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 489

créer > partager > vendre

MAI & JUIN 2021
ELEKTORMAGAZINE.FR

# station de soudage 2021

nouvelle version pour les fers Weller RT et autres

60 years young

p. 30

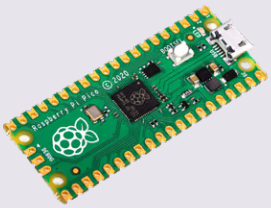
## dans ce numéro :



et bien d'avantage !


**supercharger LiPo DIY**
projet commun de GreatScott! et Elektor
p. 06

**bouton connecté de gestion du temps**
ESP32 Basic Core de M5Stack & service de suivi en ligne Toggl
p. 48



**carte Raspberry Pi Pico à RP2040**
du nano-ordinateur au microcontrôleurs
p. 57

L 19624 - 489 - F: 15,50 € - RD

# 60 elektor MAG

sixty › years › young

Cette année, Elektor aura soixante ans. Au cours des douze prochains mois, nous rendrons hommage aux fondateurs ainsi qu'à la communauté qui ont rendu tout cela possible ! Nous sommes surtout fiers qu'Elektor participe a la révolution électronique depuis *1961!*

1961 1971 1981 1991 2001 2011 2021 ›

44ème année
n° 489 – mai/juin 2021

ISSN 0181-7450
Dépôt légal : mai 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
Phone: +31 (0)6 4403 9907
E-mail: raoul.morreau@elektor.com



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*

# Restez jeune avec nous !

Le moment est venu : Elektor a 60 ans. En avril 1961, Bob van der Horst publie le premier numéro de son nouveau magazine « Elektronika Wereld » (« monde de l'électronique »), qui sera plus tard rebaptisé Elektuur puis Elektor. L'éditeur a senti qu'il y avait un manque sur le marché. La plupart des connaissances en électronique de l'époque étaient enfouies dans la littérature académique, dans d'innombrables fiches techniques, dans des documents difficiles à comprendre et dans des diagrammes ennuyeux. Van der Horst avait senti que beaucoup de gens comme lui seraient enthousiastes à l'idée de réaliser des montages électroniques, d'en adapter de manière créative à leurs propres besoins, voire d'en concevoir à partir de zéro. Pour ce faire, les électroniciens ont besoin de connaissances préalables. La meilleure façon de transmettre ces compétences aux lecteurs, c'est de les accompagner pas à pas pour qu'ils réussissent. L'apprentissage se fait au fil de l'eau.

Soixante ans plus tard, nous restons fidèles à ce modèle. L'article sur la programmation orientée objet, celui sur MicroPython ou encore celui sur la connexion à un service en ligne comme Toggl sont des parfaits exemples de cette philosophie – avec un mélange éprouvé de théorie et de pratique. En tant que rédacteur en chef, je bénéficie du fait que nos rédacteurs et nos auteurs sont tous infectés par le virus du bricolage. Par conséquent, je ne reçois presque jamais d'articles « trop secs » et uniquement théoriques. La plupart du temps, je dois même ralentir ces esprits créatifs pour que le projet ne prenne pas trop d'ampleur (après tout, il faut bien boucler le magazine à un moment donné). Le contenu que nous recevons permet, à notre équipe et à notre magazine, d'être à l'affût de l'actualité et de rester dans le coup. Je ne sais pas ce qu'Elektor couvrira dans quelques décennies, mais il est certain que nos successeurs dans les réunions de rédaction réfléchiront à des projets qui mettront en avant efficacement des technologies passionnantes !

## 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



*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. »*

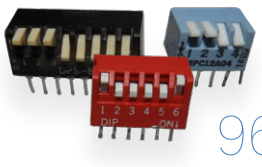
# DANS CE NUMÉRO




## Rubriques



Drôle(s) de composant(s)




## Articles de fond



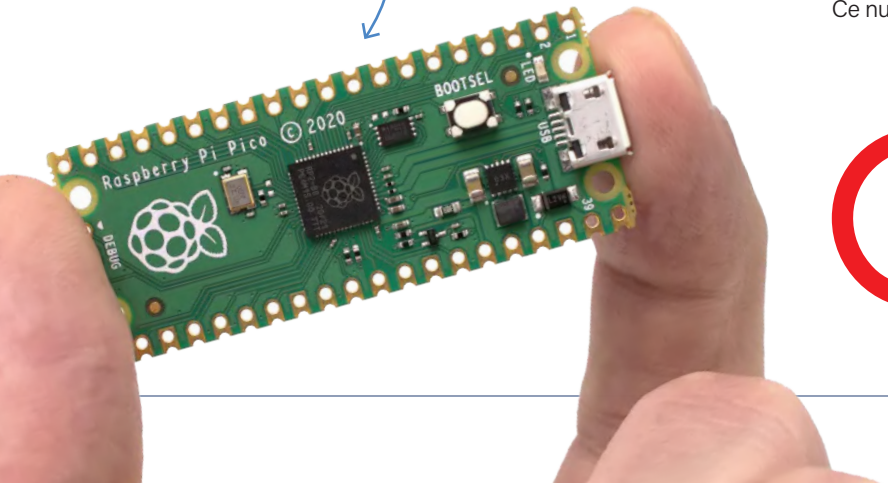
## Réalisations





Tour d'horizon de la carte
**RASPBERRY PI PICO**
à RP2040

57




# Bientôt dans ces pages

**Le numéro de juillet-août 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 :**

- Lévitation magnétique
- Module LoRa avec Raspberry Pi Pico
- Charge électronique pour CC et CA
- Alimentation pour platine d'expérimentation, avec tensions positive et négative, à partir du micro-USB
- Système de caméra DIY pour Raspberry Pi
- Boussole électronique
- MicroPython sur l'ESP32 : pas à pas
- Écrans pour les projets à Raspberry Pi

**et bien d'avantage !**

Ce numéro paraîtra le 1er juillet 2021.

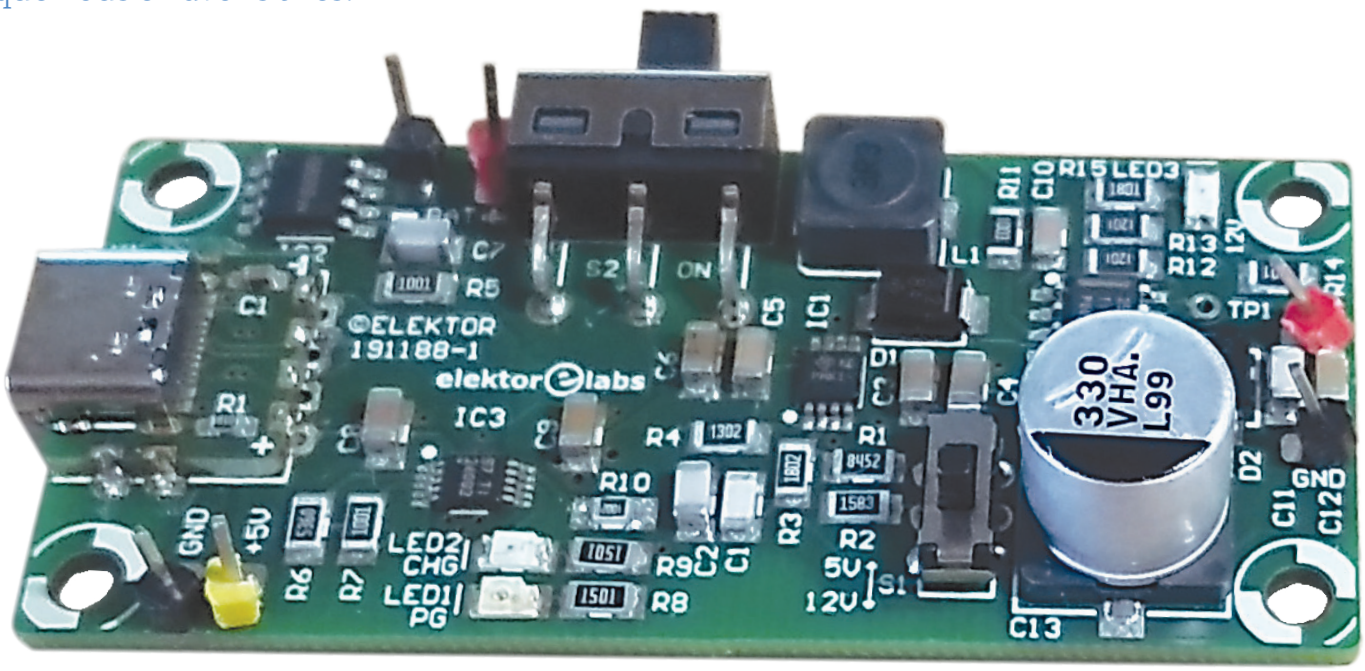
elektor
créer > partager > vendre

# superchargeur LiPo DIY

## De l'artisanat au marché de masse

**Mathias Claußen** (Elektor)

*Elektor* et *GreatScott!*, chaîne YouTube axée sur l'électronique amateur avec plus d'1,4 million d'abonnés présentent une alimentation rechargeable *LiPo* en kit. Ce kit permet aux novices de faire leurs premières armes en *CMS*, avec un choix judicieux de composants 1206 et un circuit imprimé (*PCB*) en partie monté. La conception de ce kit à monter soi-même (*DIY - Do It Yourself*) a été difficile, c'est pourquoi nous voulons partager avec vous les enseignements que nous en avons tirés.




### SPÉCIFICATIONS

- Entrée : 5 V +/−10 %
- Sortie : 5 V / 1,5 A ou 12 V / 0,75 A
- Batterie au lithium

GreatScott! et Elektor se sont réunis au salon *productronica 2019* au sujet d'un projet commun – une alimentation en kit dotée d'une batterie rechargeable au lithium. Pour la plupart de nos projets, nous utilisons une alimentation secteur fiable. Mais pour des réalisations à terme autonomes, par ex. modèle réduit de voiture ou capteur extérieur, les câbles sont vite rédhibitoires. Ces projets utilisent en général un jeu de batteries maintenues par du ruban adhésif et collées à chaud à un convertisseur CC/CC peu coûteux pour former une alimentation mobile. Une alimentation robuste avec protection améliorée serait préférable. GreatScott! et Elektor se sont associés pour concevoir une alimentation rechargeable fonctionnant avec des batteries au lithium courantes. L'équipe savait d'emblée qu'il faudrait du « DIY » et des composants CMS pour enseigner les techniques de soudage CMS et démontrer que le soudage des CMS n'a besoin ni de magie ni de magiciens. Lors de la réunion,

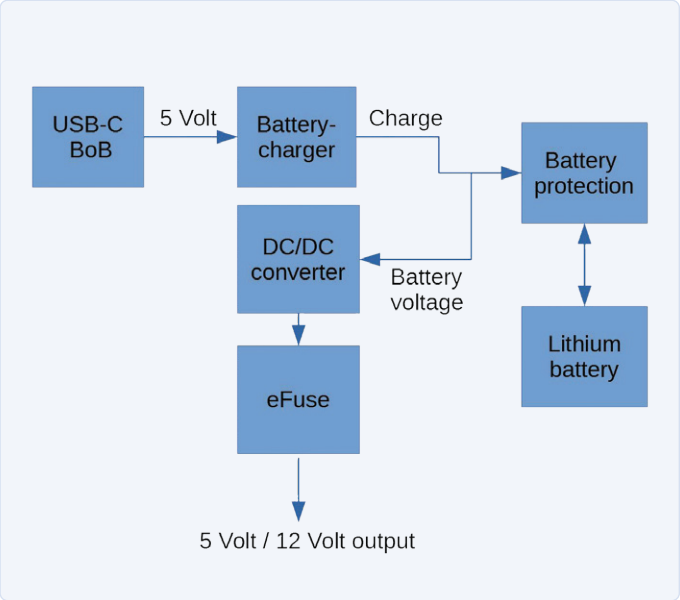



*Figure 1. Blocs fonctionnels.*

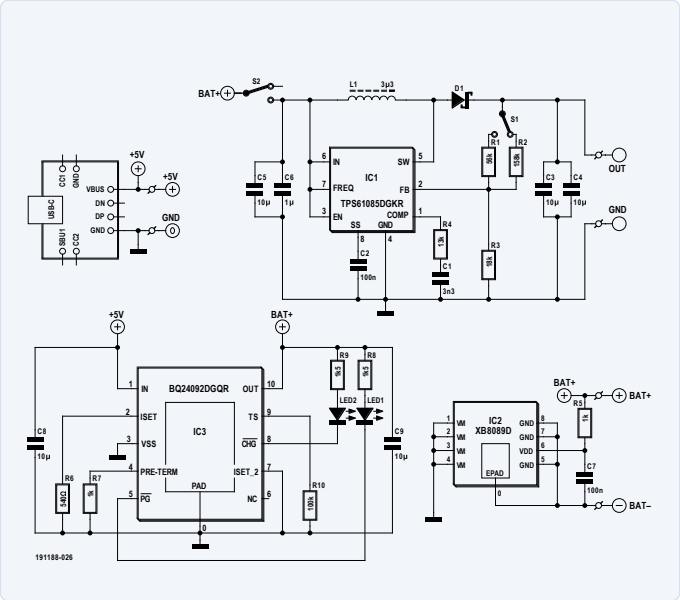


*Figure 2. Schéma de la première version.*

nous avons appris que GreatScott! avait déjà fait quelques calculs et ébauché des schémas. À la différence des autres alimentations, l'utilisateur peut commuter la sortie sur 12 V ou 5 V.

## Blocs fonctionnels

Pour construire une alimentation à batterie au lithium rechargeable, il faut judicieusement relier les blocs fonctionnels. La **figure 1** montre l'interconnexion de ces blocs.

Le premier bloc est le connecteur USB : un adaptateur secteur-USB pourra alimenter l'appareil. C'est un petit circuit imprimé à composants passifs qui distribue les signaux nécessaires sur sa périphérie. Il prendra place sur le circuit imprimé principal. Il peut servir dans d'autres projets utilisant l'USB-C. Il y a ensuite un circuit qui, à partir de la tension d'entrée, charge la batterie au lithium connectée.

La protection de la batterie est assurée par un bloc distinct (**fig. 1**) qui prévient toute surintensité, surcharge et décharge excessives et protège des erreurs de polarité. Au centre figure le convertisseur CC/CC qui fournira une sortie de 5 V ou 12 V. Le dernier bloc est l'*eFuse* qui limite le courant de sortie débité par le convertisseur et le protège de tout dommage. En connectant ces blocs convenablement, nous obtenons une source d'énergie rechargeable portable.

## Des composants courants

La liste des composants a été optimisée pour commander en Chine, car en décembre 2019 cela semblait raisonnablement facile, l'import-export chinois étant capable de traiter de grandes quantités à bas prix. Les composants de cette liste sont pour la plupart également disponibles chez les fournisseurs européens. Les circuits intégrés importants pour ce projet étaient estampillés Texas Instruments. Un seul CI nécessitait un achat chez un distributeur chinois, mais cela ne semble pas être un problème. Nous avons confié la charge à un BQ24092DGQR de Texas Instruments. Il est conçu pour les batteries rechargeables de type lithium-ion et lithium-polymère. Un TPS61085DGKR, également de Texas Instruments régule la tension de sortie de 12 V ou 5 V. Il est monté en convertisseur *boost* CC/CC. Les autres composants (condensateurs, résistances et inductances) étaient proposés par les distributeurs européens. Dès la conception nous avons choisi des composants classiques. Un ingénieur d'Elektor a pris en charge le schéma, la liste des composants et les premiers calculs. Sur cette base, le dessin et le routage d'un circuit imprimé ont été vite réalisés et le 1er prototype fut commandé. Il était prêt à être essayé juste avant Noël 2019. La **figure 2** montre le schéma et la **figure 3** le circuit imprimé qui en résulte. Le premier essai effectué par GreatScott! semblait très prometteur. Le circuit imprimé pouvait être produit, mais tout s'est compliqué. En effet, le COVID-19 a entamé son voyage autour du monde et a « infecté » les chaînes d'approvisionnement. Le nombre de composants disponibles se réduisait comme peau de

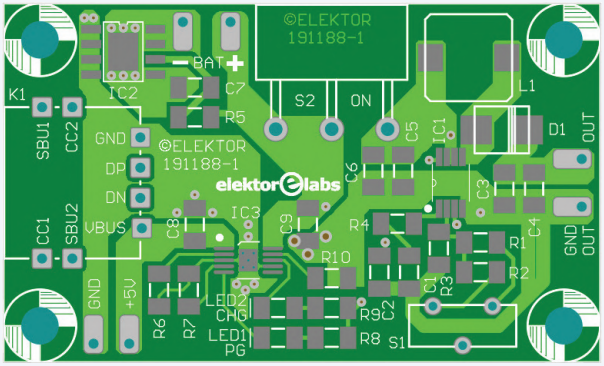



*Figure 3. Circuit imprimé de la première version.*

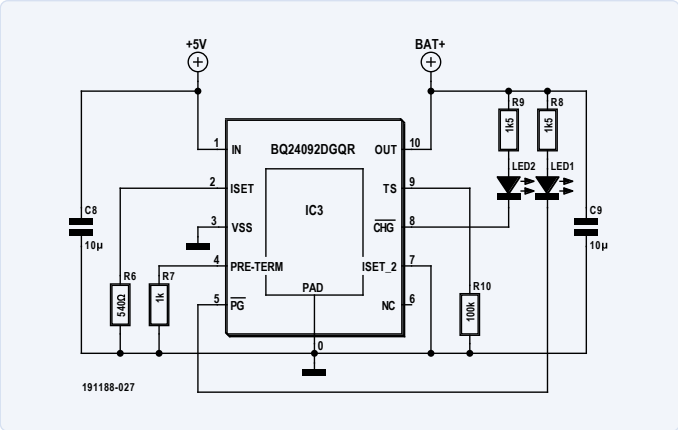



*Figure 4. Schéma du bloc « circuit de charge ».*

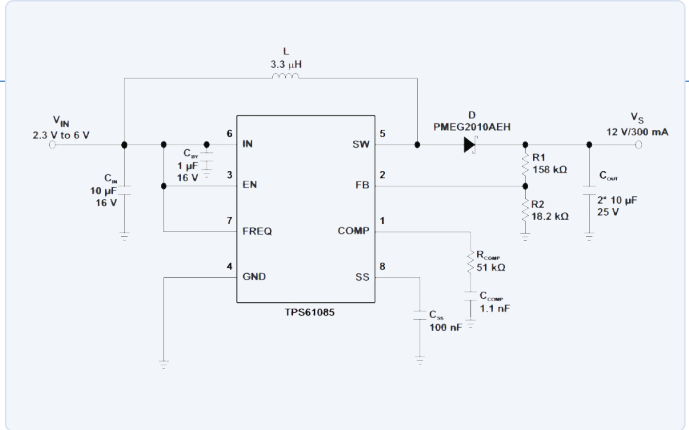



*Figure 5. Schéma du bloc « convertisseur CC/CC ».*

chagrin. De plus, deux erreurs de conception majeures sont apparues.

## Erreur de puissance de 10 et CI qui partent en fumée

Un 1er essai a montré que l'utilisation en alimentation d'une batterie rechargeable fonctionnait, mais la charge non. Nous avions déjà observé ce phénomène sur un autre prototype après l'installation du BQ24092DGQR. La **figure 4** montre le schéma de recharge de la batterie. La résistance R10 est nécessaire s'il n'y a pas de thermistance à l'intérieur de la batterie. Selon le type de BQ2409x choisi, il peut s'agir de 10 ou 100 kΩ, et ici, elle était de dix fois la valeur attendue par le BQ24092DGQR. La batterie ne se chargeait pas, car le CI détectait une température hors de la plage attendue. La solution est triviale : remplacer R10 de 100 kΩ par une résistance de 10 kΩ.

Le deuxième problème était un peu plus grave. Au cours des essais, le convertisseur CC/CC se désintégrait dès que le courant était élevé. Selon le schéma (fig. 1), il n'y a qu'un CI qui protège la batterie, il ne fait que limiter le courant à 10 A. Donc sous 3,7 V, environ 37 W peuvent être délivrés à l'étage d'amplification CC/CC. Dans des conditions idéales, pour 5 V en sortie, cela donnerait 7,4 A max. disponibles. Les conditions n'étant jamais idéales, cela implique que le convertisseur CC/CC peut être surchargé et par conséquent détruit. Il fallait donc protéger ce convertisseur, mais voyons d'abord comment un tel circuit fonctionne.

## Fonctionnement d'un convertisseur CC/CC

Un convertisseur CC/CC doit produire une tension de sortie inférieure ou supérieure à la tension d'entrée. En alternatif, c'est plus simple, car on peut utiliser un transformateur et la tension de sortie dépend du rapport de spires primaire/secondaire. Les convertisseurs CC/CC existaient avant l'arrivée des CI au silicium. Électromécaniques, ils étaient des dizaines de fois plus volumineux que ce que propose l'électronique mobile moderne. Le site Wikipédia [1] vous en dira plus.

Un convertisseur CC/CC moderne ne nécessite que quelques composants externes, car ses circuits sont maintenant intégrés dans un boîtier minuscule. Si le convertisseur abaisse la tension, il est dit abaisseur (*buck*), ou dévolteur c'est le plus courant. Si le convertisseur élève la tension, il est dit élévateur (*boost*) ou survolteur. Ici c'est un convertisseur *boost* TPS61085DGKR qui produit 5 ou 12 V à partir d'une entrée de 3 à 4,2 V. Pour décrire son fonctionnement, examinons d'abord la **figure 5**. De l'entrée à la sortie, il y a une inductance L suivie d'une diode D. L'inductance est également connectée à la broche SW, un interrupteur interne qui peut connecter l'inductance à la terre. Pour commencer, supposons que l'interrupteur interne SW est fermé et que le courant va à la terre via l'inductance et la broche SW. L'inductance stocke de l'énergie sous forme de champ magnétique. Si l'interrupteur est maintenant ouvert, le courant dans l'inductance diminue, ce qui inverse la polarité à ses bornes, et un courant traverse la charge tandis que le champ magnétique diminue (restitution de l'énergie). L'inversion de polarité place de fait l'inductance en série avec la batterie ce qui double la tension disponible pour la diode (comme si on avait mis deux batteries identiques en série).

La tension d'alimentation et la tension ajoutée par l'inductance passent par la diode D. Cette tension charge le condensateur Cout. Quand le champ magnétique a diminué au point que l'inductance n'ajoute plus rien à la sortie, on referme l'interrupteur interne. Le champ magnétique de l'inductance remonte à son état initial dès que le courant circule à nouveau dans la bobine et l'interrupteur SW. Le processus peut alors recommencer en ouvrant l'interrupteur.

La diode D empêche le condensateur Cout de se décharger tant que l'interrupteur est fermé et relié à la terre. En répétant le cycle en continu, on obtient une tension de sortie supérieure à la tension d'entrée.

Examinons la **figure 6** qui décrit les blocs internes du CI du convertisseur CC/CC. Voyons le contrôle des durées nécessaires au mode « boost » pour obtenir une tension de sortie stable. Le FET interne joue le rôle de l'interrupteur qui relie la sortie de l'inductance à la terre. Pour synchroniser correctement le FET interne, la tension de sortie est comparée à une tension Vref, ici 1,238 V. Il faut calculer un diviseur de tension relié à l'entrée FB (*Feed Back* = rétroaction) qui renvoie 1,238 V quand notre sortie est à 12 V ou 5 V. Si la tension de rétroaction est supérieure à 1,238 V, la logique interne tente de réduire la tension de sortie jusqu'à ce que la broche de rétroaction soit à 1,238 V. Si la rétroaction est inférieure à 1,238 V, la tension de sortie augmente jusqu'à ce que 1,238 V soient à nouveau atteints. Des protections internes limitent la quantité de courant traversant le FET interne, verrouillent le CI en cas de sous-tension, et sont censées arrêter le convertisseur CC/CC si la température du CI s'élève trop. Voilà pour le principe du fonctionnement du convertisseur CC/CC.

## Surintensité et protection

Si la sortie du convertisseur CC/CC est connectée à une charge élevée ou court-circuitée, cela surcharge les FET et des circuits internes, et détruit donc brutalement le CI. Pour éviter cela, il y a deux endroits où on peut ajouter une protection, devant le convertisseur CC/CC et à sa sortie. Avec très peu de composants, il peut sembler évident et rapide de limiter courant et puissance délivrés au convertisseur CC/CC. Comme élément de protection, un composant polymère à coefficient de température positif (fusible réarmable CTP) a été inséré pour limiter le courant en

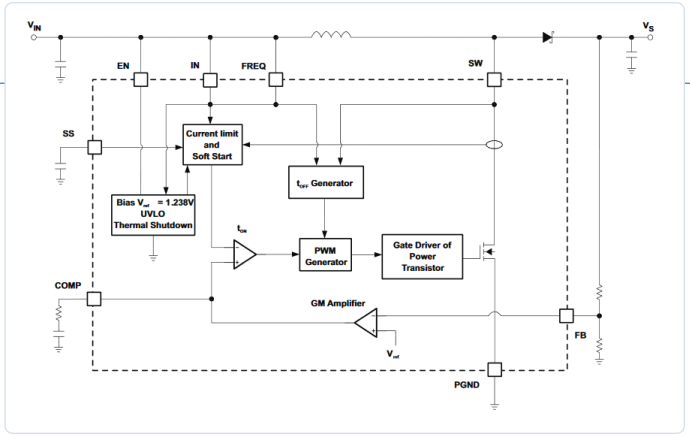



*Figure 6. Schéma de principe du convertisseur CC/CC.*

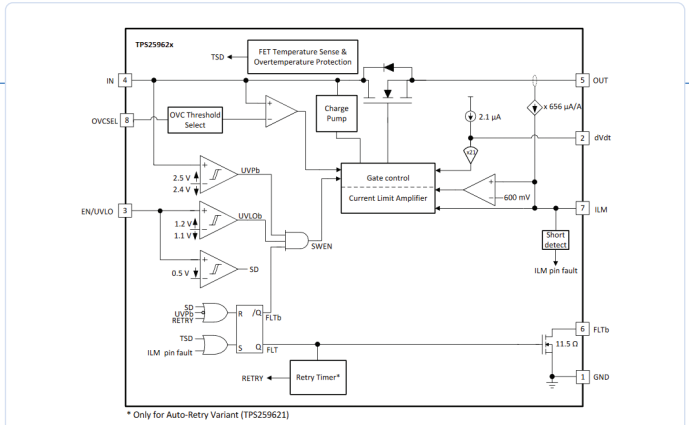



*Figure 7. Schéma de principe de l'eFuse.*

amont du convertisseur. Même si cela semble rapide et peu coûteux pour le protéger, le principe de fonctionnement d'un convertisseur CC/CC induit dans ce cas des effets secondaires indésirables. À des fins éducatives, un fusible réarmable a été ajouté au circuit imprimé. Cela a bien protégé le convertisseur CC/CC à court terme, mais a engendré des oscillations indésirables. Il n'y a pas besoin d'effectuer de mesures ni d'essais pour en déterminer la raison, il suffit de réfléchir. Un fusible réarmable limite l'intensité du courant, car sa résistance interne augmente avec la température. Comme le fusible lui-même a une résistance définie, le courant qui le traverse le fait chauffer. Par conséquent, plus le courant traversant le fusible est élevé, plus sa température augmente et plus sa résistance augmente. En fin de compte, cette résistance limite l'intensité du courant qui passe à travers le fusible.

En supposant que le convertisseur ait une charge de sortie élevée, le courant d'entrée est lui aussi élevé. Le fusible va alors s'échauffer et sa résistance augmenter, limitant le courant qui va circuler. Qui dit augmentation de résistance du fusible, dit chute de tension accrue à ses bornes et réduction de la tension d'entrée du convertisseur. Le convertisseur CC/CC voit son courant limité et ne peut plus être détruit, mais il tente aussi de maintenir sa tension de sortie et de fournir la puissance demandée avec une tension d'entrée plus basse, ce qui demande une augmentation du courant dont il a besoin. Celle-ci entraîne une nouvelle augmentation de la résistance du fusible et donc de la chute de tension aux bornes du fusible.

De nouveau, la tension d'entrée du convertisseur se réduit. Très vite, la tension d'entrée tombe au-dessous du seuil de verrouillage de sous-tension et le convertisseur CC/CC s'arrête. Une fois arrêté, l'appel de courant est fortement réduit. Cela permet au fusible de refroidir et sa résistance diminue. Par conséquent, la chute de tension sur le fusible diminue et la tension d'entrée du convertisseur CC/CC réaugmente. Le convertisseur redémarre et la boucle étant bouclée ce qui doit arriver arrive, ça oscille. Cela a aussi des conséquences sur la tension de sortie. Comme pour toute oscillation, la tension de sortie montre des pointes et autres instabilités non filtrées.

Si l'ajout d'un fusible réarmable en entrée n'est pas la solution idéale, en sortie c'en serait une. Ce serait une approche simple avec une seule tension et une seule limite de courant. Nous avons deux limites de tension et de courant de sortie différentes, un seul fusible ne peut pas protéger pour les deux. Un fusible électronique (*eFuse*), déjà présenté dans un article en ligne d'*Elektor* [2] peut être une solution plus générale à ce problème, notamment en cas de variation de tension et de courant. La recherche d'*eFuses* s'adaptant à une tension d'entrée de 5 à 12 V et capables de gérer 0,5 à 2 A, nous a conduits au TPS259621DDAR de Texas Instruments. Comme on le voit sur la **figure 7**, ces composants offrent plus qu'une simple protection contre les surintensités. Ils peuvent aussi protéger des surtensions et des sous-tensions. Sur la figure 2, un inverseur pour changer les tensions est déjà présent. Comme la limite de courant est également fixée par des résistances et pilotée par les tensions définies pour protéger le convertisseur CC/CC, avoir un commutateur qui gère deux voies indépendantes permet de définir à la fois la tension de sortie et la limite de courant. Nous avons trouvé et utilisé de tels commutateurs pour ce projet. C'est une solution moderne et réglable pour protéger contre les surintensités.

*Figure 8. Batterie plate endommagée et gonflée.*

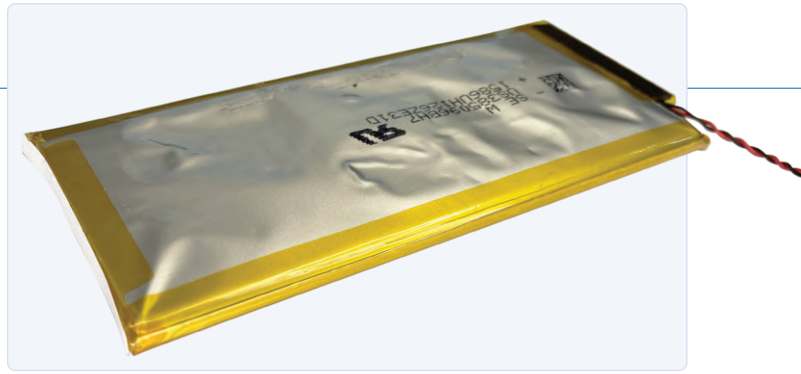


*Figure 9. Batterie de téléphone portable endommagée.*

## Batteries au lithium : faciles à recharger et à enflammer !

De nos jours, les appareils électroniques portables utilisent en général des batteries à la chimie basée sur le lithium. Grâce aux progrès techniques de ces batteries, la densité de puissance par volume et par poids a bien augmenté. Les batteries antérieures (NiCd et NiMh) nécessitaient des circuits complexes pour être rechargées et détecter la fin de charge, le lithium simplifie les choses. Si la pile n'est pas épuisée et que sa tension à vide est ≥2,5 V, la recharge se fait selon une approche CC/CV à courant constant d'abord, avec une tension de fin de charge, puis arrêt de la charge dès qu'une tension définie est atteinte. Pour le lithium-ion et le lithium-polymère, la tension type est de 4,15 V. C'est ce que fait le BQ24092DGQR pour recharger une batterie connectée, en appliquant un courant limité ≤1 A jusqu'à atteindre une tension de cellule de 4,15 V. La puce lance en outre une temporisation de charge qui, une fois écoulée, arrête la charge si la batterie n'a pas atteint 4,15 V.

Le circuit intégré de charge offre en outre une protection thermique optionnelle. Les chargeurs et batteries de haute qualité sont équipés d'une surveillance thermique, généralement une thermistance, pour éviter un échauffement excessif des batteries en cours de charge. Vu la polyvalence d'utilisation recherchée (divers types et marques de batteries au lithium), l'exploiter ici n'est pas utile. Si cette surveillance n'est pas utilisée, le chargeur fonctionnera sans elle en installant une résistance ordinaire à la place de la thermistance. Il faut juste choisir la bonne résistance ; sinon, il n'y aura pas de charge. L'utilisation de batteries au lithium incompatibles, par ex. au phosphate de lithium-ion (LiFePo), peut provoquer leur l'inflammation. Il faut utiliser des cellules de haute qualité, car une cellule endommagée se met à gonfler ou même à chauffer et peut prendre feu. La **figure 8** montre des batteries au lithium endommagées ayant gonflé. Plates à l'origine elles sont devenues presque rondes. La **figure 9** montre une cellule de téléphone portable endommagée, légèrement dilatée, mais avec suffisamment de force pour décoller l'écran du téléphone de son châssis.

## Spécifications finales

Avec l'ajout de l'*eFuse* au schéma de circuit, il fallait également définir les paramètres de

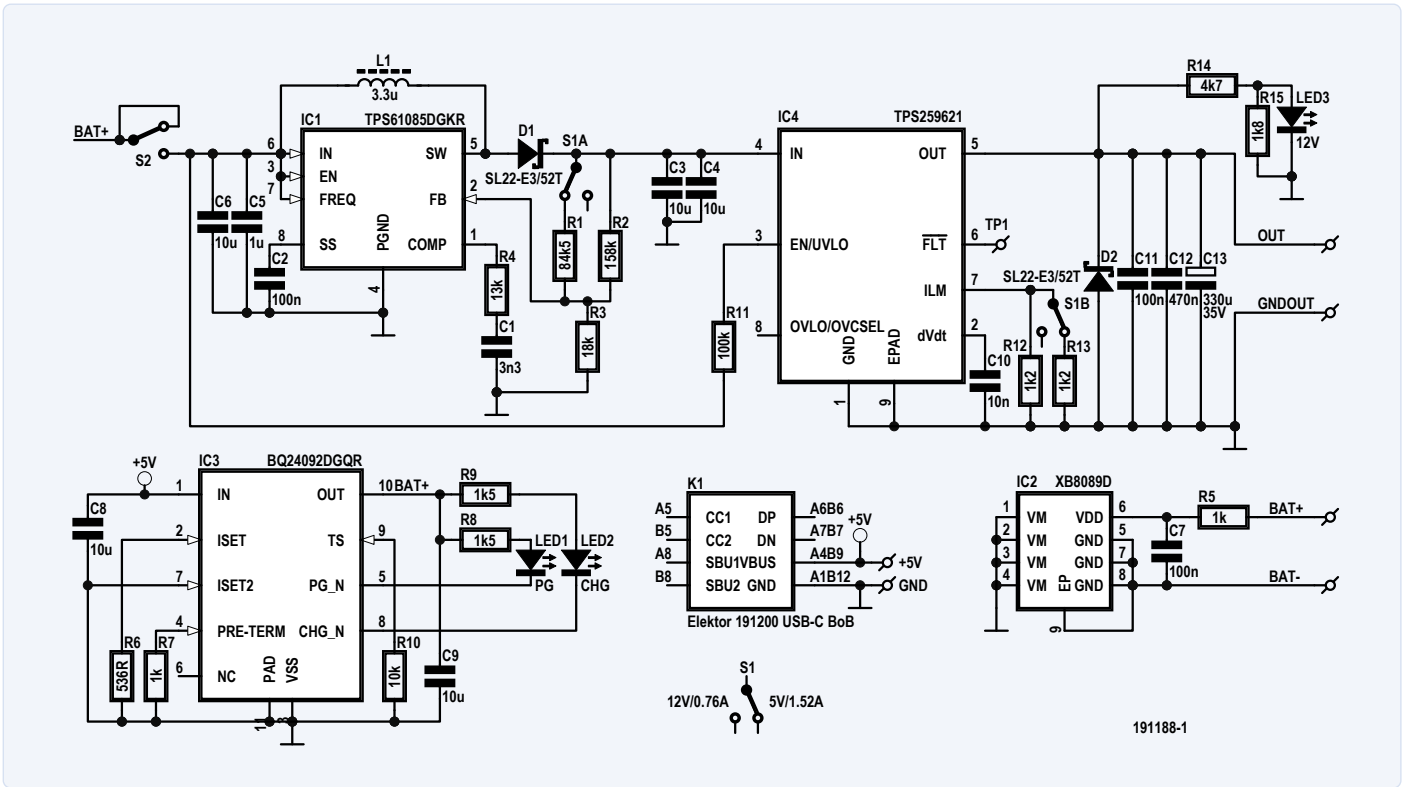



*Figure 10. Schéma du superchargeur LiPo DIY en kit.*

l'*eFuse*. Nous avons fixé les valeurs pour 5 et 12 V à 1,5 et 0,75 A. C'est sans risque pour le convertisseur. Il faut également définir le courant de charge utilisé pour la batterie connectée. Le BQ24092DGQR offre jusqu'à deux réglages de courant de charge, selon la configuration des broches ISET et ISET2. Comme ISET2 est reliée à la terre, la puce utilisera le réglage de courant d'ISET, et avec 536 Ω, le courant de charge de la batterie connectée est limité à 1 A. Assurez-vous de monter une batterie compatible avec le courant de charge.

## Retour sur le schéma et finalisation du circuit imprimé

Après l'ajout de l'*eFuse* au schéma, un point restait à examiner. Le schéma initial ne prévoyait pas de moyen visuel de savoir si la tension de sortie valait 5 V ou 12 V. GreatScott! et Elektor sont tombés d'accord : une LED indicatrice améliorerait la convivialité du montage. Nous avons aussi pensé que ce retour visuel aiderait à éviter d'alimenter accidentellement en 12 V un système prévu pour 5 V. Le circuit ajouté pour signaler le mode 12 V utilise une LED et deux résistances. La LED ne s'allume que si la sortie est à 12 V . La 2e version du circuit imprimé a fonctionné comme prévu. Il a fallu ensuite achever la conception, ajouter les dernières retouches au texte et au logo, placer les références idoines et remettre tout le matériel à la production. La **figure 10** montre le schéma final du super-chargeur LiPo DIY. La **figure 11** est le schéma de la carte additionnelle (BoB) USB-C.

## De l'artisanat au marché de masse

La conception de montages et la réalisation de prototypes en interne diffèrent d'une production par le biais d'installations automatisées et de prestataires de services. Nous n'avons dit mot du petit BoB USB-C (**fig. 12**) monté sur le circuit imprimé principal et qui a son propre numéro de projet. Une connexion USB-C *alimentation et données* serait utile pour d'autres montages, un petit circuit imprimé a également été dessiné et produit à cet effet. Les connexions importantes pour la vitesse et l'alimentation USB2.0 ont été acheminées sur les trois côtés sous forme de trous crénelés.

Cela les a rendus pratiques à utiliser, mais pas si faciles à produire. Les circuits imprimés sont si petits que les machines ont du mal à les manipuler. Pour qu'elles y parviennent, le prestataire a dû prendre certaines mesures techniques. Finalement, ils ont été produits et

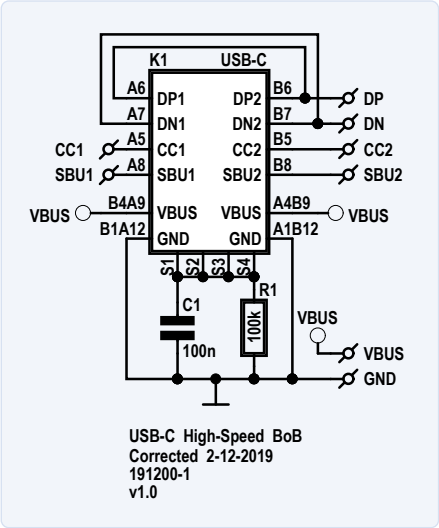



*Figure 11. Schéma du BoB USB-C.*

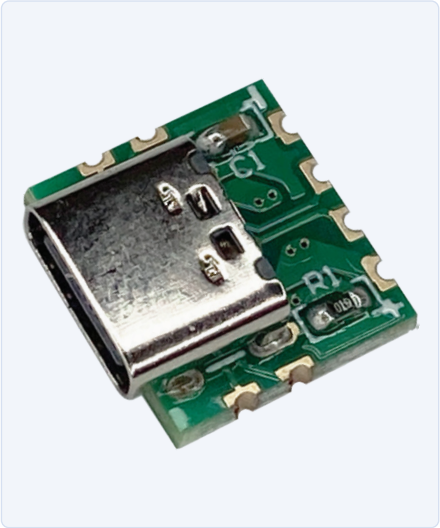


*Figure 12. BoB USB-C.*

font partie du kit. Pour fabriquer ce minuscule circuit imprimé à trois côtés à trous crénelés, il fallait d'abord vérifier auprès du prestataire et obtenir un devis pour les coûts additionnels. Dans un kit DIY CMS, il faut aussi que les composants soient emballés ou rangés à l'intérieur du kit. Manuel, ce travail peut faire s'envoler les coûts. À la fin, il peut être moins cher de proposer un circuit imprimé garni au lieu du kit en pièces détachées. Ici c'est différent, car l'apprentissage du soudage des CMS implique de ne pas souder en usine les CMS qui peuvent l'être avec un simple fer à souder. Fournir les fichiers Gerber et les données de placement automatique à un fabricant peut aussi être une aventure.

Nous avons reçu un prototype presque à la phase finale du processus. Si résistances, inductances, LED et autres pièces mécaniques étaient bien emballées dans des sachets, le circuit imprimé nous réservait une surprise : alors que les composants à préassembler étaient bien là, un problème affectait la pâte à souder. Cette pâte à souder pour circuit imprimé (chose que la plupart de ceux qui montent encore leurs circuits imprimés à la main ne connaissent pas) avait été appliquée pour tous les composants. Il y en avait donc sur toutes les pastilles du circuit imprimé, ce qui ne facilite pas du tout le soudage des composants restants. La **figure 13** vous en dira plus qu'un long discours. Il a fallu revoir le calque de pâte à souder du jeu de fichiers Gerber, de sorte que pendant la fabrication, seules les pastilles des composants préassemblés reçoivent cette pâte. Pour la production, il faut également discuter à l'avance avec votre fabricant et votre prestataire de services de la quantité de pièces CMS à emballer dans de petits sacs. L'emballage, c'est de la main-d'œuvre qui peut grever les coûts de façon étonnante.

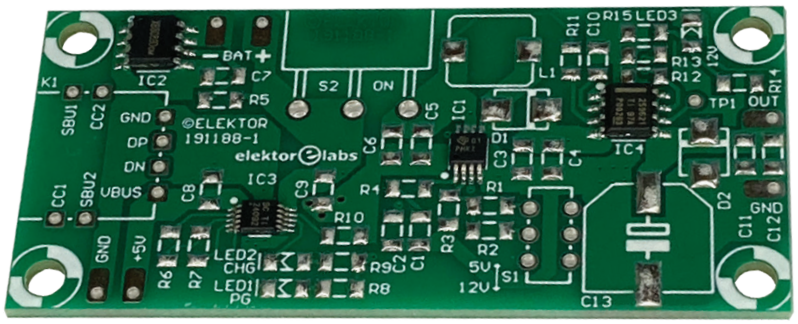



*Figure 13. Circuit imprimé fabriqué avec le masque de soudure erroné.*


*Figure 14. Emballage du kit « superchargeur LiPo DIY ».*


*Figure 15. Pièces du kit « superchargeur LiPo DIY ».*

**USB-C *Power Delivery* (PD) et résistances nécessaires**

Si un connecteur USB-C branché sur une alimentation classique de 5 V / 1 A avec sortie USB-A fonctionne en recharge avec un câble adapté, c'est faux pour les connecteurs de type *USB-C **PD***. Avec le micro-USB, l'énergie est disponible entre VCC et la terre. Ce n'est pas vrai pour l'USB-C PD. Le superchargeur LiPo DIY a été conçu selon les formules éprouvées des connexions micro-USB, sans tenir compte que pour les alimentations USB-C PD, deux résistances supplémentaires sont nécessaires pour déterminer la puissance. Cela signifie que les adaptateurs USB-C PD ne fonctionneront pas avec la carte.

## Une boîte et un manuel

Il est temps de se pencher sur le produit final. La **figure 14** montre le kit en boîte. Une fois toutes les pièces du kit produites et emballées, difficile de se contenter d'expédier un sac et un petit manuel. Si vous passez du prototype artisanal à la production de masse, vous devez également considérer la nature de l'emballage et le mode de fourniture du manuel au client. Selon le pays destinataire, il peut y avoir certaines réglementations concernant l'impression de manuels dans les langues appropriées. Le kit du superchargeur LiPo DIY comprend tous les composants nécessaires (triés dans des sacs étiquetés) et un QR code imprimé avec lien vers les instructions. N'inclure que le QR code a permis d'économiser des ressources et de contribuer à garder la planète un peu plus verte. Consultez la **figure 15** pour voir le contenu du kit. Vous trouverez sur YouTube [3] une vidéo de montage de GreatScott! Et si vous avez des doutes sur le soudage de composants CMS, consultez le tutoriel vidéo [4]. Ce kit n'inclut pas de batterie rechargeable. Mais en ligne c'est facile à trouver ou bien utilisez

## LISTE DES COMPOSANTS DU KIT DU SUPERCHARGEUR LIPO DIY

### Résistances

R1 = 84,5 kΩ, 1 %, 0,25 W, CMS 1206
R2 = 158 kΩ, 1 %, 0,25 W, CMS 1206
R3 = 18 kΩ, 1 %, 0,25 W, CMS 1206
R4 = 13 kΩ, 1 %, 0,25 W, CMS 1206
R5,R7 = 1 kΩ, 1 %, 0,25 W, CMS 1206
R6 = 536 Ω, 1 %, 0,25 W, CMS 1206
R8,R9 = 1,5 kΩ, 1 %, 0,25 W, CMS 1206
R10 = 10 kΩ, 1 %, 0,25 W, CMS 1206
R11 = 100 kΩ, 1 %, 0,25 W, CMS 1206
R12,R13 = 1,2 kΩ, 1 %, 0,25 W, CMS 1206
R14 = 4,7 kΩ, 1 %, 0,25 W, CMS 1206
R15 = 1,8 kΩ, 1 %, 0,25 W, CMS 1206

### Inducteurs

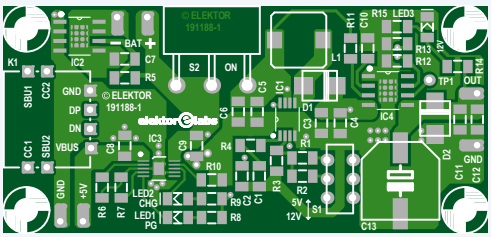
L1 = 3,3 µH, 20 %, 3,5 A, 35 mΩ, CMS 7,3×7,3×4,5 mm

### Condensateurs

C1 = 3,3 nF, 5 %, 50 V, NP0, CMS 1206
C2,C7,C11 = 100 nF, 5 %, 50 V, C0G, CMS 1206
C3,C4,C6,C8,C9 = 10 µF, 10 %, 25 V, X7R, CMS 1206
C5 = 1 µF, 10 %, 50 V, X7R, CMS 1206
C10 = 10 nF, 5 %, 50 V, X7R, CMS 1206
C12 = 470 nF, 10 %, 50 V, X7R, CMS 1206
C13 = 330 µF, 20 %, 35 V, CMS 10,3x10,3

### Semi-conducteurs

D1,D2 = SL22-E3/52T, CMS SMB
LED1 = 11-21/GPC-AM2P1/2T, LED verte, CMS 1206
LED2,LED3 = 15-21SDRC/S530-A2/TR8, LED rouge, CMS 1206
IC1 = TPS61085DGKR, CMS VSSOP-8



IC2 = XB8089D, CMS SOIC-8-EP
IC3 = BQ24092DGQR, CMS MSOP-10-EP
IC4 = TPS259621DDAR, CMS SO-PowerPad-8

### Autres

PC1 à PC6 = barrette mâle, 1×1
S1 = interrupteur DPDT, THT, 9,1×3,5 mm (K2-2235D-F1)
S2 = interrupteur SPDT, 250 VCA, 3 A (XKB, SS-12D06L5)
K1 = BoB USB-C, Elektor 191200-1
Circuit imprimé 191188-1 v2.1

## LISTE DES COMPOSANTS DU BOB USB-C

### Résistances

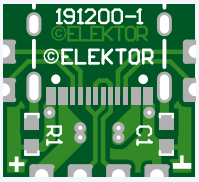
R1 = 100 k Ω, 1 %, 100 mW, CMS 0603

### Condensateurs

C1 = 100 nF, 10 %, 100 V, X7R, CMS 0603

### Autres

K1 = USB – type C, femelle, courant nominal 3 A, CMS/THT
Circuit imprimé 11200-1 v1.0




*agrandissement à 200%*

une batterie en bon état si vous en avez sous la main. Vérifiez que cette batterie lithium-ion ou lithium-polymère supporte un courant de charge de 1 A et délivre 3,7 V nominaux (un seul élément).
Pour l'assemblage, prenez votre temps et soyez patient. Même en format 1206, les composants CMS ont une forte propension à disparaître en tombant de l'établi ou à se cacher derrière les objets. Une fois le montage et les tests terminés, votre superchargeur LiPo sera prêt à fonctionner. L'interrupteur S2 du circuit imprimé sert à connecter/déconnecter la batterie du convertisseur CC/CC. Cela permet de couper la sortie et d'économiser de l'énergie. Un convertisseur CC/CC, même non chargé peut consommer quelques mA. Une fois votre batterie branchée, mieux vaut la laisser se charger complètement. Sur le superchargeur DIY, la LED1 indique si la tension fournie par le connecteur USB-C est dans la bonne plage. Pendant la charge, vous verrez que la LED2 s'allume. Avec une batterie bien chargée, elle ne s'allume plus. Si c'est le cas, votre source d'énergie portable est prête à alimenter vos appareils.

## Sortez des sentiers battus !

Avoir sous la main une alimentation portable est très utile, en particulier si sa sortie est commutable en 5/12 V. La **figure 16** montre un Raspberry Pi Zero alimenté par le superchargeur LiPo DIY. Si votre budget est serré et que vous ne voulez rien souder, vous pouvez vous contenter de colle thermofusible, ruban adhésif, fils volants et autres composants chinois bon marché. Avec ce kit, vous assemblerez tout vous-même, et l'expérimentation de la soudure CMS DIY le rend particulièrement attrayant. Avoir un produit tout assemblé et prêt à fonctionner à peine sorti de son sac plastique est beaucoup moins gratifiant que de le monter soi-même. L'assemblage de ce kit tient plus de la boîte de Lego®. Un produit sorti tout fait d'une boîte comme par magie ce n'était pas pour vous ! Ce long chemin, c'était pour le plaisir, et au bout, vous disposez d'une alimentation portable, pratique à utiliser partout.
Si vous cherchez les schémas, vous les trouverez sur la page Elektor Labs de ce projet [5]. Si vous avez besoin d'aide, vous pouvez y laisser un commentaire.

(191188-B-03)

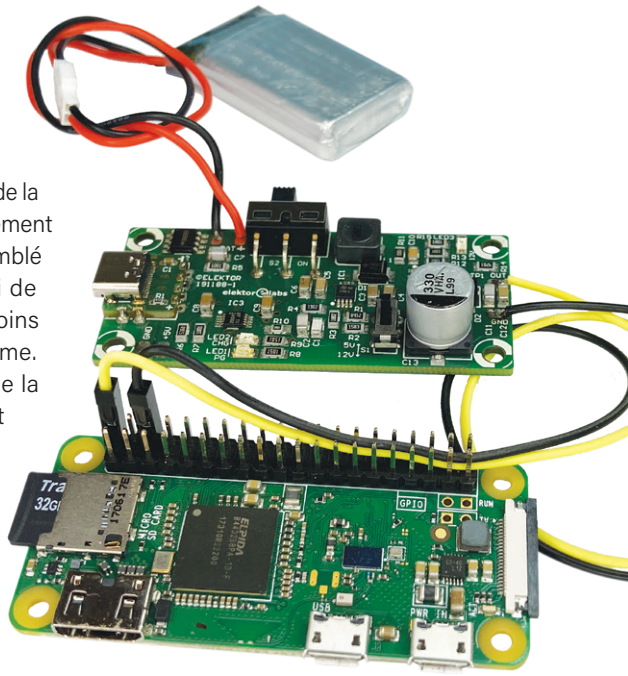


*Figure 16. Raspberry Pi Zero alimenté par le superchargeur LiPo DIY.*

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


**Contributeurs**
Idée : **GreatScott!**
Conception : **Ton Giesberts**
Auteur : **Mathias Claußen**
Rédaction : **Jens Nickel et C. J. Abate**
Illustrations : **Patrick Wielders**
Mise en page : **Giel Dols**
Traduction : **Yves Georges**




## LIENS


[1] **Convertisseur CC/CC à vibreur** : www.club-des-collectionneurs.com/Les-Vibreurs.htm
[2] **S. Cording, « The Modern Fuse », ElektorMagazine.com, 8/12/2020** : www.elektormagazine.com/articles/modern-fuse
[3] **GreatScott!, « DIY LiPo Supercharger ! (Charge, Protection, 5 V/12 V Boost V2) », 2020** : https://youtu.be/6LxRnf6sQNQ
[4] **GreatScott !, « How to Solder Properly : Through-hole (THT) & Surface-Mount (CMS) », 2017** : www.youtube.com/watch?v=VxMV6wGS3NY
[5] **Page Elektor Labs** : www.elektormagazine.fr/labs/diy-lipo-supercharger-kit-by-greatscott
[6] **En savoir plus sur les CMS** : wiki.jelectronique.com/doku.php?id=cms

# 60 ans

## Réflexions sur six décennies d'électronique


Regardez la première couverture d'*Electronica Wereld*. Ce nom – en français « le Monde électronique » – fut plus tard remplacé par *Elektuur*, évoquant « e-lecture ». En 1970, la première édition allemande fut publiée sous le nom d'Elektor.

**Jens Nickel** (Elektor)

*Elektor* a soixante ans ! C'est considérable pour un magazine d'électronique. Cette discipline évolue si vite qu'il peut être difficile d'en suivre tous les développements. Depuis 1961, nous avons rendu compte des technologies, appareils, cartes et composants les plus récents en équilibrant théorie et pratique ; nous avons partagé nos idées et projets avec des ingénieurs professionnels et les fabricants. Pour marquer cet anniversaire, à notre demande, certains de nos auteurs et rédacteurs ont sélectionné les projets et articles qu'ils considèrent comme les plus intéressants de ces six décennies d'*Elektor*.

# d'Elektor

Notre célèbre magazine a été lancé aux *Pays-Bas* en avril 1961. À n'en pas douter, personne n'est mieux placé pour vous en parler que *Harry Baggen*, qui fut pendant des décennies rédacteur en chef de l'édition néerlandaise et, de nombreuses années durant, celui de l'édition internationale.

*« Au lancement de son propre magazine d'électronique sous le nom d'**Electronica Wereld**, Bob W. van der Horst avait pour objectif de faire un magazine meilleur que l'offre existante. Il voulait dissiper le voile de mystère qui entourait cette discipline et surtout montrer son côté pratique. Au début, le magazine proposait principalement un éventail de critiques de produits et d'articles techniques de fond, mais le côté réalisations pratiques l'a emporté en quelques années. Il en est resté ainsi jusqu'à ce jour. Aucun exposé théorique sans fin, mais des réalisations avec des composants modernes, tel fut et tel reste l'objectif. »*

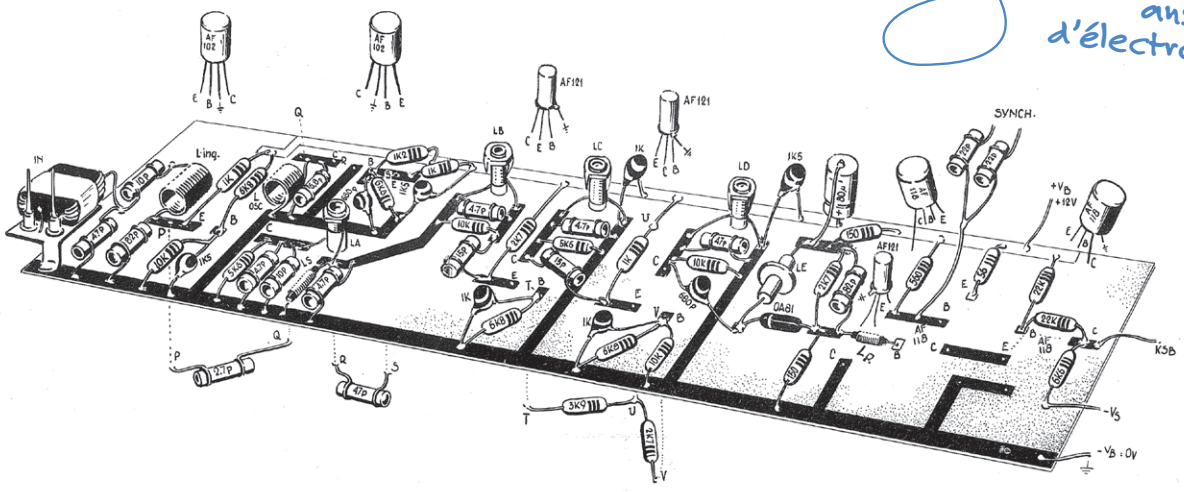
*« Bob voulait aussi mettre en œuvre des composants d'avant-garde. Les premières années, le magazine publiait principalement des circuits à lampes, mais très vite les semi-conducteurs ont pris leur place. Outre les transistors PNP/NPN, les FET étaient souvent utilisés. Elektor a vite été perçu comme obstiné, transversal, novateur. L'objectif de Van der Horst était atteint ! »*

Le premier numéro incluait plus de 50 circuits, par ex. un distorsiomètre et un émetteur à un seul transistor (à télécharger gratuitement sur *www.elektormagazine.fr/210025-04*). La suite fut un foisonnement ininterrompu de projets. Les pages ci-dessous vous en proposent un bel aperçu accompagné de commentaires d'experts et d'auteurs.

Les membres d'Elektor peuvent télécharger ces articles via les liens web fournis. Pour célébrer le 60e anniversaire d'Elektor, sept de ces 'Choix de la rédaction' sont en libre accès jusqu'au 30 juin 2021. Profitez-en.

(210025-04 – VF : Y. Georges)

60 ans d'électronique




*L'ADN d'Elektor, c'est la passion de l'approche pratique de l'électronique. Dès le premier jour, nous voulions que nos lecteurs puissent construire et apprendre en même temps. Cela explique que les membres de la communauté Elektor aient toujours plébiscité nos projets.*

## Harry Baggen

(ancien rédacteur en chef)

### Amplificateur Edwin (1975)

Dès le début, ses projets audio sortant des sentiers battus firent connaître Elektor, et c'est toujours vrai aujourd'hui. L'amplificateur audio Edwin de 1970, très simple (par rapport à ce qui se fait aujourd'hui) innovait avec son étage de sortie à courant de repos nul. Il est devenu un classique, c'est l'un des amplificateurs audio les plus copiés du siècle dernier.

**www.elektormagazine.com/magazine/elektor-197509/57506**

### Elektorscope (1976)

Dans les années 70, les oscilloscopes étaient hors de portée des amateurs. C'était une bonne raison pour l'équipe d'Elektor d'alors de concevoir et de publier un oscilloscope maison. À l'époque, c'était difficile, notamment parce que le tube de l'oscilloscope, sa commande et la haute tension devaient être soigneusement réglés et que beaucoup de ces pièces (surtout le tube) étaient introuvables. Néanmoins, il fut possible de présenter un circuit sûr, susceptible d'être reproduit.

**www.elektormagazine.com/magazine/elektor-197612/57794**

### Fréquencemètre commandé par microprocesseur (1985)

Au début des années 1980, Elektor présenta une série d'appareils de mesure dont le fréquencemètre commandé par microprocesseur était le fleuron. Les caractéristiques et fonctions de ce bijou pouvaient rivaliser avec les appareils professionnels : gamme de 0,01 Hz à 1,2 GHz, système d'autoréglage, touches à effleurement et écran alphanumérique.

**www.elektormagazine.fr/magazine/elektor-198501/52433**

### Table traçante (1988)

À une époque où on ne trouvait les traceurs XY que dans les labos et bureaux d'études, *Elektuur* publia la réalisation d'un traceur très simple à construire soi-même. Le prix des pièces de ce traceur était dérisoire par rapport à un vrai traceur. Il utilisait des feutres pour le tracé. Un pilote HP-GL spécifique est même ensuite apparu.

**www.elektormagazine.fr/magazine/elektor-198801/53141**

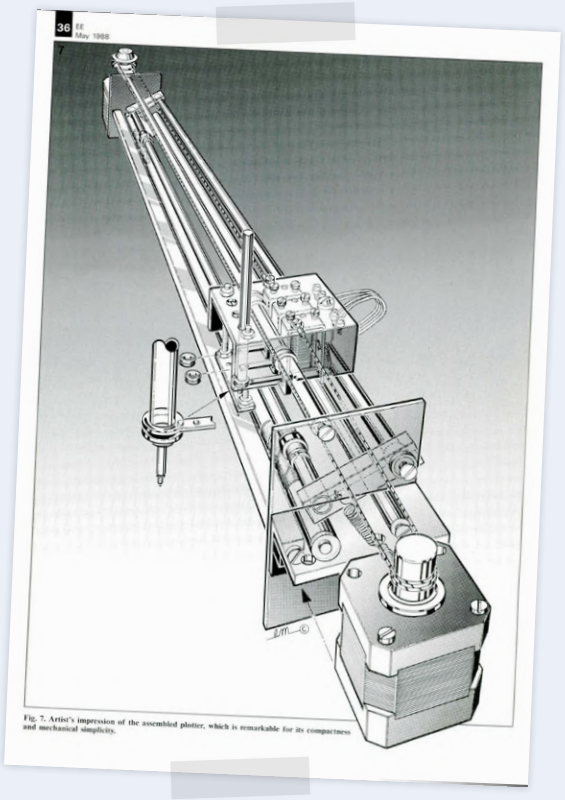



Fig. 7. Artist's impression of the assembled plotter, which is remarkable for its compactness and mechanical simplicity.

### Processeur de son surround (1995)

À une époque où les appareils de son surround venaient d'apparaître sur le marché, Elektor mit au point un circuit maison. Le circuit fonctionnait sans les circuits intégrés Dolby spéciaux, car ils n'étaient pas distribués pour les amateurs, mais le labo d'Elektor réussit à concevoir un décodeur avec des moyens conventionnels. Il ajoutait deux canaux (médian + surround) au son stéréo. Les circuits imprimés furent produits en faible quantité.

