d'onde de cette manière me paraît un peu laborieux. Un petit assistant logiciel aurait été le bienvenu !

Le fichier *CAL.INI* contient également quelques paramètres que vous pouvez régler à votre guise, comme le temps au bout duquel l'instrument s'éteint automatiquement, l'orientation de l'affichage, la luminosité de l'écran et les différentes valeurs de fréquences préprogrammées pour les signaux sinus, utilisateur et impulsion. Toutes les options de réglage sont détaillées dans le manuel à télécharger sur le forum de Miniware [1]. Vous y trouverez également la dernière version du micrologiciel.

En pratique

J'ai testé la DT71 avec une poignée de composants (traversants et CMS) de ma collection et je l'ai comparée avec un autre testeur de composants et un multimètre précis. Miniware spécifie une précision de 0,5% pour les résistances, 2% pour les condensateurs, 5% pour les inductances et 1% pour les tensions continues. C'est plus que suffisant pour identifier les composants. De toute façon, elle n'affiche que 3 chiffres (4 dans certains cas).

En travaillant avec la DT71, j'ai remarqué que l'écran (**fig. 4**) est sympathique et net, mais très petit. J'aurais aimé qu'il fût un peu plus grand. Le comportement « élastique » des branches de la pincette avec les aimants est très agréable, mais les pointes de mesure métalliques ne sont pas assez pointues. Elles dérapent facilement du composant, notamment lorsqu'il est soudé sur un circuit imprimé. Selon le fabricant, d'autres types de pointes de mesure seront disponibles dans un proche avenir.



Figure 2. Les parties de l'écran et de la pincette peuvent être connectées ainsi et chargées via un câble USB-C.

```
/*****************************************************
          DT71 calibration parameter file
*****************************************************/
SLEEP_TIME=60
DISPLAY_DIRECTION=4
OLED_BRIGHTNESS=2
SINE_FREQ_OPT=0
NOISE_FREQ_OPT=1
USER_FREQ_OPT=2
PULSE_FREQ_OPT=3
USER_WAVEFORM = {
 0x7FF, 0x87F, 0x8FF, 0x97E, 0x9FC, 0xA77, 0xAF0, 0xB66, 0xBD9, 0xC48,
 0xCB2, 0xD18, 0xD78, 0xDD3, 0xE29, 0xE77, 0xEC0, 0xF01, 0xF3C, 0xF6F,
 0xF9A, 0xFBE, 0xFDA, 0xFEE, 0xFFA, 0xFFE, 0xFFA, 0xFEE, 0xFDA, 0xFBE,
 0xF9A, 0xF6F, 0xF3C, 0xF01, 0xEC0, 0xE77, 0xE29, 0xDD3, 0xD78, 0xD18,
 0xCB2, 0xC48, 0xBD9, 0xB66, 0xAF0, 0xA77, 0x9FC, 0x97E, 0x8FF, 0x87F,
 0x7FE, 0x77E, 0x6FE, 0x67F, 0x601, 0x586, 0x50D, 0x496, 0x424, 0x3B5,
 0x34B, 0x2E5, 0x285, 0x22A, 0x1D4, 0x186, 0x13D, 0x0FC, 0x0C1, 0x08E,
 0x063, 0x03F, 0x023, 0x00F, 0x003, 0x000, 0x003, 0x00F, 0x023, 0x03F,
 0x063, 0x08E, 0x0C1, 0x0FC, 0x13D, 0x186, 0x1D5, 0x22A, 0x285, 0x2E5,
 0x34B, 0x3B5, 0x424, 0x497, 0x50D, 0x586, 0x601, 0x67F, 0x6FE, 0x77E,
 0x000, 0x000, 0x000, 0x000, 0x000, 0x000, 0x000, 0x000, 0x000, 0x000,
 0x000, 0x000, 0x000, 0x000, 0x000, 0x000, 0x000, 0x000, 0x000, 0x000,
 0x000, 0x000, 0x000, 0x000, 0x000, 0x000, 0x000, 0x000,
}

              LV      MV      HV      RL      RM      RH      CX1     CX2     CX3     CX4     CX5
    CX6     CX7     CX8     CX9     CH1     CH2     LX1     LX2     LX3     LX4     LX5     LX6
CALRB_K0 =   -1.856,   5.286, 109.890, -10.059,  -9.349, 167.794,  -5.263,  -5.263,  -5.263,   0.000,   0.000,
  0.000,   0.000,   0.000,   0.000,   0.000,   0.000,   0.000,   0.000,   0.000,   0.000,   0.000,   0.000,
CALRB_K1 =    1.045,   1.011,   1.009,   1.020,   1.021,   1.008,   1.000,   1.000,   1.000,   1.000,   1.000,
  1.000,   1.000,   1.000,   1.000,   1.000,   1.000,   1.000,   1.000,   1.000,   1.000,   1.000,   1.000,
```

Figure 3. Le fichier de configuration CAL.INI contient un certain nombre de paramètres configurables par l'utilisateur, une table hexadécimale pour le générateur de formes d'onde arbitraires et quelques valeurs d'étalonnage.



Figure 4. L'écran OLED n'affiche généralement que la valeur, mais dans le réglage Auto, une deuxième valeur peut apparaître, comme dans le cas de cette inductance.



Figure 5. Signal de sortie du générateur de signaux. Les sommets de la sinusoïde (ici 5 kHz) sont un peu aplatis et les pas de synthèse numérique sont clairement visibles.

Je trouve que la fonction la plus importante d'une telle pincette est sa capacité à identifier le type de composant. Avec les CMS en particulier, il est souvent impossible de reconnaître leur vraie nature et c'est là que la DT71 fait du très bon travail. Il y a bien quelques situations où elle se trompe, en particulier avec les composants pour lesquels la différence entre inductance et capacité est difficile à distinguer, par ex. avec de très petites valeurs d'inductance. Mais c'est aussi le cas avec d'autres testeurs de composants. Si vous savez de quel type de composant il s'agit et que vous passez en mode manuel, la valeur correcte s'affiche.

La précision s'est avérée dépasser mes espérances. Avec les résistances et les inductances, elle était bien conforme aux spécifications. Pour les condensateurs, les résultats de mesure des différents testeurs présentaient un écart de plusieurs pour cent. Cela est dû, entre autres, à la méthode de mesure utilisée. La valeur de la DT71 était en général trop faible de quelques pour cent, mais restait très bonne. Les inductances qui ont été mesurées étaient toutes dans une tolérance de 5%. Les diodes doivent être orientées correctement par rapport aux pointes plus et moins, sinon, la DT71 n'indique rien. Une LED connectée dans le bon sens clignote, mais ça ne fonctionne pas avec les LED bleues et blanches pour lesquelles la tension de mesure n'est pas assez élevée.

Avec les tensions continues, ma pincette ne s'est trompée que de 0,1%. Ici aussi, il faut que la polarité soit correcte, sinon l'appareil indique « Negativ ». La précision des mesures de fréquence était bien en deçà de 0,1%. Le générateur de signaux produit une forme d'onde sinusoïdale dont les sommets sont quelque peu aplatis et on peut aussi voir clairement les pas de synthèse numérique, surtout aux basses fréquences (**fig. 5**). Il n'est donc pas utilisable tel quel pour les mesures audio, mais reste très bon comme signal de test. Le signal d'impulsion est en réalité une onde carrée, qui a encore une bonne forme à 100 kHz.

J'ai trouvé la DT71 très agréable à utiliser. Elle s'allume automatiquement lorsque vous la prenez en main (à partir de la version 1.08 du micrologiciel), et l'affichage bascule automatiquement lorsque vous la changez de main. La pression de son système à aimants est très douce, ce qui rend cette pincette, combinée à sa légèreté, très agréable à utiliser.

Un instrument polyvalent

La DT71 de Miniware est une pincette de mesure très pratique qui non seulement permet d'identifier divers composants passifs, mais offre également de nombreuses fonctions supplémentaires telles que les mesures de fréquence et de tension. Elle peut également servir de mini-générateur de signaux. Sa structure en deux parties est originale, avec son afficheur rotatif et ses ressorts magnétiques. Ma seule critique est la petite taille de l'écran : on aurait pu faire un peu plus grand. Mais sinon, la DT71 est un instrument de mesure polyvalent que tout électronicien amateur rêverait de recevoir comme cadeau d'anniversaire !

(210182-04)

LIEN

[1] **Forum de Miniware :** https://minidso.com/forum.php?mod=viewthread&tid=4244

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

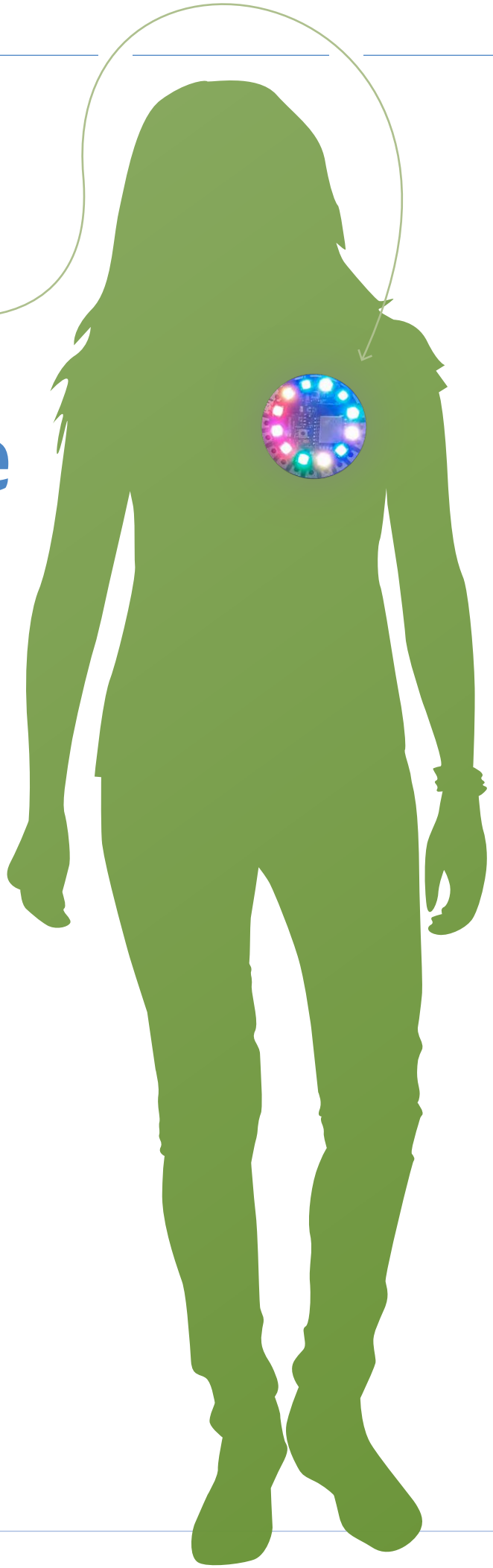
Contributeurs
Texte et illustrations : **Harry Baggen**
Rédaction : **Jens Nickel**
Traduction : **Helmut Müller**
Mise en page : **Giel Dols**

PRODUITS

Pincette de mesure DT71 de Miniware
www.elektor.fr/miniware-dt71-mini-digital-tweezers

gadget Wi-Fi vestimentaire

ESPHome à nouveau à la manœuvre !



Clemens Valens (Elektor)

Vous connaissez ces caisses pleines de cartes que vous gardez pour un jour où vous aurez plus de temps ? Ayant besoin d'un module ESP8266 pour mes expériences de domotique, j'y ai jeté un coup d'œil et j'ai trouvé une carte « vestimentaire » à base d'ESP8266 que j'avais conçue il y a quelques années. Rien de spectaculaire, juste un module ESP-12E avec une passerelle USB-série et un pilote pour une chaîne de LED adressables WS2812. J'ai redonné vie à cette carte – bien sûr avec l'aide d'ESPHome !

Il y a quelques années, un collègue m'a demandé de concevoir une carte à microcontrôleur « prête à porter » intégrant un module Wi-Fi à base d'ESP8266 ; il s'occuperait de l'aspect micrologiciel, car il avait d'ambitieux projets pour un tel montage. Lorsque le prototype a été prêt, il a essayé la carte, puis a quitté l'entreprise. Ma conception l'avait-elle déçu ou découragé à ce point ? Je ne l'ai jamais su.
Le projet a été abandonné, et il aurait sombré dans l'oubli si, quelques

mois plus tard, je n'avais pas eu un besoin urgent d'un module ESP8266 pour mes expériences de domotique. En fouillant dans des caisses remplies d'objets qui pourraient m'être utiles un jour, je suis tombé sur mon prototype ESP8266 vestimentaire. Et, comme j'étais obsédé par ESPHome [1] à l'époque, j'ai immédiatement compris que j'avais maintenant les outils pour enfin créer le logiciel pour cette carte.

Un circuit de type NodeMCU

Le schéma de la carte (**fig. 1)** est en gros un NodeMCU où la passerelle USB-série CP2101 de Silicon Laboratories (Silabs) a été remplacée par le FT231XS de FTDI, beaucoup moins cher. De plus, j'ai ajouté un port pour une chaîne de LED adressables à base de WS2812 (NeoPixels si vous préférez).

Comme le gadget vestimentaire est quasiment un NodeMCU, tout ce qui suit est également valable pour un module NodeMCU normal. Le logiciel présenté fonctionne également très bien.

Les ports GPIO disponibles et l'alimentation sont amenés sur des *pads* spéciaux avec de grands trous placés tout autour de la carte ronde. Ces *pads* sont prévus pour être utilisés avec un filetage conducteur, mais vous pouvez bien sûr aussi y souder des fils ou utiliser des pinces crocodiles.

L'alimentation du circuit est fournie soit par le port micro-USB, soit par la connexion d'une alimentation externe de 5 V à l'une des broches de 5 V. C'est utile lorsque vous utilisez de longues chaînes de LED qui demandent plus d'énergie que ce qu'un port USB normal peut fournir. On peut aussi choisir une banque d'alimentation USB à haute capacité. Notez que l'interface est destinée à des chaînes de LED de 5 V. Notez également que l'entrée 5 V n'a pas de protection contre l'inversion de polarité.

Conception du logiciel avec ESPHome

Pour mes expériences, j'ai connecté une chaîne de douze LED WS2812 en forme d'anneau au port K2 (**fig. 2)**. Bien sûr, vous pouvez écrire le logiciel de cette carte en partant de zéro, comme mon ancien collègue avait prévu de le faire, et aurait dû probablement faire, car il n'y avait pas autant de code ESP8266 disponible qu'aujourd'hui, mais adopter un projet *open source* comme ESPHome vous épargne une énorme quantité de travail.

Je l'ai déjà dit et je le répète encore une fois, ESPHome permet de créer en quelques minutes une application connectée pour l'ESP8266 ou l'ESP32, avec une programmation *over-the-air* (OTA), un *hotspot*

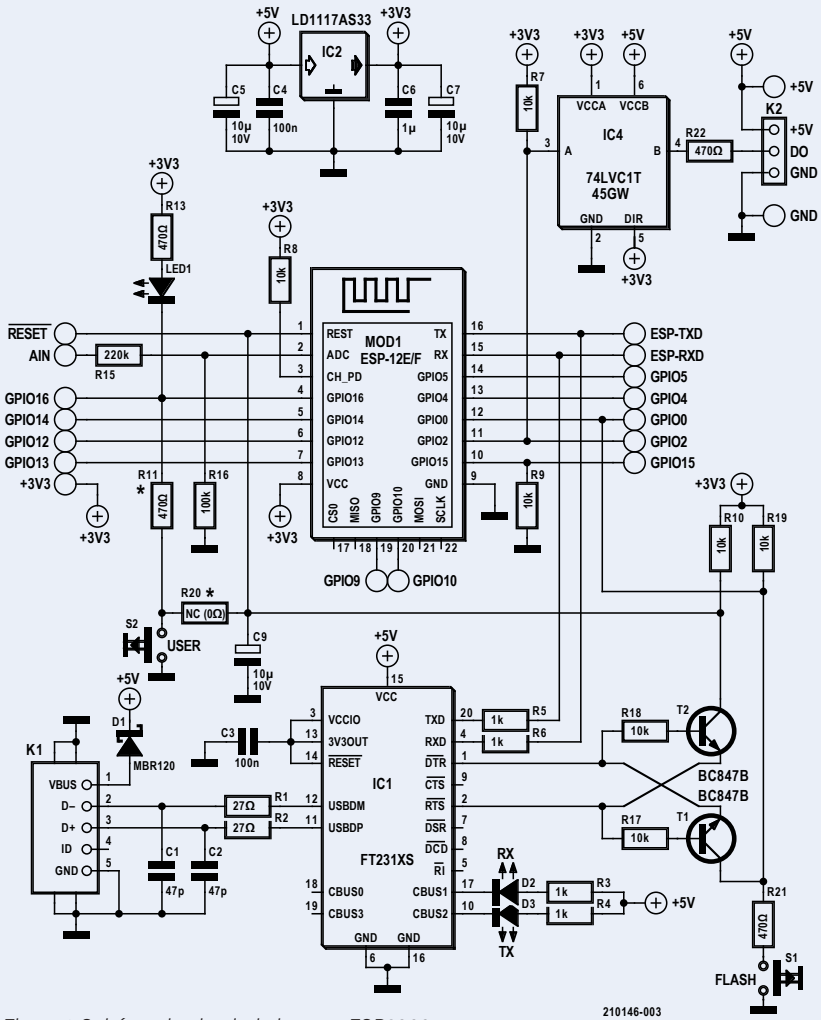


Figure 1. Schéma du circuit de la carte ESP8266.

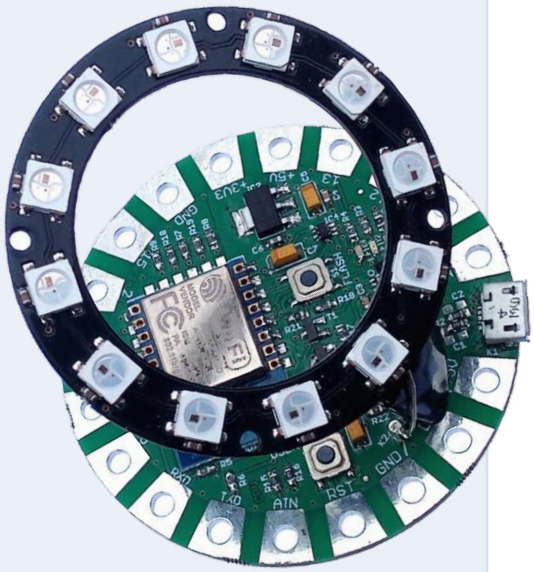


Figure 2. Le prototype du Gadget Wi-Fi vestimentaire, équipé d'un anneau de 12 LED RVB adressables (type NeoPixel).



Listage 1. Le fichier de configuration YAML [2].

```
# Elektor 160112 Wearable ESP8266
 # Configuration file for ESPHome

esphome:
   name: wearable
   platform: ESP8266
   board: nodemcu

wifi:
   ssid: "my_ssid"
   password: "my_passphrase"
  ap:
     ssid: "Wearable Fallback Hotspot"
     password: "12345678"

captive_portal:

# Enable logging
 logger:

# Enable Home Assistant API
 api:

# Enable Over-the-Air updates.
 ota:

output:
   - platform: gpio
     id: "blue_led"
     pin:
       number: GPIO16
       inverted: True

light:
   - platform: binary
     name: "Blue LED"
     output: "blue_led"
   - platform: neopixelbus
     name: "Light Ring"
     num_leds: 12
     type: GRB
     pin: GPIO2
     method: ESP8266_UART1
     effects:
       - addressable_color_wipe:
           name: "Color Wipe"
       - addressable_fireworks:
           name: "Fireworks"
       - flicker:
           name: "Flicker All"
       - addressable_flicker: # Doesn't work?
           name: "Flicker Individually"
       - addressable_rainbow:
           name: "Rainbow"
       - random:
           name: "Random All"
       - addressable_scan:
           name: "Scan"
       - strobe:
           name: "Strobe All"
       - addressable_twinkle:
           name: "Twinkle"
       - addressable_random_twinkle:
           name: "Twinkle Random"

# GPIO11 is somehow related to flash and should
not be used.
 switch:
   - platform: gpio
     name: "GPIO4"
     pin: GPIO4
   - platform: gpio
     name: "GPIO5"
     pin: GPIO5
   - platform: gpio
     name: "GPIO12"
     pin: GPIO12
   - platform: gpio
     name: "GPIO14"
     pin: GPIO14
   - platform: gpio
     name: "GPIO15"
     pin: GPIO15

# Pushbutton on GPIO0.
 binary_sensor:
   - platform: gpio
     name: "Flash"
     pin:
       number: GPIO0
       inverted: True

sensor:
   - platform: adc
     name: "Analog Input"
     pin: A0
     update_interval: 60s
     filters:
       - multiply: 3.2 # voltage divider is 100k/
(220k+100k)
```

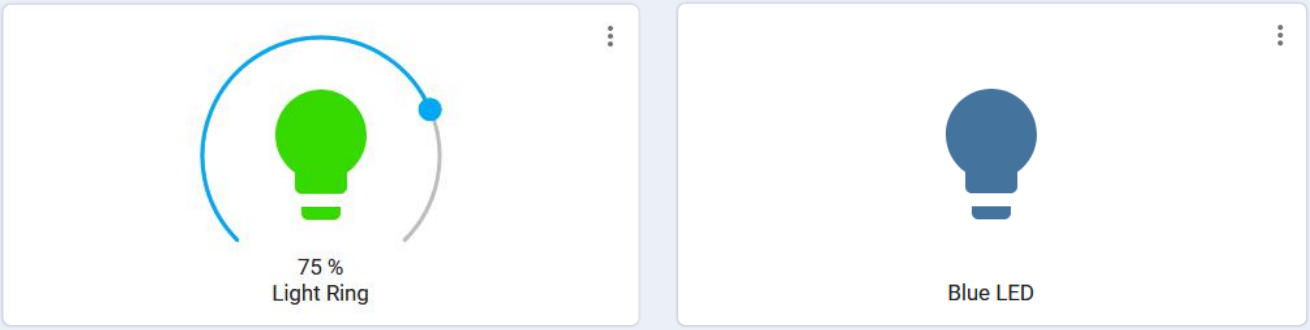


Figure 3. Ces cartes dans Home Assistant permettent de contrôler l'anneau de LED et la LED bleue du gadget Wi-Fi vestimentaire.

de secours, une interface utilisateur de serveur web et des interfaces pour plus de 200 appareils. Vraiment ! Reportez-vous à [1] pour plus de détails.

ESPHome utilise une approche modulaire où des blocs de code prêts à l'emploi sont combinés pour former une application. Les blocs requis par l'application sont répertoriés dans un fichier de configuration dit YAML (pour « *Yet another markup language* », en français « Encore un autre langage avec balises ») ; ce n'est pas un langage, mais un ensemble de règles de formatage de fichiers texte pour spécifier des paramètres et des valeurs [1]. Chaque bloc est configuré individuellement, pour spécifier par ex. le ou les ports GPIO qu'il doit utiliser, ou le protocole de communication ou son type.

Le fichier de configuration est lu par ESPHome et transformé en code C++, qui est ensuite compilé en un exécutable qui peut être programmé dans la mémoire flash du module. Une fois que vous avez compris comment composer un fichier de configuration, c'est parti. Voyez [1] pour plus de détails.

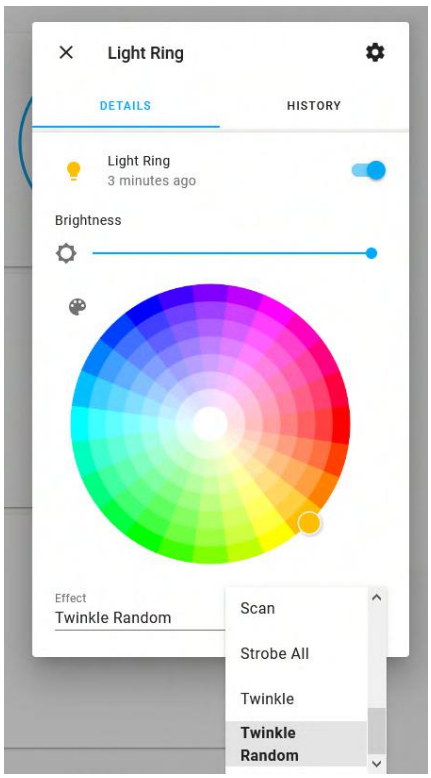


Figure 4. Les effets lumineux déclarés dans le fichier de configuration YAML de ESPHome s'affichent dans Home Assistant sous forme d'une liste dans laquelle vous pouvez choisir celui qui vous convient.

Gros plan sur le fichier de configuration

Le fichier de configuration YAML (**listage 1)** commence par la section `esphome:` obligatoire pour spécifier un nom, le MCU utilisé (ESP8266) et le type de carte (NodeMCU).

Vient ensuite la section Wi-Fi pour indiquer le réseau auquel se connecter et les options de secours en cas de problèmes de réseau. Notez que l'ordre des sections n'a aucune importance.

La spécification de l'option `logger:` active la sortie d'état sur le port série. L'option `api:` permet une intégration facile avec le logiciel de contrôle domotique gratuit et *open source* Home Assistant (voir [1]). L'option `ota:` permet de programmer le dispositif sans fil (à partir de Home Assistant, par ex.), c'est très pratique, car il n'est plus nécessaire d'avoir une connexion physique avec le dispositif.

Vient ensuite l'essentiel de l'application, à commencer par la spécification que le port GPIO16 doit être une sortie. Il le faut si vous voulez utiliser la LED qui y est connectée comme un dispositif *light* (éclairage), ce qui est intéressant – car les éclairages ont différentes options comme des *switches* (interrupteurs) (**fig. 3**).

Éclairages

Comme *light*, j'ai défini la LED bleue sur le port GPIO16 et la chaîne de LED. Cette dernière est gérée par la plateforme *neopixelbus*, où l'on doit spécifier quelques options comme la longueur de la chaîne et le port auquel elle est connectée. Dans ce cas, le port est GPIO2, ce qui permet d'utiliser la méthode *ESP8266_UART1*. En fait, lorsque vous spécifiez cette méthode, vous n'avez pas besoin de spécifier GPIO2, car cela est implicite.

Les éclairages peuvent avoir des effets, et ESPHome en a intégré quelques-uns que vous pouvez utiliser (si votre éclairage les supporte, bien sûr). Il va sans dire que vous pouvez également programmer vos propres effets lumineux. J'ai spécifié la plupart des effets intégrés pour l'anneau de LED. Les effets peuvent avoir des paramètres, mais sans eux, les valeurs par défaut seront utilisées. Home Assistant vous permet de choisir quel effet est actif (**fig. 4**). J'aime l'effet de scintillement aléatoire.

Autres entrées et sorties

Les ports GPIO inutilisés sont déclarés comme des interrupteurs afin que vous puissiez les activer et les désactiver depuis Home Assistant. En domotique, un interrupteur est un dispositif qui est contrôlé par le

système (par ex. un relais). Un interrupteur commandé par l'utilisateur (ou l'occupant) est un capteur binaire. Un bouton-poussoir est connecté à GPIO0 et il est donc spécifié comme un capteur binaire.
Enfin, la carte possède une entrée analogique sur la broche A0 avec un diviseur de tension devant elle (R15 & R16), ce qui explique le facteur de multiplication de 3,2 pour reconvertir la tension d'entrée divisée en volts. La tension d'entrée maximale est de 3,3 V.

Construire une alarme de distanciation sociale

ESPHome permet l'automatisation et vous pouvez ajouter un capteur de proximité à la carte pour changer la couleur de l'anneau de LED en fonction de ce que le capteur voit. De cette façon, la carte pourrait servir d'alarme de distanciation sociale. On peut en faire un simple bijou électronique décoratif ou une broche ; laissez libre cours à votre imagination.
Une possibilité intéressante ici est d'utiliser Home Assistant. Avec quelques-unes de ces cartes ESP8266 prêtes-à-porter intégrées à Home Assistant, il est possible de créer des effets lumineux fantaisistes (par ex. pour Noël). Mais, même si Home Assistant est optimisé pour la domotique, il peut aussi faire d'autres choses, comme animer un jeu à la fête d'anniversaire de votre enfant. Épinglez une carte sur chaque enfant et utilisez Home Assistant pour créer et commander des équipes ou décider quel candidat peut répondre à une question ou qui est qui lorsqu'on joue à chat perché. Je suis sûr qu'avec un peu de créativité, il est possible d'inventer de nombreuses applications amusantes.
Les fichiers de conception peuvent être téléchargés en [2].

(210146-04)

Produits

- **ESP-12F, module Wi-Fi basé sur l'ESP8266**
  www.elektor.fr/esp-12f-esp8266-based-wi-fi-module-160100-92
- **Carte microcontrôleur NodeMCU ESP8266**
  www.elektor.fr/nodemcu-microcontroller-board-with-esp8266-and-lua
- **H. Henrik Skovgaard,** *IoT Home Hacks with ESP8266*, **Elektor, 2020**
  www.elektor.fr/iot-home-hacks-with-esp8266

Des questions, des commentaires ?
Envoyez un courriel à l'auteur (clemens.valens@elektor.com) ou contactez Elektor (redaction@elektor.fr).

Contributeurs
Idée, conception, texte et photographies : **Clemens Valens**
Schéma : **Patrick Wielders**
Rédaction : **Jens Nickel, C. J. Abate**
Mise en page : **Giel Dols**
Traduction : **Denis Lafourcade**

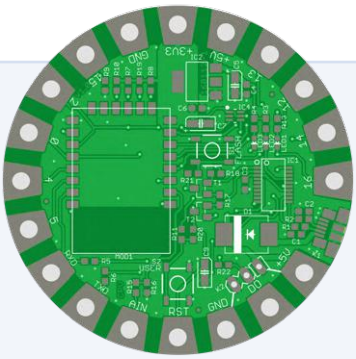
LISTE DES COMPOSANTS

Résistances
Toutes 5%, 50 V, 0,1 W, 0603
R20 = 0 Ω
R1, R2 = 27 Ω
R11, R13, R21, R22 = 470 Ω
R3, R4, R5, R6 = 1 kΩ
R7, R8, R9, R10, R17, R18, R19 = 10 kΩ
R16 = 100 kΩ
R15 = 220 kΩ

Condensateurs
C1, C2 = 47 pF, 0603
C3, C4 = 100 nF, 0603
C6 = 1 µF, 0603
C5, C7, C9 = 10 µF, 16 V, boîtier A

Semi-conducteurs
D1 = MBRS540
IC4 = 74LVC1T45GW
IC1 = FT231XS
IC2 = LD1117AS33
LED1 = LED, bleue, 0603
LED2 = LED, jaune, 0603
LED3 = LED, rouge, 0603
T1, T2 = BC847C

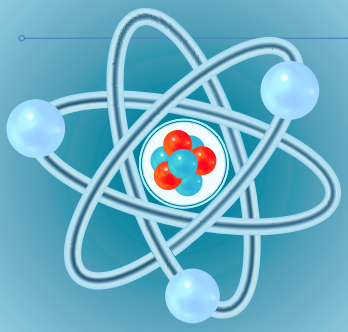
Divers
K1 = connecteur micro-USB de type B, montage en surface
K2 = connecteur à 3 broches, pas de 2,54 mm
S1, S2 = interrupteur tactile, 5,1 × 5,1 mm
MOD1 = ESP-12F
Circuit imprimé réf. 160112-1





LIENS

[1] **C. Valens, « la domotique, c'est facile avec... », Elektor 09-10/2020** : www.elektormagazine.fr/200019-04
[2] **Gadget Wi-Fi vestimentaire, Elektor Labs** : www.elektormagazine.fr/labs/4382

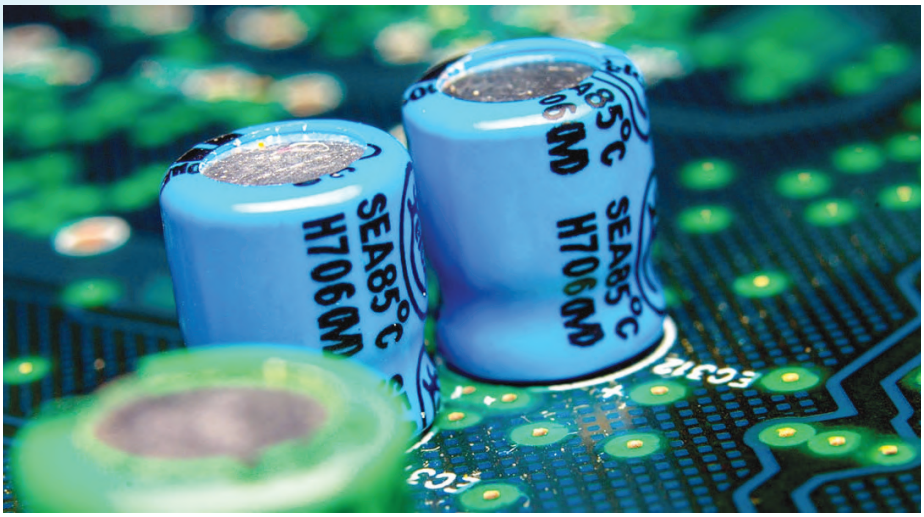


démarer en électronique… (8)

…est moins difficile qu'on ne l'imagine ! Les condensateurs : suite et fin

Eric Bogers (Elektor)

Dans l'article précédent [1], nous avons vu comment les condensateurs bloquent le courant continu, mais laissent passer le courant alternatif. Mais ce n'est pas tout : la tension et le courant sont également déphasés l'un par rapport à l'autre, ce qui peut rendre les calculs un peu plus difficiles. Mais il n'y a pas là de quoi s'effrayer !



Angle de phase

La phase ? Qu'est-ce que c'est ? Pour comprendre, revenons brièvement à notre humble résistance. Lorsque nous appliquons une tension à ses bornes, un courant circule immédiatement. À tout instant, le courant est proportionnel à la tension : la tension et le courant sont *en phase*.

Il en va autrement avec un condensateur, ce que nous pouvons expliquer comme suit. À l'instant précis où nous connectons une source aux bornes d'un condensateur déchargé, celui-ci se comporte comme un court-circuit : le courant de charge maximal circule, mais la tension à ses bornes est nulle. Au fur et à mesure que le condensateur se charge, la tension à ses bornes augmente tandis que le courant diminue. Lorsque le condensateur est complètement chargé, la tension à ses bornes a atteint sa valeur maximale tandis que le courant est tombé à zéro. Ainsi, on peut dire : il y a d'abord le courant, puis vient la tension. En jargon technique : le courant *précède* la tension. Lorsqu'une tension sinusoïdale pure est appliquée aux bornes d'un condensateur, il existe une *différence de phase* d'exactement 90° entre la tension et le courant.

Lorsque nous appliquons une tension alternative aux bornes d'un condensateur, un courant alternatif circule dans ce condensateur – c'est ce que nous avons vu précédemment. Et cela signifie que nous pouvons calculer une résistance en courant alternatif pour ce condensateur. On l'appelle *impédance*, et on la désigne par le symbole *X*. L'unité d'impédance est à nouveau l'ohm. La formule suivante s'applique :

$$X_C = \frac{1}{2 \cdot \pi \cdot f \cdot C}$$

Plus la capacité du condensateur est grande, plus il peut stocker de charge et plus l'intensité du courant qui circule pendant sa charge est élevée – et plus son impédance est faible. Il en est de même pour la fréquence du courant : plus elle est élevée, plus les cycles de charge et de décharge sont fréquents par unité de temps et plus le courant qui circule est élevé et donc plus l'impédance est faible.

Ce qu'il faut retenir, c'est que l'impédance d'un condensateur n'est pas une valeur fixe, mais dépend de la fréquence.

Vous pouvez, bien sûr, hausser les épaules en vous disant : « D'accord, c'est bien joli, cette histoire de différence de phase, mais il n'y a pas de quoi en perdre le sommeil ». Si seulement c'était aussi simple… Dans les applications pratiques de l'électronique, les résistances et les condensateurs sont rarement utilisés seuls ; le plus souvent, ils se retrouvent en série ou en parallèle avec d'autres composants. Et c'est à cause de ce déphasage que, par exemple, dans un circuit en série composé d'un condensateur et d'une résistance, on ne peut pas simplement additionner l'impédance et la résistance ; celles-ci doivent être additionnées *de manière vectorielle*.

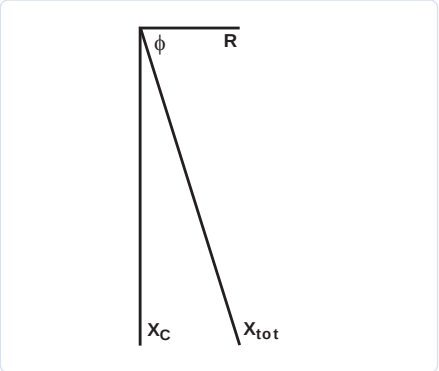


Figure 1. Circuit d'une résistance et d'un condensateur en série.

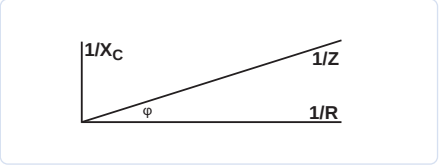


Figure 2. Circuit d'une résistance et d'un condensateur en parallèle.

Nous allons le démontrer avec l'exemple d'un circuit en série composé d'un condensateur d'une valeur de 0,1 µF et d'une résistance d'une valeur de 10 kΩ. Nous voulons connaître l'impédance totale à une fréquence de 50 Hz. Tout d'abord, nous calculons l'impédance du condensateur :

$$X_C = \frac{1}{2 \cdot \pi \cdot f \cdot C} = \frac{1}{2 \cdot \pi \cdot 50\,Hz \cdot 0{,}1\,\mu F} = 31{,}830\,k\Omega$$

Pour l'impédance totale du circuit en série, la règle suivante s'applique (souvenez-vous du théorème de Pythagore) :

$$X_{tot} = \sqrt{X^2{}_C + R^2} = \sqrt{(31{,}830\,k\Omega)^2 + (10\,k\Omega)^2} = 33{,}364\,k\Omega$$

L'impédance totale qui est calculée de cette façon est souvent désignée par la lettre *Z* (au lieu de X_{tot}).

L'impédance *Z* a également un certain angle de phase (qui est logiquement quelque part entre le −90° du condensateur et le 0° de la résistance), donné par la formule :

$$\tan\varphi = \frac{X_C}{R}$$

L'angle de phase est alors (désolé, vous aurez probablement besoin d'une calculatrice de poche « scientifique » pour cela) :

$$\varphi = \arctan\frac{X_C}{R} = \arctan\frac{-31{,}830\,k\Omega}{10\,k\Omega} = -72{,}56^\circ$$

Dans un circuit en série, le même courant circule dans tous les composants ; comme dans un condensateur, la tension est en retard sur le courant, l'angle de phase est négatif. On peut aussi résoudre ce problème graphiquement (**fig. 1**). Le sens de rotation (au sens mathématique) étant le sens inverse des aiguilles d'une montre, l'angle de phase négatif doit être dessiné vers le bas (si vecteur horizontal pour la résistance).

Naturellement, une résistance et un condensateur peuvent également être connectés en parallèle ; l'impédance totale *Z* est alors (nous utilisons les mêmes composants et la même fréquence que pour le circuit en série) :

$$Z = \frac{1}{\sqrt{\frac{1}{X^2{}_C} + \frac{1}{R^2}}} = \frac{1}{\sqrt{\frac{1}{(31{,}830\,k\Omega)^2} + \frac{1}{(10\,k\Omega)^2}}} = 9{,}54\,k\Omega$$

Pour l'angle de phase, nous avons alors :

$$\varphi = \arctan\frac{R}{X_C} = \arctan\frac{10\,k\Omega}{31{,}830\,k\Omega} = 17{,}44^\circ$$

Ici aussi, une solution graphique est possible (**fig. 2**). Dans un circuit parallèle, la même tension est appliquée à tous les composants concernés, mais le courant qui traverse le condensateur précède la tension : l'angle de phase est positif et dessiné vers le haut.

Une dernière remarque : la différence entre les angles de phase dans un circuit série et un circuit parallèle (avec les mêmes composants) est exactement de 90°.

Filtres passe-haut et passe-bas

Dans les numéros d'*Elektor* de septembre/octobre 2020, novembre/décembre 2020 et janvier/février 2021, nous avons traité en détail la conception des circuits de filtrage. Mais ces trois articles allaient peut-être un peu au-delà du niveau de l'électronicien débutant. Ces circuits étant néanmoins d'importance essentielle en électronique, nous allons les aborder brièvement ici, sans trop de théorie.

La formule de l'impédance d'un condensateur montre qu'elle est infinie à une fréquence de 0 Hz (c'est-à-dire pour une tension continue) et diminue à mesure que la fréquence augmente. C'est pourquoi un condensateur est utilisé pour éliminer (bloquer) la partie continue d'un « mélange » de tensions continues et alternatives.

Nous avons besoin d'un tel circuit, par exemple, à l'étage d'entrée d'un amplificateur de microphone : la tension alternative (l'équivalent électrique du signal acoustique) produite par le microphone est superposée à la tension continue de son alimentation (on parle d'une alimentation fantôme ; elle a l'avantage de ne nécessiter aucun fil supplémentaire pour l'alimentation). Nous ne voulons évidemment pas amplifier la tension d'alimentation, mais uniquement la tension du signal.

Pour éliminer cette tension continue, nous utilisons un filtre passe-haut (**fig. 3**). Il est appelé ainsi, car il laisse passer les hautes fréquences et bloque les basses fréquences.

La **figure 4** montre les réponses en fréquence et en phase d'un filtre passe-haut ; la précision du système de mesure utilisé pour réaliser ces graphiques laisse malheureusement à désirer aux extrémités de la gamme de fréquences. Néanmoins, nous pouvons clairement distinguer trois régions :

- La région de la bande passante commence (sur ce graphique) à environ 1 kHz. Dans cette région, le signal passe pratiquement sans entrave et il n'y a pratiquement pas de déphasage.
- La zone d'arrêt ou de blocage se situe en dessous de 100 Hz environ. Ici, le signal diminue avec la fréquence – pour être exact de 6 dB par octave ou 20 dB par décade ; la phase s'approche asymptotiquement de 90°.
- Entre ces deux régions se trouve la fréquence de coupure. Elle est définie comme la fréquence à laquelle le niveau du signal est réduit de 3 dB par rapport à la bande passante. À la coupure, l'impédance capacitive et la résistance sont égales en valeur absolue et nous pouvons écrire :

$$f_{fréquence\,de\,coupure} = \frac{1}{2 \cdot \pi \cdot R \cdot C}$$

Le comportement d'un filtre simple comme celui-ci ne dépend que de sa fréquence de coupure ; pour une réponse en fréquence plus complexe, on utilise des filtres à étages multiples (« filtres d'ordre supérieur »). Cette question a été traitée en détail dans la série d'articles déjà mentionnée d'*Elektor* [2].

Là où il existe des filtres passe-haut, il existe aussi des filtres passe-bas. Ils se ressemblent, la résistance et le condensateur ont seulement été permutés (**fig. 5**). Plus la fréquence est élevée, plus le signal est court-circuité à la masse par le condensateur. Ce type de filtre est utilisé (entre autres) pour éliminer le bruit à haute fréquence d'un signal.

La **figure 6** montre les réponses en fréquence et en phase d'un filtre passe-bas (dans ce cas précis, les valeurs des composants utilisés sont différentes de celles de l'exemple de la figure 4). Ici aussi, nous pouvons distinguer trois régions : la région de la bande passante en dessous de la fréquence de coupure, la région de la bande d'arrêt ou de blocage au-dessus de la fréquence de coupure et la fréquence de coupure elle-même, où l'impédance capacitive et la résistance sont égales en valeur absolue.

Nous concluons ainsi notre description des condensateurs. La prochaine fois, nous nous intéresserons aux inductances – des composants qui sont (injustement) détestés par de nombreux amateurs d'électronique...

(210183-04)

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

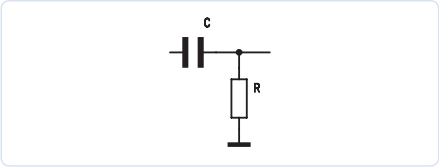


Figure 3. Filtre passe-haut.

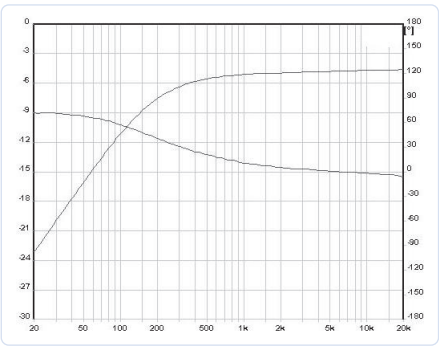


Figure 4. Réponses en amplitude et en phase d'un filtre passe-haut.



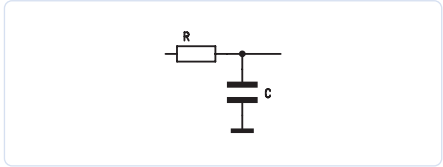


Figure 5. Filtre passe-bas.

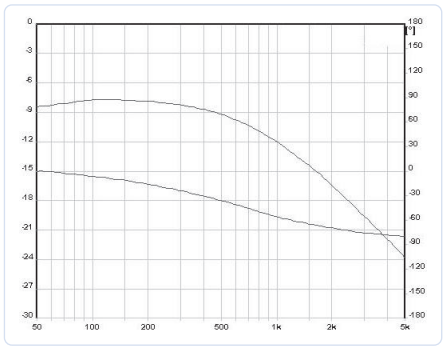


Figure 6. Réponses en amplitude et en phase d'un filtre passe-bas.

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

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

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



LIENS

[1] **E. Bogers, « démarrer en électronique (3) », Elektor, 05-06/2020** : www.elektormagazine.fr/200106-03

[2] **A. Rosenkränzer, « conception de filtres analogiques », Elektor, trois articles à partir de 09-10/2020** : www.elektormagazine.fr/200318-03

Petits circuits avec l'écosystème Qwiic

Marcus Stevenson (États-Unis)

Le système Qwiic est une innovation de SparkFun où le protocole I²C et des connecteurs JST à 4 broches avec détrompeur permettent de créer des circuits modulaires à partir de plus de 100 cartes et microcontrôleurs compatibles. Vous n'êtes pas sûr de bien comprendre ? Ne vous inquiétez pas ! Ce qu'il faut savoir, c'est que Qwiic vous dispense de souder des fils de connexion et de déchiffrer des plans de câblage compliqués. Même un novice peut commencer à expérimenter immédiatement avec un Arduino et n'importe quel capteur. Vous ne me croyez pas ? Voici la preuve administrée par des petits circuits qui utilisent le *Qwiic Pro Micro* et une poignée d'autres cartes d'extension

Pour tirer profit de la lecture de cet article, il est souhaitable d'avoir déjà configuré l'EDI Arduino et installé les bibliothèques nécessaires. Vous obtiendrez de l'aide et des instructions de configuration pour chaque produit sur le site SparkFun.com.

Codeur rotatif à changement de couleur

ⓘ Circuit : **Fig. 1** ; Code du programme : **Lien [1]**.

Pour cet exemple, vous aurez besoin du *Qwiic Pro Micro*, du *Qwiic Twist RGB Rotary Encoder* et du *Qwiic Micro OLED Breakout*. Vous verrez dans les exemples qui suivent que le câblage est pratiquement le même pour tous les circuits.

Il suffit d'utiliser des câbles Qwiic pour connecter le Pro Micro au codeur rotatif et à l'afficheur OLED. L'ordre dans lequel vous les mettez n'a pas d'importance, tant qu'ils sont tous interconnectés. Il vous suffit ensuite de brancher le Pro Micro à votre ordinateur et de télécharger le croquis *QwiicTwistMicroOLEDDisplay.ino*. Veillez à sélectionner « ATMEGA 32U4 (5 V, 16 MHz) » sous l'option « processeur », sinon vous aurez une erreur au téléchargement ! Si tout se passe bien, vous devriez voir la flamme du logo de SparkFun

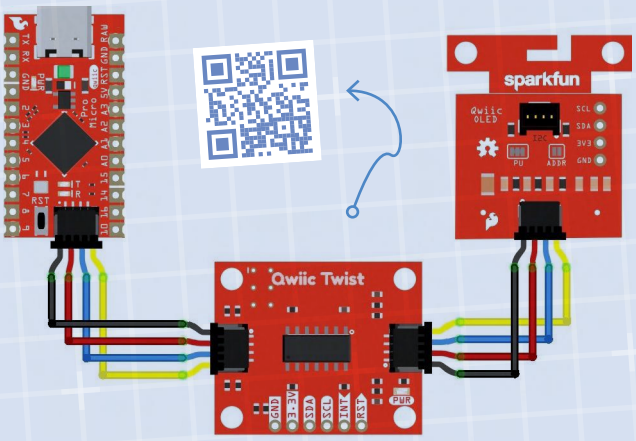


Figure 1. Connexion de la carte pour « Codeur rotatif à changement de couleur ».

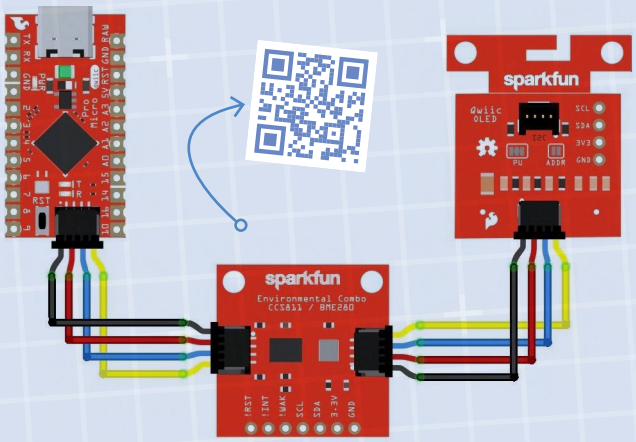


Figure 2. Connexion de la carte pour « Thermomètre environnemental Qwiic ».

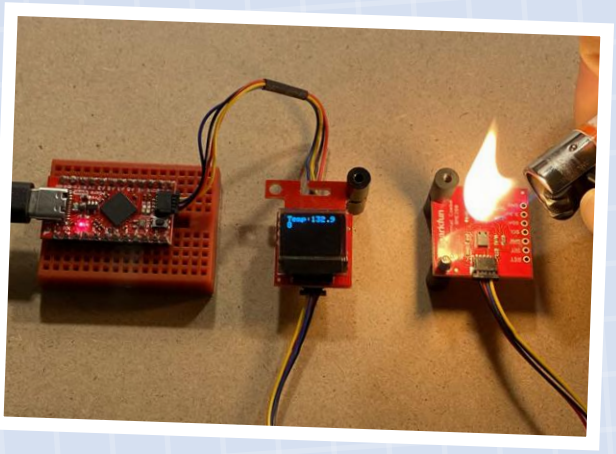


Figure 3. « N'essayez pas cela à la maison » …

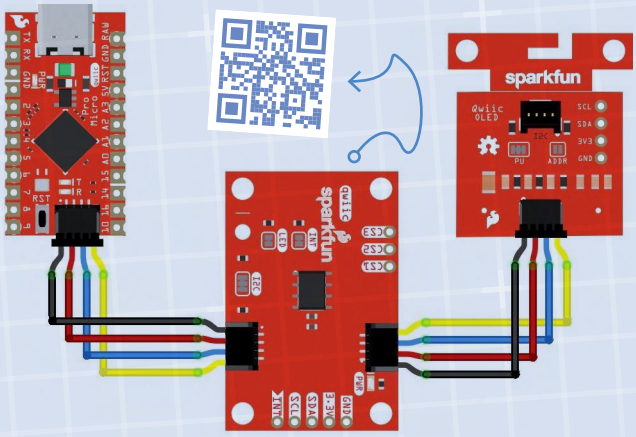


Figure 4. Connexion de la carte pour « Saisie tactile capacitive pour tous ».

apparaître brièvement sur l'écran OLED, puis l'écran affichera le nom de la couleur sur laquelle le codeur rotatif est actuellement réglé. Lorsque vous tournez le bouton, la couleur du bouton devrait changer, ainsi que le nom affiché sur l'OLED. De plus, lorsque vous appuyez sur le bouton comme sur une touche, vous verrez s'afficher le mot « Pressed ! ».

Ce croquis utilise la fonction `getCount()` pour déterminer le nombre actuel d'impulsions du codeur et utilise la fonction `setColor()` pour changer la couleur de sa LED RVB en fonction du nombre d'impulsions, en alternant entre le rouge, le vert, le violet, le jaune, le rose, le bleu et l'orange. Il utilise également les méthodes `clear()`, `setCursor()`, `print()` et `display()` de la bibliothèque OLED pour afficher les noms des couleurs.

Thermomètre environnemental Qwiic

Circuit : **Fig. 2** ; Code du programme : **Lien [2]**.

Cet exemple utilise également le *Qwiic Micro OLED Display* et le Pro Micro, mais cette fois-ci, nous allons remplacer le codeur rotatif par la carte de liaison *Qwiic Environmental Combo*. Ce capteur permet à l'utilisateur de collecter toutes sortes de données environnementales, notamment la pression barométrique, l'humidité, la température, les COVT et les niveaux de CO_2 équivalents (eCO_2). Mais cet exemple ne sera pas très original. Tout ce que nous allons faire, c'est imprimer la température actuelle sur l'afficheur micro-OLED, en degrés Fahrenheit. Je vous laisse le soin de les convertir en degrés Celsius. En supposant que vous ayez construit le circuit précédent, débranchez le codeur rotatif et connectez l'*Environmental Combo Breakout* avec le câble Qwiic.

Vous pouvez maintenant télécharger le croquis *QwiicEnvironmentalComboMicroOLEDDisplay.ino* sur le Pro Micro. Une fois fait, vous devriez voir brièvement le logo, puis l'affichage devrait indiquer « Temp: » suivi de la température en degrés Fahrenheit. Si vous soufflez sur le capteur, ou si vous le chauffez *légèrement* (**fig. 3**), vous devriez voir la température fluctuer. Ce schéma utilise la fonction `readTempF()` pour afficher la température du circuit intégré BME280. Pour plus d'originalité, vous pouvez utiliser la méthode `readTempC()` pour des lectures cette fois en degrés Celsius.

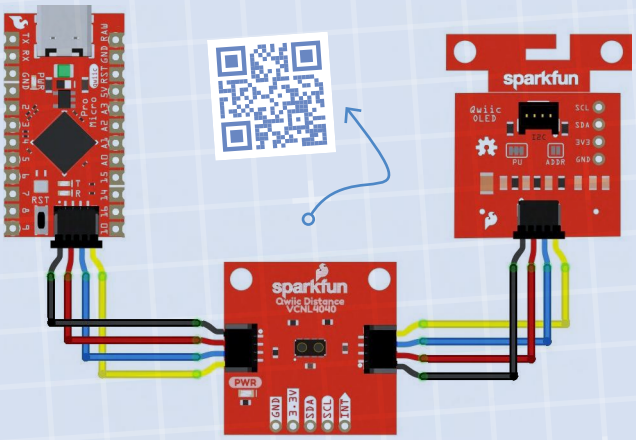


Figure 5. Connexion de la carte pour « Détecteur de présence Qwiic ».

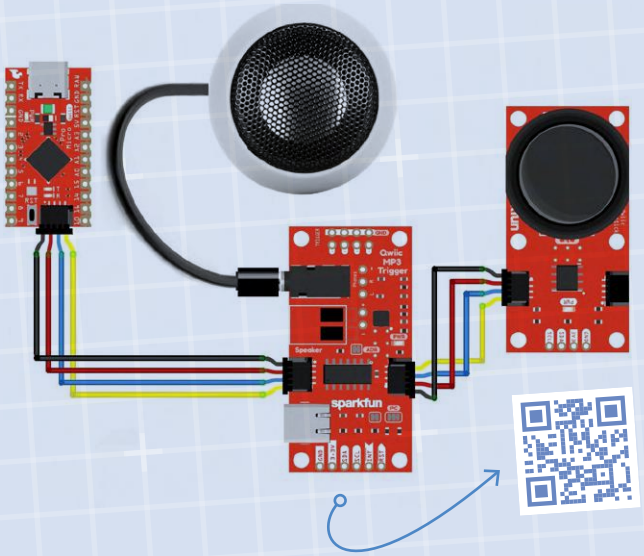


Figure 6. Connexion de la carte pour « Entrée tactile, sortie audible ».

Saisie tactile capacitive pour tous

ⓘ Circuit : **Fig. 4** ; Code du programme : **Lien [3]**.

Les boutons, c'est dépassé. Leur nature mécanique les voue à la panne. La saisie tactile capacitive vous permet d'interagir avec vos circuits *sans* pièces mobiles. Dans ce croquis, nous reprenons l'afficheur Micro OLED, et allons interagir avec lui en utilisant le *Capacitive Touch Slider Qwiic*. Il suffit de déconnecter l'*Environmental Combo Breakout* et de le remplacer par le *Capacitive Touch Slider*. Cette petite carte permet d'utiliser les trois touches comme entrées individuelles, ou de détecter les gestes de glissement sur celles-ci. On peut aussi utiliser les broches d'extension et souder ses propres touches pour des applications personnalisées.

Une fois tout connecté, téléchargez le croquis *CapSliderMicroOledExample.ino* sur le Pro Micro. Au début, vous verrez le logo Flame, puis l'afficheur devrait être vide. Lorsque vous touchez une des trois touches de la carte, un rectangle vertical correspondant devrait s'afficher sur l'afficheur OLED. Chez SparkFun, nous utilisons des pavés tactiles capacitifs sur tous nos bancs de test de production car, contrairement aux interrupteurs tactiles, ils ne s'usent pas et n'ont jamais besoin d'être remplacés, mais il faut renoncer à l'agréable « clic » des vrais interrupteurs tactiles…

Détecteur de présence Qwiic

ⓘ Circuit : **Fig. 5** ; Code du programme : **Lien [4]**.

Vous voulez savoir si un objet est présent, alors qu'il ne l'était pas auparavant ou qu'il s'est déplacé au-delà d'une certaine limite ? Ce détecteur fait les deux. Nous conserverons l'afficheur Micro OLED de l'exemple précédent, mais remplacerons le *Capacitive Touch Slider* par le *SparkFun Proximity Sensor Breakout*. Ce petit capteur ultra simple et tout aussi utile combine un émetteur infrarouge, un capteur de lumière ambiante et un capteur de proximité pour la détection d'objets à courte distance.

Lorsque tout est branché, téléchargez le croquis *ProximityMicroOLEDexample.ino* sur le Pro Micro. Là encore, vous devriez voir brièvement le logo Flame. Dans ces exemples, si l'afficheur OLED se fige sur le logo Flame, essayez d'appuyer sur le bouton de réinitialisation du Pro Micro. Si le problème persiste, il est probablement dû aux câbles ou aux connecteurs Qwiic. Essayez alors de débrancher et de reconnecter votre circuit. Si tout fonctionne, l'écran devrait s'éteindre, puis se rallumer lorsque vous passez votre main ou un objet dans un rayon d'environ 20 cm autour du détecteur de proximité. Ces capteurs sont courants dans toutes sortes d'appareils sans contact comme les distributeurs automatiques de serviettes en papier.

Entrée tactile, sortie audible

ⓘ Circuit : **Fig. 6** ; Code du programme : **Lien [5]**.

Qwiic ne se limite pas à fournir des données provenant de codeurs et de capteurs. La modularité du système permet de combiner toutes les fonctions dont votre projet a besoin par de simples interconnexions. Dans cet exemple, nous nous passerons d'OLED et de capteurs. Avec le *Qwiic MP3 Trigger* et le *Qwiic Joystick*, ce croquis montre comment utiliser Qwiic pour créer facilement des objets interactifs.

Comme dans les exemples précédents, il suffit de connecter vos cartes *MP3 Trigger* et *Joystick* au Pro Micro via une paire de câbles Qwiic. Il faudra également un haut-parleur ou un casque pour l'audio, donc branchez une sortie audio dans la prise casque du

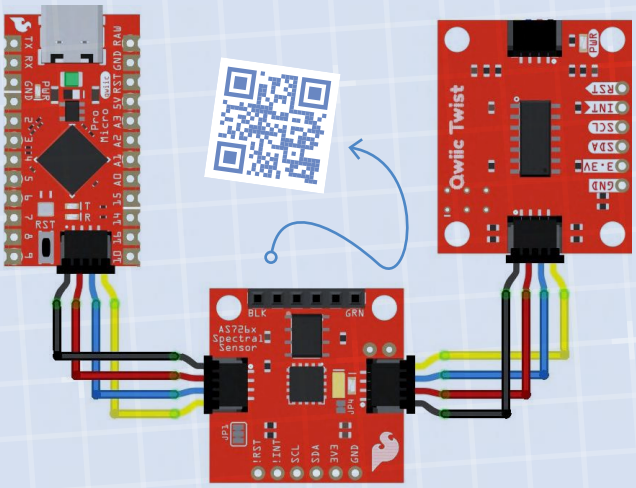


Figure 7. Connexion de la carte pour « Rouge, vert ou bleu ? ».

Accessoires

La plupart des accessoires mentionnés dans cet article sont disponibles chez Elektor et SparkFun !

- **SparkFun Qwiic Pro Micro - USB-C (ATmega32U4)**
  www.elektormagazine.fr/esfe-en-smallcircuits1
- **SparkFun Qwiic Twist - RGB Rotary Encoder Breakout**
  www.elektormagazine.fr/esfe-en-smallcircuits2
- **SparkFun Micro OLED Breakout (Qwiic)**
  www.elektormagazine.fr/esfe-en-smallcircuits3

MP3 Trigger. Vous devrez également copier quatre fichiers MP3 nommés *F001.mp3*, *F002.mp3*, *F003.mp3* et *F004.mp3* dans le dossier racine d'une carte micro SD, et insérer cette carte SD dans la fente SD du MP3 Trigger. Vous serez alors prêt à télécharger le croquis *joystickmp3example.ino* sur le Pro Micro.
Si tout est bien connecté, vous devriez entendre les fichiers MP3 de la carte microSD dans votre écouteur/haut-parleur chaque fois que vous appuyez sur la manette vers le haut, le bas, la gauche ou la droite. Si vous téléchargez les pistes audio du dépôt GitHub *Simple Sketches* de SparkFun, l'audio prononcera « *up, down, left, right* », selon la direction dans laquelle vous orienterez le manche.

Rouge, vert ou bleu ?

Circuit : **Fig. 7** ; Code du programme : **Lien [6]**.

Compte tenu du nombre de combinaisons possibles de cartes Qwiic, il serait amusant d'en saisir une paire au hasard et d'essayer d'en faire un gadget. Reprenons le codeur *Qwiic Twist RGB Encoder* et introduisons la carte de liaison *Qwiic Spectral Sensor*. Ce capteur existe en plusieurs variantes, mais pour ce croquis, nous utiliserons la version pour le spectre visible. Le capteur spectral vous permet de détecter les couleurs visibles, et dans cet exemple, nous l'utiliserons pour détecter si une bille transparente est rouge, verte ou bleue. Comme dans tous les exemples précédents, il suffit de relier le *Qwiic Twist RGB Rotary Encoder* et le *Qwiic Spectral Sensor Breakout* par des câbles Qwiic, et de les connecter au Pro Micro. Trouvez aussi quelques objets rouges, verts et bleus à tester. J'ai utilisé des billes, mais à peu près n'importe quel autre objet de couleur que vous pouvez placer sur le capteur fonctionnera.
Téléchargez *VisSpectrumTwistColorExample.ino* sur le Pro Micro. Si tout est correctement configuré, vous devriez pouvoir tester différents objets pour voir s'ils sont rouges, verts ou bleus en tenant l'objet au-dessus du capteur spectral et en appuyant sur le bouton du Qwiic Twist. La LED RVB du codeur rotatif devrait changer en fonction de la couleur détectée par le capteur. Si vous testez un objet qui n'est pas de l'une des trois couleurs, le résultat sera la longueur d'onde (R, V ou B) qui renvoie la plus grande valeur.