**www.elektormagazine.fr/magazine/elektor-199502/35879**

## Jan Buiting

(éditeur de livres, ancien rédacteur en chef de l'édition anglaise)

### Décodeur ATN-Filmnet (1989)

Au cours de mes 36 ans chez *Elektor*, jamais la communauté des électroniciens amateurs ne m'a manifesté autant d'enthousiasme et d'éloges ni n'a été autant à l'unisson que lors de la publication du « Décodeur ATN-Filmnet » en 1989. Ce brillant projet propulsa d'emblée *Elektor* dans la cour des grands pourvoyeurs de publications « instruites, spirituelles et raffinées » sur l'art de pirater les chaînes de TV payantes couvrant l'Europe grâce à quelques satellites *Astra* et *Intelsat*. Rien qu'en Scandinavie, on estime que 20.000 décodeurs Elektor Filmnet étaient en service. « À usage privé et expérimental, Jan » – mais bien sûr ! « Nos hivers sont longs et sombres, Jan, tu comprends ? »

L'ingénierie du projet et la publication de l'article ont bousculé les habitudes au sein d'Elektor. Déjà le projet n'a pas été publié aux Pays-Bas dans *Elektuur* par crainte d'implications légales. Ensuite, par crainte de coups de fils et de visites d'avocats, le circuit imprimé n'a pas été listé sur les pages « EPS » (boutique d'Elektor à présent). Enfin, le concepteur publia sous le pseudonyme « P.N.P. Wintergreen », d'après un personnage du livre *Catch-22* de Joseph Heller.

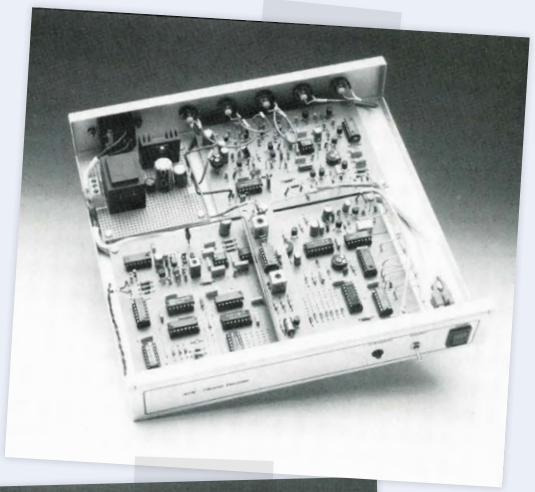
Considéré comme loufoque par les éditeurs d'Elektor, cet article a en fait incité des milliers d'amateurs européens à tenter de construire le décodeur. La nouvelle du piratage de Filmnet par Elektor s'est vite répandue. Pour la 1ère fois depuis 1974, l'édition anglaise d'Elektor fut épuisée et dut faire l'objet d'un second tirage. Aux Pays-Bas, les détaillants en électronique ont copieusement distribué des milliers de copies illégales de l'article en anglais. Submergées d'appels pendant des semaines, les opératrices du central téléphonique d'Elektor durent lire un texte écrit. Des circuits imprimés « fabriqués par quelqu'un » ont été vendus sous le manteau en boutique et lors de congrès de radioamateurs au R-U, en Allemagne et aux Pays-Bas.

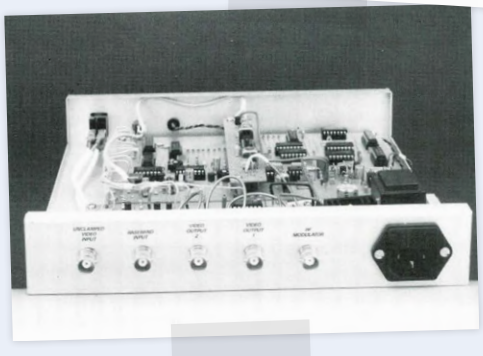
L'électronique du décodeur des années 1980 est brillante avec uniquement des composants bon marché, dont des CI CMOS, le TBA120S, et un tas de transistors *p-n-p*. Pas de microcontrôleurs ni autres nouveautés – les jours de gloire des « TUN & TUP » inventés par Elektor revivent.

Au cours des mois suivant la publication d'Elektor, Filmnet a changé plusieurs fois le format du cryptage. En vain, une mise à jour astucieuse du décodeur Elektor en vint à bout à chaque fois. J'ai eu l'honneur de publier ces mises à jour dans des épisodes de la série *From the Satellite TV Desk*. Après trois mises à jour avec textes correspondants pour les opératrices du central, P.N.P. Wintergreen proposa son ingénieux circuit complémentaire « learn & adapt » basé sur un échantillonneur-bloqueur à condensateur. Dès lors, l'utilisateur d'un décodeur Elektor Filmnet entièrement mis à jour ne remarquait même plus les changements de cryptage. C'était un piratage sublime d'un système de cryptage très coûteux vendu aux propriétaires de Filmnet comme étant « très difficile à contourner ». Plus tard, le cryptage « SAVE » du BBC World TV Service et plusieurs stations de TV par satellite DMAC/CMAC/Irdeto ont dû s'incliner devant le piratage d'Elektor qui obtint des résultats probants et une « résonance » massive du lectorat avec un *facteur Q* qui fait rêver aujourd'hui.

Je connais personnellement le concepteur, dont je respecte l'anonymat, c'est un vrai électronicien. « Problème ? – Pas d'entrée dans mon dictionnaire de l'électronique. »

**www.elektormagazine.fr/magazine/elektor-198702/52932**





### Générateur d'odeurs avec le CD4711 (1986)

La fameuse édition d'été d'Elektor était le numéro double de juillet-août, rempli de plus d'une centaine de petits (sous-)circuits souvent remarquables, de trucs et astuces. Le traditionnel « circuit amusant » de 1986 (basé sur un *CI générateur d'odeurs programmable* CD4711) est probablement le plus célèbre. Des années plus tard, des lecteurs nous demandaient encore si nous avions les coordonnées du fabricant *Odorant Elektronik GmbH* à Cologne.

**www.elektormagazine.fr/magazine/elektor-198607/52828**

## Luc Lemmens (Elektor Lab)

### Ordinateur BASIC (1987)
### Ordinateur monocarte 80C32/8052AH-BASIC (1991)
### Cours d'assembleur 8051/8032 (1992)

Digne successeur de la carte 8052AH-BASIC de 1987, l'ordinateur monocarte 8032/8052 fut l'un des premiers grands projets que j'ai supervisés au labo d'Elektor. Le matériel n'occupait pas une grande place dans l'article publié en mai 1991, mais il fut suivi d'un cours d'assembleur en huit leçons basées sur cette carte que les lecteurs ont plébiscitée en grand nombre. Cette carte pouvait également accueillir le contrôleur 8052AH-BASIC d'Intel – pour les réfractaires à l'assembleur. Un programme simple permettait de copier illégalement l'interpréteur BASIC interne (la totalité de sa mémoire de programme) sur une EPROM externe.

**www.elektormagazine.fr/magazine/elektor-198711/53122**
**www.elektormagazine.com/magazine/elektor-199105/32356**
**www.elektormagazine.com/magazine/elektor-199202/32571**


### GBDSO Gameboy Digital Sampling Oscilloscope (2000)

*L'oscilloscope numérique* basé sur une *Gameboy* de *Nintendo* présenté dans le numéro d'octobre 2000 connut aussi un succès retentissant et fut l'un des premiers modules tout montés disponibles dans la boutique Elektor. Au départ, nous ne devions produire qu'un seul lot, mais finalement, la demande dépassa largement les prévisions. J'ai oublié combien de « tout derniers lots » nous avons commandés. Il semble que de nombreux lecteurs disposaient d'une Gameboy reléguée au grenier et voulaient lui accorder une nouvelle vie comme instrument de mesure portable.

**www.elektormagazine.fr/magazine/elektor-200010/9041**

## Jens Nickel (rédacteur en chef)

### Embarquez Linux (2012)

En 2012, je venais d'être nommé rédacteur en chef pour l'édition allemande. J'ai été très fier que la diffusion payante ait augmenté de 20 % par rapport à l'année précédente. Mais je n'y étais pas pour grand-chose. C'était grâce à un projet en avance sur son temps. Notre auteur Benedikt Sauter et son équipe avaient conçu une carte compacte sous Linux d'un prix outrageusement faible pour l'époque : 50 € !

**www.elektormagazine.fr/magazine/elektor-201205/12118**

### Le bus arrive ! (2011)

Au début, ma qualité de « rédac chef » d'Elektor, m'interdisait de bricoler. Ma peur de « mettre les mains dans le cambouis » devait être jugulée de toute urgence. Le rédac chef international de l'époque m'a donc fait écrire une chronique sur les gars de notre labo. Mais il ne se doutait pas de l'effet boule de neige qui allait s'en suivre. Comme il n'y avait rien de neuf dans le labo, j'ai réfléchi à un système de bus utilisable pour toutes sortes de commandes. Les retours ont été sidérants. J'ai reçu des centaines de mails, et une douzaine d'inventeurs se sont regroupés pour définir les spécifications de l'*ElektorBus*. Mais, il y a eu aussi beaucoup d'excès verbaux. (J'ai été traité d'amateur, de designer des années 60).

Par la suite, plusieurs projets ont été basés sur le bus, et il a également été utilisé pour soutenir une thèse. Une entreprise de services BTP qui voulait développer des modules m'a appelé. Les gars de notre labo susmentionnés furent les seuls à ne pas l'aimer. Ainsi, au grand dam de nombreux lecteurs, ce grand projet a été gentiment mis en sommeil.

**www.elektormagazine.fr/magazine/elektor-201101/11758**

### Xmega sur carte polyvalente (2013)

L'incident avait cependant éveillé ma passion pour le développement. Deux ans plus tard, aidé de quelques ingénieurs professionnels, je conçus ma propre carte à microcontrôleur. Nous y avions mis un Xmega et de nombreux périphériques et pour finir, l'ensemble fut proposé (sans écran) à environ 100 €. Les ingénieurs nous ont fait cadeau de leur temps, mais ils avaient fabriqué des cartes et des prototypes pour environ 1.000 €. Nous n'avons même pas pu récupérer les frais engagés. Un flop !
J'avais pourtant fait de gros efforts sur le logiciel, allant jusqu'à créer un cadre avec bibliothèque de micrologiciels embarquée qui libérait non seulement du matériel du contrôleur, mais aussi du câblage de la carte. L'un de nos lecteurs et moi-même nous étions beaucoup amusés à écrire ce programme. Certains lecteurs qualifièrent mon approche d'ingénieuse. D'autres dirent que cette réalisation était juste superflue. Mais en fin de compte, j'ai beaucoup appris seul et j'ai peut-être même donné quelques idées à la communauté – c'est justement ça, Elektor !

**www.elektormagazine.fr/magazine/elektor-201310/23459**
**www.elektormagazine.fr/magazine/elektor-201305/20559**


## Clemens Valens (Elektor Lab)

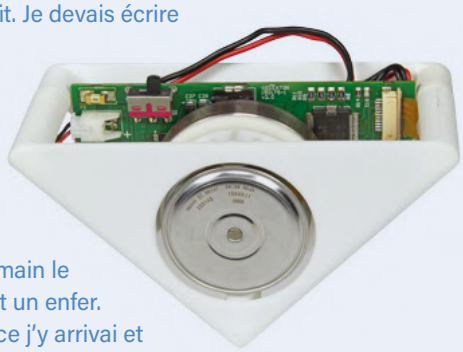
### Simulateur d'EPROM (1989)

En débutant dans ce qu'on appelle les systèmes embarqués, j'ai vite découvert que la reprogrammation des EPROM prenait beaucoup de temps. C'est pourquoi, lorsqu'en 1989 Elektor publia le simulateur d'EPROM, j'en ai construit un. Télécharger un micrologiciel dans une carte cible devint aussi simple que de copier un fichier sur le port d'imprimante parallèle du PC. Bien plus rapide que l'actuel Arduino, sans chargeur d'amorçage, mes collègues et moi l'avons utilisé des années durant, jusqu'à la disparition du port parallèle des PC et l'apparition des microcontrôleurs à mémoire flash intégrée.

**www.elektormagazine.fr/magazine/elektor-200101/9083**

### i-Pendulum (2016)

Le i-Pendulum n'est pas l'appareil le plus utile jamais conçu, mais quel excellent projet ! Placé sur une surface horizontale et allumée, il se relevait soudainement, se dressait sur son sommet puis s'immobilisait. Je devais écrire l'article d'Elektor et construire le prototype. Jean-Sébastien Gonsette avait magistralement conçu l'appareil, mais souder à la main le CI du moteur était un enfer. À force de patience j'y arrivai et obtins un appareil fonctionnel. Je l'ai ensuite montré à un ami… Il se mit à pouffer de rire. C'est l'une des réactions les plus gratifiantes qu'ait jamais suscitées l'une de mes propres réalisations.



**www.elektormagazine.fr/magazine/elektor-201604/28900**

### Circuits d'été & guides de semi-conducteurs

Je n'achetais Elektor que si un article m'intéressait vraiment, sauf le numéro double d'été avec plus de cent schémas que j'achetais chaque année les yeux fermés. Outre les schémas, j'aimais aussi les publicités, en particulier les annonces mimant des catalogues

qui répertoriaient des centaines de composants. J'ai continué sans regarder jusqu'en 2007 où, pour la première (et dernière) fois, il avait un thème : la robotique. La robotique concernait 95 % des circuits et rien ne m'a plu. L'année suivante, au lieu d'acheter le numéro d'été, j'ai commencé à travailler pour Elektor.

**www.elektormagazine.fr/magazine/elektor-198507**
**www.elektormagazine.fr/magazine/elektor-200707**

### Formant (1977)

Trop jeune lorsqu'il fut publié, trop pauvre lorsque je commençai à m'y intéresser, j'ai enfin mis la main sur un synthétiseur musical Formant quand deux amis en achetèrent un presque en état de marche. Il a rapidement atterri dans ma chambre. Je l'ai recâblé, j'ai remplacé les générateurs d'enveloppe par un modèle publié dans un magazine concurrent et utilisé mon tout nouveau fréquencemètre numérique maison pour tout calibrer. Avec mes amis, nous avons joué des heures entières sur cette machine ressuscitée.

**www.elektormagazine.com/magazine/elektor-197704/57836**

## Mathias Claußen (Elektor Lab)

### Analyseur logique pour Atari ST (1989)

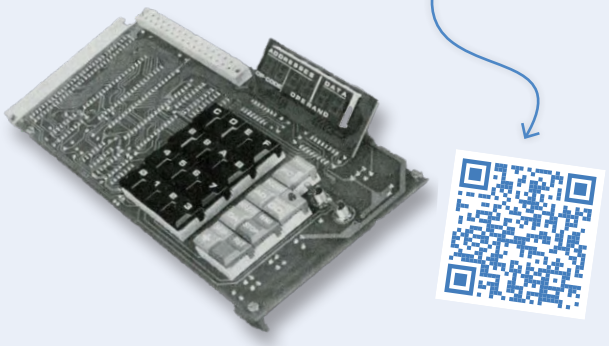
Mon premier ordinateur fut un Atari ST, c'est pourquoi j'ai été assez impressionné de voir dans les archives que des analyseurs logiques maison avaient été fabriqués sur la base de ces machines. Avec juste une série de puces logiques et un logiciel bien conçu, votre ST devenait un analyseur logique à huit voies échantillonnant à 2 MHz. Cela reposait sur l'exploitation astucieuse de l'interface HDD externe du ST. Cette idée divulguée en 1989 fut réinventée presque 20 ans plus tard avec le Saleae Logic 8. Au lieu de réutiliser une interface HDD et des CI 74HC, le Logic 8 utilisait un microcontrôleur et l'USB. Bien sûr, l'échantillonnage était plus rapide et le logiciel plus puissant, mais le principe était identique.

**www.elektormagazine.fr/magazine/elektor-198909/53524**

### Ordinateur junior (1980)

Restons dans cette vague rétro : partout des amateurs font vivre le MOS6502 et ses dérivés. Ils programment et développent des systèmes autour de cette puce. Elle fut le µprocesseur de systèmes célèbres, comme la Nintendo NES, l'Atari 800, l'Apple II et, dans une version modifiée, le C64. En 1980, Elektor publia le Junior Computer, un ordinateur 6502 qui fut une porte d'entrée dans le monde des microprocesseurs. Avec une horloge à 1 MHz, 1 Ko de RAM et 1 Ko de ROM, il était plus lent et moins puissant que l'Arduino actuel basé sur l'ATmega ; mais à l'époque, tout un chacun pouvait le construire et le dépanner. Des systèmes à 6502, vieux de 30 ans existent toujours, et avec le Commander X16 de David Murray, un nouveau design basé sur ce CPU est en devenir. Si construire vos propres ordinateurs à 6502 vous tente, sachez que ces derniers sont toujours produits.

**www.elektormagazine.fr/magazine/elektor-198004/51229**



### e-lock : La première puce de sécurité Elektor (2014)

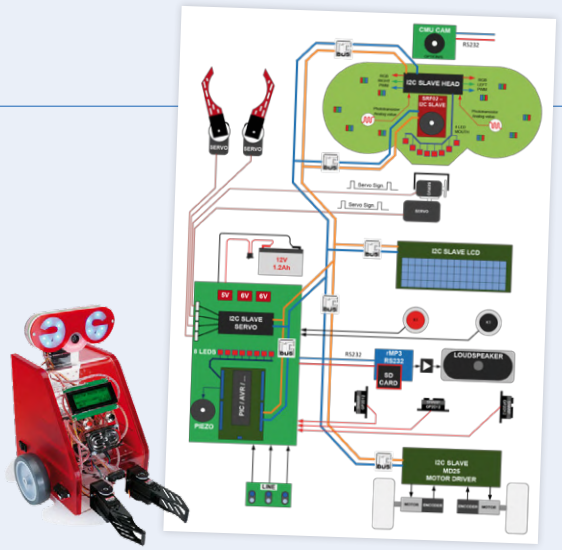
Eh oui, même Elektor *a* produit sa propre puce. En 2014, la puce e-lock fit son apparition, intégrant l'Ethernet filaire, un CA/N, l'I²C, un CN/A et des GPIO avec communication sécurisée et chiffrement. Le chiffrement matériel et les éléments de sécurité la destinaient à l'Internet des Objets (IoT) émergeant.

Ce projet d'Elektor est inclassable, car des éléments internes de cette puce sont restés secrets. Saura-t-on jamais quel noyau fut utilisé et comment les registres internes fonctionnaient ? De nouvelles puces comme l'ESP8266 ou d'autres de WIZnet ont pris sa place. L'Ethernet filaire a toujours sa place, mais les utilisateurs exigent toujours plus d'IdO sans fil.

**www.elektormagazine.fr/magazine/elektor-201404/26234**

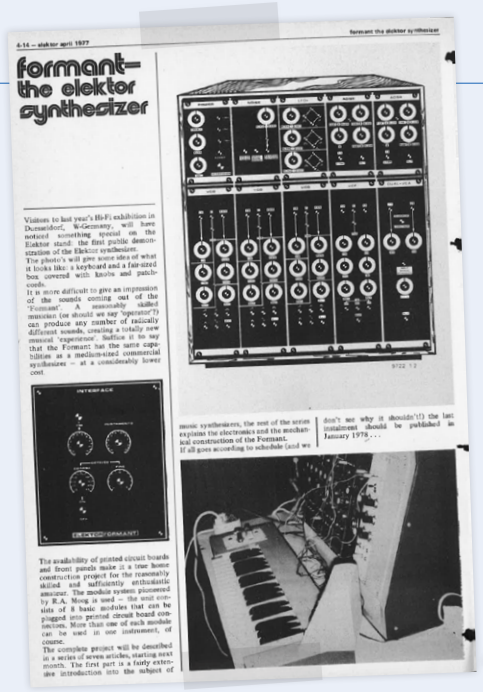
### Le robot Proton d'Elektor (2011)

Coup du sort : oublié au bureau, il a pris la poussière pendant trop longtemps, voici Elektor Proton. C'est un robot autopiloté destiné à servir de plateforme éducative. De design modulaire basé sur bus I²C, il fut conçu pour mettre la robotique à la portée de tous avec un vaste jeu de fonctions ciblant débutants et professionnels afin de s'adapter à leurs besoins propres. Vu qu'une décennie s'est écoulée depuis son apparition en 2011, le Proton resté au bureau




aurait besoin d'un remaniement avec des capteurs récents, la vision et un puissant MCU, sans doute avec Wi-Fi. Grâce à son bus I²C, cela ne devrait pas poser problème.

**www.elektormagazine.fr/magazine/elektor-201105/11829**


formant– the elektor synthesizer

# C. J. Abate

## (directeur éditorial)

Nombreux sont ceux qui sont tombés dans le « rabbit hole » de YouTube. C'est inéluctable : après dîner, vous suivez d'abord la vidéo GitHub de Clemens Valens (http://bit.ly/elektor-github) sur ElektorTV et deux heures après vous êtes toujours là, captivé par une vidéo sur la construction d'une cabane loin de tout réseau en Alaska. Mais êtes-vous déjà descendu dans le puits sans fond d'Elektor ? Je n'ai pas tous les numéros imprimés d'Elektor dans ma bibliothèque, mais j'ai accès aux archives numérisées. Ces dernières années, j'ai passé d'innombrables heures à choisir et parcourir les PDF. Il y a peu, j'ai vu sur YouTube une vidéo sur Elektor Formant et j'ai commencé à chercher dans les archives d'Elektor les premiers articles de ce projet de synthé musical tant apprécié. 90 minutes plus tard, j'étais en train de lire l'article « Radar Detector » de mars 1991. Avec des décennies d'articles sous la main, il est difficile de choisir ses favoris. Voici donc quelques articles que je recommande vivement parce qu'ils présentent de belle façon quelques sujets que je juge vraiment intéressants.

### Formant – le synthétiseur de musique Elektor (1977)

Ce projet m'est bien antérieur. Mais en ayant entendu parler depuis des années, j'ai décidé de faire une petite recherche. C'est vrai, aujourd'hui encore on trouve sur YouTube des gens qui jouent sur Formant.

**www.elektormagazine.com/magazine/elektor-197705/57847**

### et l'homme créa sa puce : facile, la vie, avec un puissant FPGA(2012)

Voici quelques années, Jan Buiting, rédacteur en chef d'Elektor et moi-même sommes allés visiter Xilinx en Californie. En échangeant avec des ingénieurs et en visitant l'usine, j'ai vite réalisé la vacuité de mes connaissances en FPGA. L'article « et l'homme créa sa puce » fut un bon point de départ.

**www.elektormagazine.fr/magazine/elektor-201212/12258**

### Les tubes électroniques (1984)

Qui n'est pas fasciné par les lampes, vous savez, ces « choses fragiles en verre avec toutes sortes de trucs compliqués à l'intérieur » ? Cet article est un classique d'Elektor. Que vous soyez passionné d'électronique ancienne ou attiré par les designs contemporains qui mélangent électronique haut de gamme et tubes rétro, vous allez adorer cet article.

**www.elektormagazine.fr/magazine/elektor-198411/52406**

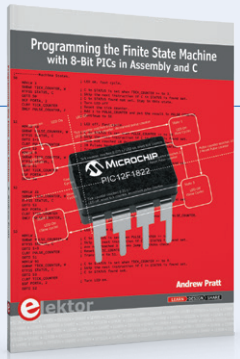
## Michel Kuenemann (auteur)

J'ai découvert Elektor au milieu des années 80, et j'ai été étonné par la qualité des projets. J'ai construit le fréquencemètre à base de 6502 publié en 1985 ; je m'en sers encore. Lors de mes études d'ingénieur en électronique, j'ai fait partie d'une association d'amateurs de fusées. Nous prévoyions de mesurer la variation de pression atmosphérique durant l'ascension de la fusée. Elektor avait publié un projet très soigné avec capteur barométrique et plusieurs AOP. L'intégration de la carte dans la fusée fut une réussite. Après mon diplôme, je créai une entreprise et au fil des ans, Elektor resta pour moi une source constante d'inspiration. Grâce à Elektor, j'ai découvert le bus I²C et bien d'autres progrès techniques. Plus tard, je pus contribuer au magazine avec des articles liés au modélisme, par ex. banc de rodage de moteur (avril 2009) et une télécommande basée sur ZigBee (octobre 2011). À 58 ans, Elektor continue de m'inspirer au quotidien. Joyeux 60e anniversaire Elektor ! Merci à toi, à ton équipe et aux auteurs pour ce que vous apportez.

**Articles de Michel :**
**www.elektormagazine.fr/authors/171/michel-kuenemann**

## Andrew Pratt (auteur)

Je découvris Elektor dans les années 1970, bien avant l'Internet. Les magazines techniques étaient didactiques et parsemés de publicités utiles sur les composants et les équipements. À présent, Internet est une source illimitée d'informations, alors à quoi bon lire des magazines ? Les sites sont pratiques, mais le papier garde bien des avantages. Elektor offre les deux. Le site




Internet centralise tout, de sorte qu'il répond à votre quête d'information. J'ai fait publier trois livres par Elektor. Un livre ou un article avec une partie pratique, est un bon moyen d'apprendre « comment ça marche ». L'expérimentation reste possible, même si vous ne construisez pas le projet complet. Vous pouvez aussi télécharger un projet sur Elektor Labs Online et ainsi, tout un chacun peut le copier ou l'améliorer. Les logiciels libres ont connu un succès spectaculaire, Elektor fait de même pour l'électronique.

**Articles d'Andrew :**
**www.elektor.com/catalogsearch/result/?q=andrew%20pratt**

## GreatScott! (auteur, YouTubeur)

Pour ma part, j'ai mis un bon bout de temps à découvrir Elektor. Je produisais des vidéos YouTube sur l'électronique depuis des années quand la recommandation d'un amateur attira mon attention. Il s'agissait d'un kit DIY d'Elektor. Le lien internet me conduisit sur le site d'Elektor, et j'y ai passé des heures. J'ai vite vu que les thèmes abordés présentaient des similitudes avec mes propres vidéos. Quelques mois plus tard, Elektor m'a contacté et proposé de réaliser ensemble des productions vidéo – ça a coulé de source : je suis devenu lecteur d'Elektor et je produis des vidéos sur toutes sortes de sujets d'électronique en coopération avec Elektor. Une situation gagnant-gagnant !

**Article en ligne gratuit de GreatScott sur le superchargeur LiPo en kit DIY : www.elektormagazine.fr/articles/superchargeur-lipo-booster-en-kit**

# multimètre de table Siglent SDM3045X


**Harry Baggen** (Elektor)

Tout passionné d'électronique dispose d'un ou plusieurs multimètres. C'est logique, mais avez-vous déjà envisagé d'acheter un multimètre de table plutôt qu'un multimètre portatif ? Ceux-ci offrent généralement beaucoup plus de fonctions et un meilleur affichage. Le Siglent SDM3045X est un multimètre de table à 4½ chiffres. Sa précision de base est de 0,02% et ses options d'interface sont nombreuses. Voici mon impression après quelques mois d'utilisation.

On peut difficilement faire de l'électronique sans au moins un, voire deux ou plusieurs multimètres. On en trouve pour seulement 10 €, mais si l'on veut conjuguer robustesse, sécurité, fonctions variées et précision, il faudra quelques centaines d'euros. La précision et le prix sont corrélés.

À la maison, j'ai toujours travaillé avec un multimètre simple. L'année dernière, après des décennies de bons et loyaux services, il a finalement été remplacé par une version améliorée, à double affichage et, surtout, une meilleure précision. J'en ai été très satisfait. Lorsque j'ai eu l'occasion de tester la version de table d'un multimètre de Siglent, j'ai pensé que ce serait une bonne occasion de le comparer à mon multimètre portable. Ce Siglent est plus cher, mais il se pourrait que cette dépense soit justifiée. Je voulais savoir aussi lequel est le plus facile à utiliser et quels sont leurs avantages et inconvénients respectifs.

## L'appareil

Il s'agit du « plus petit » d'une série de trois multimètres de table de Siglent. Les deux autres affichent plus de chiffres, leur précision est plus grande, mais ce SDM3045X offre par ailleurs les mêmes caractéristiques. Si sa précision vous suffit, ce modèle vous offrira le meilleur rapport qualité/prix.

Le boîtier métallique du SDM3045X est robuste. Ses coins sont protégés par des pare-chocs en plastique. La poignée de transport est articulée et permet d'incliner l'avant du compteur vers le haut. Ses dimensions sont les mêmes que celle de beaucoup d'autres appareils Siglent (les générateurs de signaux notamment), de sorte que ces appareils s'empilent bien.
Devant, un écran en couleur de 4,3 pouces est associé à six boutons de menu, dont les fonctions apparaissent sur l'écran. À droite, les boutons de fonction habituels ont pour la plupart une fonction secondaire. Un ensemble de touches de curseur facilite la navigation dans les menus et les réglages.
Il y a cinq entrées (**fig. 1**), deux normales, deux de détection pour les mesures de résistance à quatre fils et une de mesure de courant. Derrière, nous trouvons une embase de cordon secteur IEC, un connecteur USB et un connecteur LAN pour se connecter à un ordinateur ou un réseau, ainsi que deux fiches BNC pour le déclenchement externe. Sans oublier le porte-fusible pour la fonction de mesure du courant.

## Mesures

Le SDM3045X offre évidemment toutes les fonctions d'un bon multimètre portable : gammes de tension et de courant, résistance, capacité, fréquence, dB(m), température et mesures de conti-

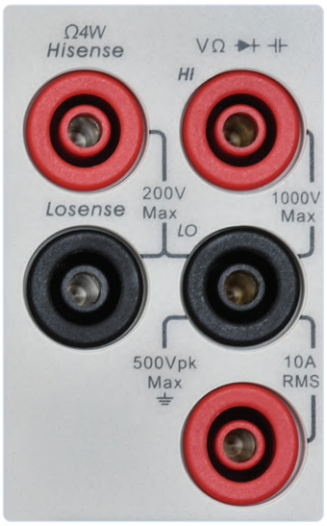



*Figure 1. Entrées pour la mesure de résistance à quatre fils.*

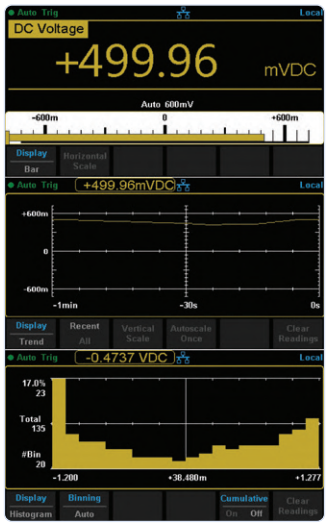



*Figure 2. Options d'affichage.*


*Figure 3. Le multimètre peut être connecté à un PC par USB ou LAN.*

nuité, polarité de diode, autoréglage et, comme c'est maintenant courant sur les meilleurs multimètres, la possibilité d'afficher deux valeurs de mesure simultanément (p. ex. tension et fréquence CA). L'écran affiche 66000 points et sa précision de base en continu est de 0,02%. Il effectue jusqu'à 150 mesures/s. L'instrument n'a pas été conçu pour des circuits sous haute tension (CAT I (10000 V)/ CAT II (300 V)), mais qui utiliserait un tel multimètre de table pour effectuer de telles mesures ? Ce multimètre peut être commandé depuis un PC via un réseau local ou par l'USB, lequel pourra donc stocker les résultats de mesure.

Pour les mesures de tension, vous avez diverses options d'affichage : barre supplémentaire sous la valeur, graphique de tendance ou


*Figure 4. Le logiciel gratuit, EasyDMM, commande le multimètre et lit les données de mesure qu'il sauvegarde sur le PC.*

histogramme (**fig. 2**), et différentes fonctions mathématiques. De nombreuses fonctions offrent différentes options, je ne vais pas vous assommer avec l'énumération de tout ce dont ce multimètre est capable. Voici quelques exemples : dans la gamme des 600 mV, vous avez le choix entre une impédance d'entrée de 10 MΩ ou 10 GΩ ; avec la fonction de test de la diode, vous pouvez régler la tension de test entre 0 et 4 V ; avec la mesure de la température, vous avez différentes options de configuration, selon le type de capteur de température utilisé. Le déclenchement du multimètre sera manuel ou externe. Des seuils haut et bas peuvent être spécifiés pour vérifier si une valeur se situe dans cette plage. La mémoire flash interne de 1 Go peut retenir jusqu'à 10 000 résultats de mesure. Les paramètres de configuration et les résultats des mesures peuvent être stockés sur une clé USB, enfichée dans le connecteur USB en façade. Si vous souhaitez connecter l'instrument à un PC, vous avez le choix entre USB ou LAN (**fig. 3**). Le logiciel d'accompagnement, EasyDMM, (**fig. 4**) n'existe que pour Windows, ce qui ne posera guère de problème à la plupart des électroniciens. Ce logiciel permet à la fois de faire fonctionner le multimètre et de stocker des séquences de mesure. Comme la plupart des autres instruments Siglent, il utilise également des commandes SCPI standard, pratiques pour ceux qui souhaitent mettre en place leurs propres procédures de test.

## En pratique

Ce qui distingue le plus ce multimètre de table d'un multimètre portable, c'est d'être alimenté par le secteur. Est-ce un inconvénient ? Pas s'il reste toujours au même endroit, vous le remarquerez à peine. Vous placez le SDM3045X sur paillasse, vous le branchez et il restera à côté de l'oscilloscope, de l'alimentation et de vos autres instruments de mesure. En pratique les raisons et les occasions de le déplacer sont rares. La plupart du temps, je travaille sur ma paillasse de labo et c'est là que j'utilise le multimètre. Ce qui est frappant, c'est que depuis que le SDM3045X est installé sur ma paillasse (**fig. 5**), je n'ai pratiquement pas utilisé mon multimètre portable. Je n'ai utilisé les deux que pour faire deux mesures simultanément. Le 3045X est beaucoup plus stable sur la paillasse qu'un DMM portable et, grâce à son support rabattable, j'ai trouvé le fonctionnement des boutons poussoirs beaucoup plus pratique qu'un commutateur rotatif.

Il faut un peu de temps pour s'habituer à son fonctionnement et à toutes ses fonctions tant les options de configuration sont nombreuses. À l'allumage, il faut un peu de patience, il y a un temps de démarrage comme sur un ordinateur, pas du tout comme sur un multimètre portatif. Tout le reste n'est qu'avantages : un afficheur grand et clair et à peu près toutes les fonctions imaginables pour un multimètre. La précision est fantastique et, pour les mesures de tension alternative, la vaste gamme de fréquences jusqu'à 100 kHz est un bonus ; la plupart des multimètres portatifs d'atteignent pas même la gamme audio.

S'il vous faut plus de précision et que votre bourse le permet, tournez-vous vers le SDM3055, il offre un chiffre de plus. Mais vous perdrez un autre avantage considérable du SDM3045X : il n'a pas de ventilateur, contrairement au 3055. Il est donc silencieux comme une souris. Le logiciel pour PC qui l'accompagne est bien pensé, idéal pour l'enregistrement des mesures. Vous l'aurez compris : cet instrument offre tellement plus que je ne sors plus très souvent mon multimètre portable de son étui !

## Conclusion

Si vous êtes prêt à investir un peu dans un bon multimètre, je vous recommande de jeter un coup d'œil appuyé au SDM3045X et à toutes les caractéristiques énumérées dans sa fiche technique. Ce multimètre de table est un excellent compagnon, que vous soyez électronicien de profession ou de loisir. C'est autre chose qu'un multimètre portatif, avec notamment une lecture et une utilisation plus faciles.

200720-02


*Figure 5. Le SDM3045X est à l'aise entre les autres instruments de laboratoire.*

**Des questions ou des commentaires**
Veuillez les adresser par courriel à redaction@elektor.fr.

**Ont contribué à cet article**

Texte et illustrations : **Harry Baggen**
Rédaction : **Eric Bogers**
Traduction : **Alba Boudhine**
Maquette : **Giel Dols**

**PRODUITS**

› **Multimètre de table Siglent SDM3045X**
www.elektor.fr/17892

# pince ampèremétrique pour courant continu

## Capteur à effet Hall + noyau de ferrite + Arduino

**Martin Oßmann** (Allemagne)

Pour mesurer l'intensité du courant dans un fil, on insère d'habitude un ampèremètre en série avec le fil. Avec une pince ampèremétrique, ou « Amp-clamp », il n'est pas nécessaire d'ouvrir le circuit. Les versions les plus rudimentaires de la pince ne peuvent mesurer que le courant alternatif car les capteurs utilisés sont insensibles au courant continu (ils fonctionnent sur le principe du transformateur). Pour mesurer le courant continu, on peut utiliser des capteurs à effet Hall.

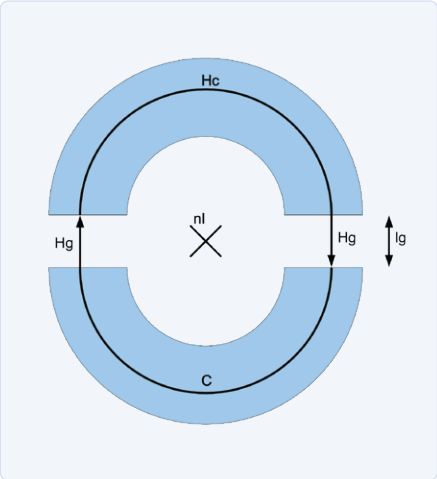



*Figure 1. La géométrie du champ magnétique.*

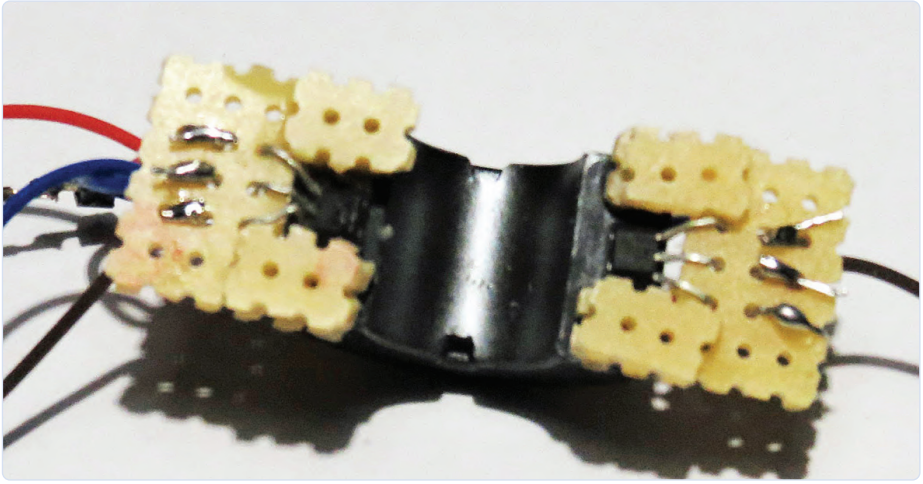


*Figure 2. Disposition des capteurs à effet Hall sur le tore de ferrite.*

Avant de se lancer dans la conception d'un tel projet, il est toujours bon de faire des recherches sur le sujet et d'étudier les modèles du commerce pour voir comment ils sont conçus et s'ils peuvent être améliorés. Une pince ampèremétrique se compose essentiellement d'un noyau de deux moitiés de tore de ferrite, assemblées de manière à ce qu'elles soient séparées par un entrefer (**fig. 1**). Le courant qui circule dans le fil entouré par le noyau a la valeur *nI*.

### Relations mathématiques

Le facteur *n* est le nombre de tours que le fil dans lequel circule le courant à mesurer fait autour du noyau du capteur ; c'est un facteur qui multiplie l'excitation magnétique *H* qui magnétise le noyau. Le champ magnétique dans le noyau $B_c$ (*c* pour *core*, le noyau) est le même que dans l'entrefer $B_g$ (*g* pour *gap*, l'entrefer) de sorte que $B_g = B_c$. C'est une conséquence des équations de Maxwell (pour en savoir plus, lisez « A Student's Guide to Maxwell's Equations », D. Fleisch). Pour simplifier, nous pouvons supposer que les champs magnétiques sont uniformes dans le noyau et dans l'entrefer. L'équation $B_g = \mu 0 \times H_g$ s'applique pour l'entrefer tandis que dans la ferrite on a $B_c = \mu r \times \mu_0 \times H_c$. ($\mu_0 = 4\pi \cdot 10^{-7}$ *H/m* est la perméabilité magnétique du vide, $\mu_r$ la perméabilité relative du matériau, très voisine de 1 pour l'air et la plupart des matériaux non ferreux). Pour la ferrite, on a typiquement $\mu_r \approx 2000$. L'équation $H_g = \mu_r \times H_c$ montre que dans la ferrite, l'excitation magnétique est plus faible que dans l'entrefer d'un facteur $\mu_r$. Dans la pratique, on peut donc souvent la négliger. Selon le théorème d'Ampère, l'intégrale curviligne de l'excitation magnétique *H* le long d'une courbe fermée est égale à l'intensité du courant qui la traverse, soit *nI*. Dans la figure 1, la courbe est représentée par *C*. L'entrefer a ici la longueur $l_g = 1{,}6$ *mm* et la longueur d'une moitié de noyau est donnée par $l_c = \pi \times d_c / 2$ où $d_c$ est le diamètre du noyau, $d_c = 18$ *mm*. Le théorème d'Ampère donne :

$$2 \times l_g \times H_g + 2 \times l_c \times H_c = nI$$

Remplaçons *Hc* par sa valeur et résolvons l'équation pour *Hg*, nous obtenons :

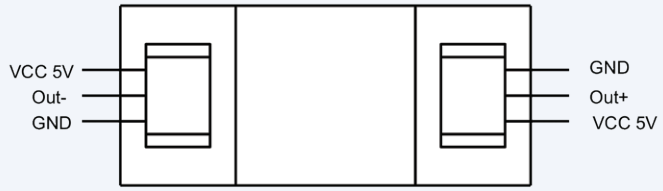



*Figure 3. Schéma de connexion des capteurs à effet Hall.*

*Figure 4. De multiples tours autour du noyau augmentent la sensibilité.*

$H_g = nI / (2 \times l_g + 2 \times l_c / \mu_r)$

Voilà qui nous donne l'excitation magnétique détectée par les capteurs en fonction du courant. Les deux capteurs à effet Hall A 1324 LUA-T utilisés dans ce projet ont une sensibilité de $S = 5\ mV / G$ et fournissent donc une tension de sortie égale à :

$V = 2 \times S \times \mu_0 \times nI / (2 \times l_g + 2 \times l_c / \mu_r)$

Soit en substituant les valeurs numériques ($n = 1$) :

$V = I \times 38{,}92\ mV / A$

Pour les valeurs de courant rencontrées dans les applications électroniques courantes, cela correspond à des tensions de l'ordre du millivolt, encore d'un maniement raisonnable. Si nous ignorons le champ dans la ferrite, nous pouvons simplifier l'expression :

$V = S \times \mu_0 \times nI / l_g$

ce qui donne $V = I \times 39{,}26\ mV / A$, soit un coefficient simplifié suffisamment proche de la valeur obtenue précédemment.

## Le noyau de ferrite

L'élément central du dispositif est constitué par les deux moitiés d'une ferrite antiparasite standard à charnière que l'on trouve souvent autour des câbles qui sortent d'un boîtier contenant des équipements électroniques (**fig. 2**). Les deux capteurs à effet Hall sont collés aux surfaces de la section transversale du noyau pour former les entrefers. Un petit morceau de pertinax ou de carte à trous de 1,6 mm d'épaisseur (sans aucune trace de cuivre) porte le capteur et détermine l'entrefer. La **figure 3** montre le positionnement des capteurs.

Les deux moitiés du noyau peuvent alors être serrées l'une contre l'autre au moyen de ruban adhésif ou d'un bracelet de caoutchouc. On peut ensuite faire passer le fil où circule le courant à travers le trou central du noyau de ferrite. La **figure 4** montre un exemple où un fil monobrin a été enroulé plusieurs fois autour du noyau, ce qui a pour effet d'augmenter linéairement la sensibilité du capteur à effet Hall.

## L'électronique

La conversion analogique/numérique est effectuée sur 18 bits par une puce bon marché MCP3421. Elle possède une référence de tension interne de 2,048 V et utilise un étage d'entrée différentiel. Au repos, la tension de sortie des capteurs Hall est la moitié de la tension d'alimentation. Lorsque les deux capteurs sont dans la configuration de la figure 2, le flux magnétique produit par le courant circulant dans le fil provoque sur leurs sorties des variations de tension opposées. Ces variations s'additionnent donc dans l'étage d'entrée différentiel du convertisseur A/N. La résolution du convertisseur A/N à 18 bits est de $2{,}048\ V / 217 = 0{,}015\ mV$, ce qui est suffisant pour cette application.

Une carte Arduino Nano sert d'unité centrale ; elle dispose de suffisamment de broches GPIO pour notre réalisation. Un écran OLED de 128×64 pixels bon marché sert d'afficheur. La **figure 5** montre le circuit complet, qui reste relativement simple. Si vous préférez utiliser

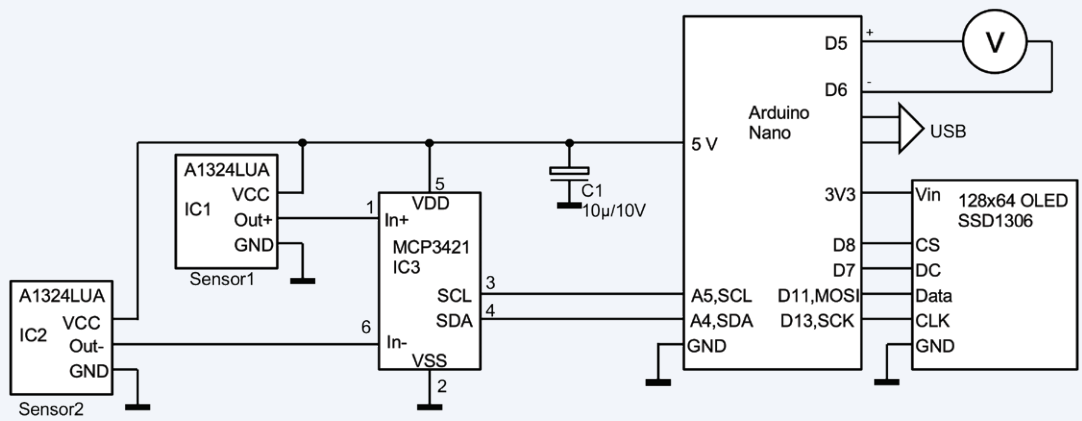



*Figure 5. Le circuit complet de l'ampèremètre.*

un indicateur analogique, vous pouvez également connecter un voltmètre à cadre mobile à la sortie. Un volt en sortie correspond alors à un courant mesuré d'un ampère.

## Les commandes

Pour le contrôle du fonctionnement et le réglage des paramètres de tarage utilisés par le logiciel, il existe quelques commandes qui peuvent être entrées via l'interface série de l'EDI Arduino. Les valeurs mesurées sont également affichées via l'interface série.

**Commande '0' = réglage du zéro**

Cette commande ne doit être utilisée qu'en l'absence de tout courant à mesurer. Le logiciel effectue un réglage du zéro et enregistre la valeur du décalage dans une EEPROM non volatile. Les capteurs à effet Hall ont une tension de décalage (qui dérive avec la température) qui doit être compensée pour obtenir des mesures précises.

**Commande '1' = gain pour 1 A**

Cette commande sert à évaluer le gain interne pour un courant à mesurer *nI* de 1 A. Le gain est calculé et enregistré dans l'EEPROM.

**Commande '5' = gain pour 500 mA**

Comme la commande '1', mais pour un courant de 0,5 A.

**Commande 'u'**

La valeur numérique saisie à la suite de 'u' est affichée sous la forme d'une tension. L'ampèremètre passe ensuite en « mode calibrage DVM » pendant un temps où vous pouvez utiliser les touches « + » et « – » pour annuler l'écart éventuel entre la valeur affichée et la valeur saisie.

**Commande 'd'**

Les valeurs par défaut des paramètres sont écrites dans l'EEPROM afin de l'initialiser pour le premier démarrage.

## Réalisation

Pour tester le principe de ce projet, j'ai réalisé le prototype sur une carte de prototypage avec des fils à connecteurs Dupont pour câbler les différents composants utilisés (**fig. 6**). On voir au premier plan le tore de ferrite flanqué du petit afficheur à gauche et de l'Arduino Nano à droite. Le logiciel a été développé avec l'EDI Arduino, puis téléchargé sur la carte Nano. La dernière version de ce logiciel est disponible gratuitement sur la page web d'Elektor de cet article [1]. Le prototype du projet « Amp-clamp » a été testé et s'est comporté comme prévu lors de sa mise sous tension. Nous espérons que ce circuit vous inspirera pour réaliser vos propres expériences ou pour créer d'autres projets relatifs à la mesure magnétique des courants !

(200595-04)

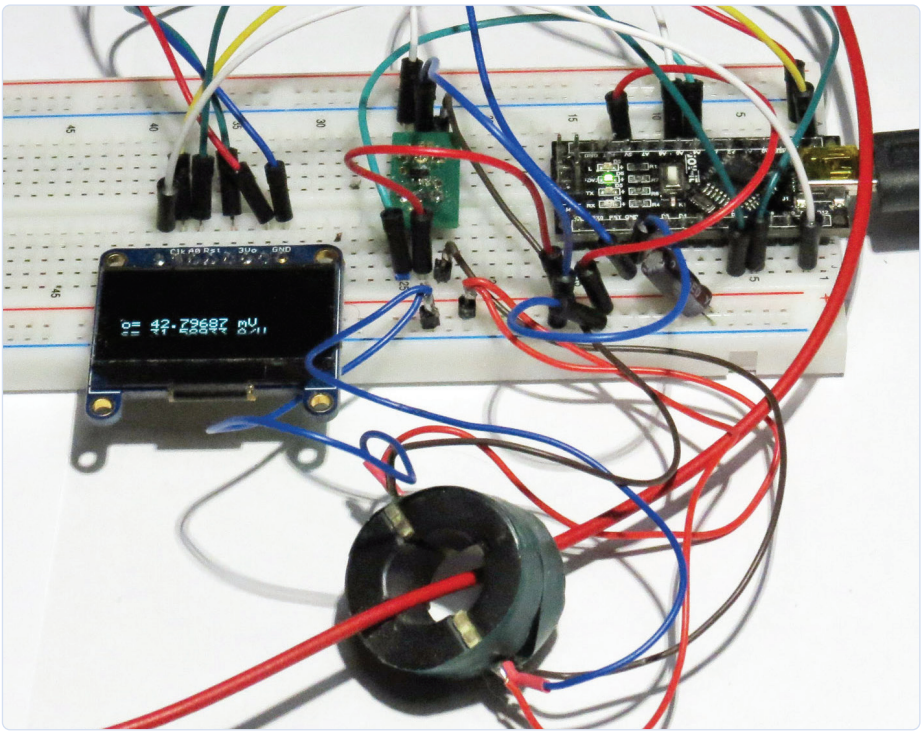


*Figure 6. La réalisation du prototype sur une carte de prototypage.*

**Des questions, des commentaires ?**
Contactez-nous par courriel (redaction@elektor.fr).


**Contributeurs**
Projet et texte : **Martin Oßmann**
Rédacteur : **Thomas Scherer**
Mise en page : **Giel Dols**
Traduction : **Helmut Müller**


## LIEN

[1] **Page web de l'article** : www.elektormagazine.fr/200595-04

## PRODUITS

- **Afficheur OLED de 0,96", 128×64 pixels, pour Arduino**
  www.elektor.fr/blue-0-96-oled-display-spi-6-pin
- **Pince ampèremétrique PeakTech 4350**
  www.elektor.fr/peaktech-4350-clamp-meter

# 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**

# station de soudage 2021

## Facile à construire !

**Luc Lemmens et Mathias Claußen (Elektor)**

L'offre de stations de soudage prêtes à l'emploi est vaste, mais nombreux sont ceux qui préfèrent en construire une eux-mêmes. La station de soudage pour fer Weller RT présentée dans notre numéro d'octobre 2018 [1] a connu un grand succès auprès de nos lecteurs. Les réactions que ce projet a suscitées nous a convaincu de revoir notre copie et d'assurer la compatibilité avec les fers Weller RT, Hakko FX-8801 et JBC T245.

### INFOS SUR LE PROJET

**Mots clés**
outils, laboratoire, soudage, Weller, Hakko, JBC

**Niveau**
débutant – **connaisseur** – expert

**Temps**
env. 4 h

**Outils**
outils de soudage, outils mécaniques, imprimante 3D (facultatif)

**Coût**
70 € environ

Le faible prix de revient de cette réalisation était un critère prépondérant. Rien de superflu, mais robuste et fonctionnel. C'est un peu démodé à l'ère des *CMS*, mais lorsqu'il s'agit de souder, la plupart des amateurs préfèrent encore les composants traversants. Bien qu'il soit de plus en plus difficile d'en trouver (en particulier pour les circuits intégrés), nous avons voulu présenter une réalisation complète respectant ce principe afin de faciliter l'assemblage. La construction et le câblage de cette station de soudage sont simples. Elle utilise deux cartes à circuit imprimé (principale + face avant) qui sont interconnectées par un câble plat. Nous avons retenu un nouveau microcontrôleur *AVR* de *Microchip*. Il permet de concevoir le logiciel avec l'*EDI Arduino*, plateforme qui offre à ceux qui le souhaitent de pouvoir modifier le logiciel.

Nous étions plutôt fiers et satisfaits de la compacité de la précédente station de soudage, avec alimentation externe standard de 12 V DC. À chaque médaille son revers : beaucoup de nos lecteurs se sont plaints de son poids trop faible pour rester sur l'établi

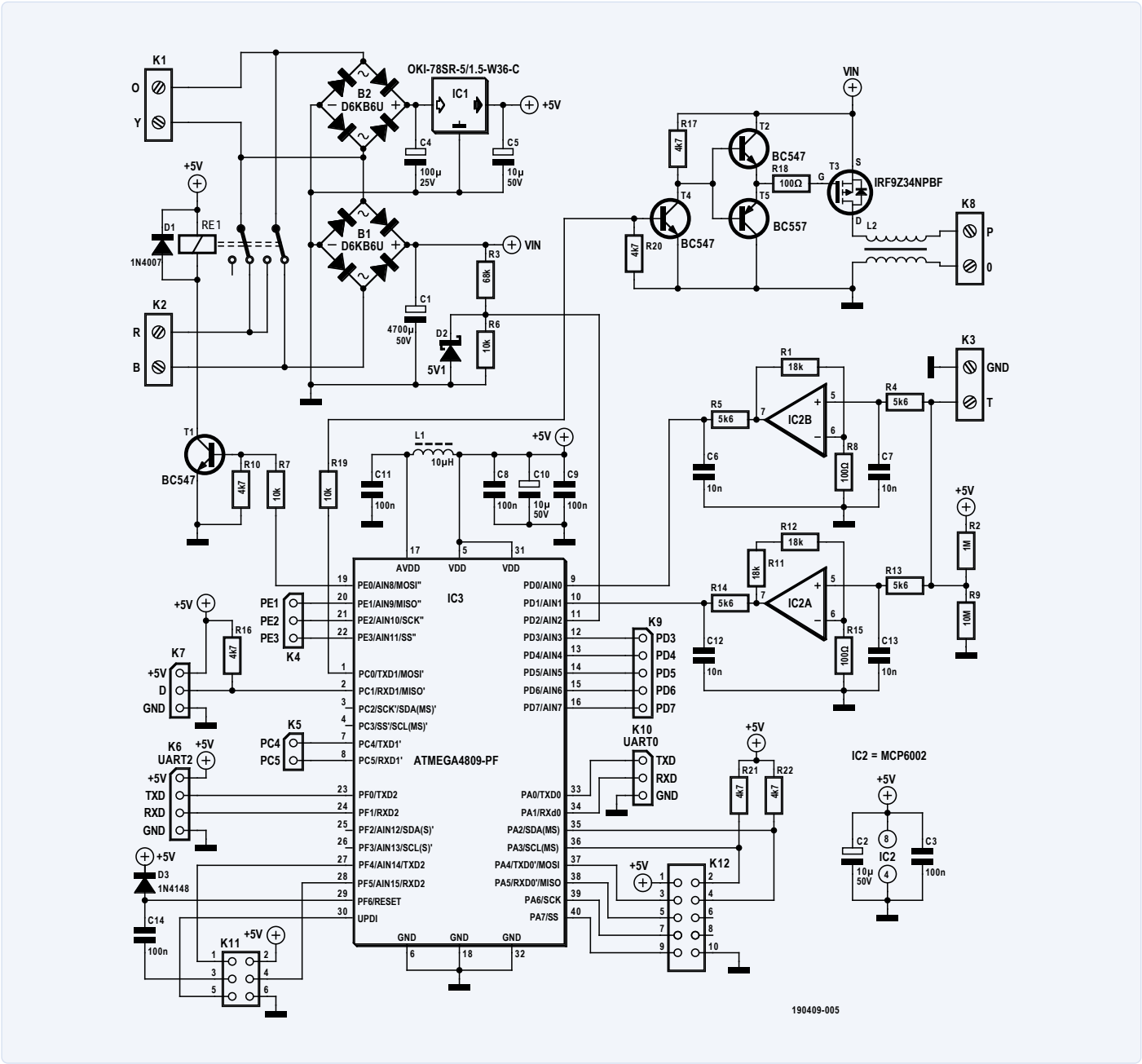



*Figure 1. Schéma de la carte principale.*

pendant le soudage. Nostra culpa : excellente en itinérance, elle est loin d'être idéale pour une station de soudage de labo. Nous avons facilement résolu ce problème en intégrant l'alimentation électrique, dans le boîtier, surtout qu'un transformateur toroïdal de 60 VA est un argument de poids ! L'afficheur à LED, plus grand que l'écran OLED de l'ancienne station, améliore la lisibilité du réglage de la température. Il affiche moins d'informations, mais la température (et c'est le plus important) peut maintenant être lue d'un coup d'œil.

## Pourquoi changer...

Sur le plan matériel, la réalisation précédente fonctionnait bien, et nous n'y avons pas vraiment touché (voir **fig. 1**). Les circuits de mesure et de contrôle de la température sont d'ailleurs similaires. Cependant, pendant la phase de prototypage, nous avions déjà quelques doutes sur le bien-fondé de la mesure du courant par résistance shunt et amplificateur de détection INA138. Cela compliquait inutilement le matériel et le logiciel de contrôle de température, et nous avons abandonné cette partie du circuit.

## Alimentation électrique

La nouvelle station de soudage est alimentée par un gros transformateur toroïdal (2 × 12 V, 60 VA), dont les deux secondaires sont connectés respectivement à K1 et K2. Le type indiqué dans la liste des composants a également deux primaires, permettant une tension de secteur de 115 VAC (enroulements en parallèle) et de 230 VAC (enroulements en série). Bien sûr, vous pouvez utiliser un transformateur avec un secondaire de mêmes caractéristiques, mais un primaire correspondant uniquement à votre tension de secteur.

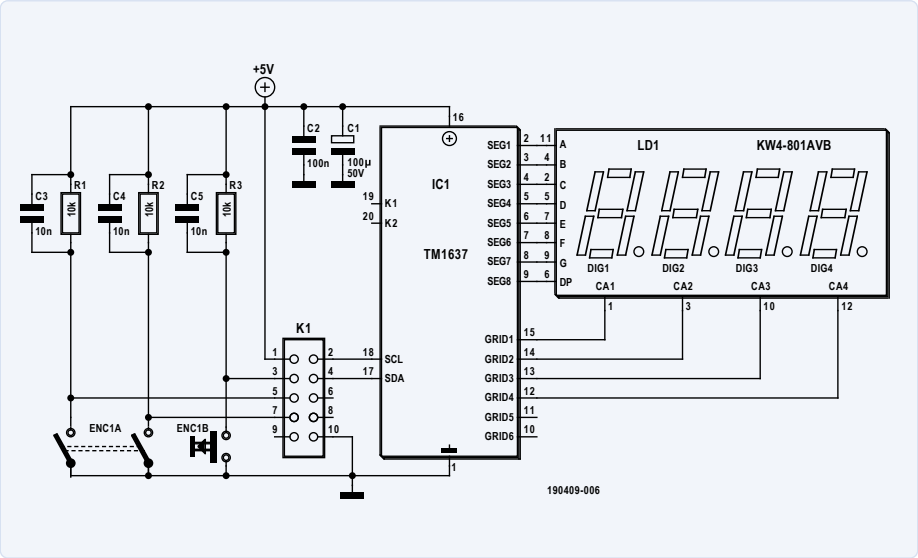



*Figure 2. Schéma de la carte affichage.*

L'un des deux secondaires est utilisé pour alimenter en +5 V les circuits logiques et de commande via le pont de diodes B2 et le convertisseur DC/DC IC1. Le microcontrôleur commande le relais RE1 par le transistor T1. Cela permet de commuter les secondaires en parallèle ou en série selon le type de fer à souder utilisé. Avec RE1 au repos, la tension la plus basse est sélectionnée. La tension alternative est redressée et lissée (B1 et C1). Le diviseur de tension R3-R6 la réduit pour qu'elle soit dans la plage d'entrée du CA/N interne du microcontrôleur. La diode Zener D2 protège contre les surtensions.

La puissance délivrée au fer à souder, c.-à-d. la température de la panne, est contrôlée par *MLI*, à l'aide des mêmes circuits (T2 à T5) que ceux utilisés dans le « poste de soudage de CMS compact » (180348-03). La self en mode commun L2 supprime les interférences RF sur le câble du fer à souder.

## Mesure de la température

Les amplificateurs opérationnels (*AOP*) IC2A et IC2B amplifient la tension de sortie du capteur de température, respectivement 361 fois (type C pour Hakko) et 181 fois (type K pour Weller et JBC). Pour adapter l'amplification, nous avons préféré deux AOP fixes à la commutation des résistances. Le type de fer à souder est réglé par le logiciel et le microcontrôleur sélectionne la tension d'alimentation et l'entrée CA/N adéquates pour mesurer la température de la panne. Une sonde de température à un fil *DS18B20* peut être connectée à K7 pour mesurer la température ambiante afin de compenser la soudure froide du thermocouple interne du fer à souder.

## Programmation et débogage

Les fabricants de microcontrôleurs comme *Microchip/Atmel* savent que les composants classiques sont toujours demandés. Ainsi, même le récent *ATmega4809* avec la « nouvelle » interface de programmation *UPDI* (*Unified Program and Debug Interface*) de Microchip est disponible en version DIP à 40 broches, idéale pour le prototypage et le bricolage. K11 fournit une UPDI au microcontrôleur. Cette interface, comme son nom l'indique, est utilisée tant pour la programmation que le débogage. Pour en savoir plus sur ce successeur de l'AVRISP (qui nous a servi pendant tant d'années !), voir le site d'Elektor Labs [2]. Cette page montre également comment exploiter une carte Arduino Uno comme interface de programmation viable pour ce projet. Notez qu'à ce jour, l'interface de programmation/débogage bon marché de Microchip *Snap UPDI*, n'est pas prise en charge dans l'EDI Arduino.

Le connecteur K12 est la connexion à la carte d'affichage (afficheur et codeur rotatif), que nous verrons plus loin. La broche marquée « spare » n'est utilisée ni par le matériel ni par le logiciel, mais le sera (développement ultérieur) pour une entrée ou sortie supplémentaire sur la face avant.