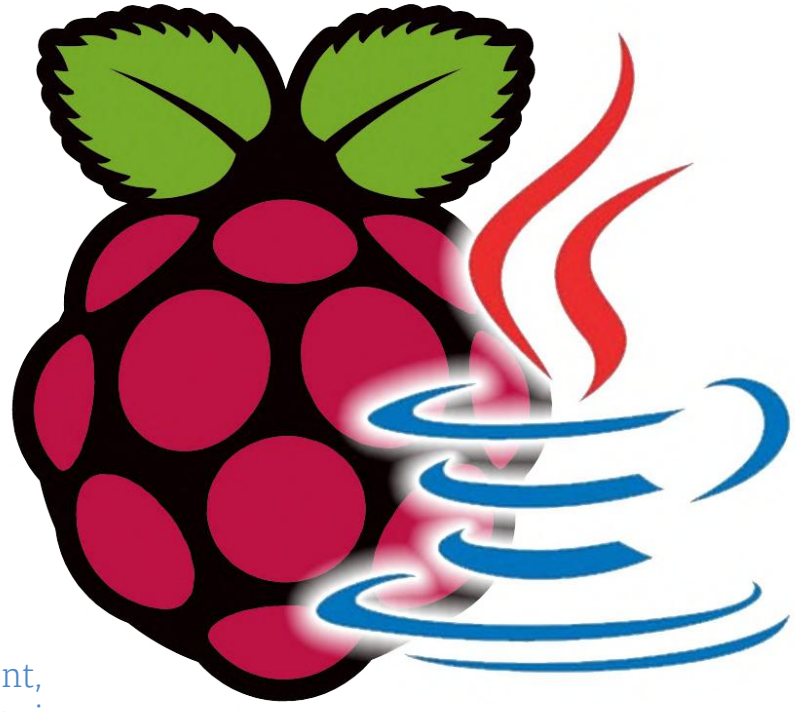
(200696-04)

LIENS

[1] **Code du programme « Codeur rotatif à changement de couleur »** : https://bit.ly/2XHeLfv
[2] **Code du programme « Thermomètre environnemental Qwiic »** : https://bit.ly/35BP5VJ
[3] **Code du programme « Saisie tactile capacitive pour tous »** : https://bit.ly/2XB5FAO
[4] **Code du programme « Détecteur de présence Qwiic »** : https://bit.ly/3qjqjBN
[5] **Code du programme « Entrée tactile, sortie audible »** : https://bit.ly/3idt4BU
[6] **Code du programme « Rouge, vert ou bleu ? »** : https://bit.ly/3ibXnJa

Java sur Raspberry Pi

Partie 2 : commande des broches GPIO avec un service REST de Spring



Frank Delporte (Belgique)

Java est-il un langage de programmation adapté au Raspberry Pi ? Dans l'article précédent, nous avons répondu à cette question par un « oui » retentissant ! Avec quelques applications de démonstration Java à fichier unique à notre actif, il est maintenant temps d'approfondir. En développant ce que nous avons couvert, nous irons maintenant plus loin et construirons une application complète avec plusieurs classes. Celles-ci vont nous fournir des services web REST pour commander les broches GPIO d'un Raspberry Pi.

La version « Raspberry Pi OS Full (32 bits) » du système d'exploitation comprend déjà la version 11 du kit de développement Java (JDK). Cependant, pour développer des applications Java, un EDI nous aidera à écrire des applications faciles à maintenir. Comme indiqué dans l'article précédent [1], Visual Studio Code peut être utilisé sur le RPi. L'autre solution est de programmer l'application dans Visual Studio Code sur un PC, puis de la compiler et l'exécuter sur le RPi. Pour cet article, nous écrirons le code directement sur le RPi. Pour cela, nous avons besoin de quelques outils supplémentaires, alors commençons par les installer.

Maven

Nous nous servirons de Maven pour construire l'application sur notre RPi. Maven compile le code, ainsi que les dépendances requises, dans un seul fichier JAR. Ceci est possible grâce au fichier de configuration *pom.xml* qui se trouve à la racine du projet.
Une seule commande suffit pour installer Maven, après quoi nous pouvons immédiatement vérifier l'installation en demandant la version comme suit :

```
$ sudo apt install maven
$ mvn -v
Apache Maven 3.6.0
Maven home: /usr/share/maven
```

Pi4J

Pour commander les broches GPIO, nous aurons recours à la bibliothèque Pi4J qui établit un pont entre notre code Java et les broches d'entrée/sortie à usage général (GPIO) du RPi. Cela nous permet de commander les broches GPIO et de nous connecter à divers composants électroniques.
Pour une prise en charge complète de la bibliothèque Pi4J sur le RPi, nous devons installer quelques logiciels supplémentaires. Une nouvelle fois, une seule commande suffit pour le faire :

```
$ curl -sSL https://pi4j.com/install | sudo bash
```

Mise à jour de WiringPi

Une dernière étape est nécessaire pour être fin prêt. Si vous utilisez un RPi 4, vous devrez mettre à jour WiringPi. Pi4J l'utilise comme bibliothèque native pour commander les broches GPIO. Comme l'architecture du système sur puce (SoC) a changé avec la version 4, il vous faut la nouvelle version de WiringPi. Malheureusement, ce projet est obsolète depuis l'année dernière, mais grâce à la communauté *open source*, une version « non officielle » est disponible et peut être installée par un script fourni par Pi4J. Exécutez la commande :

```
sudo pi4j -wiringpi
```

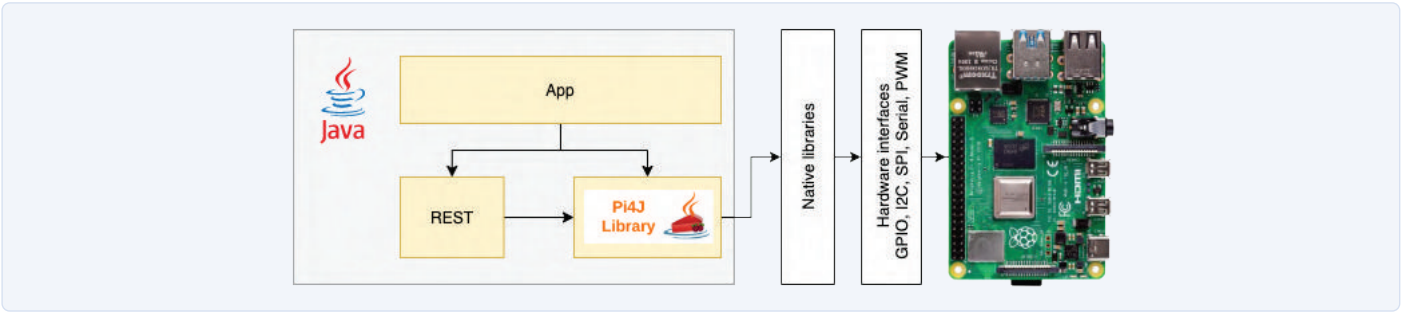


Figure 1. Aperçu de l'application utilisant le service REST de Spring et la bibliothèque Pi4J.

Elle récupère le projet depuis GitHub, le compile et l'installe sur votre RPi d'un seul coup. En demandant la version, vous verrez qu'elle a été mise à niveau de la version par défaut, de 2.50 à 2.60 :

```
$ gpio -v
gpio version: 2.50
$ sudo pi4j -wiringpi
$ pi4j -v
------------------------------------------
              THE Pi4J PROJECT
------------------------------------------
  PI4J.VERSION     : 1.3
  PI4J.TIMESTAMP   : 2021-01-28 04:14:07
------------------------------------------
  WIRINGPI.PATH : /usr/lib/libwiringPi.so /usr/local/lib/
    libwiringPi.so
  WIRINGPI.VERSION : 2.60
------------------------------------------
```

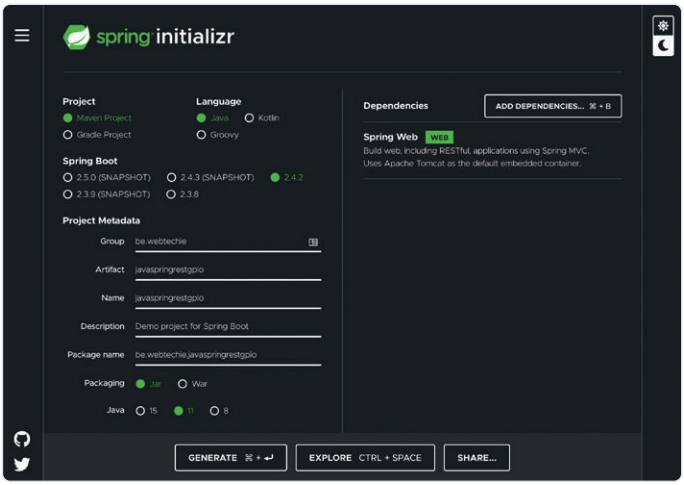


Figure 2. Paramètres de configuration requis pour Spring Initializr.

L'application

Le code complet de cette application est disponible sur le dépôt GitHub qui accompagne cet article [2] dans le dossier `elektor/2106`. Ce projet est une démonstration de faisabilité qui commande les broches GPIO en utilisant un service web REST. Il utilise le canevas Spring, un logiciel qui fournit de nombreux outils pour construire des applications puissantes avec un minimum de code (**fig. 1**). Nous décrivons dans ce qui suit le processus de création des différents fichiers qui composent ce projet. Si cela ne fonctionne pas comme décrit, le code du dépôt devrait fonctionner tel quel.
Depuis GitHub, on récupère le projet complet ainsi :

```
pi@raspberrypi:~ $ git clone https://github.com/FDelporte/
    elektor
Cloning into 'elektor'...
remote: Enumerating objects: 34, done.
remote: Counting objects: 100% (34/34), done.
remote: Compressing objects: 100% (24/24), done.
remote: Total 34 (delta 2), reused 34 (delta 2), pack-reused 0
Unpacking objects: 100% (34/34), done.
pi@raspberrypi:~ $ cd elektor/2106
pi@raspberrypi:~/elektor/2106 $ ls -l
total 8
-rw-r--r-- 1 pi pi 1720 Feb 15 14:23 pom.xml
drwxr-xr-x 3 pi pi 4096 Feb 15 14:23 src
```

Qu'est-ce que Spring et ses outils ?

Spring est un canevas qui simplifie et accélère le développement d'applications Java (commerciales). Spring Boot est une couche qui se trouve au-dessus de Spring et qui fournit des paquets « prêts à l'emploi ». Ceux-ci permettent de créer des applications autonomes basées sur Spring que vous pouvez « exécuter simplement ». Ceci est rendu possible grâce à un principe connu sous le nom de « convention plutôt que configuration ». Cela signifie que, par défaut, tout fonctionne selon une convention prédéfinie. Si vous avez besoin de faire quelque chose de différent, vous pouvez alors modifier la configuration. Les possibilités les plus importantes, telles que répertoriées sur le site Web de Spring Boot [3], sont les suivantes :

- Création facile d'applications Spring autonomes.
- Intégration d'un serveur web dans votre application (Tomcat, Jetty ou Undertow).
- Fourniture de dépendances de type « *starter* » pour simplifier votre configuration de *build*.
- Configuration automatique de Spring et des bibliothèques tierces lorsque cela est possible.
- Fourniture de fonctions prêtes pour la production, telles que des indicateurs des bilans de santé et la configuration externalisée.
- Absolument aucun besoin de génération de code ou de configuration XML.

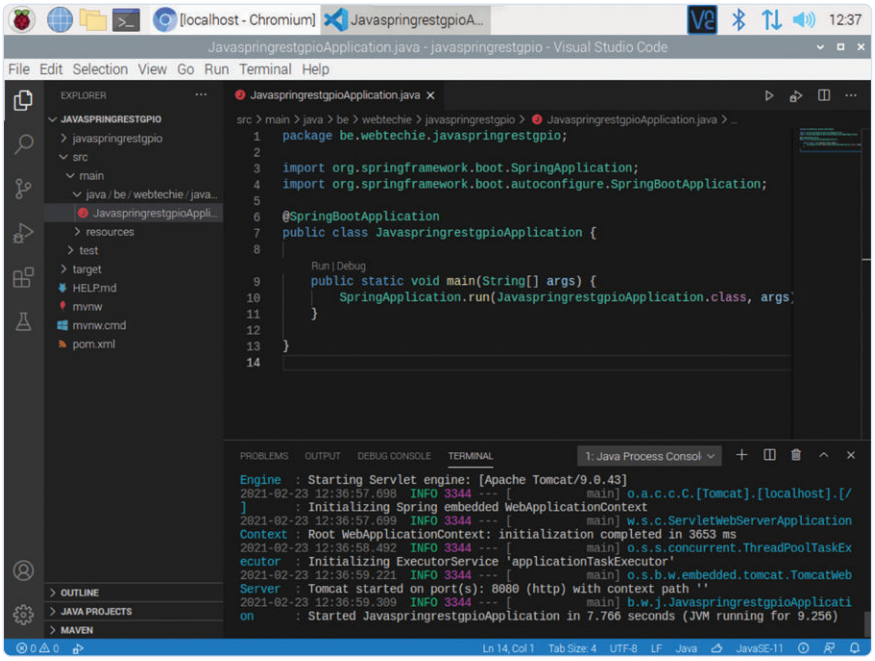


Figure 3. Exécution du projet dans Visual Studio Code.

Enfin, il existe Spring Initializr [4], un outil en ligne permettant de créer rapidement une application de démarrage comprenant tous les paquets Spring Boot requis.

Pour commencer

Pour créer ce projet à partir de zéro, nous commençons par utiliser Spring Initializr, un formulaire de configuration en ligne, qui créera un package de démarrage pour nous. Suivez le lien [4] et recopiez les options dans le formulaire en utilisant les informations de la capture d'écran de la **figure 2**. Sur le côté droit, assurez-vous également de cliquer sur le bouton « *Dependencies* ». Dans la fenêtre qui s'ouvre, nous devons ajouter la dépendance « Spring Web ». On peut la trouver rapidement en utilisant la boîte de recherche. En cliquant sur cette entrée, on l'ajoute au projet. Si une dépendance non désirée est ajoutée par erreur, il suffit de cliquer sur le bouton rouge (tout à droite) pour la supprimer. L'ajout de la dépendance « Spring Web » permet de construire une application REST [5], nous pourrons ainsi nous connecter à l'application via des pages Web dans notre navigateur.

En cliquant sur le bouton « *Generate* », on obtient un fichier ZIP contenant un projet Maven prêt à l'emploi. Une fois qu'il a été décompressé dans un répertoire approprié, ouvrez-le dans Visual Studio Code [6]. Cela fonctionne, à ce stade, aussi bien sur le RPi que sur un PC. Utilisez simplement *Fichier -> Ouvrir le dossier...* et sélectionnez le dossier décompressé. Ensuite, trouvez le fichier *JavaSpringRestApplication.java* (sous *src/main/java*). En cliquant avec le bouton droit de la souris sur ce nom de fichier, l'option *Run* est disponible. Sélectionnez *Run* et l'application devrait démarrer (**fig. 3**). Si votre installation Visual Studio Code ne dispose pas déjà du support Java requis, une fenêtre pop-up apparaîtra pour demander son installation.

Dans le panneau Terminal, nous obtenons le résultat du **listage 1** (ici sans horodatage). Que nous apprend cette sortie ?

- La JVM (Java Virtual Machine) a fonctionné pendant 7 s, dont 4,5 s pour lancer l'application.
- Tomcat (le serveur web) est intégré à l'application et fonctionne sur le port 8080.

Listage 1. Sortie d'une application starter.

```
°  .°°  ____ °°°°°°°°°°  _°°°°°°°°°°°  __ _ _
°/\\ / ___'_ __ _ _(_)_ __°  __ _ \ \ \ \
( ( )\___ | '_ | '_| | '_ \/ _` | \ \ \ \
°\\/° ___)| |_)| | | | | || (_| |° ) ) ) )
°  '°  |____| .__|_| |_|_| |_\__, | / / / /
°=========|_|==============|___/=/_/_/_/
°:: Spring Boot ::°°°°°°°  (v2.4.2)
°
Starting JavaSpringRestApplication
No active profile set, falling back to default profiles: default
Tomcat initialized with port(s): 8080 (http)
Starting service [Tomcat]
Starting Servlet engine: [Apache Tomcat/9.0.41]
Initializing Spring embedded WebApplicationContext
Root WebApplicationContext: initialization completed in 1607 ms
Initializing ExecutorService 'applicationTaskExecutor'
Tomcat started on port(s): 8080 (http) with context path ''
Started JavaSpringRestApplication in 4.582 seconds (JVM running for 7.196)
Initializing Spring DispatcherServlet 'dispatcherServlet'
Initializing Servlet 'dispatcherServlet'
Completed initialization in 9 ms
```

Joli, alors ouvrons un navigateur et allons sur `http://localhost:8080/` (**fig. 4**).

Nous obtenons une page d'erreur – un peu décevant. En effet, nous n'avons encore rien créé qui puisse être affiché sur cette page ! Cependant, cette page d'erreur prouve que notre application fonctionne et se comporte comme un serveur web. Elle ne sait simplement pas encore ce qu'elle doit partager avec nous..

Construction de l'application

Maintenant, étendons ce package de démarrage Spring pour inclure la fonction voulue dans notre application, à savoir la commande des broches GPIO avec du code Java. Pour cela, nous devons inclure la bibliothèque Pi4J dans notre projet. Grâce à elle, nous n'avons pas besoin de commander le matériel nous-mêmes, car nous pouvons utiliser l'ensemble des méthodes Java fournies par Pi4J. Cependant, n'oubliez pas que cette bibliothèque dépend de matériel présent sur le RPi, donc nous ne pouvons pas exécuter ou tester entièrement cette application sur le PC.

Le fichier JAR produit devra être exécuté sur le RPi pour tester la commande des broches GPIO.
Dans les exemples de code fournis ici, chaque variable ou méthode est documentée dans le style JavaDoc (en commençant par `/**`) pour expliquer son but.

Ajout des dépendances

Avec le projet créé avec Spring Initializr ouvert, nous ajoutons la dépendance `pi4j-core` au fichier *pom.xml*. Cela garantit que nous pouvons accéder aux méthodes pi4j :

```
<dependency>
<groupId>com.pi4j</groupId>
<artifactId>pi4j-core</artifactId>
<version>1.3</version>
<scope>compile</scope>
</dependency>
```

Pendant que nous y sommes, nous pouvons également ajouter la dépendance OpenAPI (`springdoc-openapi-ui`) que nous utiliserons plus tard pour tester les services REST :

```
<dependency>
<groupId>org.springdoc</groupId>
<artifactId>springdoc-openapi-ui</artifactId>
<version>1.5.1</version>
</dependency>
```

Ajout d'un contrôleur d'information REST

D'abord nous souhaitons que l'application nous donne les informations sur le RPi accessibles avec la bibliothèque Pi4J. Nous commençons par créer un package « *controller* » avec un fichier *InfoRestController.java*. Dans Visual Studio Code, cliquez avec le bouton droit de la souris sur l'entrée *java\be\webtechie\javaspringrestgpio* et sélectionnez *Nouveau dossier*. Nommez le nouveau dossier « *controller* ». Ensuite, cliquez avec le bouton droit de la souris sur le dossier *controller* et sélectionnez *Nouveau fichier.* Nommez le fichier *InfoRestController.java*.
Dans les sources, vous trouverez le code complet, mais le **listage 2** fournit un court extrait de son contenu. Chaque méthode est une projection REST qui renvoie un ensemble spécifique de paires clé-valeur contenant des informations sur le RPi.

Ajout du gestionnaire GPIO

Avant de pouvoir créer le contrôleur GPIO REST, nous ajoutons un fichier *GpioManager.java* pour gérer les appels Pi4J. Ce fichier est placé dans un package « *manager* » (créé comme un *nouveau dossier* et un *nouveau fichier* comme précédemment). Nous utiliserons ce gestionnaire pour stocker les broches GPIO initialisées et appeler les méthodes Pi4J pour interagir avec les broches GPIO. Encore une fois, un court extrait du code est fourni dans le **listage 3**,

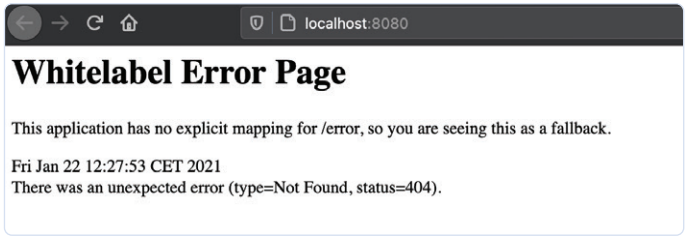


Figure 4. La page web d'erreur par défaut prouve que le serveur web est opérationnel.

Listage 2. Extrait du fichier InfoRestController.java.

```
/**
 * Provides a REST-interface to expose all board info.
 */
@RestController
@RequestMapping("info")
public class InfoRestController {
    private Logger logger = LoggerFactory.getLogger(this.getClass());

    /**
     * Get the OS info.
     */
    @GetMapping(path = "os", produces = "application/json")
    public Map<String, String> getOsInfo() {
        Map<String, String> map = new TreeMap<>();
        try {
            map.put("Name", SystemInfo.getOsName());
        } catch (Exception ex) {
            logger.error("OS name not available, error: {}",
                    ex.getMessage());
        }
        try {
            map.put("Version", SystemInfo.getOsVersion());
        } catch (Exception ex) {
            logger.error("OS version not available, error: {}",
                    ex.getMessage());
        }
        return map;
    }

    /**
     * Get the Java info.
     */
    @GetMapping(path = "java", produces = "application/json")
    public Map<String, String> getJavaInfo() {
        Map<String, String> map = new TreeMap<>();
        map.put("Vendor ", SystemInfo.getJavaVendor());
        map.put("VendorURL", SystemInfo.getJavaVendorUrl());
        map.put("Version", SystemInfo.getJavaVersion());
        map.put("VM", SystemInfo.getJavaVirtualMachine());
        map.put("Runtime", SystemInfo.getJavaRuntime());
        return map;
    }

    ...
}
```

Listage 3. Extrait du fichier GpioManager.java.

```
/**
 * Service instance managing the {@link GpioFactory}.
 */
@Service
public class GpioManager {
    private Logger logger = LoggerFactory.getLogger(this.getClass());

    /**
     * The GPIO controller.
     */
    private final GpioController gpio;

    /**
     * List of the provisioned pins with the address as key.
     */
    private final Map<Integer, Object> provisionedPins =
            new HashMap<>();

    /**
     * Constructor which initializes the Pi4J {@link GpioController}.
     */
    public GpioManager() {
         String osName = SystemInfo.getOsName();
         if (osName.toLowerCase().contains("raspberry") || osName.toLowerCase().contains("linux")) {
             this.gpio = GpioFactory.getInstance();
         } else {
             logger.error("GPIO could not be initialized. Not running on Raspberry Pi but '{}'", osName);
             this.gpio = null;
         }
     }

    /**
     * Get the pin for the given address.
     *
     * @param address The address of the GPIO pin.
     * @return The {@link Pin} or null when not found.
     */
    private Pin getPinByAddress(int address) {
        Pin pin = RaspiPin.getPinByAddress(address);
        if (pin == null) {
            logger.error("No pin available for address {}", address);
        }
        return pin;
    }

    /**
     * Provision a GPIO as digital output pin.
     *
     * @param address The address of the GPIO pin.
```

tandis que le code complet de cette classe se trouve dans les sources du dépôt. Notez que nous utilisons `@Service` dans cette classe qui indique au canevas Spring de garder une seule instance de cet objet en mémoire. Cela garantit que nous maintenons en permanence une cartographie des broches GPIO sélectionnées.

Ajout du contrôleur GPIO REST

Enfin, nous ajoutons un contrôleur GPIO avec une interface REST au package (dossier) du *contrôleur*. Ce fichier nommé *GpioRestController.java* expose les méthodes GPIO de Pi4J définies dans la classe *GpioManager.java*. Une fois encore, un extrait du code est présenté dans le **listage 4** et le code complet est disponible dans le dépôt.

Exécution de l'application

Cette étape peut être réalisée à la fois sur le PC et sur le RPi. Sur le PC, il est préférable d'exécuter l'application à partir de Visual Studio Code. Cela évite d'avoir à installer Maven sur le PC.

Comme nous avons ajouté la dépendance *springdoc-openapi-ui* dans le fichier *pom.xml*, l'application nous fournira une page web Swagger très utile pour tester les services REST. Swagger est un autre projet *open source* qui crée une interface de page web simple pour tester notre code Java en visualisant automatiquement les contrôleurs que nous avons créés.

Il existe deux façons de lancer l'application. Dans Visual Studio Code, vous la lancez avec *Run* (RPi ou PC). L'alternative est de construire l'application dans un fichier jar avec *mvn package* (RPi uniquement). Une fois

```
     * @param name The name of the GPIO pin.
     * @return True if successful.
     */
    public boolean provisionDigitalOutputPin(final int address,
            final String name) {
        if (this.provisionedPins.containsKey(address)) {
            throw new IllegalArgumentException("There is already"
                  + " a provisioned pin at the given address");
        }

        final GpioPinDigitalOutput provisionedPin = this.gpio
            .provisionDigitalOutputPin(
                this.getPinByAddress(address), name, PinState.HIGH);
        provisionedPin.setShutdownOptions(true, PinState.LOW);

        this.provisionedPins.put(address, provisionedPin);

        return true;
    }

    /**
     * Toggle a pin.
     *
     * @param address The address of the GPIO pin.
     * @return True if successful.
     */
    public boolean togglePin(final int address) {
        logger.info("Toggle pin requested for address {}", address);

        Object provisionedPin = this.provisionedPins.get(address);

        if (provisionedPin == null) {
            throw new IllegalArgumentException("There is no pin"
                + " provisioned at the given address");
        } else {
            if (provisionedPin instanceof GpioPinDigitalOutput) {
                ((GpioPinDigitalOutput) provisionedPin).toggle();

                return true;
            } else {
                throw new IllegalArgumentException("The provisioned pin"
                        + " at the given address is not of the type"
                        + " GpioPinDigitalOutput");
            }
        }
    }

    ...

}
```



Figure 5. La page Swagger permet d'accéder aux API REST du contrôleur info et GPIO.



Listage 4. Extrait du fichier GpioRestController.java.

```
/**
 * Provides a REST-interface to interact with the GPIOs.
 */
@RestController
@RequestMapping("gpio")
public class GpioRestController {
    private Logger logger = LoggerFactory.getLogger(this.getClass());

    /**
     * Reference to the GPIO manager service
     */
    private final GpioManager gpioManager;

    /**
     * Constructor used by Spring to "inject" the GPIO manager
     * into this class.
     *
     * @param gpioManager {@link GpioManager}
     */
    public GpioRestController(GpioManager gpioManager) {
        this.gpioManager = gpioManager;
    }

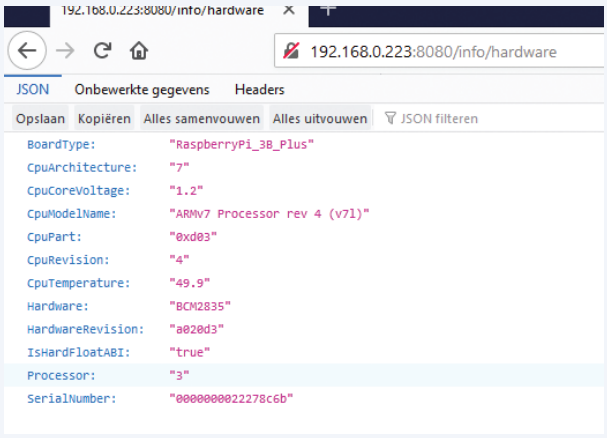
    ...

    /**
     * Provision a GPIO as digital output pin.
     *
     * @param address The address of the GPIO pin.
     * @param name The name of the GPIO pin.
```

construite, elle est exécutée comme suit à partir de la ligne de commande :

```
java -jar target/javaspringrestgpio-0.0.1-
    SNAPSHOT.jar
```

Dans un navigateur sur le RPi, ouvrez la page Swagger en utilisant `http://localhost:8080/swagger-ui.html`. Autre solution : depuis n'importe quel PC sur le même réseau, utilisez `http://<IP_ADDRESS_RASPBERRY_PI>:8080/swagger-ui.html`. Les deux contrôleurs avec leurs méthodes seront affichés comme à la **figure 5**.



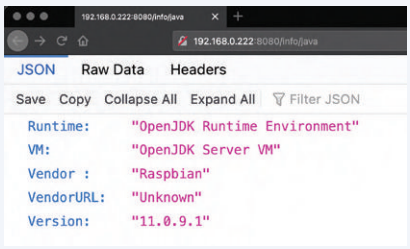


Figure 6. Les données JSON peuvent être acquises directement en ajoutant le nom de la méthode à l'URL.

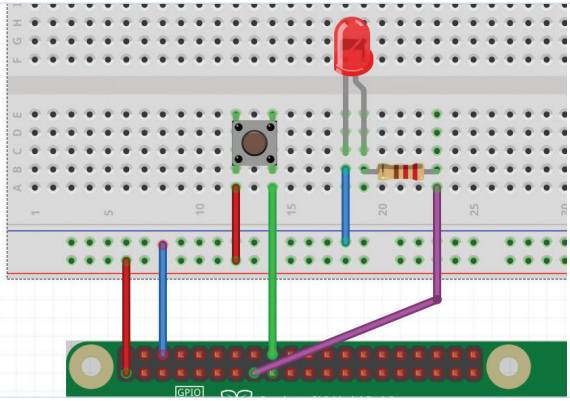


Figure 7. Schéma de câblage pour connecter l'interrupteur et la LED au Raspberry Pi.

Test du contrôleur d'information REST

Nous pouvons cliquer sur les boutons de la page Swagger et exécuter les options disponibles. Par exemple, dans la section *info-rest-controller*, cliquez sur *GET* à côté de la méthode *info/hardware*. Ensuite cliquez sur *Try it out* puis sur *Execute* qui affiche la réponse dans la section *Response body*.

Toutes ces méthodes peuvent également être appelées directement depuis le navigateur. Il suffit d'ajouter le nom de la méthode à l'URL. Pour *info/hardware*, il suffit d'entrer `http://localhost:8080/info/hardware`, et pour *info/java*, c'est `http://localhost:8080/info/java`. Les données issues de l'appel de ces méthodes sont affichées au format JSON, comme le montre la **figure 6**.

```
     * @return True if successful.
     */
    @PostMapping(
        path = “provision/digital/output//”,
        produces = “application/json”)
    public boolean provisionDigitalOutputPin(
            @PathVariable(“address”) int address,
            @PathVariable(“name”) String name) {
        return this.gpioManager
            .provisionDigitalOutputPin(address, name);
    }

    ...

    /**
     * Toggle a pin.
     *
     * @param address The address of the GPIO pin.
     * @return True if successful.
     */
    @PostMapping(
        path = “digital/toggle/”,
        produces = “application/json”)
    public boolean togglePin(@PathVariable(“address”) long address) {
        return this.gpioManager.togglePin((int) address);
    }

    ...
}
```



Tester le contrôleur GPIO REST avec une LED et un bouton

Pour montrer comment cette application peut interagir avec les broches GPIO, un montage très simple sur plaque d’essai suffira (**fig. 7**) :

- LED sur la broche GPIO 5, BCM 22, WiringPi n°3
- Bouton sur la broche GPIO 18, BCM 24, WiringPi n°5

Avant de commander la LED connectée et de lire l’état du bouton, il faut initialiser les broches

Publicité



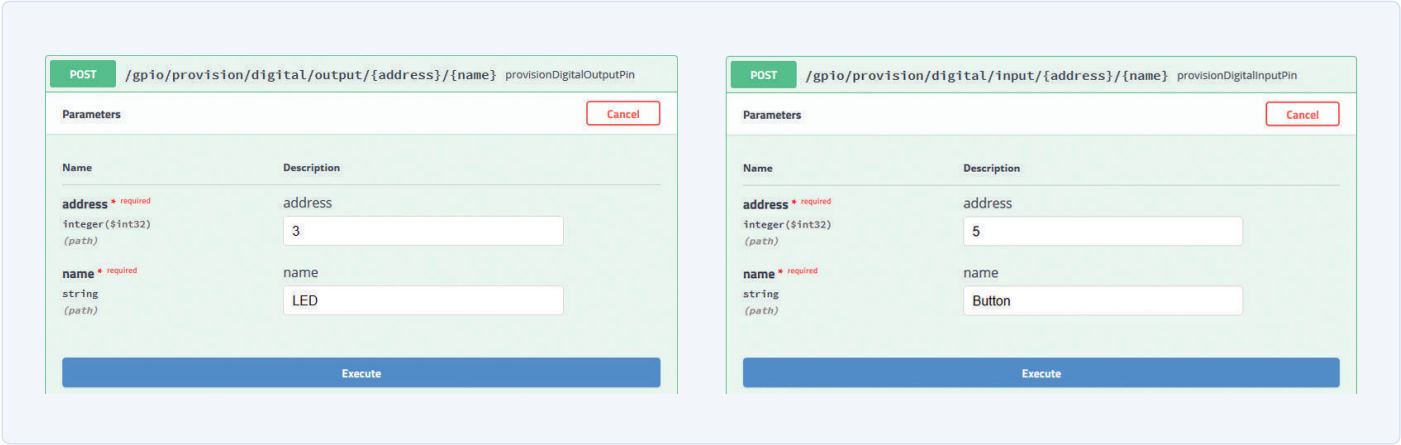


Figure 8. Initialisation des GPIO en utilisant les méthodes `/gpio/provision/digital/`.