Presque toutes les broches inutilisées du microcontrôleur sont acheminées vers des connecteurs SIL sur le circuit imprimé. On trouve deux *UART* (K6 et K10 respectivement) et d'autres E/S (analogiques et numériques) sur K4, K5 et K9, mais aucun d'entre eux n'a de fonction définie ni n'est pris en charge dans la version actuelle du microprogramme.

## Carte d'affichage PCB 190409-2

De nombreux afficheurs à quatre chiffres, pilotés par *I2C* et prêts à l'emploi, sont abordables. Ils sont souvent commandés par un pilote d'afficheur à LED *TM1637* produit par *Titan Microelectronics*. Ces afficheurs sont largement utilisés par les fabricants et conviennent pour notre station de soudage. Trois chiffres auraient suffi pour afficher la température, mais les quatre chiffres se sont avérés être l'option la plus abordable. La face avant accueille aussi le codeur rotatif avec bouton-poussoir. Nous voulions que le câblage entre les cartes soit aussi simple et ordonné que possible. Nous avons donc décidé d'inclure tous ces composants sur la carte d'affichage et de la relier à la principale par un seul câble plat. À cet effet, nous avons réalisé notre propre circuit en utilisant le TM1637 et un quadruple afficheur à LED à 7 segments, voir **fig. 2**. Le TM1637 est disponible en boîtiers DIP et SOIC et bien que nous voulussions une station de soudage avec uniquement des composants traversants, nous avons fait une exception pour ce CI en fournissant des empreintes pour les deux versions de boîtier sur la carte. En effet, l'expédition des CI DIP que nous avions commandés en Asie fut retardée à cause de la COVID-19, mais nous avions des modules tout faits équipés en SOIC au labo. Le dessoudage d'un SOIC sur l'un d'eux était le moyen le plus rapide de reprendre le développement de notre prototype.

## Logiciels

Le code lui-même comprend plusieurs parties qui s'imbriquent. Elles définissent des classes et des objets créés à partir de celles-ci. Quiconque travaille avec l'EDI Arduino utilise des objets et des classes, souvent sans le savoir. En bref, une classe est un plan de construction servant à créer l'objet adéquat dans le code, comme un bâtiment est érigé selon le plan de l'architecte. Une introduction à l'orientation objet du C++ dépasse largement le cadre de cet article, c'est pourquoi nous aborderons ici les objets et les classes de manière très abstraite.

Prenons par exemple l'affichage, appelé *frontend* (frontal) dans le logiciel. Il sert à visualiser les valeurs (par ex., température,

menu et codes d'erreur) transférées du cœur de la station vers le frontal. Il ne serait pas très judicieux d'effectuer dans le code les adaptations nécessaires à chaque type d'écran (OLED, LCD alphanumérique ou afficheur à 7 segments).

C'est là que les classes et les objets entrent en jeu : pour le noyau du logiciel, le frontal est toujours un écran, qui fournit toujours les mêmes fonctions. Ainsi, avec les classes adéquates (plans de construction), le logiciel d'une station de soudage peut être conçu pour le matériel décrit ici. Si par ex., vous préférez utiliser un OLED de 0,96" au lieu de l'afficheur à LED, seul le code de ce frontal doit être intégré.

Le programme du « poste de soudage de CMS compact » (180348-03) avait été conçu pour cela. Le noyau de l'actuelle station est presque le même, seules les parties périphériques ont été adaptées au matériel.

Le suivi de la température a aussi subi quelques ajustements. Le schéma du circuit montre deux AOP distincts, l'un pour les thermocouples de type K et l'autre pour ceux de type C, de sorte que les pannes et fers de différents fabricants sont utilisables. En outre, un DS18B20 optionnel permet une compensation de soudure froide ; un capteur adéquat est automatiquement détecté et évalué. En fonction de la panne à souder configurée dans les paramètres de la station, l'entrée A0 ou A1 est automatiquement sélectionnée et la tension convertie à la température correcte. Puis, si disponible, la compensation de soudure froide est appliquée. À ce stade, un capteur *OneWire DS18B20* mesure la température. Nous utilisons quelques astuces pour le lire. Au démarrage du logiciel, le système recherche ce capteur OneWire, si aucun n'est détecté, le logiciel le note et aucune tentative ne sera faite pour le rechercher à nouveau. En présence d'un capteur détecté, la lecture de température est divisée en deux étapes. La 1ère étape s'effectue toutes les 2 s et lance la conversion de la température instantanée délivrée par le capteur. Il a besoin d'au moins 750 ms pour la mesurer, ensuite la valeur est prête à être lue et traitée. Comme le MCU ne peut s'arrêter 750 ms, un drapeau signale le début de conversion. Si le début de conversion remonte à plus d'une seconde, 2e étape, la valeur est extraite du capteur et il est noté que la lecture est terminée. Une seconde plus tard, la séquence de lecture suivante est lancée. De cette façon, le MCU peut effectuer d'autres tâches pendant la conversion de la température.

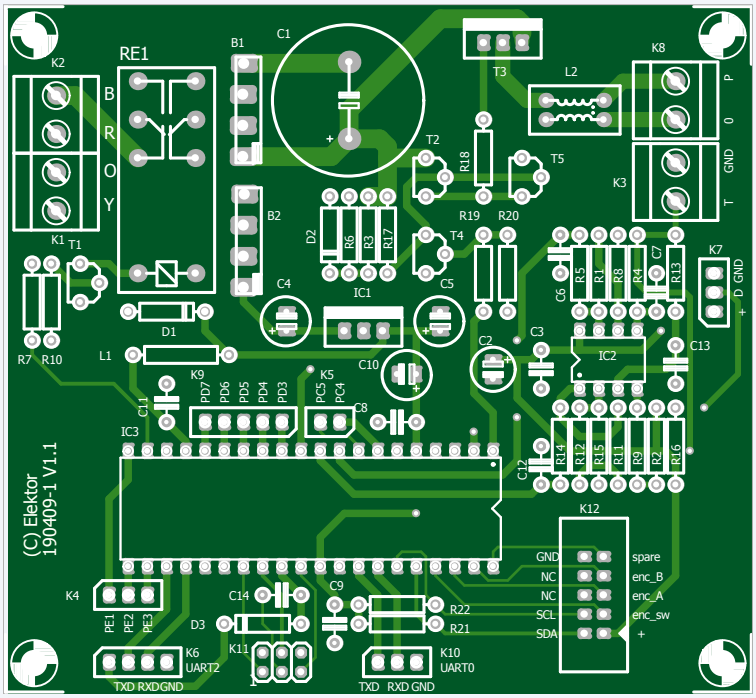



*Figure 3a. Circuit imprimé principal.*

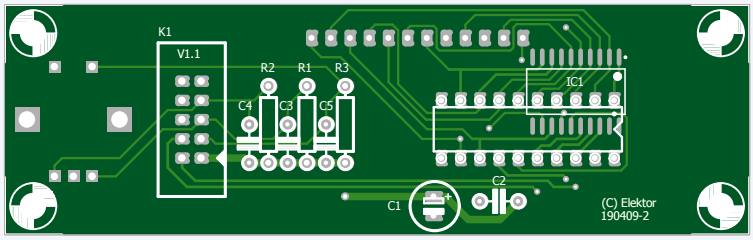



*Figure 3b. Circuit imprimé pour l'affichage.*

## MLI avec le temporisateur A

Les temporisateurs (*timer*) de l'ATmega4809 sont différents de ceux de ses prédécesseurs. Dans ce nouveau MCU, le timer A peut fournir six sorties MLI, ce qui est très pratique pour piloter des servos, des LED et d'autres composants. C'est aussi la configuration standard de *MegaCoreX*, le matériel qui doit être installé dans l'EDI Arduino pour l'ATmega4809. Le timer A est un timer à 16 bits avec trois sorties MLI, mais il est configuré par défaut comme un timer à 8 bits avec six sorties. C'est assez pour des servos, mais insuffisant pour commander la puissance de la station de soudage à 10 kHz. C'est pourquoi, le timer A doit être reconfiguré en mode 16 bits pour la MLI de notre station de soudage.

## Codeur rotatif et bouton-poussoir

Le codeur est lu par une routine sur interruption du timer. Toutes les 250 µs, les broches du codeur sont lues, et le sens de rotation est déterminé à partir des quatre derniers états des broches. Comme les contacts du codeur sont mécaniques, la rotation ne produit pas de transitions nettes et sans filtrage il y aurait toujours du bruit sur les signaux des broches d'entrée de l'ATmega. Trois filtres passe-bas RC et un filtrage logiciel permettent de suppri-

## LISTE DES COMPOSANTS

### de la carte mère

**Résistances**

R1,R11,R12 = 18 kΩ
R2 = 1 MΩ
R3 = 68 kΩ
R4,R5,R13,R14 = 5,6 kΩ
R6,R7,R19 = 10 kΩ
R8,R15,R18 = 100 Ω
R9 = 10 MΩ
R10,R16,R17,R20,R21,R22 = 4,7 kΩ

**Inductances**

L1 = self d'arrêt 10 µH, 130 mA
L2 = self d'arrêt en mode commun 10 A, Laird CM2545x171B-10

**Condensateurs**

C1 = 4700 µF, 50 V, pas de 10 mm, 22×41 mm
C2,C5,C10 = 10 µF, 50 V, pas de 2 mm, 5×11 mm
C3,C8,C9,C11,C14 = 100 nF, 50 V, X7R, pas de 5,08 mm
C4 = 100 µF, 50 V, pas de 3,5 mm, 8×11 mm
C6,C7,C12,C13 = 10 nF, 50 V, pas de 5 mm, X7R

**Semi-conducteurs**

D1 = 1N4007, 1000 V, 1 A
D2 = diode Zener simple, 5,1 V, 500 mW
D3 = 1N4148, 100 V, 200 mA, 4 ns
B1,B2 = pont redresseur D6KB6U, 600 V, 6 A
T1,T2,T4 = BC547C, 45 V, 100 mA, 500 mW, hfe=400
T3 = IRF9Z34NPBF, MOSFET-P, 55 V, 17 A, 100 mΩ
T5 = BC557C, –45 V, –100 mA, 500 mW, hfe=400
IC1 = convertisseur DC/DC OKI-78SR-5/1,5-W36-C, 5 V, 1,5 A
IC2 = AOP double MCP6002-E/P
IC3 = MCU 8 bits ATmega4809-PF

**Autres**

RE1 = relais de puissance, 5 VDC, DPDT, 8 A, Schrack RT424005
K1,K2,K3,K8 = bornier 5,08 mm, 2 voies, 630 V
K4,K7,K10 = connecteur, barrette sécable, 1 rangée, 3 voies, vertical
K5 = barrette sécable, 1 rangée, 2 voies, verticale
K6 = barrette sécable, 1 rangée, 4 voies, verticale
K9 = barrette sécable, 1 rangée, 5 voies, verticale
K11 = barrette sécable, 2 rangées, 6 voies, verticale
K12 = connecteur à 10 voies, pas de 2,54 mm

**Divers**

Transformateur secteur toroïdal 60 VA, 2×115 V, 2×12 V, MCTA060/12
Fusible primaire 20 mm, 630 mA @240 VAC
Fusible primaire 20 mm, 1,25 A @115 VAC
Connecteur K & B 59JR101-1FR-LR, CEI série 42R (connecteurs secteur, Conrad 736709)
Connecteur IDC à 10 voies (2×)
Câble plat à 10 voies, 20 cm env.
Circuit imprimé 190409-1 V1.1

### de la carte d'afffichage

**Résistances**

R1,R2,R3 = 10 kΩ

**Condensateurs**

C1 = 100 µF, 50 V, pas de 3,5 mm, 8×11 mm
C2 = 100 nF, 50 V, X7R, pas de 5,08 mm
C3,C4,C5 = 10 nF, 50 V, pas de 5 mm, X7R

**Semi-conducteurs**

LD1 = affichage LED à 7 segments à quatre chiffres KW4-801AVB (Luckylight)
IC1 = circuit spécial de pilotage de LED, I²C, TM1637

**Autres**

ENC1 = codeur avec bouton-poussoir, Bourns PEC11R-4225F-N0024
K1 = connecteur à 10 voies, pas de 2,54 mm
Circuit imprimé 190409-2 V1.1

mer le bruit. Le bouton-poussoir du codeur est traité par la même fonction et donc également contrôlé toutes les 250 µs.

## Défauts...

Toutes les 50 ms, la température réelle du fer à souder est comparée à la consigne. Si la station détecte une température supérieure à 650 °C, c'est interprété comme un défaut du capteur et le chauffage est coupé pour éviter la surchauffe. Si la station ne détecte pas d'augmentation de température de la panne après 6 s de chauffage, c'est aussi considéré comme un défaut et le chauffage est coupé. Pour un défaut du capteur de température, la station affiche *E-01*, pour un défaut du chauffage *E-03*. Une brève pression sur le bouton du codeur rotatif, permet de quittancer le défaut, et la station tentera de reprendre son fonctionnement normal.

## Fabriquer le matériel

Les fichiers Gerber des deux cartes (**fig. 3**) sont téléchargeables ici [3], vous pouvez les utiliser pour commander ces cartes. Souder des composants traversants sur ces cartes est aisé. Notez que la carte d'affichage comporte la plupart des composants sur la face supérieure, avec une double empreinte pour IC1 pour utiliser soit la version SOIC, soit la version DIL. L'afficheur à LED et le

## WEB LINKS


[1] **« poste de soudage de CMS compact », Elektor, 01/2019 »** : www.elektormagazine.fr/180348-03
2] **Programmeur UPDI pour ATmega4809 et ATtiny816/817** : www.elektormagazine.fr/labs/arduino-for-updi-programming-for-atmega4809-and-tiny816817-with-jtag2updi
[3] **Téléchargements de ce projet** : www.elektormagazine.fr/190409-04
[4] **Fichiers STEP du boîtier** : github.com/DK1CMB/Elektor-SolderironCase
[5] **Fichiers STEP du bouton** : www.thingiverse.com/tracert/designs

codeur rotatif doivent être soudés sur la face inférieure de la carte.

Notre petite station précédente était fournie avec un circuit imprimé fonctionnel, mais sans boîtier. Certains de nos lecteurs ont eu du mal à trouver ou réaliser un boîtier adéquat. Cette fois-ci, le kit DIY va jusqu'au boîtier imprimable en 3D qui a été aimablement construit en externe par Caroline Claußen. Tous les fichiers STEP peuvent être téléchargés ici [4] et imprimés sur les imprimantes 3D les plus courantes, comme l'Anycubic I3 Mega-S (voir « Produits »). Un avertissement cependant : certaines pièces sont longues à imprimer, mais ne laissez jamais une imprimante 3D en marche sans surveillance, elles fonctionnent avec des courants élevés et peuvent prendre feu.

Savez-vous pourquoi les outils de CAO comme KiCad et Altium intègrent des modèles 3D pour les composants ? Parce que ça facilite la conception d'un boîtier. Ces outils de CAO de circuits imprimés peuvent exporter votre création en fichier STEP 3D et les outils qui gèrent ce format peuvent les utiliser. Presque tous les logiciels de création 3D permettent de concevoir un joli boîtier autour du modèle 3D du circuit imprimé. Ici, il a fallu imprimer neuf pièces. Au total, il faut compter jusqu'à 24 h d'impression, ce délai peut varier en fonction de l'imprimante. Si toutes les pièces ont été imprimées, il reste à les assembler. Il y a quelques pièces non imprimables en 3D : une entrée secteur CEI avec interrupteur intégré et porte-fusible, avec les vis et écrous pour la fixer sur le panneau arrière. Comme le montre la **figure 4**, des fils et connecteurs sont nécessaires pour relier l'interrupteur et le fusible aux primaires du transformateur. Lors du raccordement du transformateur, notez que les bornes de K1 et K2 sont repérées par l'initiale des couleurs des isolants des fils des secondaires : noir (B), rouge (R), orange (O) et jaune (Y). Avec un transformateur de marque ou type différent, assurez-vous que les repères de « début d'enroulement » des fils, d'habitude représentés par des points dans la fiche technique, correspondent au transformateur Multicomp que nous avons utilisé. Et bien sûr : n'oubliez pas d'installer le fusible sur le primaire (secteur) du transformateur.

Pour assembler l'avant, le dos et le capot, utilisez des vis M2,5. Pour le circuit imprimé, il y a quatre pattes qui doivent être placées dans la base, puis il faut mettre le circuit imprimé au-dessus et insérer les vis, car cela compensera l'imprécision que peut avoir votre imprimante 3D. Avec un peu de patience, vous obtiendrez un résultat proche de la **fig. 5**.

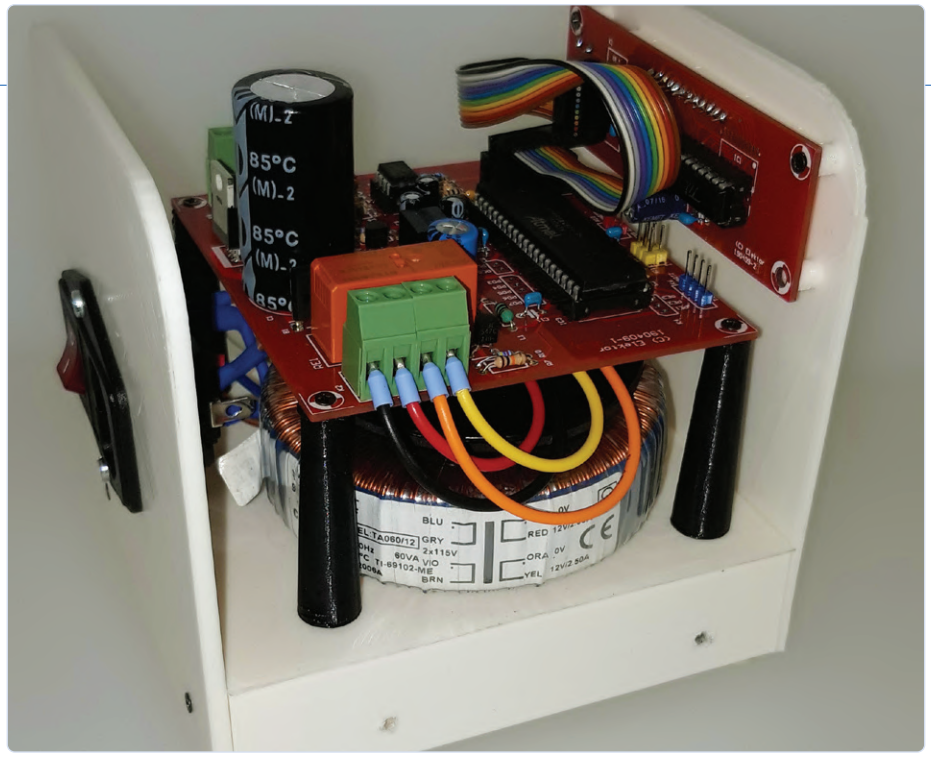



*Figure 4. Entrée secteur avec porte-fusible et interrupteur.*

**Note : cette station de soudage a été testée de manière approfondie avec une panne Weller RT. Malheureusement, nous n'avions ni fer à souder Hakko FX-8801 ni JBC T245 disponibles pour les tests, mais la station devrait fonctionner correctement avec ces derniers.**

Le raccordement du fer à souder (**fig. 6**) peut se faire de différentes manières. Il n'y a pas de trou prévu à cet effet dans le boîtier. Pour notre prototype pour fer à souder Weller RT, nous avons monté un connecteur jack audio de 3,5 mm. Ce n'est pas la meilleure solution possible, mais elle fonctionne. N'utilisez pas

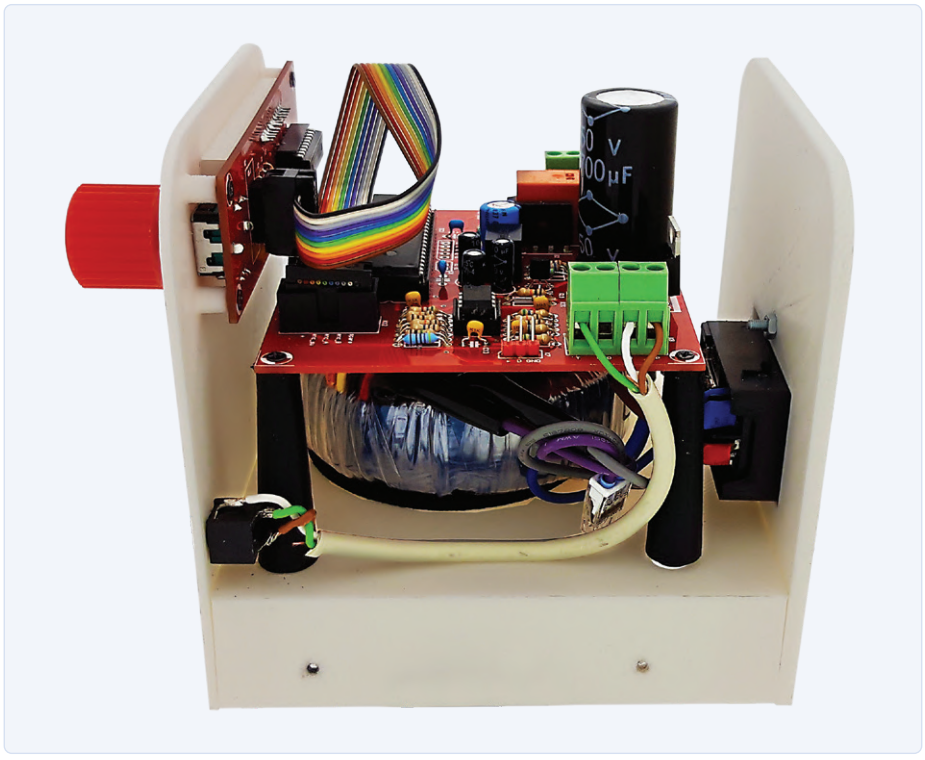


*Figure 5. Intérieur du boîtier imprimé en 3D.*

de rallonge audio flexible, la section du fil de cuivre est trop petite pour supporter le courant de l'élément chauffant. Nous avons essayé et au lieu de chauffer la panne, en une minute le câble est devenu brûlant à la main. Nous y avons remédié en utilisant un câble de plus forte section. Dernier point, mais non des moindres : le bouton. Chacun ayant ses préférences de taille et de forme en matière de boutons, il suffit de chercher « rotary knob » sur *Thingiverse* et de choisir le plus séduisant. Nous avons utilisé ici par ex. le modèle de *Domain Mittu* [5]. Nous l'avons imprimé en TPU, plus doux, ce qui donne une bonne prise en main. Pour le capot rouge, nous avons choisi le PTEG, plus résistant que le PLA, mais moins facile à manipuler. Il est parfaitement possible d'imprimer le capot en PLA.

## Utilisation

À la première mise sous tension, avant de brancher un fer à souder, il faut maintenir le bouton-poussoir du codeur rotatif enfoncé jusqu'à l'ouverture du menu de sélection du type de fer à souder afin de choisir entre C0, C1 et C2 qui correspondent respectivement à Hakko FX-8801, JBC T245 et Weller RT. Appuyez sur le codeur rotatif pendant dix secondes pour mémoriser ce choix et la station redémarrera avec la tension correcte (12 V/24 V) et l'entrée analogique adaptée à la mesure de la température. Bien entendu, cette sélection est mémorisée et conservée après l'arrêt. En mode normal, le codeur rotatif est utilisé pour régler la température du fer à souder. L'afficheur passe en luminosité maximale lorsque ce réglage est modifié. Le nouveau réglage de la température est appliqué 5 s après le relâchement du bouton rotatif. L'afficheur repasse alors à la luminosité normale et affiche la température réelle de la panne. Lorsque le fer à souder chauffe, le point décimal le plus à gauche de l'afficheur clignote.

Nous pensons que cette nouvelle station de soudage dissipe la plupart des objections faites à la station précédente, mais des améliorations restent toujours possibles. Afin de garantir ces possibilités d'amélioration, nous avons acheminé presque toutes les broches inutilisées du MCU vers des connecteurs d'extension de la carte mère. Si vous faites vous-même des modifications ou extensions susceptibles d'intéresser d'autres lecteurs, merci de nous en faire part !

(190409-04)


*Figure 6. Connexions de la RT Weller.*

**Des questions ou des commentaires ?**
Si vous avez des questions ou des commentaires sur cet article, envoyez un courriel aux auteurs (mathias.claussen@elektor.com et luc.lemmens@elektor.com) ou contactez Elektor (redaction@elektor.fr).

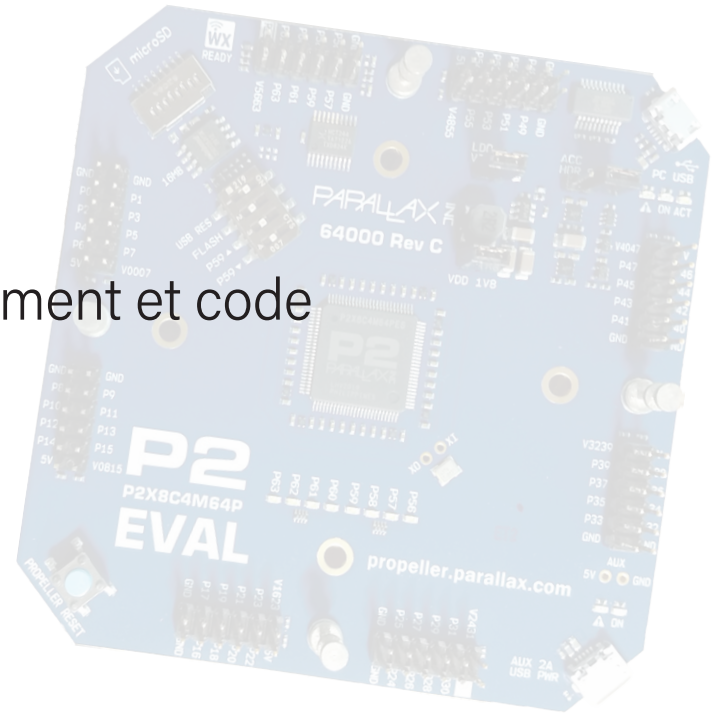
**Contributeurs**
Idée, conception et texte : **Mathias Claussen, Luc Lemmens**
Illustrations : **Patrick Wielders**
Rédaction : **Luc Lemmens**
Mise en page : **Giel Dols**
Traduction : **Yves Georges**

**PRODUITS**

- **Station de soudage numérique Weller WE 1010 (kit pédagogique)**
  www.elektor.fr/weller-we-1010-digital-soldering-station-education-kit
- **Extracteur de fumée avec éclairage à LED**
  www.elektor.fr/fumetractor-with-led-light
- **Imprimante 3D i3 Mega-S de Anycubic (kit)**
  www.elektor.fr/anycubic-i3-mega-s-3d-printer-kit

# Propeller 2 de Parallax (2)

## Environnement de développement et code




**Mathias Claußen** (Elektor)

Dans la première partie de cette série d'articles, je vous ai présenté le Propeller 2, le nouveau microcontrôleur multicœur avancé de Parallax. Voyons maintenant comment utiliser le langage Spin2 pour piloter une LED.

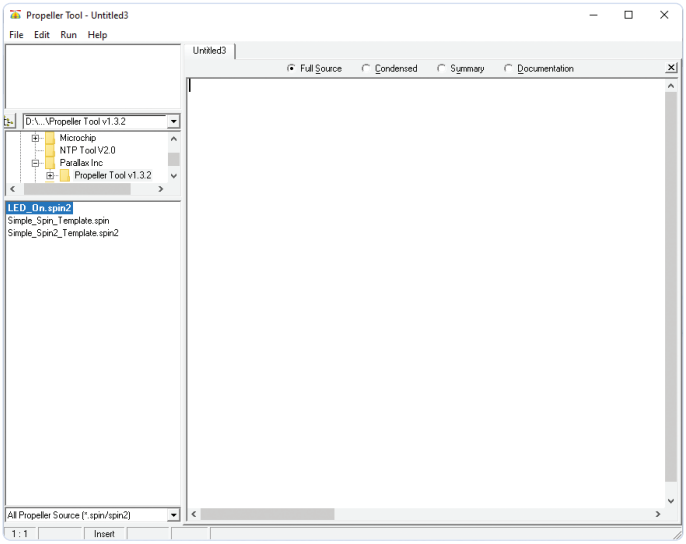



*Figure 1. Interface utilisateur de Propeller Tool.*

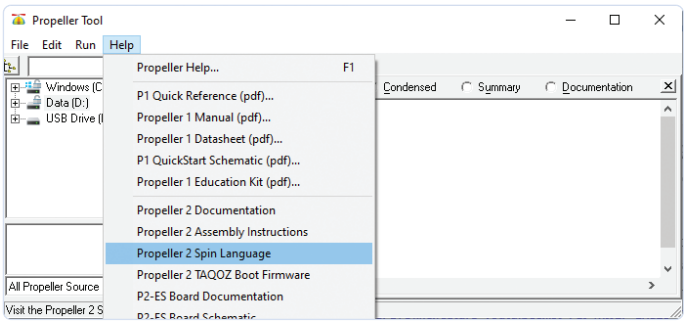



*Figure 2. Accès à la documentation Spin2 via le menu Aide (Help).*

Maintenant que le Propeller 2 de Parallax et ses caractéristiques vous sont familiers, je vais vous faire découvrir l'environnement de développement et l'écriture du code. Lisez la suite pour apprendre à piloter une LED.

### Accéder aux LED intégrées

Commençons par l'environnement logiciel et la commande de l'une des LED intégrées. Nous ferons notre développement sous Windows 10 avec les outils fournis par le constructeur. Nous mettrons en place la configuration logicielle minimale pour compiler et charger le code sur le Propeller 2.

Parallax a inclus les langages Spin/Spin2 pour le Propeller 2 dans son outil de développement Propeller Tool Version 2.3 Alpha. Si vous préférez le langage assembleur, vous pouvez utiliser PNut comme environnement de développement ou un assembleur en ligne (*inline*). Si vous aimez coder en C/C++ comme moi, malheureusement, le compilateur et l'environnement de développement ne sont pas encore prêts pour ces langages. Actuellement, les équipes de développement s'efforcent de travailler sur le compilateur C/C++ et la vidéo de présentation [1] du Propeller 2 montre où ils en sont.

### Environnement de développement

Vous pouvez télécharger le logiciel Propeller Tool 2.3 Alpha sur [2]. Téléchargez d'abord Propeller Tool 1.3.2 et placez l'exécutable de Propeller Tool 2.3 Alpha dans son répertoire d'installation. Une fois le processus terminé, vous pouvez démarrer l'éditeur et commencer à coder. L'interface utilisateur de l'outil est illustrée à la **figure 1.**

### Assembleur ou Spin2 ?

La question est : Spin2 ou Assembler pour commencer l'écriture du code ? Pour rester simples, nous continuerons, pour l'instant, avec Spin2. Spin2 est un langage interprété, comme le BASIC, mais différent. Vous trouverez dans le menu d'aide (*Help*) les commandes et les références du langage Spin2 (**figure 2)**.

Pour démarrer avec le langage Spin2, vous pouvez utiliser le manuel de référence intégré, mais gardez à l'esprit qu'il n'est pas encore terminé.

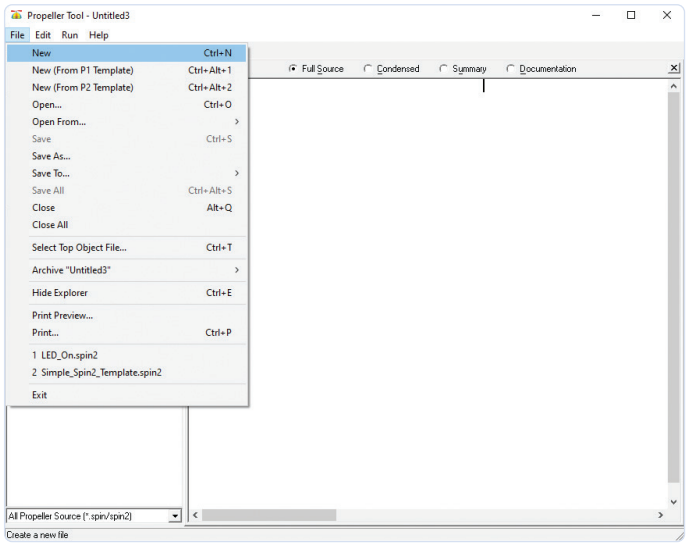



*Figure 3. Création d'un nouveau projet Spin2.*


*Figure 4. Détail des LED connectées aux broches 56 à 63 du microcontrôleur.*

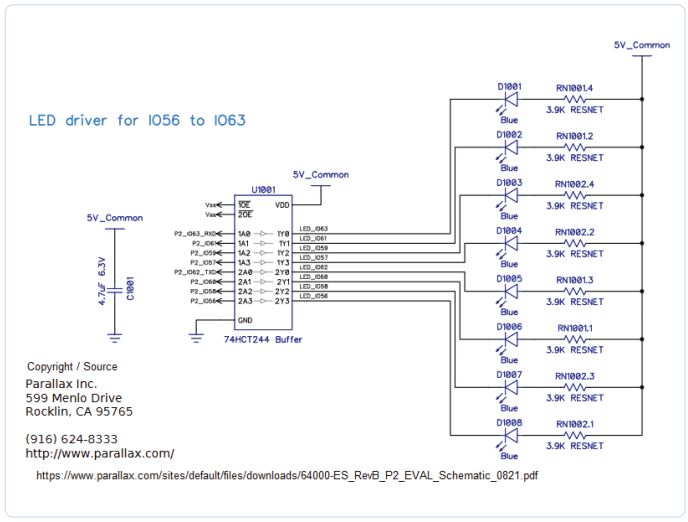



*Figure 5. Schéma des LED pilotées par les tampons du 74HCT244.*

Cela vous mènera à la documentation officielle des fonctions mises en place dans le microcontrôleur Propeller 2.
Comme ce langage est de type interprété, vous aurez besoin d'environ six cycles d'instructions, soit douze cycles d'horloge, pour exécuter une commande. Ce n'est pas le moyen le plus rapide de faire avancer les choses en matière de cycles d'horloge, mais le langage Spin2 est plus facile à prendre en main que l'assembleur. Comme le Spin2 est un langage de haut niveau, la plupart des commandes cachent beaucoup d'instructions en assembleur, mais elles sont plus intuitives. Vous pouvez créer un nouveau projet Spin2 en suivant la **figure 3**.

## Broches d'entrées-sorties (E/S)

Avant de commencer à coder, examinons brièvement le fonctionnement des broches d'E/S. Nous les utilisons ici comme des broches d'E/S ordinaires et verrons plus tard toutes leurs fonctions en détail. En général, nous devons définir la broche choisie comme sortie pour pouvoir la placer à un niveau logique bas ou haut. Sur la carte d'évaluation, vous pouvez apercevoir clairement les LED intégrées, celles-ci sont marquées P56 à P63 (**figure 4**). Il s'agit d'un groupe de LED connectées aux broches 56 à 63 du Propeller 2, nous choisissons la première (en face du marquage P56) pour nos expériences. Si vous observez attentivement le schéma, vous remarquerez qu'elles ne sont pas directement reliées au microcontrôleur, mais qu'elles passent par un circuit intégré 74HCT244, contenant huit tampons TTL. Notez également que l'anode de chaque LED est connectée de manière permanente à la tension VCC. La cathode de chaque LED est reliée à une broche d'E/S du microcontrôleur, elle est ramenée à la masse (0 V) grâce au tampon, ce qui permet d'allumer la LED correspondante (**figure 5**).

## LED inversées

Pour notre code, cela signifie que placer la broche du microcontrôleur à un niveau logique haut éteindra la LED alors qu'un niveau bas l'allumera. Avec cette logique inversée en tête, nous devons dire à notre code de placer la broche P56 à un niveau haut pour l'éteindre. De plus, si nous ne configurons rien, les broches sont configurées par défaut en entrées, les tampons agiront comme si elles étaient placées à un niveau logique haut et chaque LED connectée sera éteinte. En résumé, pour allumer notre LED avec le langage Spin2, nous devons :

- Initialiser le microcontrôleur et au moins un cœur (*cog* en anglais)
- Configurer la broche 56 en tant que sortie
- Placer la broche 56 à un niveau logique bas
- Ne plus rien faire

Première étape : l'initialisation. Dans le cas présent, elle est déjà effectuée pour nous et nous n'avons pas besoin de nous en soucier. Notre exemple en langage Spin2 n'a besoin que de quelques lignes. Notre code commencera par la fonction `pub main ()` suivie de la méthode `pinwrite()` (**figure 6**). La méthode `pinwrite()` est intégrée au langage Spin2 et permet de définir le niveau logique d'une ou plusieurs broches grâce à une valeur donnée. Ici, nous utilisons uniquement la broche 56, à laquelle notre LED est connectée. Nous passons en deuxième argument un '0' pour placer la broche à un niveau logique bas et ainsi, allumer notre LED. La dernière ligne `repeat` forcera le processus dans une boucle sans fin, car il n'y a plus de code sous l'instruction `repeat` l'obligeant à ne rien faire d'autre qu'à se répéter indéfiniment. Sans l'instruction `repeat` ici, le processus s'arrêterait après une itération, les broches d'E/S ne seraient plus pilotées.

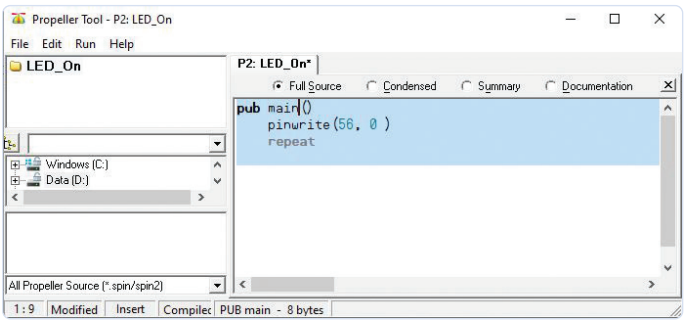



*Figure 6. Code complet pour allumer la LED de la broche 56.*

## Chargement du code dans le Propeller 2

Si votre code est prêt, vous pouvez le charger sur la carte d'évaluation du Propeller 2 et l'exécuter. Pour cela, il suffit de connecter la carte à votre PC par l'un de ses ports USB et de sélectionner dans le menu *Run->Compile Current->Load RAM* pour charger le code directement dans la RAM du microcontrôleur et l'exécuter. La LED connectée à la broche 56 doit s'allumer, bravo, vous avez réussi une sorte de « Bonjour tout le monde ! » (*Hello, world!*). Maintenant vous vous demandez : « Et je peux la faire clignoter ? ». Oui, c'est possible. Le langage Spin2 contient l'instruction `WAITMS` qui permet de retarder l'exécution du code, par exemple `WAITMS(500)` provoque un retard de 500 ms. Vous savez que le code après `REPEAT` est exécuté en boucle. Maintenant vous devriez pouvoir adapter le code pour que la LED clignote. Utilisez les informations que vous avez rassemblées jusqu'à présent pour modifier le code afin d'obtenir une LED clignotante. Ne vous inquiétez pas, nous donnerons la solution plus tard.

Nous pouvons maintenant piloter une broche d'E/S, l'étape suivante consiste à déterminer comment envoyer des données série. La plupart d'entre nous sont habitués à cet apprentissage progressif, en particulier lors des premiers pas avec un microcontrôleur, comme un AVR. Comme nous aurons besoin des fonctions Smart-Pin, nous nous familiariserons également avec celles-ci.

200479-B-03

**Des questions ou des commentaires ?**
Vous avez des questions ou des commentaires sur cet article. Envoyez un courriel à l'auteur (mathias.claussen@elektor.com) ou contactez Elektor (redaction@elektor.fr).

**Contributeurs**
Auteur : **Mathias Claußen**
Rédacteurs : **Jens Nickel** et **CJ Abate**
Mise en page : **Giel Dols**
Traduction : **Nicolas Bishop**

### LIENS

[1] **Parallax, « Propeller 2 Live Forum Early Adopter Series - C Programming with Eric Smith », 6 juillet 2020 :** https://bit.ly/propeller2-c

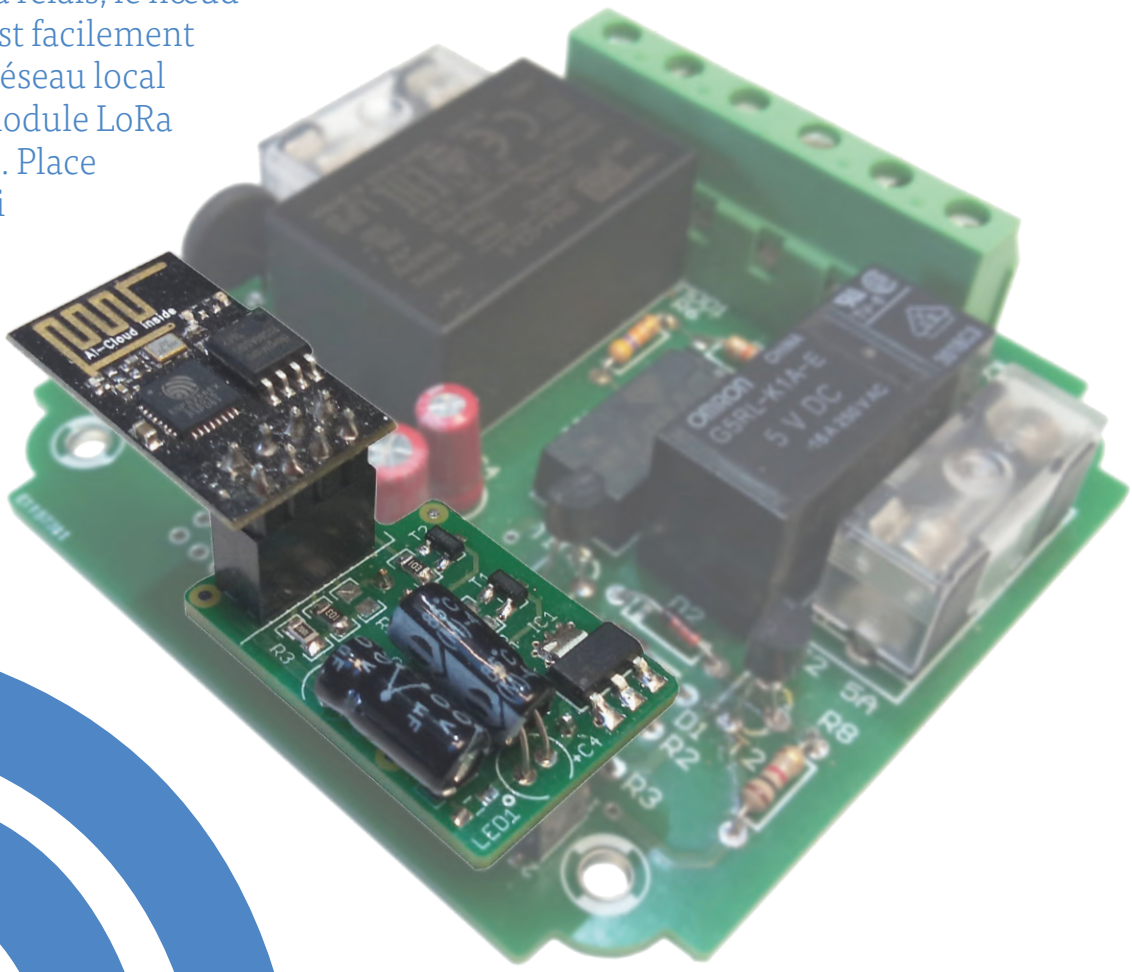
[2] **Téléchargement de Propeller Tool 2.3 Alpha :** https://propeller.parallax.com/p2.html#software

# wifi pour le nœud LoRa d'Elektor

## Interrupteur intégré dans Home Assistant avec ESPHome

**Clemens Valens** (Elektor)

Avec sa belle carte à relais, le nœud LoRa d'Elektor [1] est facilement utilisable dans un réseau local en remplaçant le module LoRa par un modèle Wifi. Place à l'interrupteur wifi d'Elektor.




Un interrupteur télécommandé a fait l'objet d'un article dans le magazine *Elektor* de mars-avril 2020 [1]. Il dispose du retour d'état du relais et communique sur le réseau LoRa. Logé dans un boîtier étanche IP66, il est adapté à une utilisation en extérieur. C'est également un projet modulaire avec le relais et le circuit d'alimentation sur une carte et la partie communication LoRa sur une autre.

Il a retenu mon attention, car je cherche un interrupteur extérieur, facile à intégrer à mon système de domotique. Ce dernier repose sur le logiciel Home Assistant et le réseau wifi, mais pas sur LoRa. Je suis sûr qu'on peut y ajouter LoRa, mais je n'ai pas envie de chercher. À la place, j'espère qu'en remplaçant simplement le module LoRa par un module Wifi, on peut y exécuter le *framework* ESPHome, qui fonctionne très bien avec Home Assistant (**fig. 1**).

## Description de l'interface

La première chose à faire quand on essaie d'adapter quelque chose est de trouver comment s'y connecter. Le schéma de la carte de l'interrupteur n'est pas très compliqué (**fig. 2**). En haut, nous avons l'alimentation électrique avec un convertisseur CA/CC MOD1 qui fournit 5 V avec quelques éléments de protection et de filtrage.

Au centre, il y a un relais bistable RE1, qui allume et éteint l'alimentation de la charge. Un relais bistable ressemble beaucoup à un interrupteur mécanique, car tout comme lui il conserve son état même après avoir été mis hors tension. Deux signaux permettent de basculer son état : *Set* et *Reset*. Ils sont pilotés par deux MOSFET T1 et T2 commandés par des signaux actifs à l'état haut.

Vous noterez que le relais est protégé par un fusible F2 de 5 A alors que le relais peut commuter jusqu'à 16 A. C'est parce que les pistes du circuit imprimé ne peuvent pas supporter un tel courant, donc le fusible les protège contre la fusion. La carte de l'interrupteur peut supporter des charges allant jusqu'à environ 1 kW.

Dans le coin inférieur gauche, on trouve un optocoupleur IC1 en parallèle avec la charge. Il est activé en même temps que la charge. Sa LED rend son transistor conducteur et la sortie passe à l'état bas. Lorsque la charge n'est pas alimentée, la sortie est à l'état haut. Le signal CA à 50 ou 60 Hz est filtré par C4 et R4 pour maintenir un niveau bien stable.

Le connecteur dans le coin inférieur droit permet d'accéder à tous les signaux nécessaires. S1 est destiné à un bouton-poussoir placé sur l'appareil, pour pouvoir aussi actionner l'interrupteur localement.

Il faut savoir que l'optocoupleur est connecté à une alimentation de 3,3 V, mais il n'y en a pas sur la carte. Cette tension doit être fournie par la carte de commande.

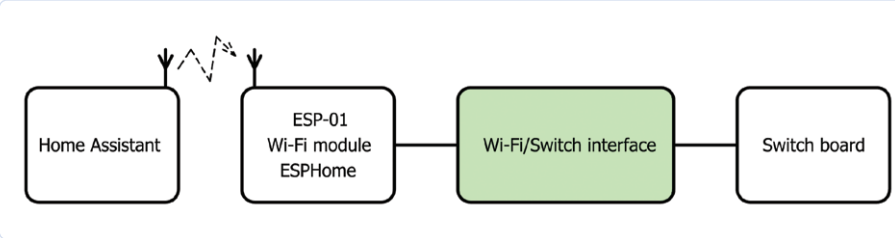



*Figure 1. Vue d'ensemble du système. Dans cet article, nous nous intéressons principalement au bloc vert clair.*

## Le module Wifi ESP-01

Nous pouvons maintenant établir les spécifications de la carte de commande wifi qui pilotera la carte de l'interrupteur :

- Il faut deux entrées numériques, une pour lire la sortie de l'optocoupleur et une pour S1 ;
- Il faut deux sorties numériques pour commander T1 et T2 ;
- Il faut fournir 3,3 V pour l'optocoupleur.

Maintenant, quand vous lisez « quatre ports d'E/S numériques, wifi et 3,3 V », vous pensez bien sûr immédiatement au module ESP-01, car il a exactement quatre ports GPIO, le wifi

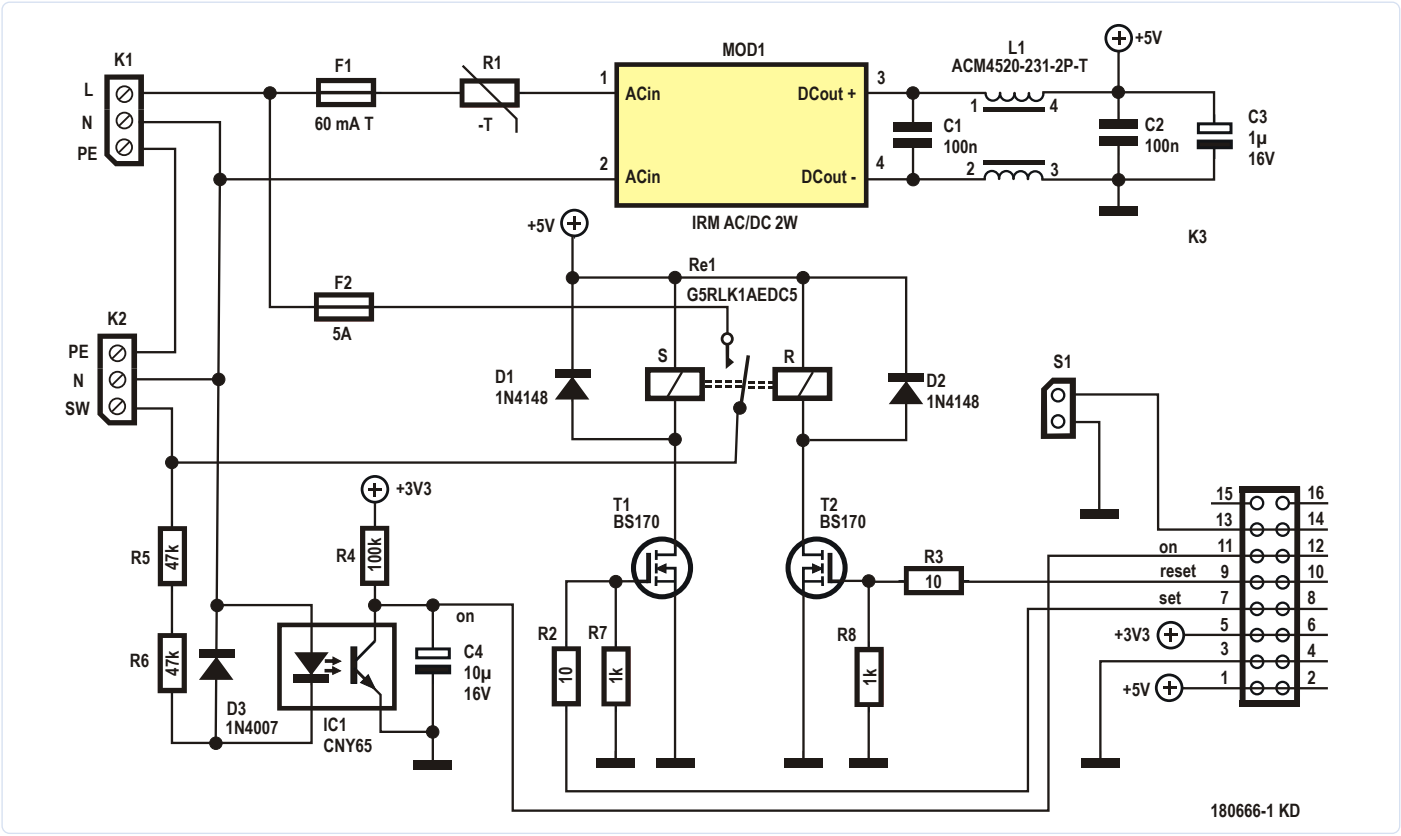



*Figure 2. La carte de l'interrupteur du nœud Lora d'Elektor (mars-avril 2020, [1]) est centrée autour d'un relais bistable.*

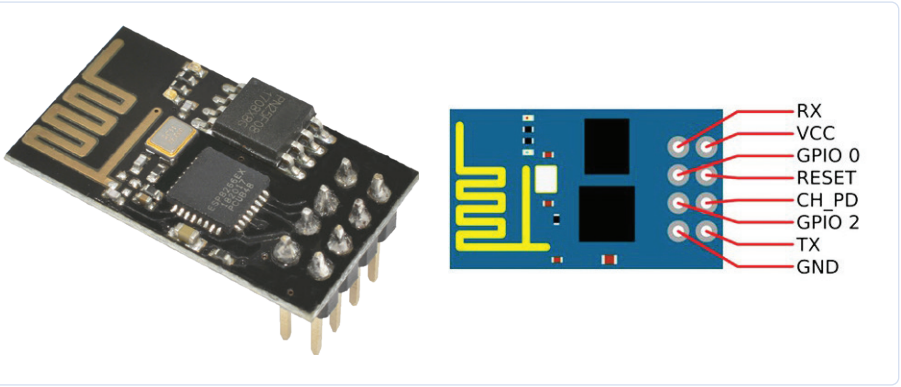



*Figure 3. Le module ESP-01, petit et bon marché, est équipé d'un microprocesseur ESP8266EX avec wifi intégré. Il possède quatre ports d'E/S (GPIO 0, GPIO 2, RX & TX) et nécessite une alimentation en 3,3 V (VCC).*

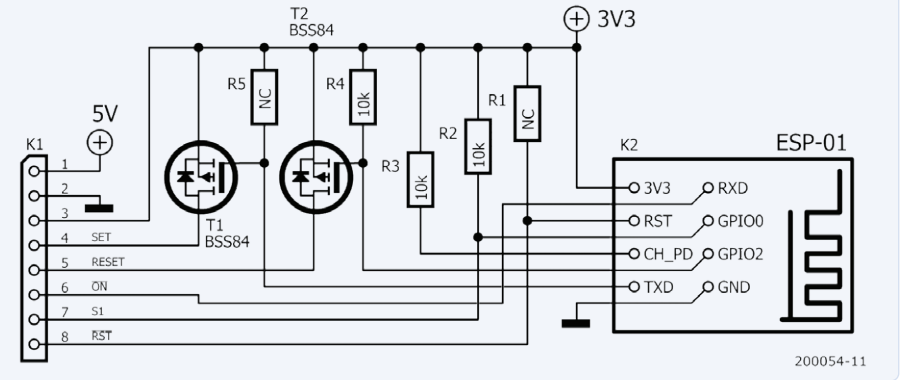



*Figure 4. Une façon possible de connecter un module ESP-01 à la carte de l'interrupteur LoRa. Les MOSFET découplent les entrées des sorties.*

## LISTE DES COMPOSANTS

### Résistances

Toutes des 5%, 50 V, 0,1 W, 0805
R1, R2, R3, R4, R5 = 10 kΩ
R6 = 470 Ω

### Condensateurs

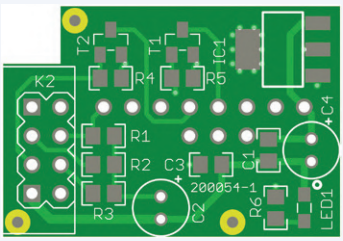
C1, C3 = 100 nF, 0805
C1, C4 = 10 µF, 16 V, pas de 2 mm

### Semi-conducteurs

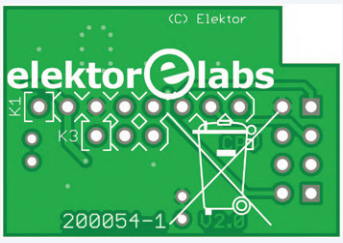
IC1 = LD1117AS33
LED1 = LED, verte
T1, T2 = BSS84

### Divers

K1 = barrette mâle à 8 broches, pas de 2,54 mm
K2 = barrette femelle 2×4 broches, pas de 2,54 mm
K3 = barrette mâle à 3 broches, pas de 2,54 mm
Circuit imprimé 200054-1




*Vue à 150% de la taille réelle*




et fonctionne en 3,3 V. Ce module répandu, basé sur l'ESP8266EX est bon marché et facile à trouver (fig. 3).
Il y a cependant un mais. Les quatre ports GPIO de l'ESP-01 doivent être manipulés avec précaution, car ils déterminent également la manière dont le module fonctionnera à la mise sous tension. Pour un fonctionnement normal, les ports GPIO0, GPIO1 et GPIO2 doivent être à l'état haut à la mise sous tension, comme indiqué dans les notes dans le tableau « *Pin Definitions* » de la fiche technique de l'ESP8266EX. Beaucoup d'utilisateurs de l'ESP8266 le savent pour les GPIO0 et GPIO2, mais tout le monde ne le sait pas pour le GPIO1, alias TXD, la sortie série.

## Association des ports GPIO aux signaux de commande

Grâce aux résistances R7 et R8, deux des quatre signaux de la carte de l'interrupteur, « Set » et « Reset », sont toujours à l'état bas à la mise sous tension. Lorsque la charge est alimentée au moment où la carte de l'interrupteur est mise sous tension, le signal « On » est aussi à l'état bas. N'oubliez pas que cela est possible parce que le relais est bistable et conserve son dernier état, même lorsqu'il n'est pas alimenté.
L'entrée S1 est flottante. Son état à la mise sous tension est déterminé par la carte de commande wifi, à moins que quelqu'un n'appuie sur le bouton-poussoir lors de la mise sous tension du système…
En résumé, nous avons donc affaire à quatre signaux qui peuvent tous être à l'état bas à la mise sous tension. Ils doivent être connectés à quatre ports, dont trois doivent être à l'état haut à la mise sous tension. Utiliser un module ESP-01 dans cette situation est donc un défi.

## La solution : ajouter des transistors

Je suis parvenu à la solution suivante (**fig. 4**) :

- RXD (alias GPIO3) est la seule broche qui peut être connectée librement. Je l'ai donc connectée à la sortie de l'opto-coupleur car son niveau à la mise sous tension est imprévisible. RXD va être une entrée.
- Lorsque le GPIO0 est à l'état bas à la mise sous tension, le module démarre en mode de programmation Flash. Cela peut être utile pour mettre à jour le microprogramme, c'est pourquoi je l'ai connecté à S1. Par conséquent, GPIO0 va être une entrée aussi.
- Il reste alors GPIO2 et TXD (alias GPIO1) qui doivent donc devenir les sorties. Pour

les découpler des faibles impédances créées par R7 et R8 sur la carte de l'interrupteur, j'ai inséré des P-MOSFET avec une résistance de tirage sur leurs grilles. Cela garantit qu'à la mise sous tension, le GPIO2 et TXD verront un niveau haut. Après le démarrage de l'ESP-01, on peut les configurer en sorties à l'état bas.

## Le pilote de sortie

Le pilote de sortie est illustré à la **figure 5**. Lorsque le signal GPIO passe à l'état bas, le P-MOSFET BSS84 conduit. Cela tire la grille du N-MOSFET BS170 vers le haut, donc il va conduire aussi et la bobine du relais est alimentée. Lorsque le signal GPIO est à l'état haut, le P-MOSFET BSS84 se bloque. Le N-MOSFET BS170 se bloque également à cause de la résistance de tirage de 1 kΩ sur sa grille, et la bobine du relais n'est pas alimentée. La résistance de 10 kΩ sur la grille du BSS84 assure que la broche GPIO est tirée vers le haut à la mise sous tension.

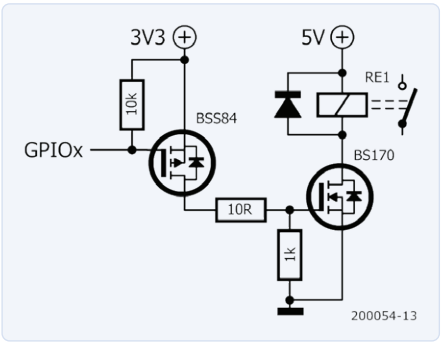



*Figure 5. Le pilote de sortie : un P-MOSFET pilote un N-MOSFET.*

## Compléments

Pour l'alimentation en 3,3 V, j'ai simplement rajouté un régulateur à faible chute sur la ligne d'alimentation en 5 V (**fig. 6**). En pratique, R1, R6 et LED1 ne sont pas nécessaires car déjà présentes sur le module ESP-01, d'où la valeur « NC » (c.-à-d. non câblé). R5 n'est pas nécessaire non plus, mais je l'ai ajouté au cas où. La valeur appropriée pour R1 et R5 serait de 10 kΩ. R6 pourrait valoir quelque chose comme 470 Ω.
J'ai conçu un petit circuit imprimé pour l'interface entre le module wifi et la carte de l'interrupteur (**fig. 7**). Les fichiers se trouvent en [3].

## Fichier YAML pour ESPHome

Une fois les entrées et les sorties définies, nous pouvons établir le fichier YAML de

**Configuration YAML pour l'interrupteur wifi**

```
esphome:
  name: wifiswitch
  platform: ESP8266
  board: esp01_1m

wifi:
  ssid: "my_ssid"
  password: "my_passphrase"

# Connexion impossible car les broches UART0
# sont utilisées pour les E/S
logger:
  baud_rate: 0 # UART désactivé.

# Activer l'API Home Assistant
api:

# Activer la programmation Over-the-Air
ota:

output:
  - platform: gpio
    id: relay_set
    pin: GPIO1 # TXD, peut être rappelé au niveau bas au démarrage.
    inverted: true
  - platform: gpio
    id: relay_reset
    pin: GPIO2 # Ne doit pas être rappelé au niveau bas au démarrage.
    inverted: true

switch:
  - platform: template
    name: "Wi-Fi switch"
    id: wifiswitch
    turn_on_action:
      # Impulsion sur la broche Set.
      - output.turn_on: relay_set
      - delay: 0.1s
      - output.turn_off: relay_set
    turn_off_action:
      # Impulsion sur la broche Reset.
      - output.turn_on: relay_reset
      - delay: 0.1s
      - output.turn_off: relay_reset
    # Utilise l'état de l'optocoupleur comme état de l'interrupteur.
    lambda: return id(optocoupler).state;

# Bouton-poussoir & optocoupleur.
binary_sensor:
  - platform: gpio
    name: "Wi-Fi switch pushbutton"
    id: pushbutton
    pin:
      number: GPIO0 # pas de rappel au démarrage.
      inverted: true
    on_press:
      then:
      - if:
          condition:
            binary_sensor.is_on: optocoupler
          then:
            switch.turn_off: wifiswitch
          else:
            switch.turn_on: wifiswitch
  - platform: gpio
    name: "Wi-Fi switch optocoupler"
    id: optocoupler
    pin:
      number: GPIO3 # RXD, peut être rappelé au niveau bas au démarrage.
      inverted: true
```

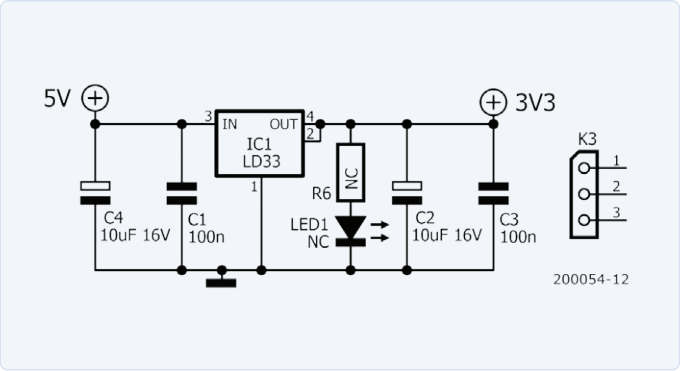



*Figure 6. Un simple convertisseur de 5 V à 3,3 V. K3 est chargé de fournir un peu plus de stabilité mécanique lorsque le module wifi est branché sur le connecteur de la carte de l'interrupteur.*

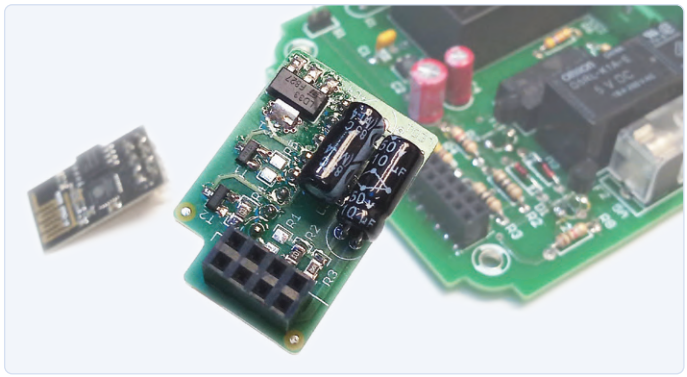


*Figure 7. L'interface entre le module Wifi EPS-01 et la carte de l'interrupteur est disposée sur un petit circuit imprimé.*

configuration d'ESPHome (voir l'encadré, disponible en [3]). C'est un peu plus compliqué que d'habitude en raison du relais bistable qui utilise deux broches au lieu d'une. ESPHome et Home Assistant connaissent un composant appelé « *cover* » qui peut commander de tels périphériques, mais il est destiné aux portes de garage, aux stores et autres qui ont un bouton d'ouverture et de fermeture. Ils ont également une icône spéciale. Notre interrupteur n'est qu'un interrupteur et nous voulons donc qu'il se comporte comme tel et qu'il en ait l'apparence. On peut le faire à l'aide d'un modèle d'interrupteur qui permet de spécifier séparément le comportement à la mise en marche et à l'arrêt.

Tout d'abord, nous définissons le GPIO1, alias TXD, comme une sortie inversée nommée *relay_set*. GPIO2 est déclaré comme une sortie inversée nommée *relay_reset*. Nous définissons également GPIO3 comme entrée numérique active à l'état bas pour la sortie de l'optocoupleur.

Ensuite, nous spécifions une courte impulsion de 100 ms sur la sortie *relay_set* comme action de mise en marche et nous faisons de même sur la sortie *relay_reset* comme action d'arrêt. Pour le retour d'état, nous renvoyons l'état de l'optocoupleur.

Pour que le bouton-poussoir S1 fonctionne comme prévu, c'est-à-dire qu'appuyer dessus commute la charge, nous définissons le GPIO0 comme entrée numérique active à l'état bas. Lorsqu'on appuie sur S1, on vérifie d'abord l'état de l'optocoupleur. Si la charge est active, alors on désactive le relais ; si la charge était inactive, alors on l'active. Une clause « *if-then-else* » fera l'affaire ici.

## C'est fini !

Après avoir compilé ESPHome avec ce fichier YAML et flashé le module ESP-01, l'interrupteur wifi devrait apparaître dans Home Assistant, prêt pour l'automatisation. Il peut maintenant être installé et « commuter une charge ».

200054-04

**Des questions, des commentaires ?**

Vous avez des questions ou des commentaires sur cet article ? Envoyez un courriel à l'auteur (clemens.valens@elektor.com) ou contactez Elektor (redaction@elektor.fr).

**Contributeurs**

Idée, conception, texte et illustrations : **Clemens Valens**

Mise en page : **Giel Dols**

Traduction : **Denis Lafourcade**

### PRODUITS

- **Module wifi ESP8266 ESP-01 (150445-91, réf. 17326)**
  www.elektor.fr/17326
- **Livre en anglais « IoT Home Hacks with ESP8266 » (réf. 19158)**
  www.elektor.fr/19158
- **Nœud LoRa avec retour d'état - circuit imprimé nu de l'unité esclave (180666-1, réf. 19143)**
  www.elektor.fr/180666-1

### LIENS

[1] **L. Lemmens et M. Claussen, « nœud LoRa d'Elektor » , Elektor, mars/avril 2020** : www.elektormagazine.fr/180666-02

[2] **C. Valens, « la domotique, c'est facile avec… », Elektor, sept./oct. 2020** : www.elektormagazine.fr/200019-02

[3] **La page du projet** : www.elektormagazine.fr/labs/wifi-switch

# module cellulaire

## Même pas peur !

**Tam Hanna** (République slovaque)

Nombreux sont ceux pour qui la conception d'un système de communication à base de module cellulaire sans fil relève de la magie noire. On est pourtant bien loin de la vérité. Tam Hanna a récemment eu le plaisir de superviser divers projets de ce type. Voici une compilation de ses notes de laboratoire pour vous aider à réaliser vos propres modules.

### Pourquoi concevoir votre propre module cellulaire ?

Si les modules de communication 3G et 4G prêts à l'emploi et les adaptateurs USB sont faciles à se procurer sur l'internet, le diable, qui les rend impropres à votre application au point de devoir reconcevoir le système, se cache généralement dans les détails. Certains opérateurs et applications utilisent par exemple des bandes bizarres incompatibles avec le module (la bande 13 américaine étant particulièrement problématique), tandis que pour d'autres systèmes, l'ajout d'un adaptateur USB ordinaire est impossible. Et puis il y a les applications qui nécessitent des antennes peu communes (pensez au SMA) ou qui ont aussi besoin d'une fonction GPS.

### Pour commencer

Tout commence par la recherche d'un module de communication adapté à la ou aux bandes de communication souhaitées et qui réponde aux critères de taille et de coût. La qualité de l'équipe d'assistance technique du fabricant, du vendeur ou du distributeur est également importante, car vous pourrez avoir besoin d'aide à un moment donné. L'auteur a trouvé celle de Quectel particulièrement accueillante et serviable. Le distributeur YAWiD de Gemalto en Allemagne et le distributeur polonais de Telit étaient bien aussi. En revanche, l'auteur a placé le fameux fabricant suisse U-blox à l'autre bout de l'échelle, car il n'a pas réussi à obtenir de réponse de leur part à cause de leur système de courrier électronique. Mais il s'agit bien sûr d'une expérience personnelle. Tout dépend du pays où vous vous trouvez, et vous pouvez avoir d'autres expériences.

L'étape suivante consiste à acheter un kit de développement ou d'évaluation pour le module de communication. Une fois que vous l'avez, forcez vos ingénieurs en logiciel à faire fonctionner le logiciel de gestion du module à 100%, en mettant l'accent sur 100%. Ne laissez aucun problème en suspens qui pourrait - et va probablement, selon M. Murphy - invalider votre projet par la suite. En outre, la logique veut que l'équipe d'assistance technique du fabricant soit votre premier contact pour les problèmes logiciels que vous pourriez rencontrer. La qualité du support montrée ici sera un test décisif en prévision du support pour le matériel.

Une fois traité le logiciel, il est temps d'obtenir la documentation du module, qui se cache généralement derrière un formulaire de connexion à un compte sur le site web du fabricant. Deux documents sont particulièrement intéressants : le schéma de la carte d'évaluation et le document de conception du matériel décrivant le système réel. Ce que vous recherchez, c'est l'empreinte ou le schéma d'implantation du circuit imprimé du module et un tableau énumérant les broches et leurs fonctions.

### Allo l'interface !

La plupart des modules de communication ne connaissent que deux moyens de dialoguer avec l'hôte : un port série et un port USB. Les autres interfaces comme I2C et SPI sont en général réservées à la connexion de périphériques et n'ont pas d'intérêt pour nous ici. Il nous reste donc l'USB et le port série. Commençons par l'USB, l'interface favorite des systèmes hôtes basés sur Linux grâce à la facilité d'intégration des pilotes. Le module se présente souvent sous la forme d'un ou plusieurs périphériques TTY. Bien qu'on ait vu fonctionner l'USB 2.0 avec une paire de fils à connecteurs Dupont, ne faites pas n'importe quoi : essayez de respecter les règles de base de conception des circuits imprimés, et placez le connecteur aussi près que possible du module. Mais ne vous préoccupez pas de l'impédance exacte des pistes USB.

Pour le port série, c'est différent. Bien que l'auteur n'ait pas encore vu de système basé sur Linux utiliser le port série pour communiquer avec un module de communication, ils sont fréquemment exposés sur ses cartes. Les fabricants de modules réservent généralement un de leurs UART à l'interface de débogage et s'il n'est pas exposé, la carte ne peut pas être analysée lorsqu'elle est envoyée au laboratoire du

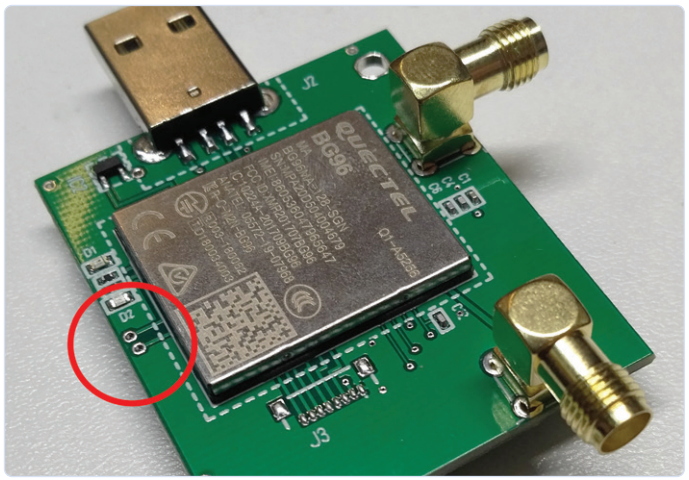
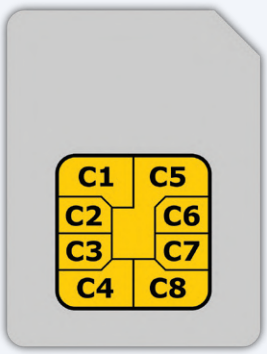



*Figure 1. Les deux pastilles non connectées sur le côté gauche sont reliées au module de débogage UART. Le connecteur est monté sur demande uniquement.*

**Tableau 1. Le brochage de l'interface de la carte SIM (carte à puce) est normalisé.**

| Broche | Nom | Description |
|---|---|---|
| C1 | VCC | Entrée d'alimentation VCC, +5 V CC (utilisation facultative par la carte) |
| C2 | reset | Signal de réinitialisation, utilisé pour réinitialiser les communications de la carte. Utilisé seul (signal de réinitialisation fourni par le dispositif d'interface) ou en combinaison avec un circuit de contrôle de réinitialisation interne (utilisation facultative par la carte). Si la réinitialisation interne est mise en œuvre, l'alimentation en tension par VCC est obligatoire. |
| C3 | horloge | Fournit à la carte un signal d'horloge à partir duquel est dérivé le cadencement de communication des données. |
| C4 | réservé | AUX1, utilisé en option pour les interfaces USB et autres applications |
| C5 | GND | Masse (tension de référence) |
| C6 | Vpp | Entrée de tension de programmation (en option). Ce contact peut être utilisé pour fournir la tension nécessaire à la programmation ou pour effacer la mémoire non volatile interne. La norme ISO/CEI 7816-3:1997 a désigné ce contact comme tension de programmation : une entrée pour une tension plus élevée pour programmer la mémoire persistante (par ex. EEPROM). La norme ISO/CEI 7816-3:2006 le désigne comme SPU, pour un usage standard ou propriétaire, en entrée et/ou en sortie. |
| C7 | E/S | Entrée ou sortie pour les données série (*half-duplex*) vers le circuit intégré à l'intérieur de la carte. |
| C8 | réservé | AUX2, utilisé en option pour les interfaces USB et autres applications |




*(Source : pinoutguide.com [1])*

fabricant. Il s'ensuit que le port série doit être accessible d'une manière ou d'une autre. L'auteur prévoit souvent un connecteur positionné sans grand effort, puis ne le monte pas sur la production en série (**fig. 1**).

## Composants divers

Presque tous les modules de communication font partie d'une famille. Même si on peut concevoir des cartes compatibles avec plusieurs types de modules, ce n'est pas toujours rentable. En particulier pour les unités à faible consommation (pensez au LTE M1, etc.), renoncer à la compatibilité croisée permet d'utiliser une alimentation plus petite, et donc de faire des économies. En outre, dans certains cas, les circuits de réinitialisation et autres ne sont pas nécessaires pour tous les membres de la famille, ce qui permet de réaliser une petite économie supplémentaire.

Si une impulsion de réinitialisation bien définie est nécessaire, utilisez un circuit superviseur de réinitialisation de microcontrôleur tel que le STM1001 de STMicroelectronics. Ces composants SOT23 ne coûtent presque rien et sont plus fiables qu'un bricolage avec des composants RC.

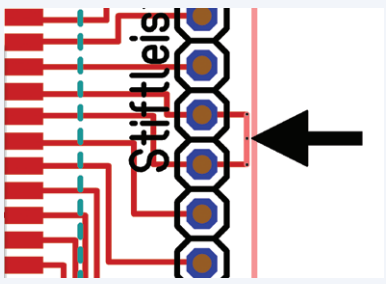



*Figure 2. La piste sur laquelle pointe la flèche peut être coupée facilement et reconnectée si nécessaire.*

Il n'y a pas grand-chose à dire concernant l'alimentation électrique – le choix entre un régulateur à découpage ou un régulateur linéaire ne dépend que de vous. Mais ne lésinez pas sur les condensateurs de découplage. Les modules 4G par exemple sont notoirement « impulsifs » en termes de consommation d'énergie.

Il faut également tenir compte des « broches magiques ». Lisez attentivement la description matérielle de chaque broche pour être sûr de bien comprendre quoi faire. Certaines doivent être reliées à un niveau de tension spécifique, tandis que d'autres, comme la broche de sélection du mode de démarrage, doivent être exposées au moyen d'un pad ou au moins d'un via. En cas de doute, vous pouvez généralement utiliser une disposition telle que celle de la **figure 2** qui vous laissera le choix après réception du prototype.

## Carte SIM

La connexion de la carte SIM au module de communication est toujours un défi intéressant. Les connecteurs de cartes SIM ont généralement des contacts numérotés alors que le module utilise des libellés (par ex. « Vcc », « Rst », « Data » et « Clk »). Heureusement, les contacts de la carte SIM sont standardisés (voir **tableau 1**). Selon le circuit, une résistance de rappel vers le haut entre Data et Vcc peut être nécessaire. Parfois, des condensateurs de découplage sont utilisés sur Rst, Clk et Data. L'auteur omet généralement les diodes de protection ESD, car sur ses cartes, l'utilisateur n'aura jamais accès au connecteur de carte SIM. Sur certains modules de communication, il y a ici un piège supplémentaire. On peut détecter la présence d'une carte SIM par un niveau de tension sur une broche particulière. Si vous n'utilisez pas cette fonction, assurez-vous de relier la broche au niveau de tension qui correspond à « carte SIM présente ». Si vous prévoyez de l'utiliser, faites attention à la façon dont elle est mise en œuvre sur votre connecteur de carte SIM. Tenez également compte de l'espace nécessaire pour la mécanique. Certains porte-cartes SIM s'ouvrent par basculement, tandis que d'autres exigent que vous fassiez glisser la carte SIM (rigide) dans une direction précise.

Enfin, oubliez l'idée selon laquelle il faut des circuits imprimés à quatre

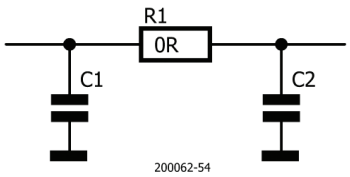



*Figure 3. Un réseau en Pi est nécessaire si on prévoit une adaptation d'antenne. On peut le neutraliser en production en réglant R1 sur 0 Ω et en omettant C1 et C2.*

couches pour les modules de communication. Tous les outils modernes de conception de circuits imprimés permettent de placer des plans cuivrés. Des plans de masse des deux côtés de la carte, reliés par de nombreux vias, permettent généralement de remplir la fonction.

## Au sujet de l'antenne

Reste l'antenne. Jusqu'à présent, l'auteur a réussi à éviter d'avoir à concevoir et évaluer une antenne sur circuit imprimé. (L'analyseur de réseau vectoriel HP 8753C qu'il avait acheté à la demande d'un client s'est depuis avéré être une « reine de hangar » de première classe, nécessitant à la fois une maintenance et des kits d'étalonnage coûteux, etc.). S'il n'y a pas d'antenne sur le circuit, il faut un connecteur sur la carte pour en raccorder une. Qu'il soit de type SMA ou U.FL, veillez toujours à consulter le concepteur de votre boîtier.

Efforcez-vous de maintenir la liaison avec le connecteur de l'antenne aussi courte que possible. De plus, utilisez un calculateur de ligne de transmission (disponible en ligne) et essayez de respecter à peu près la géométrie recommandée (c.-à-d. une bonne terre des deux côtés et pas de vias dans le trajet du signal). Bien qu'en général, il n'y ait pas de sérieux problèmes tant que les pistes sont courtes.

La plupart des montages de référence des fabricants recommandent de placer un circuit en Pi similaire à celui de la **figure 3**, au cas où. En production, il est souvent neutralisé avec une résistance de 0 Ω et les deux condensateurs ne sont pas montés. Si vous ne prévoyez pas de faire une adaptation d'antenne avancée, dans de nombreux cas vous obtiendrez de meilleurs résultats en réduisant la distance entre le module de communication et le connecteur d'antenne et en omettant le réseau en Pi.

Notez qu'il existe deux types d'antennes GPS : active et passive. Les antennes actives ont besoin d'un circuit d'alimentation fantôme composé de quelques composants passifs, que l'on trouve généralement dans le schéma de référence du fabricant du module.

## Un coup d'essai

Maintenant que nous avons passé en revue les différents écueils, passons à la réalisation. Selon la relation entre vous et les ingénieurs de l'assistance technique du fabricant évoquée précédemment, vous pouvez envoyer pour examen le schéma et le dessin du circuit imprimé. L'auteur a été particulièrement impressionné par les services de révision fournis par Quectel.

Le principal problème lors de l'assemblage manuel d'un prototype est le soudage des composants. Bien que l'auteur ait réussi à souder certains modules dans son four à refusion, la précision du placement des pochoirs et des composants ainsi que le manque d'équipement d'inspection aux rayons X font qu'il vaut mieux confier ce travail à une entreprise spécialisée dans la réalisation de prototypes (en lui rappelant d'effectuer une inspection aux rayons X).

Lorsque le prototype assemblé est du type adaptateur USB, utilisez un hub USB auto-alimenté pour les premiers tests – cela protège votre coûteux poste de travail des dommages causés à sa carte mère.

## Finir le travail

Comme pour tant d'autres tâches, le plus dur, c'est de se lancer. Presque tous les montages réalisés par l'auteur ont fonctionné au premier ou deuxième coup. Cependant, faire fonctionner le projet ne représente qu'une petite partie du travail – les problèmes logiciels par exemple peuvent compliquer l'intégration du système. Néanmoins, n'ayez pas peur ! Grâce à l'expérience durement acquise par l'auteur et présentée ici, vous pouvez vous aussi y arriver.

(200062-04)

## CERTIFICATION DES OPÉRATEURS ET DU GOUVERNEMENT !

Pour respecter totalement la loi, la version finale de votre projet peut être soumise à une certification. Mais ceci est une tout autre histoire.

**Des questions, des commentaires ?**
Si vous avez des questions ou des commentaires sur cet article, contactez Elektor (redaction@elektor.fr).

**Contributeurs**
Idée et texte : **Tam Hanna**
Rédaction : **Clemens Valens**
Mise en page : **Giel Dols**
Traduction : **Denis Lafourcade**

## LIEN

[1] **Brochage de la carte SIM :** https://pinoutguide.com/Memory/SmartCardIso_pinout.shtml

# gestion du temps avec l'ESP32 et Toggl

## Pratiquer le kit *ESP32 Basic Core* de M5Stack

**Mathias Claußen** (Elektor)

Il peut s'avérer utile de suivre les heures passées sur des projets électroniques. Différents services permettent de le faire. Nous utilisons ici Toggl.com pour vous montrer comment effectuer des requêtes web en HTTPS et comment mettre en œuvre les fonctions de base d'une interface graphique. Le tout est réalisé sur un kit de développement *ESP32 Basic Core* de M5Stack, une plate-forme de prototypage rapide pour les montages à base d'ESP32.


*Figure 1. Le kit ESP32 Basic Core de M5Stack.*

De nos jours, surtout avec le travail à domicile, le suivi de vos tâches et du temps que vous y consacrez peut vous aider à rester concentré. Cela vous permet aussi de discuter plus facilement de votre travail, que ce soit avec vos collègues ou vos clients. Il y a de nombreuses solutions sur le marché – comme openTimetool, Toggl et Kimai – qui proposent une gestion du temps et des projets et permettent aussi de produire les factures pour les clients. J'ai utilisé pour cet article le service de suivi en ligne Toggl, qui fournit également pour la gestion du temps, un greffon pour votre navigateur ou une appli pour votre téléphone. Son API ouverte permet de s'interfacer avec un système de suivi des temps, et de créer ainsi, avec ces services, votre propre application ou dispositif matériel pour suivre les temps.

### KISS

« *Keep it simple stupid* » (KISS) : ne pas compliquer les choses. Pour un système de suivi des temps, c'est le mieux que vous puissiez offrir à un utilisateur. Pour certains d'entre nous, le suivi des temps est plus une corvée qu'un plaisir. Juste appuyer sur un bouton quand on se met au travail et sur un autre quand on s'arrête serait assez simple. Comme Toggl est un service en ligne, il nous faut quelque chose qui puisse se connecter à l'internet (avec ou sans fil) pour soumettre notre requête. L'ESP32 nous vient à l'esprit pour cette tâche. Pour éviter un fouillis de fils et de composants sur une platine d'expérimentation, nous suivons le principe KISS et nous prenons un kit de développement *ESP32 Basic Core* de M5Stack (**fig. 1**), dont la documentation est accessible en ligne [1]. En plus de l'EPS32, nous disposons de trois boutons, un écran, une batterie et quelques autres périphériques dans un joli boîtier compact. Il ne manque plus que quelques lignes de code.

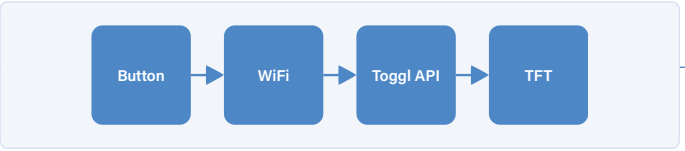



*Figure 2. Flux de données vers le service Toggl.*


*Figure 3. Flux de données pour obtenir l'état actuel.*

## Idée de base

Il suffit d'un appareil qui puisse accéder au service web Toggl et réagir à au moins un bouton pour signaler la présence ou l'absence au travail. Avec le kit M5Stack, cela signifie utiliser le wifi pour la connexion à l'internet, puisqu'un point d'accès est à portée et que nous pouvons considérer être en ligne en permanence. Comme vous pouvez le voir sur la figure 1, nous avons le choix entre trois boutons. Et puisqu'il y a un écran LCD, pourquoi ne pas afficher quelques graphiques et indiquer l'état d'occupation (au travail ou absent du bureau) ? Est-ce difficile d'accéder à Toggl ? En théorie, pas trop, car l'API est bien documentée. En clair, il faut appuyer sur un bouton et envoyer une commande appropriée au service web Toggl. Cette commande est suivie d'un changement d'état qui sera affiché pour l'utilisateur. Cela paraît simple au départ. La **figure 2** montre le flux de données entre la pression d'un bouton et l'API Toggl. La **figure 3** montre le cheminement pour une mise à jour de l'écran LCD avec les données courantes, avec un flux vers le serveur Toggl, quelque part sur le réseau, et un flux de retour vers l'ESP32. C'est aussi simple que cela en a l'air.
La partie wifi a déjà été traitée plusieurs fois et nous récupérons le code des nombreux projets d'Elektor comprenant un ESP32. Ceci nous permet d'avoir un serveur web modulaire, un mécanisme intégré pour gérer le temps et les mises à jour OTA, c'est plus qu'il n'en faut. On utilise une pile de protocoles pour la communication avec le service Toggl. La **figure 4** montre cette pile et les bibliothèques concernées. La bibliothèque `WIFIClient` incluse dans le canevas Arduino ESP32 permet de se connecter à des serveurs accessibles par le réseau wifi auquel l'ESP32 est connecté. Au sommet, vous avez la bibliothèque `HTTPClient` qui gère le protocole et les requêtes HTTP, également inclus dans le canevas Arduino ESP32. Comme Toggl exige que les données soient échangées par HTTP au format JSON, nous avons besoin d'une bibliothèque supplémentaire pour gérer JSON : ce sera la bibliothèque `ArduinoJSON`.
Outre la communication, nous avons également besoin d'une bibliothèque pour piloter l'écran du kit M5Stack. `TFT_eSPI` est une bibliothèque bien connue qui fonctionne avec l'écran du kit M5Stack et qui fournira un large éventail de fonctions prêtes à l'emploi pour le dessin de graphiques. Comme elle est compatible, qu'elle fournit toutes les fonctions nécessaires pour dessiner du texte et des graphiques et qu'elle a été utilisée dans des projets antérieurs, ce serait dommage de s'en passer. Un bref aperçu de la pile graphique qu'offre `TFT_eSPI` est présenté à la **figure 5**.
Pour commencer, une brève introduction à plusieurs sujets sera utile. Je commencerai par le protocole HTTP, indispensable pour accéder au service Toggl.

## HTTP

Le protocole *Hypertext Transfer Protocol* (HTTP), développé en 1989 au CERN, est utilisé par un client (par ex. votre navigateur web) pour demander une ressource et obtenir une réponse d'un serveur web. Par exemple, si vous voulez ouvrir www.elektor.com, votre client envoie une requête au serveur web. Pour ce faire, une connexion TCP/IP est établie et une chaîne de requête est envoyée comme indiqué dans le **listage 1**.

**Listage 1. Requête HTTP**

```
GET / HTTP/1.1
Host: www.elektor.com
User-Agent: curl/7.55.1
Accept: */*
```

La première ligne, un en-tête HTTP avec `GET`, indique au serveur que nous demandons quelque chose. Le « / » indique au serveur que nous voulons que la page web par défaut soit émise. Avec `HTTP/1.1`, nous disons au serveur que nous parlons dans la version 1.1 de HTTP. Le travail a commencé sur HTTP/3 (la troisième génération du protocole), mais pour la plupart de nos cas d'usage, HTTP/1.1 suffit.
Pour la deuxième ligne, l'hôte, il faut savoir que les noms que vous saisissez dans votre navigateur sont traduits par un service DNS en une adresse IP. Cela signifie que elektor.com sera traduit en 83.96.255.227, l'adresse IP utilisée pour la connexion TCP/IP. Derrière cette adresse IP peut se trouver un serveur web qui traitera non seulement elektor.com, mais aussi elektor.de ou elektor.nl. Pour permettre au serveur web d'identifier la page que vous voulez, la deuxième ligne avec `Host` est nécessaire. Avec `User-Agent` à la ligne 3, nous disons au serveur web quel type de client HTTP nous sommes, ici *curl*. Cela permet à un serveur web, par exemple, de présenter des pages optimisées pour votre navigateur web, car Safari se comporte différemment de Firefox. Le dernier en-tête HTTP de la ligne 4 indique à notre client d'accepter toutes sortes de contenus. Chaque requête HTTP se termine par une ligne vide.
À ce stade, le serveur web confectionnera une réponse à notre requête, en commen-

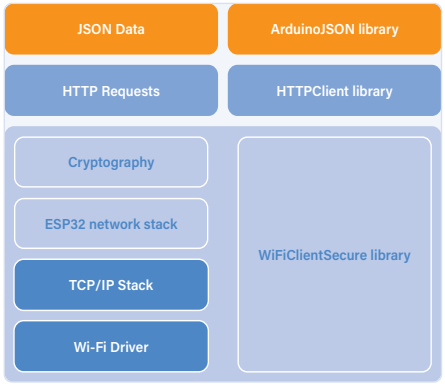



*Figure 4. Pile de protocoles et bibliothèques.*

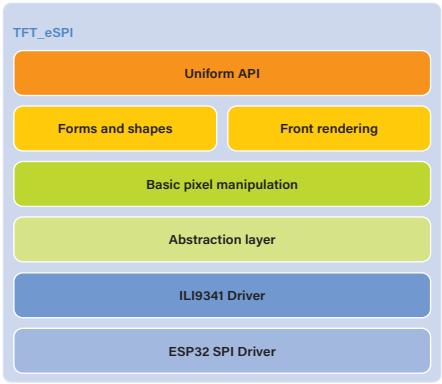



*Figure 5. Fonctions de dessins empilés pour l'écran LCD.*

**Listage 2. Réponse HTTP**

```
HTTP/1.1 200 OK
Server: unknown
Date: Tue, 17 Nov 2020 13:34:04 GMT
Content-Type: text/html; charset=UTF-8
Transfer-Encoding: chunked
Connection: keep-alive
X-Content-Type-Options: nosniff
X-XSS-Protection: 1; mode=block
X-Frame-Options: SAMEORIGIN
Strict-Transport-Security: max-age=63072000
Pragma: no-cache
Expires: -1
Cache-Control: no-store, no-cache, must-revalidate, max-age=0
Accept-Ranges: bytes
Strict-Transport-Security: max-age=63072000
```

| indice | binaire | codé |
|---|---|---|
| 0 | 000000 | A |
| 1 | 000001 | B |
| 2 | 000010 | C |
| 3 | 000011 | D |
| 4 | 000100 | E |
| 5 | 000101 | F |
| 6 | 000110 | G |
| 7 | 000111 | H |
| 8 | 001000 | I |
| 9 | 001001 | J |
| 10 | 001010 | K |
| 11 | 001011 | L |
| 12 | 001100 | M |
| 13 | 001101 | N |
| 14 | 001110 | O |
| 15 | 001111 | P |
| 16 | 010000 | Q |
| 17 | 010001 | R |
| 18 | 010010 | S |
| 19 | 010011 | T |
| 20 | 010100 | U |
| 21 | 010101 | V |
| 22 | 010110 | W |
| 23 | 010111 | X |
| 24 | 011000 | Y |
| 25 | 011001 | Z |
| 26 | 011010 | a |
| 27 | 011011 | b |
| 28 | 011100 | c |
| 29 | 011101 | d |
| 30 | 011110 | e |
| 31 | 011111 | f |
| 32 | 100000 | g |

| indice | binaire | codé |
|---|---|---|
| 33 | 100001 | h |
| 34 | 100010 | i |
| 35 | 100011 | j |
| 36 | 100100 | k |
| 37 | 100101 | l |
| 38 | 100110 | m |
| 39 | 100111 | n |
| 40 | 101000 | o |
| 41 | 101001 | p |
| 42 | 101010 | q |
| 43 | 101011 | r |
| 44 | 101100 | s |
| 45 | 101101 | t |
| 46 | 101110 | u |
| 47 | 101111 | v |
| 48 | 110000 | w |
| 49 | 110001 | x |
| 50 | 110010 | y |
| 51 | 110011 | z |
| 52 | 110100 | 0 |
| 53 | 110101 | 1 |
| 54 | 110110 | 2 |
| 55 | 110111 | 3 |
| 56 | 111000 | 4 |
| 57 | 111001 | 5 |
| 58 | 111010 | 6 |
| 59 | 111011 | 7 |
| 60 | 111100 | 8 |
| 61 | 111101 | 9 |
| 62 | 111110 | + |
| 63 | 111111 | / |
| | | |
| PADDING | | = |

*Tableau 1. Tableau de codage en Base64*

çant également par un ensemble d'en-têtes HTTP, comme indiqué dans le **listage 2**. La première ligne indique la version du protocole et un code de réponse. Comme nous avons demandé du HTTP/1.1, le serveur répondra également dans cette version du protocole. Le code de réponse que nous obtenons est `200`, signalant que notre requête peut être traitée. Cette ligne est suivie d'autres en-têtes qui donnent des informations supplémentaires.

Si nous sommes uniquement intéressés par le contenu demandé, nous pouvons sauter l'en-tête. L'en-tête est séparé du contenu demandé par une ligne vide (contenu ne figurant pas dans le listage 2). Pour votre information, jusqu'à 8 Ko de données, voire plus, peuvent se trouver juste dans l'en-tête lui-même. On peut définir le corps et sa longueur avec `Transfer-Encoding: chunked` pour les contenus de longueur inconnue, comme les pages web créées dynamiquement, ou `Content-Length:<length>` pour les contenus de taille connue à l'avance, comme les images.

Il existe d'autres méthodes HTTP comme le POST, où les données sont transférées du client au serveur, pour soumettre des informations avec un formulaire web. La méthode PUT peut également être utilisée pour transférer des données vers un serveur web. Pour plus d'informations sur les autres méthodes définies, consultez le « Mozilla Developer Network » [2].

## De HTTP à HTTPS

Lors du développement de http, la sécurité intrinsèque du protocole de communication a été oubliée. Les informations sont transmises en texte clair et toute personne avec un peu de connaissance sur les réseaux peut les extraire ou les modifier. HTTPS, où le « S » signifie sécurisé, a rajouté une couche supplémentaire à la pile de communication. Le serveur et le client continuent à se parler en utilisant le protocole HTTP, mais au lieu d'avoir une communication en texte clair sur une connexion TCP/IP, tout le trafic entre les deux parties est d'abord traité par la couche *Transport Layer Security* (TLS) qui gère le chiffrement et l'échange de ses clés.

Il faudra aussi plus de puissance de calcul, de flash et de RAM si nous utilisons HTTPS, car il faut maintenant effectuer le chiffrement et le déchiffrement, et inclure dans notre logiciel les routines supplémentaires pour TLS. La puissance de calcul d'un ordinateur moderne suffira. Sur les

systèmes embarqués les plus simples, on observera des ralentissements dans la communication si le chiffrement et le déchiffrement, ainsi que la gestion et l'échange des clés, sont effectués uniquement dans le logiciel. C'est pourquoi, sur les UC modernes, vous trouverez des accélérateurs de chiffrement qui la déchargent de ce travail. Pour l'ESP32, on dispose d'une accélération matérielle pour effectuer rapidement ce processus.

## Requête et réponse

Le service Toggl propose une API ouverte et documentée [3] pour l'échange de données. L'API fonctionne en soumettant des requêtes POST, PUT et GET à une URL, avec un en-tête et une charge utile définis. Toutes les actions nécessitent une clé d'authentification qui peut être obtenue sur la page web de Toggl. Grâce aux informations fournies, la construction d'une requête peut commencer. Nous utilisons la classe `WiFiClientSecure` en conjonction avec l'objet `HTTPClient`. C'est une méthode éprouvée pour communiquer avec le serveur et éviter de réinventer la roue lorsqu'il s'agit de traiter des données HTTP. Nous constituons à la main le jeton d'autorisation, une combinaison codée en Base64 du nom d'utilisateur et du mot de passe. Pour le codage en Base64, il faut une clé API et la chaîne `:api_token`. On peut obtenir la clé API auprès du service Toggl. Comme elle ne sera pas envoyée en texte clair, il peut y avoir des caractères qui interfèrent avec les caractères réservés pour l'en-tête lui-même. Pour inclure tout type de données dans ces requêtes, la solution est un encodage en Base64. D'habitude, les valeurs d'un octet sont comprises entre 0 et 255. La Base64 n'utilisera que 64 valeurs mises en correspondance avec les lettres ASCII équivalentes `A-Z`, `a-z`, `0-9`, `+`, `/` et `=` pour le remplissage (**tableau 1**). Par exemple, si nous encodons Elektor comme une chaîne de caractères en Base64, nous obtiendrons RWxla3Rvcg== (**fig. 6**). Sachez que les pièces jointes des courriers électroniques sont encodées en Base64 pour des raisons similaires, et comme le montre l'exemple, la taille de la pièce jointe augmentera d'environ 37% par rapport à la taille originale.
Une fois les informations d'identification transmises au client HTTP pour autorisation, la requête peut maintenant être faite. L'exemple suivant montre comment mettre en œuvre une requête GET et traiter les informations renvoyées. L'extrait du

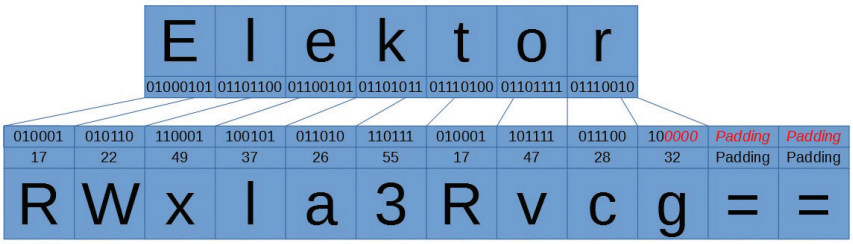



*Figure 6. Elektor codé en Base64.*

**listage 3** vous donnera quelques indications. La fonction `GET()` renverra le code de réponse du serveur pour notre requête, ou si nous avons un problème de communication, une valeur négative. On peut voir en [4] les différentes valeurs possibles du code de réponse. Le code le plus connu est 404, *Resource not found*. Ici, le code 200, *Response okay*, est attendu et vérifié. Si des données sont reçues du serveur, elles peuvent être récupérées avec `getString()` pour un traitement ultérieur.
Il existe un code similaire pour PUT et POST, avec quelques différences mineures, mais importantes, comme le montre le **listage 4**. La fonction `addHeader()` utilisée avant `PUT()` ajoutera deux en-têtes supplémentaires à la requête. Le premier est *Content-Type* qui indique au serveur web le type de données que nous lui enverrons, le second est *Content-length*, et c'est ici que HTTP devient soudainement un peu différent de ce que nous avons vu auparavant. D'habitude, nous transmettons après l'en-tête des données, comme du texte, qui commencent et se terminent par deux sauts de ligne. Cela fonctionne pour le texte, mais si nous envoyons maintenant une image

**Listage 3. Extrait de la requête HTTP**

```
httpclient.begin(client,link);
   httpclient.setReuse(true);
   httpclient.setAuthorization((const char*)b64.c_str());
   int httpCode = httpclient.GET();
    if (httpCode > 0) { //Check for the returning code
        if(httpCode!=200){
            Serial.println("Response Error");
            String payload = httpclient.getString();
            Serial.println(httpCode);
            Serial.println(payload);
            *error = 1;
        } else{
            *Result = httpclient.getString();
        }
    } else {
        Serial.print("Request Error");
        Serial.println(httpclient.errorToString(httpCode));
    }
    httpclient.end();
```

**Listage 4. En-tête HTTP additionnel**

```
httpclient.setAuthorization((const◦char*)b64.c_str());
   httpclient.addHeader("Content-Type","application/json");
   httpclient.addHeader("Content-length","0");
   int httpCode = httpclient.PUT((uint8_t*)(NULL),0);
      if (httpCode > 0) { //Check for the returning code

      . . .
```

**Listage 5. Réutilisation de la connexion**

```
httpclient.begin(client,link);
httpclient.setReuse(true);
httpclient.setAuthorization((const char*)b64.c_str());
httpclient.addHeader("Content-Type","application/json");
int httpCode = httpclient.POST(payload);
   if (httpCode > 0) {

. . .
```

**Listage 6. Exemple de chaîne JSON**

```
{ "name": "Test", "value": 1351824120,"array": [ 123.45, 321.89 ] }
```

**Listage 7. Sérialisation JSON**

```
DynamicJsonDocument doc(capacity);
JsonObject time_entry = doc.createNestedObject("time_entry");
   if(description.length()>0){
       time_entry["description"] = description.c_str();
   }
time_entry["created_with"] = "ESP32";
serializeJson(doc, payload);
```

**Listage 8. Désérialisation JSON**

```
DeserializationError er = deserializeJson(doc, Result.c_str());
 if (er) {
      //something went wrong
         } else {
         if(false == doc["data"].isNull() ){
             JsonObject data = doc["data"];
             uint32_t data_id = data["id"];
             const char* data_start = data["start"];
            if(false == data["description"].isNull()){
                 const char* data_description = data["description"];
                  Element->description = String(data_description);
             } else {
                 Element->description = "";
             }
```

**Listage 9. Chaîne JSON Toggl**

```
{
"data":
   {
           "id":436694100,
           "pid":123,
           "wid":777,
           "billable":false,
           "start":"2013-03-05T07:58:58.000Z",
           "duration":1200,
           "description":"Brew some coffee",
           "tags":["billed"]
   }
}
```

JPEG au serveur, elle peut être constituée de n'importe quelle combinaison d'octets arbitraires. C'est là que *Content-length* entre en jeu. Les en-têtes sont disposés et séparés de nos données par deux sauts de ligne. Avec *Content-length*, vous pouvez envoyer n'importe quel type de caractère ou de données binaires. Le serveur web a été informé du nombre d'octets qu'il doit s'attendre à recevoir avec *Content-length*. Pour POST, il suffit de fournir le type de contenu qui sera transféré. Aucune charge utile réelle ne sera transmise. Si nous devions fournir une charge utile ou un contenu supérieur à 0 octet, le client HTTP ajouterait un en-tête *Content-Length* de son propre chef. Dans ce cas particulier, avec une longueur fixée à zéro, puisque nous n'envoyons pas de contenu, on n'ajoute pas du tout l'en-tête *Content-Length*. Comme le serveur se plaindra s'il manque la longueur du contenu, nous devons l'ajouter nous-mêmes avec la fonction `addHeader()`, comme indiqué dans le listage 4. On peut voir dans le **listage 5** que `setReuse()` est réglé sur *true*. Après cette requête, que ce soit POST, PUT ou GET, la connexion au serveur sera normalement fermée par le client. Si `setReuse()` est réglé sur *true*, cette connexion restera ouverte et sera réutilisée pour la requête suivante. On gagne du temps, car on évite de rétablir la connexion s'il faut effectuer une autre requête. En outre, au moment de la rédaction de ce document, cela contourne un bogue quelque part dans la pile réseau, où la mémoire n'est pas correctement libérée. Tôt ou tard, cela peut entraîner des échecs de connexion, car l'ESP32 n'a plus de mémoire disponible.

## Échange de données avec JSON

JSON signifie *JavaScript Object Notation* et décrit un format d'échange de petites quantités de données entre deux systèmes. Toutes les données sont regroupées dans une chaîne de caractères qui peut ressembler à celle du **listage 6**. Cette chaîne doit être constituée et analysée. La plupart des environnements qui traitent et échangent des données pour des applications web utilisent JSON et disposent de fonctions intégrées ou de bibliothèques supplémentaires toutes faites pour la constitution et l'analyse de données au format JSON. C'est aussi vrai pour les petits systèmes comme un Arduino ou un ESP32.
`ArduinoJSON` est une bibliothèque compatible avec le canevas Arduino et l'ESP32.

Elle peut analyser et constituer des chaînes JSON à échanger avec d'autres systèmes. Comme Toggl utilise JSON pour échanger des données, cette bibliothèque est adaptée à nos besoins. L'API de Toggl nous permet de déterminer les informations qui devraient se trouver à l'intérieur des chaînes JSON. Cette documentation décrit également comment organiser les données pour envoyer des informations. Pour une nouvelle saisie de temps, nous devons créer une chaîne JSON qui contient un objet `time_entry`. À l'intérieur de cet objet `time_entry`, nous avons besoin d'une chaîne `description` et d'une chaîne optionnelle `created_with` pour la nouvelle entrée.

Le code du **listage 7** va créer un nouveau document `DynamicJsonDocument` avec une capacité définie dans la pile de l'ESP32. Cela signifie que la mémoire est allouée avec `malloc()` puis libérée avec `free()` à partir de la mémoire non utilisée statiquement par l'ESP32. Par opposition à `StaticJsonDocument` qui sera alloué de manière statique dans la RAM ou, s'il est utilisé localement dans une fonction, sera placé dans la pile. `StaticJsonDocument` est plus rapide, car localement lié à la pile, mais sa taille est inférieure à celle qu'on peut utiliser avec `DynamicJsonDocument`. La fonction `serializeJson()` créera la chaîne de caractères souhaitée qui pourra être envoyée.

`deserializeJson()` permet de décoder les données JSON et signalera toute erreur rencontrée lors de l'analyse. Cette partie du code est présentée dans le **listage 8**. On y voit comment accéder aux éléments dans la chaîne JSON. La chaîne de caractères renvoyée par Toggl ressemble à celle du **listage 9**. Le code vérifiera après « désérialisation » si les données (`data`) de l'objet existent, c'est-à-dire si nous avons un résultat valide. Ensuite, le code accède aux éléments à l'intérieur de `data` et ne récupère que les valeurs nécessaires plus tard pour l'affichage et le traitement, à savoir `id`, `start` et `description`. Ce dernier champ est spécial, il n'est intégré dans `data` que s'il contient une valeur de la base de données Toggl. Cela nécessite de vérifier avec `data[«description»].isNull()` que l'entrée existe bien avant de tenter d'y accéder ; sinon, la bibliothèque provoquera une exception.

L'échange de données complet est encapsulé dans la bibliothèque `toggleClient`. Ici, elle prend en charge le strict nécessaire : afficher l'entrée courante, démarrer une nouvelle entrée et terminer l'entrée courante. Le code n'est pas parfait et chacun est invité à contribuer aux améliorations et aux corrections d'erreurs, car il s'agit plus d'un point de départ que d'une bibliothèque finalisée. Outre la collecte et la soumission de données, il faut aussi les afficher, et c'est l'objet de ce qui suit.

## Dessiner des pixels

Pour le dessin, vous vous souvenez peut-être d'un projet d'Elektor plus ancien, « GUI tactiles » [5], qui utilisait la bibliothèque `TFT_eSPI`, compatible avec diverses combinaisons d'écrans et d'ESP32. Comme cette bibliothèque prend en charge l'écran du kit M5Stack, on pourra réutiliser du code provenant d'autres projets. Dans le dossier de la bibliothèque, on doit modifier le fichier *User_Setup_Select.h* pour utiliser la configuration du kit M5Stack. Dans le fichier TogglButtonM5.ino, la ligne `TFT_eSPI tft = TFT_eSPI() ;` créera un nouvel objet TFT_eSPI qui servira ultérieurement pour le dessin.

Pour initialiser l'affichage, nous n'avons besoin que de quatre lignes de code à l'intérieur de la fonction `setup()` (**listage 10**). On doit régler `inverted display` sur *true* pour transmettre les couleurs dans le bon ordre pour l'écran du kit M5Stack. Avec `setRotation(1)`, le bas de l'écran sera là où se trouvent les boutons du kit M5Stack. Après cette ligne de code, on peut utiliser toutes les manipulations de pixels possibles qu'offre la bibliothèque.

Une petite astuce pour accélérer le tracé, en particulier pour un afficheur basé sur l'ILI9341. Ces afficheurs sont optimisés pour traiter des images en plein écran ou de plus gros volumes de données. L'accès pixel par pixel ralentira le dessin d'au moins sept fois, voire plus. Ceci est dû au mode d'émission des commandes et d'accès aux pixels individuels. Déjà optimisée pour le dessin avec l'ESP32, la bibliothèque `TFT_eSPI` optimisera, dans la mesure du possible, les temps d'accès. Outre le placement de pixels ou le remplissage de l'écran avec une seule couleur, la bibliothèque fournira aussi d'autres fonctions de dessin comme les lignes, les rectangles et l'utilisation des fonctions `print` pour imprimer des chaînes de caractères sur l'écran. Si on vous demande pourquoi utiliser une bibliothèque pour ça au lieu d'écrire un pilote, vous pouvez répondre : ne jamais réinventer la roue.

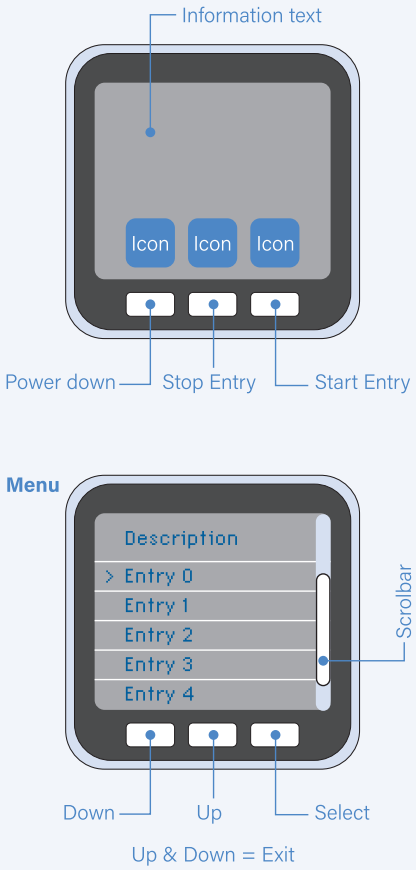



*Figure 7. Schéma conceptuel de la GUI.*

**Listage 10. Initialisation de l'écran LCD**

```
tft.init();
tft.invertDisplay( true );
tft.setRotation(1);
tft.fillScreen(TFT_WHITE);
```

## Interface graphique (GUI) et menu

Le kit M5Stack dispose d'un écran et de trois boutons. Les solutions de suivi des temps ne fournissent généralement qu'un seul bouton pour démarrer et terminer une entrée. Comme le kit M5Stack peut aussi fonctionner sur batterie, il peut être très pratique d'avoir un bouton d'arrêt. La **figure 7** montre le principe de l'interface graphique et les positions approximatives des quelques éléments à afficher.

Comme il est assez simple de dessiner du texte, la question se pose : qu'allons-nous afficher ? Les éléments essentiels à afficher sont la description, si nous en avons une, et l'heure de début. Pour les icônes, nous utiliserons celles de la bibliothèque Open Icon [6] et on peut voir l'écran principal qui


*Figure 8. Menu principal de la GUI.*

en résulte sur la **figure 8**. Il nous permet de démarrer une nouvelle entrée, de l'arrêter et de mettre l'appareil hors tension. L'appareil reste ainsi facile à utiliser.

Nous pouvons démarrer une nouvelle entrée Toggl, mais il n'y a pas de description de l'activité. On n'a pas besoin de plus, mais ce serait bien de pouvoir au moins choisir parmi quelques activités prédéfinies. Même s'il est possible de saisir la description des activités avec seulement trois boutons, ce n'est pas très convivial. Une refonte totale basée sur le projet d'horloge à trois affichages [7] a permis d'introduire un système de menu simple à utiliser avec la bibliothèque `TFT_eSPI`. La résolution de l'écran et l'orientation du système de menu correspondent également à notre configuration du bouton Toggl. On peut récupérer le code de ce projet pour présenter un menu comme celui de la **figure 9**.

Le menu n'est pas aussi universel et flexible que celui d'un canevas pour GUI. Il est léger, fonctionne avec `TFT_eSPI` et a été conçu pour être utilisé avec seulement quelques boutons. Si j'avais eu plus de temps, j'aurais utilisé la bibliothèque LVGL (*Light and Versatile Graphics Library*) [8], car elle fonctionne également pour les écrans non tactiles. Voyons comment fonctionne notre système de menu fait maison et comment les graphiques sont dessinés. La logique du menu réside dans une fonction `RenderMenu()`. Selon l'état, elle dessine le menu et retourne *true*. Si


*Figure 9. Sélection de l'activité dans la GUI.*

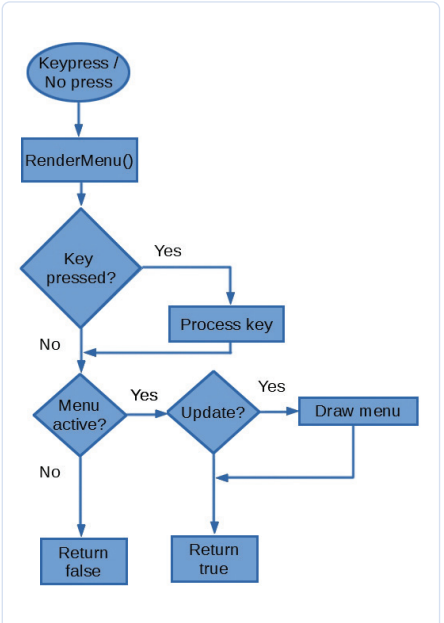



*Figure 10. Organigramme du système de menu.*

**Listage 11. ShowMenuSettingsList**

```
void Menus::ShowMenuSettingsList(uint8_t selected_idx, bool refresh){

  /* Draw Headline */
  if(true == refresh){
    _lcd->fillRect(0,0,320,40,_lcd->color565( 45,47,50 ) );
    _lcd->setTextColor(_lcd->color565( 112,116,122 ),TFT_BLACK);
    _lcd->setFreeFont(FSB18);
    _lcd->setCursor(160- ( _lcd->textWidth("Description") / 2) ,30);
    _lcd->print("Description");
  }
  /* Headline done */

  /* Draw scrollbar */
  _lcd->fillRect(303,43,15,197,TFT_WHITE);
  _lcd->drawRect(300,40,20,200,_lcd->color565( 112,116,122 ));
  _lcd->drawRect(301,41,18,198,_lcd->color565( 112,116,122 ));
  _lcd->drawRect(302,42,16,196,_lcd->color565( 112,116,122 ));
  _lcd->fillRect(305,45 +(196/GetUsedEntryCount())*selected_idx
   ,10,(196/GetUsedEntryCount())-2 , TFT_DARKGREY);
  /* Scrollbar done */

  /* Menu entries */
  uint8_t startidx=0;
  uint8_t endindex=0;
  if(selected_idx>4){ //We can display 5 Elements
    startidx = selected_idx -4 ;
  }
  if((startidx+5)>GetUsedEntryCount()){ //Limit last drawn item
      endindex=GetUsedEntryCount();
  }else{
      endindex=startidx+5;
  }
  /*Draw up to five elements*/
  for(uint8_t i=startidx;i<endindex;i++){
          DrawSettingsMenuEntry( (40*(i-startidx))
   ,DescriptionArray[i].c_str() ,( i==selected_idx ) );
  }
  /* Menu entries drawn */
}
```

aucun menu n'est dessiné, la fonction retournera false, ce qui indique que le contenu de l'affichage n'a pas été modifié. Si *true* est renvoyé, il faut éviter de faire un autre tracé par-dessus. Voir l'organigramme simplifié de la **figure 10**.

La logique du menu sera assurée par `RenderMenu()`, mais la partie dessin sera faite avec `ShowMenuSettingsList()`. Cette fonction dessinera l'entête du menu et jusqu'à cinq items de menu. La logique du menu garantit que l'élément sélectionné reste dans une plage valide. Ainsi, la routine de dessin calcule la position de départ et affiche cinq items sous forme de liste à l'écran. En outre, une barre de défilement, comme dans de nombreux autres systèmes Windows, sera calculée et dessinée avec un petit indicateur de la position dans la liste.

Le **listage 11** montre le code de `ShowMenuSettingsList()`, structuré en trois parties. La première étant le titre. Pour gagner du temps de traitement, elle n'est redessinée que lorsque nécessaire.

Le centrage du texte du titre se fait manuellement en calculant la moitié de la longueur du texte en pixels et en soustrayant la moitié de la largeur de l'écran. Cela fonctionne si le texte fait moins de 160 pixels et peut produire des résultats imprévisibles s'il est plus long. La seconde partie est la barre de défilement, dessinée à droite du menu. Elle se compose de quelques rectangles. Sa longueur est calculée à partir du nombre d'éléments utilisés dans le menu. De même, cela fonctionne s'il y a moins de 196 items dans la liste, sinon les résultats peuvent être imprévisibles.

Dans la troisième partie, on calcule les indices de début et de fin des items de liste à dessiner. Comme nous ne pouvons dessiner que cinq éléments, nous devons déplacer cette plage en conséquence par rapport à l'indice sélectionné par l'utilisateur. Les trois parties se chargent de la liste affichée. Comme tous les items de la liste se ressemblent, chacun est dessiné avec sa propre fonction qui n'a besoin que d'un décalage pour dessiner à la bonne hauteur et d'un drapeau si l'item est sélectionné. On obtient un système de menu sommaire qui remplit sa fonction et qui illustre quelques notions de base des menus.

## Configuration à partir du Web

Le logiciel n'a pas été écrit à partir de zéro. J'ai réutilisé des parties de projets antérieurs. C'est aussi vrai pour la configuration à partir du web. Si vous appuyez sur les deux boutons de gauche pendant le démarrage, le logiciel démarre un point d'accès auquel vous pouvez vous connecter avec un ordinateur ou un appareil portable compatible wifi. Sinon, le logiciel essaiera de se connecter au réseau wifi qui a été configuré, le cas échéant. Il est facile de démarrer un serveur web sur l'ESP32 : après le démarrage du wifi le logiciel nous présente une interface web, qui nous permet de configurer le réseau wifi (**fig. 11**) et d'activer une clé d'API Toggl pour l'authentification de l'utilisateur (**fig. 12**).

## Prolongements et enseignements

Bien que le bouton Toggl fonctionne et permette un efficace suivi des temps, il est encore possible de l'améliorer. Dans cet article, je n'ai abordé que quelques-uns des nombreux aspects des dispositifs IdO modernes, en particulier les interac-

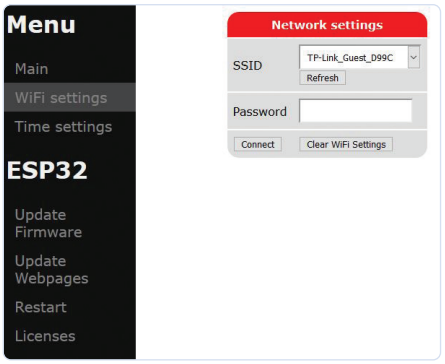



*Figure 11. Page de configuration du wifi.*

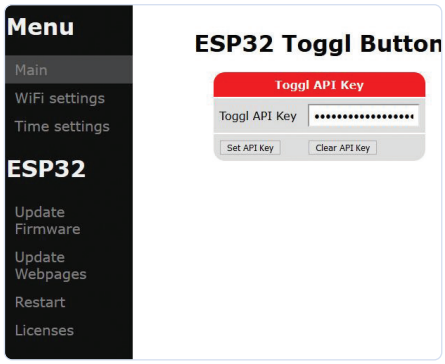



*Figure 12. Clé d'API de Toggl.*

tions avec tout type de serveur ou de navigateur web. Des termes tels que wifi, serveur HTTP, client HTTP, TLS, JSON, JavaScript, HTML, GUI et C/C++ peuvent sembler être des mots à la mode, mais ils sont en réalité les ingrédients essentiels de ce modeste projet. Le fait de s'occuper de HTTP et HTTPS, JSON d'un côté et de la conception de l'interface graphique (GUI) de l'autre, tout en fournissant une interface de configuration à partir du web, était peu courant aux premiers temps de l'IdO. Mais aujourd'hui, c'est le strict minimum pour un appareil connecté. Ce mélange de différents domaines – dont le développement web de base, JSON, HTML, C/C++ et JavaScript – pourrait remplir un livre entier. Et si un tel livre est publié, Elektor en rendra compte.

En écrivant cet article, j'ai trouvé quelques idées pour rendre le bouton encore plus convivial et ajouter des fonctions qui amélioreront la comptabilité. La première amélioration serait de traiter la GUI avec LVGL. En parlant de la GUI, passons rapidement aux enseignements. Si vous voulez ajouter une GUI à votre projet, cherchez quelque chose qui a fait ses preuves et qui est disponible. Il y a peut-être même des solutions avec une licence amiable pour votre projet. Développer votre propre système de menus pour le plaisir d'apprendre peut être génial, mais réinventer la roue ne l'est pas.

Pour l'ESP32, si vous utilisez le canevas et les bibliothèques Arduino, pensez à consulter son dépôt GitHub et les anomalies signalées. C'est une bonne ressource pour éviter des problèmes ou contourner des limitations connues. On trouve le code de ce bouton Toggl sur la page GitHub d'Elektor [9] ce qui est un moyen pratique de le cloner et de l'examiner, car son évolution y est conservée. Si vous trouvez des bogues ou avez des suggestions, vous pouvez utiliser la fonction issues de GitHub. Comme ce projet touche de nombreux sujets, j'attends vos retours pour savoir s'il faudrait revenir plus en détail sur certains d'entre eux dans un prochain article.

(200631-04)

## PRODUITS

- **Kit de développement *ESP32 Basic Core* de M5Stack**
  www.elektor.fr/m5stack-esp32-basic-core-development-kit
- **Kit de développement ESP32 Arduino MPU9250 de M5Stack**
  www.elektor.com/m5stack-esp32-arduino-mpu9250-development-kit
- **W. Gay, *FreeRTOS pour ESP32-Arduino*, Elektor 2020 (livre en anglais)**
  www.elektor.fr/freertos-for-esp32-arduino

**Contributeurs**
Développement, texte et diagrammes : **Mathias Claußen**
Illustrations : **Patrick Wielders**
Mise en page : **Giel Dols**
Traduction : **Denis Lafourcade**

**Des questions, des commentaires ?**
Si vous avez des questions ou des commentaires sur son article, envoyez un courriel à l'auteur (mathias.claussen@elektor.com) ou contactez Elektor (redaction@elektor.fr).

## LIENS

[1] **M5Core Basic** : https://docs.m5stack.com/#/en/core/basic
[2] **HTTP | MDN** : https://developer.mozilla.org/en-US/docs/Web/HTTP
[3] **Documentation de l'API de Toggl.com** : https://github.com/toggl/toggl_api_docs
[4] **Codes de retour d'état HTTP | MDN** : https://developer.mozilla.org/en-US/docs/Web/HTTP/Status
[5] **« GUI tactiles – pour ESP32, RPi & Co. », Elektor** : www.elektormagazine.fr/news/guitactilespouresp32rpico
[6] **Bibliothèque Open Icon** : https://sourceforge.net/projects/openiconlibrary/
[7] **Horloge à 3 affichages avec écran TFT** : https://bit.ly/37KBhJC
[8] **Bibliothèque Light and Versatile Graphics** : https://lvgl.io/
[9] **Page GitHub du projet** : https://github.com/ElektorLabs/200631_ESP32_Toggl_Button

# Faites connaissance avec la **carte Raspberry Pi Pico** à **RP2040**

**Mathias Claußen** (Elektor) et **Luc Lemmens** (Elektor)

https://youtu.be/ijn-QDAgZss

Regardez « Raspberry Pi Pico Review » sur Elektor.TV!

Voilà qui s'appelle rebattre les cartes : en lançant la carte Pico, la fondation Raspberry Pi vient marcher sur les plates-bandes des Arduino, ESP32 et autres STM32 qui constituent *de facto* la référence en matière de plateformes à microcontrôleur. Écosystème logiciel, fonctions, prix, la petite nouvelle a tout pour concurrencer les anciennes. « Il est des nôtres, il a son microcontrôleur comme les autres », ont sans doute tonné *Les compagnons du comptoir de l'embarqué* à l'annonce de sa sortie.