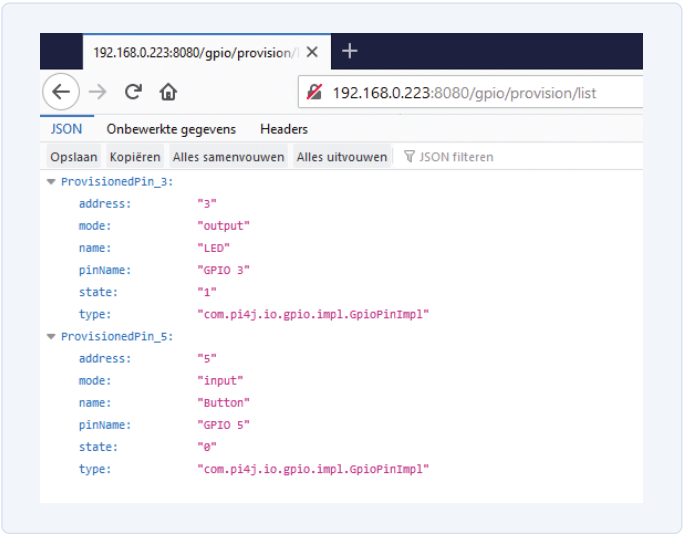


Figure 9. Demande de la configuration GPIO avec la méthode `/gpio/provision/list`.

GPIO. La méthode `/gpio/provision/digital/output` permet de configurer une sortie. De retour à la page `http://localhost:8080/swagger-ui.html`, cliquez sur *GET by this method*, *Try it out* puis entrez *3* dans *address* et *LED* dans *string*. Validez la configuration en cliquant sur *Execute* (**fig. 8**, à gauche).

La méthode `/gpio/provision/digital/input` permet de configurer une entrée. Cliquez sur *GET by this method*, *Try it out*, puis entrez *5* dans *address* et *Button* dans *string*. Validez la configuration en cliquant sur *Execute* (**fig. 8**, à droite).

Une fois les broches GPIO initialisés, la méthode `/gpio/provision/list` permet d'en obtenir la liste. Il suffit d'utiliser *GET*, *Try it out*, et *Execute* ou de taper l'URL `http://localhost:8080/gpio/provision/list` directement dans le navigateur (**fig. 9**).

Maintenant que nous avons vérifié que les broches GPIO sont prêtes à être utilisées, nous pouvons allumer et éteindre la LED en utilisant la méthode `/gpio/digital/toggle` en cliquant plusieurs fois sur le bouton *Execute* (**fig. 10**).

Il existe également une méthode supplémentaire qui permet d'allumer la LED pendant une durée donnée, par ex. 2 s (**fig. 11**). Nota : la durée doit être fournie en millisecondes.

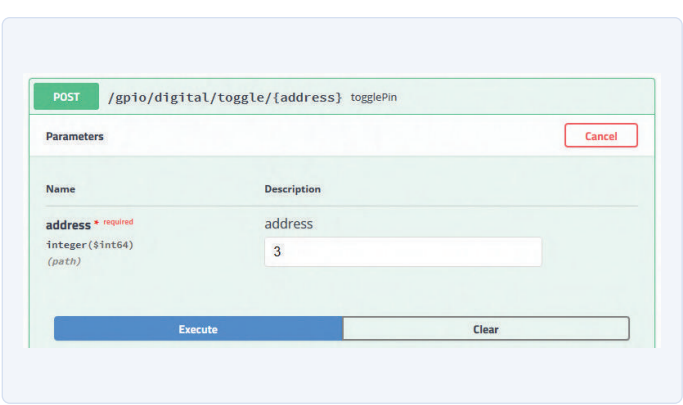


Figure 10. Utilisation du bouton Execute dans l'interface Swagger pour faire basculer la LED.

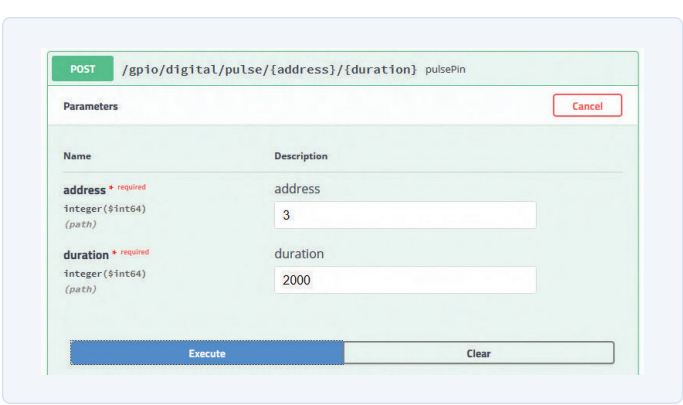


Figure 11. La LED peut également être activée pendant une durée déterminée.

L'interface Swagger permet également de consulter l'état du bouton, également disponible directement via l'URL `http://localhost:8080/gpio/state/5`. Dans le cas de la **figure 12,** le bouton est pressé et renvoie un 1.

Poursuite de l'exploration

Cette application ne présente que quelques-unes des méthodes Pi4J en tant que services REST, cela montre les possibilités et la puissance de cette approche. En fonction de votre projet, vous pouvez étendre ou retravailler cet exemple pour qu'il réponde à vos besoins.
La bibliothèque Pi4J est en cours de réécriture afin d'offrir une meilleure prise en charge du RPi 4 et des futures versions. Cela permettra également de la mettre à jour avec les dernières versions de Java. En attendant, la version actuelle vous permet de démarrer et de créer des applications dans lesquelles vous intégrerez le contrôle du matériel via une API REST.

(200617-B-04)

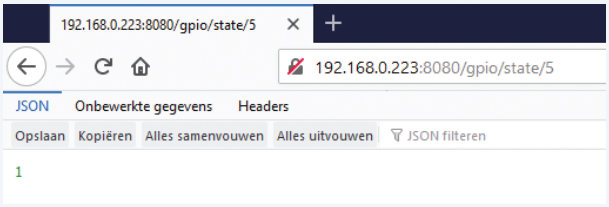


Figure 12. L'URL avec la méthode `gpio/state/5` acquiert l'état du bouton.

Contributeurs
Idée, texte et images : **Frank Delporte**
Rédaction : **Stuart Cording**
Mise en page : **Giel Dols**
Traduction : **Denis Lafourcade**

Des questions, des commentaires ?
Envoyez un courriel à l'auteur (javaonraspberrypi@webtechie.be) ou contactez Elektor (redaction@elektor.fr).

PRODUITS

- **F. Delporte, « Getting Started with Java on the Raspberry Pi » (livre en anglais)**
  www.elektor.fr/19292
- **Kit de démarrage du Raspberry Pi 4**
  www.elektor.fr/19427



LIENS

[1] **F. Delporte, « Java sur le Raspberry Pi – partie 1 : les broches GPIO », Elektor, 05-06/2021** : www.elektormagazine.fr/200617-04
[2] **Dépôt GitHub pour cet article** : https://bit.ly/3i9bP4v
[3] **Documentation de Spring Boot** : https://spring.io/projects/spring-boot
[4] **Spring Initialzr** : https://start.spring.io/
[5] **« What Is REST? », Codecademy** : https://bit.ly/31odThv
[6] **Visual Studio Code** : https://code.visualstudio.com/

Raspberry Pi Compute Module 4

Un Raspberry Pi industriel

Mathias Claußen (Elektor)

Le Raspberry Pi Compute Module 4 est arrivé. Avec un nouveau facteur de forme et de nouveaux périphériques, que vaut-il ? Associé à la carte d'E/S pour Rasbperry Pi Compute Module, il ouvre, pour des systèmes animés par des Raspberry Pi, des possibilités inédites et irréalisables avec les anciens modules. L'interface PCI Express permet maintenant de choisir des périphériques à haute vitesse pour les raccorder au Raspberry Pi.

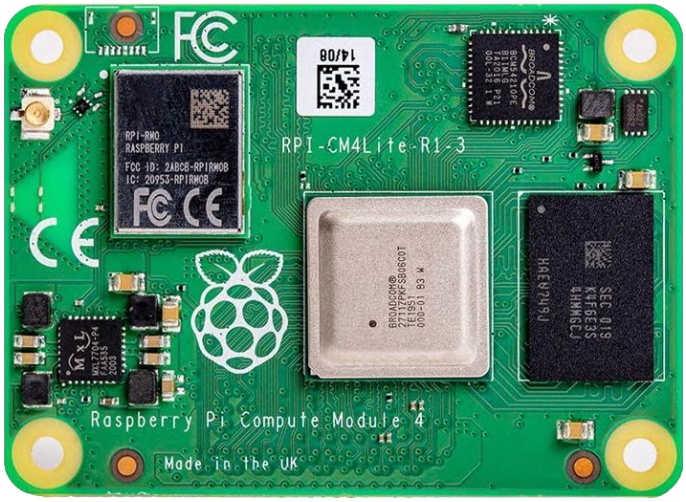


Figure 1. Recto du Raspberry Pi Compute Module 4.

Les rumeurs allaient bon train, et maintenant c'est réel, il existe bel et bien un « module de calcul » basé sur le Raspberry Pi 4 – le Raspberry Pi Compute Module 4 (CM4). Le Compute Module est un Raspberry Pi qui a rétréci pour être intégré dans d'autres produits nécessitant une plus grande souplesse de conception que le Raspberry Pi habituel. Il permet également d'ajouter ses propres circuits électroniques autour du SoC du Raspberry Pi (par ex. pour une utilisation industrielle ou de la signalétique numérique). Les modules de calcul dotés des SoC que l'on trouve sur les Raspberry Pi Zero et 3B(+) ont été commercialisés avec un facteur de forme SO-DIMM, qui s'installe comme de la RAM dans votre PC. Même si cette solution offre une grande souplesse, elle présente aussi quelques inconvénients, car le module ne comporte ni Ethernet ni hub USB. Vous deviez donc les ajouter par vos propres moyens. De même, il n'y avait pas d'option Wi-Fi. Il fallait l'installer sur votre propre carte de base, d'où une augmentation du coût de la carte. Le Compute Module 3 et le Compute Module 1 n'étaient donc pas adaptés pour certaines applications. De plus, l'absence d'une interface à haut débit (les deux modules n'ayant qu'un seul port USB2.0) n'aidait pas pour ajouter des périphériques à large bande passante ou à faible latence. Compte tenu des prix des CM1 et CM3, vous pouviez utiliser un Raspberry Pi ordinaire pour certains projets, car tout ce dont vous aviez besoin était déjà installé dans un facteur de forme bien connu. Le Compute Module 4 change radicalement les choses, car on a beaucoup réfléchi pour le rendre meilleur que les versions précédentes.

Exploration du circuit imprimé

La **figure 1** et la **figure 2** permettent de découvrir le circuit imprimé. Le CM4 est maintenant disponible en 32 variantes : avec ou sans

liaison sans-fil ; EMMC embarquée de 8, 16 ou 32 Go ; sans EMMC embarquée ; 1, 2, 4 ou 8 Go de RAM. Choisissez la version qui vous convient le mieux. Consultez l'encadré **Caractéristiques** (elles sont identiques à celles du RPi 4). La **figure 3** montre l'emplacement des composants que l'on trouve dans un CM4 avec et sans EMMC installée. Sur la gauche se trouve le PMIC, en charge de tous les rails d'alimentation nécessaires au fonctionnement du CM4 (**fig. 4**). À côté se trouve le module sans fil qui prend en charge le Wi-Fi à 5 et 2,4 GHz et fournit également la connectivité Bluetooth. Au centre (**fig. 5**), on voit le SoC qui anime le CM4 : le BCM2711 à quatre cœurs Cortex-A72. En haut à droite (**fig. 6**), un BCM54210PE s'occupe de l'Ethernet 1000Base-T, avec prise en charge de l'IEEE1588-2008 pour la synchronisation de précision. Cela signifie qu'il vous suffit de monter un connecteur MagJack® sur votre carte de base pour disposer d'une connexion Ethernet (toutefois n'oubliez pas d'ajouter une protection contre les décharges électrostatiques). Sur le côté droit (**fig. 7**), nous avons un module de RAM et sur le dessus une petite puce flash avec les informations primaires de démarrage.

Au dos de la carte (fig. 2), vous trouvez les traditionnels condensateurs et oscillateurs nécessaires pour le fonctionnement du SoC. Vous pouvez également voir deux connecteurs Hirose à 100 broches prévus pour fixer le CM4 sur une carte de base. Il s'agit d'un changement par rapport à l'ancien CM. L'un des côtés du connecteur est relié aux alimentations et aux signaux à faible vitesse ; l'autre côté porte tous les signaux à haute vitesse, ce qui rend le routage moins compliqué. Pour l'alimentation, il suffit de fournir 5 V au module ; le convertisseur CC/CC embarqué produit toutes les tensions requises pour le CM4. Reportez-vous à nouveau à l'encadré **Caractéristiques**. L'interface USB 3.0 du RPi 4B n'est pas répertoriée, et il y a une bonne raison à cela. Avec le RPi 4B, il y avait un contrôleur USB 3.0 relié au BCM2711 via une connexion PCI Express 2.0 à ligne unique. Ce circuit n'est pas inclus dans le CM4 et rend disponible la ligne PCI Express pour notre usage propre. Cela ouvre de nouvelles possibilités pour utiliser un Raspberry Pi là où les précédents ne pouvaient pas fonctionner. Dans la section suivante, je vais expliquer comment débuter avec le module et développer vos propres montages.

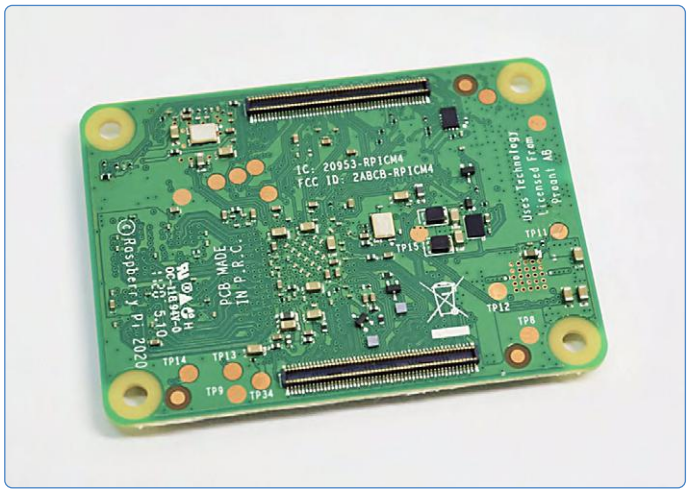


Figure 2. Verso du Raspberry Pi Compute Module 4.

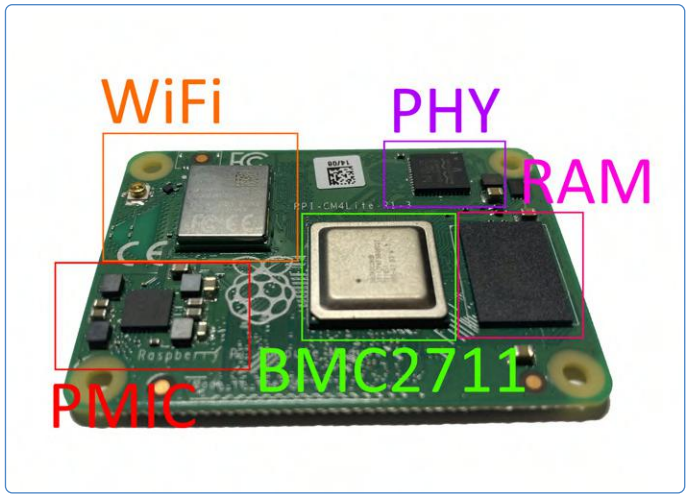


Figure 3. Emplacement des composants du Raspberry Pi Compute Module 4.



Figure 4. PMIC.



Figure 5. SoC BCM2711.



Figure 6. Ethernet.

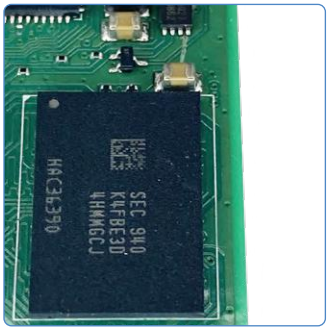


Figure 7. Puce RAM et EEPROM.

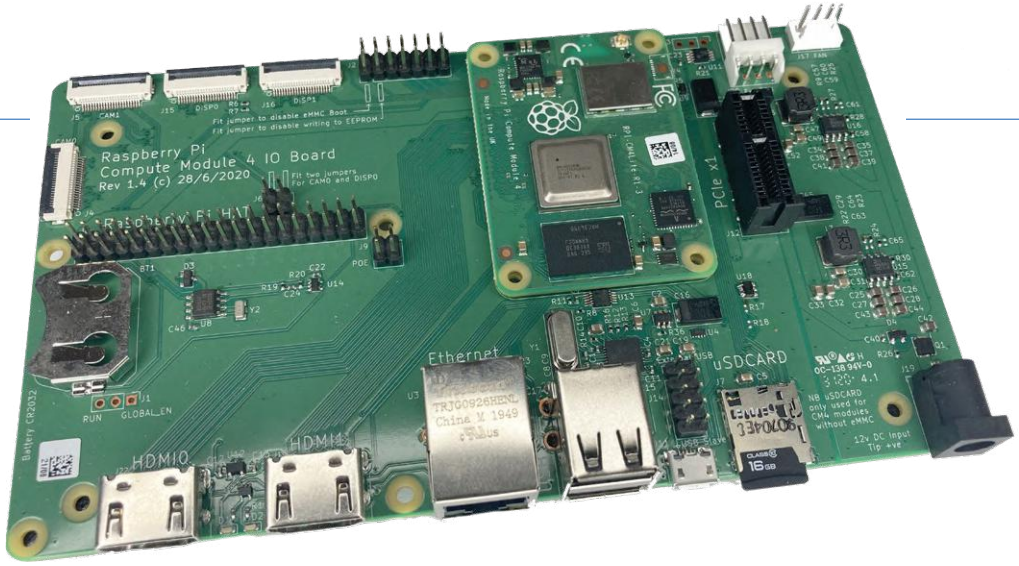


Figure 8. Carte d'E/S.



Figure 9. Ports CSI.



Figure 10. Ports DSI.

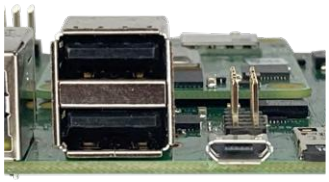


Figure 11. Ports USB.

Carte Raspberry Pi Compute Module 4 I/O

Pour faciliter la prise en main du CM4, la Fondation Raspberry Pi a créé une carte d'E/S (**fig. 8**) qui donne accès à toutes les fonctions intéressantes du module de calcul et inclut également un emplacement PCI Express 1x. La carte offre plus de connectivité que les autres produits RPi 4, comme deux entrées caméra (interface CSI-2 avec deux ou quatre canaux, voir **fig. 9**) ou deux connecteurs d'affichage (DSI avec deux ou quatre canaux, voir **fig. 10**). Cela permet l'emploi de plusieurs moniteurs et de fonctions de traitement de la vision. Besoin de connecter une souris et un clavier ? Le port USB 2.0 interne du CM4 a été ajouté sur la carte d'E/S sous la forme d'un hub USB 2.0, ce qui fournit jusqu'à quatre ports USB 2.0 (**fig. 11**). La carte d'E/S comprend également un module RTC qui peut être utilisé comme chien de garde ou pour démarrer le système à intervalles donnés.

Pour son alimentation, la carte se contente d'un bloc d'alimentation de 12 V (si vous utilisez des cartes PCI Express). Si vous n'utilisez pas l'emplacement PCI Express, la tension d'entrée peut aller jusqu'à 26 V. Un changement bienvenu sur la carte d'E/S est la présence de deux ports HDMI de taille standard qui permettent d'utiliser les câbles HDMI normaux (**fig. 12**). Enfin, vous disposez d'un port LAN et d'un emplacement pour carte micro-SD. Ce qui rend la carte d'E/S intéressante, c'est la publication des fichiers de projet KiCad qui permettent un démarrage rapide si vous utilisez KiCad. Le CM4 est ajouté ici en tant que composant et peut facilement être relié à vos propres cartes et montages sans avoir à créer un composant dans KiCad (**fig. 13**). Il manque juste un modèle 3D.

Caractéristiques

- SoC à 64 bits Broadcom BCM2711 à quatre cœurs Cortex-A72 (ARM v8) @ 1,5 GHz
- LPDDR4 à 1 Go, 2 Go, 4 Go ou 8 Go (selon la variante)
- LAN sans fil en option, IEEE 802.11b/g/n/ac à 2,4 GHz et 5,0 GHz
- Bluetooth 5.0, BLE avec options pour antenne embarquée ou externe
- Gigabit Ethernet intégré prenant en charge IEEE1588
- 1× interface USB 2.0
- Interface PCIe Gen 2 ×1
- 28 signaux GPIO
- Interface pour carte SD ou eMMC externe (à utiliser uniquement avec les variantes du CM4 sans eMMC)
- Double interface HDMI (prise en charge jusqu'à 4Kp60)
- Interface d'affichage MIPI DSI à 2 voies
- Interface de caméra MIPI CSI à 2 voies
- Interface d'affichage MIPI DSI à 4 voies
- Interface de caméra MIPI CSI à 4 voies
- Multimédia : H.265 (décodeur 4Kp60) ; H.264 (décodeur 1080p60, codeur 1080p30)
- Graphique OpenGL ES 3.0
- Alimentation : 5 V CC
- Température de fonctionnement : –20 °C à +85 °C
- Durée de vie : le Raspberry Pi Compute Module 4 restera en production au moins jusqu'en janvier 2028.

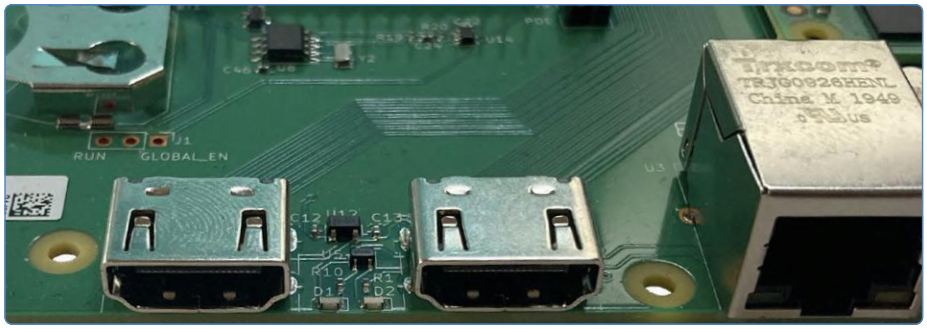


Figure 12. Ports HDMI.

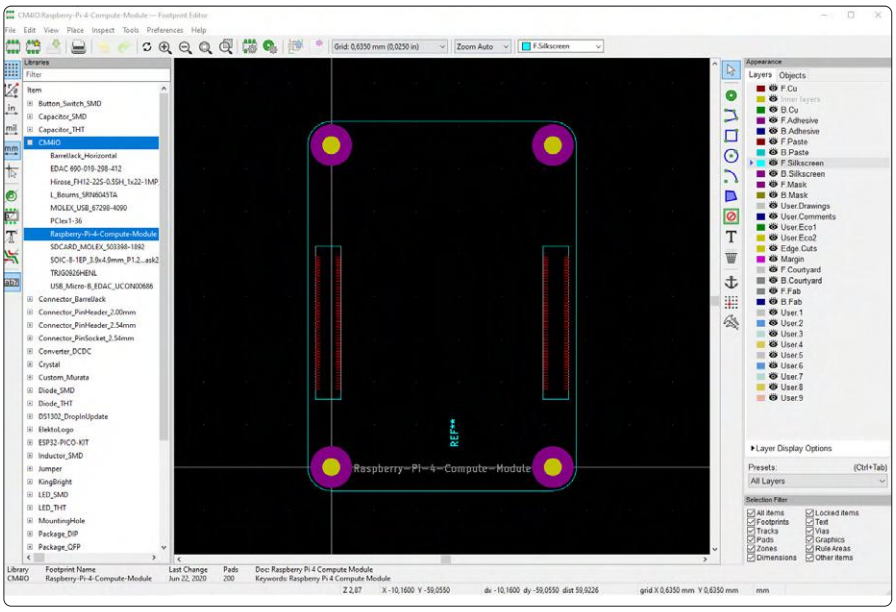


Figure 13. Capture d'écran du composant KiCad.

Assemblage du puzzle

Si vous avez l'intention de développer avec le CM4, l'associer à la carte d'E/S est souhaitable car elle expose toutes les interfaces d'une manière agréable à utiliser et donne également accès à la ligne PCI Express. La mise en route est aussi simple qu'avec un RPi 4 ordinaire : écrivez une image sur une carte SD et démarrez-le. Si une souris et un clavier sont connectés aux ports USB, ils n'apparaîtront pas et ne fonctionneront pas comme sur le RPi 4. Si vous voulez utiliser les ports USB avec des périphériques, ajoutez une ligne dans le fichier *config.txt*, qui se trouve dans la partition *BOOT* de la carte SD. Ajoutez `dtoverlay=dwc2,dr_mode=host` à la fin de *config.txt*, pour que les ports USB soient activés au démarrage suivant. Avec l'emplacement PCI Express 1x (une ligne) monté, l'ajout de cartes compatibles n'est pas un problème, mais vous devez garder à l'esprit que ces cartes sont alimentées en 12 V, donc assurez-vous que la puissance fournie soit suffisante. Une partie intéressante de la spécification de PCI Express permet aux cartes qui sont conçues pour des emplacements avec une configuration 4x, 8x ou 16x de fonctionner également avec moins de lignes – ce qui signifie qu'elles devraient fonctionner dans des emplacements 1x. Mais pour des raisons mécaniques, une telle carte ne peut pas être branchée directement sur la carte d'E/S.

Heureusement, avec l'augmentation du minage de cryptomonnaies à l'aide de cartes graphiques, ce problème est également apparu sur les systèmes PC modernes et des solutions ont été développées. Pour moins de 10 €, vous pouvez trouver une carte d'adaptation appropriée avec sa propre alimentation (**fig. 14**) et la monter sur la carte d'E/S. Après le démarrage du CM4 et l'exécution de Linux, `lspci` permet d'avoir un aperçu des cartes PCI Express détectées. Comme vous pouvez le voir sur la **figure 15,** une carte réseau Intel à double port est connectée. Celle-ci est connue pour avoir un pilote Linux qui en général fonctionne et devrait ajouter des ports réseau supplémentaires au CM4. La sortie de `lspci` semble prometteuse, la carte est reconnue, mais il n'y a pas de pilote chargé. La raison en est simple : le noyau de base livré avec le système d'exploitation Raspberry Pi ne contient que les

Produits

- **Raspberry Pi Compute Module 4 (CM4)**
  www.elektor.fr/raspberry-pi-compute-module-4-cm4
- **Carte d'E/S pour Raspberry Pi Compute Module 4**
  www.elektor.fr/raspberry-pi-compute-module-4-io-board
- **Kit d'antennes pour Raspberry Pi Compute Module 4**
  www.elektor.fr/raspberry-pi-compute-module-4-antenna-kit

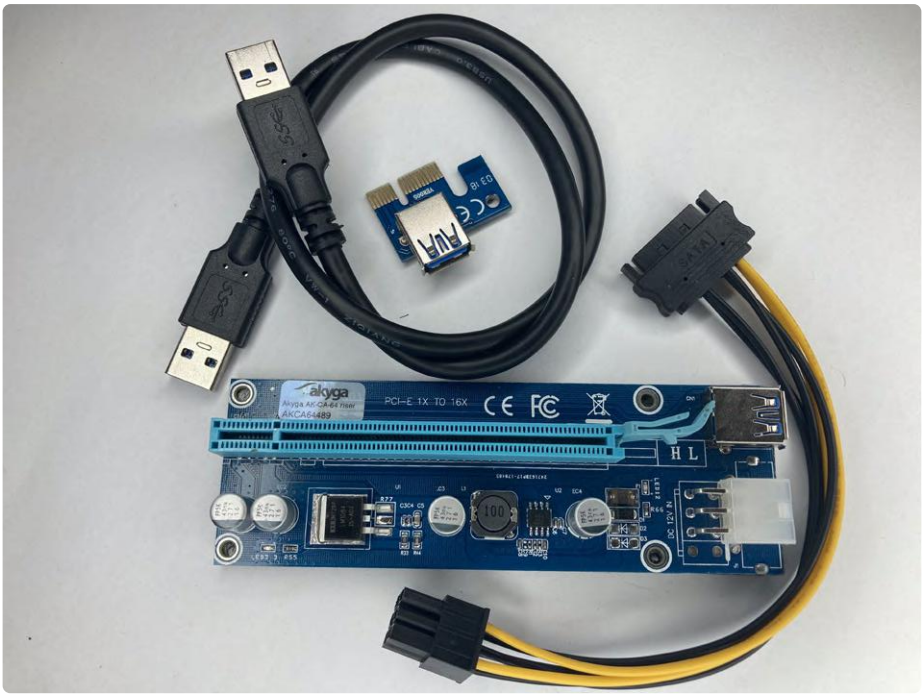


Figure 14. Carte d'adaptation PCI Express.



Figure 15. Cartes réunies : CM4 + carte d'E/S + autres composants.

pilotes nécessaires à l'utilisation quotidienne. Est-ce aussi simple de connecter une carte graphique PCIe ? La connexion est possible, mais elle ne fonctionnera pas comme vous pouvez le voir dans cette vidéo [1]. Pour un contrôleur Ethernet et d'autres périphériques PCI Express comme les cartes SATA, vous devez recompiler le noyau du RPi 4 [2] pour les faire fonctionner. Comme il s'agit d'un problème logiciel, il peut être résolu, mais ce n'est pas le moment de construire un nouveau noyau pour le pilote de base. Le logiciel, la construction du noyau et du pilote constituent un sujet distinct qui sera abordé plus tard. En outre, certains périphériques PCI Express ne fonctionnent pas du tout avec le CM4 car le pilote s'attend à quelque chose qui ne peut être trouvé que dans le monde X86/AMD64, comme c'est actuellement le cas avec le pilote de cartes graphiques.

Lancez-vous !

Le Compute Module se présente désormais sous un nouveau facteur de forme et facilite la conception d'une première carte. En outre la carte d'E/S est fournie avec des symboles et des schémas KiCad, ce qui simplifie la conception de montage. Pourquoi ne pas intégrer un CM4 à un décodeur TV ou un ordinateur de poche de fabrication artisanale ? Avec la ligne PCI Express exposée, nous pouvons ajouter des cartes Ethernet à quatre ports pour avoir en tout cinq ports gigabit. Cela ressemble aux ingrédients d'un routeur basé sur le Raspberry Pi Compute Module 4.
Compte tenu du prix du CM4 et de celui de la carte d'E/S, le démarrage ne coûte pas trop cher, avec des accessoires abordables. Cette nouvelle version du Compute Module est idéale pour vos projets à base de Raspberry Pi.