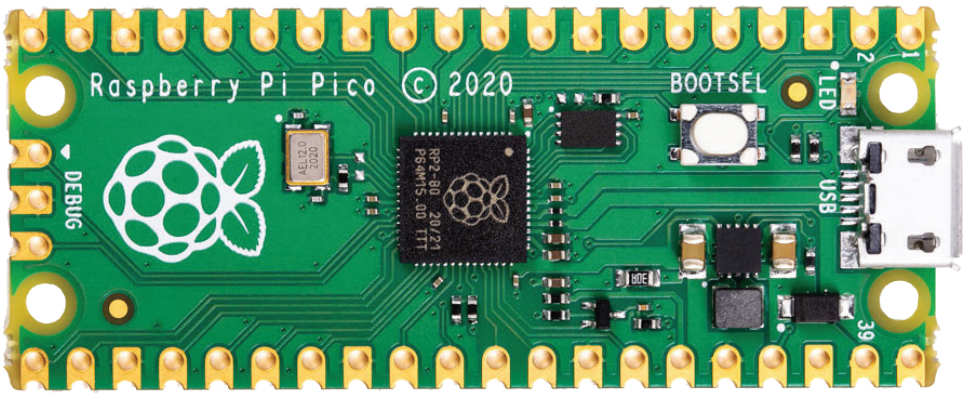



*Figure 1. La carte Raspberry Pi Pico.*

## Encore une carte Raspberry Pi ?

Oui, mais celle-ci est d'une autre sorte (**fig. 1**) : la *Raspberry Pi Pico* se veut le complément et le compagnon de la famille Raspberry Pi. D'ailleurs, à y regarder de près, la Pico ne ressemble en rien aux cartes Raspberry Pi que nous connaissons. Elle embarque un processeur ARM à double cœur assisté de juste ce qu'il faut d'électronique pour orchestrer l'ensemble. C'est une sorte de RPi Zero modifié, alors ? Non, car le processeur en question n'est pas l'habituelle puce de Broadcom, mais le *RP2040*, un microcontrôleur conçu par la fondation RPi elle-même.

Ce RP2040 est plus précisément un microcontrôleur à double cœur ARM Cortex-M0+. Disparu, donc, le traditionnel système sur puce pour Linux : la Raspberry Pi Pico s'invite au royaume des plateformes à microcontrôleur dont les étendards les plus notoires s'appellent Arduino, Blue Pill et ESP32, sans oublier les cartes à processeurs RISC-V de popularité croissante (comme la GD32VF103). Avec un prix tournant autour de 5 €, soit moitié moins que la carte ESP32 Pico Kit, la Raspberry Pi Pico peut d'ores et déjà rivaliser avec la Blue Pill et, disons, un clone de l'Arduino Nano.

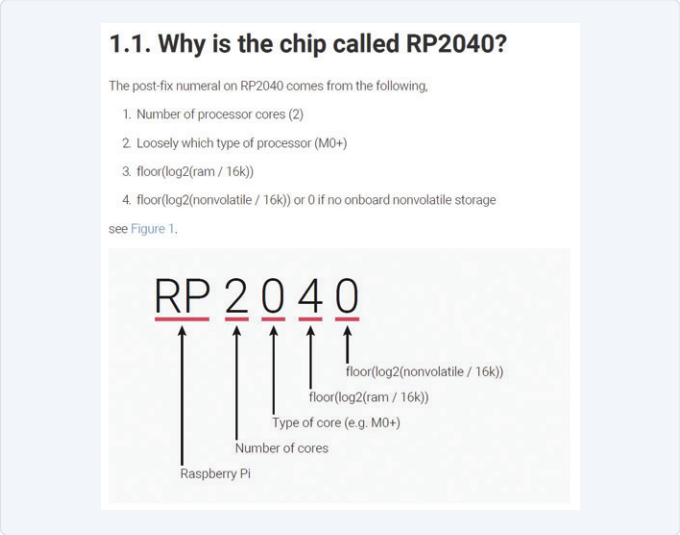



*Figure 2. Schéma de nommage [9].*

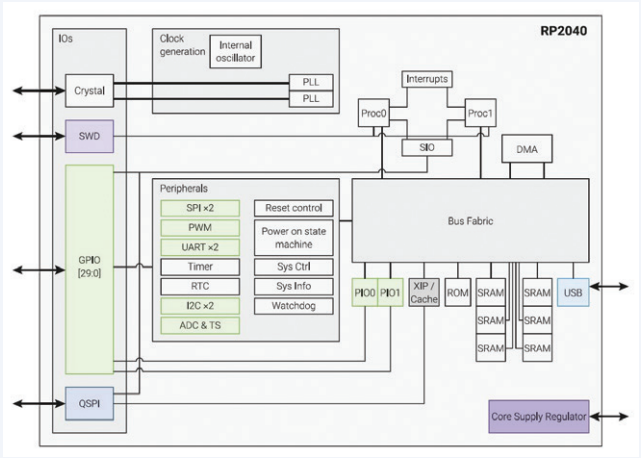



*Figure 3. Diagramme fonctionnel du RP2040 [9].*

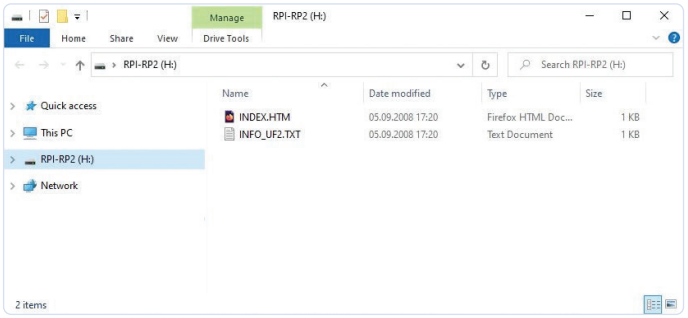



*Figure 4. La carte Pico reconnue comme périphérique de stockage.*

**Tableau 1. Caractéristiques de la Raspberry Pi Pico**

| | |
|---|---|
| Processeur | Cortex M0+ |
| Cœurs | 2 |
| Fréquence max. | 133 MHz |
| SRAM | 264 Ko sur 6 bancs |
| Flash interne | 0 Ko |
| Flash externe | 2 Mo flash QSPI (jusqu'à 16 Mo pris en charge) |
| GPIO | 26 broches (dont 4 CA/N) |
| USB | 1.1 hôte / esclave |
| CA/N | 12 bits @ 500 kéch./s |
| Canaux CA/N | 5 (dont le capteur de T°) |
| SPI | 2 |
| UART | 2 |
| I²C | 2 |
| PWM | 16 canaux |
| Timer | 1× 64 bits |
| RTC | oui |
| Fonctions uniques | automates finis programmables à E/S, ROM de démarrage reconnue comme mémoire USB. |

Avant de passer en revue les caractéristiques de la carte Pico et du RP2040, signalons-en les deux courtes présentations signées Clemens Valens sur la chaîne YouTube d'Elektor [1] et sur le site du magazine Elektor [2].

## Tour d'horizon de la carte RPi Pico

Les deux cœurs ARM Cortex-M0+ du microcontrôleur (µC) RP2040 sont cadencés à 133 MHz. Trouver un microcontrôleur à double cœur Cortex-M n'a rien d'extraordinaire en soi, mais habituellement il ne s'agit pas de Cortex-M0+ mais de cœurs plus puissants, comme le Cortex-M4, le Cortex-M7 ou le récent Cortex-M33.

Le **tableau 1** résume les caractéristiques du RP2040. À propos, d'où vient ce nom ? La réponse se trouve sur la **figure 2** (où *floor()* est la fonction « partie entière »). Cette méthode de nommage laisse suggérer que des variantes suivront, mais la fondation Raspberry Pi, que nous avons interrogée sur ce point, ne nous a pas encore répondu.

L'agencement des périphériques et blocs de fonctions du RP2040 est schématisé sur la **figure 3**. L'absence de mémoire flash interne y est notable. Les concepteurs de la Pico ont en effet choisi d'implanter cette mémoire à l'extérieur du µC, plus précisément dans le petit W25Q16JUXIQ, une mémoire flash NOR de 2 Mo. Le RP2040 prenant en charge 16 Mo de mémoire, il suffit de remplacer cette puce par une autre pour disposer d'un espace de stockage plus grand. Le port USB de la carte peut être configuré en tant qu'hôte ou esclave en mode USB 1.1 (12 Mo/s). Autrement dit on peut y relier des périphériques USB ou l'utiliser comme périphérique USB. La ROM de démarrage permet de charger du code depuis un ordinateur. Nul besoin ici d'outils de programmation spéciaux, la mémoire est reconnue par le PC comme mémoire externe. Le RP2040 peut donc être programmé par copie de fichiers dans le dossier associé au nom de volume de la carte (**fig. 4**). Simple et pratique.

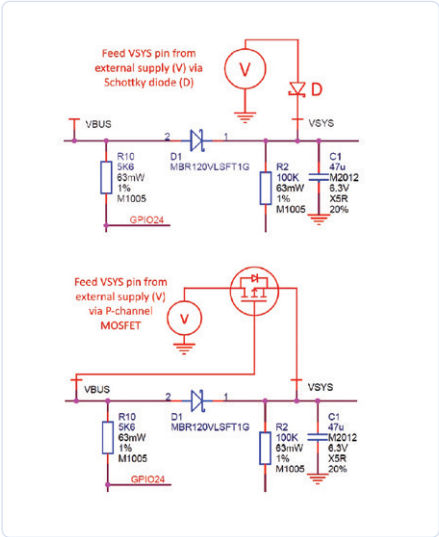



*Figure 5. Deux façons d'alimenter la carte [3].*

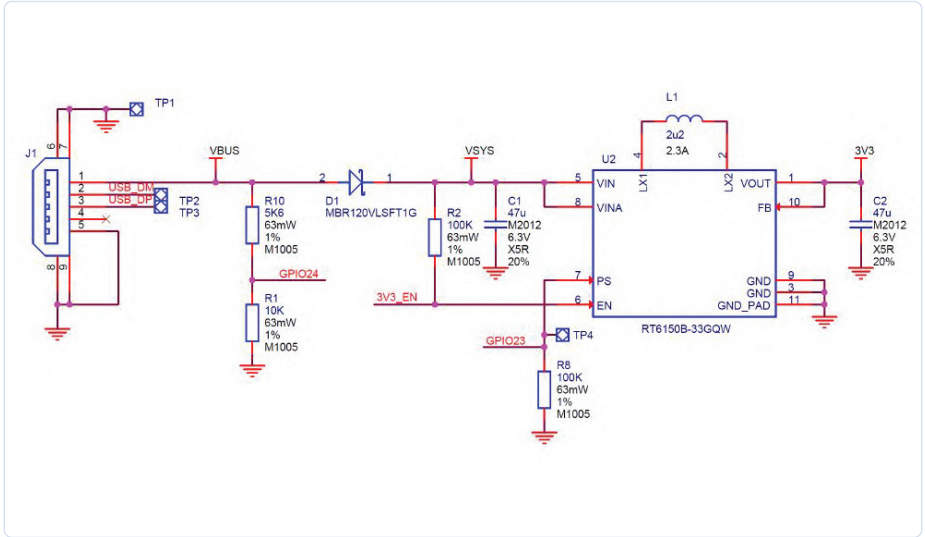



*Figure 6. Schéma du convertisseur Buck-Boost embarqué [3].*

Pour ce qui est de l'alimentation, pas de tourment non plus en vue. La carte peut être alimentée en 5 V par micro-USB, mais accepte aussi des tensions de 1,8 V à 5,5 V délivrées par un convertisseur CC/CC Buck-Boost relié à sa broche VSYS. On peut l'alimenter avec p. ex. une batterie au lithium ou un jeu d'accus NiMH. La **figure 5** (extraite de la fiche technique) illustre le câblage de deux sources d'alimentation. Le convertisseur CC/CC est une puce RT6150 qui ne nécessite que quelques composants externes (**fig. 6**, fiche technique en [4]).

Parmi les 40 broches réparties en vis-à-vis le long de la carte figurent 26 broches GPIO, dont trois entrées CA/N, le reste étant pour l'alimentation et la masse. Le nom des broches est indiqué côté cuivre (**fig. 7**). La tension maximale d'E/S est de 3,3 V pour l'ensemble du port GPIO. Les trois broches placées sur un des petits côtés de la carte (SWCLK, SWIO et GND) forment le port *Serial Wire Debug*, c'est-à-dire le connecteur de programmation et de débogage du RP2040.

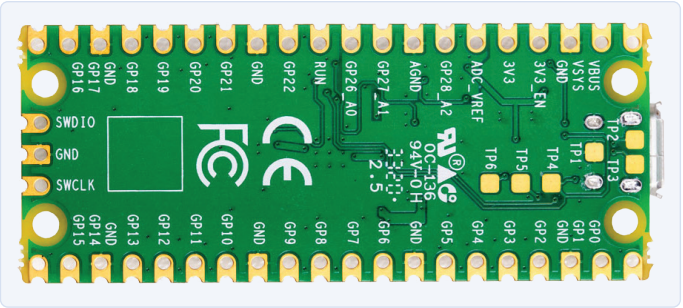



*Figure 7. Le nom des broches est indiqué sur la face cuivre.*

## Périphériques et blocs d'entrée/sortie programmables

La liste des périphériques offerts par la carte Pico est classique : deux UART, deux contrôleurs I²C, deux contrôleurs SPI, 16 canaux MLI (PWM), un CA/N à 12 bits et 500 kéch./s, un capteur de température intégré, une horloge à temps réel, un temporisateur/compteur, et les fonctions GPIO de base. Moins commune est l'interface USB hôte et esclave, encore plus rares sont les deux blocs à E/S programmables (PIO) comprenant chacun quatre automates finis.

Ces automates finis permettent d'implanter des interfaces UART, I²C et SPI supplémentaires, mais aussi des interfaces matérielles telles que VGA, DPI, SD-Card et bien d'autres. Leur usage est décrit dans la fiche technique et illustré par des programmes d'exemple. La **figure 8** montre un bloc PIO et ses quatre automates finis. Neuf instructions servent à les commander : JMP, WAIT, IN, OUT, PUSH,

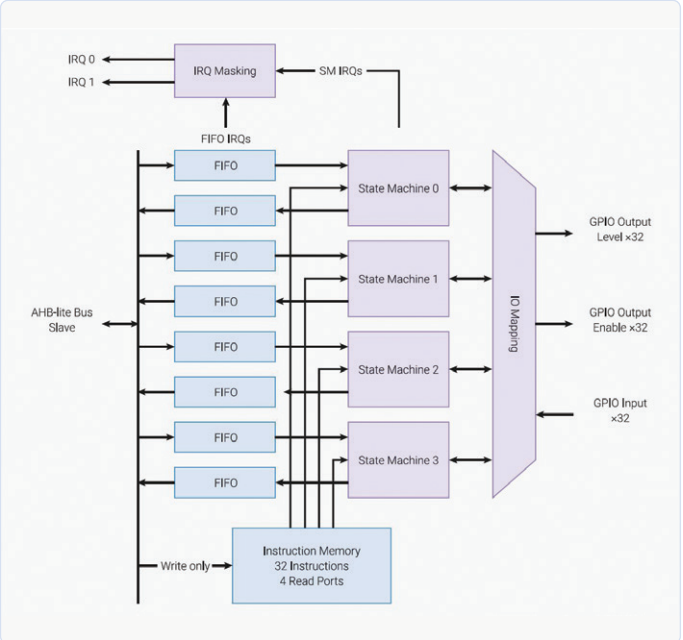



*Figure 8. Bloc PIO à automates finis programmables [9].*

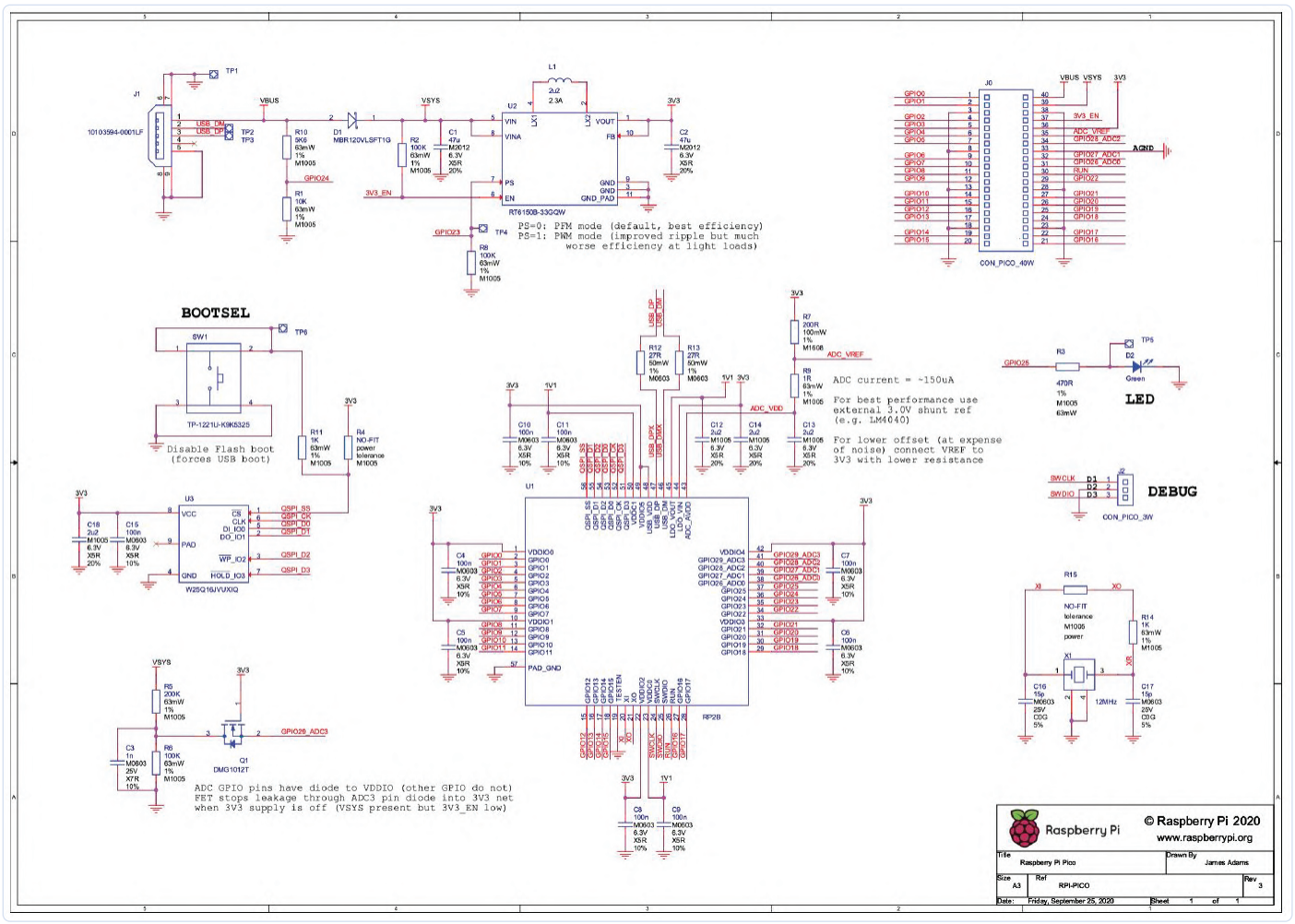



*Figure 9. Un bloc MLI (PWM) du RP2040 [9].*

PULL, MOV, IRQ, et SET. Ils peuvent fonctionner indépendamment des deux cœurs, et ainsi servir à la mise en œuvre d'interfaces absentes de la carte sans solliciter le processeur.

Les UART du RP2040 reposent sur l'UART PrimeCell d'ARM (PL011) présent sur d'autres cartes à RPi. Il autorise un débit de 961,6 kbauds. C'est encore ARM qui est à la manœuvre dans les deux interfaces SPI avec son port SSP PrimeCell (PL022), lui aussi exploité sur diverses cartes à RPi. En mode maître, le débit maximal vaut 1,843 Mbauds/s pour une fréquence d'horloge SPI d'au moins 3,686 MHz. L'horloge du contrôleur I²C peut opérer selon trois modes : *Standard* (100 kHz), *Fast* (400 kHz) et *Fast Plus* (1 MHz), avec un adressage sur 7 et 10 bits en maître ou esclave. Ces trois interfaces permettent de connecter la plupart des matériels en offrant à leur égard un bon équilibre entre fonctions et complexité.

Le CA/N du RP2040 est un simple SAR offrant 500 kéch./s à une résolution de 12 bits. Il possède quatre entrées et un capteur de température. Sur un AVR (p. ex.), un CA/N fonctionne avec une tension de référence interne de 2,6 V ou 1,1 V. Cette référence est externe sur le RP2040 et s'applique sur la broche ADC_AVDD ; le schéma de principe de la carte Pico (**fig. 9**) la montre reliée à 3,3 V et filtrée par un circuit RC (R7/R9/C13).

L'interface USB est également intéressante. Nous l'avons dit plus haut, elle peut être configurée en hôte ou esclave. Son mode *Full Speed* (USB 1.1, soit 12 Mo/s) permet l'utilisation de périphériques USB (p. ex. un clavier et une souris), mais aussi de relier la carte à un PC ou à un autre RPi. En mode hôte, le contrôleur prend en charge les concentrateurs USB, autrement dit il est possible de relier simultanément plusieurs périphériques à la Pico.

Le temporisateur/compteur (*timer*) de 64 bits du RP2040 repose sur une base de temps de 1 µs. Il permet de programmer jusqu'à quatre alarmes et des temporisations de l'ordre de la µs. Pour les

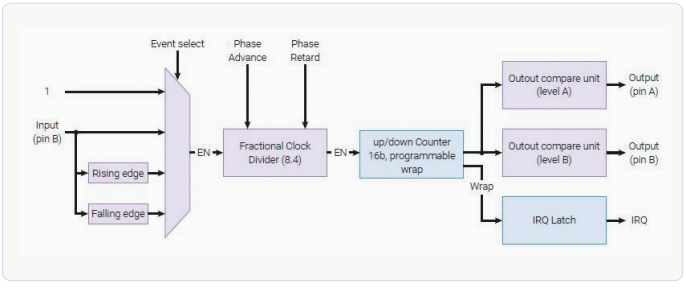



*Figure 10. Un bloc MLI (PWM) du RP2040 [9].*

interruptions ou évènements à déclencher périodiquement, vous pouvez utiliser une des huit unités MLI (PWM) à 16 bits et 2 canaux. Vous pouvez aussi mesurer des fréquences ou des rapports cycliques avec leurs broches GPIO afin de produire des interruptions et des requêtes DMA. Un diviseur d'horloge « fractionnaire » permet un réglage plus précis des fréquences.
L'unité DMA (à accès mémoire direct) permet de transférer des données dans le système sans le recours du processeur. Vous pouvez ainsi, sans processeur donc, effectuer des conversions A/N et placer les valeurs obtenues dans un tampon prédéfini de la mémoire, ou encore transférer des données de la mémoire à un UART. Copier et déplacer le contenu de plusieurs tampons d'un endroit à l'autre avec cette méthode peut même s'avérer plus rapide qu'avec le processeur. L'unité DMA se montre tout aussi pratique pour envoyer des données sur un écran externe. Elle peut de même être reliée aux automates finis programmables (PIO) pour échanger rapidement des données avec des périphériques externes.

## Schémas, manuels et fichiers de conception

La carte Pico est la première à exhiber un RP2040. Sur le schéma de la **figure 9**, on le voit entouré du convertisseur CC/CC, de la mémoire flash NOR QSPI, de l'interface USB, et d'un circuit chargé de mesurer la tension d'alimentation et de mettre le convertisseur CC/CC en mode basse consommation.
La fondation RPi fournit des fichiers KiCad et Fritzing à l'intention des concepteurs désireux de se familiariser avec le RP2040. Notons ici avec un certain intérêt que la conception de référence n'est pas la carte Pico, mais une carte à RP2040 à assembler et configurer soi-même. Alors que le brochage de la plupart des microcontrôleurs semble avoir été organisé par un amateur de Mikado, celui du RP2040 (**fig. 11**) est parfaitement ordonné. Les broches sont regroupées par fonctions, d'où un routage aisé vers le matériel externe. Tous les concepteurs de matériel apprécieront.
La documentation et le logiciel n'étant (à ce jour) pas encore dans leurs versions finales, quelques modifications mineures sont à attendre. Il n'en reste pas moins que la documentation est à la hauteur de celles auxquelles nous a habitués la fondation RPi : soignée et complète. La carte Pico est à matériel ouvert et accompagnée d'un SDK *open source* que vous pourrez exploiter aussi bien sur un RPi que sur un PC ou un Mac. Pour ce qui est de l'écriture de vos applications, vous aurez le choix entre MicroPython et C/C++.

## Développement en MicroPython

Les codeurs aguerris choisiront sans doute C/C++ pour programmer le RP2040 (ou utiliseront l'EDI Arduino, dont la compatibilité prochaine avec la carte Pico a été annoncée). MicroPython a l'avantage d'être plus simple. La plateforme de développement peut être un RPi 4 avec Raspberry Pi OS ou – d'après le manuel – un PC avec une distribution Linux basée sur Debian. De là il vous faudra : soit compiler vous-même un portage de MicroPython pour RPi OS en suivant les instructions du manuel [5] ; soit télécharger le fichier binaire UF2 d'installation depuis la page [6].
La dernière méthode est la plus simple : on maintient enfoncé le bouton BOOTSEL, on relie la Pico à un RPi (ou à un PC), puis on relâche le bouton une fois la carte connectée. Celle-ci sera reconnue comme nom de volume RPI-RP2. Ne reste alors plus qu'à y glisser le fichier UF2 pour flasher la mémoire du RP2040 et installer MicroPython.

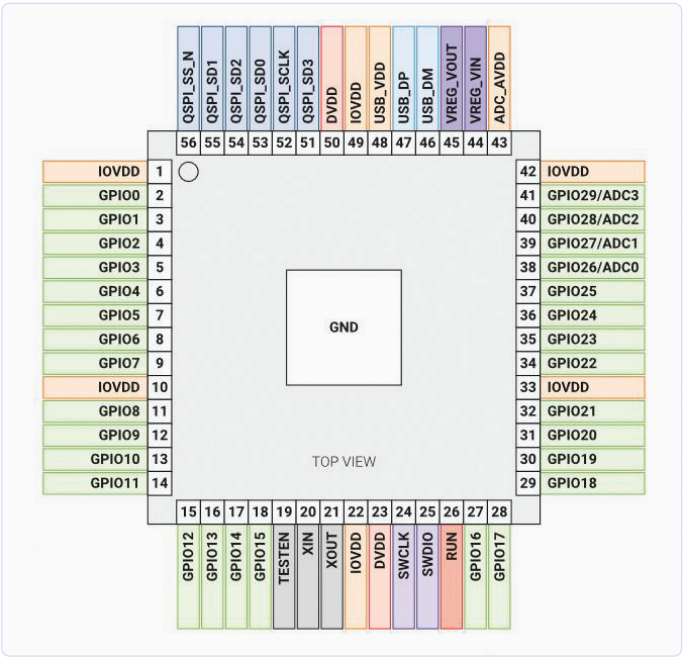



*Figure 11. Brochage du RP2040 [9].*

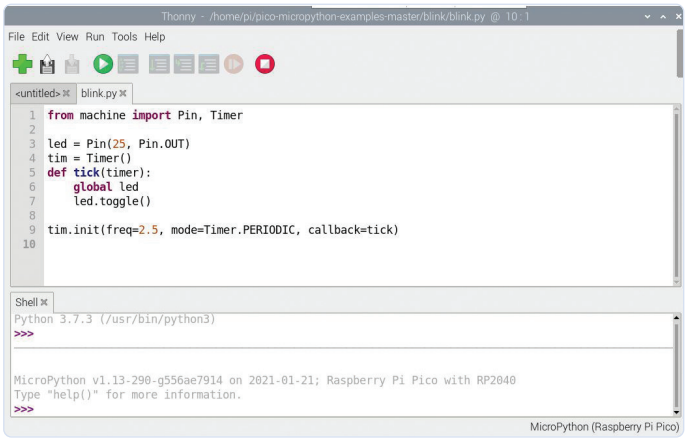



*Figure 12. L'EDI Thonny sur un RPi 4.*

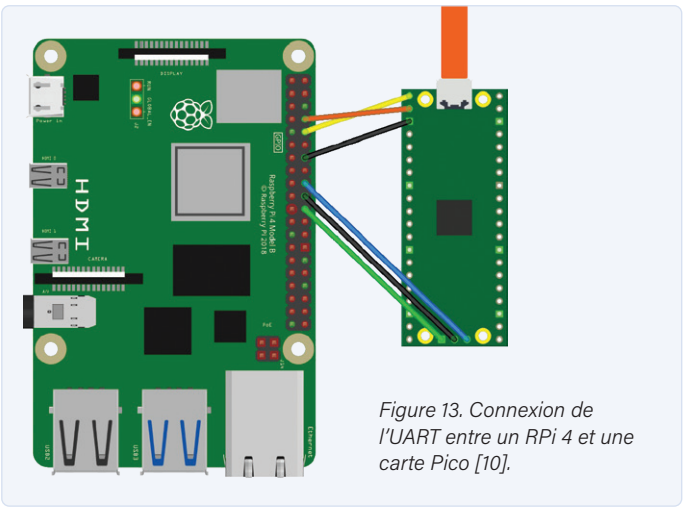


*Figure 13. Connexion de l'UART entre un RPi 4 et une carte Pico [10].*

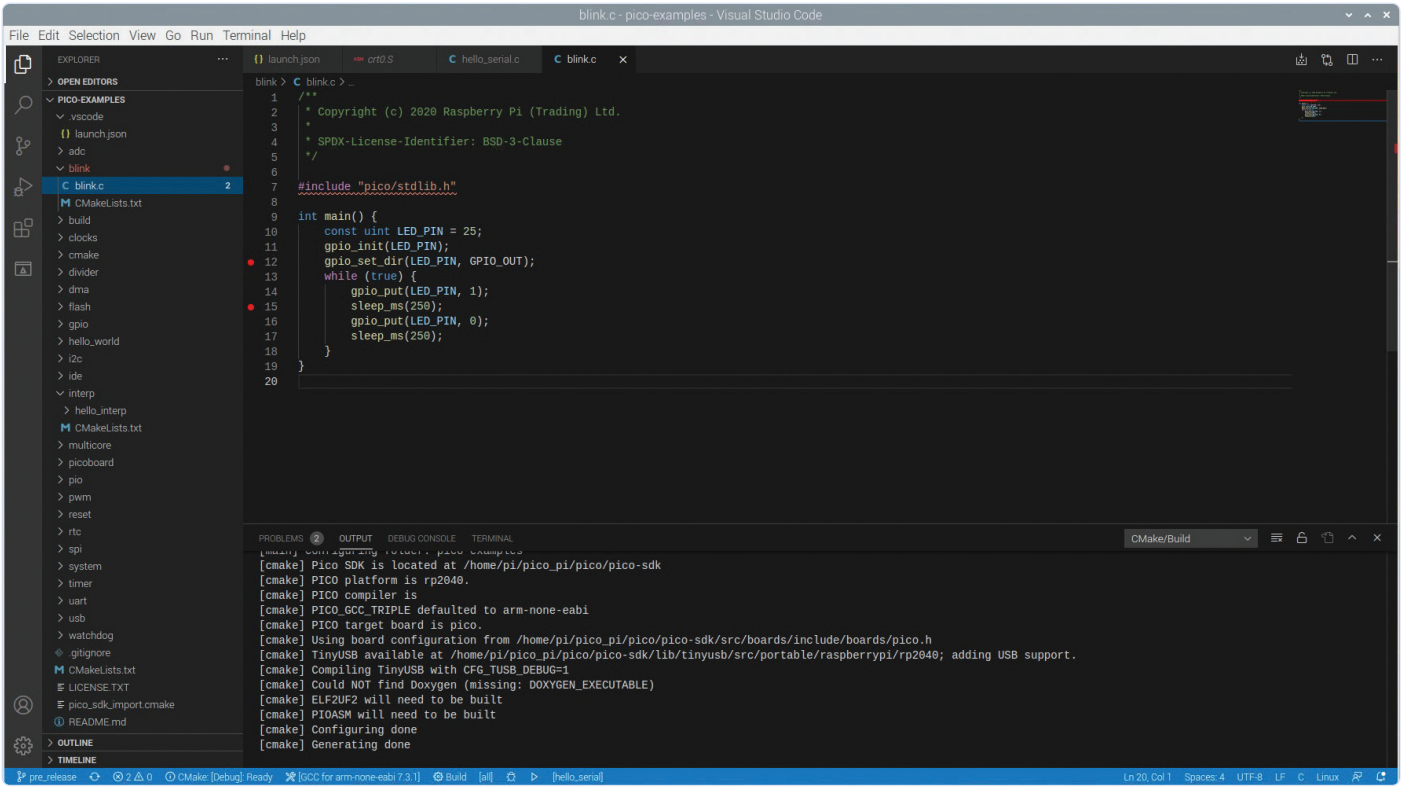



*Figure 14. L'EDI Visual Studio Code sur un RPi 4.*

Le manuel explique comment accéder à l'interpréteur interactif Python (aussi appelé REPL) via l'interface USB ou l'UART de la carte Pico. Ce REPL n'est pas toujours des plus pratique. L'interface MicroPython pour la Pico et d'autres cartes à RP2040 est heureusement reconnue par divers EDI, dont Thonny (**fig. 12**). Là encore le manuel explique comment l'installer et le configurer.

La fondation RPi a publié un livre d'initiation à MicroPython appelé *Get Started with MicroPython on Raspberry Pi Pico* (Halfacree & Everard, 2021). Les auteurs expliquent notamment l'usage des blocs PIO et comment interagir avec le monde extérieur (lien dans l'encadré **Produits**).

## Développement en C/C++

La plateforme privilégiée est le RPi 4, mais la mise en place d'autres environnements de développement est couverte par le manuel [7]. Un script à télécharger depuis [8] se charge de toutes les étapes nécessaires à l'installation sur le RPi. Le schéma de la **figure 13** illustre la connexion physique à établir entre le nano-ordinateur et la carte Pico. Dans cette configuration le RPi 4 sert également

## POURQUOI NE PAS AVOIR ÉVOQUÉ LE MODE BASSE CONSOMMATION DE LA PICO ?

C'est effectivement une des premières caractéristiques à étudier lorsqu'on acquiert un nouveau microcontrôleur. Un microcontrôleur en sommeil consomme généralement quelques centaines de nA ou quelques µA. D'après sa fiche technique, le RP2040 affiche une consommation moyenne de 0,18 mA dans son état de sommeil le plus profond appelé *dormant*. Cela semble de prime abord tout à fait acceptable, mais sur la carte Pico les conditions changent. Le RP2040 n'est en effet plus isolé, mais flanqué de la mémoire flash externe et du convertisseur CC/CC. Alors que le courant de veille de la mémoire flash ne vaut que 50 µA et peut même descendre à 15 µA, le convertisseur CC/CC doit encore fonctionner en présence de charges très basses. Si on prend ces éléments en compte, on trouve que la Pico en veille consomme 0,8 mA à 25 °C. Cette valeur est à garder à l'esprit si l'on projette d'alimenter la carte avec des piles ou accus durant une longue période. Dans ce cas, il faudra trouver un compromis entre les fonctions exploitées, la consommation de la carte et l'usage qui en est prévu. Le microcontrôleur parfait n'existe pas, mais cette première puce de la fondation RPi est prometteuse. L'avenir nous dira si ses concepteurs auront su gagner quelques µA en mode basse consommation, et comment ils y sont parvenus. On peut faire ici le parallèle avec le développement logiciel, où l'optimisation du code consiste à le rendre à la fois plus rapide et moins énergivore sur les appareils alimentés par piles ou accus.

d'interface de débogage grâce à une version modifiée d'OpenOCD. Avec un PC ou un Mac, évidemment dépourvus de broches GPIO, il faut utiliser une seconde carte Pico exécutant Picoprobe. La procédure est décrite dans le manuel.

Parmi les outils installés par le script figure l'EDI Visual Studio (**fig. 14**), a priori plus commode que l'interpréteur de commandes du RPi. Les programmes d'exemple que nous avons reproduits fonctionnaient comme attendu, mais le générateur de projets (conçu pour vous épargner l'instanciation en ligne de commande d'un nouveau projet) n'était pas encore tout à fait au point. Ce genre de petits défauts est le propre des nouveautés. Il en a été de même avec le RPi 4, mais tout cela sera corrigé à mesure que les utilisateurs signaleront les bogues et problèmes rencontrés.

Le RP2040 bénéficie d'un ensemble fourni de bibliothèques et d'exemples, suffisamment étayé pour que vous n'ayez pas à écrire vous-même les pilotes de chaque élément de la puce. La bibliothèque TinyUSB a été complétée pour prendre en charge le RP2040, donc pour l'USB aussi vous aurez sous la main tout ce qu'il faut pour démarrer rapidement. De plus le code est ouvert. Libre à vous donc de fouiller dans ses dossiers et de l'étudier à profit. Les EDI comme Wiring ou Arduino prennent en charge toutes les cartes intéressantes, donc gageons que la brillante Pico n'échappera pas à leur radar et que nous aurons bientôt au moins un EDI de plus à disposition.

Sur un PC ou un Mac, privilégiez une machine virtuelle pour la mise en place de votre environnement de développement, et créez régulièrement des instantanés du système. En cas de problème, il vous suffira de revenir à un état de la machine où le monde était encore merveilleux.

## Laissez-vous Picoter

Si jusque-là vous fuyiez les microcontrôleurs parce que : 1) trop chers à vos yeux et, 2) moins bien documentés que votre machine à laver, laissez-nous résumer cet article : une carte Pico ne coûte que 5 €, et la documentation de la fondation RPi explique tout, dit tout, sait tout, a réponse à tout. La question ne devrait pas être de savoir si vous devez acheter une Pico, mais combien en acheter. Les outils de développement, les exemples fournis et la pédagogie des manuels rendent la découverte de la Pico amusante. Même l'aridité apparente de la fiche technique peut vous happer ! La Pico se veut accessible à tous, et à ce titre saura aussi initier les plus jeunes aux microcontrôleurs. Partagez avec nous vos réalisations, nous sommes impatients de voir ce que vous tirerez de la Pico !

210045-04

**Votre avis, s'il vous plait ?**
Veuillez adresser vos questions et vos commentaires par courriel à mathias.claussen@elektor.com ou à redaction@elektor.fr.

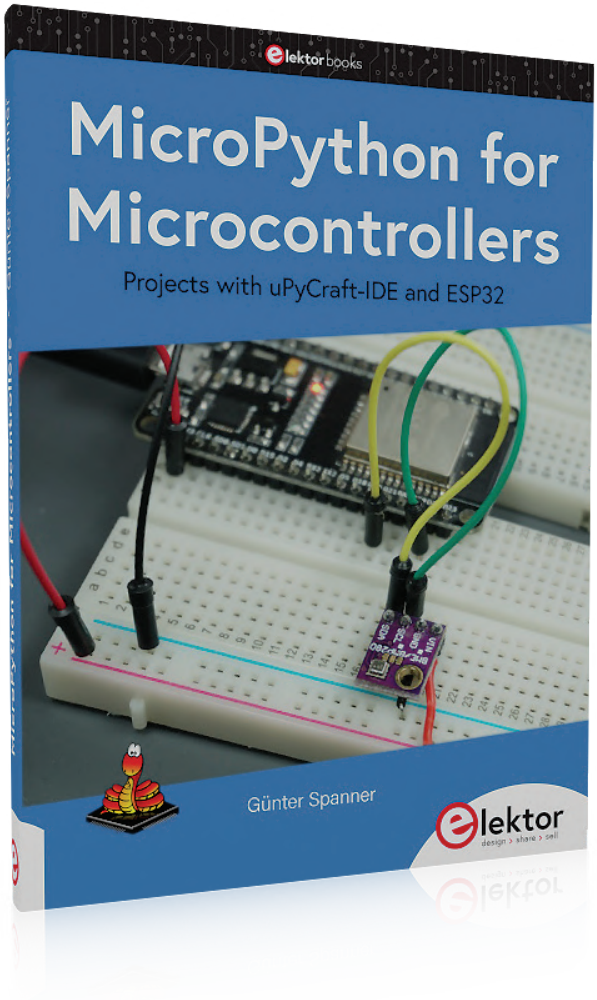
**Ont contribué à cet article**
Auteurs : Mathias Claußen et Luc Lemmens
Rédaction : Jens Nickel
Maquette : Giel Dols
Traducteur : Hervé Moreau



## LIENS

[1] **C. Valens, « Raspberry Pi Pico Review », Elektor.TV, 1/21/2021** : https://youtu.be/ijn-QDAgZss
[2] **C. Valens, Place au µC RP2040. Hue Pico, ElektorMagazine.fr, 22/01/2021** : http://elektormagazine.fr/news/microcontroleur-raspberry-pi-rp2040-pico-board
[3] **Fiche technique du RT6150** : https://datasheets.raspberrypi.org/pico/pico_datasheet.pdf
[4] **RT6150 datasheet** : www.richtek.com/assets/product_file/RT6150A=RT6150B/DS6150AB-04.pdf
[5] **Manuel pour utiliser MicroPython** : https://datasheets.raspberrypi.org/pico/sdk/pico_python_sdk.pdf
[6] **Démarrer avec la Pico** : www.raspberrypi.org/documentation/pico/getting-started/
[7] **Configuration C/C++** : https://datasheets.raspberrypi.org/pico/sdk/pico_c_sdk.pdf
[8] **Script d'installation** : https://github.com/raspberrypi/pico-setup/blob/master/pico_setup.sh
[9] **Fiche technique du RP2040** : https://datasheets.raspberrypi.org/rp2040/rp2040_datasheet.pdf
[10] **Getting started with Pico** : https://datasheets.raspberrypi.org/pico/getting_started_with_pico.pdf

# MicroPython
## pour les microcontrôleurs
### Afficheur riquiqui




**Günter Spanner** (Allemagne)

Voici un extrait du livre de Günter Spanner, « MicroPython for Microcontrollers » (Elektor 2021). Ce passage décrit l'association d'une carte à microcontrôleur ESP32 et d'un afficheur OLED SSD1306, le tout animé avec du code en MicroPython. Vous découvrirez ainsi la puissance et la facilité d'utilisation du langage MicroPython. Pour la pratique, réalisez l'afficheur d'ECG et l'horloge numérique.

Pour des tests et des applications simples, afficher des textes et des valeurs numériques envoyés sur le port série est largement suffisant. Toutefois, cette méthode exige la présence d'un PC ou au moins d'un ordinateur portable. Si cela dure plusieurs jours ou semaines, l'usage de ces appareils entraîne une consommation électrique inutile. En outre, réserver un ordinateur à chaque projet avec un microcontrôleur n'est évidemment ni envisageable ni faisable en pratique.

Pour juste afficher l'heure ou les valeurs de mesure de capteurs d'environnement, il est alors préférable de le faire sur un petit afficheur géré par le contrôleur. Très répandu, le modèle OLED SSD1306 a une taille de 2,4 cm (0,96 pouce) et une résolution de 128×64 pixels.

Pour cet afficheur, MicroPython contient par défaut un pilote lorsqu'il est chargé dans l'ESP32. Ce pilote permet d'afficher du texte et des valeurs numériques ainsi que des graphiques simples. Le SSD1306 est équipé d'une mémoire vive interne et d'un oscillateur embarqué et n'a besoin d'aucun composant externe supplémentaire. De plus, il dispose d'une commande de luminosité à 256 niveaux. Ses principales caractéristiques sont les suivantes :

- Résolution : matrice de 128×64 pixels
- Tension d'alimentation : 1,65 V à 3,3 V
- Plage de température de fonctionnement : –40 °C à +85 °C
- Tampon d'affichage SRAM de 128×64 bits intégré
- Fonction de défilement continu sur les axes horizontal et vertical
- Oscillateur embarqué

Les écrans OLED se passent de rétroéclairage, car chaque pixel de l'écran est capable d'émettre de la lumière. Cela les rend faciles à lire même en cas de conditions d'éclairage ambiant défavorables. En outre, leur contraste est nettement supérieur à celui des écrans à cristaux liquides (LCD). Comme un pixel ne consomme de l'énergie que lorsqu'il s'allume, ils sont très efficaces sur le plan énergétique. Les versions les plus simples du SSD1306 n'ont que quatre broches, suffisantes pour gérer l'affichage via le bus I²C. Certaines versions disposent de broches de réinitialisation ou d'une interface SPI supplémentaire, mais la version simple suffit pour la plupart des applications. Le tableau suivant spécifie toutes les connexions nécessaires.

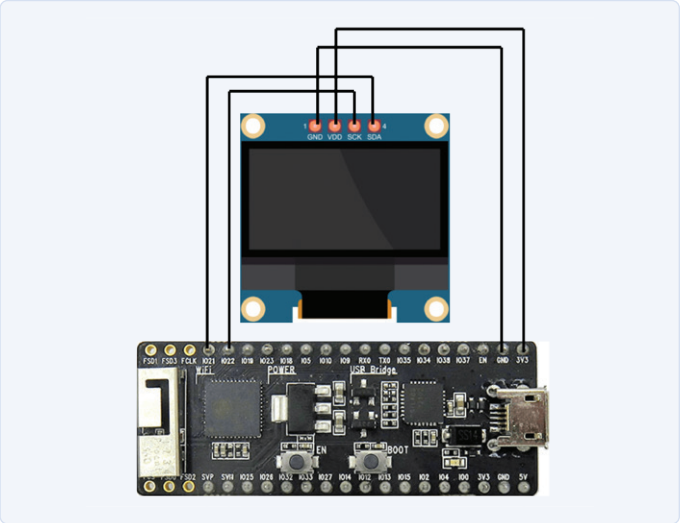


*Figure 1. Afficheur SSD1306 connecté à la carte à microcontrôleur ESP32.*

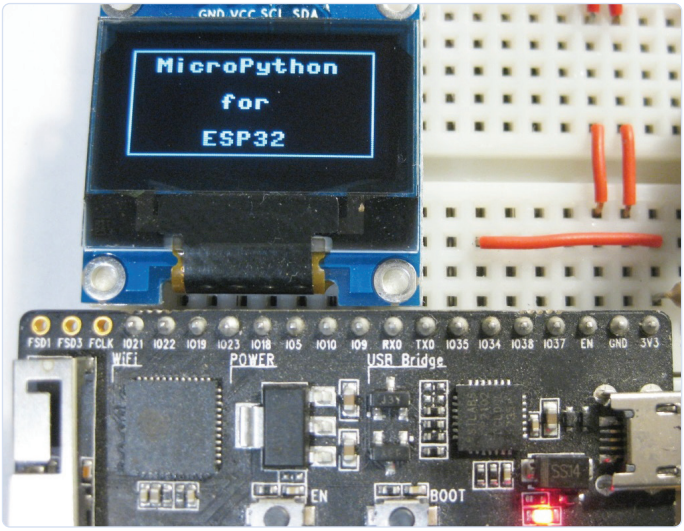



*Figure 2. Affichage de texte et graphique sur le SSD1306.*

| broche OLED | broche ESP32 |
|---|---|
| VDD | 3V3 |
| GND | GND |
| SCK | GPIO 22 |
| SDA | GPIO 21 |

La **figure 1** montre le schéma de câblage correspondant. Le programme suivant affiche un texte et un élément graphique simple (un cadre) :

```
# SSD1306_DEMO.py

from machine import Pin, I2C
from ssd1306 import SSD1306_I2C

i2c=I2C(-1,scl=Pin(22),sda=Pin(21))
# I2C Pin assignment
oled_width=128
oled_height=64
oled = SSD1306_I2C(oled_width, oled_height, i2c)
lin_hight = 9
col_width = 8
def text_write(text,lin, col):
oled.text(text,col*col_width,lin*lin_hight)

oled.fill(0)
text_write("MicroPython", 1, 2)
text_write("for", 3, 6)
text_write("ESP32", 5, 5)

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

La **figure 2** montre le résultat sur l'afficheur. Ensuite, pour piloter l'affichage, il faut importer les modules requis. Comme déjà mentionné, les bibliothèques *machine* et *SD1306* sont en principe déjà disponibles en tant que bibliothèques standard. En cas de nécessité, un fichier *ssd1306.py* peut être téléchargé séparément sur la carte. La déclaration des broches pour le bus I²C se fait ainsi :

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

Le nombre de pixels disponibles sur le module connecté est défini par les variables :

```
oled_width = 128
oled_height = 64
```

Le paramètre '`-1`' indique que le module utilisé n'a pas de broches de réinitialisation ou d'interruption. Avec ces informations passées en arguments, on crée un objet `SSD1306_I2C` appelé `oled :`

```
oled = ssd1306.SSD1306_I2C (oled_width, oled_height,
   i2c)
```

L'écran est maintenant prêt à fonctionner. La fonction `text()` est utilisée pour afficher des informations sur l'écran. L'affichage est rafraîchi avec la méthode `show()`. La fonction `text()` accepte les arguments suivants :

- Message (texte)
- Coordonnées X et Y du début du texte en pixels
- Couleur du texte (option) : 0 = noir (sombre) et 1 = blanc (clair)

L'instruction suivante affiche un message en blanc ou bleu sur fond sombre. Le texte commence à la position X = 0 et Y = 0 :

```
oled.text ('MicroPython!', 0, 0)
```

La méthode `show()` rend les modifications visibles sur l'écran. La bibliothèque contient également d'autres méthodes utiles. Avec `oled.fill(1)`, on allume tous les pixels (écran complètement clair) alors qu'avec `oled.fill(0)`, on les éteint tous (écran complètement sombre).

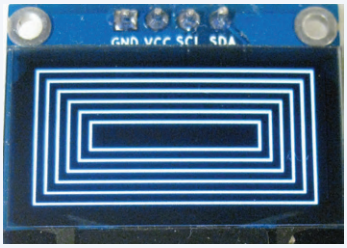


*Figure 3. Cadres concentriques dessinés sur un écran OLED.*

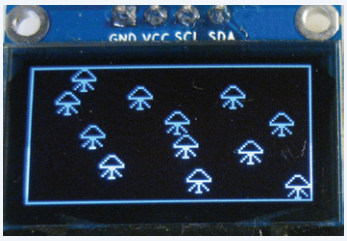


*Figure 4. Éléments graphiques sur l'écran OLED.*

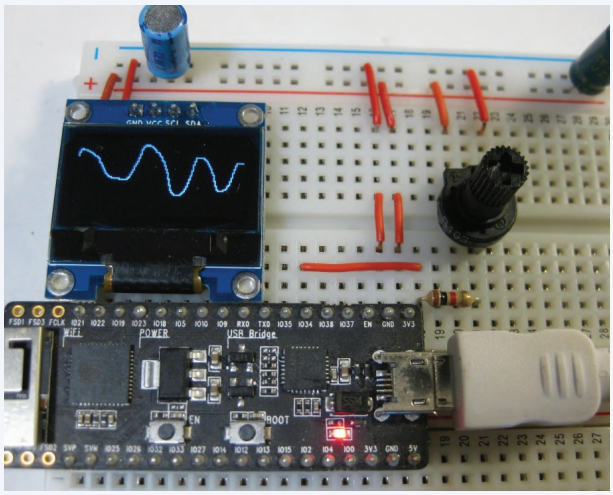


*Figure 5. Affichage de courbe sur l'écran OLED.*

**Listage 1. Démonstration de l'affichage de bitmap sur le SSD1306**

```
# SSD1306_bitmap_DEMO.py

from machine import Pin, I2C
import ssd1306
import urandom

i2c = I2C(-1, scl=Pin(22), sda=Pin(21))
oled_width = 128
oled_height = 64
oled = ssd1306.SSD1306_I2C(oled_width, oled_height,
   i2c)

# frame
oled.hline(0, 0, oled_width-1, 1)
oled.hline(0, oled_height-1, oled_width-1, 1)
oled.vline(0, 0, oled_height-1, 1)
oled.vline(oled_width-1, 0, oled_height, 1)
oled.show()
```

```
ICON = [
    [0, 0, 0, 0, 1, 1, 1, 0, 0, 0, 0],
    [0, 0, 0, 1, 0, 0, 0, 1, 0, 0, 0],
    [0, 0, 1, 0, 0, 0, 0, 0, 1, 0, 0],
    [0, 1, 0, 0, 0, 0, 0, 0, 0, 1, 0],
    [1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1],
    [1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1],
    [0, 0, 0, 0, 1, 1, 1, 0, 0, 0, 0],
    [0, 0, 0, 1, 0, 1, 0, 1, 0, 0, 0],
    [0, 0, 1, 0, 0, 1, 0, 0, 1, 0, 0],
    [0, 1, 0, 0, 0, 1, 0, 0, 0, 1, 0],
    [0, 0, 0, 0, 1, 1, 1, 0, 0, 0, 0],
]
for n in range(12):
  xofs = urandom.randint(1, oled_width-12)
  yofs = urandom.randint(1, oled_height-12)
  for y, row in enumerate(ICON):
    for x, c in enumerate(row):
      oled.pixel(x+xofs, y+yofs, c)
oled.show()
```

**Listage 2. Affichage d'ECG en continu.**

```
# Rolling_ECG_display.py
from machine import Pin, ADC, I2C from time import
   sleep
import ssd1306

i2c = I2C(-1, scl=Pin(22), sda=Pin(21))
  oled_width = 128
  oled_height = 64
  oled = ssd1306.SSD1306_I2C(oled_width, oled_
   height, i2c)

  pot = ADC(Pin(34))
  pot.atten(ADC.ATTN_11DB) #Full range: 3.3v vMax =
   3.4
  dotPos_old = int(oled_height/2)
while True:
  pot_value = pot.read()
  voltage = 0.000816*pot_value + 0.037822 #
   print(voltage)
dotPos_new = int(voltage/vMax*oled_height)

oled.line(0, dotPos_new, 0, dotPos_old, 1)
oled.scroll(1, 0)
oled.line(0, dotPos_new, 0, dotPos_old, 0)

dotPos_old = dotPos_new oled.pixel(0, int(oled_
   height/2), 1)
oled.show()
```

La méthode `pixel()` permet des représentations graphiques. Elle accepte les arguments suivants :

- Coordonnée X : position horizontale du pixel
- Coordonnée Y : position verticale du pixel
- Couleur du pixel : 0 = noir, 1 = blanc

Pour allumer un seul pixel dans le coin supérieur gauche :

```
oled.pixel (0, 0, 1)
```

Avec

```
oled.invert (True)
```

on inverse les couleurs sur l'écran : le blanc devient noir et vice versa. On revient aux couleurs d'origine avec `oled.invert(False)`.

## Représentations graphiques

En plus des instructions de gestion des pixels, d'autres commandes graphiques sont disponibles. Pour tracer des lignes horizontales et verticales, utilisez `.hline()` ou `.vline()` qui acceptent comme arguments la position de départ XY, la longueur et la couleur de la ligne.

Le petit programme suivant dessine des cadres rectangulaires concentriques sur l'écran. La **figure 3** montre le résultat.

```
# SSD1306_frames.py
from machine import Pin, I2C
import ssd1306

i2c = I2C(-1,scl=Pin(22),sda=Pin(21))
oled_width = 128
oled_height = 64
oled = ssd1306.SSD1306_I2C(oled_width, oled_height,
   i2c)

for n in [0,5,10,15,20,25]:
oled.hline(n, n, oled_width-1-2*n, 1-2*n)
oled.hline(n, oled_height-1-n, oled_width-1-2*n,
   1-2*n)
oled.vline(n, n, oled_height-1-2*n, 1-2*n)
oled.vline(oled_width-1-n, n, oled_height-2*n, 1-2*n)
oled.show()
```

Les diagonales sont tracées en utilisant une ligne (x1, y1, x2, y2, c) entre deux points définis (x1, y1) et (x2, y2). Le paramètre `c` indique la couleur de la ligne dessinée. Les graphiques simples peuvent être écrits pixel par pixel dans la mémoire tampon de l'écran. Le programme du **listage 1** en fournit un exemple dont l'affichage est reproduit sur la **figure 4**.

## Afficheur OLED comme traceur de courbes

Outre les bitmaps, les données de mesure peuvent être affichées graphiquement sur l'écran OLED, c'est-à-dire qu'il est possible d'effectuer des affichages graphiques indépendants sans passer par la fonction de traceur de Thonny (EDI utilisé pour MicroPython), Le programme du **listage 2** fournit un affichage à défilement comparable aux ECG professionnels dans les hôpitaux.

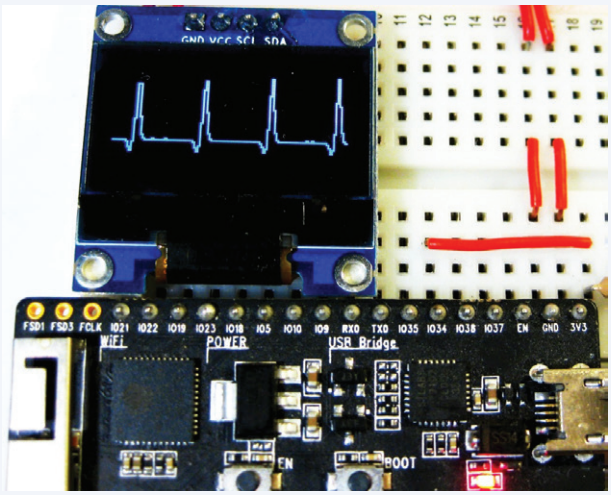



*Figure 6. Un électrocardiogramme sur l'écran OLED.*

En connectant un potentiomètre à l'entrée 34 du convertisseur A-N, on peut enregistrer la valeur mesurée. Après le démarrage du programme, les valeurs de tension sont affichées sur l'écran en continu grâce à la fonction de défilement intégrée. L'instruction

```
oled.scroll(1, 0)
```

déplace l'ensemble du contenu de l'écran d'un pixel. Pour avoir une courbe continue au lieu de pixels isolés, on les relie par des segments en utilisant les deux variables :

```
dotPos_old and dotPos_new
```

Elles servent à tracer une ligne entre la valeur courante et la valeur précédente. Puis, l'affichage est décalé d'un pixel, la valeur courante est transférée dans la mémoire tampon :

```
dotPos_old = dotPos_new
```

et le cycle recommence. La **figure 5** montre un exemple de valeurs de potentiomètre qui changent continuellement. Si vous disposez d'un amplificateur d'ECG, vous pouvez enregistrer les signaux électriques du cœur humain et obtenir ainsi l'électrocardiogramme typique bien connu dans les unités de soins intensifs des hôpitaux (**fig. 6**).

## Horloge numérique avec affichage OLED

Une autre application d'affichage utile est celle de l'heure. Elle transforme l'ESP32 et le SSD1306 en horloge numérique. La fonction `time()` du module de l'heure renvoie le nombre de secondes depuis la mise sous tension ou la réinitialisation de la carte. Au moyen de la variable

```
time_offset=20*3600+00*60+0 # hh+mm+ss
```

on spécifie une heure de début en secondes. Pour la valeur ci-dessus, l'horloge commence à 20:00:00. La fonction `time_text()` :

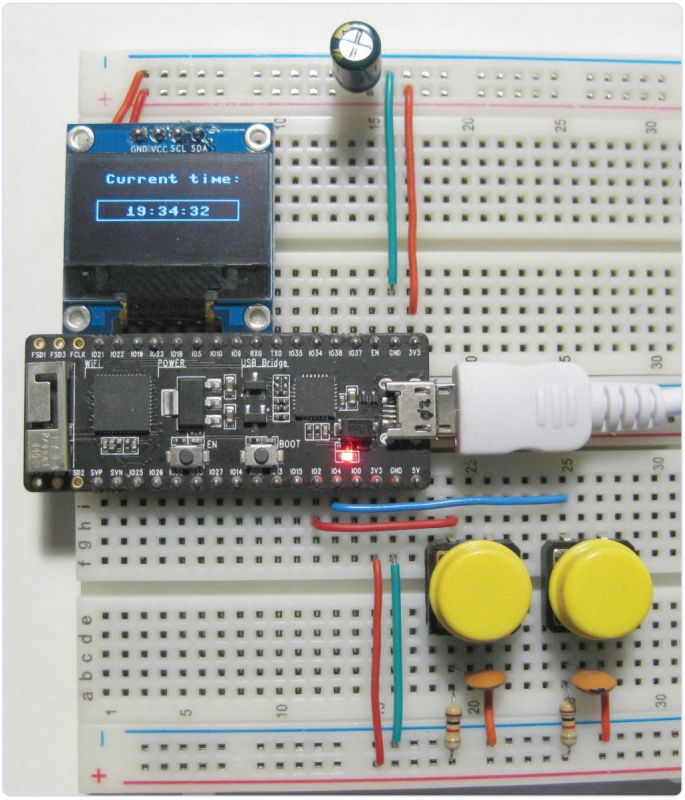



*Figure 7. Horloge numérique avec affichage OLED.*

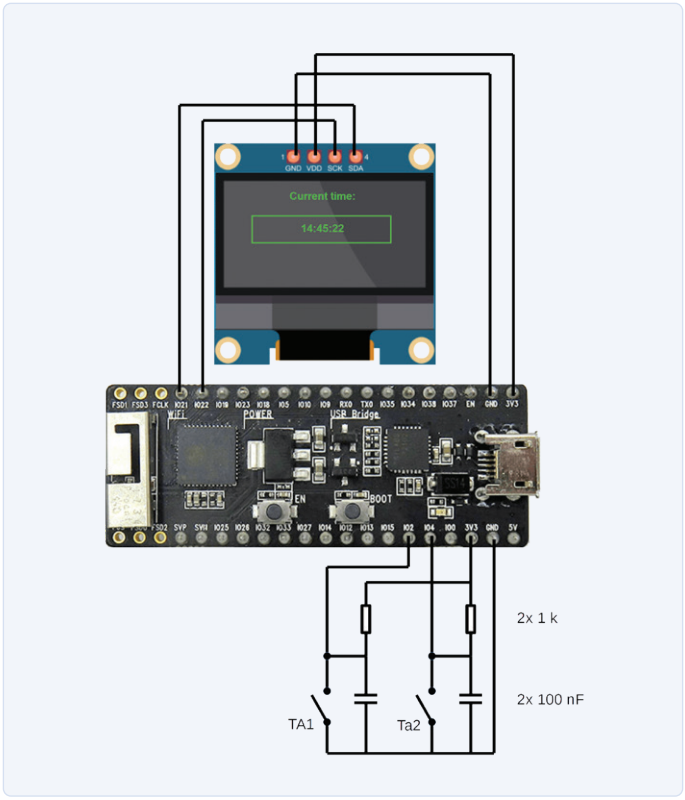



*Figure 8. Schéma de l'horloge numérique programmée en MicroPython.*

**Listage 3. Horloge ESP32 / SSD1306 en MicroPython.**

```
# setable_clock.py

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

i2c = I2C(-1,scl=Pin(22),sda=Pin(21))
oled_width=128
oled_height=64
oled = SSD1306_I2C(oled_width,oled_height,i2c)

time_offset=20*3600+00*60+0 # hh+mm+ss
lin_hight=5
col_width=8

def handle_interrupt_min(pin):
  global time_offset
  time_offset+=60
  time.sleep(.2)

def handle_interrupt_hr(pin):
  global time_offset
  time_offset+=3600
  time.sleep(.2)

button_min = Pin(4, Pin.IN)
button_min.irq(trigger=Pin.IRQ_RISING,
   handler=handle_interrupt_min)

button_hr = Pin(2, Pin.IN)
button_hr.irq(trigger=Pin.IRQ_RISING,
   handler=handle_interrupt_hr)

def text_write(text, lin, col=0):
  oled.text(text, col*col_width, lin*lin_hight)

def time_text(time):
  secs=time%60
  mins=(time//60)%60
  hours=(time//3600)%24
  return "{:02d}:{:02d}:{:02d}".
   format(hours,mins,secs)

def show():
  oled.fill(0)
  text_write("Current time:",1,2)
  current_text = time_text(current_time)
  text_write(current_text,6,4)
  oled.rect(10,25,108,16,1)
  oled.show()

while True:
  current_time=time_offset+time.time()
  show()
```

```
def time_text(time):
secs=time%60
mins=(time//60)%60
hours=(time//3600)%24
return "{:02d}:{:02d}:{:02d}".format(hours,mins,secs)
```

décompose cette valeur en heures, minutes et secondes et retourne la représentation habituelle *hh:mm:ss*, (**fig. 7**). Deux boutons-poussoirs (Ta1 et Ta2) reliés aux ports 02 et 04 comme indiqué sur la **figure 8** permettent de régler l'horloge. Les boutons ont en parallèle des condensateurs de 100 nF pour l'anti-rebond. Les résistances de 1 kΩ servent de rappel. À chaque pression, les routines d'interruption associées :

```
def handle_interrupt_min(pin):
global time_offset
time_offset+=60
time.sleep(.2)
```

et

```
def handle_interrupt_hr(pin):
global time_offset
time_offset+=3600
time.sleep(.2)
```

ajoutent respectivement 60 s (1 min) ou 3600 s (1 h) à l'heure de début. Le **listage 3** donne le programme complet de l'horloge numérique.

(200581-04)

Note de l'éditeur : le livre « MicroPython for microcontrollers » est disponible à l'adresse suivante www.elektor.fr/micropython-for-microcontrollers.
Le code extrait du livre est disponible gratuitement sur la page de l'article [1].

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

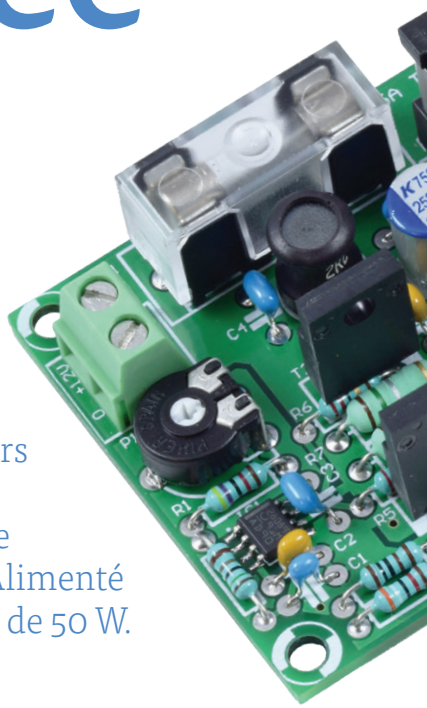

**Contributeurs**
Texte et illustrations : **Günter Spanner**
Rédaction : **Jan Buiting**
Mise en page : **Giel Dols**
Traduction : **Helmut Müller**


## LIEN

[1] **Code extrait du livre « MicroPython for microcontrollers » :** www.elektormagazine.fr/200581-04

# convertisseur CC/CC 12 à 200 V

## pour amplificateurs à tubes



**Ton Giesberts** (Elektor)

Voici un convertisseur CC/CC assez facile à construire pour amplificateurs à tubes. Il utilise des composants traversants (un seul CI *SO-8*), sa sortie haute tension est isolée galvaniquement. De fabrication maison, le transformateur a un rapport adaptable pour régler la tension de sortie. Alimenté par un adaptateur secteur de 12 VDC/5 A, sa puissance de sortie max. est de 50 W.

La partie haute tension du circuit d'un amplificateur à tubes est importante, mais aussi dangereuse : à l'entrée nous avons la tension du secteur – potentiellement mortelle, et en sortie, il délivre une tension continue assez élevée pour vous électrocuter.

L'alimentation la plus simple offrant la sécurité requise se compose d'un transformateur bien isolé, d'un redresseur et d'un condensateur de filtrage. Sa tension de sortie dépend de la tension du secteur, du transformateur, de la chute de tension aux bornes du redresseur, de la charge et de l'ondulation résiduelle. Ce type de montage fournit souvent la haute tension d'amplificateurs à tubes. Malheureusement les transformateurs du commerce ont souvent un rapport fixe, et la tension de sortie est trop haute ou trop basse, même si de nombreux amplificateurs s'en accommodent.

Nous utiliserons un adaptateur secteur de 12 V et un convertisseur CC/CC sur mesure pour obtenir jusqu'à 200 V. Ainsi, il n'y a pas de liaison directe avec le secteur, et la basse tension CC d'entrée pourra être employée pour les chauffages.

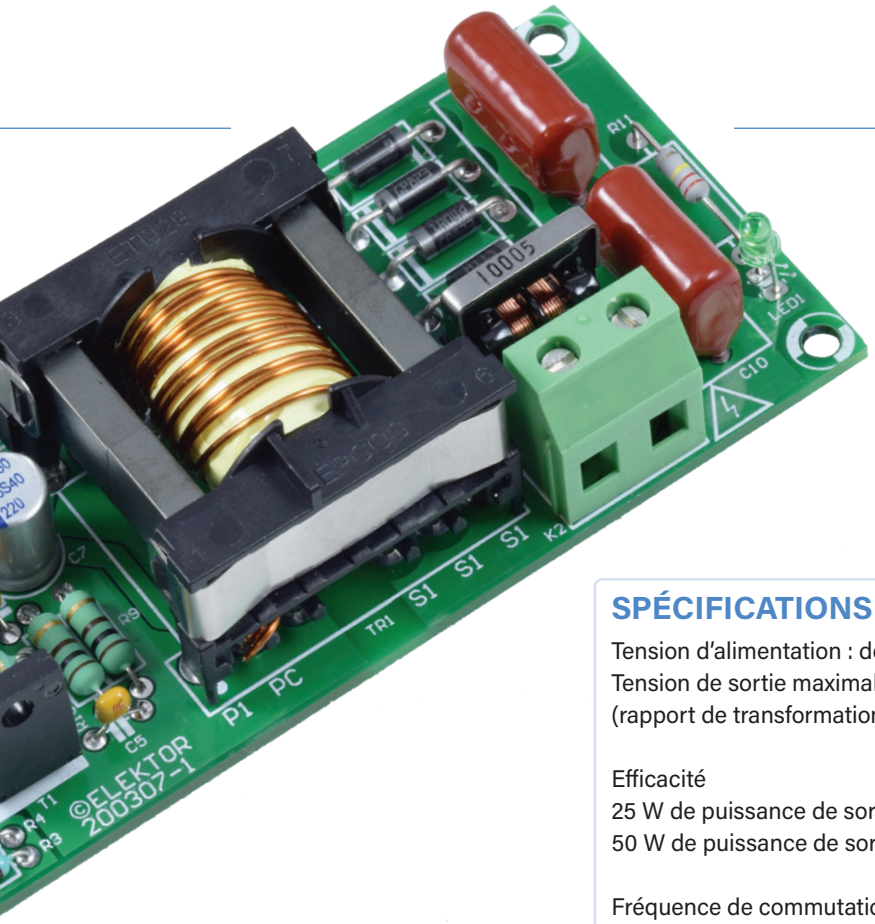
Nous présentons ici un résumé des informations complètes – théoriques et pratiques – que le concepteur de ce projet a rassemblées sur une page *Elektor Labs*. Si vous êtes juste intéressé par des notions de base et la construction de ce convertisseur 12 VCC/200 VCC, cet article répondra à votre attente. Si les considérations et calculs théoriques et pratiques à aborder pour construire un convertisseur CC/CC *push-pull* et son transformateur vous intéressent, suivez le lien [1].

### Fonctionnement du convertisseur CC/CC

Le schéma du convertisseur CC/CC est présenté à la **figure 1.** Pour faciliter la réalisation, nous utilisons des composants traversants, hormis un CI en boîtier CMS SO-8 : un *UCC28089* (IC1), oscillateur *push-pull* avec contrôle du temps mort monté du côté primaire. Les étages de sortie peuvent absorber 1 A et fournir 0,5 A et sont idéaux pour piloter des *MOSFET*. L'absence de boucle d'asservissement dans le schéma implique l'absence d'instabilité et de bruit induits par la commande MLI. Le rapport cyclique est maximal, c.-à-d. que le temps mort entre deux mises en conduction de chacun des MOSFET est réglé au minimum absolu (voir ci-dessous). La tension de sortie redressée au secondaire est presque continue : la capacité de filtrage nécessaire est faible. L'UCC28089 est limité en intensité, mais le seuil de 0,725 V de limitation du courant est relativement élevé. La perte de puissance dans le *shunt* mesurant le courant à travers les MOSFET est trop élevée et influence notablement le rendement global. À 5 A, (limite que nous avons fixée pour ce circuit), cela équivaudrait à 5 A × 0,725 V = 3,625 W ! Il faudrait une résistance *shunt* de forte valeur et de puissance nominale élevée, dans ce cas 5 W au moins. Avec 12 V, 50 W de puissance de sortie et un rendement de 86 %, cela équivaudrait à une perte de rendement de 7 %. Une façon de réduire cette perte dans le *shunt* est d'utiliser un diviseur de tension (R3 et R4) à partir de la tension d'alimentation pour décaler un peu l'entrée de détection de courant.

Le potentiomètre P1 règle la fréquence de l'oscillateur de l'UCC28089 (200 à 560 kHz env.). La fréquence de commutation est la moitié de celle-ci, soit 100 à 280 kHz. Selon la fiche technique, la résistance passante $RDS_{on}$ des MOSFET *TK30A06N1* est de 15 mΩ max. (VGS 10 V), donc presque négligeable. Ils ont une très faible capacité de transfert inverse de 33 pF et de faibles temps de commutation ($t_{on}/t_{off}$ 21/28 ns). De plus, la capacité d'entrée de 1050 pF est inférieure à celle de la plupart des MOSFET de $RDS_{on}$ identique. À puissance maximale, aucun refroidissement auxiliaire des transistors n'est nécessaire, mais rester longtemps à la puissance maximale est déconseillé.

Dans un convertisseur *push-pull*, l'énergie est directement transférée vers le secondaire. Lorsqu'un MOSFET est conducteur, l'autre est coupé et l'énergie est directement trans-




## INFOS SUR LE PROJET

**Mots clés**
alimentation, convertisseur CC/CC, amplificateur à tubes, haute tension, composants traversants, transformateur maison

**Niveau**
débutant – **connaisseur** – expert

**Temps**
8 h environ

**Outils**
outils de soudure, petite panne pour la soudure du SO-8

**Cost**
45 € environ

## SPÉCIFICATIONS

Tension d'alimentation : de 9 à 14 VCC
Tension de sortie maximale : 350 V (rapport de transformation max.)

Efficacité
25 W de puissance de sortie 88 %
50 W de puissance de sortie 86 %

Fréquence de commutation
100/150/280 kHz P1 min/milieu/max

Courant à vide 12 VDC
300/190/120 mA P1 min/milieu/max

*Une charge élevée empêchera le convertisseur de démarrer. Ce convertisseur a été conçu pour les amplificateurs à tubes où la charge sur la haute tension est très faible à la mise sous tension !*

férée. Un temps mort est nécessaire entre les commutations pour éviter que l'enroulement primaire ne soit court-circuité. Si les deux MOSFET conduisaient en même temps, les champs magnétiques des deux primaires s'annihileraient mutuellement entraînant un courant extrêmement élevé, principalement limité par la résistance série des enroulements primaires, les MOSFET, le *shunt*, les pistes de cuivre et l'alimentation de 12 V CC. Les résistances *shunt* brûleraient et les MOSFET aussi si le court-circuit durait ne serait-ce que le temps d'un léger chevauchement. Un fusible de 5 A protège donc l'alimentation de 12 V. Un avantage du convertisseur *push-pull* est que le courant de crête à travers les MOSFET n'est que légèrement supérieur au courant de charge moyen du côté primaire (courant que l'adaptateur CA doit fournir). Un courant de crête faible dans les MOSFET réduit les pertes par conduction et améliore le rendement du convertisseur.

## Le transformateur

Le transformateur est la partie la plus délicate de ce convertisseur CC/CC ; pour un premier prototype, il doit être fabriqué à la main. À 150 kHz, la puissance de sortie peut théoriquement être supérieure à 170 W en utilisant un corps de bobine *ETD29* [2] et du matériau *N97*, mais il y a aussi des pertes dans le cuivre. Ce transformateur n'a pas d'entrefer. Cela réduit le nombre de spires des enroulements (primaires et secondaire) et facilite sa

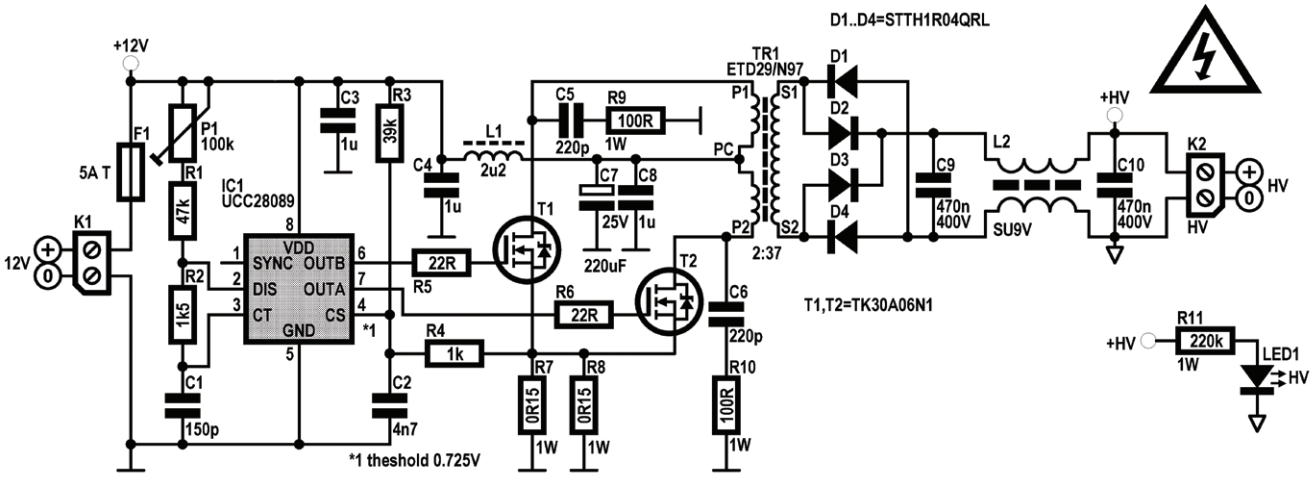



*Figure 1. Schéma de principe du convertisseur CC/CC.*

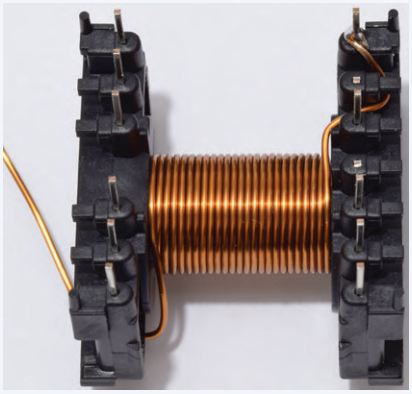


*Figure 2a. Première couche secondaire.*

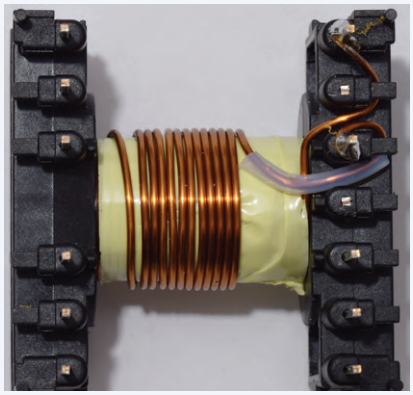


*Figure 2b. Deuxième couche secondaire.*

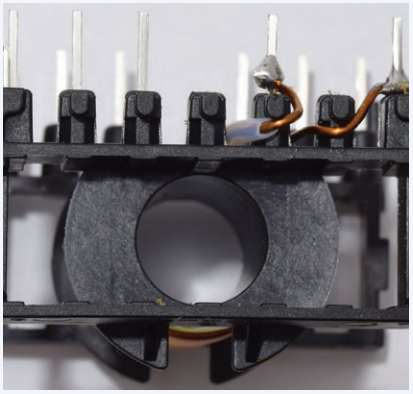


*Figure 2c. Bobinage secondaire soudé.*

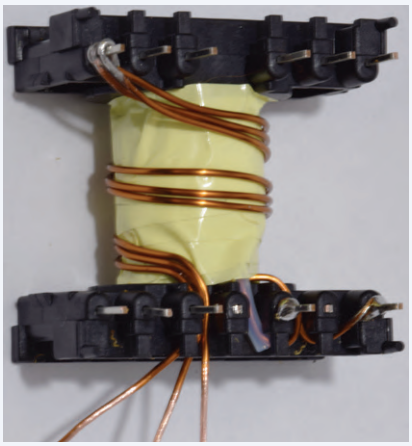


*Figure 2d. Premier enroulement primaire, notez les trois fils en parallèle.*

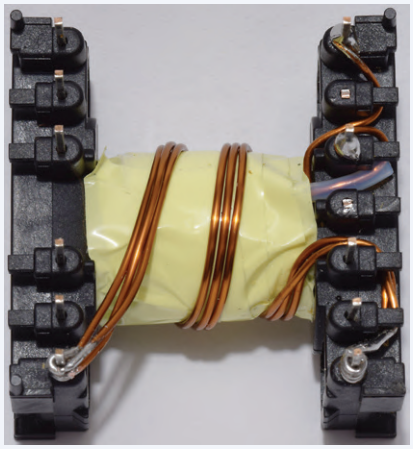


*Figure 2e. Premier primaire, avant la soudure.*

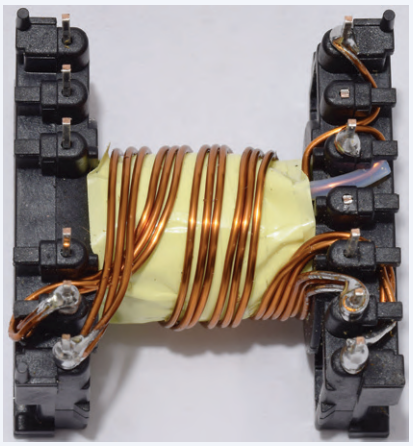


*Figure 2f. Deuxième primaire ajouté, transformateur prêt.*

*Figure 2. Construction du transformateur.*

construction. L'absence d'entrefer implique une inductance plus élevée et un meilleur couplage entre les enroulements. Le même fil convient pour tous les enroulements : du Ø 0,7 mm s'enroule facilement autour du corps de bobine et permet compter les tours.
L'idée est que le rapport de transformation détermine la tension de sortie. Pour un convertisseur CC-CC *push-pull*, c'est le rapport entre le nombre de spires du primaire et celui du secondaire. Ici, nous avons deux primaires de deux tours chacun et un seul secondaire (37 tours). Avec une tension primaire de 12 V, un rapport de transformation de 2/37 donnerait théoriquement une tension au secondaire de 222 V. Comme aucun transformateur n'est idéal, il fournira une tension de sortie plus faible pour une charge donnée. La tension de sortie étant élevée, la chute de tension aux bornes du redresseur et du filtre est négligeable. Pour pouvoir modifier le transformateur fini, par ex. si l'écart de la tension de sortie est plus élevé que prévu, il est logique de commencer par les enroulements primaires et de les isoler avec un ruban spécial et des manchons isolants. L'enroulement secondaire sera bobiné par-dessus et

## UN PEU DE THÉORIE DES TRANSFORMATEURS

La tension induite dans une bobine est conforme à la loi de Faraday (simplifiée) :

$E = N * B * A / t$

où
E tension [V].
N nombre de tours
B densité de flux [T]
A surface de la bobine [$m^2$]
$t = T/2 = 1 / 2f$

Dans un convertisseur *push-pull*, les tensions appliquées aux enroulements primaires sont des ondes presque carrées. En une demi-période, l'induction du noyau varie de $+B_{max}$ à $-B_{max}$ par enroulement, et vice versa. Les enroulements sont en opposition de phase. Donc l'amplitude de l'induction vaut $2 * B_{max}$ et la tension induite est :
$E = N * 2 * B_{max} * A * 2f$

Pour l'un des enroulements primaires, on obtient alors l'équation suivante :
$NP = EP * 10^4 / (4 * f * B_{max} * A)$
avec la surface A en $cm^2$ (d'où le $10^4$)

## TAILLE DU NOYAU ET PUISSANCE MAXIMALE DE SORTIE

Ce sujet reste la partie la plus difficile de la conception d'un convertisseur CC/CC utilisant un transformateur. Sur l'internet, on trouve tout sur le calcul des transformateurs, même des calculateurs complets. La conception d'un transformateur devient très facile. Mais la plupart des sites passent sous silence le calcul de la taille du noyau et se réfèrent à des tableaux similaires pour toutes sortes de noyaux et corps de bobines tels que Exx, EExx, EFxx, EFDxx, EIxx, ETDxx et EERxx.
En général, les tableaux sont triés par puissance max., mais la fréquence de travail est souvent absente. Pour connaître la taille du noyau, il y a une formule qui utilise le produit *WaAc* (multiplication de la surface de la section du noyau *Ac* et de la surface de la fenêtre disponible pour les enroulements *Wa*). Dans les formules, il faut entrer des facteurs tels que la constante de topologie (liée au type de convertisseur), la densité de courant (liée à l'élévation max. de température permise) et le facteur d'utilisation de la fenêtre. Mais ce n'est pas un calcul exact, il y a trop de dépendances.
Autrefois, on trouvait chez Block des kits de construction de transformateurs. Un kit contenait un petit rouleau de film isolant, des manchons isolants, deux petits cartons pour définir l'entrefer, un corps de bobine, deux demi-noyaux et deux clips. Bien sûr, il fallait se procurer du fil de cuivre ailleurs. La boîte contenait également un dépliant intitulé « Berechnungsbogen für Schaltnetzteil-Übertrager EB » (fiche de calcul pour transfos d'alimentation à découpage). Elle donnait des formules pour calculer la taille minimale du noyau, les tours et le diamètre du fil pour le primaire et le secondaire. À l'époque, le matériau du noyau était N27. Le gros avantage du N97 est la diminution des pertes du noyau : 300 kW/m³ au lieu de 920 kW/m³ (à 100 kHz/200 mT/100 °C). Voici la formule de calcul de la taille min. du noyau :

taille min. du noyau [mm] =
4,7 * 106 * puissance de sortie totale max. / fréquence [W/Hz]

Selon cette formule, la puissance max. à 150 kHz pour un noyau de 5350 mm³ est de 170 W, c'est pourquoi j'ai dit que le noyau ETD29 était plus grand que le strict nécessaire.
Si quelqu'un sait adapter la formule, en connait une similaire (meilleure) ou une qui donne un aperçu du calcul de la taille du noyau plus précis, je le remercie de partager ses connaissances.

le nombre de tours pourra être modifié lors d'un essai préliminaire. Nous avons procédé ainsi pour fabriquer le premier prototype de transformateur. Le second prototype, avec l'enroulement secondaire à l'intérieur et les deux primaires à l'extérieur présentait de meilleures caractéristiques que le premier (voir [1]), de sorte que seul le second transformateur est décrit ici.

### Construction du transformateur

Pour que le transformateur soit facile à reproduire, nous avons pris le plus petit modèle *ETD* de corps de bobine (son noyau est plus gros que nécessaire pour sortir 60 W environ au rendement théorique de 100 %).
Pour assurer l'orientation correcte du transformateur sur le circuit imprimé, il faut couper les broches 8 et 10 telles que numérotées sur le corps de bobine *TDK/Epcos*. Examinez la sérigraphie du circuit imprimé pour être sûr de couper les bonnes broches. Sur le circuit imprimé, les plots du primaire et du secondaire sont bien séparés, tout comme les sorties du secondaire. À l'origine, il y a 13 broches sur un corps de bobine ETD29, il y a déjà un espace entre les broches 3 et 4. Une borne de l'enroulement secondaire a deux connexions possibles. Selon le nombre de tours et l'épaisseur du fil de cuivre utilisé, il faudra peut-être plus d'une couche et l'extrémité de l'enroulement sera soit plus proche du côté opposé, soit du même côté que le départ. C'est pourquoi les broches 4, 5, 6 et 9 sont connectées sur le circuit imprimé. La broche 7 est l'autre borne du secondaire. Toujours commencer le secondaire par celle-ci.
Les broches 1 et 13 sont les connexions du 1er enroulement primaire, les broches 2 et 12 celles du 2e. Sur le circuit imprimé, les noms sont placés à côté du transformateur. P1/PC pour le 1er enroulement primaire, P2/PC pour le 2e. PC signifie connexion commune des deux primaires. S1 et S2 sont les connexions du secondaire.
Pour réaliser le secondaire, à partir de la broche 7, enrouler 24 spires autour du corps de bobine (**fig. 2a**) et isoler avec du ruban adhésif. Ensuite, enrouler la deuxième couche de 13 spires pour atteindre 37 tours, le total de l'enroulement secondaire (**fig. 2b**). Isoler la 2e connexion avec un morceau de manchon en PTFE, voir la **fig. 2c**. *Il aurait été préférable d'isoler également la 1ère connexion du secondaire* (isolant absent sur les photos de TR2). Isoler ensuite la 2e couche secondaire avec du ruban adhésif. Ne pas chauffer trop longtemps les fils lors du raccordement aux broches du corps de bobine. Cela ferait fondre le plastique du corps. C'est pourquoi il faut bien étamer les extrémités des fils avant de les enrouler autour des broches !
Ensuite, le premier tour du premier primaire commence à la broche 1. Utiliser un couteau aiguisé pour retirer l'isolant de chaque fil et étamer déjà cette extrémité avant de l'enrouler sur une broche. Avec une pince, serrer l'extrémité du fil autour de la broche avant de faire deux tours sur le corps de bobine. Ajuster le fil, le faire un peu trop long pour atteindre la broche opposée, ne pas le connecter tout de suite. Faire de même avec deux fils de plus (trois fils sont connectés en parallèle pour le primaire, voir **fig. 2d**). Ensuite, retirer l'isolant des trois extrémités, les étamer et les serrer autour de la broche pour obtenir un aspect optimal de l'enroulement (bien plat et réparti uniformément, **fig. 2e**). Répéter cette procédure pour le deuxième primaire, en commençant par la broche 2, voir **fig. 2f**. La **fig. 3** montre le transformateur avec tous les enroulements terminés. N'oubliez pas de mettre le noyau !

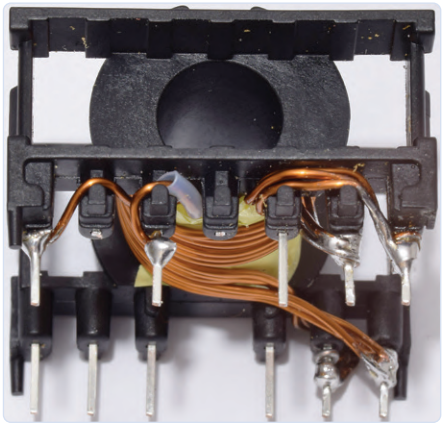


*Figure 3. Vue sur les broches du transformateur fini.*

## LISTE DES COMPOSANTS

### Résistances

R1 = 47 kΩ
R2 = 1,5 kΩ
R3 = 39 kΩ
R4 = 1 kΩ
R5,R6 = 22 Ω
R7,R8 = 0,15 Ω, 1 W
(*Multicomp Pro MCKNP01SJ015KA10*)
R9,R10 = 100 Ω, 1 W
(*Multicomp Pro MCKNP01SJ0101A10*)
R11 = 220 kΩ, 1 W, 350 V
P1 = 100 kΩ pot., réglage par le haut

### Condensateurs

C1 = 150 pF, 50 V, pas de 5 mm
C2 = 4,7 nF, 50 V, pas de 5 mm
C3,C4,C8 = 1 µF, 50 V, pas de 5 mm
C5,C6 = 220 pF, 100 V, pas de 5 mm
C7 = 220 µF, 25 V, 20 %, aluminium-polymère, pas de 3,5 mm, D 8 mm, ESR 15 mΩ
A750KS227M1EAAE015 Kemet
C9,C10 = 470 nF, 400 V, 5 %, polypropylène, pas de 15 mm (19×9 mm max.)

### Inductances / Transformateurs

TR1 = 2× noyau ETD29, N97, sans entrefer, B66358G0000X197 TDK/Epcos
TR1 = corps de bobine, ETD29, B66359W1013T001, TDK/Epcos
TR1 = 2× clips, B66359S2000X000, TDK/Epcos
L1 = 2,2 µH, 10 %, 5,6 ARMS, 21 mΩ, pas de 5 mm, D 8,5 mm max, RLB0913-2R2K Bourrelets
L2 = self d'arrêt mode commun 500 µH, 1 A, 0,3 Ω, SU9V-10005 Kemet

### Semi-conducteurs

D1,D2,D3,D4 = STTH1R04QRL
LED = 3 mm, verte
T1,T2 = TK30A06N1, TO-220SIS
IC1 = UCC28089D, SOIC-8

### Autres

K1 = bornier à 2 voies, pas de 5,08 mm
K2 = bornier à 2 voies, pas de 7,68 mm
F1 = fusible à cartouche, 5×20 mm, 5 A temporisé, avec porte-fusible idoine, 500 V/10 A et couvercle
TR1 = ruban isolant électrique, film polyester, 3M 1350 12 MM
TR1 = fil de cuivre, 0,71 mm pour tous les enroulements, 49 tours au total, 3 m
TR1 = gaine de protection en PTFE, diam. intérieur 1,02 mm min. Pro Power STFE 18 CLR, 10 cm
Circuit imprimé 200307-1 v1.1

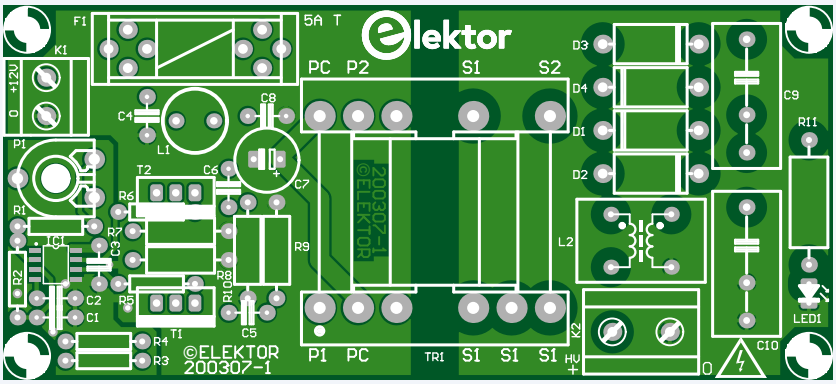
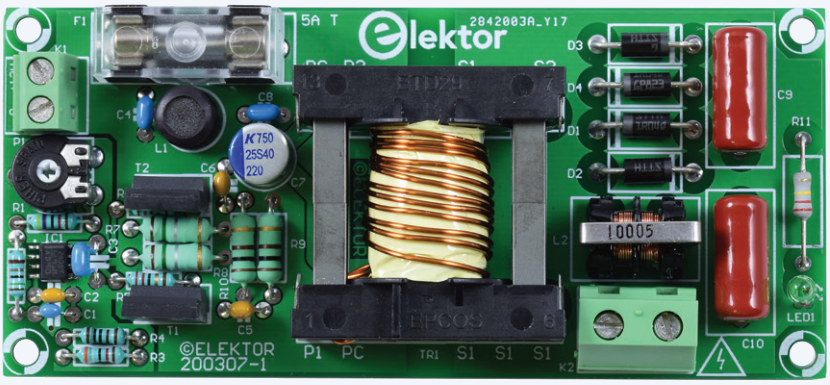



*Figure 4. Circuit imprimé.*




## Fabrication du convertisseur CC/CC

Les fichiers Gerber et de perçage pour la fabrication du circuit imprimé (**fig. 4**) sont disponibles à l'adresse [3]. Commandez le circuit imprimé auprès de votre fournisseur préféré. Les photos du prototype montrent sa version 1.0, la version 1.1 apporte une petite correction en sérigraphiant « +HV » et « 0 » à côté de la borne à vis K2.

Commencer par souder le petit CMS IC1 (SO-8) sur le circuit imprimé. Pour gagner de la place, les résistances d'1 W (R7-R10) côté primaire sont miniaturisées (corps 10 mm × 3,5 mm). Les modèles de stocks anciens ou achetés au petit bonheur pourraient ne pas convenir. Utilisez la référence donnée dans la liste des composants. C7 doit être du type aluminium polymère, n'utilisez pas une version électrolytique ordinaire, il risque fort de brûler ! Le modèle de la liste des composants supporte un courant d'ondulation de 4,42 A à 100 kHz et a une très faible résistance série de 15 mΩ (100 kHz/20 °C). Le montage des autres composants est assez simple. La résistance 1 W de la LED (R11) indiquant la présence de la haute tension (HV) doit supporter au moins 350 V. Sa taille peut être plus grande. Notez que le circuit imprimé ne répond pas à la classe II d'isolation secteur, mais l'espacement des composants et pistes est accru, en particulier pour les pistes côté HT/sortie. Le transformateur (s'il est correctement construit) fournit la haute tension avec une très bonne isolation galvanique. Pour une commutation à 150 kHz, régler P1 à mi-course.

**Pour utiliser ce convertisseur CC/CC, assurez-vous qu'il n'y a pas d'objets électriquement conducteurs à proximité des composants sous haute tension. Plus la distance est élevée, mieux c'est !**

La tension de sortie peut être modifiée en ajustant le nombre de tours du secondaire, mais aussi en changeant le nombre de tours des primaires, si la fréquence de commutation correcte est respectée. Changer C1 (et R2) pour une fréquence d'oscillation différente modifie le temps mort. Avant de changer les valeurs, regardez bien les formules de la fiche technique de l'UCC28089.

Concevoir un convertisseur CC/CC *push-pull* est assez compliqué, le calcul et la construction du transformateur « sur mesure » nécessaire le sont plus encore. Comme indiqué plus haut, pour en savoir plus, voir [1] ; vous


## PRODUITS

- **M. van der Veen, *Vanderveen Trans Tube Amplifiers*, Elektor (e-book)**
  www.elektor.fr/vanderveen-trans-tube-amplifiers-e-book
- **M. van der Veen, *Designing Tube Amplifiers*, Elektor (e-book)**
  www.elektor.fr/designing-tube-amplifiers-ebook

pourrez y partager avec d'autres lecteurs vos questions, commentaires, suggestions et expériences concernant ces convertisseurs ou la conception de transformateurs.

(200583-04)

**Des questions, des commentaires ?**
Envoyez un courriel à l'auteur (ton.giesberts@elektor.com).

**Contributeurs**
Idée, conception, auteur : **Ton Giesberts**
Rédaction : **Luc Lemmens**
Illustrations : **Ton Giesberts, Patrick Wielders**
Mise en page : **Giel Dols**
Traduction : **Yves Georges**

## LIENS

[1] **La page Elektor Labs de ce projet** : www.elektormagazine.fr/labs/12v-200v-dc-dc-converter-for-valve-amplifiers
[2] **Fiche technique du corps de bobine TDK** : https://product.tdk.com/info/en/documents/data_sheet/80/db/fer/etd_29_16_10.pdf
[3] **Téléchargements de ce projet** : www.elektormagazine.fr/200583-04

# traqueur de chaleur

## Caméra thermique Seek Shot Pro

**Alfred Rosenkränzer** (Allemagne)

Nous avons récemment évalué une épatante caméra thermique pour tester son utilité pour notre labo. Nous avons constaté que cette caméra de poche peut rapidement détecter les problèmes dans les applications courantes. Comme elle est petite, son écran est-il vraiment pratique ? Voyons cela de plus près.


*Figure 1. Vue de face montrant la lentille IR, la lentille normale et la source lumineuse à LED.*

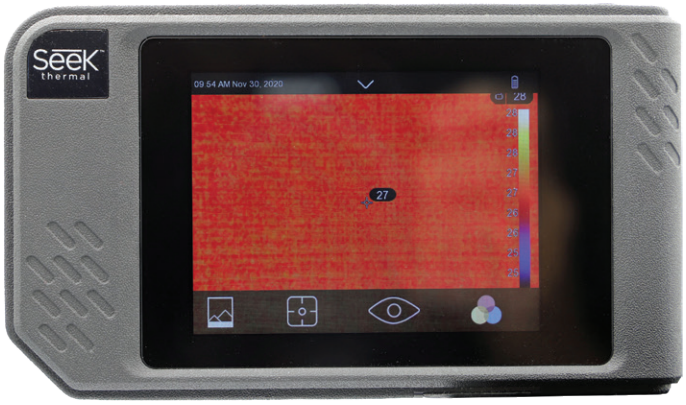



*Figure 2. Vue arrière montrant l'affichage actif.*

Le fabricant Seek Thermal, basé à Santa Barbara en Californie, propose dans son catalogue une gamme de produits d'imagerie IR professionnelle. La caméra thermique Seek Shot Pro est arrivée dans une solide boîte en carton de 173×115×44 mm. Elle était accompagnée d'un câble USB, d'une bandoulière et d'une fiche d'information multilingue.

La caméra mesure 140×80×25 mm et son format ressemble plus à celui d'un smartphone qu'à celui d'un appareil photo de 35 mm. Le boîtier est doté d'une surface caoutchoutée agréable à prendre en main, et qui réduit le risque de la faire tomber tout en offrant un peu de protection mécanique. Le boîtier ne comporte que deux boutons-poussoirs mécaniques : un bouton marche/arrêt et un bouton de déclenchement de l'obturateur pour la prise de photos ou de vidéos. Tous les autres réglages se font sur l'écran tactile.

### Prise de contact

Le port USB de la caméra est caché sous un couvercle rabattable en dessous. Il sert à charger la batterie intégrée et à transférer les images vers un PC. La mémoire interne ne peut pas être étendue, car il n'y a pas d'emplacement pour une carte SD. La plus grande différence avec un smartphone est le filetage femelle de 0,25 pouce qui permet de fixer la caméra sur un trépied standard.

La **figure 1** montre une vue de face. Sur la face avant de la caméra, dans le coin supérieur droit, se trouve le grand objectif d'imagerie IR. En dessous, la petite lentille de lumière visible. À gauche de la lentille IR se trouve une LED utilisée pour fournir un éclairage supplémentaire au flash dans des conditions de faible luminosité. On peut aussi l'allumer indépendamment pour servir de lampe de poche si vous vous perdez dans l'obscurité.

Il existe deux versions dans la gamme de caméras thermiques Seek Shot. La version de base produit une image IR à basse résolution de 206×156 pixels, tandis que la version Pro offre une image de 320×240 pixels. Cela peut sembler modeste par rapport à la densité de pixels que l'on attendrait normalement d'une caméra de smartphone, mais les matrices d'imagerie thermique IR ont une résolution beaucoup plus faible. Pour la plupart des applications, une image thermique IR de plus haute résolution n'apporterait pas de réels avantages.

La version Pro offre un champ de vision de 57°, ce qui correspond à une focale d'environ 40 mm pour un format d'appareil photo de 35 mm. Ce champ est un peu plus large que le champ de vision de l'œil. La version économique de la caméra a un champ de vision de 36° (≈ 66 mm pour le format 35 mm), ce qui lui donne davantage la caractéristique d'un téléobjectif. La seule différence visible entre les deux caméras est la couleur du liseré autour de l'objectif IR. Dans la version économique,

il est gris, tandis que dans la version Pro, il est rouge. Comme vous pouvez le voir dans la figure 1, je teste ici la version Pro.

## Premières impressions

Tout d'abord, il faut brancher le câble USB pour charger la batterie de la caméra. Profitez-en pour vous familiariser avec le mode d'emploi et visionner quelques vidéos instructives sur le site web du fabricant [1] – c'est du moins ce que j'ai fait. Cela vaut la peine d'apprivoiser la caméra et ses caractéristiques avant de l'utiliser sérieusement.

La caméra est principalement un appareil « cadrer-déclencher ». La plupart des réglages sont automatiques, on ne peut donc pas modifier l'ouverture ou le temps d'exposition, et comme les deux objectifs sont à mise au point fixe, il n'y a pas de réglage de mise au point. Selon les spécifications, la distance focale minimale est d'environ 10 à 15 cm. Cela s'applique aux deux objectifs, de sorte que la superposition d'images « Seek Fusion » fonctionne également à cette distance.

Pour moi, l'une des caractéristiques les plus utiles du système est sa capacité à effectuer un post-traitement d'une image pour la stocker ensuite sous forme de copie dans la galerie. Cela vous permet de créer plusieurs versions différentes d'une même image et de stocker les résultats selon les besoins. Pour capturer un seul événement thermique ou de surcharge, vous pouvez prendre une série de photos et étudier la séquence d'images plus tard.

## L'écran

Après la mise sous tension, il faut quelques secondes pour que l'heure et la date s'affichent à gauche dans la barre supérieure de l'écran (voir **fig. 2**). L'état de charge de la batterie est indiqué sur la droite. Une flèche pointant vers le bas est affichée au milieu de cette barre. Elle donne accès au menu où vous pouvez régler des choses comme la luminosité de l'écran ou les unités (*Celsius*, *Fahrenheit* ou *Kelvin*) de la température affichée. Vous pouvez également régler ici l'émissivité des surfaces à photographier. C'est important pour que les photos montrent des températures correctes.

On peut utiliser l'icône de lampe de poche sur la droite pour allumer la LED blanche frontale et une icône de flash activera la fonction flash de la LED. On peut établir la liaison wifi ou consulter une description de l'appareil photo à l'aide de la fonction d'aide intégrée. Une icône classique de roue dentée vous permet d'accéder au menu des paramètres avancés où vous pouvez régler l'heure, la date, la langue du menu et d'autres caractéristiques de configuration. Au retour du menu, l'affichage ressemble maintenant à la figure 2.

En bas de l'écran se trouvent quatre icônes. À gauche, l'icône de

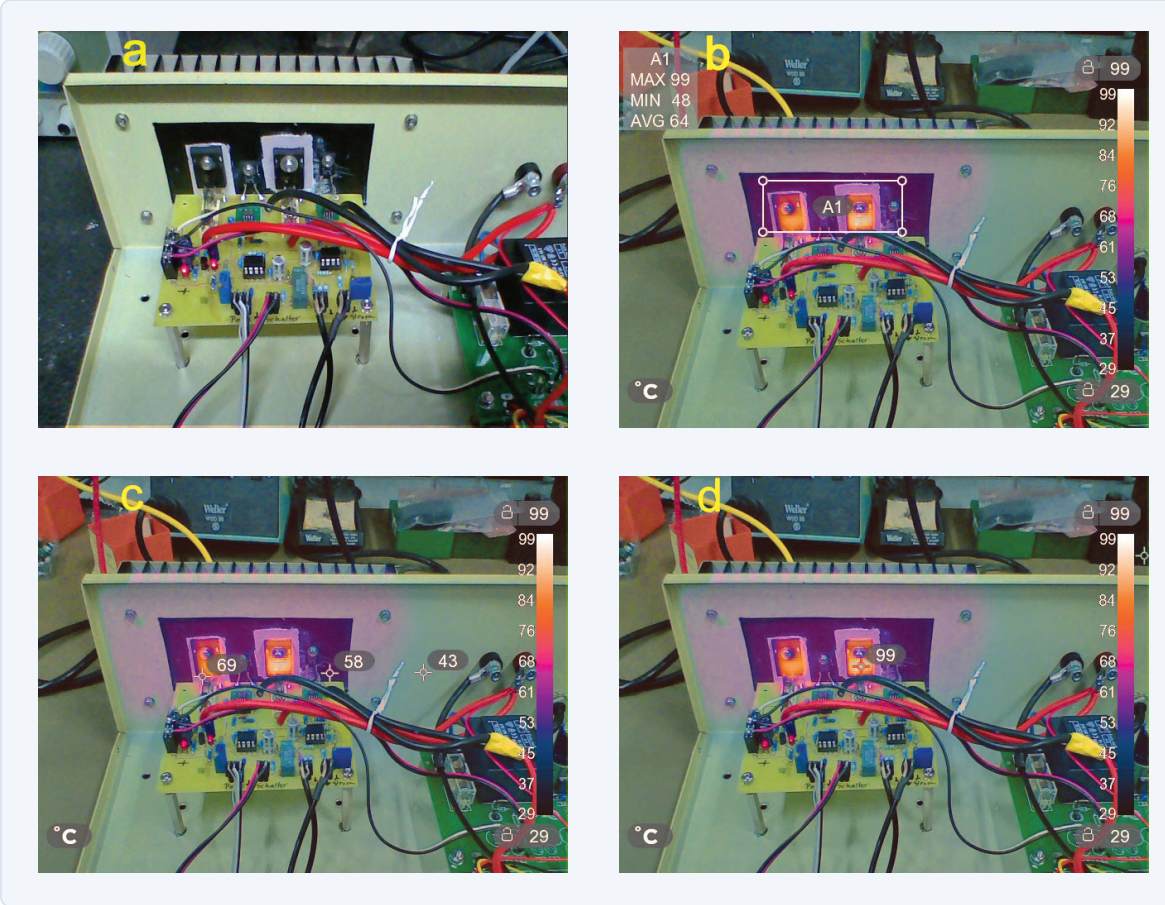



*Figure 3. Le prototype de charge électronique sous forme d'image normale (3a), d'image mixte (3b), d'image mixte avec trois points de mesure (3c) et d'image mixte montrant la température maximale (3d).*

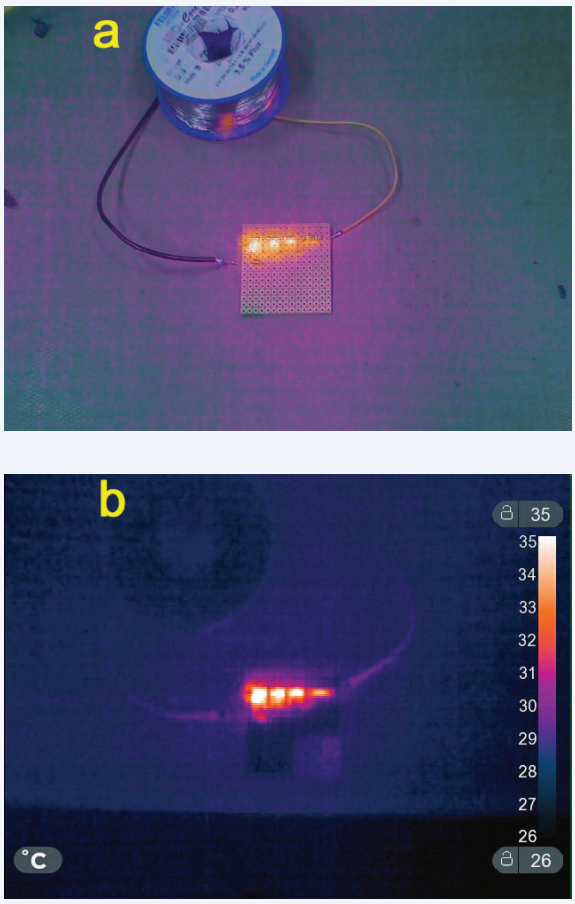



*Figure 4. (4a) montre le circuit sur la platine d'expérimentation avec une vue fusionnée, tandis que (4b) est l'image thermique seule.*

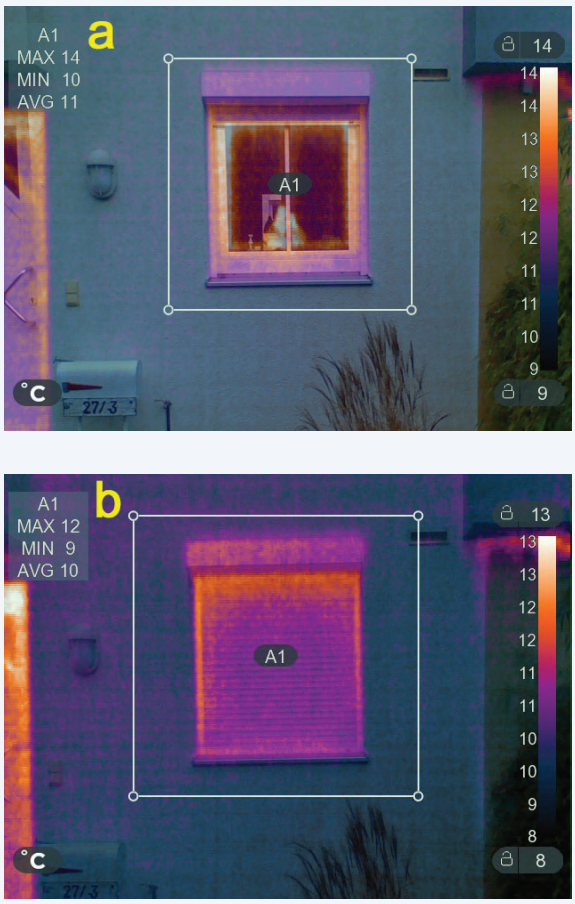



*Figure 5. Il fait froid dehors avec le store levé (5a) puis avec le store baissé (5b).*

paysage vous amène à la galerie d'images contenant des photos déjà stockées. Celles-ci peuvent être rappelées et éditées, comme nous le verrons plus loin. La deuxième icône en partant de la gauche ouvre un autre menu dans lequel on peut sélectionner les différentes options de mesure de la température. Un clic sur l'icône en croix à gauche indique la température de la surface affichée au centre de l'écran actif. En déplaçant la caméra, on peut lire la température, mais il y a un peu de retard dans l'affichage de la température, il faut donc avoir la main stable. L'icône à trois points indique la température du spot à trois endroits de l'image. L'icône montrant un carré avec des points aux coins crée jusqu'à trois zones d'intérêt redimensionnables et repositionnables sur la scène active. La caméra peut alors afficher les températures maximale, minimale et moyenne pour chacune de ces zones. L'icône avec les signes plus et moins à l'extrême droite active la mesure du point le plus chaud et du plus froid de l'image. Regardez les vidéos de formation pour voir comment cela fonctionne. L'icône de l'œil ouvre un menu permettant de changer de vue. Si l'icône 'œil' représentée par une ligne continue est sélectionnée, l'image visible normale est affichée. L'icône 'œil' à droite avec les lignes brisées sert à afficher l'image thermique. L'image du milieu montre les deux yeux superposés pour indiquer la vue de fusion. Un clic ouvre deux commandes. Celle de gauche ajuste le mélange d'images entre l'image visible et l'image thermique. Avec celle de droite, les deux images peuvent être décalées verticalement pour compenser l'erreur de parallaxe dans le plan vertical qui augmente à mesure que les deux objectifs de la caméra se rapprochent de l'objet. En appuyant sur l'icône de couleur, on fait apparaître la palette de couleurs qui permet d'appliquer différents profils de couleurs à l'image active. En fonction de la plage de température dans l'image, une palette spécifique peut être mieux adaptée pour distinguer les différences de température.

## Images d'essai

L'échelle de température est d'habitude « à gamme automatique », mais elle peut être fixée selon les besoins afin que les images puissent être comparées plus facilement. La **figure 3a** montre l'image visible normale d'une charge électronique avec son couvercle retiré. Dans la **figure 3b,** nous pouvons voir une image composée de l'image visible avec son image thermique superposée – c'est-à-dire la vue « Seek Fusion ». La température maximale enregistrée dans la zone rectangulaire sélectionnée est de 99 °C. Le long du côté droit de l'image, nous pouvons voir l'échelle de couleurs de la température. La **figure 3c** montre la température de trois positions de mesure sélectionnées sur les surfaces du dissipateur thermique et de l'enceinte. Dans la **figure 3d,** on peut montrer où la température maximale est enregistrée dans l'image.
Pour un autre exemple (**fig. 4a**), j'ai construit un petit circuit de test sur une platine d'expérimentation, composé de cinq résistances de 100 Ω

connectées en série. La première résistance est un modèle traversant conventionnel tandis que les quatre autres sont des CMS 1206, 0805, 0603 et 0402. On fait circuler un courant de 10 mA dans la chaîne. Les différences de température entre chaque composant sont assez faibles, mais l'image thermique les rend plus évidentes. L'étroite plage de température utilisée dans cette image est visible sur la **figure 4b**.
Il va sans dire qu'une caméra thermique n'est pas seulement une bonne aide pour les ingénieurs en électronique, c'est aussi une méthode très pratique pour identifier rapidement les zones de dissipation de chaleur et de défaillance de l'isolation dans un environnement domestique. Pour la tester, je suis allé dehors et j'ai pris quelques photos. Ici, en Europe, les fenêtres sont souvent équipées de stores métalliques extérieurs. J'ai pris quelques photos pour montrer l'effet de ces stores sur la réduction des pertes de chaleur. La **figure 5a** montre la fenêtre avec son volet roulant ouvert alors que sur la **figure 5b,** il est complètement fermé. Vous pouvez voir une différence de température maximale de 2 °C.
Il va sans dire qu'avec un appareil photo en main, il faut absolument faire un selfie ou, dans ce cas, un thermoselfie. La résolution et la précision de ce modèle particulier de caméra ne permettent pas de détecter un état fiévreux en mesurant la température de surface de la peau (Seek Thermal Inc. produit des appareils spécialisés pour de telles applications). À la vue de la **figure 6**, je dois être immédiatement alité avec une boisson fraîche. La précision de la température est de ±2° et dépend fortement de la valeur de l'émissivité de surface qui peut être réglée dans le menu.
J'ai pris toutes ces photos d'affilée, puis je les ai éditées dans la galerie et je les ai ensuite sauvegardées sous forme de copies. Pour ce faire, ouvrez la galerie en appuyant sur l'icône de paysage, sélectionnez la bonne image avec les touches fléchées et activez l'édition. Lorsque vous avez terminé, sauvegardez l'image en tant que copie. De cette façon, l'original est conservé pour d'autres traitements. Tout cela, ainsi que la diffusion d'images en direct, est également possible à distance par wifi en utilisant l'application correspondante sur un téléphone portable ou une tablette, mais je ne l'ai pas testée. Une application pour PC, comme celle qui est disponible pour les autres caméras thermiques, serait également très souhaitable.
Faire de l'édition sur le petit écran avec mes gros doigts s'est avéré pénible et demande beaucoup de concentration et de patience. J'utilise principalement cette caméra infrarouge pour vérifier rapidement les circuits électroniques et je l'ai trouvée remarquablement utile.

## En résumé

Une caméra thermique au labo est un excellent outil pour déboguer les problèmes des circuits prototypes et détecter les conditions de surcharge. C'est une approche beaucoup plus pratique que, par exemple, l'utilisation d'une sonde thermométrique ou d'un doigt mouillé. Dans certains cas, elle peut fournir une indication rapide sur le fait qu'un composant de votre dernier projet approche sa limite de dissipation et peut même vous donner assez de temps pour débrancher la prise et éviter que la fumée magique ne s'échappe. Il est également extrêmement utile et pratique de pouvoir prendre et enregistrer des images thermiques pour analyse et évaluation ultérieures, le cas échéant.
La vue « Seek Fusion » qui combine des images normales et thermiques est une aide réelle pour identifier une source de chaleur indiquant un composant défaillant ou pour montrer une zone d'où la chaleur s'échappe d'un bâtiment ou d'une enceinte isolée. Pour les applications générales de laboratoire, la résolution des images thermiques n'a pas besoin d'être particulièrement élevée. En gardant cela à l'esprit, je pense aussi (bien que je n'aie pas eu l'occasion de la tester) que la résolution plus faible de la version économique de la caméra Seek Shot serait plus que suffisante pour la plupart des travaux de laboratoire. Pour ce banc d'essai, j'ai choisi la version Pro, meilleure et plus coûteuse. Pour un peu plus de 800 €, la Seek Shot Pro supporte la comparaison avec une caméra professionnelle similaire de Flir, qui est environ dix fois plus cher (et qui a maintenant plus de 10 ans). Dans l'ensemble, je l'ai trouvé assez impressionnante et particulièrement pratique pour une utilisation générale en laboratoire !

(200654-04)


*Figure 6 : Un « thermoselfie ». Hé, j'ai de la température !*

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

**Contributeurs**
Révision et texte : **Alfred Rosenkränzer**
Rédaction : **Thomas Scherer**
Mise en page : **Giel Dols**
Traduction : **Denis Lafourcade**

**PRODUIT**

› **Seek Shot Pro**
www.elektor.fr/seek-shot-pro-thermal-imaging-camera-320x240

**LIEN**

[1] **Page web de Seek Shot :** www.thermal.com/seekshot-series.html

# programmation orientée objet

## Une brève introduction avec le C++

**Roland Stiglmayr** (Allemagne)

Vous voulez écrire de puissants programmes orientés objets ? Pour commencer, il vous faut familiariser avec les avantages de la programmation orientée objet par rapport à la programmation procédurale. D'abord un peu de théorie avant de passer aux exemples pratiques.

La programmation orientée objet (POO) existe depuis plus de 40 ans. Parmi ses nombreux avantages, citons ses procédures d'assurance qualité élaborées, une maintenance logicielle plus facile et une excellente structuration qui contribue à simplifier le travail en équipe dans le développement de logiciels. Ces dernières années, la POO a aussi trouvé son emploi dans le domaine du logiciel embarqué. L'EDI Arduino permet également son utilisation ; c'est le bon moment pour approfondir le sujet. Cet article ne donnera qu'un aperçu des principales caractéristiques de la POO, mais ce sera suffisant pour vous apporter une bonne base de connaissances et assez d'expérience pour vous inviter à écrire vos propres programmes orientés objet.

En matière de code informatique, il y a deux approches différentes : la programmation procédurale traditionnelle et la programmation

**Listage 1. Déclaration de la classe Virt_M (leçon 1).**

```
class Virt_M
  {
    private:                          //les membres suivants sont seulement
                                      //accessibles à l'intérieur de la classe
      float voltage;                  //tension virtuelle/ V
      float current;                  //courant virtuel/ A
      float p_result;                 //puissance calculée/ W
      float r_result;                 //résistance calculée/ Ohm.

    public:                           //les membres suivants sont également
                                      //accessibles depuis l'extérieur de la classe
      String s= ("public: Demo attribute s, type String");  //for demo only

  // Déclaration du constructeur, appelé quand un objet
  // de cette classe est créé

      Virt_M (float v, float c);      //déclaration du constructeur

  // Déclaration des méthodes/ fonctions de la classe Virt_M

      float get_P ();                    //méthode de lecture de la puissance
      float get_R ();                    //méthode de lecture de la résistance
      float get_Voltage ();              //mméthode de lecture de la tension
      float get_Current ();              //méthode de lecture du courant
      void prep_Meas (float v, float c);  //méthode de simulation de la mesure

    private:                          //seulement accessible de l'intérieur de la classe

      void set_Voltage (float v);     //méthode d'écriture de la tension
      void set_Current (float c);     //méthode d'écriture du courant

  };                                  //fin de la déclaration de la classe
```

orientée objet. La première est celle que nous avons toujours utilisée (par ex. pour la programmation en assembleur ou en C standard). Le programme est décomposé en petits modules (des procédures ou des fonctions) et exécuté de manière séquentielle. Les modules communiquent en utilisant des données communes, souvent déclarées globales, et s'en servent pour échanger les résultats de leurs traitements. En revanche, la POO enferme les données et les fonctions qui les utilisent dans des *objets* et en réserve l'accès à des fonctions spécifiques, l'*interface* de l'objet. Cette *encapsulation* protège les données contre tout accès inapproprié.

Avec la POO, les unités fonctionnelles sont constituées de membres logiquement liés entre eux, comme les données et les fonctions. Les fonctions sont appelées des *méthodes*. Prenez un enregistreur de données avec son unité d'acquisition de données équipée de nombreux canaux de mesure, son unité de traitement et son interface utilisateur. Chaque unité pourrait représenter une unité fonctionnelle. Une unité fonctionnelle peut hériter des propriétés d'autres unités fonctionnelles, de sorte qu'une structure modulaire peut être constituée à partir d'unités de base. La protection des données et des méthodes est assurée par le principe de l'encapsulation.

Une unité fonctionnelle peut être considérée comme un nouveau type de données appelé *classe*. La classe représente un « plan de construction » à partir duquel sont créées des entités appelées *objets* ou *instances* de cette classe. Attention à la différence entre la classe et l'objet ! Un nombre quelconque d'objets peut être *instancié* à partir d'une classe. Chaque personne travaillant dans une entreprise peut être considérée comme un objet créé à partir d'une seule et même classe.

Assez de théorie. Passons à la pratique ! Comme toujours, le plus simple est de commencer par des exemples tout faits que vous pouvez manipuler et modifier pour essayer vos propres idées. Le faire dans un environnement de développement intégré (EDI) familier fait gagner du temps et diminue la pente de la courbe d'apprentissage. Nous avons choisi l'EDI Arduino, populaire et gratuit, pour traiter les exemples proposés. Il intègre un compilateur C++ complet qui convient parfaitement à la POO. Nos exemples de programmes, appelés *croquis* dans le monde Arduino, répartis en sept « leçons », sont téléchargeables à partir de la page du projet [1] de cet article. Une fois téléchargés, il faut les décompresser et les copier dans le dossier *Sketch* de l'EDI Arduino ; l'environnement est alors prêt. Les exemples sont tous basés sur un appareil de mesure virtuel, sans réalité physique. Simples illustrations des processus impliqués dans le transfert d'informations, ils ne sont pas conçus pour gérer un appareil de mesure « réel » comprenant du matériel, mais pour présenter les éléments de base de la POO au moyen d'un logiciel facile à comprendre.

## Une première classe

Pour commencer, nous apprendrons la création d'une classe, la signification du spécificateur d'accès, les bases de l'encapsulation et la création d'instances. Pour ce faire, il faut charger le croquis de la *leçon 1* dans l'IDE, le compiler et le télécharger sur la carte. Avant de visualiser la sortie produite par le programme sur le moniteur série de l'EDI Arduino, examinons d'abord son code source (**listage 1**).

Sur la première ligne, la déclaration `classe` est suivie du nom `Virt_M`. Cela indique au compilateur qu'une classe portant le nom `Virt_M` va être définie. Les lignes suivantes déclarent les variables internes (ou *attributs*) et les méthodes appartenant à la classe. Le contrôle d'accès est précisé en attribuant aux membres, aux attributs et aux méthodes de la classe l'un des spécificateurs d'accès `private` ou `public`. Les membres ayant un accès `private` ne sont accessibles qu'à l'intérieur de la classe. Tout membre ayant un accès `public` peut être accessible depuis l'extérieur de la classe. Nous verrons plus tard ce que cela signifie.

La première méthode, qui porte le même nom que la classe et ne retourne aucune valeur, est le *constructeur*. Le constructeur est toujours appelé lorsqu'un objet du type de cette classe est créé. Si des valeurs sont passées au constructeur, elles servent à initialiser les attributs de l'objet.

Parmi les autres méthodes, celles qui commencent par `get_` servent à lire les attributs privés de la classe ; celles qui commencent par `prep_` sont utilisées pour modifier ces attributs. Ces méthodes sont publiques et constituent l'interface de la classe. Les méthodes `set_` spécifiées comme `private` ne peuvent être appelées qu'au sein de la classe. La chaîne `string s` est utilisée pour démontrer le spécificateur d'accès `public`. L'accolade fermante suivie d'un point-virgule termine la déclaration de la classe.

Viennent ensuite les définitions des méthodes qui ont été déclarées ci-dessus. Elles sont identiques aux définitions des fonctions du C standard, sauf pour la syntaxe utilisée, qui est la suivante :