(200590-04)

LIENS

[1] **GPU sur un Raspberry Pi** : www.youtube.com/watch?v=ikpgZu6kLKE

[2] **Interface réseau à quatre ports sur un Raspberry Pi 4** : www.youtube.com/watch?v=KL0d68j3aJM

Contributeurs
Auteur : **Mathias Claußen**
Rédaction : **Jens Nickel**
Mise en page : **Harmen Heida**
Traduction : **Denis Lafourcade**

Des questions, des commentaires ?
Envoyez un courriel à l'auteur (mathias.claussen@elektor.com).

moniteur de la qualité de l'air, portable et autonome, pour particules de 2,5 µm

Gardez un œil sur votre santé



Laurent Labbe (France)

En cette période de pandémie du COVID-19, on en oublierait presque qu'il existe d'autres raisons de porter un masque. Et qu'il y a des endroits où cela est recommandé, voir nécessaire, depuis bien plus longtemps. Dans les grandes villes d'Asie, les sports en plein air sont souvent problématiques en raison de la pollution de l'air, en particulier pour les particules de 2,5 µm, également appelées PM2.5. Des concentrations élevées de ces particules dans l'air ambiant peuvent être nocives et même dangereuses pour votre santé. Le moniteur de PM2.5 présenté dans cet article peut être utilisé pour surveiller la qualité de l'air.

J'ai déjà réalisé un moniteur de PM2.5, rechargeable avec un panneau solaire, mais il n'était pas adapté à une utilisation en intérieur. J'ai également conçu une version basée sur un ESP32, où j'ai testé différents capteurs et dont les données étaient envoyées par WiFi à Thingspeak, mais il n'était pas portable. L'idée principale de ce projet est de concevoir un appareil le plus compact possible, alimenté par batterie, avec une autonomie de plusieurs semaines et pouvant afficher en permanence le taux de PM2.5 présent dans l'air. Ce moniteur portable peut être placé à l'extérieur, par exemple avec une ventouse collée sur une fenêtre afin que la mesure soit visible de l'intérieur. Évidemment, vous pouvez parfaitement l'utiliser en intérieur.

Détails de la conception

Le choix du capteur de particules est très important. Dans un précédent projet, j'en ai testé plusieurs types, comme une version d'extérieur : le capteur Sharp GP2Y10. Cependant, si nous souhaitons une conception compacte, seul le capteur Plantower PMS7003 [10] est envisageable. Il dispose d'une interface série pour l'envoi des données mesurées. Lorsque vous commanderez ce capteur, assurez-vous d'acheter également le connecteur correspondant ! Comme ce capteur a besoin de 100 mA sous 5 V (principalement en raison de son mini-ventilateur interne), il ne peut pas fonctionner en continu, les batteries s'épuiseraient en un rien de temps. La tension d'alimentation est fournie par une batterie au lithium, et il faut un convertisseur CC/CC (3,8 V vers 5 V) possédant une réelle mise hors tension. L'afficheur sélectionné est un LCD alphanumérique monochrome de 2×8. Il consomme moins de 1 mA en affichage permanent. Des afficheurs couleur ou OLED ne conviendraient pas en raison de leur forte consommation électrique, ce qui affecterait gravement l'autonomie du projet. Pour une lecture rapide de la mesure des PM2.5, un bargraphe à LED est ajouté. Les six niveaux présents dans le **tableau 1** sont affichés au moyen de cinq LED avec le niveau le plus élevé signalé par les LED rouge et bleue allumées simultanément (indice bordeaux dans le tableau). Toutes les LED sont de type basse consom-

Air Quality Index (AQI Values)	Levels of Health Concern	Colors
0 to 50	Good	Green
51 to 100	Moderate	Yellow
101 to 150	Unhealthy for Sensitive Groups	Orange
151 to 200	Unhealthy	Red
201 to 300	Very Unhealthy	Purple
301 to 500	Hazerdous	Maroon

Tableau 1. Explication des niveaux IQA.

mation, par ex. Série Kingbright WP71xx qui ont un courant direct de 2 mA. Une version faible consommation et forte luminosité peut être envisagée si l'équipement est placé en extérieur.
Un affichage de la température et de l'humidité (optionnel) est prévu à l'aide d'un capteur I²C SHT20/SHT21. Le microprogramme prend également en charge le capteur de température 1-Wire DS1820, mais il n'a pas été testé dans ce projet.
Le cœur du projet est un PIC18F2520, accompagné d'un MAX931 pour surveiller la tension de la batterie et d'un LTC4054 pour la recharger. La consommation globale du détecteur dépend principalement de l'intervalle entre les mesures. Selon la spécification du PMS7003, il doit être mis sous tension pendant au moins 30 s avant que la première mesure fiable puisse être effectuée. Par défaut, l'intervalle est réglé sur 20 mn, mais il pourra être modifié dans le menu. Avec cet intervalle et une batterie de 1000 mAh, ce détecteur peut fonctionner pendant deux semaines sans le recharger. N'importe quelle batterie au lithium, LiPo ou de forme prismatique convient, cependant ses dimensions dépendent du boîtier. Pour ce projet, et ayant déjà conçu un boîtier compact, j'ai fait en sorte que la batterie soit la plus grande possible (en volume) dans l'espace restant. J'ai trouvé une batterie de 1100 mAh qui s'adapte parfaitement.

Caractéristiques de l'affichage de PM2.5

La mesure du niveau de particules de 2,5 µm est effectuée à intervalles réguliers. Celle-ci peut être ajustée dans le menu entre 10 mn et 60 mn, 20 mn étant un bon compromis entre précision de mesure et consommation électrique. Toutes les heures, le programme calcule la moyenne des dernières mesures conservées dans un tableau. Il existe plusieurs façons de calculer le niveau de qualité de l'air (associé aux PM2.5). Dans ce projet, deux méthodes sont implémentées :

- IQA : moyenne arithmétique sur 24 h
- IQA nouvelle version : moyenne glissante pondérée sur les 12 h dernières heures (appelée NQI). La mesure la plus récente a un poids plus important que celle faite il y a 11 h.

Par défaut, la méthode NQI est employée.
Les LED du bargraphe peuvent afficher :

- La mesure la plus récente
- La moyenne sur une heure
- L'IQA ou le NIQ

Les LED peuvent être désactivées pendant la nuit pour économiser les batteries. Une photorésistance mesure la luminosité ambiante et permute le mode (jour/nuit).
Une option intéressante est l'affichage instantané du niveau de PM2.5 en temps réel. Le capteur est actif en permanence et la valeur affichée est actualisée toutes les secondes. Cependant, cette option consomme beaucoup d'énergie et décharge la batterie en peu de temps.

Les autres valeurs pouvant être affichées sont :

- La température et l'humidité
- La tension de la batterie sur 3 chiffres (par exemple « 384 » signifie 3,84 V)
- Lorsque la tension de la batterie est faible, les LED clignotent, indiquant qu'il faut recharger l'appareil

Un port est prévu pour connecter, en option, une cellule photovoltaïque de 5 V pour permettre le rechargement lors d'une utilisation en extérieur.
Les paramètres du menu sont enregistrés dans l'EEPROM du microcontrôleur. Toutes les autres valeurs ainsi que les tableaux et les variables sont réinitialisés lorsque l'appareil est mis hors tension ou redémarré.

Calcul de l'indice de qualité de l'air

Il existe de nombreuses méthodes pour calculer l'indice de qualité de l'air. Dans ce projet, deux versions sont implémentées. La première est la moyenne sur les dernières 24 h en utilisant la méthode de calcul des États-Unis (voir [1] pour des détails sur cette méthode). L'inconvénient de cette méthode est qu'elle utilise une moyenne glissante sur 24 h. Par conséquent, la valeur affichée est influencée par des mesures relativement anciennes. Pour cette raison, la seconde méthode de calcul (*NowCast*) repose sur une moyenne pondérée. La mesure effectuée lors de la dernière heure a un poids plus fort dans le calcul final que celle d'il y a 11 h. C'est aussi une moyenne glissante, mais sur 12 h. Baptisée NQI (*New Quality Index*), elle garde les mêmes limites que la première méthode. Vous trouverez en [2] une explication détaillée de cette méthode. Mais pour faire simple, la valeur mesurée lors de l'heure précédente (H–1) possède un poids de 50%, celle de H–2 a un poids de 25%, celle de H–3 a un poids de 2,5%, etc. L'algorithme est discuté en [3].

Le matériel

Le schéma de la **figure 1** montre les entrailles de ce moniteur de la qualité de l'air. Le convertisseur élévateur CC/CC du capteur PMS7003 est un LT3525ESC6-5 (U6). Il s'agit d'un convertisseur élévateur dit « à coupure réelle » (*true shutdown* en anglais). La plupart des convertisseurs élévateurs CC/CC ont une fonction d'arrêt qui stoppera simplement la commutation du transistor dans le circuit. Mais la tension de sortie suivra toujours l'entrée à travers la bobine et le circuit consommera toujours de l'énergie. Les circuits à coupure réelle arrêtent bel et bien le convertisseur et aucun courant ne peut circuler vers la sortie. Le seul vrai problème avec ce circuit intégré est de le souder. Il est vraiment petit, de la pâte à braser et une station à air chaud sont recommandées…
Le régulateur LP2980 3,3 V (U7) est commandé par U1, un comparateur à hystérésis MAX931. Il éteint le régulateur si la tension de la batterie descend en dessous de 3,2 V et ne peut être réactivé que si celle-ci dépasse 3,8 V. Cette fonction, pas forcément utile en mode normal, est nécessaire lors de l'utilisation d'un panneau solaire pour la recharge. Le MAX931 stoppera également le convertisseur CC/CC via la diode D5 pour éviter les démarrages involontaires lorsque la batterie est faible. La liaison série est partagée entre le connecteur externe J8 (pour le monitoring et

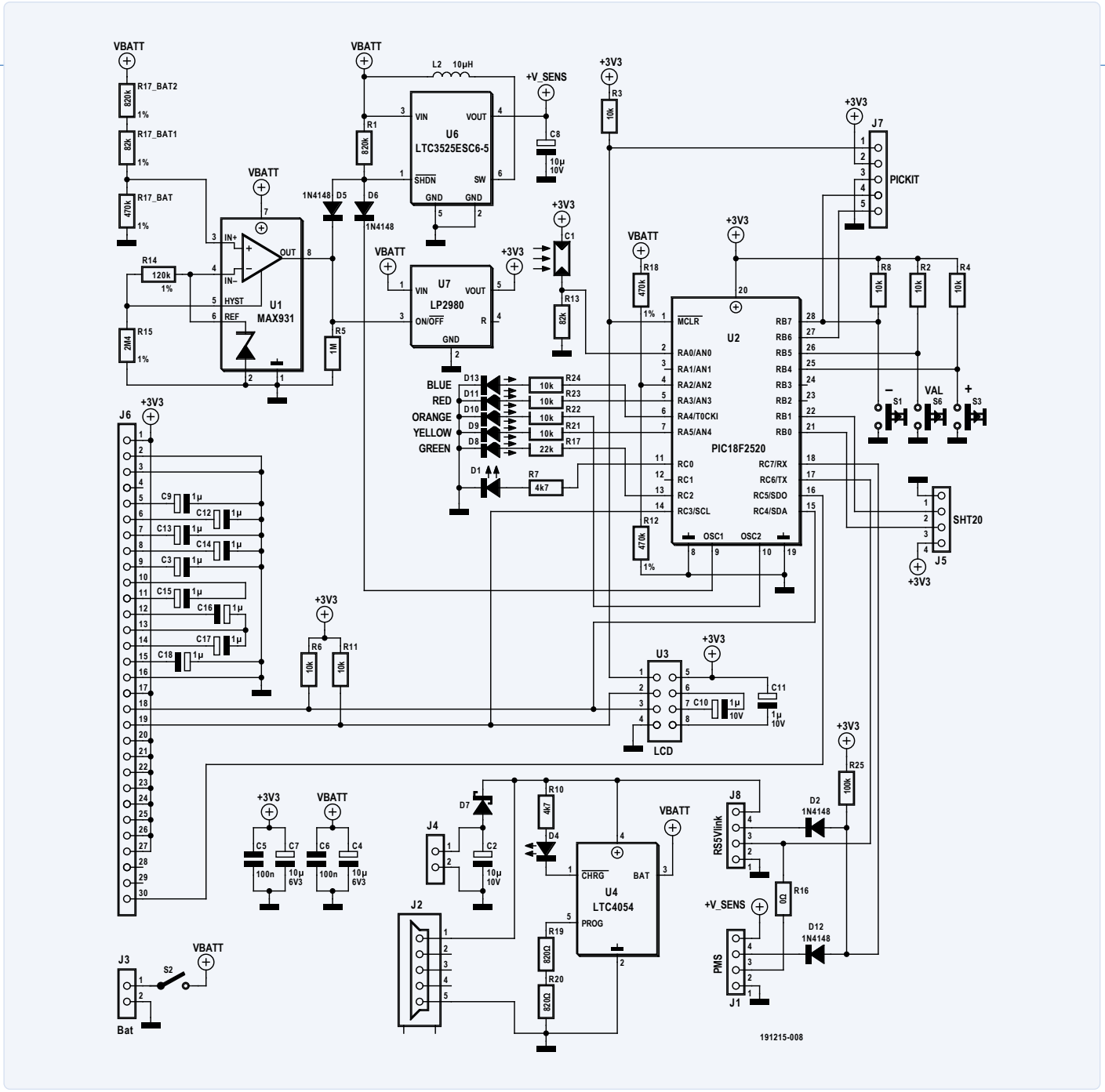


Figure 1. Schéma du moniteur de PM2.5.

le téléchargement) et le PMS7003 (connecté à J1). Le signal TX du microcontrôleur est connecté aux deux connecteurs. Le signal RX est, quant à lui, connecté via une fonction OU logique créée avec les diodes D2 et D12. Comme le microprogramme n'envoie pas de commandes au PMS dans sa version actuelle, la résistance R16 (0 Ω) n'est pas montée.

Pour le capteur SHT20/21, la connexion I²C se fait par émulation logicielle, via RB0 et RB1, sans utiliser l'I²C interne du PIC (réservé à l'affichage). L'afficheur est un LCD de Eastrising, de type COG (*Chip-on-Glass*) à 8×2 caractères, piloté par I²C [7]. Il est connecté via U3.

Le chargeur de batterie au lithium est un LTC4054 (ou un EUP8054 équivalent, U4). Il peut être alimenté soit par USB (J2) soit avec un panneau solaire externe connecté à J4. Le courant est limité à 400 mA par les deux résistances R19 et R20. La LED D4 indique que la batterie est en charge, elle est placée à proximité du connecteur USB. Le connecteur J4 sert à connecter un panneau solaire de 5 V, qui rechargera la batterie via la diode Schottky D7.

Les cinq LED du bargraphe (D8 à D11 et D13) sont commandées par le PIC. La résistance de limitation de courant de chaque LED dépend du type de LED utilisé (faible puissance ou non) et de la tension directe de chaque LED de couleur. La LED D1 (rouge) s'allume lorsque le convertisseur CC/CC est actif et donc lorsque le capteur de particules fonctionne.

Concernant l'afficheur, les condensateurs C10 et C11 peuvent être compris entre 1 µF et 2,2 µF, pour une tension de 10 V. Le rétroéclairage de l'afficheur n'est pas utilisé (et peut être retiré avant le montage). Il est attaché au circuit imprimé avec du ruban adhésif double face. Il est recommandé d'acheter l'afficheur avec son connecteur spécifique.

La luminosité permutant le mode jour/nuit est détectée par une photorésistance (C1) placée sur le panneau avant. Elle est connectée à l'entrée analogique AN0 du microcontrôleur.

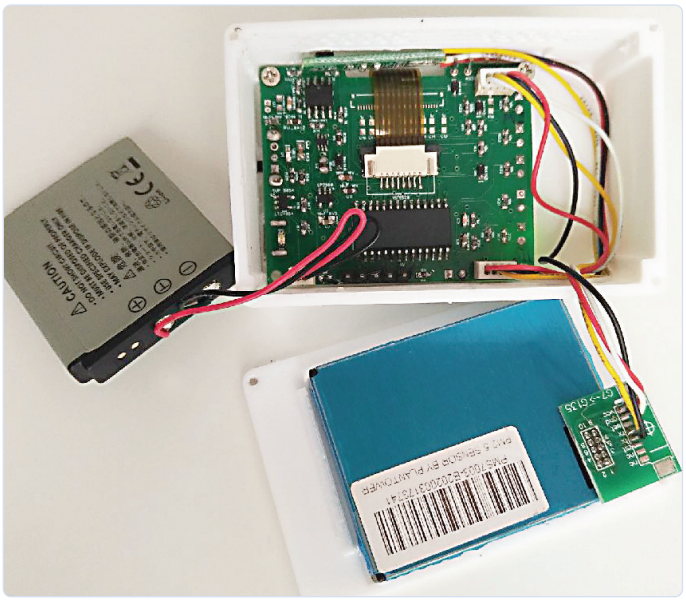


Figure 2. Le circuit imprimé, la batterie et le capteur PMS7003 dans le boîtier ouvert.



Figure 3. Le moniteur de PM2.5 dans son boîtier imprimé en 3D, vue de face.

Assemblage du circuit imprimé

Le schéma et le circuit imprimé ont été conçus avec le logiciel de CAO DipTrace de Novarm, les fichiers de conception sont disponibles en [9]. Tous les composants à l'exception des LED sont des CMS. La partie la plus difficile à souder est le convertisseur CC/CC, j'ai utilisé de la pâte à braser et une station à air chaud pour le faire. Compte tenu de la compacité du boîtier, il est difficile de monter des connecteurs standards pour les deux capteurs. Il est donc préférable de souder les fils directement sur le circuit. Seuls la batterie et l'afficheur utilisent des connecteurs.

Il existe deux solutions pour programmer le microcontrôleur : connecter un PicKit3 à J7 ou via le connecteur série partagé avec le capteur PMS7003. Dans ce cas, le µC doit être préprogrammé avec un chargeur d'amorçage (*bootloader* en anglais) avant montage et le PMS7003 doit être déconnecté chaque fois que le µC est reprogrammé. Personnellement, j'utilise toujours le chargeur d'amorçage appelé Tiny bootloader, disponible en [6]. Il faut que la version flashée sur le PIC ait la bonne valeur pour le chien de garde (*watchdog* en anglais) (ici 256 ms) et donc la source doit être recompilée avec celle-ci. Les fichiers ASM et HEX de ce chargeur d'amorçage pour le PIC18F2520 font également partie des téléchargements de la page Elektor Labs du projet [9].

Le microprogramme

Le microprogramme est écrit en C, avec une ancienne version du compilateur MikroC. Il peut être porté sur la nouvelle version, MikroC Pro, mais je n'ai pas essayé. Les fichiers sources et HEX sont disponibles en [9]. Le microprogramme est assez simple :

- Initialisation
- Une boucle sans fin scrute l'état des boutons-poussoirs
- Affichage des différents écrans et du niveau des LED (en fonction de la lecture de la luminosité pour la commutation jour/nuit)
- Toutes les xx minutes, le convertisseur 5 V CC/CC du capteur de particules est activé. 30 s plus tard, la mesure des PM2.5 est lue et le convertisseur CC/CC est désactivé.
- Toutes les heures, la valeur moyenne est calculée
- Enregistrement de cette valeur dans un tableau à 24 positions (pour 24 h)
- Calcul de l'IQA (moyenne glissante sur 24 h) ou NQI (moyenne pondérée, glissante sur 12 h)

La fonction `calculation_aqi()` calcule la valeur de l'indice de qualité de l'air, suivant le principe de la moyenne simple sur 24 h. Un tableau contient les 24 dernières valeurs et leur simple moyenne est calculée. La fonction `calculation_nqi()` calcule la valeur pondérée de l'indice de qualité de l'air. La formule est expliquée en [3].

La conversion de ces valeurs en indices (0-500) est effectuée par la fonction `conversion_aqi()`, selon la formule indiquée dans le document [1] pour les USA.

La boucle principale du microprogramme se termine par une instruction « sleep ». Le processeur se réveille lorsque le chien de garde est de nouveau déclenché, l'intervalle de surveillance doit être réglé sur 256 ms (dans le cas du PIC18F2520, cela doit être fait lors de la programmation du µC). Cet intervalle est suffisamment court pour ne pas manquer les pressions sur les touches. Un temps plus long serait préférable en ce qui concerne la consommation d'énergie, mais il serait alors possible de manquer une pression sur les boutons. Le microprogramme fonctionne également sans capteur de température/humidité câblé, mais bien sûr, l'afficheur affichera alors des valeurs invalides pour la température et l'humidité.

Le boîtier

Le boîtier a été conçu dans ThinkerCad et imprimé sur une imprimante 3D. C'est assez simple, avec des ouvertures pour le capteur SHT sur le dessus ainsi que pour le capteur PMS7003, une pour l'entrée d'air, une pour la sortie. La version la plus récente du boitier est téléchargeable en [4]. Un support pour cellule solaire est également proposé. Il est fixé au boîtier à l'aide de petits crochets [5].

Cet appareil compact et alimenté par batterie fonctionne parfaitement, mais il est toujours possible d'améliorer ce montage. Par exemple, sur le schéma et le circuit imprimé, on voit le connecteur (J6) prévu pour monter un LCD graphique de faible puissance, à

UTILISATION

Une pression longue sur le bouton VAL (S6) permet d'entrer dans le menu de configuration et autorise l'utilisateur à :

- Changer l'intervalle de mesure (10 à 60 mn) avec les touches + et –, valider en appuyant sur la touche VAL.
- Activer le bargraphe pendant la nuit (oui/non).
- Sélectionner la valeur indiquée par les LED (dernière mesure, IQA/NQI, moyenne de la dernière heure).
- Sélectionner l'indice à afficher (IQA/NQI).

Appuyer sur le bouton VAL fait défiler les différents écrans. Le caractère sur l'afficheur LCD avant la valeur indiquera la nature de la donnée affichée :

L : *Last*, dernière valeur mesurée
H : *Hour*, valeur moyenne de la dernière heure
Q : *Quality*, indice de qualité de l'air, IQA ou NQI (selon le réglage choisi)
T : température en degrés centigrades
H : humidité (en %)
I : *Instant*, l'affichage est actualisé toutes les secondes

Le bargraphe affichera la valeur spécifiée dans le menu, sauf pour la valeur instantanée. Dans ce cas, le bargraphe montre l'indice de la dernière valeur reçue du capteur, qui est actualisé toutes les secondes.
L'écran de valeur instantanée contient sur la deuxième ligne la valeur du nombre de particules renvoyée par le capteur, puis la valeur calculée de l'indice de qualité de l'air précédé de la lettre Q.
Un écran affiche également les valeurs minimale et maximale de l'indice au cours des dernières 24 heures.

128 × 64 pixels [8], mais je n'ai pas testé cette option et elle n'est pas prise en charge dans la version actuelle du microprogramme. Vous pouvez omettre J6 et les condensateurs adjacents sur le schéma.

Malheureusement, la pollution de l'air reste un problème d'avenir, pas seulement dans les grandes villes d'Extrême-Orient. Il est bon d'avoir un moniteur de PM2.5 à portée de main pour surveiller la qualité de l'air, mais le plus inquiétant c'est d'en avoir besoin.

191215-04

Contributeurs
Idée, conception, rédaction et illustrations : **Laurent Labbe**
Schéma : **Patrick Wielders**
Rédaction : **Luc Lemmens, CJ Abate**
Maquette : **Giel Dols**
Traduction : **Nicolas Bishop**

Des questions, des commentaires ?
Vous avez des questions ou des commentaires sur cet article. Envoyez un courrier électronique à l'auteur (laurent.elektor@gmail.com) ou contactez Elektor (redaction@elektor.fr).

PRODUITS CONNEXES

Kit Velleman Earth Listener
www.elektor.fr/velleman-earth-listener-kit



LIENS

[1] **Indice de qualité de l'air** : https://fr.wikipedia.org/wiki/Indice_de_qualit%C3%A9_de_l%27air
[2] **Calcul de l'indice de qualité de l'air (Nowcast)** : https://en.wikipedia.org/wiki/NowCast_(air_quality_index)
[3] **Formule de calcul du NQI** : www.epa.gov/airnow/faq/Nowcast-formula.pptx
[4] **Boîtier** : www.tinkercad.com/things/gznm3a4WifP-new-boitier-dust-11
[5] **Support de panneau solaire** : www.tinkercad.com/things/b3mohI2DzIa-copy-of-copy-of-boitier-dust-6
[6] **Chargeur d'amorçage pour PIC** : www.etc.ugal.ro/cchiculita/software/picbootloader.htm
[7] **Afficheur alphanumérique** : www.buydisplay.com/character-display/character-display-panel?interface=461&resolution=135
[8] **Afficheur graphique** : www.buydisplay.com/1-4-inch-graphic-128x64-lcd-module-serial-spi-st7565-black-on-white
[9] **La page Elektor Labs de ce projet** : www.elektormagazine.fr/labs/portable-display-pm25-1-1
[10] **Fiche technique du PMS7003** : www.pdf-archive.com/2017/04/12/plantower-pms-7003-sensor-data-sheet/plantower-pms-7003-sensor-data-sheet.pdf

sur le vif

Le futur était meilleur dans le passé

Ilse Joostens (Belgique)

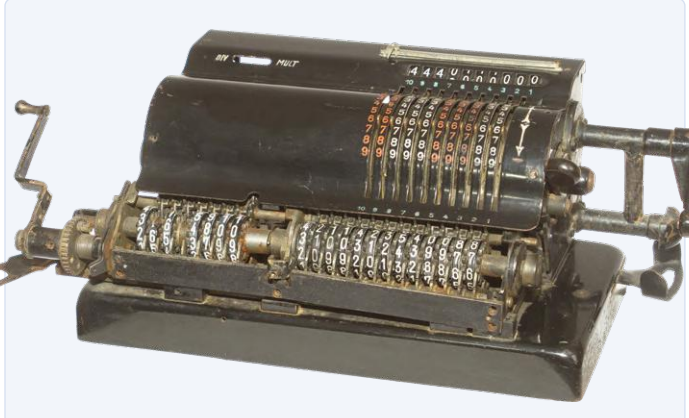
Le soleil était haut. L'air brûlant tournoyait au-dessus du tarmac et brouillait les contours. Accablé de chaleur, Andrew regardait le ballet des avions de ligne par-delà la baie du salon d'affaires. Sans vraiment les voir. De temps à autre, il épongeait les gouttes de sueur qui perlaient de son front. Alors qu'il plongeait sa main dans une poche à la recherche d'un nouveau mouchoir, il remarqua un homme d'affaires japonais assis à quelque distance de lui. Deux énormes valises reposaient à ses pieds. L'homme semblait engagé dans une discussion animée avec sa montre. Perplexe, hypnotisé, Andrew le fixa. Soudain l'homme mit fin à son étrange dialogue et se leva. Incapable de résister à la curiosité, Andrew décida de l'interroger.

Quand (un autre) Napoléon en avait sous le chapeau

Oh là, sommes-nous dans un magazine littéraire ou technique ? vous demandez-vous peut-être. Oui, vous avez raison, pardon pour cette foucade littéraire, je voulais simplement vous raconter la blague du Japonais et de ses deux valises. Blague que je vais donc terminer dans un style plus adapté à ce genre d'histoire :
L'Américain dit au Japonais : « Que faisiez-vous avec ce truc ? » « C'est le dernier gadget en date de Casio, répond le Japonais. Du tout-en-un. Il fait horloge atomique, stylo laser, appareil photo, il permet de passer des appels vidéo, d'accéder à l'internet, d'envoyer des fax, de gérer un agenda, et on peut même nager avec parce qu'il est étanche. » « Fantastique ! dit l'Américain. Je vous l'achète. Combien il coûte ? » « Dix mille dollars », répond le Japonais. « Marché conclu ! » dit l'Américain, qui sort son portefeuille, paie le Japonais, puis s'empresse de passer sa nouvelle acquisition autour du poignet. « Hé, vous oubliez vos valises ! crie-t-il à l'adresse du Japonais qui s'en va déjà. « Pas du tout, répond celui-ci, ce sont les vôtres maintenant, parce qu'elles font partie de la montre ! »
Lorsque j'ai entendu cette blague pour la première fois, les téléphones GSM n'existaient que depuis quelques années, la connexion à l'internet démarrait par les hululements envoûtants d'un modem, et la navigation par ligne commutée qui s'ensuivait était aussi coûteuse que désespérément lente. Jamais à l'époque je n'aurais imaginé que le gadget du Japonais puisse un jour prendre la forme d'un smartphone et devienne un objet de la vie courante. Prendre des photos, filmer, surfer sur l'internet, regarder des vidéos, gérer son agenda, tout cela se fait aujourd'hui sans effort, d'un seul doigt (Dieu merci, on nous a épargné l'envoi des fax). La montre intelligente est elle aussi devenue réalité, et certains smartphones,



La Curta et la Clover 2001. (Photo : Ilse Joostens)



Une calculatrice mécanique comme vous n'en avez jamais rêvé.

surtout en Asie, sont étanches (avouez-le, vous aussi avez un jour rêvé de prendre un appel sous la douche.)
Les smartphones me rappellent l'Appendiscoop, un émetteur-récepteur inventé par Napoléon, le détective d'une série flamande pour enfants appelée Merlina et diffusée dans les années 1980. L'Appendiscoop se portait comme une amulette et servait à transmettre des images et du son. D'autres inventions de Napoléon sont plus ou moins devenues réalité, comme les lunettes caméra, Evarist (l'ordinateur qui a réponse à tout), le Contradecibel (un appareil qui supprime les ondes sonores), et l'Holomat (un appareil photo 3D).
Un des objets qui m'a le plus intrigué lorsque j'étais enfant est la calculatrice que m'offrit mon père pour la Saint-Nicolas. Il s'agissait d'une Clover 2001, et sans doute était-elle à la fois dotée d'excellentes piles et très peu énergivore car, aussi incroyable que cela puisse paraître, près de quarante ans plus tard il m'arrive encore de l'utiliser, même si l'affichage est devenu très faible. Dans les années 60 et 70, les employés de la banque où travaillait mon père utilisaient encore des calculatrices mécaniques, machines complexes et coûteuses qui de surcroît étaient souvent lourdes et encombrantes. Parmi les plus petites de ces calculatrices mécaniques, l'une est devenue un objet de collection très prisé : la Curta. Surnommée « moulin à poivre » ou « grenade à maths » en raison de sa ressemblance avec ces deux objets, la Curta a été utilisée par les pilotes de rallye jusque dans les années 80. Lorsque vous comparez cette stupéfiante mécanique de précision avec les calculettes offertes aujourd'hui comme objets promotionnels, le contraste technologique est vraiment saisissant.



Téléphoner sous la douche : comment l'humanité a-t-elle pu survivre jusque-là sans cette possibilité ?

Madame Soleil et moi

Le futur tel qu'on l'imaginait en l'an 2000 était bien meilleur que ce qu'il est finalement devenu, mais même si l'Avenir a souvent ricané dans le dos de ceux qui s'aventuraient à jouer les augures, je ne peux m'empêcher à mon tour de faire quelques prédictions. Connaissez-vous cette publicité de Caltex ? Une vieille dame entre dans un supermarché en tirant un cabas à roulettes grinçant bruyamment. Serviable, un vendeur s'empare d'une burette d'huile et propose à la cliente de lubrifier les roues gémissantes. Et tandis qu'il s'affaire, on voit la vieille dame s'éloigner à petits pas vers un rayon en faisant entendre le même couinement. Cette sorte de parabole sur l'usure anatomique me fait croire à une révolution prochaine dans le domaine des implants et des prothèses – à un humain bionique, pourrait-on dire. Je m'attends de même à une évolution radicale dans le domaine des organes imprimés en 3D et des techniques chirurgicales non-invasives.
Je vois aussi une intelligence artificielle prendre de plus en plus de place dans nos vies – au risque de nous rapprocher du 1984 de Georges Orwell – et prévois de même un avenir brillant pour les petits réacteurs nucléaires de type TWR (réacteurs à onde de combustion). Un vol habité vers Mars sur les traces de l'astromobile Perseverance me semble également vraisemblable.
Et vous, comment voyez-vous l'avenir ? Polissez votre boule de cristal et faites-le-nous savoir !

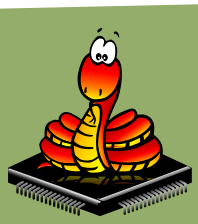
(210185-04 - VF : Hervé Moreau)

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

Contributeurs
Texte : **Ilse Joostens**
Rédaction : **Eric Bogers**
Traduction : **Hervé Moreau**
Maquette : **Giel Dols**

LIENS

[1] **Le son des modems RTC** : www.youtube.com/watch?v=gsNaR6FRuO0
[2] **Merlina** : https://nl.wikipedia.org/wiki/Merlina
[3] **Calculatrice Curta** : www.vcalc.net/cu.htm
[4] **Calculatrice Curta** : https://satadorus.eu/x_ite/yacs_2_0/yacs_2_0.html
[5] **Windows XP est-il obsolète ?** : https://youtu.be/zbsq4-KwnE4?t=51
[6] **Publicité Caltex** : https://youtu.be/aLmWU7cCW10



MicroPython
pour l'ESP32 et ses copains
Partie 1 : installation et premiers programmes

Günter Spanner (Allemagne)

Python remplace de plus en plus le C comme langage de programmation de prédilection, et cette tendance s'observe désormais aussi dans le monde des microcontrôleurs. En prenant l'ESP32 comme exemple, nous allons voir dans cet article comment programmer un microcontrôleur moderne en MicroPython.

Matériel nécessaire

1. Petite plaque d'expérimentation*
2. ESP32-PICO-KIT V4*
3. LED rouge
4. Résistance de 150 Ω
5. Module d'affichage compatible SSD1306*
6. Fils

* Disponible dans la boutique d'Elektor

Ces dernières années, Python a connu un énorme regain de popularité, notamment en raison de la disponibilité de systèmes monocartes tels que le Raspberry Pi. Mais Python a également trouvé des applications plus larges dans d'autres domaines tels que l'intelligence artificielle et l'apprentissage automatique. Il semble donc naturel d'examiner comment employer Python (ou la variante MicroPython en tout cas) dans des montages à microcontrôleurs. Cet article se penche sur les bases de MicroPython, et en particulier sur les instructions les plus importantes et les bibliothèques disponibles. Nous utiliserons un microcontrôleur ESP32 pour réaliser quelques petites applications : le microcontrôleur sera programmé avec un micrologiciel qui interprète les commandes Python. Le programme Python lui-même est écrit dans un environnement de développement sur PC, et les commandes Python peuvent être envoyées du PC au microcontrôleur, soit sous forme d'un programme entier, soit une par une. Cela peut se faire soit par USB, soit par un réseau sans fil. Examinons ces aspects un par un.

Environnements de programmation et de développement

Contrairement à l'écosystème Arduino, MicroPython peut fonctionner avec pas mal d'environnements de développement intégrés, ou EDI. Actuellement, les deux environnements les plus utilisés sont :

1. µPyCraft
2. Thonny

On peut aussi envisager d'installer Anaconda Navigator, surtout utilisé pour la programmation de microcontrôleurs dans le domaine de l'intelligence artificielle. Chaque approche présente des avantages et des inconvénients particuliers. Par exemple, l'EDI µPyCraft n'offre qu'une interface utilisateur relativement peu sophistiquée, avec des éléments graphiques simples qui rappellent les systèmes d'exploitation orientés texte.

Thonny, quant à lui, possède une interface utilisateur graphique complète dans le style de Windows (**fig. 1**). Cet EDI est très populaire dans la communauté des makers, notamment parce qu'il est disponible pour le Raspberry Pi avec le système d'exploitation Raspbian. De nombreux utilisateurs de Raspberry Pi sont donc déjà très familiers avec Thonny. Thonny tourne sur tous les principaux systèmes d'exploitation, notamment Windows, Mac OS X et Ubuntu Linux, et peut être téléchargé gratuitement sur l'internet [1].
Avant d'utiliser Thonny, il est nécessaire que Python 3 soit installé sur la machine qui sera utilisée pour la programmation. Si ce n'est pas encore le cas, vous pouvez trouver les instructions d'installation sur le site web correspondant [2].
Ensuite, installez Thonny lui-même depuis [1]. Sélectionnez d'abord le système d'exploitation approprié en haut à droite de la page. L'exécution du fichier téléchargé lancera le processus d'installation habituel. L'EDI est maintenant prêt pour la création de notre première application Python.

Installation de l'interpréteur

L'étape suivante consiste à installer le micrologiciel MicroPython lui-même. On peut l'obtenir sur le site officiel de MicroPython [3], avec toute une liste de microcontrôleurs disponibles. Dans cet article, nous nous intéresserons plus particulièrement à l'ESP32, et nous devons donc télécharger la dernière version compatible avec ce type de microcontrôleur. Sous *Espressif ESP-based boards*, cliquez sur l'image dont la légende est *Generic ESP32 module*. La page qui s'ouvre alors propose un certain nombre de variantes de micrologiciel. Notez que la version la plus récente du micrologiciel est souvent qualifiée « d'instable ». Cette option est plutôt destinée aux développeurs du micrologiciel de l'interpréteur lui-même et à d'autres personnes de nature aventureuse. Ceux d'entre nous qui préfèrent un système plus stable choisiront la version la plus récente qui ne soit pas qualifiée « d'instable », par exemple :

GENERIC: esp32-idf3-20200902-v1.13.bin

Cliquez sur le lien approprié et le micrologiciel sera téléchargé.

Maintenant connectez la carte à microcontrôleur (par ex. un ESP32-PICO-KIT, voir l'encadré **Matériel nécessaire**) au PC sur l'USB, puis lancez l'EDI Thonny. Ici, il faut choisir *Sélectionner l'interpréteur* dans le menu *Exécuter*, ce qui ouvre la fenêtre des *Options de Thonny* (**fig. 2**). Sous l'onglet *Interpréteur*, vous pouvez choisir parmi une série d'options, dont certaines sont spécifiques à l'ESP32.
Sous *Port*, sélectionnez le port USB auquel l'ESP32 est connecté. Vous pouvez également sélectionner l'option *Essayer de détecter le port automatiquement*. Toutefois, cette option ne fonctionne pas toujours de manière fiable.
En cliquant dans la case sous *Micrologiciel*, le programme d'installation démarre. Le port n'est pas configuré automatiquement et doit être saisi à nouveau. Ensuite, naviguez vers le micrologiciel que nous venons de charger. En cliquant sur *Installer*, le processus d'installation démarre. Lorsque le processus est terminé, la fenêtre reste ouverte : fermez-la et vous êtes prêt pour commencer à programmer votre ESP32 en MicroPython.

Bibliothèques

Travailler avec Python signifie recourir à des bibliothèques. Écrire chaque programme à partir de zéro est, pour le moins, très inefficace, et l'immense succès de Python est en grande partie dû à la quantité de bibliothèques disponibles pour ce langage.

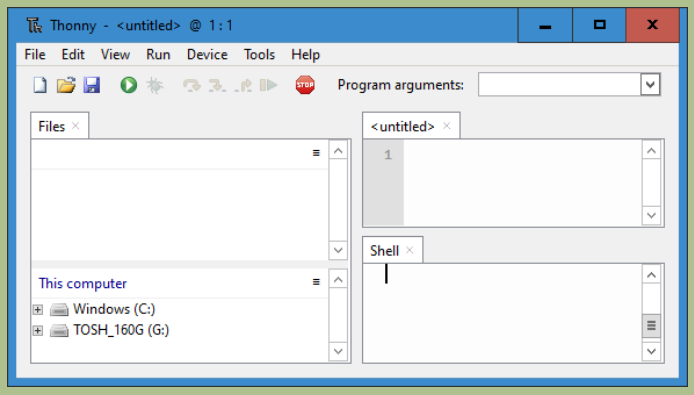


Figure 1. L'EDI Thonny.

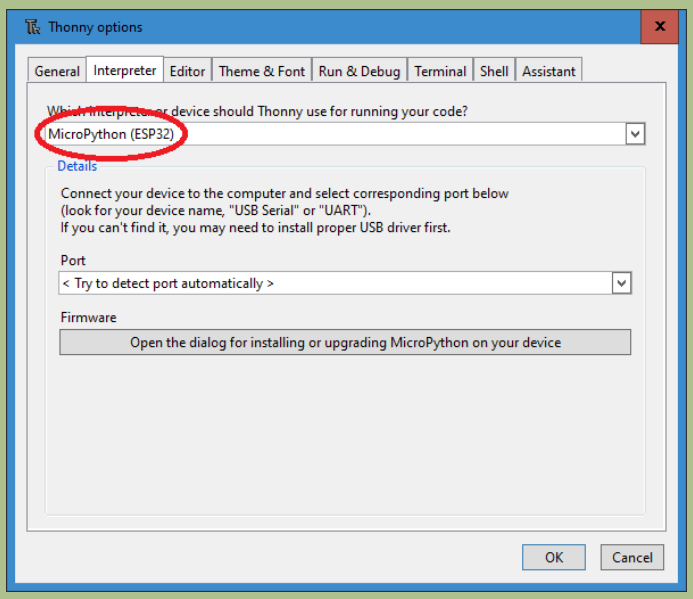


Figure 2. Options d'installation du microprogramme Thonny.

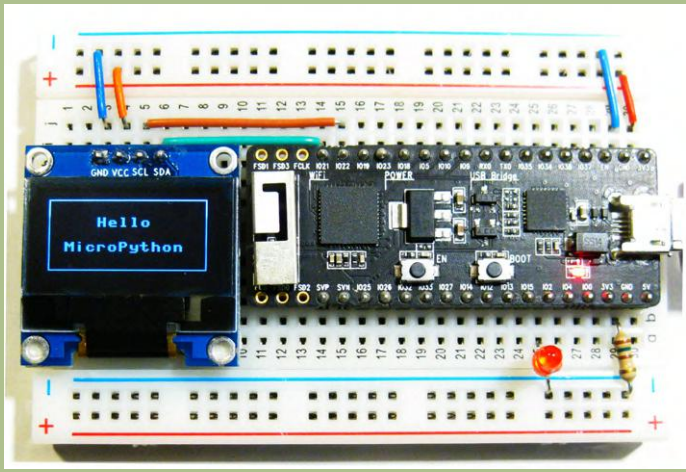


Figure 3. Microcontrôleur ESP32 avec un écran OLED et une LED connectée à la broche 2.

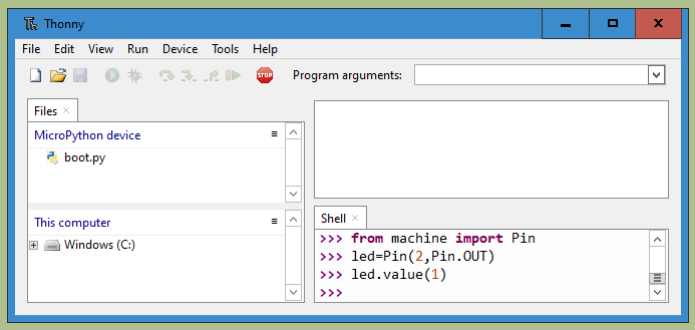


Figure 4. Commutation de l'état d'un port d'E/S à l'aide de la console REPL.

Malheureusement, toutes les bibliothèques ne peuvent pas être exploitées avec les ressources limitées d'un microcontrôleur, et une sélection spéciale de bibliothèques a donc été créée pour être utilisée avec MicroPython. Un grand nombre d'entre elles sont disponibles en standard dans le téléchargement de l'EDI Thonny. Les deux bibliothèques standard les plus importantes dans MicroPython sont *machine* et *time*. L'instruction `import` est utilisée pour rendre les bibliothèques disponibles pour une utilisation sur le microcontrôleur. Les lignes suivantes (entre autres) rendent une fonction de bibliothèque accessible :

```
> import module
> from module import name
```

Dans le premier cas, le module entier est importé ; dans le second, seules les fonctions spécifiées le sont.
Le module *machine* contient des fonctions spécifiques au matériel d'un microcontrôleur particulier. Ces fonctions permettent d'accéder directement et sans limites aux composants matériels et de les commander, notamment l'unité centrale, les temporisateurs, les bus et les broches d'entrée/sortie. Sachez qu'une mauvaise utilisation de ce module peut provoquer des erreurs, des plantages, voire au pire endommager le matériel.
La classe `Pin` est l'une des parties les plus importantes du module *machine*. Un objet `Pin` commande une broche d'entrée/sortie et, par convention, un objet `Pin` est affecté à une broche physique sur le microcontrôleur. L'objet permet de piloter les niveaux de sortie et de lire les niveaux d'entrée.
La classe `Pin` fournit des méthodes pour définir le mode de la broche. Par exemple, `IN` et `OUT` peuvent être utilisés pour configurer respectivement une broche en entrée ou sortie.
Le module *Time* fournit une série de fonctions liées au temps. La classe `sleep` de ce module met en pause l'exécution du programme en cours pendant le nombre de secondes spécifié. L'argument peut même être une valeur à virgule flottante, ce qui permet de définir des délais exacts d'une fraction de seconde. Les commandes

```
from machine import Pin
from time import sleep
```

rendent disponibles les fonctions `Pin` et `Sleep`. Avec la première, nous pouvons dialoguer individuellement avec les broches des ports du microcontrôleur, et avec la seconde nous pouvons implémenter une fonction basée sur le temps. En utilisant la commande

```
led = Pin(2, Pin.OUT)
```

nous pouvons créer un objet `led` associé à la broche d'E/S numéro 2, et configurer cette broche en sortie. Nous pouvons maintenant passer diverses valeurs à cet objet. Par exemple, si nous exécutons la commande

```
led.value(1)
```

alors la valeur 1 sera écrite dans l'objet. Cela signifie que la broche d'E/S associée, la broche 2, sera placée à un niveau de tension haut. Dans le cas de l'ESP32, il s'agit de 3,3 V.

La console REPL

Il est possible de surveiller l'état d'une broche de port en y connectant simplement une LED via une résistance de limitation de courant. (La **figure 3** montre comment faire, et comment aller plus loin en connectant un écran OLED : nous y reviendrons plus tard.) Le port, et donc la LED, peuvent être contrôlés très simplement

à l'aide de ce que l'on appelle la « console REPL ». REPL est l'abréviation de « Read Evaluate Print Loop » (lecture-évaluation-affichage), une méthode d'entrée interactive de MicroPython qui permet d'exécuter des commandes directement sur l'ESP32. La console REPL nous offre donc un moyen très simple de tester des commandes et d'exécuter des programmes.
La console REPL est appelée « *Console* » dans Thonny et se trouve en bas de la fenêtre principale. Vous pouvez saisir les commandes des extraits ci-dessus directement dans cette zone (**fig. 4**).
Après avoir entré la dernière des commandes ci-dessus, la LED doit s'allumer. La commande `led.value(0)` permet de l'éteindre.
La console REPL possède quelques fonctions intéressantes qui facilitent le travail avec MicroPython. Par exemple, les lignes de commande précédentes sont stockées, et les touches de déplacement vers le haut et le bas permettent de rappeler les commandes précédemment saisies si nécessaire.
Une autre fonction pratique est la complétion par tabulation. En appuyant sur la touche « Tab » du clavier, on tente automatiquement de compléter un mot partiellement saisi. Cette fonction permet même d'obtenir des informations sur les fonctions et les méthodes disponibles dans un module ou applicables à un objet.
Par exemple, en entrant « ma » et en appuyant sur la touche « Tab », on obtient automatiquement « machine » (en supposant que le module *machine* a déjà été importé comme décrit ci-dessus). En appuyant sur la touche point (« . ») puis sur la touche « Tab », vous obtiendrez une liste complète de toutes les fonctions disponibles dans le module *machine* (**fig. 5**). Dans de nombreux cas, cette fonction rend inutile la recherche de commandes, d'objets et de méthodes dans la documentation.

Accès sans fil à l'aide de WebREPL

L'un des principaux avantages de l'ESP32 est son excellente prise en charge de la connectivité sans fil. Quoi de plus naturel, donc, que de rendre la console REPL disponible via une telle connexion ? Pour ce faire, nous pouvons utiliser l'interface WebREPL.
La première étape pour rendre WebREPL opérationnel est de s'assurer qu'il est installé et activé sur l'ESP32. Par défaut, WebREPL n'est pas activé et doit être activé en envoyant une fois la commande suivante :

```
import webrepl_setup
```

sur le port série. Vous aurez la possibilité d'activer ou de désactiver la fonction, et de définir un mot de passe. Après la configuration, il faut redémarrer l'ESP32.
Pour utiliser WebREPL sur un réseau sans fil, l'ESP32 doit d'abord être connecté à ce réseau. Pour ce faire, envoyez les commandes suivantes à la console série REPL.

```
import network
wlan = network.WLAN(network.STA_IF)
wlan.active(True)
wlan.connect('ssid', 'password')
```

Les champs `ssid` et `password` doivent bien sûr être remplacés par les informations d'identification de votre réseau sans fil local. La commande

```
wlan.ifconfig()
```

affichera alors les paramètres de l'adresse IP que l'ESP32 utilise pour communiquer avec le réseau (**fig. 6**). Les commandes

```
import webrepl
webrepl.start()
```

vont maintenant activer le client WebREPL. Dans un navigateur web, vous pouvez aller à l'adresse

```
http://micropython.org/webrepl/#xxx.xxx.xxx.xxx:8266
```

Vous pouvez ensuite utiliser l'onglet *Connecter* pour vous connecter à l'ESP32 en utilisant le mot de passe défini précédemment.
Lorsque WebREPL est lancé, vous pouvez constater que la console est en mode « raw REPL ». Ce mode permet la saisie directe de commandes par copier-coller. Dans ce contexte, vous pouvez appuyer sur CTRL-B pour repasser en mode normal et à la saisie ordinaire des commandes.
Nous sommes maintenant en mesure de commander l'ESP32

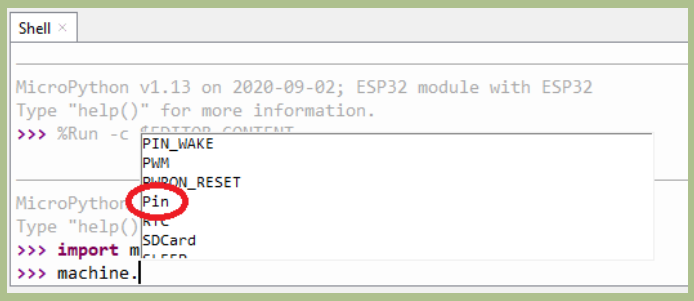


Figure 5. Autocomplétion.

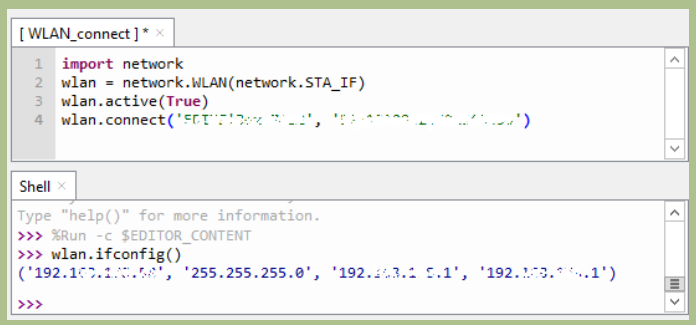


Figure 6. L'ESP32 est connecté au réseau sans fil local.

totalement sans fil. Les commandes simples pour commuter des sorties nous permettent déjà de mettre en œuvre des fonctions de base de domotique (**fig. 7**). Si un système de fichiers approprié est installé sur l'ESP32, on peut aussi effectuer des mises à jour logicielles sans fil, appelées mises à jour OTA (*over-the-air*). Lorsqu'on travaille avec WebREPL, il faut se rappeler qu'il s'agit d'une fonction expérimentale et qu'il ne faut pas s'attendre à ce qu'elle se comporte de manière absolument fiable dans toutes les situations.

Commande par programme

La console REPL ou WebREPL est la plus adaptée pour tester des idées. Pour l'écriture de programmes conventionnels, le panneau de l'éditeur (au-dessus de la console REPL dans la fenêtre Thonny) est plus approprié. Nous pouvons par ex. y préparer le programme décrit ci-dessous, conçu pour la commande automatique de l'éclairage nocturne et celui des cages d'escalier (**fig. 8**).
Une fois le code saisi et lancé à l'aide de l'icône de démarrage (flèche blanche dans un cercle vert), vous serez invité à enregistrer le programme. Choisissez ici l'option *MicroPython device* et entrez un nom de programme (par ex. *Automatic_LED*) et confirmez la sauvegarde. La LED va maintenant s'allumer pendant trois secondes, puis s'éteindre automatiquement. Si vous appuyez à nouveau sur le bouton de démarrage, le programme peut être relancé immédiatement.
Pour transformer ce programme en une démo classique de LED clignotante, il suffit d'ajouter une instruction `while`. Celle-ci fera en sorte qu'une commande ou un bloc de commandes données soient exécutées de manière répétée. Le cas particulier de `while True:` crée une boucle qui se répète sans fin : dans ce cas, cela signifie que la LED clignotera en continu.

```
from machine import Pin
from time import sleep

led = Pin(2, Pin.OUT)

while True:
  led.on()
  sleep(0.5)
  led.off()
  sleep(0.5)
```

Remarquez ici que l'instruction `while` doit se terminer par un deux-points. Le bloc de commandes suivant est, conformément à la convention Python standard, indenté par un nombre fixe d'espaces. Thonny fournit cette indentation automatiquement : dès que vous saisissez un deux-points suivi de la touche Entrée, le curseur apparaît sur la ligne suivante indentée d'un pas. Et bien sûr, l'éditeur de programmes offre également la fonction de complétion par tabulation. La combinaison CTRL-C permet d'arrêter un programme en cours d'exécution.

MicroPython en quelques mots

Bien que l'essence même de MicroPython soit sa collection de bibliothèques, une bonne maîtrise des commandes de base est nécessaire pour comprendre les programmes des autres et pour écrire les siens. Nous examinerons brièvement les commandes les plus puissantes de MicroPython.
Les commentaires simples sont introduits par le symbole '#'. Le commentaire s'étend du symbole '#' à la fin de la ligne.

```
>>> print("hello ESP32")          # this is a comment
  hello ESP32
```

Les commentaires sont ignorés pendant l'exécution du programme ; ils ne sont là que pour fournir des informations au programmeur. La commande `print()` utilisée ici permet d'afficher des informations

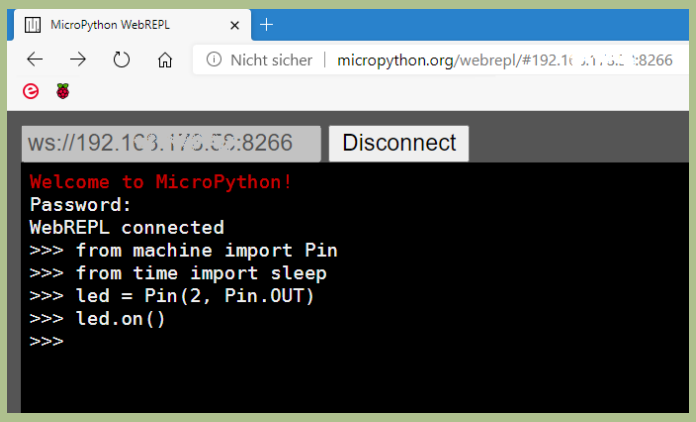


Figure 7. WebREPL dans le navigateur.

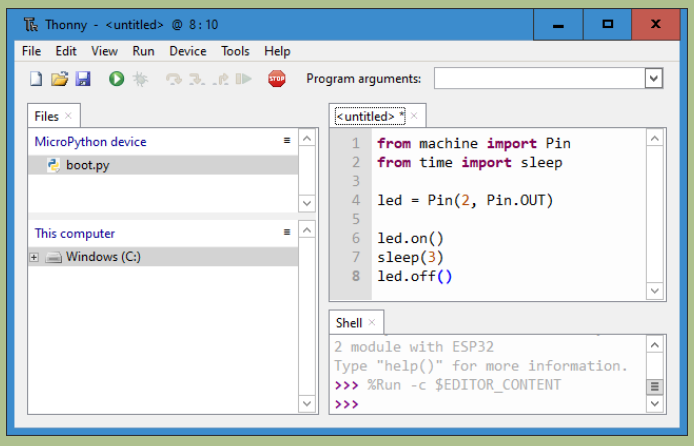


Figure 8. LED automatique.

Produits

- **Livre en anglais « MicroPython for microcontrolers »**
  www.elektor.fr/micropython-for-microcontrollers
- **ESP32-PICO-KIT V4**
  www.elektor.fr/esp32-pico-kit-v4
- **Plaque d'expérimentation**
  www.elektor.fr/breadboard-830-tie-points
- **Module d'affichage compatible SSD1306**
  www.elektor.fr/blue-0-96-oled-display-i2c-4-pin

dans la console, qu'elles soient textuelles ou numériques ; vous pouvez aussi appeler `print()` directement depuis la fenêtre du terminal.
Comme nous l'avons vu plus haut dans l'exemple de la LED clignotante, les instructions peuvent être regroupées en blocs identifiés par leur indentation. Cela signifie que les accolades (« { » et « } ») et autres mécanismes similaires ne sont pas nécessaires. L'avantage de ce principe est que l'on est plus ou moins forcé d'adopter un style de programmation structuré.
Il est très facile de créer une variable dans MicroPython, et en particulier il n'est pas nécessaire de spécifier son type. On peut aussi utiliser directement les variables dans la console comme suit :

```
>>> a=17
>>> b=12
>>> print(a*b)

          204
```

Dans MicroPython, les opérateurs arithmétiques ont leurs interprétations mathématiques conventionnelles. Outre l'addition, la soustraction, la multiplication et la division, nous disposons également de l'opérateur `//` pour la division d'entiers, de `%` pour le modulo (ou le reste de la division) et de `**` pour l'exponentiation.
Les instructions habituelles de branchement et de bouclage existent aussi dans MicroPython. Le branchement est pris en charge par le mot clé `if`, suivi d'une condition ; si la condition est vraie, les instructions suivantes sont exécutées. Un mot-clé `else` peut suivre, introduisant un ensemble d'instructions à exécuter à la place si la condition est fausse. Par exemple :

```
if True:
  # block 01
  print ("True")
else:
  # block 02
  print ("False")
```

Les boucles permettent d'exécuter des instructions de manière répétée. Un ensemble d'instructions est exécuté tant qu'une condition spécifiée est remplie. Deux variantes sont disponibles :

- Boucles « while »
- Boucles « for »

Ainsi, une façon d'imprimer les chiffres de 1 à 9 sur la console serait d'utiliser une boucle `while` comme suit :

```
number=1
while number<10:
  print(number)
  number=number+1
```

L'ensemble des instructions à répéter est indiqué par l'indentation. Cette tâche peut également être effectuée en utilisant une boucle `for` comme suit :

```
for number in range(1, 10):
  print(number)
```

Signal d'alarme automatique

Nous en savons maintenant assez pour écrire notre premier programme d'application pratique. Il s'agira d'une balise SOS automatique qui pourrait être utilisée pour signaler une urgence en voile ou en escalade.

```
from machine import Pin
from time import sleep
led=Pin(2,Pin.OUT)

while True:
  for n in range(3):
    led.value(1)
    sleep(.1)
    led.value(0)
    sleep(.5)
  sleep(1)
  for n in range(3):
    led.value(1)
    sleep(.4)
    led.value(0)
    sleep(.5)
  sleep(1)
  for n in range(3):
    led.value(1)
    sleep(.1)
    led.value(0)
    sleep(.5)
  sleep(2)
```

Panneau OLED : un petit écran d'affichage

Il est sans doute possible d'utiliser une seule LED pour communiquer des informations utiles, en utilisant par ex. le code morse comme dans l'application de balise SOS ci-dessus. Une approche bien plus moderne consiste bien sûr à afficher des données sur un panneau OLED. De nombreux panneaux de ce type utilisent le pilote d'affichage SSD1306. Ici, nous utilisons un écran d'une diagonale de seulement 0,96 pouces (environ 2,5 cm) et d'une résolution de 128×64 pixels.
MicroPython est livré en standard avec une bibliothèque pour les écrans à base de SSD1306. Elle permet d'afficher des données textuelles et numériques ainsi que de créer des graphiques simples. Les modules les plus simples à base de SSD1306 ont une interface

qui n'utilise que quatre broches, ce qui suffit pour piloter l'écran avec le protocole de bus I2C. La connexion à l'écran est illustrée à la **figure 3**, et les connexions nécessaires sont présentées dans le tableau ci-dessous.

Broche OLED	Broche ESP32
VDD	3V3
GND	GND
SCK	GPIO 22
SDA	GPIO 21

Le script présenté dans le listage 1 envoie un texte à l'écran et dessine quelques graphiques simples comme ce cadre autour du texte (fig. 3).
La bibliothèque correspondante est disponible sous forme de paquetage standard (*ssd1306.py* dans l'archive de téléchargement [5]) et peut être téléchargée séparément sur la carte. Les broches utilisées pour l'interface I2C sont déclarées comme suit :

```
i2c = I2C (-1, scl = Pin (22), sda = Pin (21))
```

Le premier paramètre, « -1 », indique que le module utilisé ne dispose pas d'une broche de réinitialisation ou d'interruption. Le nombre de pixels horizontaux et verticaux du module est spécifié à l'aide de la commande :

```
oled = SSD1306_I2C(128, 64, i2c)
```

L'écran est maintenant prêt. La fonction `text()` sert à écrire des informations dans le tampon d'affichage, et l'affichage lui-même est mis à jour avec la méthode `show()`. La fonction `text()` accepte les arguments suivants :

- Le message (une chaîne de caractères)
- Les coordonnées x et y du texte en pixels
- En option la couleur du texte : 0 pour le noir (non allumé) et 1 pour le blanc (allumé)

La méthode `show()` permet de rendre les modifications visibles à l'écran. La méthode `rect()` permet de dessiner un rectangle à l'écran. Elle accepte les arguments suivants :

- Coordonnées x et y du coin inférieur gauche du rectangle
- Coordonnées x et y du coin supérieur droit du rectangle
- Couleur de pixel : 0 pour le noir et 1 pour le blanc

La commande

```
oled.rect(5, 5, 116, 52, 1)
```

fait donc apparaître un cadre rectangulaire près du bord de l'écran. Avec ces simples commandes, il est possible pour une application d'afficher toutes sortes d'informations, du texte de base aux relevés complexes de capteurs.

Listage 1. Message sur l'écran OLED.

```
from machine import Pin, I2C
from ssd1306 import SSD1306_I2C

i2c=I2C(-1,scl=Pin(22),sda=Pin(21))
oled = SSD1306_I2C(128, 64, i2c)
lin_hight = 9
col_width = 8

oled.fill(0)
oled.text("Hello", 5*col_width, 2*lin_hight)
oled.text("MicroPython", 2*col_width,
4*lin_hight)

oled.rect(5, 5, 116, 52, 1)
oled.show()
```

Perspectives

MicroPython est un langage de programmation moderne et puissant. De plus, ses bibliothèques permettent de réaliser rapidement et facilement des projets complexes. Dans cet article, nous avons montré comment installer un EDI approprié et créer quelques applications simples. Dans la deuxième partie de cette série, nous examinerons d'autres aspects de MicroPython et, à titre de démonstration pratique, nous donnerons un exemple de pilotage d'un panneau d'affichage matriciel à LED de grand format. Vous trouverez de plus amples informations sur MicroPython, sur le microcontrôleur ESP32 et sur les exemples présentés ici dans le livre *MicroPython for Microcontrollers* [4].

(210179-04)

Contributeurs
Texte et illustrations : **Günter Spanner**
Rédaction : **Jens Nickel**
Mise en page : **Harmen Heida**
Traduction : **Denis Lafourcade**

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

LIENS

[1] **Thonny** : https://thonny.org/
[2] **Python** : www.python.org/downloads
[3] **MicroPython** : http://micropython.org/download
[4] **G. Spanner, *MicroPython for Microcontrollers*, Elektor 2020** : www.elektor.fr/micropython-for-microcontrollers
[5] **Logiciel** : www.elektormagazine.fr/210179-04

composants à couplage de charge dans les oscilloscopes

Neil Gruending (Canada)

Je parie que le premier capteur à couplage de charges (*Charge Coupled Device, CCD*) qui vous vient à l'esprit est un capteur de caméra CMOS. En plus d'être utilisés dans des millions d'appareils photo, saviez-vous qu'ils ont également été utilisés à d'autres fins ? Nous allons voir comment Tektronix y a eu recours il y a trente ans pour améliorer les performances de ses premiers oscilloscopes à échantillonnage numérique.

Un dispositif à couplage de charges est réalisé comme son nom le laisse entendre : il s'agit d'un réseau de condensateurs à oxyde métallique (MOS) interconnectés par des transistors MOSFET, comme le TDA1022 [1] de Philips illustré à la **figure 1**. Le signal analogique injecté sur la broche 5 est décalé vers la droite lorsque les grilles des MOSFET sont cadencées par deux signaux d'horloge déphasés appliqués aux broches 1 et 4. Ces composants ont été créés à l'origine pour servir de ligne à retard dans les circuits analogiques, mais on s'est vite rendu compte qu'ils pouvaient également être utilisés comme mémoire tampon.