```
nom de la classe::nom de la méthode(liste des paramètres)
```

La référence à la classe est effectuée avec l'opérateur de résolution de portée, le double deux-points.

Notre première classe est maintenant prête à servir à créer des objets. Comme pour la déclaration de fonction, le nom de la classe est composé du nom de la classe suivi du nom de l'objet et (s'il y a lieu) d'une liste de valeurs initiales (**listage 2**). Lors de la création (ou instanciation) d'un objet, le constructeur de la classe est appelé et les attributs de cet objet sont créés et initialisés. Il faut bien avoir à l'esprit que chaque objet possède sa propre copie des attributs.

Quelle est l'activité d'un objet de cette classe ? Dans notre exemple, il simule l'acquisition de valeurs mesurées de tension et de courant et met ces valeurs à la disposition des méthodes correspondantes avec les valeurs de résistance et de puissance qui en sont déduites. L'appel d'une méthode publique est fait ainsi :

```
Nom_objet.nom_méthode();
```

Les attributs protégés par `private` ne sont accessibles qu'en utilisant les méthodes prévues à cet effet. Toute tentative d'accès direct est refusée par le compilateur avec un message d'erreur, comme on peut le vérifier en supprimant le marqueur de commentaire `//` au début d'une ligne de code du **listage 3**.

Comme autre exemple de contrôle d'accès, la chaîne `string s` a été délibérément déclarée `public`. Les attributs déclarés de cette manière sont accessibles directement par `nom_objet.nom_attribut`. Mais attention ! Ils ne bénéficient d'aucune protection contre une utilisation incorrecte par le programme utilisateur.

Le programme principal modifie les valeurs lues en utilisant la méthode `prep_Meas`, qui appelle les méthodes internes `set_`.

Les exemples de programmes contiennent des commentaires utiles et de nombreuses informations qui devraient contribuer à une meilleure compréhension. Après le démarrage de l'application, la routine de configuration affiche la liste des attributs des objets sur le moniteur série (**fig. 1**).

**Listage 2. Création des instances (leçon 1).**

```
// Création des instances / objets de type Virt_M
// Initialisation des attributs
//-----------------------------------------

   Virt_M My_M1 (220.0,0.5);  //crée l'object My_M1 en appelant
                              //le constructeur de Virt_M

   Virt_M My_M2 (10.0,1.0);   //crée l'object My_M2 en appelant
                              //le constructeur de Virt_M
```

**Listage 3. Test du contrôle d'accès (leçon 1).**

```
// Pour tester le contrôle d'accès, effacer l'instruction de commentaire "//"
//  ------------------------------------------------------------------

//  My_M1.voltage= 0;               //erreur > l'attribut est privé
//  My_M1.set_Voltage(0);           //erreur > la méthode est privée
  String str= (My_M1.s);            //permis > l'attribut s est public
  Serial.println (My_M1.s);
```

## Une classe comme bibliothèque de programmes

Le concept de bibliothèque n'est pas nouveau dans le monde de la programmation informatique. Vous en avez sûrement déjà utilisé si vous avez une quelconque expérience du codage en C standard. Elles sont d'usage encore plus intensif en POO.

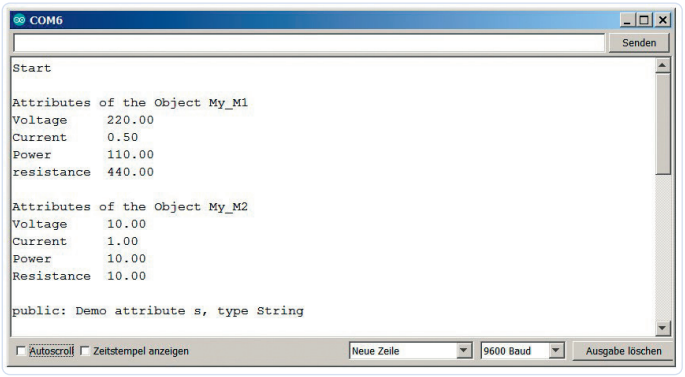



*Figure 1. Sortie de la routine de configuration de la leçon 1.*

La raison en est que le développeur a besoin de connaître les détails du code source d'une classe pour pouvoir l'utiliser. L'utilisation de bibliothèques encourage la réutilisation de parties de programme et promeut ainsi le concept de modularité dans la conception du programme.

Un exemple est donné dans la *leçon 2*, qui réutilise la classe du premier croquis comme bibliothèque. Un élément de bibliothèque se compose de deux fichiers ; le fichier d'en-tête (.h) contient la déclaration de la classe et le fichier source (.cpp) la définition des méthodes de la classe. Bien que cela ne soit pas strictement nécessaire, il vaut mieux utiliser le même nom pour les deux fichiers. Pour mettre notre classe `Virt_M` sous la forme d'une bibliothèque, la partie déclaration doit apparaître dans le fichier d'en-tête et la partie définition dans le fichier source. Pendant le développement, il est recommandé d'enregistrer les fichiers de la bibliothèque dans le même répertoire que le programme utilisateur. L'utilisation d'onglets dans l'EDI Arduino nous aide à cet égard, mais nous y reviendrons plus tard.

Au début d'un fichier d'en-tête, il est d'usage de vérifier s'il a déjà été inclus par le compilateur, ce qui est le cas si un autre fichier

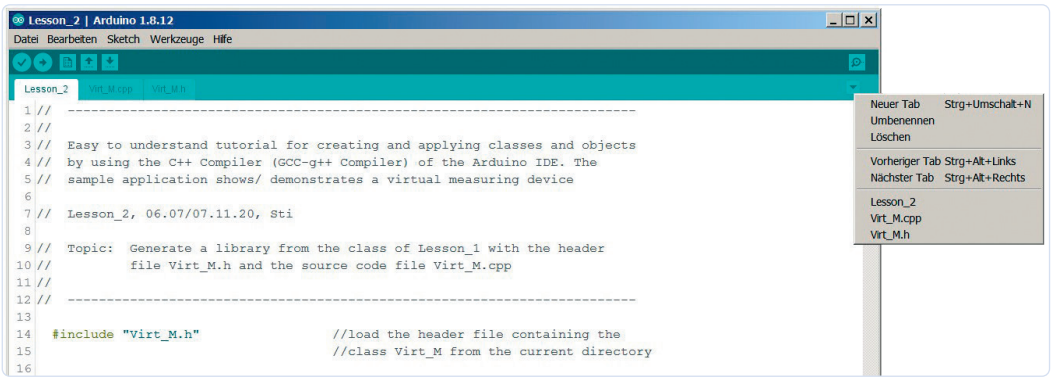



*Figure 2. La fonction de gestion des onglets (leçon 2).*

d'en-tête l'a déjà inclus. Des membres du même nom seraient alors déclarés plusieurs fois. La syntaxe pour que le préprocesseur le vérifie est la suivante :

```
  #ifndef Virt_M_h
// le symbole Virt_M_h est utilisé pour tester
//si le fichier d'en-tête a déjà été inclus
  #définir Virt_M_h // si le symbole n'existe pas
   encore, on le définit
    classe ....// déclaration de la classe
      { ....}; // fin de la déclaration
  #endif // fin de la compilation conditionnelle
```

L'instruction `#include "Virt_M.h"` dans le fichier source est utilisée pour inclure le fichier d'en-tête. Dans notre exemple, `Arduino.h` est inclus par défaut pour donner au compilateur l'accès aux bibliothèques standard. Avec cela, notre bibliothèque est terminée. Maintenant, pour utiliser la classe `Virt_M` dans un programme, il suffit de charger le fichier d'en-tête avec `#include "Virt_M.h"`.
Les bibliothèques Arduino fournissent souvent une instance de leur bibliothèque de classe. Par exemple, dans la bibliothèque Wire, *Wire* est un objet de la classe *TwoWire*. L'objet est inclus en utilisant `#include <Wire.h>`.

Une fois la programmation terminée, nous pouvons sauvegarder les fichiers de notre bibliothèque dans leur propre répertoire. Pour cela, Arduino crée le répertoire *libraries* dans le carnet de croquis. Nous pouvons y créer un autre répertoire dans lequel nous sauvegarderons nos fichiers. Cela fonctionne également via l'IDE avec *Croquis -> Inclure une bibliothèque -> Ajouter la bibliothèque .ZIP*. Comme déjà mentionné, la fonction de gestion des onglets de l'IDE Arduino est très utile pour gérer plusieurs fichiers appartenant à un même projet. Elle est activée par le petit triangle en haut à droite de la fenêtre (**fig. 2**).

## L'héritage - un concept fondamental de la POO

Une fois la POO abordée, le concept d'héritage n'est pas loin. Examinons-en les avantages. L'héritage consiste à mettre les attributs et les méthodes d'une classe de base à la disposition d'une classe héritière. La classe héritière peut être appelée classe enfant, classe dérivée ou sous-classe. La classe de base est également appelée classe parente ou ancêtre ou superclasse. La classe enfant ajoute d'autres propriétés à celles dont elle hérite et transmet le tout à sa propre descendance. Cela signifie que de nombreuses classes dérivées différentes peuvent être issues d'une classe parente, de sorte que la structure résultante n'est pas nécessairement linéaire, mais peut être arborescente. Une relation composée de nombreuses branches !
La *leçon 3* montre comment nous pouvons utiliser l'héritage dans notre tâche. Pour rendre l'exemple un peu plus significatif, nous utilisons des adresses multiplexées pour sélectionner les différents canaux de mesure et les lier aux objets créés. `Virt_M` est la classe de base. Ici, vous pouvez voir qu'il y a trois constructeurs avec des nombres différents de paramètres. Le C++ permet généralement cette *surcharge* des définitions de fonctions (c'est-à-dire des fonctions utilisant le même nom avec des nombres différents et/ou des types de paramètres différents). En POO, cela est connu sous le nom de *polymorphisme*. Notez le niveau d'accès `protected` qui indique que les méthodes, les classes et les autres membres sont accessibles à la classe héritière. Dans notre exemple, toutes les classes héritières ont accès à la méthode `set_Mux`.

Ici, la déclaration de la méthode contient aussi sa définition. Cela est possible, mais n'a d'intérêt que pour des fonctions très courtes. La première classe enfant est `Volt_M`, qui hérite de la classe `Virt_M`. La syntaxe est la suivante :

```
classe Volt_M : public Virt_M
// la classe Volt_M hérite de la classe Virt_M
```

Le mot-clé `public` garantit l'accès aux membres de `Virt_M`. `Volt_M` fournit à ses membres hérités l'attribut `sV`, le constructeur et la méthode `get_Unit`. La déclaration du constructeur n'apporte rien de nouveau, mais la définition, oui. Ici, le constructeur 2 de `Virt_M` est appelé. Lors de l'instanciation, les paramètres passés à `Volt_M` sont transmis à la classe `Virt_M`, la valeur initiale `ini_m` directement par le constructeur `Virt_M`, l'adresse multiplexée par la méthode `set_Mux` de `Volt_M`. La définition des constructeurs est la suivante :

```
Volt_M::Volt_M (byte v_mux, float ini_m):Virt_M(ini_m)
   // ini_m fourni par le constructeur de la classe
   Virt_M
   { set_Mux(v_mux); }     // v_mux par la méthode set_
```

La classe `Amp_M` est également une fille de `Virt_M`. Elle a une structure similaire à celle de `Volt_M`, sauf que le constructeur appelle le constructeur 3 de `Virt_M`. Le transfert complet des valeurs vers `Virt_M` est effectué sans aucune méthode supplémentaire. Il convient de noter ici qu'un héritage peut surcharger une méthode héritée.
Les classes sont maintenant entièrement déclarées, prêtes à l'utilisation. Les objets `M1`, `M2` et `M3` du type de classe `Virt_M` montrent l'appel au constructeur surchargé. `My_Volt_M` et `My_Amp_M` sont des objets des classes héritières `Volt_M` et `Amp_M`.

Après le démarrage du programme, le comportement de nos objets peut être suivi et compris grâce au moniteur série. Des accès invalides sont délibérément intégrés dans le croquis, mais sont initialement mis en commentaires dans l'exemple. En enlevant les marqueurs de commentaire `//` et en recompilant, nous pouvons étudier les messages d'erreur fournis par le compilateur.
Les objets enfants utilisent les structures héritées comme si elles faisaient partie d'eux-mêmes. Cependant, chaque objet possède sa propre copie des données. Tout se passe comme si le code de `Virt_M` avait été copié-collé dans `Volt_M` et `Amp_M`. Le principal inconvénient de ce type de relation (en plus d'un surcroît de travail et d'un code plus volumineux) est que toute modification de `Virt_M` entraîne des modifications chez toutes les classes qui la contiennent.

## Une classe peut être aussi une amie

Dans le dernier exemple, nous avons vu qu'en raison de l'encapsulation, il n'est pas possible de modifier l'adresse multiplexée d'un objet. Cependant, surtout lorsque la programmation est proche du matériel, il est souvent souhaitable de disposer de nombreuses possibilités d'accès. Par exemple, si, lors du débogage nous voulons parcourir séquentiellement toutes les adresses multiplexées, nous devrons créer un objet pour chacune. Une autre possibilité serait de détruire l'objet après l'appel, puis de le recréer. Bien sûr, nous pourrions mettre la classe de base en accès public dès le début. Mais cela reviendrait à renoncer à l'un des avantages de la PPO, l'assurance d'un certain niveau de sécurité pour les données. La *leçon 4* nous montre une solution élégante à ce problème. Une nouvelle classe appelée `Debug_M` hérite de la classe

**Listage 4. Déclaration de la classe Virt_M avec Debug_M comme classe « friend » (leçon 4).**

```
class Virt_M
  {
    friend class Debug_M;               //la classe Debug_M obtient
                                        //l'accès aux membres privés
                                        //de Virt_M

    private:                            //membres privés de la classe
      float value=0;                    //valeur de mesure virtuelle, fixée à 0
      byte mux_adr= MUX_INVALID;        //adresse multiplexée pour la simulation

    public:                             //membres également accessibles depuis l'extérieur
      Virt_M (byte mux, float ini_m);   //constructeur avec valeurs initiales
      void set_Value (float val);       //méthode d'écriture des valeurs initiales
      float get_Value ();               //méthode de lecture de l'octet de mesure virtuel
      byte get_Mux ();                  //méthode de lecture de l'adresse mux
      String s="public: Demostring";    //pour démo seulement

  };                                    //Fin de la déclaration de la classe
```

**Listage 5. Déclaration de la structure t_result (leçon 5).**

```
struct t_result    //type pour retourner des données
  {
    byte mux;      //adresse mux
    String s;      //chaîne de caractères
    float val;     //valeur de mesure de type float
  };
```

**Listage 6. Déclaration des attributs de la classe Dev_M (leçon 5).**

```
class Dev_M
  {
    private:
      Volt_M *ptr_v;                        //pointeur vers un objet de type Volt_M
      Amp_M *ptr_a;                         //pointeur vers un objet de type Amp_M
      t_result result;                      //pour le retour de données
      const String sP=( "Power/ W   : ");
      const String sR= ("Resist/ Ohm: ");
```

de base `Virt_M` en tant qu'amie (`friend`) de cette classe. `Debug_M` est identique à `Volt_M`, sauf qu'elle a accès aux membres privés de `Virt_M`. Pour modifier l'adresse multiplexée, nous pouvons utiliser la méthode `set_` de `Debug_M` qui est déclarée `public`. Le **listage 4** montre la déclaration d'une classe `friend`.

La classe `Virt_M` ne contient pas de méthode pour définir l'adresse multiplexée. Au lieu de cela, le constructeur reçoit deux valeurs pour initialiser les attributs de la classe. L'une de ces valeurs est l'adresse multiplexée. Les paramètres sont transmis lorsque la classe de base et les sous-classes sont instanciées.

Une fois de plus, les informations de la routine `Setup` montrent les résultats de notre travail. Les classes amies sont rarement utilisées, bien qu'elles aient l'avantage d'utiliser des méthodes existantes, d'offrir les fonctions souhaitées sans compromettre le concept de sécurité de l'encapsulation lorsqu'elles sont utilisées correctement.

## Accès d'une classe à une méthode d'une autre classe

Dans les derniers exemples, nous avons effectué des calculs de puissance et de résistance dans le programme principal. Nous avons lu les attributs de `Volt_M` et `Amp_M` via leurs méthodes. Nous allons maintenant le faire dans une classe qui leur est propre. La *leçon 5* définit la nouvelle classe `Dev_M`, qui accède aux méthodes des objets de `Volt_M` et `Amp_M`. Pour cela, lorsqu'une instance de `Dev_M` est créée, des pointeurs vers les objets à appeler sont passés par le constructeur à `Dev_M`.

Mais d'abord, en utilisant `struct`, nous créons un type de données nommé `t_result` (**listage 5**), avec plusieurs membres pour contenir les résultats. Ensuite, nous déclarons la classe `Dev_M` (**listage 6**). Les pointeurs `ptr_v` et `ptr_a`, qui sont utilisés dans les classes `Volt_M` et `Amp_M`, sont initialisés par le constructeur (**listage 7**). L'opérateur

**Listage 7. Le constructeur de Dev_M enregistre les pointeurs (leçon 5).**

```
    // Le constructeur passe les pointeurs aux objets de Volt_M, Amp_M
    //----------------------------------------------------------------

          Dev_M::Dev_M (Volt_M *ptrv, Amp_M *ptra)
            {
               ptr_v= ptrv;              //pointeur vers un objet de Volt_M
               ptr_a= ptra;              //pointeur vers un objet de Amp_M
            }
```

**Listage 8. Définition de la méthode get_Powr de Dev_M (leçon 5).**

```
t_result Dev_M::get_Powr ()               //calcul de la puissance
  {
    result.s= sP;                         //nom, unité
    result.val= ptr_v -> get_Value () * ptr_a -> get_Value ();
    return result;                        //données retournées dans une structure t_result
  }
```

**Listage 9. Instanciation des objets, passage des pointeurs aux objets (leçon 5).**

```
  Volt_M My_Volt_1 (V_MUX0, 1.00);          //objet de type Volt_M
  Volt_M My_Volt_2 (V_MUX1, 2.00);          //objet de type Volt_M
  Amp_M My_Amp_1 (A_MUX0, 1.10);            //objet de type Amp_M
  Amp_M My_Amp_2 (A_MUX1, 2.10);            //objet de type Amp_M
  Dev_M My_Dev_1 (&My_Volt_1, &My_Amp_1);   //objet de type Dev_M, passage de pointeurs
  Dev_M My_Dev_2 (&My_Volt_2, &My_Amp_2);   //objet de type Dev_M, passage de pointeurs
```

'`*`' indique que le nom suivant est interprété comme un pointeur. Les méthodes `Dev_M` peuvent maintenant utiliser les pointeurs pour accéder aux objets externes. L'accès est réalisé en utilisant une instruction de la forme `pointeur->méthode`. L'opérateur '`->`' à la place de l'opérateur '`.`' (point) permet d'appeler une méthode en utilisant un pointeur (**listage 8**).
Pour pouvoir créer une instance de `Dev_M`, il est nécessaire de créer au préalable de nouvelles instances des classes référencées. Celles-ci doivent être du type `Volt_M` et `Amp_M`. Lors de l'instanciation de `Dev_M`, nous déclarons les pointeurs vers ces objets dans la liste des paramètres du constructeur en utilisant l'opérateur '`&`' (**listage 9**). Comme le montre la sortie de la routine `Setup` (**fig. 3**) de notre application, les appels des méthodes des objets `My_Dev_1` et `My_Dev_2` fournissent tous les résultats.

## Héritage multiple

Dans l'exemple précédent, nous avons d'abord dû créer deux instances distinctes avant de pouvoir utiliser les méthodes de `Dev_M`. Il y a sans doute des applications où cela a un sens, mais ici, c'était surtout pour illustrer l'usage de pointeurs vers des objets. Les classes de base et les classes enfants peuvent être héritées directement par une autre classe comme le montre la *leçon 6* ; c'est l'*héritage multiple*. Les deux classes dérivées `Volt_M` et `Amp_M` sont héritées par la classe de base `Dev_M`. Comme les classes dérivées contiennent `Virt_M`, `Dev_M` a accès aux membres de toutes les classes.

Afin de renvoyer simplement les résultats complexes des méthodes `Dev_M`, nous déclarons à nouveau le type de données `t_result`. Lorsque la classe est créée, les classes héritées sont listées après l'opérateur '`:`', séparées par des virgules :

```
class Dev_M: public Volt_M, public Amp_M
```

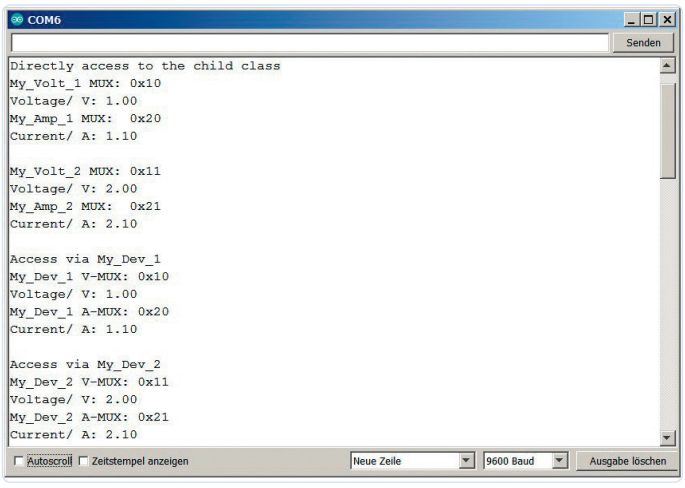



*Figure 3. Sortie de la routine Setup de la leçon 5.*

**Listage 10. Référence explicite à un membre en cas d'ambiguïté (leçon 6).**

```
t_result Dev_M::get_Volt ()             //lecture de la tension courante
  {
    result.s= Volt_M::get_Unit ();      //appel de la méthode de Volt_M
    result.val= Volt_M::get_Value ();   //appel de la méthode de Volt_M
    return result;
  }
```

**Listage 11. Ambiguïté de la méthode get_Mux (leçon 6).**

```
  Serial.print ("My_Dev_1, Volt MUX: 0x"),
  Serial. println (My_Dev_1.Volt_M::get_Mux(), HEX);  //méthode de la classe de base
  Serial.print (My_Dev_1.get_Volt().s);
  Serial.println (My_Dev_1.get_Volt().val);           //valeur initiale de l'instanciation de My_Dev_1
  Serial.println ();
```

**Listage 12. Utiliser ses propres méthodes pour lever l'ambiguïté (leçon 6).**

```
  Serial.print ("My_Dev_2, Volt MUX: 0x");
  Serial.println (My_Dev_2.get_VMux(),HEX);   //méthode de la classe héritée
  Serial.print (My_Dev_2.get_Volt().s);
  Serial.println (My_Dev_2.get_Volt().val);   //valeur initiale de l'instanciation de My_Dev_M
  Serial.println ();
```

Les paramètres à transmettre aux classes héritées sont énoncés dans une liste de paramètres en fin de déclaration du constructeur :

```
Dev_M (byte v_x, float i_v, byte a_x, float i_a);
  // le constructeur de la classe Dev_M
```

La définition du constructeur attribue ensuite les valeurs de transfert aux paramètres. A l'instanciation de `Dev_M`, les constructeurs de `Volt_M` et `Amp_M` sont appelés et les attributs de ces classes sont initialisés avec les paramètres.

```
Dev_M::Dev_M (byte v_x, float i_v, byte a_x, float
  i_a): Volt_M (v_x, i_v), Amp_M (a_x, i_a) {}
```

Comme déjà mentionné, nous pouvons accéder à tous les attributs et méthodes des classes héritières via `Dev_M`, à condition que la spécification d'accès le permette. `Dev_M` nous fournit les méthodes appropriées pour ce faire. Il y a cependant un problème : comme les deux classes héritées sont elles-mêmes dérivées de la classe `Virt_M`, elles ont des méthodes et des attributs homonymes, ce qui conduit à des ambiguïtés qu'on lève en utilisant l'opérateur de résolution de portée '`:`' pour spécifier explicitement de quel membre de quelle classe il s'agit (**listage 10**).
Nos classes sont maintenant définies et nous pouvons en instancier les objets. En principe, les méthodes de `Dev_M` sont tout à fait suffisantes pour écrire une application. Pour une meilleure compréhension, nous créons également des instances supplémentaires de la classe de base et des classes dérivées. Comme une adresse multiplexée différente est attribuée à chaque objet lors de l'instanciation, cela nous indique quel objet est actuellement adressé.
Après le démarrage de l'application, le moniteur série affiche les résultats des appels aux méthodes. L'exécution de la routine `Setup` est intéressante. Les attributs de la classe de base sont lus de différentes manières. L'accès à l'adresse multiplexée de l'objet `My_Dev_1` appelle la méthode de la classe de base. On a le même problème d'ambiguïté qu'avec les définitions des méthodes de `Dev_M`. Pour lire l'adresse multiplexée, on doit préciser explicitement si c'est celle de la tension ou du courant. La méthode est appelée `get_Mux` dans les deux cas (**listage 11**). Le problème est résolu en spécifiant la référence de la classe.
Il est bien sûr beaucoup plus clair et finalement plus simple de fournir leurs propres méthodes aux classes enfants `Volt_M` et `Amp_M` (**listage 12**). Dans l'exemple, ces méthodes sont respectivement `get_VMux` et `get_AMux`. Elles sont utilisées lors de la lecture des attributs de `My_Dev_2`.
La fonction `sizeof` sert à déterminer le nombre d'octets dans les champs de données de notre objet. La classe de base alloue 11 octets tandis que les classes dérivées en utilisent 17. Comme la classe de base est incluse dans chaque classe dérivée, les classes dérivées ajoutent 6 octets chacune. `Dev_M` hérite deux fois de 17 octets et nécessite elle-même 22 octets ; on arrive à un total de 56 octets (**fig. 4**).

## Un régal pour les fans d'Arduino

Je suis sûr qu'il y a des fans d'Arduino qui ont souvent souhaité pouvoir intégrer une des méthodes Arduino dans leur programme. Peut-être avez-vous développé votre propre afficheur que vous

aimeriez gérer avec la puissante méthode `print`. Comme déjà suggéré, cela peut être réalisé en créant une classe qui hérite de la classe `Print`. Le croquis de la *leçon 7* montre comment faire. La classe `My_Print_C` hérite de toutes les méthodes de `Print` et surcharge la méthode `write` originale, qui normalement envoie les données d'affichage sur un matériel spécifique. La méthode `write` reçoit un seul caractère de la méthode `print(ln)` comme argument, qu'elle transfère ensuite à l'afficheur via une interface matérielle, telle qu'un bus I²C. Notre exemple n'utilise aucun matériel : les caractères transférés sont écrits dans une chaîne dont le contenu peut être visualisé sur le moniteur série.
Comme les méthodes `print(ln)` sont surchargées, le type de valeurs que nous pouvons « imprimer » n'a pratiquement aucune importance. C'est tout. Nous pouvons maintenant utiliser `print` pour envoyer des caractères à notre propre afficheur.

## Conclusion

J'espère que cet article vous a fourni une bonne base pour débuter dans la programmation orientée objet. Nous avons examiné les bibliothèques de classes et la manière de les intégrer dans nos applications. Dans le dernier exemple, nous avons montré qu'hériter de classes externes et les modifier n'est pas un problème. Même s'il reste encore beaucoup à apprendre, les bases pour écrire vos propres programmes orientés objet devraient maintenant toutes être en place.
J'espère vous avoir convaincus des avantages de la POO sur la programmation procédurale. Ces notions peuvent être difficiles à maîtriser au début, surtout si vous êtes un familier de la programmation procédurale. Mais courage, c'est en forgeant qu'on devient forgeron !

(200563-04)

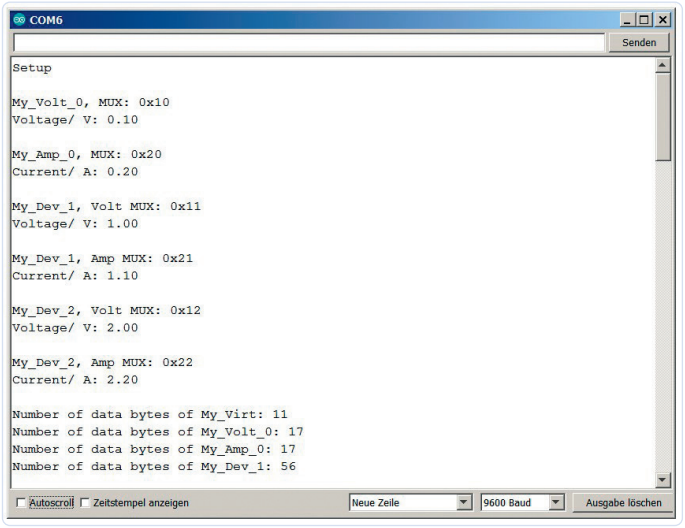
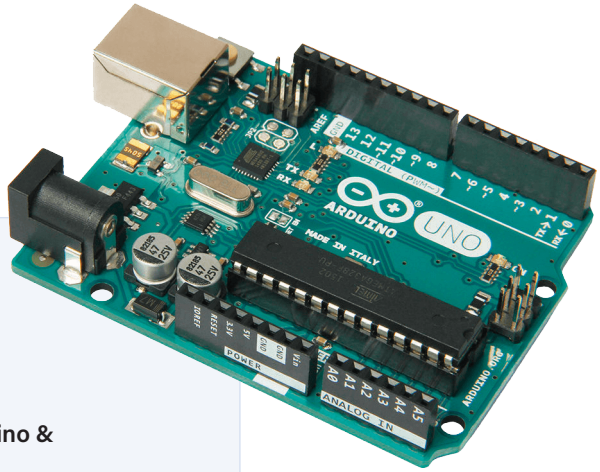



*Figure 4. Sortie de la routine Setup de la leçon 6.*







**Contributeurs**
Idée et texte : **Roland Stiglmayr**
Rédaction : **Rolf Gerstendorf**
Mise en page : **Giel Dols**
Traduction : **Helmut Müller**

**Des questions, des commentaires ?**
Envoyez un courriel à l'auteur (1134-715@online.de) ou contactez Elektor (redaction@elektor.fr).


## LIENS


[1] **Page de cet article** : www.elektormagazine.fr/200563-04
[2] **Programmation en C++ (en anglais)** : https://en.wikibooks.org/wiki/Subject:C%2B%2B_programming_language
[3] **Séminaire à l'Université technique de Munich** : www.ei.tum.de/fileadmin/tueifei/ldv/Vorlesungen/cpp/CPP-Skript.pdf

# dans l'antre de...

## Kurt Diedrich et de son synthétiseur analogique

**Kurt Diedrich** (Allemagne) et **Eric Bogers** (Elektor)

C'est en 1964, alors en pleine Beatlemania, qu'est apparu le premier synthétiseur analogique modulaire Moog [1]. Comme les quatre garçons, l'instrument fut lui aussi rapidement dans le vent et révolutionna la musique grâce à ses modules de synthèse sonore. Le dernier modèle date de 1980. Le Moog inspira de nombreux projets de lutherie électronique dont, en 1977, le Formant d'Elektor – oublié un temps, puis redevenu populaire avec le regain d'intérêt pour les instruments pionniers [2]. Kurt Diedrich s'est lui aussi attaqué à la fabrication d'un synthétiseur modulaire de type Moog, un projet idéal pour les loisirs puisqu'il repose sur des composants courants et bon marché.

*Figure 1. Le labo de Kurt, sans Kurt.*

*Figure 2. Le labo de Kurt, avec Kurt.*

*Kurt Diedrich fréquente les pages d'Elektor comme lecteur et auteur depuis plus de 40 ans. Mais laissons-le plutôt se présenter lui-même :*

J'ai étudié la géologie de 1972 à 1980, et à l'époque déjà je trouvais l'électronique beaucoup plus intéressante que les cailloux. J'ai assemblé le premier de mes synthétiseurs en 1973, année qui fut aussi celle de mes premiers contacts avec la revue *Elektuur*, plus tard rebaptisée Elektor. Mon premier livre date quant à lui de 1981. Il portait sur les synthétiseurs et fut publié par Frech Verlag [3].
Après mes études, j'ai d'abord travaillé comme assistant à l'université, puis Elektuur m'a proposé un poste de rédacteur en chef et de concepteur à Beek (Pays-Bas). C'était en 1981, et surtout le début d'une période aussi formidable que les collègues qui m'entouraient. J'ai notamment conçu pour Elektuur un synthétiseur à puces CEM de Curtis, et aussi de nombreux circuits de mesure et de musique électronique.
À partir de 1985, j'ai également été concepteur et auteur pour Elex, une revue pour électroniciens en herbe. J'ai malheureusement dû quitter ce poste en 1987 pour motif économique. J'ai ensuite été rédacteur de manuels techniques pour diverses entreprises, et passé les 17 dernières années de ma carrière chez *HEAD Acoustics*, près d'Aix-la-Chapelle.
En 2014, parvenu à l'âge de la retraite, je me suis demandé comment occuper mon temps libre de façon utile et plaisante. Je me suis souvenu de mon intérêt pour les synthétiseurs, et depuis je m'efforce de construire des instruments qui fonctionnent bien, qui « *foncsonnent* » bien devrais-je dire, en utilisant des composants ordinaires, faciles à trouver et peu chers. J'ai l'impression que je m'en tire très bien.

*Kurt dispose de plusieurs espaces de travail qui le font se sentir privilégié. Voyez plutôt le labo spacieux et bien rangé (**fig. 1**) dans lequel il travaille sur la dernière version de son synthétiseur (**fig. 2**). Son bureau semble tout aussi confortable (**fig. 3**), et il dispose également d'un atelier pour ses travaux plus mécaniques (**fig. 4**). Reconnaissons-le, c'est Versailles ! Les circuits de son synthétiseur sont-ils eux aussi agencés d'une manière aussi enviable ? N'en doutons*

*pas, mais les détails n'ayant pas leur place ici, nous devrons nous contenter d'un bref survol du diagramme fonctionnel de l'instrument (**fig. 5**), en rappelant qu'il s'agit d'un projet en constant développement. Mais redonnons la parole à Kurt :*

Le diagramme fonctionnel montre la topologie de base du synthétiseur. Chaque rectangle vert représente une carte de circuit imprimé, et chacune de ces cartes loge un module, à l'exception de la carte 3-*fach VCA* (triple VCA) qui contient aussi les circuits du générateur de bruit et du mixeur (car ils n'occupent que très peu d'espace).
L'unité *Netzteil* (bloc d'alimentation) est une alimentation à découpage externe délivrant jusqu'à 20 VCC. De cette façon la dangereuse tension du secteur reste à l'extérieur de l'instrument. Les 2×12 V requis sont délivrés par une carte d'alimentation interne à convertisseur CC/CC (2×500 mA). Le bloc d'alimentation peut être celui d'un vieux portable, si tant est bien sûr que sa tension de sortie convienne.
Le convertisseur MIDI opère avec l'appui d'une carte Arduino Nano. Je n'ai pas conçu moi-même son circuit, juste la carte qui le loge.
L'unité *Retrigger-Unit* permet le jeu *legato* (qui revient à jouer la note suivante en gardant enfoncée la touche de la note actuelle – un geste presque inévitable avec un jeu rapide). Sans fonction « legato », un synthé comme le mien ne pourrait se jouer qu'à un doigt – et autant se mettre au pipeau si c'est pour ça.
L'unité *Delayed Vibrato* fournit une fréquence de vibrato qui augmente lentement lorsqu'une touche reste enfoncée plus d'une ou deux secondes. L'effet obtenu apporte un charme singulier à certains morceaux. Parmi les ondes produites par l'unité LFO, l'onde carrée permet de créer des sons intéressants.
Le clavier du diagramme est la seule partie de l'instrument que j'ai achetée prête à l'emploi. Comme j'ai conçu ce synthétiseur pour être compatible avec les claviers MIDI standard et qu'il existe de nombreux modèles bon marché, le choix est vaste. J'utilise un clavier *Easy Key* 49 de SwissSonic.
La plupart des instruments électroniques à clavier sont dotés d'une sortie MIDI additionnelle qui peut également servir à commander mon synthétiseur, du moins celui présenté ici.

*Si ce prélude vous a donné envie de connaître l'œuvre complète de Kurt Diedrich, vous pouvez le contacter par courriel (cf. encadré), il vous fournira les schémas et autres listes de composants de son synthétiseur.*

(200681-04)

*Figure 3. Le bureau de Kurt.*

*Figure 4. L'atelier de Kurt réservé aux travaux mécaniques.*

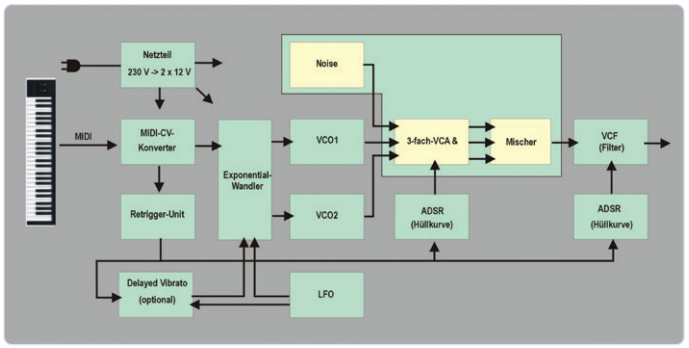



*Figure 5. Diagramme fonctionnel du synthétiseur.*

**Des questions, des commentaires ?**
Envoyez un courriel à l'auteur en anglais (subroutine-sy@gmx.de) ou contactez Elektor (redaction@elektor.fr).

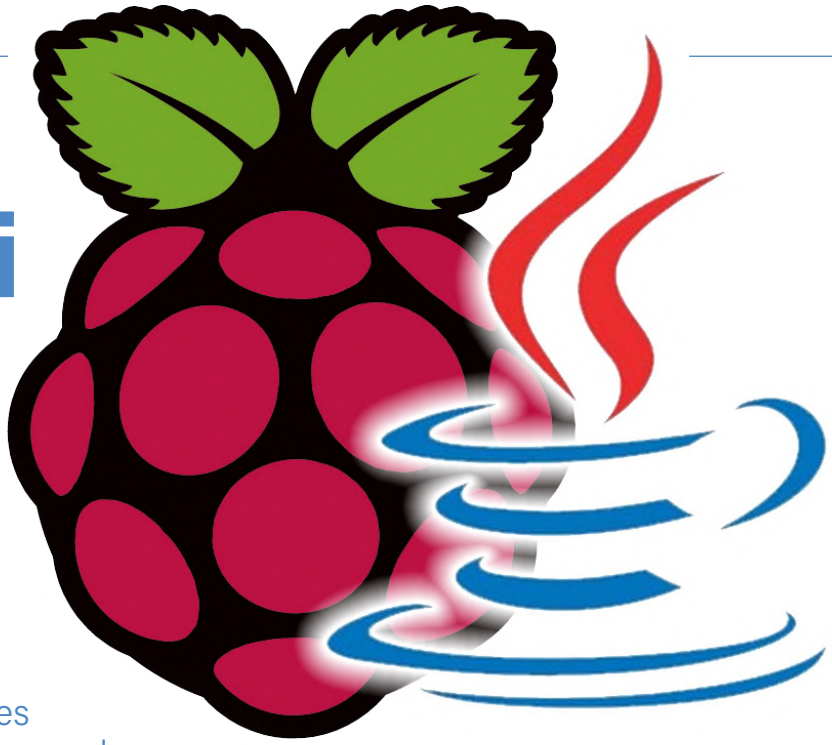
**Contributeurs**
Texte, photos et diagramme : **Kurt Diedrich**
Rédaction : **Eric Bogers**
Maquette : **Giel Dols**
Traduction : **Hervé Moreau**

## LIENS

[1] **Synthétiseur Moog** : https://fr.wikipedia.org/wiki/Moog
[2] **Jan Buiting, « Le synthétiseur Formant », Elektor 4/2008** : www.elektormagazine.fr/magazine/elektor-200804/18762
[3] **Livres de Kurt Diedrich** : www.subroutine.info/bücher/

# Java sur Raspberry Pi

## Partie 1 : les broches GPIO



**Frank Delporte** (Belgique)

Même si le langage de programmation Java existe depuis longtemps, il reste très adapté au développement de code pour les plates-formes informatiques récentes telles que le Raspberry Pi. Pour démontrer ses capacités, la première partie de cette série fournit quelques informations sur ce langage de programmation et examine comment Java peut commander et lire les broches GPIO.

### Brève introduction à Java

Java fait partie des langages de programmation qui commencent à dater. La première version est sortie en 1995 [1], en même temps que JavaScript, PHP et Qt. Python est un peu plus ancien, avec une première version en 1991. Lorsque la marque Java est passée de Sun Microsystems à Oracle, le développement est passé en mode *open source* et a été transféré sur GitHub en 2020 [2]. Depuis 2018, deux nouvelles versions de Java sortent chaque année, ce qui permet de mettre plus régulièrement à jour nos machines avec les correctifs et les nouvelles fonctions. Cela inclut des améliorations qui ciblent les plates-formes embarquées telles que le Raspberry Pi.

On pourrait penser que « Java est mort », mais les multiples projets et conférences (même si elles sont toutes devenues virtuelles en ces temps de Corona) liés à Java sont la preuve que ce langage est toujours vivant. Les principaux contributeurs au développement de Java sont une longue liste de sociétés bien connues, dont Oracle, Microsoft, SAP, Google et d'autres [3]. La rumeur veut également que la moitié du nuage Azure de Microsoft fonctionne en Java ! Aujourd'hui, les distributions Java sont disponibles auprès de différents fournisseurs de logiciels libres (jdk.java.net, adoptopenjdk.net) et commerciaux (Azul, BellSoft, Oracle, et d'autres).

### Java sur Raspberry Pi ?!

Le Raspberry Pi n'a-t-il pas été conçu pour être programmé en Python ? Peut-être que oui, mais cela ne veut pas dire que vous ne pouvez pas utiliser d'autres langages. Bien qu'il s'agisse d'un article sur Java, nous ne voulons pas insinuer qu'il est préférable à Python, C ou tout autre langage ! Chaque projet a ses propres exigences

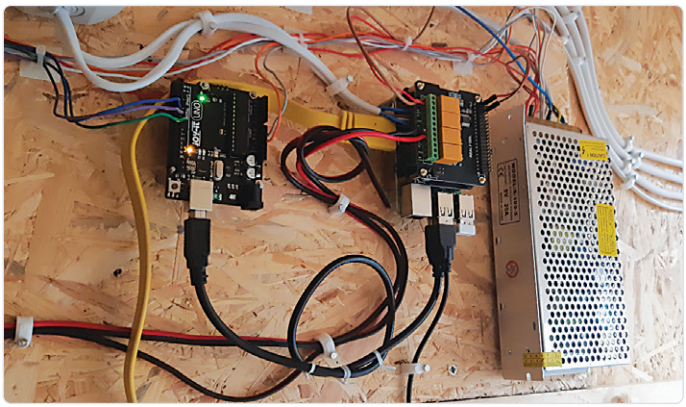


*Figure 1. Le cœur du contrôleur à écran tactile pour la cabine acoustique de batterie.*


*Figure 2. Interface de commande à écran tactile pour le relais et les rubans de LED.*

pour lesquelles un langage spécifique pourrait être parfaitement adapté, ou mieux correspondre aux compétences de l'équipe de développement.
Dans mon cas, en tant que développeur Java, cela m'intéressait vraiment de voir si je pouvais appliquer mes connaissances au Raspberry Pi. C'est arrivé après ma première expérience de programmation d'un jeu de pong en Python, qui s'est mal terminée avec une interface utilisateur décevante. Justement, les interfaces utilisateur peuvent être créées avec JavaFX, un projet indépendant [4, 5] qui complète Java avec un canevas pour construire des GUI (*Graphical User Interface*). Il fournit tous les composants de base typiques (boutons, étiquettes, tableaux, graphiques), et beaucoup de bibliothèques gratuites et à code source ouvert peuvent enrichir cette liste.
C'est au moment de construire une interface à écran tactile pour la cabine acoustique de la batterie de mon fils (**fig. 1** et **fig. 2**) que j'ai trouvé l'association parfaite entre Java et le Raspberry Pi. Combiné à une carte relais, un Arduino et quelques rubans de LED, cela constituerait la base de mon expérimentation dans un nouveau monde de programmation Java embarquée.

## Se préparer

Pour reproduire les expériences de cet article, vous aurez besoin d'un Raspberry Pi récent avec un processeur ARMv7 ou ARMv8. Les cartes plus anciennes avec un ARMv6 ont une constitution interne différente pour laquelle le Java par défaut ne fonctionnera pas. Pour utiliser Java sur une carte ARMv6, il y a une solution : le JDK Azul Zulu [6]. De plus, pour vous éviter une fastidieuse saisie, tous les exemples de code accompagnant cette série sont disponibles sur un dépôt GitHub [7].
Commencez par préparer une carte SD avec l'outil « Imager » [8] et sélectionnez « Raspberry Pi OS Full (32-bit) » (**fig. 3**). Une fois votre Raspberry Pi lancé, ouvrez le terminal et tapez `java -version` pour vérifier que la version est bien « 11 ». Comme vous le verrez, OpenJDK est préinstallé dans cette version complète de l'OS :

```
$ java -version
openjdk version "11.0.9" 2020-10-20
OpenJDK Runtime Environment (build
   11.0.9+11-post-Raspbian-1deb10u1)
OpenJDK Server VM (build 11.0.9+11-post-Raspbian-
   1deb10u1, mixed mode)
```

## Code pour Visual Studio

Vous pouvez développer, tester et exécuter votre code Java sur un PC avant de le transférer sur le Raspberry Pi une fois terminé. Mais il existe une autre approche : Visual Studio Code [9]. Cet EDI (Environnement de Développement Intégré) gratuit de Microsoft possède une longue liste d'extensions qui en font le compagnon idéal pour tout projet de programmation. Il existe une version pour les systèmes ARM à 32 bits (comme Raspberry Pi OS) et ARM à 64 bits (nouveau Raspberry Pi OS, encore en cours de développement) ou même Ubuntu Desktop [10].
Il existe aussi un « Java Extension Pack » qui ajoute plusieurs extensions Java à l'EDI pour en faire un EDI complet pour les développeurs Java (**fig. 4**) !

## Essai avec *Hello World!*

Essayons notre tout premier programme Java sur le Raspberry Pi.

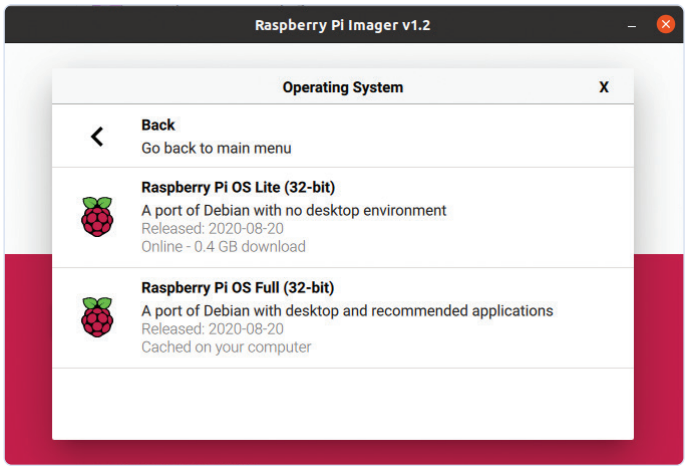



*Figure 3. L'outil Raspberry Pi Imager – la meilleure façon de préparer une carte SD.*

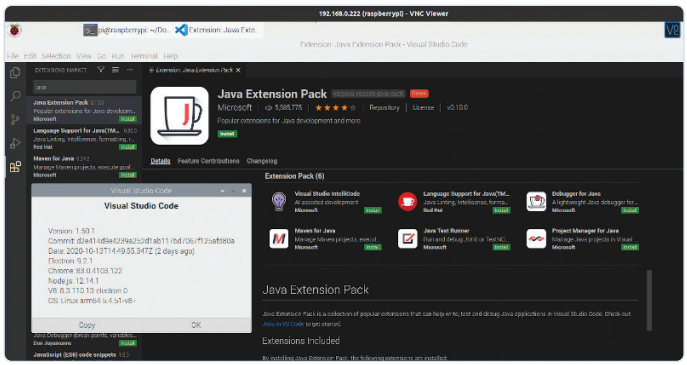



*Figure 4. On peut utiliser Visual Studio Code sur Raspberry Pi avec un pack d'extensions Java.*

Avec Visual Studio Code, un éditeur de texte, ou dans le terminal, créez un nouveau fichier texte nommé « HelloWorld.java » avec ce contenu :

```
public class HelloWorld {
  public static void main(String args[]) {
    System.out.println("Hello World!");
  }
}
```

Tout notre code appartient à la classe `HelloWorld` qui, par convention, porte le même nom que le fichier. Un programme Java démarre avec la méthode publique `static void main(String args[])`. La seule chose que nous faisons ici est d'imprimer « Hello World! » comme sortie du programme.
Avec Java 11, il est possible d'exécuter ce genre de programme simple sans compiler le code. Dans le terminal, depuis le répertoire contenant votre fichier Java, exécutez la commande `java HelloWorld.java` pour obtenir le résultat suivant :

```
pi@raspberrypi:~/elektor $ java HelloWorld.java
Hello World!
```

**Listage 1. Code CPUtemp.java pour lire la température du processeur du Raspberry Pi [7].**

```
import java.io.BufferedReader;
import java.io.File;
import java.io.FileReader;
import java.util.List;
import java.util.ArrayList;

public class CPUTemp {
    private static final String FILE = “/sys/class/thermal/thermal_zone0/temp”;
    private static final List<Integer> values = new ArrayList<>();

    public static void main(String[] args) throws InterruptedException {
        while(true) {
            checkTemp();
            Thread.sleep(1000);
        }
    }

    private static void checkTemp() {
        try (BufferedReader br = new BufferedReader(new FileReader(FILE))) {
            int value = Integer.valueOf(br.readLine());
            values.add(value);
            int total = values.stream().mapToInt(Integer::valueOf).sum();
            System.out.println(“Now: “ + value
                + “ - Average: “ + (total / values.size())
                + “ - Number of measurements: “ + values.size());
        } catch (Exception ex) {
            System.err.println(“Error during temperature check: “
                + ex.getMessage());
        }
    }
}
```

Certes, `print()` en Python c'est plus court que `System.out.println()` en Java, mais pardonnons ce « péché de jeunesse ».

## Lecture de la température de l'UC

De nombreuses valeurs du système, entrées et sorties, sont accessibles via le répertoire `/sys/` du système de fichiers Linux. Les valeurs stockées peuvent simplement être lues comme un fichier texte. Essayons de trouver la température du processeur du Raspberry Pi à partir d'un des fichiers de ce répertoire `sys`. Le nommage dans ce répertoire n'est pas toujours très clair, et il n'est pas toujours simple de déterminer comment interpréter les valeurs, mais c'est un bon point de départ pour apprendre à connaître un peu le système Linux. Exécutez la commande suivante :

```
$ ls -l /sys/
total 0
drwxr-xr-x   2 root root 0 Dec  2 15:44 block
drwxr-xr-x  29 root root 0 Feb 14  2019 bus
drwxr-xr-x  64 root root 0 Feb 14  2019 class
drwxr-xr-x   4 root root 0 Feb 14  2019 dev
drwxr-xr-x  10 root root 0 Feb 14  2019 devices
drwxr-xr-x   3 root root 0 Feb 14  2019 firmware
drwxr-xr-x   8 root root 0 Jan  1  1970 fs
```

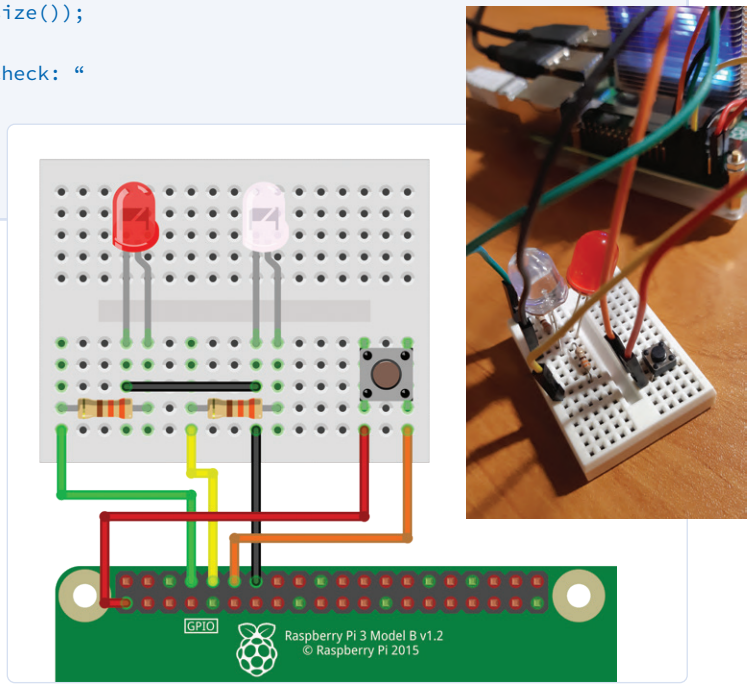



*Figure 5. Diagramme Fritzing (à gauche) pour les LED et le bouton-poussoir, avec une photo (à droite).*

```
drwxr-xr-x  12 root root 0 Jan  1  1970 kernel
```

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

```
drwxr-xr-x 122 root root 0 Feb 14  2019 module
drwxr-xr-x   2 root root 0 Dec 15 11:39 power
```

On dirait qu'on peut trouver beaucoup d'informations ici ! Essayons d'en approfondir certaines :

```
$ cat /sys/firmware/devicetree/base/cpus/cpu@0/
   compatible
arm,cortex-a72
$ cat /sys/class/net/wlan0/address
dc:a6:32:c5:b7:9d
$ cat /sys/class/bluetooth/hci0/uevent
DEVTYPE=host
$ cat /sys/class/thermal/thermal_zone0/temp
30667
```

La dernière ressemblerait vraiment à une température en degrés si on la divisait par 1000 ! Écrivons un programme simple pour lire cette valeur chaque seconde et calculer la température moyenne. Le code du **listage 1** utilise plus de méthodes Java, Il faut donc commencer par plusieurs importations. Les méthodes `io` sont utilisées pour lire un fichier `sys` comme un fichier texte. Les méthodes `List` et `ArrayList` servent à conserver une liste de toutes les valeurs mesurées. À partir de la méthode `main()`, nous appelons une méthode séparée `checkTemp()` pour acquérir et stocker la température dans la liste. On garde ainsi le code propre, car il vaut mieux isoler chaque fonction dans sa propre méthode. Comme on peut le voir, `Thread.sleep(1000)` est utilisé pour attendre une seconde entre chaque lecture de la température. Dans le cadre de cette méthode, la moyenne est calculée en additionnant la liste des valeurs à l'aide d'un flux. Les flux ont été introduits dans Java 8 et constituent un moyen très puissant de travailler avec des listes d'éléments. Comme auparavant, le nom du fichier est aussi le nom de la classe « CPUTemp.java ».
Comme auparavant, il s'agit d'une simple classe Java sans dépendances supplémentaires, ce qui nous permet de l'exécuter sans avoir à la compiler. La sortie peut être interrompue en utilisant « CTRL+C » :

```
pi@raspberrypi:~/elektor $ java CPUTemp.java
Now: 36998 - Average: 36998 - Number of measurements: 1
Now: 34563 - Average: 35780 - Number of measurements: 2
Now: 35537 - Average: 35699 - Number of measurements: 3
Now: 36024 - Average: 35780 - Number of measurements: 4
Now: 35537 - Average: 35731 - Number of measurements: 5
```

## Commander une LED et lire un bouton-poussoir

Créons une autre application Java à un seul fichier pour faire clignoter deux LED et lire l'état d'un bouton. Le schéma de câblage est assez simple – un circuit rudimentaire pour relier deux LED et un bouton à différentes broches GPIO. Le bouton a besoin de 3,3 V, et non de 5,0 V, alors attention à utiliser la bonne broche d'alimentation ! Dans cette configuration, les LED sont connectées aux broches BCM 14 et 15, tandis que le bouton-poussoir est connecté à BCM 18 (**fig. 5**).

## Essais avec le Terminal

Grâce aux commandes intégrées au Raspberry Pi OS, on peut accéder aux broches GPIO depuis le terminal. Utilisons-le pour tester notre câblage. Exécutez les commandes suivantes pour configurer la broche 14 en sortie (op) et la placer à l'état haut (dh) ou bas (dl). Essayez la même chose pour l'autre LED (BCM 15) :

```
$ raspi-gpio set 14 op
$ raspi-gpio set 14 dh
$ raspi-gpio set 14 dl
```

On peut utiliser une approche similaire pour tester le bouton en configurant la broche en entrée et en demandant l'état de l'entrée :

```
$ raspi-gpio set 18 ip
$ raspi-gpio get 18
GPIO 18: level=0 fsel=0 func=INPUT pull=DOWN
$ raspi-gpio get 18
GPIO 18: level=1 fsel=0 func=INPUT pull=DOWN
```

Lorsque le bouton est enfoncé, la valeur « `level` » passe à 1.

## Idem en Java

Adoptons maintenant la même approche de base que celle utilisée dans « CPUTemp.java » et utilisons les commandes du terminal dans un programme Java en utilisant `Runtime.getRuntime().exec(cmd)`. En ajoutant la classe `Scanner` de saisie de l'utilisateur, on peut aussi acquérir le résultat de la commande et l'utiliser pour vérifier l'état du bouton. Ceci est montré dans le **listage 2**.
Le code utilise une énumération `Actions` pour simplifier le traitement des commandes. Dans sa forme la plus simple, une énumération n'est qu'une liste de noms prédéfinis, mais dans cet exemple, nous ajoutons une valeur à chaque nom. C'est un avantage important des énumérations, car elles permettent de rendre le code beaucoup plus lisible tout en le simplifiant dans de nombreux cas. Comme on peut le voir dans la méthode `doAction`, l'énumération `Action` est utilisée comme paramètre et sa valeur sert à constituer la commande qui doit être exécutée.
Les commentaires dans le code devraient suffire, car c'est du Java simple pour illustrer l'utilisation des broches GPIO en entrée et en sortie. La partie la plus complexe est la méthode `runCommand` qui utilise des méthodes combinées pour exécuter la commande, lire le résultat et gérer les éventuelles erreurs. Si ce n'est pas clair pour l'instant, pas de soucis – ça le sera quand vous exécuterez le code ! Comme cet exemple n'utilise pas non plus de dépendances tierces, on peut l'exécuter sans compilation. Le résultat devrait être évident puisque vous devriez voir vos LED s'allumer et s'éteindre à différents intervalles. Lorsqu'elles cessent de clignoter, vous pouvez appuyer sur le bouton et son état sera examiné dix fois à intervalle d'une seconde. Voici le résultat attendu :

**Listage 2. Code RaspiGpio.java pour commander les LED et obtenir l'état du bouton [7].**

```
import java.io.IOException;
import java.util.Scanner;

public class RaspiGpio {

    private static final String LED_1 = “14”;
    private static final String LED_2 = “15”;
    private static final String BUTTON = “18”;

    private enum Action {
        OUTPUT_PIN(“op”),
        INPUT_PIN(“ip”),
        DIGITAL_HIGH(“dh”),
        DIGITAL_LOW(“dl”);
        private final String action;
        Action(String action) {
            this.action = action;
        }
        String getAction() {
            return action;
        }
    }

    public static void main(String[] args) throws InterruptedException {
        // Configurer en sorties les deux broches des LED
        doAction(LED_1, Action.OUTPUT_PIN);
        doAction(LED_2, Action.OUTPUT_PIN);

        // Configurer en entrée la broche du bouton
        doAction(BUTTON, Action.INPUT_PIN);

        // Faire clignoter les LED à différents intervalles
        for (int i = 0; i < 20; i++) {
            System.out.println(“Blink loop: “ + i);
            if (i % 2 == 0) {
                doAction(LED_1, Action.DIGITAL_HIGH);
            } else {
                doAction(LED_1, Action.DIGITAL_LOW);
            }
```

```
$ java RaspiGpio.java
Executing: raspi-gpio set 14 op
Executing: raspi-gpio set 15 op
Executing: raspi-gpio set 18 ip
Blink loop: 0
Executing: raspi-gpio set 14 dh
Executing: raspi-gpio set 15 dh
Blink loop: 1
Executing: raspi-gpio set 14 dl
Executing: raspi-gpio set 15 dl
...
Executing: raspi-gpio get 18
Button check 0: GPIO 18: level=0 fsel=0 func=INPUT pull=DOWN
 - PUSHED: false
...
Button check 3: GPIO 18: level=1 fsel=0 func=INPUT pull=DOWN
 - PUSHED: true
...
```

## Préparez-vous pour la prochaine fois !

Avec quelques notions de base en poche, nous avons réussi à couvrir une petite introduction à Java sur le Raspberry Pi. Pour améliorer votre compréhension, n'hésitez pas à vous amuser avec le code en exemple et à lire la documentation en suivant les liens fournis. Nous espérons que cela simplifiera les étapes suivantes du prochain article, où nous développerons une application Java complète pour relier nos broches GPIO à une page web.

(200617-04)


**Contributeurs**

Idée, texte et illustrations : **Frank Delporte**

Rédaction : **Stuart Cording**

Mise en page : **Giel Dols**

Traduction : **Denis Lafourcade**

```
            if (i % 3 == 0) {
                doAction(LED_2, Action.DIGITAL_HIGH);
            } else {
                doAction(LED_2, Action.DIGITAL_LOW);
            }
            Thread.sleep(500);
        }
        doAction(LED_1, Action.DIGITAL_LOW);
        doAction(LED_2, Action.DIGITAL_LOW);