On se souvient des oscilloscopes analogiques traditionnels qui affichaient le signal mesuré sur l'écran d'un tube cathodique, avec un résultat très précis pour les signaux répétitifs. Mais lors de la mesure de signaux très lents ou de transitoires rapides, un affichage clair du signal peut être difficile à obtenir sans un oscilloscope numérique. L'un des premiers oscilloscopes à échantillonnage numérique était le Tektronix 2440. Lors de son développement, les concepteurs ont eu un sérieux problème, car les convertisseurs analogiques-numériques (CA/N) de l'époque n'étaient pas assez rapides pour échantillonner des signaux numériques à grande vitesse. La solution astucieuse trouvée par les ingénieurs de Tektronix a consisté à placer un registre à décalage analogique à entrée rapide et sortie lente (FISO) avant le CA/N. Le FISO était en fait une matrice CCD servant de mémoire tampon où enregistrer les valeurs d'échantillonnage rapide (*fast-in*) pour les présenter plus lentement au traitement en aval (*slow-out*). Dans le Tektronix 2440, le FISO était suffisamment grand pour mettre en mémoire tous les points d'échantillonnage d'une trame.

Cette technique FISO était également utilisée dans les oscilloscopes TDS300 et TDS600 car ils disposaient d'une mémoire d'échantillonnage relativement petite. Malheureusement, il est difficile d'augmenter la mémoire d'échantillonnage sans utiliser des CA/N beaucoup plus rapides et sans stocker les données échantillonnées directement dans une RAM, comme cela a été fait avec la série TDS700, laquelle a connu une extension de la mémoire et des capacités d'échantillonnage, ce qui a également conduit aux affichages numériques au phosphore qui imitent plus fidèlement les affichages analogiques. Il n'y a plus guère d'endroits où rencontrer cette technologie, à moins que vous ne possédiez l'un de ces oscilloscopes à l'ancienne. Si ce n'est pas le cas, pourquoi ne pas en acquérir un… ou deux ! Pour améliorer vos connaissances sur les capteurs CCD, allez faire un tour ici : https://vu.fr/88Mz ou https://vu.fr/woLc.

(210181-04 – VF : Helmut Müller)

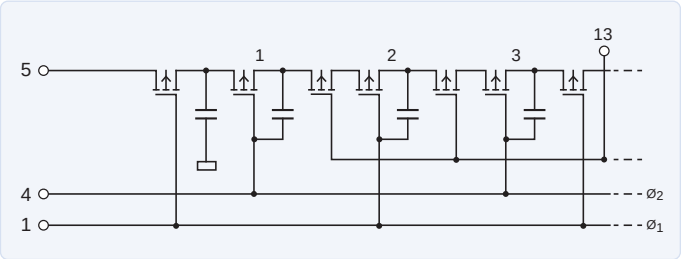


Figure 1. La ligne à retard MOS du TDA1022 [1].



Figure 2. L'oscilloscope à échantillonnage numérique Tektronix 2400 à deux canaux, 300 MHz, 500 Méch/s, datant de la fin des années 1980. (Source : tekwiki [2])

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

LIENS

[1] **Fiche technique de la ligne à retard TDA1022 pour signaux analogiques** : http://bit.ly/3cUUQAW

[2] **Page Tektronix 2440 sur tekwiki** : https://w140.com/tekwiki/wiki/2440



ESD

Le destroyer fantôme

Foudroiement spontané des composants

Figure 1. Le tragique accident à l'arrivée du Hindenburg à Lakehurst (image : Sam Shere).

Peter Beil (Allemagne)

La taille de gravure des puces se réduit sans cesse, et elles sont de plus en plus vulnérables aux dommages causés par les décharges électrostatiques. Intéressons-nous à la nature de cette menace bien réelle et aux moyens de l'écarter.

Le voyage transatlantique du dirigeable *Hindenburg* en 1937 s'est achevé par une catastrophe. Au cours de l'approche du pylône d'amarrage de *Lakehurst*, en Amérique, les 200 000 m³ d'hydrogène, qui sustentaient le Hindenburg, ont pris feu (**fig. 1.**). La source d'inflammation n'a jamais été officiellement attribuée, mais de nombreux experts soupçonnent qu'il s'agit d'une décharge d'électricité statique entre la structure du dirigeable et le pylône d'acier. Ce type de phénomène est appelé *ESD* (*ElectroStatic Discharge*), en français décharge électrostatique, mais *DES* n'est pas usité.

Qu'est-ce que l'ESD ?

Il s'agit d'une décharge électrique soudaine entre deux objets ou corps qui, portant des charges électriques différentes, sont portés à des potentiels différents. Cette différence de potentiel (*ddp* en abrégé) ou tension est neutralisée lorsque les deux corps se rapprochent assez pour qu'une décharge se produise, souvent avec une étincelle et un craquement. Une méthode courante pour donner une charge électrique à un objet est de le frotter sur un matériau isolant, c'est l'effet *triboélectrique*. Tout un chacun a remarqué ce

phénomène en se peignant ou brossant les cheveux par temps sec. Marcher sur un tapis en matière synthétique ou quitter le siège d'une voiture communique aussi une charge au corps. Après, en touchant de la main un autre objet conducteur par ex. une poignée de porte (**fig. 2.**) de charge différente, on peut ressentir une décharge au bout du doigt, car les deux charges se neutralisent. Il est probable qu'à l'amarrage du Hindenburg, une étincelle jaillit et enflamma l'hydrogène. La surface de friction sur l'air du zeppelin est énorme. La ddp entre l'enveloppe et le pylône a pu facilement atteindre quelques centaines de milliers de volts et le pic de courant pendant la décharge a pu être assez élevé.

À plus petite échelle, ce phénomène se remarque déjà souvent, par ex. en enlevant un pull en coton porté sur un maillot de corps en fibres synthétiques, on entend distinctement un craquement. Les ddp produites par ce geste atteignent facilement quelques kV. La nuit, avec les yeux habitués à l'obscurité, on perçoit ces minuscules éclairs dès 5 kV. Les chaussures de sport sont presque toujours faites de matériaux synthétiques ; marcher peut entraîner une charge de 15 kV sur sol en plastique dur et atteindre 25 kV sur tapis.

Dans nos activités quotidiennes, nos mouvements produisent des charges électrostatiques. Ce phénomène est particulièrement gênant par faible humidité, par ex. dans les pièces et espaces climatisés ou chauffés. On s'en rend rarement compte, mais le simple fait de se lever d'une chaise et de s'y rasseoir peut produire une différence de potentiel de 15 kV. Pour ressentir une décharge provenant d'un seul point isolé sur le corps, il faut que la ddp dépasse 2 à 3000 V. À plus faible ddp, la décharge passe le plus souvent inaperçue.

En introduisant certains de ces chiffres dans le modèle standard HBM (*Human Body Model*) [1] (voir en français https://cutt.ly/lbKjtUw), on conclut que le modèle électrique équivalent d'un corps humain a une capacité variant de 100 à 300 pF. Dans l'hypothèse où la peau a une résistance moyenne de contact d'1,5 kΩ env., on doit s'attendre à ce qu'une décharge sous une ddp de 15 kV produise une pointe de courant atteignant 10 A durant quelques dizaines de ns.

L'impact sur l'électronique

Tout électronicien amateur ou professionnel qui travaille avec des composants de faible puissance doit prendre garde à ces décharges indésirables. Même si nous n'en sommes pas conscients, les décharges proches du seuil de perception humaine peuvent avoir un effet catastrophique sur les microstructures électroniques. Les manipulations sans précaution ont déjà détruit d'innombrables puces. Toucher de la main un circuit intégré moderne peut entraîner une décharge statique destructrice dans ce composant. À notre échelle macroscopique, elle équivaut à un coup de foudre sur un arbre (**fig. 3.**).

Bien sûr, la ddp n'est pas le seul élément à prendre en compte ; il faut aussi considérer l'énergie dissipée pendant le processus d'annulation de la ddp. L'ESD est un phénomène transitoire qui ne dure que quelques nanosecondes, mais les structures autour d'une zone d'entrée de CI dans la puce ne mesurent que quelques microns. À 15 kV et 150 pF, selon la formule $E = ½C \times V^2$, l'énergie stockée par le condensateur humain est de 17 mJ env. Bien que ce chiffre soit assez faible, la brièveté de la décharge se traduit par une puissance instantanée très élevée, de l'ordre du kW, et une densité de puissance très élevée parcourt les microstructures.



Figure 2. Une décharge électrostatique peut être douloureuse. Méfiez-vous de cette poignée ! (Cette photo et suivantes : Beil & Kaiser)



Figure 3. Un doigt chargé peut avoir le même effet sur un CI qu'un nuage d'orage sur un arbre.

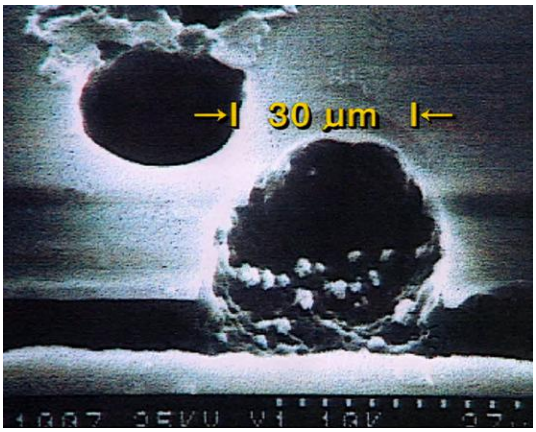


Figure 4. Image au microscope électronique des dommages causés par une ESD.

L'observation au microscope électronique des dommages dus à ces décharges révèle des connexions calcinées, des cratères et restes carbonisés de la micro-explosion produite sur le silicium (**fig. 4.**), rappelant un coup de foudre, à microéchelle. Il faut noter que les CI hautement intégrés où nous gravons désormais des nanostructures, ne sont pas les seuls concernés : les diodes, LED, MOSFET et même les semi-conducteurs de puissance le sont aussi.

Les décharges de « faible » ddp autour de 1 kV que nous ne pouvons pas percevoir nous-mêmes peuvent détruire beaucoup de composants actifs. Un tel dommage passe donc souvent inaperçu. Il est visible si un composant entier tombe en panne, mais si une petite partie d'un CI ou une fonction est affectée, il peut rester longtemps caché.



Figure 5. Lieu de travail équipé de toutes les mesures anti-ESD.

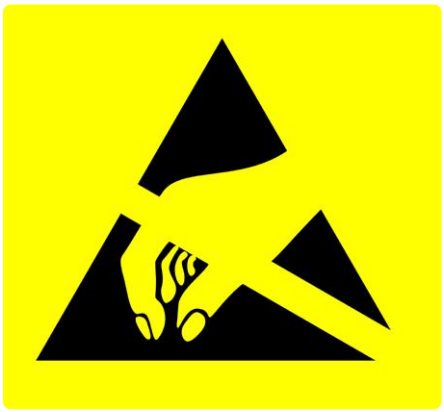


Figure 6. Logo officiel de l'ESD.

Les conséquences peuvent être graves, voire mortelles, si le composant endommagé est par ex. utilisé dans une application cruciale de sécurité : équipement de surveillance laser utilisé pour de délicates procédures cardiaques ou commande de déploiement d'airbags exécuté lors d'une collision automobile.

Protection contre les ESD

Pour réduire les dommages qu'une ESD peut infliger à un appareil électronique, il est important que le personnel manipulant des composants soit conscient du risque et applique des mesures efficaces pour le réduire. Chez les fondeurs, de la fabrication des galettes au test des composants en passant par le back-end, chaque étape est conçue de manière à éliminer tout risque d'endommager des composants par accumulation de charges. Un CI peut être implanté dans un système de contrôle de sécurité crucial dont la fiabilité est d'une importance vitale. Même chez les électroniciens amateurs, une bonne assimilation de l'ESD peut éviter bien des défaillances soudaines de CI et des déconvenues.

Les fondeurs et concepteurs de CI estiment ce problème assez important pour imaginer et intégrer des protections dès le stade de la conception des puces. Ces circuits de protection sont intégrés à la gravure des CI. Cela s'appelle le *durcissement ESD*. Toutefois, les CI ne sont pas protégés contre tous les phénomènes ESD. À mesure que la finesse de gravure et le nombre de composants des puces de générations successives augmentent, les distances entre les éléments se réduisent et la vulnérabilité aux décharges électrostatiques s'accroît. Pour les microcontrôleurs, des circuits de suppression externes sont aussi utilisés pour écrêter les pointes de tension. Semblables aux diodes Zener, ils entrent en conduction lorsqu'un signal d'entrée approche d'un niveau haut ou bas spécifié. Leur temps de réponse est très court et ils absorbent des charges relativement élevées. Malheureusement, même ces dispositifs ne garantissent pas une protection absolue. Un article détaillé [2] a déjà été publié à ce sujet dans Elektor.

Une exigence de base pour manipuler des CI à haute densité est une zone protégée contre les ESD. Dans cet environnement, des mesures sont mises en œuvre pour combattre l'apparition des charges statiques et faire qu'elles soient déchargées au potentiel de la terre de manière contrôlée (**fig. 5.**). Seul un personnel spécialisé, dûment formé et équipé, peut accéder aux zones marquées du logo illustré à la **figure 6**.

Les espaces anti-ESD sont généralement équipés d'un revêtement de sol conducteur mis à la terre. Les étagères et surfaces de travail ont toutes des résistances de fuite < 100 kΩ afin d'amortir toute apparition soudaine de charges et garantir que les charges puissent se dissiper lentement. Les matériaux présentant une faible résistance ohmique surfacique réduisent l'accumulation d'électricité statique et la mise à la terre laisse les charges s'écouler vers le potentiel de la terre.

Les étagères mobiles, les chariots et les chaises ont des surfaces et des roulettes conductrices. Idem pour les outils et autres fers à souder. Les poignées isolées en plastique de l'outillage à main sont susceptibles de porter des charges statiques élevées. Il est donc recommandé d'utiliser uniquement des matériaux conducteurs d'électricité statique assurant l'égalisation en douceur des charges entre la main, l'outil et le composant.

Mieux vaut savoir que ces outils conducteurs ne sont pas adaptés à une utilisation avec des tensions > 25 V, ce qui signifie qu'ils ne sont pas conformes à la norme VDE d'Allemagne !

Les brucelles et autres outils pointus méritent une attention particulière à cet égard ; les formes vives (pointes et angles) ont la faculté de concentrer les porteurs de charge électrique ce qui peut entraîner des décharges et étincelles, même à relativement faible ddp. Si vous travaillez avec des composants sensibles, il est essentiel de porter un bracelet (**fig. 7.**) relié à un point de mise à la terre spécifique.

Dans certains environnements, tels que les salles blanches, l'emploi de matériaux susceptibles de garder une charge statique est inévitable en raison de l'ingénierie des processus. Un flux d'air ionisé combat l'accumulation de charges sur une surface isolante, on place donc des ioniseurs en sortie de climatiseur ; c'est la méthode de choix si la mise à la terre classique ne peut être mise en œuvre (**fig. 8.**). L'ionisation de l'air n'est qu'un complément et ne doit jamais être considérée comme un substitut des mesures de protection ESD classiques.

ESD et assemblage des cartes imprimées

Les robots de placement de composants éviteront tout mouvement ou glissement inutile du composant susceptible de faire naître une charge sur le composant. Aujourd'hui, on utilise principalement des robots appliquant la méthode *pick-and-place* : un bras de préhension saisit le composant et le place sur la carte. Les capteurs de la pince mesurent les propriétés du composant à placer et le rejettent s'il n'est pas conforme aux spécifications.

Toute personne entrant dans une zone de sécurité ESD est obligée de porter chaussures et vêtements spéciaux. Une barrière d'entrée se charge de vérifier que tout visiteur respecte la règle ; l'accès n'est accordé que si la mesure de sa conductivité est conforme. Cette mesure, effectuée à l'aide d'une plaque au sol pour les pieds et une autre pour les mains (**fig. 9.**), commande l'ouverture ou le blocage de la barrière. Le cas échéant, les visiteurs sont munis d'une bande de décharge des chaussures et d'une blouse de travail en coton. Celle-ci doit d'ailleurs être complètement fermée pour littéralement confiner les charges statiques : toute charge sera piégée côté intérieur de la combinaison. En salle blanche, des restrictions particulières s'appliquent dans les zones de production de semi-conducteurs. Il faut porter des combinaisons couvrant tout le corps, à l'exception des yeux et des mains. Des fibres de carbone conductrices sont introduites dans le tissu dont ces combinaisons sont faites (**fig. 10**). Il peut être nécessaire de porter des doigtiers ou des gants dissipateurs.

Dans d'autres situations les phénomènes ESD indésirables peuvent être visibles ou même audibles : les boîtes de film 35 mm contiennent une bobine de film plastique recouvert d'une émulsion photosensible. Si l'air est sec, de minuscules décharges peuvent se produire là où les surfaces du film se séparent lors du tournage. La trace de ces éclairs apparaîtra sur le tirage lorsque le film sera développé. Un phénomène analogue peut arriver lors des prises de son sur bande magnétique, surtout si elle est large et qu'elle avance à vitesse élevée. L'enregistrement peut souffrir d'un léger crépitement que l'on peut supprimer avec un filtre adéquat dans le circuit audio. Retour à l'électronique…

Les phénomènes ESD sont par nature transitoires et la possibilité d'enregistrer leurs profils de courant et de tension n'est que très récente, grâce aux oscilloscopes à mémoire rapides avec échantillonnage > 1 GHz. Des intervalles d'échantillonnage < 1 ns sont requis avec ces phénomènes très brefs. L'intensité du champ peut être mesurée, à l'aide d'un électromètre. Toutes ces mesures sont décrites dans la spécification [3].

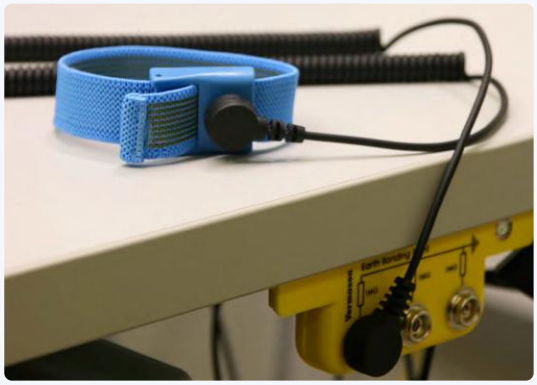


Figure 7. Bracelet mis à la terre.



Figure 8. Ventilation ionisante dans le plafond d'une salle de production de puces.



Figure 9. Barrière de contrôle de la conductivité.



Figure 10. Équipement de salle blanche.

L'ESD et le petit laboratoire

Les électroniciens amateurs et les exploitants de petits labos se demanderont s'il faut prendre ce sujet autant au sérieux. Soyons clairs : les mêmes lois physiques s'appliquent à la maison ainsi que dans les halls de production industriels. Ne lésinez pas sur les mesures de protection anti-ESD. Soyez conscient des risques. Il n'est sans doute pas indispensable d'investir dans autant de matériel onéreux que l'industrie, mais au moins, vous réduirez les risques de dommages en suivant ces règles d'or :

- Même dans un petit laboratoire d'électronique, on peut « désamorcer » le sol en y plaçant un tapis conducteur relié à la terre.
- Idem pour le plan de travail : il est assez bon marché d'y installer un tapis conducteur.
- Un bracelet relié à un point de masse commun est une assurance-vie pour les composants fragiles.
- Tout fer à souder digne de ce nom, même semi-professionnel, doit être doté d'une connexion à la terre et les outils sont disponibles avec des poignées qui dissipent les charges statiques.
- Faites très attention en utilisant des brucelles à pointe acérée.
- Les roulettes de tous les tabourets, chaises, armoires et étagères doivent être de type « conductrice des ESD ».
- Les composants fragiles sont fournis dans un emballage conducteur. Ils doivent y rester jusqu'à leur montage.
- Vérifiez l'absence de trous et déchirures dans les sacs de protection ESD et les boîtes de rangement.

S'assurer qu'un composant sorti de son emballage de protection est déplacé le moins possible, et ça va de soi, qu'aucune de ses bornes de connexion ne peut établir de contact, est une stratégie ESD sûre. Le simple fait de renverser des composants sur une surface isolante peut – par malchance – produire des ddp potentiellement dommageables. Des brucelles et pinces appropriées sont disponibles pour la manipulation et le montage des composants. L'investissement que représentent ces outils sera amplement remboursé à long terme. Même s'ils sont confortables, il vaut mieux éviter de porter des vêtements de sport ; ils sont invariablement en matière synthétique. Choisissez des combinaisons ou blouses antistatiques par nature et veillez à ce qu'elles soient boutonnées et ne bâillent pas lorsque vous vous déplacez.
Veillez à la propreté de l'espace de travail ; nettoyez-en les surfaces ; ôtez les projections de soudure et copeaux de métal ; nettoyez régulièrement les roulettes de chaise et le sol (ne le cirez pas).
L'utilisation de microcomposants complexes dans les petits labos d'électronique peut être source de déconvenues et de regrets si on n'utilise pas les outils adéquats pour manipuler les composants et monter les cartes électroniques. Il y a de fortes chances que les informations fournies ici contribuent à prolonger la durée de vie de quelques CI coûteux et à réduire les recherches de défauts longues et exaspérantes.

(200607-04)

Mots clés

ESD	Décharge électrostatique
ESA	*Electrostatic Area* (zone protégée contre l'ESD)
ESDS	Sensible aux ESD
Salle blanche	Environnement contrôlé à faibles polluants aériens tels que poussières et microbes en suspension. Utilisé pour la fabrication de puces électroniques.
Wafer	Galette de silicium, le substrat où les puces sont gravées.
Front End	Étape de gravure des puces ; dopage de la galette pour produire des zones conductrices.
Back End	Étape de gravure des puces ; contrôle des galettes, découpage en dés et test final des puces.
HBM	Modèle électrique du corps humain (*Human Body Model*).

Des questions, des commentaires ?
Envoyez un courriel à la rédaction d'Elektor (redaction@elektor.fr).

Contributions

Texte : **Peter Beil**
Rédaction : **Thomas Scherer**
Mise en page : **Harmen Heida**
Traduction : **Yves Georges**

PRODUITS

- **Jeu de tournevis ESD Bernstein 4-620 (6 pièces)**
https://www.elektor.fr/bernstein-4-620-esd-screwdriver-set-6-pieces
- **Porte-outils Vario Bernstein 2100 avec 6 outils ESD**
https://www.elektor.fr/bernstein-2100-esd-tool-holder-vario-6-tools

LIENS

[1] **HBM** : https://en.wikipedia.org/wiki/Human-body_model
[2] **P. Krüger, « protection active anti décharges électrostatiques », Elektor 01-02/2014** : www.elektormagazine.fr/130221
[3] **Association ESD (États-Unis)** : www.esda.org/about-esd/esd-fundamentals/part-6-esd-standards/

énergie solaire
pour les robots de tonte
Écologique, peu coûteux, simple !



Thomas Scherer (Allemagne)

Un robot de tonte est un bel objet, confortable et pratique. Mais écologiquement, il y a encore une marge d'optimisation, car il a besoin d'énergie électrique en permanence. Par ailleurs on ne peut pas partout poser facilement un câble de 230 V pour l'alimentation électrique. Un système solaire adapté résout les deux problèmes.

Il fut un temps (il y a trois ans), lorsque le gamin du voisin est devenu jeune étudiant, où il a préféré passer du temps à étudier plutôt qu'à tondre la pelouse de mon jardin. J'ai donc acheté un robot de tonte. En plus, cette décision permettait de recycler les tontes de gazon et donc les nutriments, au lieu de transformer rapidement mon tas de compost en montagne. Passées les difficultés initiales pour la pose du câble de délimitation en cherchant à épouser la forme plutôt complexe de mon terrain, le robot a fait ce qu'il était censé faire. Et s'il n'est pas en panne, il tond encore aujourd'hui…

Fonctionnement solaire

Tout aurait été parfait et le conte de fées se serait terminé ici, si récemment ma petite amie – qui a un doctorat en chimie – ne m'avait pas taquiné. Il se trouve qu'un bon ami à moi voulait équiper son bateau à moteur d'un réfrigérateur électrique qui devait être alimenté par l'énergie solaire. En tant qu'« Electricus » de mon clan, j'étais naturellement chargé des calculs de base de ce système solaire. Et dès qu'Alexandra a entendu parler de cela, elle a pointé les lèvres et m'a demandé avec une feinte innocence : « Et pourquoi ton robot de jardin ne fonctionne pas encore à l'énergie solaire ? ».

Bang ! Bien sûr, je ne pouvais pas laisser passer ça. J'ai donc rapidement étudié les modalités de ce projet, consulté l'internet pour connaître les prix actuels des composants et, une demi-heure plus tard, je lui ai dit : « C'est aussi facile à faire que peu coûteux. Avec environ 100 €, on peut y arriver ! ».

Elle sourit avec scepticisme, car en tant que scientifique convaincue, le système d'exploitation orienté Goethe installé dans son cerveau fonctionne selon la devise : « J'ai bien entendu le message, seule ma foi vacille ». Les belles paroles ne comptent pas pour elle, seulement les actes. Je devais donc le prouver…

Considérations de base

J'ai un robot de tonte Gardena, une version bon marché des célèbres machines Husqvarna. Pour une pelouse d'environ 550 m², il doit tondre environ 4 h/j pendant la phase de croissance – au printemps et en automne, alors qu'en plein été sec, il y arrivera avec 2 h/j ou moins. Heureusement, il y a beaucoup de soleil au moment où on en a besoin.

Le robot a une batterie qui lui permet de tondre pendant environ une heure. Il doit ensuite retourner à sa station de base et après une heure de recharge, il est prêt à

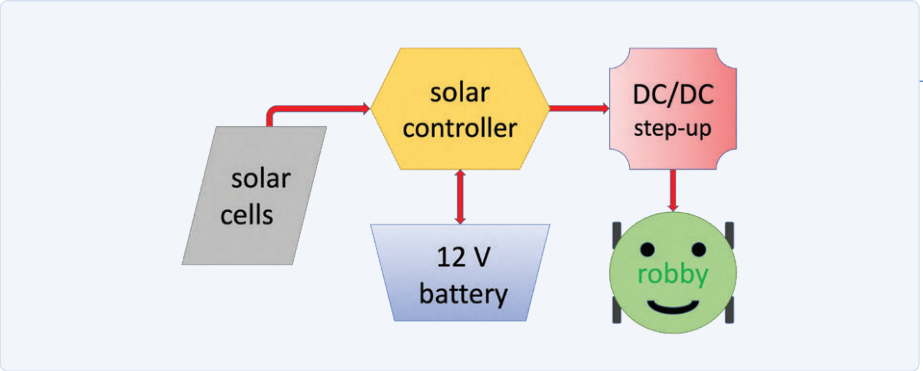


Figure 1. Le schéma fonctionnel d'un système solaire pour les robots de tonte avec le flux d'énergie (rouge).

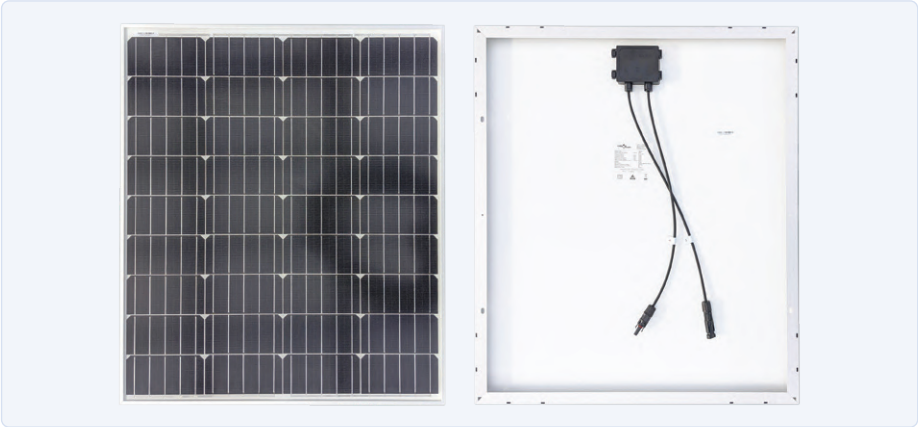


Figure 2. Avant et arrière de mon panneau solaire 12 V avec 80 $W_{crête}$.

tondre à nouveau. Si je veux le faire tondre pendant 2 h/j, je dois programmer (sans fil) une fenêtre de 3 h qui inclut une heure de pause pour la recharge.

Afin de pouvoir estimer la quantité d'énergie nécessaire par processus de charge, j'ai interrogé l'internet : le bloc d'alimentation correspondant est censé fournir 28 V à un maximum de 1,3 A. Comme je suis patient, je l'ai mesuré : en fait, mon alimentation délivrait une tension de 28,1 V (CC) en mode hors charge. Lorsque le robot chargeait sa batterie, ma pince ampèremétrique indiquait un courant de 1,29 A. Pendant la charge, la tension tombait à 26,5 V. « L'alimentation est vraiment faible » ai-je pensé, ce qui s'est révélé faux par la suite ;-) Quoi qu'il en soit, 26,5 V × 1,3 A donnent environ 35 W selon Adam Ries et la formule de puissance. Cela signifie qu'il faut environ 35 Wh pour une heure de charge. Pour les quatre phases de tonte d'une journée de début d'été, on peut s'attendre à 140 Wh. C'est la quantité d'énergie électrique quotidienne que le système solaire devrait être capable de fournir (et de stocker) même par faible ensoleillement.

À cela s'ajoute le besoin en énergie de la station de base, qui alimente H24 les câbles de délimitation avec un signal pulsé d'intensité notable, dont le champ magnétique est détecté par le robot pendant le fonctionnement (voir mon article [1]). J'ai mesuré un courant de repos (hors charge) de 85 mA à 28,1 V. Par conséquent, environ 2,4 W (≈ 58 Wh/j) sont nécessaires en plus des charges.

Même si ces valeurs ne sont valables que pour mon robot de tonte, on peut supposer qu'elles le sont plus ou moins pour la plupart des appareils actuellement disponibles. Mon robot convient pour 1 100 m² et fait donc partie de la « gamme moyenne ». Les petits robots ont certainement besoin de moins d'énergie et les grands de plus. Cependant, il vaut mieux le mesurer soi-même que de se contenter de supposer.

Système solaire

Si je table sur 200 jours de tonte par an avec une moyenne de 3 h/j, j'obtiens 600 cycles de charge, ce qui donne un besoin énergétique annuel de 21 kWh. Si l'on y ajoute les 2,4 W × 24 h × 200 j ≈ 11,5 kWh, on obtient un coût d'électricité de 9,75 € (à 30 ¢/kWh). Une installation solaire pour les 100 € supposés serait amortie au bout de dix ans. Cela en vaut-il la peine ? D'un point de vue économique, sans doute pas, mais d'un point de vue écologique oui, car pendant cette période, vous auriez économisé environ 105 kg de CO_2. En outre, le choix de l'emplacement de la station de charge de la tondeuse est indépendant du raccordement au réseau. Troisièmement, Alexandra…

Le système solaire du robot de tonte nécessite quatre composants (**fig. 1**) : en plus du **panneau solaire**, une **batterie** est nécessaire pour stocker l'énergie, sinon la tondeuse ne peut pas être chargée lorsque le soleil ne brille pas. Ce dernier point entraîne la nécessité d'un **contrôleur de charge** qui transfère l'énergie du panneau à la batterie, en tenant compte des tensions de fin de charge et de décharge de la batterie. Enfin, il faut un **convertisseur CC/CC** pour élever la tension de la batterie au niveau requis par la station de charge de la tondeuse. Le dimensionnement adéquat de chaque composant impose une petite étude et un calcul approximatif. Tous les calculs sont basés sur le besoin énergétique maximal par jour, qui dans mon cas est ≤200 Wh. Passons maintenant aux calculs.

Panneau solaire

Idéalement, le panneau devrait fournir en deux heures autant d'énergie que la tondeuse en consomme en une heure de tonte, en tenant compte de la pause de charge d'une heure. En estimant à 15% les pertes dues au contrôleur de charge et au convertisseur CC/CC, on arrive à environ 40 W par 2 h, soit 20 W de puissance continue pendant le temps où la tondeuse tond (et se charge). Ensuite, la batterie reste généralement pleine si le soleil brille. Comme la puissance de sortie des panneaux solaires est mesurée en $W_{crête}$ et que vous aurez rarement des conditions optimales sans ombre et une inclinaison idéale du panneau, doubler la puissance de sortie n'est certainement pas une erreur. Un panneau de 40 $W_{crête}$ est-il suffisant ? À mon avis, non, car les rayons du soleil sont très obliques au printemps et en automne, ce serait encore une supposition très optimiste. Un autre facteur de sécurité de 2 n'est sans doute pas exagéré. Donc 80 $W_{crête}$ dans mon cas est le minimum. J'ai commandé un tel panneau sur eBay pour 55 € (**fig. 2**), car il avait – quelle coïncidence ! – avec 77 × 66,5 cm exactement les bonnes dimensions pour servir de toit à la niche du chien, que j'ai transformée en garage à tondeuse.



Figure 3. La batterie au plomb AGM 12 V / 12 Ah.

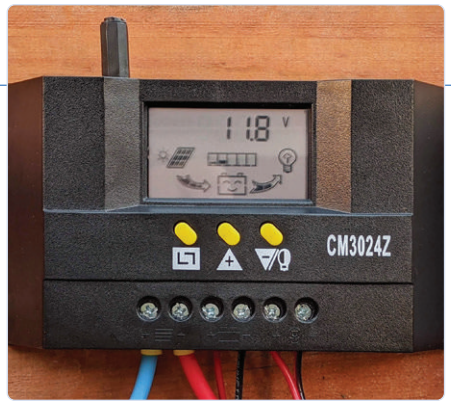


Figure 4. Ce régulateur de charge solaire bon marché en technologie MLI tolère des courants jusqu'à 30 A.

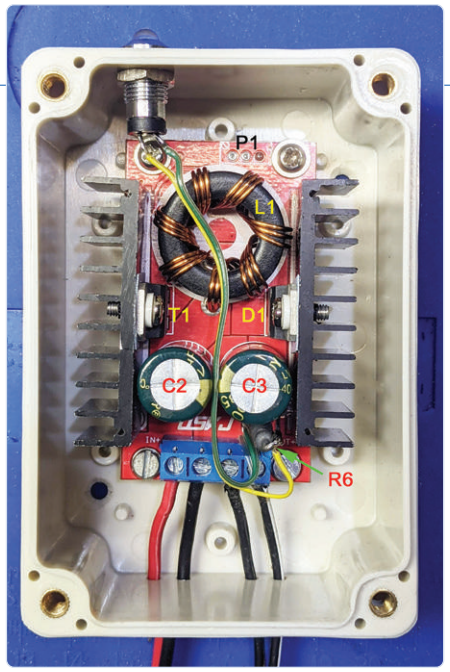


Figure 5. Un simple convertisseur élévateur a été complété par un limiteur de courant de 1,3 A.

Les mesures ont montré qu'à la fin du mois d'octobre dans le sud du land Bade (Allemagne), vers midi, lorsque le soleil est au zénith, on peut s'attendre à une puissance d'environ 22 W. Le calcul approximatif était correct !

Batterie

Le stockage d'énergie doit être suffisant pour au moins une journée de couverture nuageuse dense. La batterie doit donc être capable de stocker ≥140 Wh. Une batterie AGM 12 V plomb-acide de 12 Ah (≈ 150 Wh) coûte environ 25 € sur eBay. L'AGM (*Absorbed Glass Mat*) est nécessaire pour obtenir un maximum de cycles de charge dans cette application et donc une durée de vie longue. Un panneau de 80 W délivre un courant de pointe d'un peu moins de 6 A. La batterie est donc chargée avec un maximum de 0,5 C (généralement moins ; voir encadré **Courants en C**), ce qui est bon pour la longévité. Elle est déchargée à environ 40 W, ce qui correspond à 0,25 C. Comme il y a également de nombreux cycles courts de charge partielle, on peut s'attendre à juste titre à des années de fonctionnement. Une batterie de 12 Ah est un minimum pour cette application – plus serait mieux – mais je voulais essayer avec le minimum calculé et j'ai donc commandé la batterie de la **figure 3**.

Régulateur de charge solaire

Le marché est inondé par une multitude de régulateurs de charge de petite puissance bon marché, qui conviennent à mes fins. J'ai décidé d'acheter un régulateur relativement « cher » à 13,50 €, parce qu'il a un boîtier métallique et des ailettes de refroidissement à l'arrière. C'est peut-être exagéré, car mon système n'utilise que 20% des 30 A possibles.

Les unités bon marché fonctionnent toutes avec un simple contrôle MLI, ce qui n'est pas tout à fait optimal. Si vous voulez un meilleur contrôleur MPPT (*Maximum Power Point Tracking*) pour une efficacité maximale, vous devrez débourser au moins 75 €. Afin de rester en dessous des 100 € prévus, j'ai décidé d'utiliser la version bon marché de la **figure 4**.

Convertisseur CC/CC

J'ai d'abord pensé qu'un système solaire de 24 V avec deux batteries de 12 V connectées en série serait la solution la plus simple. La station de charge de la tondeuse fonctionne probablement bien avec 24 V, même si elle est spécifiée pour 28 V. Cela permettait d'économiser un convertisseur élévateur, qui autrement est indispensable pour augmenter la tension de fonctionnement d'une batterie de 12 V, en éliminant en outre la perte de conversion et le courant de repos du convertisseur. Cependant, les panneaux solaires de 24 V sont plus chers. Ce qui est devenu évident par la suite, c'est que j'ai eu de la chance en optant pour la technologie 12 V.

Il fallait donc trouver un convertisseur élévateur de puissance suffisante, qui amène les 12-14 V de la batterie aux 28 V requis. Il devait pouvoir fournir au moins 40 W. Sur l'internet, on trouve des modules appropriés en abondance. J'ai décidé d'acheter un module de 150 W pour un peu moins de 10 €, qui promettait des réserves suffisantes. La **figure 5** montre la version modifiée intégrée dans un boîtier en plastique. Le rendement annoncé était de 95%. Et c'était vrai, comme l'ont montré mes mesures. Ce n'est pas étonnant, car l'UC3843 n'est certainement pas un mauvais choix pour IC1.

En tout, j'ai dépensé 55 € + 25 € + 13,5 € + 10 € = 103,50 € – une bonne estimation, n'est-ce pas ? En plus de cela, j'ai dû dépenser quelques euros pour les petites pièces,



Figure 6. Les quatre étapes de la transformation d'une niche pour chien mal utilisée en un garage solaire pour mon robot de tonte.

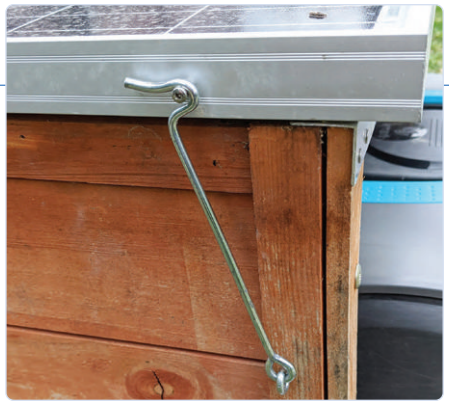


Figure 7. Comment empêcher le soulèvement du panneau par le vent.

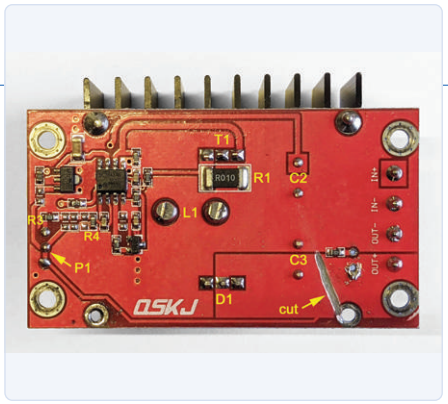


Figure 8. Face arrière de la carte du convertisseur élévateur.

les vis en acier inoxydable, les charnières et le boîtier en plastique.

Conversion avec obstacles

Il y a d'abord eu les travaux mécaniques que j'affectionne peu en tant qu'électronicien : la transformation de l'ancien logement de la station de base en un garage avec un toit solaire. Les quatre photos partielles de la **figure 6** illustrent la procédure.

Tout d'abord, j'ai retiré les deux moitiés du toit (**a**) et j'ai scié les pignons comme indiqué en (**b**). Ensuite, j'ai ajouté des profilés d'angle en aluminium pour stabiliser le support devenu quelque peu bancal, et enfin j'ai fixé le panneau solaire de manière articulée (**c**). L'acier inoxydable n'est pas de trop dans les charnières et les vis, si l'on veut éviter la rouille. En (**d**), vous pouvez voir le résultat avec le panneau déplié et la tondeuse garée. La **figure 7** montre comment j'ai bloqué le panneau pour empêcher son soulèvement par le vent, ce qui est une nécessité absolue.

L'étape suivante a déjà été plus intéressante : j'ai installé et câblé la batterie, le contrôleur de charge et le convertisseur élévateur. La LED de la station de charge est immédiatement devenue verte, même si j'avais réglé le convertisseur à seulement 27 V pour tenir compte de la chute de tension de l'alimentation d'origine. Au début, tout semblait fonctionner et le robot de tonte a reconnu les câbles de délimitation installés et s'est mis à tondre joyeusement.

Pour voir si la tondeuse se chargeait bien, je l'ai ensuite poussée dans sa station de charge. Mon smartphone était déjà en train de « pinguer » lorsque l'application de la tondeuse a reçu un message d'erreur de la part de la tondeuse…

Le message affiché était : « Courant de charge trop élevé ».

Oh ! « Ça peut faire de gros dégâts » me suis-je dit et j'ai rapidement sorti la tondeuse de la station de charge. Le soir, Alexandra m'a demandé : « Tout s'est bien passé avec ta tondeuse ? ». J'ai dû concéder un rapport de situation dans le style de Radio Erevan : « En principe oui, mais… ». Je lui ai dit que le mode de repos fonctionnait bien, mais que le message d'erreur était ennuyeux. Mais je soupçonne que le robot compte sur le fait que son alimentation a un limiteur de courant intégré, et qu'il se charge donc simplement avec un courant constant. Demain, je mesurerai l'alimentation et, si nécessaire, j'équiperai le convertisseur élévateur d'un limiteur de courant ou j'en fabriquerai un pour lui. Si j'avais connecté directement le 24 V des batteries sans limiteur de courant, il y aurait eu des nuages de fumée. J'ai de la chance dans mon malheur.

Limitation de courant

Le lendemain, il s'est avéré que mon hypothèse était correcte. Avec une charge de 24 Ω, l'alimentation d'origine délivrait un courant de 1,17 A pour une tension stable de 28,05 V. Avec une charge de 12 Ω, je n'ai pu mesurer que 15,5 V. Le courant était de 1,29 A. Un nouveau test avec 15 Ω a donné 19,4 V pour un courant stable de 1,29 A. Il s'agissait donc d'une alimentation de 28 V avec un limiteur de courant intégré d'environ 1,3 A.

J'ai maintenant examiné de plus près le circuit imprimé du convertisseur (**fig. 8**) et reproduit le circuit autant que nécessaire. La moitié supérieure de la **figure 9** montre le circuit original. La moitié inférieure montre mes modifications en rouge.

La tension de sortie est ajustée par IC1 pour que son entrée VFB reçoive une tension appliquée de 2,5 V. Elle est réglable avec P1 entre U_{in} et 34 V. La tension de sortie de ce type de convertisseur ne peut pas être plus petite que l'entrée, car même lorsque le convertisseur est éteint, un courant circule à travers L1 et D1 vers la sortie. Mais si la tension à la sortie peut varier entre 14 V et 28 V, c'est suffisant, me suis-je dit. Il suffisait donc de boucler une résistance shunt (R6) et d'augmenter la tension sur la broche VFB en cas de surcharge par un transistor PNP, pour réduire la tension à la sortie.

J'ai d'abord remplacé P1 par une résistance fixe de 6,8 kΩ. Cela a donné un résultat de 27 V – c'est bon. Ensuite, le capteur de courant d'entrée R1 a été remplacé par une résistance plus élevée de 0,22 Ω (CMS 2512), car 150 W ne sont pas du tout nécessaires. Avec 0,47 Ω pour R6 et une tension BE

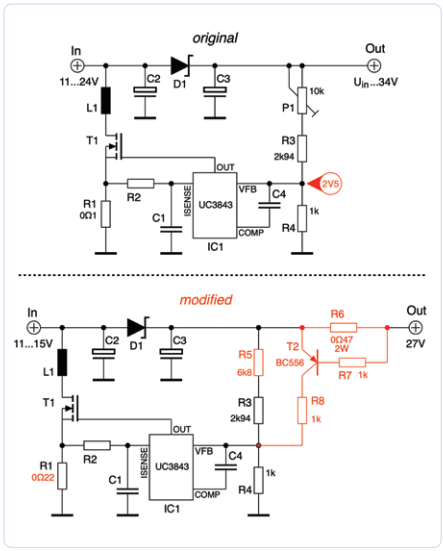


Figure 9. Circuit original et circuit modifié (partiel) du convertisseur élévateur.



Figure 10. Tout est terminé : garage solaire avec électronique intégrée.

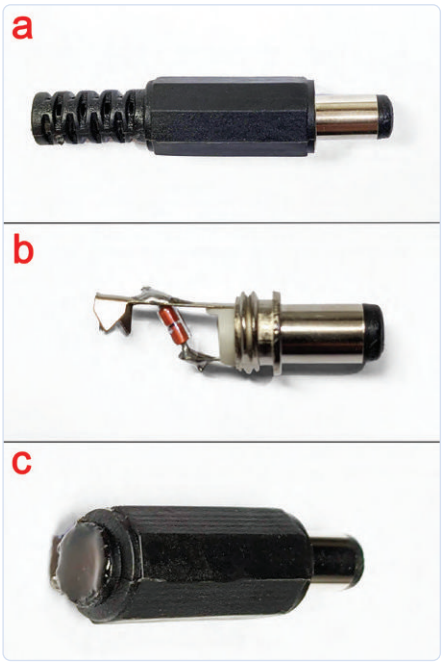


Figure 11. Une fiche jack de 3,5 mm (a) contient une diode (b) comme capteur de température. En (c), elle est scellée de manière étanche avec de la colle chaude.

d'environ 650 mV du T2 ajouté, on peut s'attendre à une limitation de courant de 1,38 A, ce qui conviendrait. J'ai donc installé les composants rouges (T2, R7 et R8 directement sur la face inférieure de la carte) et sorti l'ampèremètre : exactement 1,30 A. Gagné !

Tout est bien qui finit bien !

J'ai remis le convertisseur modifié dans son boîtier, je l'ai replacé dans le garage de la tondeuse (**fig. 10**) et je l'ai allumé : la LED verte de la station de charge s'est allumée. *Tadaah* ! Lorsque j'ai poussé la tondeuse à l'intérieur, il n'y avait plus de message d'erreur. Le contrôleur de charge indiquait qu'environ 37 W circulaient vers le robot de tonte. Que demander de plus ? Avant d'annoncer triomphalement mon succès à Alexandra, j'ai fait tondre et charger la tondeuse à plusieurs reprises. Aucun autre problème n'est apparu.

Et au cas où vous vous demanderiez ce qui se passe s'il pleut pendant une semaine en automne ou au printemps – cette question est justifiée : avec une durée de tonte de 2 h/j, la batterie ne dure que 1,5 à 2,5 j, puis elle est vide et la tondeuse reste au garage jusqu'à ce que le soleil brille suffisamment. Une légère couverture nuageuse suffit pour qu'elle se réveille à nouveau. Si vous voulez plus de réserve, la batterie et éventuellement le panneau doivent être dimensionnés plus largement. Un panneau de 100 W et une batterie de 20 Ah ne sont certainement pas exagérés. Le régulateur de charge et aussi le convertisseur peuvent tolérer une puissance bien supérieure.

Au passage, pas de problème en hiver, car c'est le moment où la tondeuse (y compris la batterie) passe de son garage à mon garage.

Addendum

Les tensions d'une batterie au plomb dépendent de la température. Par conséquent, certains régulateurs de charge sont équipés d'un capteur de température pour en tenir compte. Mon contrôleur de charge incluait une prise jack de 3,5 mm apparemment vide (**fig. 11a**). En dévissant le capuchon, une simple diode en silicium est apparue (**b**). Il est clair qu'une jonction PN a un coefficient de température d'environ –1,7 mV/K, ce qui est suffisant. Mais ce montage rudimentaire n'était pas une bonne idée à cause de l'humidité possible et des petits animaux qui cherchent un abri. J'ai donc coupé la protection contre les courbures et simplement rempli l'œillet avec de la colle thermofusible (**c**). Maintenant, il est à l'abri de l'eau et des animaux.

Si vous avez quelque chose de similaire en tête : il existe également des convertisseurs *step-up/down* ou *boost/buck* prêts à l'emploi avec un limiteur de courant réglable pour juste quelques euros supplémentaires. Vous pouvez donc éviter la modification que j'ai faite.

(200553-04)

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

Contributeurs
Idée, mise en œuvre et texte : **Thomas Scherer**
Rédaction : **Jens Nickel, CJ Abate**
Mise en page : **Giel Dols**
Traduction : **Denis Lafourcade**

COURANTS EN C

Pour les batteries rechargeables, le courant de charge ou de décharge en fonction de la capacité nominale de la batterie est donné dans l'unité non-SI couramment utilisée, le C – c'est le rapport courant (en A) sur capacité nominale (en Ah) ; pour mémoire, capacité nominale (Ah) = courant (A) × durée (h). Un C de 0,5 donne un courant de 6 A pour une batterie de 12 Ah. L'avantage de l'unité C est qu'elle indique le stress de la batterie. Les batteries sont conçues pour certaines valeurs C maximales. Plus les valeurs réelles de C sont élevées, plus la durée de vie de la batterie est courte.
=> Pour un courant donné, les batteries plus grosses durent plus longtemps.

PRODUITS

- **Régulateur de batterie pour panneau solaire avec sortie USB**
  www.elektor.fr/usb-solar-panel-battery-regulator-charge-intelligent-controller-12-24-v-10-a
- **Pince ampèremétrique 4350 de marque PeakTech**
  www.elektor.fr/peaktech-4350-clamp-meter
- **Multimètre numérique avec Bluetooth OW16B de marque OWON**
  www.elektor.fr/owon-ow16b-digital-multimeter-with-bluetooth

LIEN

[1] **« Locating Wayward Wires » (article en anglais), Elektor 03-04/2019 :**
www.elektormagazine.com/magazine/elektor-141/57129

elektorstore

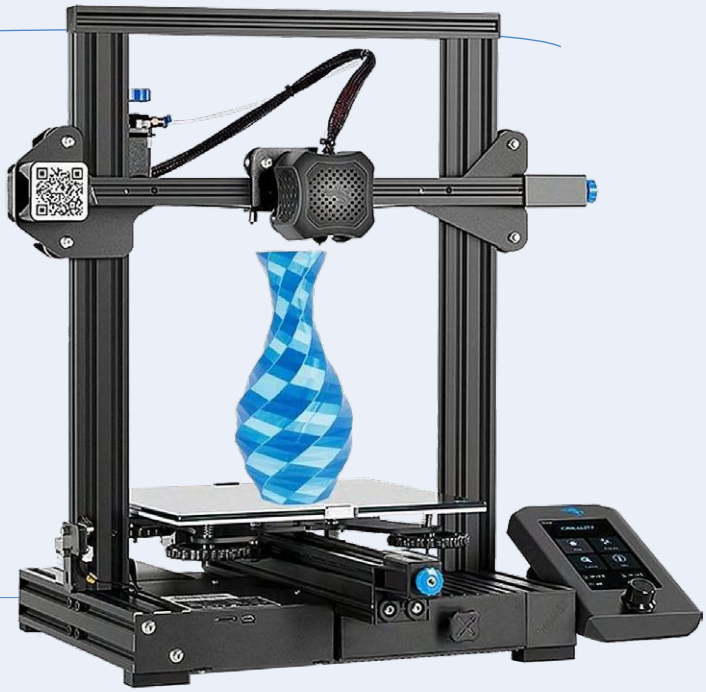
www.elektor.fr

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 !

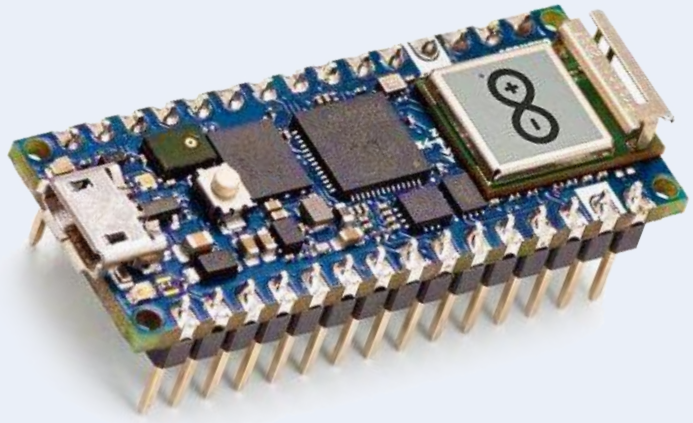
Imprimante 3D Creality Ender-3 V2



Prix : 259,00 €
Prix (membres) : 233,10 €

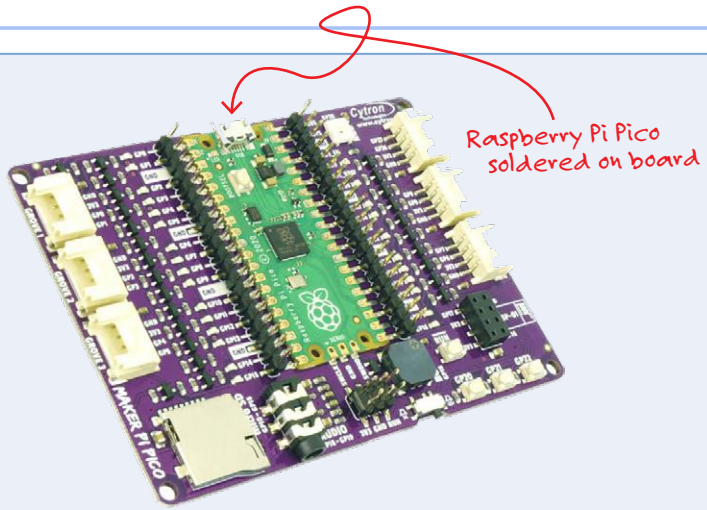
www.elektor.fr/19744

Arduino Nano RP2040 Connect with headers



Prix : 29,95 €
Prix (membres) : 26,96 €

www.elektor.fr/19754



Cytron Maker Pi Pico (Raspberry Pi Pico RP2040 inclus)

Prix : 19,95 €

Prix (membres) : 17,96 €

www.elektor.fr/19706



Microscope numérique Andonstar AD409 avec écran 10,1“

Prix : 399,00 €

Prix (membres) : 359,10 €

www.elektor.fr/19681

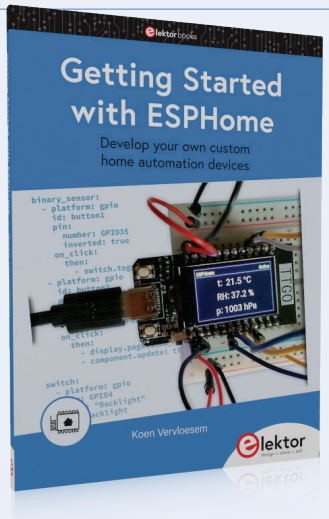


The Complete Linear Audio Library (clé USB)

Prix : 149,95 €

Prix (membres) : 89,95 €

www.elektor.fr/19672



Getting Started with ESPHome

Prix : 29,95 €

Prix (membres) : 26,96 €

www.elektor.fr/19738



l'Europe tente de dompter les GAFA

Tessel Renzenbrink (Pays-Bas)

La Commission européenne œuvre en faveur d'un internet respectueux des droits fondamentaux de l'utilisateur et offrant des règles de concurrence équitables au sein d'un marché numérique unique. Parviendra-t-elle à dompter les géants du numérique ?

L'Europe souhaite retrouver l'internet d'avant les GAFA en tentant de se libérer de leur mainmise. La Commission européenne (CE) s'efforce à cet effet de créer un marché numérique unique fondé sur les valeurs européennes – un internet centré sur l'individu, respectueux de ses droits fondamentaux et pourvu de conditions de concurrence équitables. C'est dans le cadre de cette stratégie pour l'avenir numérique de l'UE que la Commission européenne a proposé en décembre dernier deux textes législatifs : le *Digital Services Act* (DSA), qui vise à protéger les citoyens et à garantir leurs droits fondamentaux en ligne, et le *Digital Markets Act* (DMA), qui vise à limiter le pouvoir des « très grandes plateformes en ligne » ou « contrôleurs d'accès ».
Ces deux projets instaurent trois niveaux de responsabilité. Le premier, celui qui impose le moins d'obligations réglementaires, s'applique aux petites entreprises. Le second niveau concerne les grandes plateformes numériques. Le troisième vise les très grandes plateformes, comme Facebook et Google. Le texte les désigne comme « contrôleurs d'accès » car elles exercent un contrôle substantiel sur la mise en relation entre les entreprises et les utilisateurs. Cette position dominante leur permet d'établir des règles qui désavantagent les autres entreprises et peuvent conduire à une concurrence déloyale. Une plateforme est considérée comme

contrôleur d'accès si elle dessert au moins 10 % de la population de l'UE.

Projet « Digital Service Act »

Un des problèmes auxquels tente de s'attaquer le DSA est la dissémination des contenus illicites ou préjudiciables, tels que les discours haineux, le « *revenge porn* » et la vente de contrefaçons. Selon cette proposition de loi, les fournisseurs de service seront tenus de mettre en œuvre des mécanismes simples permettant à tout un chacun de signaler la présence de tels contenus. Tout signalement devra être traité rapidement et le plaignant informé des suites apportées. Les fournisseurs devront par ailleurs indiquer si le traitement est automatisé, et désigner un interlocuteur pour les autorités nationales.

Obliger les plateformes en ligne à supprimer un contenu illicite et engager des poursuites à leur encontre si elles ne le font pas n'est pas sans risque. Une plateforme pourrait combler la mesure en pensant qu'il vaut mieux prévenir que guérir, et quoi qu'il en soit deviendrait *de facto* l'arbitre de l'expression en ligne. Les organisations de défense des droits numériques ont mis en avant ce risque et fait valoir que cette obligation légale portait atteinte à la liberté d'expression et au droit à l'information. Le projet DSA a donc inscrit dans sa proposition de loi deux principes fondamentaux de la *Directive sur le commerce électronique*, le cadre juridique qui régit l'internet européen depuis 2000. Le premier stipule qu'une plateforme n'est pas responsable des informations qu'elle stocke si elle n'a pas « effectivement connaissance » de leur caractère illicite. Le second est l'interdiction pour les États membres d'imposer aux plateformes une obligation de surveillance générale de leurs systèmes.

Le projet DSA protégera les utilisateurs contre les suppressions de contenu injustifiées et les exclusions arbitraires d'une plateforme. Les fournisseurs de services devront ainsi informer les utilisateurs de toute suppression de contenu et en donner la raison. Les grandes plateformes devront quant à elles mettre en œuvre une procédure contestable de traitement des plaintes. Elles devront également prévoir des mécanismes de résolution des litiges sous la supervision d'un organisme indépendant.

En introduisant une obligation de transparence, le projet DSA entend donner plus de poids à la société dans le déséquilibre qui existe entre plateformes et utilisateurs. De nombreuses plateformes collectent actuellement un grand nombre de données sur leurs utilisateurs de façon opaque, notamment pour les besoins de leurs services de publicité ciblée ; la proposition de loi DSA exigera des grandes plateformes qu'elles fournissent aux utilisateurs l'identité de l'annonceur et justifient tout ciblage. Elles devront également expliquer clairement quels paramètres utilisent leurs systèmes de recommandation, et permettre aux utilisateurs de les modifier ou de ne pas adhérer à ce système.

> *La CE sort l'artillerie lourde pour lutter contre les géants du numérique en les exposant à des sanctions sévères.*

Projet « Digital Markets Act »

Le projet de loi DMA vise les « contrôleurs d'accès » et entend créer des conditions de concurrence équitables. Le texte propose à cet effet d'interdire aux très grandes plateformes de favoriser leurs propres produits au détriment de la concurrence (comme le fait p. ex. Google en plaçant ses services en tête des résultats de son moteur de recherche). Les entreprises verront leurs données mieux protégées puisqu'elles devront avoir libre accès à celles qu'elles auront générées sur la plateforme. Elles pourront aussi extraire ces données et les héberger ailleurs. Le contrôleur d'accès n'aura pas le droit de les utiliser pour concurrencer l'entreprise, et ne pourra plus faire valoir son statut d'intermédiaire pour empêcher une entreprise et des consommateurs de se mettre en contact en dehors de sa plateforme.

La CE sort l'artillerie lourde pour lutter contre les géants du numérique. Les contrôleurs d'accès qui ne respecteront pas la réglementation DSA s'exposeront à de lourdes sanctions : des amendes pouvant atteindre 6 % de leur chiffre d'affaires annuel, et même la suspension temporaire de leurs activités européennes en ligne en cas de non-respect persistant. Une violation de la loi DMA peut entraîner des sanctions encore plus sévères : des amendes s'élevant à 10 % du CA annuel, et la dissolution de l'entreprise en cas de comportement illégal persistant.

Le projet de règlement *Digital Services Act* suit actuellement la procédure législative usuelle : la CE, le Parlement européen et les États membres doivent tous se mettre d'accord sur le texte final avant son adoption.

210177-04 – VF : Hervé Moreau



Drapeaux européens devant le siège de la Commission européenne. (Photo : Guillaume Périgois, Unsplash)

hexadoku casse-tête pour elektroniciens

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 **16 août 2021** à l'adresse **hexadoku@elektor.fr**

Les gagnants

La solution de la grille du numéro de mai/juin 2021 est **F9E18.**

La liste des gagnants est publiée ici : www.elektormagazine.fr/hexadoku

Bravo à tous les participants et félicitations aux gagnants !

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

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

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.

Ils nous font confiance, n'est-ce pas ?



Nous aimons l'électronique et les projets, et nous faisons tout notre possible pour répondre aux besoins de nos clients.

Le magasin Elektor : **Jamais cher, toujours surprenant**

Elektor is rated **Excellent**

Based on 8 reviews

Trustpilot

Consultez d'autres avis sur notre page Trustpilot : www.elektor.fr/TP

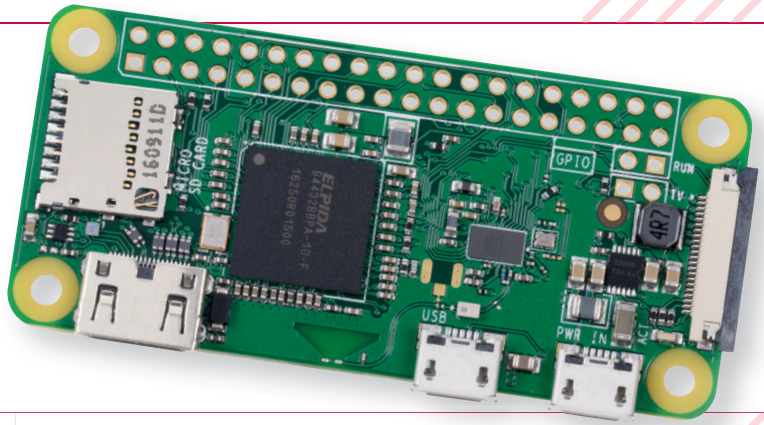
Vous pouvez également vous faire votre propre opinion en visitant notre Elektor Store, www.elektor.fr

elektor

design > share > sell

ABONNEZ-VOUS ET RECEVEZ

Raspberry Pi Zero W GRATUIT





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



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

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

- Six numéros du magazine MagPi
- Une carte Raspberry Pi Zero W
- Boîtier Pi Zero W

Vos avantages :

- Prix au numéro réduit
- Chaque numéro directement dans votre boîte aux lettres
- Tous les numéros disponibles sous forme numérique (PDF)
- Cadeau de bienvenue d'une valeur de 23,90 €
- Découverte de chaque nouveau numéro avant sa sortie en kiosque

ABONNEZ-VOUS : WWW.MAGPI.FR