        // Examiner 10 fois l'état du bouton
        for (int j = 0; j < 10; j++) {
            String result = runCommand("raspi-gpio get " + BUTTON);
            System.out.println("Button check " + j
                + ": " + result
                + " - PUSHED: " + (result.contains("level=1")));
            Thread.sleep(1000);
        }
    }

    private static void doAction(String pin, Action action) {
        runCommand("raspi-gpio set " + pin + " " +
              action.getAction().toLowerCase());
    }

    private static String runCommand(String cmd) {
        System.out.println("Executing: " + cmd);
        Scanner s = null;
        try {
            s = new Scanner(Runtime.getRuntime().exec(cmd).getInputStream())
                .useDelimiter("\\A");
            return s.hasNext() ? s.next() : "";
        } catch (Exception ex) {
            System.err.println("Error during command: " + ex.getMessage());
            return "";
        } finally {
            if (s != null) {
                s.close();
            }
        }
    }
}
```

## LIENS


[1] **Andrew Binstock, « Java's 20 Years Of Innovation », Forbes** : http://bit.ly/3qeIpVu

[2] **Projet OpenJDK** : https://github.com/openjdk

[3] **Liam Tung, Oracle, « Programming language Java 14 is out with these 16 major feature improvements », ZDNet** : http://zd.net/35wVW2O

[4] **Projet OpenJFX** : https://github.com/openjfx

[5] **OpenJFX** : https://openjfx.io/

[6] **Frank Delporte, « How to install and use Java 11 and JavaFX 11 on Raspberry Pi boards with ARMv6 processor »** : http://bit.ly/35yRrFc

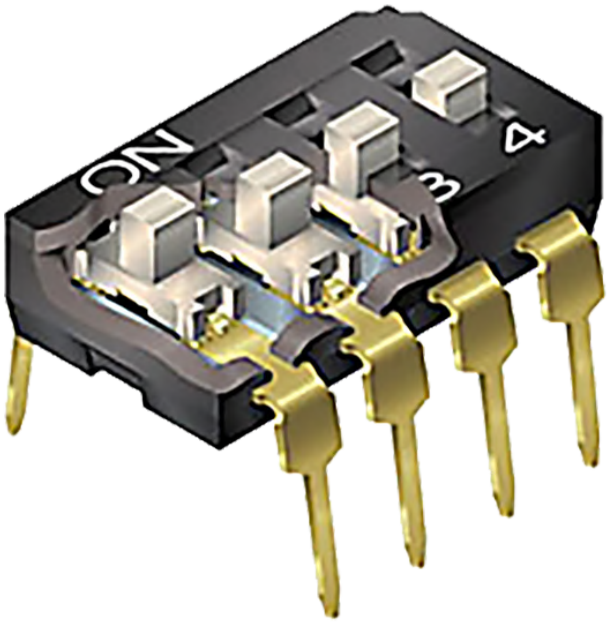
[7] **Code des exemples sur GitHub** : http://bit.ly/3i9bP4v

[8] **Outil Raspberry Pi Imager** : http://bit.ly/3i58seU

[9] **Frank Delporte, « Visual Studio Code on the Raspberry Pi (with 32 and 64-bit OS) »** : http://bit.ly/3qeh9GN

[10] **Téléchargement de Visual Studio Code pour Arm** : http://bit.ly/2LqUng3

# interrupteurs DIP




**David Ashton** (Australie)

Voyons, que peuvent-ils avoir de particulier ces interrupteurs DIP ? Après tout, ce ne sont que des interrupteurs ! Et pourtant, il y en a de très particuliers.
En voici quelques-uns de ma collection par ordre croissant de particularité.

DIP, pour les novices en électronique, signifie Dual Inline Package, c'est-à-dire que les broches sont disposées en deux rangées parallèles comme pour les boîtiers de circuits intégrés (CI). Généralement les broches sont espacées de 2,54 mm (soit 1/10e de pouce) comme sur les CI. Les interrupteurs DIP sont utilisés pour sélectionner des fonctions, des options, des adresses ou toute autre propriété nécessitant un moyen de configuration robuste et rarement utilisé. Les plus fréquents sont les modèles à huit voies, mais nous allons voir qu'il en existe de nombreux types différents.
Pour commencer, examinons les interrupteurs DIP standard que nous connaissons tous. La **figure 1** présente un portrait de groupe de membres issus de ma collection. Ils sont disponibles en plusieurs couleurs, ce qui égaye des circuits imprimés autrement assez ternes. Certains sont à bascules encastrées, pour lesquelles il faut

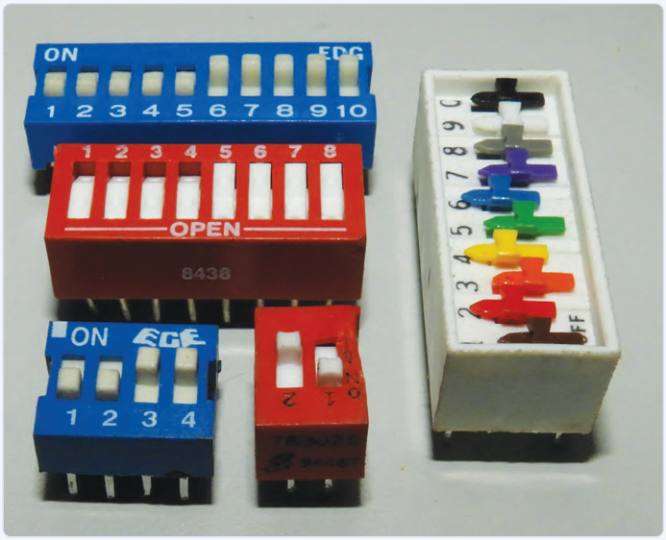



*Figure 1. Une collection d'interrupteurs DIP typiques.*

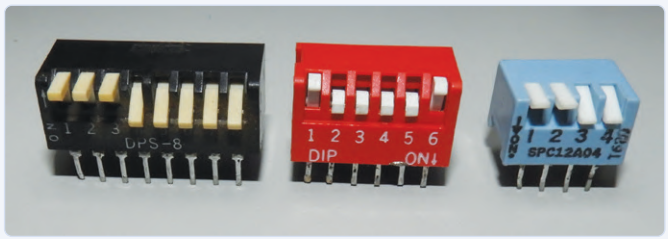



*Figure 2. Les interrupteurs DIP de type piano sont plus faciles à régler.*

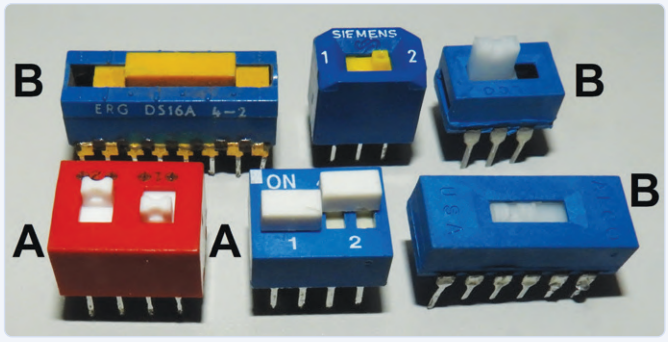



*Figure 3. Types à double contact et à glissière.*

un trombone ou un stylo (déconseillé), ou encore un crayon (détestable). D'autres ont un actionneur qui se dresse et que vous pouvez commander d'une pichenette de l'ongle. Certains, comme celui de droite, ont des actionneurs à code de couleurs. Si vous connaissez le code de couleurs des résistances, ils sont faciles à lire et à régler.

D'autres sont montés verticalement avec les actionneurs qui sortent sur le côté (**fig. 2**), ce qui améliore leur accessibilité sur les panneaux arrière des appareils. Ces types à « touches de piano » sont un peu plus faciles à commander de l'ongle.

Nous en arrivons maintenant à des exemplaires plus exotiques. Sur les modèles **A** de la **figure 3**, une seule action commute deux contacts indépendants à la fois. Les modèles **B** sont à glissière. Les plus petits ont deux inverseurs et les plus grands en ont quatre ou plus. Cela les rend utiles pour mettre en ou hors série des atténuateurs, par exemple.

J'ai également quelques sélecteurs de choix d'une position parmi huit ou dix (**fig. 4**). Les bleus de gauche ont un couvercle (enlevé sur celui du bas) et nécessitent un petit tournevis pour amener l'actionneur sur l'une des dix positions. Si vous reliez tous les contacts d'un même côté (d'origine sur certains d'entre eux), vous pouvez les utiliser pour choisir une sortie d'un CI compteur CD4017 pour un synthétiseur ou un système d'alarme.

Il existe aussi des commutateurs DIP rotatifs décimaux et hexadécimaux (**fig. 5**). Ils servent à prérégler un code binaire ou DCB, souvent pour une adresse sur une carte. C'est beaucoup plus facile que de convertir mentalement un nombre en DCB ou en hexadécimal. Ils sont très compacts et ont généralement une voie commune et quatre voies en code binaire ou DCB. Ils sont disponibles en format horizontal, vertical ou oblique.

La **figure 6 A** montre un coupe-circuit unipolaire. J'ai hésité à l'inclure, car il est assez grand et d'un pas non-standard, mais je le préfère aux cavaliers (introuvables quand on en a besoin !).

Ensuite, il y a les types illustrés sur la **figure 6 B**. Ils utilisent des contacts en fil à ressort au lieu d'être de véritables interrupteurs, mais ils font le même travail et sont faciles à utiliser. De plus, ils peuvent supporter des courants plus élevés que les minuscules interrupteurs habituels.

Enfin, il y a le monstre de la **figure 7** : seize voies équipées d'inverseurs à trois positions (fermé côté 1, ouvert, fermé côté 2). Vous pouvez les utiliser pour attribuer à une voie l'une des valeurs 0, 1 ou haute impédance. Peut-être intéressant pour un analyseur logique ?

Les télécommandes utilisent souvent ce type d'inverseur. Le fait d'avoir trois positions pour chaque voie augmente considérablement le nombre de codes possibles – de $2^8$ (256) à $3^8$ (6561) pour un inverseur à 8 voies.

*(200716-04)*

**Des questions, des commentaires ?**
Envoyez un courriel à Elektor (redaction@elektor.fr).

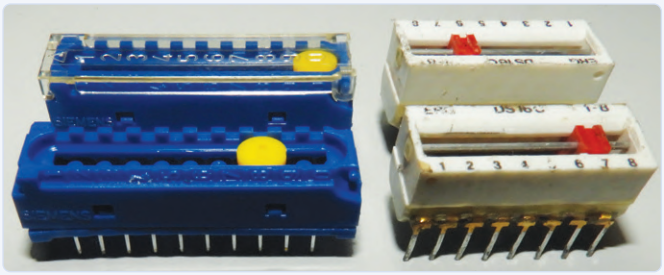


*Figure 4. Interrupteurs DIP de type « un parmi 8 ou 10 ».*

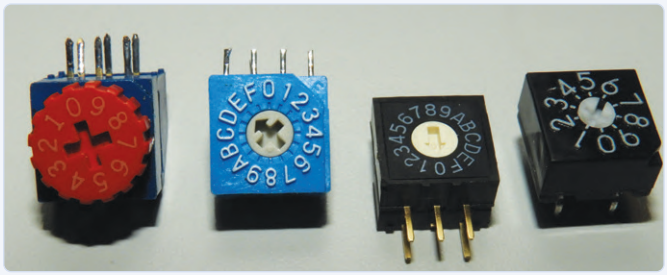


*Figure 5. Interrupteurs hexadécimaux et DCB.*

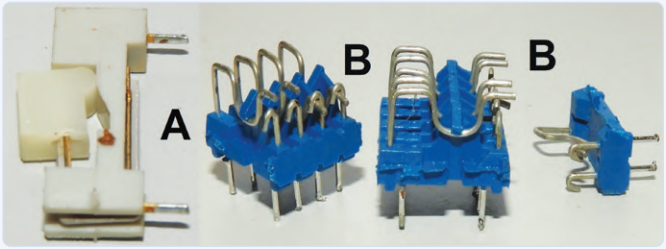



*Figure 6. Robuste coupe-circuit unipolaire ; modèle avec contact en fil à ressort.*

*Figure 7. Un commutateur DIP à trois voies 0/1/haute impédance.*

# corrections, mises à jour et courrier des lecteurs

Rédaction : **Ralf Schmiedel** & **Jens Nickel**

## Petit générateur de fonctions

**Elektor 07-08/2020, p. 20 (160548)**

Le schéma comporte une erreur : le potentiomètre P3 doit être câblé entre la broche de sortie 6 et la broche d'entrée 2 d'IC4, et non entre les broches 2 et 7. IC4 sert de tampon pour le signal de sortie à collecteur ouvert du LM311 et P3 commande le niveau du signal de sortie (Amplitude).

## Elektor Uno R4

**Elektor 06/2016, p. 40 (150790)**

L'ATmega328PB est maintenant entièrement pris en charge par la chaîne d'outils AVR installée par défaut dans l'EDI Arduino. Cela rend l'installation de l'Elektor Uno R4 beaucoup plus rapide, car il n'y a plus de chaîne d'outils spéciale à télécharger. La carte est également à l'épreuve du temps puisque les nouvelles fonctions de la chaîne d'outils sont utilisées automatiquement. Pour installer l'Elektor Uno R4, collez le lien

https://github.com/ElektorLabs/Arduino/releases/download/v1.0.1/package_elektor_uno_r4_1_8_x_index.json

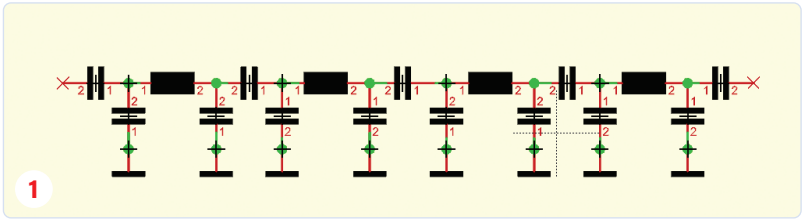
dans le champ « URL de gestionnaire de cartes supplémentaires » de l'onglet « Paramètres » des préférences de l'EDI. Dans le « Gestionnaire de carte » (sous Outils -> Type de carte), sélectionnez 'R4' et assurez-vous que la version est réglée sur '2.0.0'. Nous avons testé cela avec la version 1.8.13 de l'EDI Arduino.

## Conception de filtres analogiques (3)

**Elektor 01-02/2021, p. 74 (200522-02)**

J'ai lu avec grand intérêt l'article informatif d'Alfred Rosenkranzer. J'ai été frappé par le fait qu'il n'ait pas traité dans son article une configuration de filtre particulière. Elle présente d'excellentes caractéristiques, notamment la large sélectivité que l'on peut réaliser.

J'ai travaillé à la conception d'une SDR pour notre projet *Charly 25*, qui utilise une unité STEMlab 16 Red Pitaya comme cœur SRD dans un émetteur-récepteur. Nous avons utilisé des filtres passe-bande plus ou moins acceptables comme présélecteurs pour l'étage d'entrée. Les filtres passe-bande standard se sont avérés relativement médiocres en ce qui concerne le rejet hors bande réalisable. Notre objectif était d'atteindre > 80 dB à 100 MHz ; c'est nécessaire, car le convertisseur A/N est très bruyant et un récepteur sensible nécessite généralement un préampli large bande commutable qui, dans notre cas, offre un gain de +36 dB.



1

Dans notre recherche d'une meilleure structure de filtre, nous sommes tombés sur la configuration du « filtre tubulaire », principalement utilisée pour les applications à micro-ondes. Ce type de filtre est caractérisé par une série de réseaux pi à couplage capacitif (**1**).

Malheureusement, ce type de filtre a été largement ignoré par les programmes habituels de simulation de circuits gratuits – bien sûr nous ne pouvions pas accéder aux coûteux programmes commerciaux de simulation de circuits à micro-ondes. Les premières structures de filtres ont donc été créées par simulation et adaptation pas à pas à l'aide du logiciel de simulation de filtres Elsie. Un des membres de notre équipe a ensuite eu pitié de nous et a écrit un petit programme pour aider à calculer ce type de filtre :

https://charly25-sdr.github.io/hardware/rx-filters-v2/calculator

De plus, j'ai finalement réussi à trouver un programme gratuit « LC Filter Design Tool » qui modélise également ce type de filtre (voir « Tubular Filter ») :

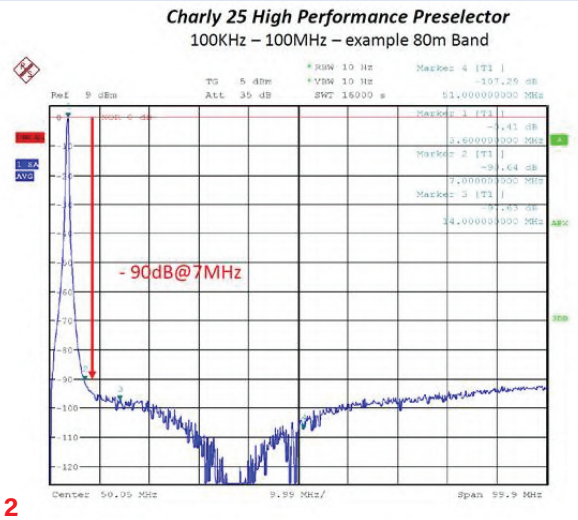
https://rf-tools.com/lc-filter/

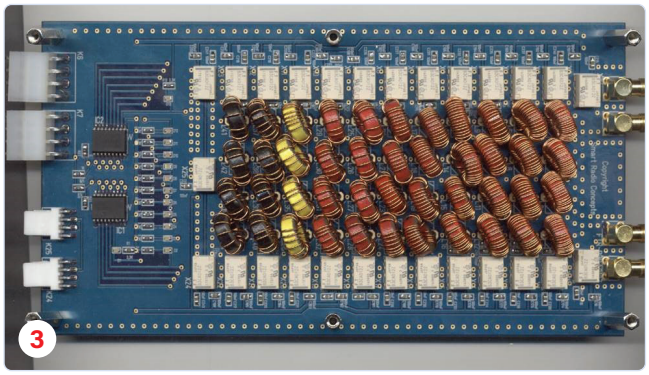
Le résultat de tous nos efforts est un tableau de présélection utilisable avec les onze bandes amateurs de 160 à 6 m (plus un débit non filtré pour la réception des émissions radio) qui dépasse largement nos exigences en matière de sélectivité (**2**).
Soulignons que les caractéristiques de cette carte ont été obtenues grâce à une disposition minutieuse des composants sur circuit imprimé à quatre couches, la configuration du plan de masse étant extrêmement critique. Une implantation plus simple sur un circuit imprimé à deux couches permet d'obtenir une atténuation hors bande d'environ 70 dB.
La carte de présélection complète est visible en **3.** Elle est commandée par un bus I2C qui la rend compatible avec la majorité des contrôleurs utilisés pour les projets de récepteurs. Pour plus d'informations sur la carte, voir

www.smartradioconcepts.com.

Pour la construction des bobines, il était clair que les noyaux en poudre de fer sont les plus adaptés à cette application et garantissent une faible distorsion d'intermodulation.
La carte de filtrage n'utilise pas de diodes PIN pour la commutation RF, là encore la possibilité d'effets d'intermodulation du signal était trop grande, d'où le grand nombre de relais.

*Edwin Richter (DC9OE)*




## Réception satellite fourrée dans un guêpier

Je profite actuellement de quelques moments de détente (bien mérités) dans la chaleur de l'État brésilien de Bahia. Cet état est peut-être connu pour être le berceau de la samba, mais la vie n'est jamais aussi simple et il m'arrive de devoir faire travailler mes méninges lorsque des problèmes techniques surviennent. Récemment, nous avons remarqué quelque chose d'étrange dans la réception de notre télévision par satellite (toujours analogique). Chaque soir, au moment du coucher du soleil, le signal devient bruyant et nous ne pouvons plus sélectionner certaines chaînes. Le matin, tout revient à la normale, sans problème.

Je sais de la diffusion terrestre que des perturbations atmosphériques peuvent se produire au coucher du soleil, mais dans la bande décimètre et en vue quasiment directe, ce qui me semblait peu probable. J'ai donc pris contact avec notre ingénieur en satellites local qui avait installé la parabole il y a quelques années.

Il n'a pas tardé à monter sur le toit et à remplacer le LNB corrodé. Le capuchon en plastique qui recouvre la gorge d'entrée du LNB était devenu cassant et était en partie tombé (très probablement dégradé par l'exposition aux UV et IR solaires). Il a également trouvé des preuves d'une certaine infestation de parasites… Il semble que certaines guêpes (plus grosses que les variétés européennes) aient construit un nid douillet dans le LNB. L'installateur avait déjà vu cela auparavant et a déclaré que comme les dernières nuits avaient été assez fraîches (20 °C) et que les LNB chauffaient pendant la journée, ces petites vermines s'envolent le matin et reviennent au coucher du soleil, où elles se mettent à l'aise et bloquent le chemin du signal. Problème résolu – pour l'équivalent d'environ 15 € !

*Wolfgang Meyer*

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

200715-04

# du tout-jetable au tout-réparable ?

## La réponse législative de l'UE

**Tessel Renzenbrink** (Pays-Bas)

Pourquoi réparer un appareil en panne plutôt que le remplacer ? La réponse est immédiate : parce que cela représente des économies d'énergie, de ressources, d'argent et, surtout, réduit le gaspillage. Réparer soi-même un appareil est de plus extrêmement satisfaisant. Ce qui semble relever de l'évidence fait pourtant l'objet d'appels citoyens au « droit à la réparation » et d'une volonté de la part de divers organes européens de légiférer sur la réparabilité. Le plus représentatif de ces efforts est une résolution du Parlement européen dressant une liste de mesures allant de l'extension des garanties à l'obligation de fournir des pièces de rechange.

### Obsolescence programmée

Selon une étude comportementale de 2018 publiée par la Commission européenne [1], la plupart des citoyens de l'UE préfèreraient réparer un appareil en panne plutôt que de le remplacer. Les chiffres vont de 62 à 83 % selon le type de l'appareil. Si les citoyens européens semblent donc mûrs pour passer d'une économie du prêt-à-jeter à une économie du prêt-à-réparer, de nombreux obstacles rendent cette transition difficile. Le plus cynique est sans doute l'obsolescence programmée, cette pratique consistant à fragiliser délibérément un appareil pour réduire sa fin de vie et augmenter son taux de remplacement. Dans le domaine du numérique, cette obsolescence peut consister à ne plus proposer de mises à jour logicielles ou de sécurité un an ou deux seulement après la mise sur le marché de l'appareil. Le produit devient de fait obsolète, même si le matériel fonctionne encore.

Une autre stratégie consiste à empêcher la réparation. La page du site iFixit [2] attribuant un indice de réparabilité aux ordiphones révèle à cet égard quelques techniques classiques, comme un boîtier trop bien collé, une batterie difficile à atteindre, l'indisponibilité des pièces de rechange ou leur coût prohibitif. Autre méthode, certains fabricants interdisent aux consommateurs et aux ateliers de réparation indépendants de se procurer le manuel d'un appareil en le déclarant propriété intellectuelle de leur entreprise.

### Indice de réparabilité

Le manque d'information rend également difficile la réparation d'un produit. L'étude de 2018 a révélé que lorsque des informations sur la réparabilité sont fournies sur le point de vente, un consommateur est « deux fois plus enclin (voire plus) à choisir le produit affichant le score de réparabilité le plus élevé ». Pour sensibiliser les consommateurs sur la possibilité d'allonger la durée de vie de leurs appareils, le ministère français de la Transition écologique a introduit en janvier 2021 un

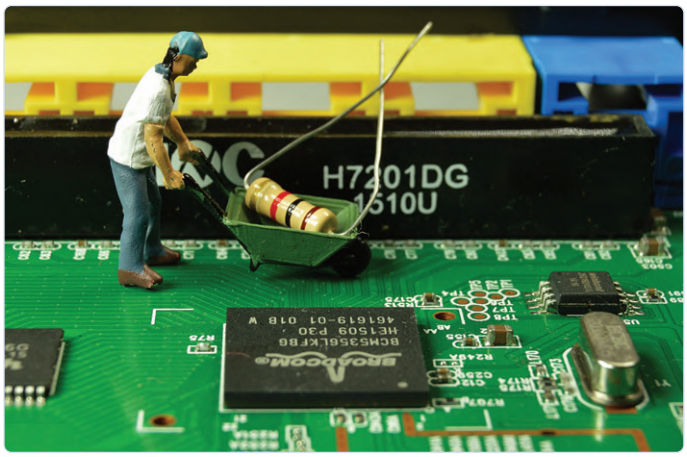



*Image de Jose Conejo Saenz, pixabay.com. Licence Pixabay.*

indice de réparabilité allant de 1 à 10 [3]. Le calcul de cet indice repose sur cinq critères, dont la facilité du démontage et la disponibilité des pièces de rechange. Son affichage est désormais obligatoire sur les lave-linges à hublot, les ordiphones, ordinateurs portables, téléviseurs et tondeuses à gazon électriques. D'autres catégories de produits seront concernées dans un second temps. L'objectif est d'atteindre un taux de réparation de 60 % des produits électroniques d'ici cinq ans.

### Vers un droit à la réparation européen

En novembre 2020, le Parlement européen a voté une résolution en faveur d'un marché unique plus durable [4]. Parmi les mesures deman-

dées figurent un droit à la réparation, l'accès gratuit aux manuels d'entretien et de réparation, la standardisation des pièces détachées pour favoriser l'interopérabilité des appareils, un prix raisonnable de ces pièces, et une période minimale obligatoire pour leur fourniture. Pour remédier à l'asymétrie de l'information, les vendeurs devront fournir des informations claires sur les caractéristiques du produit vendu, dont sa durée de vie estimée, sa réparabilité et la disponibilité de ses pièces de rechange. L'indice de réparabilité est proposé comme moyen d'atteindre cet objectif.

Pour lutter contre l'obsolescence programmée, le texte demande à la Commission d'engager une vaste stratégie assortie de mesures, comme l'interdiction d'introduire des fragilités dans un produit, ou l'obligation de fournir des mises à jour logicielles pendant toute la durée de vie estimée d'un appareil. Comme incitation à l'achat de biens de seconde main, le texte suggère que la garantie d'un produit soit transférable aux acheteurs successifs. Pour ne pas pénaliser les entreprises européennes soumises à ces mesures, le texte demande que les produits importés non conformes ne puissent pas être mis en vente.

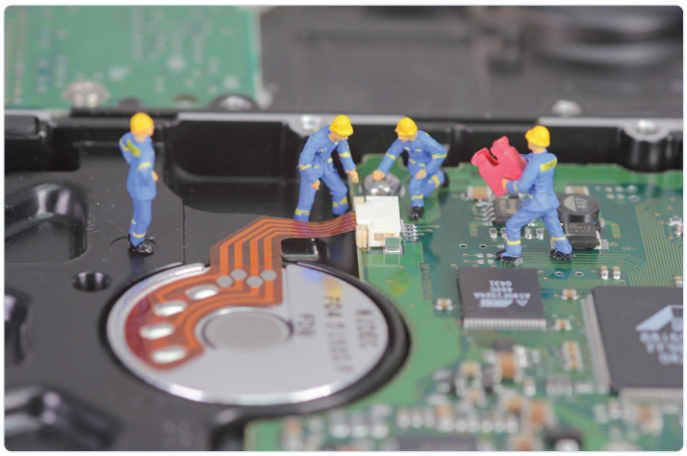


*Image de Wilfried Pohnke, pixabay.com. Licence Pixabay.*

> « Les citoyens européens semblent mûrs pour passer d'une économie du prêt-à-jeter à une économie du prêt-à-réparer. »

## Vers une économie circulaire

Le Parlement n'est pas le seul organe de l'UE à réclamer un droit à la réparation. En mars 2020, la Commission européenne a adopté un Plan d'action pour l'économie circulaire mentionnant également ce droit [5]. Parlement et Commission demandent ainsi tous deux que les mesures relatives au droit à la réparation figurent dans le Pacte vert pour l'Europe visant la neutralité climatique d'ici 2050. L'ambition est de transformer l'actuelle économie linéaire en économie circulaire durable, notamment en légiférant pour que les produits mis sur le marché de l'Union durent plus longtemps, soient plus faciles à réutiliser, à réparer et à recycler. Une économie durable est par nature plus équilibrée sur le plan écologique, mais offre aussi plus de résilience. Dans sa résolution, le Parlement souligne que la crise du Covid-19 a mis en lumière la fragilité des chaînes d'approvisionnement mondiales, et par là même celle du système économique actuel. Il suggère donc l'établissement de nouveaux modèles d'entreprise et un soutien aux PME européennes. Réparer sur place un produit plutôt que de le remplacer par un produit importé de l'étranger favoriserait l'essor des compétences et des économies locales.

Ces initiatives risquent d'être diluées en cours de route, et elles devront passer par les rouages législatifs de l'UE avant d'obtenir le statut de lois. Que le droit à la réparation soit inscrit dans le Pacte vert pour l'Europe, une des priorités de l'UE, est néanmoins très prometteur.

*(200713-04)*

*Traduction : Hervé Moreau*

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

## LIENS


[1] **Annette Cerulli-Harms et al., Behavioural Study on Consumers' Engagement in the Circular Economy, p. 11, octobre 2018** : https://bit.ly/39PvWkA

[2] **« La réparabilité des smartphones », iFixit** : http://bit.ly/2M8yZfJ

[3] **Ministère de la Transition écologique, Indice de réparabilité (janvier 2021)** : https://www.ecologie.gouv.fr/indice-reparabilite

[4] **« Vers un marché unique plus durable pour les entreprises et les consommateurs », résolution du Parlement européen du 25 novembre 2020** : http://bit.ly/2NeJcaL

[5] **« Changer nos modes de production et de consommation : nouveau plan d'action pour l'économie circulaire », Commission européenne, mars 2020** : http://bit.ly/395SdeE

# point d'omelette sans casser d'œufs

## Le grand livre des gaffes

**Ilse Joostens** (Belgique)

La vie d'un passionné d'électronique n'est pas toujours un long fleuve tranquille, qu'il soit amateur débutant ou vieux de la vieille. Que cela nous plaise ou non, la technique évolue sans cesse et il faut continuer d'apprendre pour être au fait des derniers développements. Tomber et se relever sont à la base de l'apprentissage des choses les plus importantes, car cela laisse des traces durables souvent associées à de forts sentiments de déception, d'échec et même de grande honte. À moins d'être un « enfant gâté » à qui tout « tombe tout rôti dans l'bec » [1], ce processus vous conduira nécessairement à apprendre de vos propres erreurs et vous vous en sortirez au mieux. Et si les choses ont vraiment dérapé, vous pourrez toujours rassembler vos forces pour gérer la crise et, en votre for intérieur, rejeter la faute sur quelqu'un d'autre.

### Chute de couteaux et pyrotechnie

Quand j'étais encore jeune, dans les années 80 du siècle dernier, la connaissance de l'électronique était peu accessible par rapport aux possibilités actuelles. Les ordinateurs n'étaient pas courants et l'internet était en gestation. J'ai donc dû me contenter pour l'essentiel d'un vieux livre sur les principes de l'électronique et de quelques petits ouvrages obscurs trouvés ici ou là. La bibliothèque locale n'abritait qu'un faible nombre de livres sur l'électronique et les recueils de caractéristiques (*data books*) étaient réservés aux ingénieurs en électronique (des gens à l'air sévère, à la barbe grise et en blouse de laboratoire).

Je me souviens que j'expérimentais avec enthousiasme sur des diodes, des lampes et un transformateur de train électrique miniature. En inversant la polarité ou en utilisant le courant alternatif, je pouvais contrôler indépendamment deux jeux de lampes avec seulement deux fils, jusqu'à ce que (bien sûr) j'aie la mauvaise idée d'essayer cela sur le secteur. Pour mon premier essai, j'avais construit un petit pont redresseur alimentant une petite ampoule. Je me souviens très bien que j'ai vu la lampe s'allumer une fraction de seconde avant que le pont redresseur ne m'explose bruyamment au visage dans un éclair brillant et une gerbe d'étincelles, tel un feu d'artifice. Comment aurais-je su que les diodes avaient une limite de tension inverse, et que dans ce cas, je l'avais largement dépassée ? Le disjoncteur de la maison a sauté. Que dire de la panique dans les yeux de mes colocataires ?

Un an plus tard, j'ai acheté mon premier fer à souder, genre tisonnier à manche de bakélite, au magasin de bricolage du coin. Mes premières soudures étaient plutôt « grumeleuses », mais j'étais heureuse de ne plus devoir improviser les connexions. À l'époque, les tourne-disques avec amplificateur intégré étaient en vogue et à un moment donné, j'ai promis à une jeune voisine de réparer le sien. Quelque part à l'intérieur, un fil était dessoudé. Je n'avais pas beaucoup de place, et juste à la fin, le fer à souder a glissé de la table, par réflexe j'ai voulu le rattraper, bien sûr du côté chaud… L'adage dit « ne jamais essayer d'attraper un couteau qui tombe », j'ai constaté à mes dépens et à ma grande honte que cela valait aussi pour les objets brûlants comme les fers à souder. Le résultat a été une odeur de cochon grillé et une belle brûlure à la main droite. J'ai donc tenté d'oublier la douleur et j'ai terminé le travail tout en attrapant tout ce qui était un tant soit peu froid pour la soulager, entre autres des piles alcalines de type D qui se trouvaient là par hasard et que je prenais à tour de rôle dans la main. Quelques semaines plus tard, le tourne-disque a eu de nouveau des soucis et quelqu'un d'autre y a jeté un coup d'œil, après quoi on m'a demandé pourquoi j'avais mis une pile dedans…

Nous avions une antique perceuse sans variateur de vitesse et le « variateur de vitesse de perceuse » du livre *302 circuits* d'Elektor semblait être la solution. À l'époque, je gravais déjà moi-même des circuits imprimés grâce à un stylo dont l'encre résistait à la gravure. J'ai construit le tout en vitesse et l'essai fut concluant. Malheureusement, ma capacité financière étant limitée, j'ai souvent utilisé des composants « recyclés » d'origine douteuse. Un peu plus tard, l'un des condensateurs a claqué en plein travail et une fumée nauséabonde a instantanément envahi tout l'atelier. Une fois la fumée quelque peu dissipée, le vilain condensateur neutralisé à l'aide d'une bombe anti-odeur et remplacé, le variateur a continué à rendre de loyaux services pendant des années.

### Défiez-vous de la simplicité apparente

J'ai aussi une longue expérience professionnelle des incidents, mais l'espace alloué à cet article est malheureusement trop limité pour les couvrir tous. Je ne m'étendrai donc pas trop sur la fois où sans

arrêt distraite par des coups de téléphone de clients, et sans faire gaffe (si, si j'ai osé le jeu de mots !) j'ai coupé le cordon d'alimentation d'un appareil sans l'avoir au préalable débranché. Une grande partie de l'entreprise fut plongée dans le noir. Que dire de la fois où je me suis percé la main avec une perceuse d'établi ; de celle où, là encore distraite par une conversation, j'ai touché la phase et le neutre du secteur et que la brutale commotion électrique et la réaction à celle-ci m'ont fait voler à deux mètres de là, avec la chaise de bureau et tout le reste, et atterrir durement contre un classeur. Encore aujourd'hui, les choses apparemment les plus simples sont justement celles qui me font trébucher. Placer un circuit intégré de plus de cent broches lors de la conception du circuit imprimé est un jeu d'enfant, mais mettre un connecteur n'ayant que quelques broches dans le bon sens du premier coup ça n'arrive jamais ! Eh oui, la polarité des objets sera toujours ma bête noire…

Juste pour finir, j'aimerais vous inviter à partager vos gaffes personnelles et situations embarrassantes avec la communauté d'Elektor en laissant un commentaire sur cet article. Il n'y a pas de quoi avoir honte… Bon, d'accord, peut-être un peu, mais la raillerie est aussi un vrai plaisir.

(200682-04)

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


**Contributeurs**

Texte et choix de l'illustration : **Ilse Joostens**
Rédaction : **Eric Bogers**
Mise en page : **Giel Dols**
Traduction : **Yves Georges**


## WEB LINKS

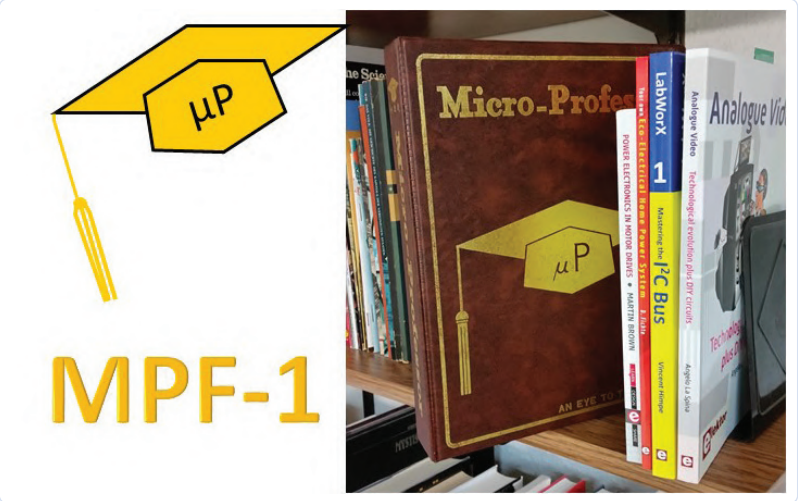

[1] **Gilbert Richer, « Par le bout du nez – La psychologie de l'enfant-roi », éditions Viamédias**

[2] **Compilation amusante d'ElectroBOOM :** www.youtube.com/watch?v=gAnBc4iFCvk

[3] **Chaîne YouTube ElectroBOOM :** www.youtube.com/c/Electroboom/featured

# Micro-Professor

## Apprentissage de l'assembleur sur Z80




**Karl-Ludwig Butte (Allemagne)**

En 1981, beaucoup d'ouvrages étaient consacrés au *Z80*, le microprocesseur de *Zilog* introduit en 1976. Le *Micro-Professor* était différent. Avec une épaisseur de 4 cm, sa documentation sûrement indigeste et « microprocessorale » promettait des centaines de pages d'informations concentrées, donc de nombreuses heures d'étude fastidieuse. Que nenni, la surprise devait être totale : en l'ouvrant, le « lecteur » faisait face au microprocesseur lui-même, monté sur une carte prête à l'emploi ! Si cela vous inspire, venez avec moi à la découverte du *MPF-1* de 1981, la formation Z80 avancée de *Multitech*.

Dans un secteur industriel qui progresse aussi rapidement, une formation continue tout aussi rapide est la seule façon d'éviter d'être rapidement dépassé. *Stan Shih*, son épouse *Carolyn Yeh* et cinq autres associés en étaient convaincus. Ils ont fondé la société *Multitech* à *Taipei (Taiwan)* en 1976 – aujourd'hui mieux connue sous le nom d'*Acer*. Multitech commercialisait des pièces détachées électroniques, faisait du conseil en micro-informatique et fabriquait des systèmes didactiques à microprocesseur dans ses propres usines. Multitech connaissait déjà bien les systèmes de formation aux microprocesseurs avant de lancer le *Micro-Professor* MPF-1 (**fig. 1**) en 1981.

### Le matériel

Le MPF-1 était un ordinateur monocarte doté d'un écran à six chiffres de sept segments et d'un clavier de 36 touches. Il reposait sur un microprocesseur *Z80* de *Zilog* cadencé à 1,79 MHz. La **figure 2** montre l'agencement de la carte. Le µP Z80 était accompagné d'un circuit intégré (CI) *PIO* Z80, d'un CI *CTC*, d'un CI *8255*, de 2 Ko de RAM et de 4 Ko d'EPROM. Le 8255, un composant d'E/S parallèles programmables, reliait le clavier et les afficheurs à 7 segments au processeur. Il faisait aussi l'interface avec


*Figure 1. Le Micro-Professor MPF-1B dans son livre-boîtier ouvert et la documentation y afférente.*

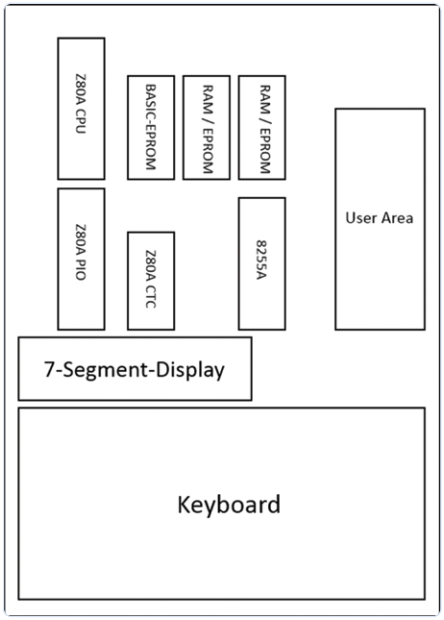



*Figure 2. Agencement de la carte MPF-1B.*

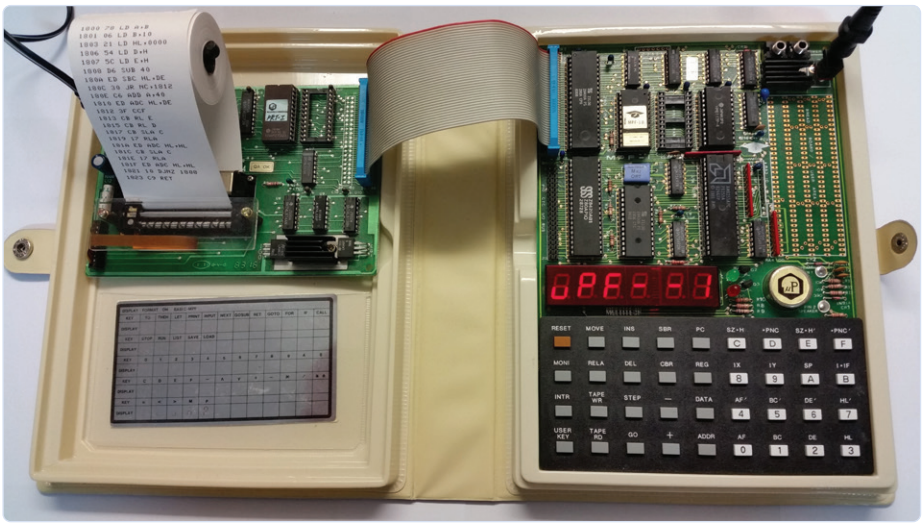


*Figure 3. Le MPF-1B connecté à sa carte imprimante thermique.*

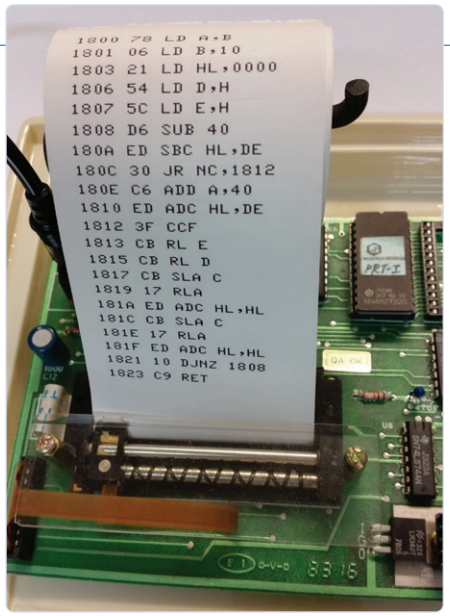



*Figure 4. Listage assembleur du programme de racine carrée.*

le haut-parleur intégré et le connecteur du magnétophone à cassettes, pour stocker données et programmes sur bande audio - l'option « mémoire de masse » la moins coûteuse à l'époque. Le Z80-CTC était un CI de comptage/temporisation abritant quatre compteurs et temporisateurs programmables. Enfin, le Z80-PIO était un CI d'entrées/sorties parallèles d'interfaçage de périphériques. Multitech proposait à cet effet une imprimante thermique (voir **fig. 3**) et un programmateur EPROM, ainsi qu'une carte de sortie vocale. D'autres sociétés fabriquaient ou vendaient des extensions pour le Micro-Professor. Par ex., à Constance (Allemagne) l'institut d'enseignement à distance *Christiani* a enrichi le Micro-Professor de sa propre carte d'E/S spécialement conçue pour l'enseignement « par correspondance ».

## Documentation

La documentation du Micro-Professor (fig. 1) était tout aussi minime que la configuration matérielle, fort loin du guide d'utilisation de l'ordinateur d'apprentissage *CP1* de *Kosmos*, basé à *Stuttgart*, décrit dans l'édition de juillet/août 2018 d'Elektor [1]. Ce n'est pas une coïncidence si un établissement renommé tel que Christiani a repris le Micro-Professor et publié son propre cours à vocation pédagogique. Multitech fournissait un manuel d'utilisation de seulement 95 pages pour le MPF-1. Au moins, celui fourni avec mon Micro-Professor était en allemand. Le manuel d'utilisation du *BASIC-MPF* inclus avec le Micro-Professor ne comptait que 39 pages. Il n'était disponible qu'en version originale anglaise, de même que le *manuel d'expérimentation du MPF-1*.

Le ou les auteurs supposaient que leur lecteur avait une bonne connaissance de base des microprocesseurs. Par ex., après les caractéristiques techniques et une brève description des différentes touches, le manuel d'utilisation du MPF-1 commence par les phrases suivantes : « *Le programme du moniteur inclut toutes les routines de service pour faciliter la tâche de l'utilisateur : (1) chargement des programmes dans la RAM, suivi d'un test et/ou d'une modification* ». Bien que l'essentiel, c.-à-d. le jeu d'instructions du Z80, son architecture de programmation, la structure de la mémoire et l'affectation des d'E/S, figurât plus loin dans le manuel, c'était seulement sous forme de tableaux sans aucune explication. Il était donc conseillé de s'armer d'un ouvrage de référence approprié, par ex. *How to Program the Z80 (Comment programmer le Z80)* de *Rodnay Zaks*. Il avait beaucoup plus à dire sur le Z80. Son livre faisait plus de 600 pages.

## Programmation en assembleur

Pour me familiariser avec le travail sous l'« égide » du Micro-Professor, j'ai pris l'exemple du programme *Square Root* du manuel. Heureusement, il a été imprimé dans la notation abrégée de l'assembleur Z80, mais aussi en hexadécimal car lui seul peut être entré au clavier. Il faut d'abord appuyer sur la touche ADDR et entrer l'adresse de départ, puis la touche DATA fait passer en mode saisie de données et on peut entrer le premier octet du programme. Ensuite, il faut appuyer sur la touche + pour passer à l'octet suivant et le taper directement. Le programme peut être saisi assez rapidement de cette façon. Pour vérifier le programme, je l'ai imprimé à l'aide du *Disassembler Listing Utility (Utilitaire de désassemblage-listage du code)*, fourni par l'imprimante thermique connectée à partir de l'adresse $6000_{hex}$ (**fig. 4**).

Après avoir vérifié que l'impression correspondait au listage du manuel, j'ai fait un essai. La tâche du Micro-Professor était de calculer la racine carrée de 81. Le programme est censé trouver ce nombre (en format hexadécimal, bien sûr) dans le registre BC du processeur. En utilisant les clés *REG* et *BC*, j'ai pu visualiser la valeur de ce registre, et avec la clé *DATA* entrer le premier octet (BC est un registre de 16 bits). À ce stade, j'étais confronté cette question : quel octet saisir en premier, $51_{hex}$ (équivalent à $81_{dec}$) ou $00_{hex}$ ? Guidé par le principe selon lequel la pratique vaut mieux que la théorie, j'ai d'abord essayé $00_{hex}$ suivi de $51_{hex}$, ce qui a donné un résultat clairement erroné. J'ai donc dû recommencer et taper la séquence REG, BC, DATA, 5, 1, +, 0, 0 (**fig. 5a**). Pour lancer l'essai, il fallait régler l'adresse sur $1800_{hex}$ et appuyer sur la touche GO (**fig. 5b**). Suspense insoutenable : j'ai vérifié

## ENTRETIEN AVEC MAX D. SOFFE, DIRECTEUR GÉNÉRAL DE FLITE ELECTRONICS INTERNATIONAL LIMITED

Max D. Soffe est membre du conseil consultatif industriel du département d'ingénierie électronique de Royal Holloway, Université de Londres et de l'Institut de technologie de l'ingénierie (www.theiet.org). Président de l'Association des fabricants en ingénierie des techniques éducatives (ETEMA), Grande-Bretagne de 2000 à 2004, il a aussi été le directeur de l'Association britannique des fournisseurs de l'enseignement (BESA).

***Karl-Ludwig Butte :*** Flite Electronics International a acheté les droits de propriété intellectuelle du Micro-Professor MPF-1B à Acer. Quelle en était la raison ?

***Max D. Soffe :*** Acer s'appelait autrefois Multitech. Je les ai croisés lors de mon 1er voyage au Japon en 1981. Le MPF-1B Z80 *Micro Processor Trainer* était exposé à Tokyo. J'ai eu la chance de rencontrer les fondateurs de Multitech et de me lier d'amitié avec eux. L'entreprise ne comptait alors que 20 personnes. Elle était basée dans le parc scientifique Hsinchu, à Taipei. J'ai fait une double page de publicité dans le magazine « Wireless world » au Royaume-Uni et j'ai vendu une centaine de MPF-1B en trois semaines. Multitech m'a accordé les droits de distribution au RU, bien que nous vendissions déjà dans le monde entier.
Acer a ensuite voulu augmenter son chiffre d'affaires et se concentra avec un grand succès sur le marché en plein essor des PC. Les Z80 éducatifs étaient en fait déficitaires. En 11 ans, Flite Electronics vendit des milliers de MPF1-B. Le produit étant trop cher pour un usage privé, nos clients étaient surtout des établissements d'enseignement, universités et collèges. Nos ventes étaient toujours bonnes et j'ai négocié avec Acer le rachat des droits de propriété intellectuelle. Ce contrat a été signé et scellé en février 1993.
Le succès n'a pas été aussi grand que je l'aurais souhaité, car Acer s'est désintéressé du produit, en nous laissant le soin d'arrêter son « piratage ». Nous n'avions pas les moyens de lutter contre trois ou quatre sociétés qui se contentaient de tout copier : manuel, circuit imprimé et micrologiciel. Malgré cela, nous avons continué en remaniant le circuit imprimé et en conservant le micrologiciel. Les composants se sont raréfiés et il y a eu des problèmes avec le processeur Z80 (déclaré), l'EEPROM et la vitesse de la RAM.

***Karl-Ludwig Butte :*** sur l'internet, certains sites (par exemple https://en.wikipedia.org/wiki/Micro-Professor_MPF-I) disent que le Micro-Professor est encore fabriqué et distribué par Flite Electronics International aujourd'hui. Est-ce exact ? Et, si oui, où peut-on le commander et à quel prix ?

***Max D. Soffe :*** oui, bien sûr, nous continuons à fabriquer le MPF-1B, en très petites quantités. Le dernier lot fabriqué, une commande d'une université du Moyen-Orient, remonte à environ six mois. Les professeurs formés au Royaume-Uni ont utilisé le MPF-1B il y a 20 ans. Ils enseignent toujours l'assembleur et le codage hexadécimal.

Il faudra un peu de patience, mais nous répondrons et enverrons un devis aux lecteurs d'Elektor qui écriront à sales@flite.co.uk.


*Figure 5. A : entrée de $81_{dec}$ dans le registre BC. B : adresse de départ. C : résultat $9_{dec}$ dans le registre DE.*

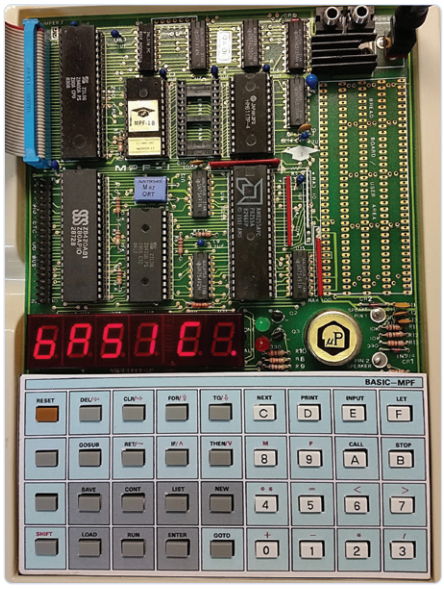


*Figure 6. Le Micro-Professor en mode BASIC et son cache de fonctions spécifiques.*

le résultat dans le registre DE, et… ouf ! Le joli chiffre « 9 » (**fig. 5c)** s'est allumé.

### L'interpréteur BASIC

La commercialisation du *MPF-1B* a succédé rapidement à celle du MPF-1, rebaptisé MPF-1A pour le distinguer de la version B. Que le MPF-1B pouvait-il faire de plus que son prédécesseur ? Eh bien, Multitech avait ajouté une nouvelle *EPROM* abritant un petit interpréteur *BASIC*. Je voulais naturellement l'essayer, car le BASIC avec écran de 6 chiffres à 7 segments et clavier hexadécimal était quelque chose de totalement nouveau pour moi. Le MPF-1B était livré avec un cache à superposer au clavier pour le BASIC (disparu de mon appareil depuis longtemps, mais aujourd'hui, l'internet résout ce genre de problème). J'ai vite trouvé un fac-similé et l'ai imprimé sur du papier assez fort.

Nous en avons huit en stock. Une fois ces unités vendues, il peut s'écouler deux à trois mois avant la fabrication d'un autre lot. Toutefois, quelques unités d'occasion sont en vente sur eBay.

Comme je l'ai dit, les composants anciens sont maintenant relativement chers. Le code de date de production d'un circuit intégré (CI), par ex. 9243, indique qu'il a été fabriqué en 43e semaine de 1992. Ainsi, nous devons passer beaucoup de temps en essais des composants et du produit fini (fréquence, vitesse) avec des défaillances pour cause d'incompatibilité. C'est pourquoi tous les CI sont sur support : nous pouvons les changer en un clin d'œil. Il va de soi que lorsque nous expédions une nouvelle unité, elle est irréprochable et garantie un an.

Bien sûr, nous réalisons qu'un tel produit ne peut prétendre être un modèle de réussite commerciale. Il fut un tremplin pour Acer et lui a permis de devenir un acteur international de poids, mais le Micro-Professor ne nous fera pas plus réaliser de bénéfices qu'à Acer. En effet, nous ne pouvons pas vendre ce produit à moins de 250 livres sterling (+TVA et frais de port).

***Karl-Ludwig Butte :*** quelle documentation livrez-vous avec le Micro-Professor ?

***Max D. Soffe :*** la documentation est très importante. Chaque Micro-Professor est accompagné d'un cahier pratique. Ce manuel doit permettre à l'étudiant de déballer et mettre sous tension le microsystème, mais aussi d'en comprendre l'architecture, le matériel, l'interfaçage et le logiciel. Au départ, il devait s'agir d'un manuel d'autoapprentissage à la programmation hexadécimale. Nous le savons tous, les ordinateurs travaillent en binaire et le niveau suivant est l'hexadécimal. À l'époque l'atout du codage hexadécimal résidait dans le prix et la taille de la mémoire. Le Micro-Professor apprenait à écrire un code très compact, et non pas un code bâclé comme c'est souvent le cas de nos jours. En effet, il ne s'agit pas vraiment de coder en déplaçant des symboles graphiques.

***Karl-Ludwig Butte :*** quelle est l'activité première de Flite Electronics International ?

***Max D. Soffe :*** en raison du piratage du Micro-Professor des années 1990, Flite Electronics s'est associée à K&H, une société taïwanaise qui fabrique du matériel pédagogique. Notre clientèle était en effet principalement composée d'établissements d'enseignement, et non de particuliers. Nous sommes les distributeurs des produits didactiques de K&H au Royaume-Uni. Nous avons récemment installé un laboratoire de machines électriques d'une valeur de 300 000 €. Ce système, nous ne l'avons pas conçu, il se compose de modules K&H existants. La semaine dernière, nous avons ajouté une extension de 60 000 € à ce laboratoire.

Nous restons le fournisseur d'organismes de formation aux anciens µP tels que le 68000 de Motorola, le 8086 et le 8032 d'Intel ; là encore, nous en vendons dix par an, surtout pour aider des clients et organismes existants et nous faisons des devis avec plaisir. Dans les années 1990, nous avons conçu un produit pour compléter tous nos systèmes de formation aux microprocesseurs : un laboratoire tout-en-un appelé APB (*applications board*). Il est aussi accompagné d'un manuel de formation.

***Karl-Ludwig Butte :*** merci beaucoup pour cette interview.

Découper les trous avec un scalpel m'a pris du temps. Voyez le résultat sur la **fig. 6**. Comme exemple, j'ai choisi une routine du manuel d'utilisation du *BASIC-MPF* qui peut générer plusieurs tonalités de signal. L'interpréteur BASIC en mémoire a une adresse de départ de $0800_h$ et peut être lancé avec la commande GO. Il annonce sa présence avec « BASIC » sur l'écran, comme le montre la figure 6. Le programme peut alors être saisi : d'abord le numéro de ligne puis l'instruction BASIC, suivie des variables, nombres ou paramètres. Chaque instruction BASIC jouit de sa propre touche de saisie. Pour finir, appuyer sur la touche ENTER, comme autrefois avec un Apple II, un PET 2001 ou un TRS-80. Tout a bien fonctionné jusqu'à ce que je veuille entrer l'instruction `FOR A=1 TO C`. Au lieu du signe égal, j'avais toujours un 5 à l'écran, bien que j'aie appuyé sur la touche SHIFT en premier. Cela venait d'une vieille habitude : le clavier du Micro-Professor ressemble tellement au clavier d'une calculette que j'appuyais sur la touche SHIFT et la relâchais au lieu de la maintenir, comme pour un PC. Je n'ai pas réalisé tout de suite que le Micro-Professor se prenait pour un PC et qu'il fallait maintenir cette touche SHIFT enfoncée lors de la pression sur la touche =.

Après avoir surmonté cet écueil, il me fut facile de saisir le reste du programme. La **figure 7** montre quelques exemples de la façon dont les instructions BASIC sont représentées sur les LED à 7 segments. La **figure 7a** est `10 Input C` en notation normale, tandis que la **figure 7b** représente la ligne `50 Next B`. Le manuel contient un tableau des instructions BASIC et de leurs représentations respectives sur l'écran.


*Figure 7. A : 10 input C. B : 50 next B. C : Invite de saisie.*

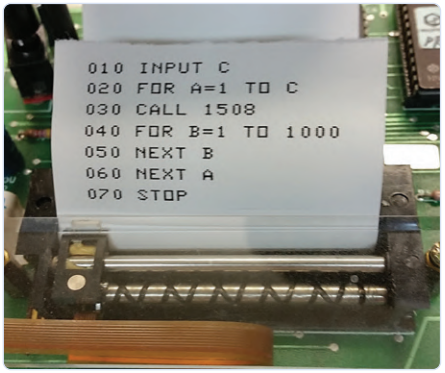



*Figure 8. Impression thermique du programme BASIC d'exemple.*

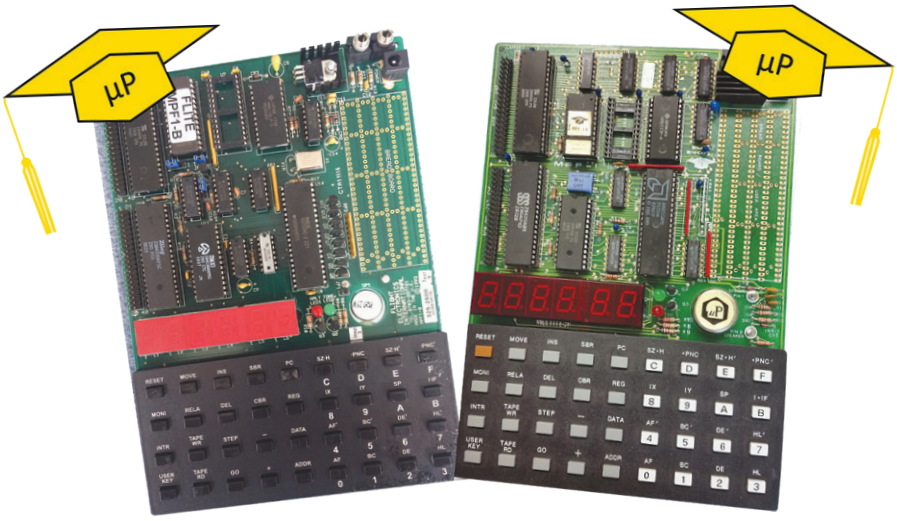



*Figure 9. Le Micro-Professor d'Flite Electronics International (à gauche) et l'original de Multitech (à droite).*

C'est nécessaire, car dans certains cas, c'est difficile à deviner. L'édition est également possible, mais alors, l'éditeur *vi* du monde *Unix* semble très convivial.

Il est temps d'essayer le programme BASIC tout frais saisi. Après l'avoir lancé par la touche RUN, il demande d'abord le nombre de signaux sonores à générer (**fig. 7c**). J'ai entré 3 et appuyé sur ENTER, puis j'ai entendu trois tonalités relativement aiguës, exactement comme prévues.

Avec un programme plus long, il y a de fortes chances de faire des fautes de frappe. À mon avis, la recherche de fautes de frappe sur un affichage « cryptique » à 7 segments doit être fastidieuse. Heureusement, avec une imprimante connectée, les programmes BASIC peuvent être imprimés en texte normal. La **figure 8** donne le listage du programme de signaux sonores.

## Un effort méritoire

Une chose est sûre : après avoir travaillé intensivement avec le Micro-Professor, vous connaîtrez intimement chaque élément du microprocesseur Z80 et de ses périphériques. Cette voie est semée d'embûches, il vous faudra persévérer, mais à mon avis, cela en vaut la peine. Il est étonnant de constater à quel point les concepts sont similaires à ceux de la série « Cours intensif d'assembleur » de Miroslav Cina publiée dans *Elektor* à partir de l'édition de juillet/août 2015 [2].

Au cours de mes recherches sur l'internet, j'ai trouvé des indications selon lesquelles le Micro-Professor MPF-1B est toujours fabriqué et vendu. Sur ces indications, j'ai contacté la société Flite Electronics International Ltd à Waltham Chase, Hampshire, Grande-Bretagne [3]. Elle m'a confirmé qu'elle avait acquis les droits sur le Micro-Professor et qu'elle continuait à le fabriquer. La **figure 9** montre le Micro-Professor de Flite Electronics côte à côte avec son homologue taïwanais de Multitech. Flite Electronics m'en a généreusement fourni un spécimen, et je peux confirmer qu'il est d'excellente qualité. Bien sûr, je voulais en savoir plus, et Max D. Soffe, directeur général de Flite Electronics, m'a aimablement accordé un entretien (voir encadré).

(200679-04)

**Des questions, des commentaires ?**
Envoyez un courriel à l'auteur (LKL_Butte@web.de) ou contactez Elektor (redaction@elektor.fr).



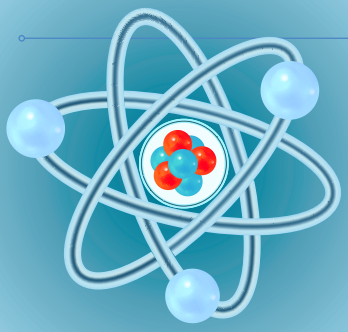
**Contributeurs**
Texte et illustrations : **Karl-Ludwig Butte**
Rédaction : **Jens Nickel**
Mise en page : **Giel Dols**
Traduction : **Yves Georges**

## LIENS

[1] **Karl-Ludwig Butte, « Initiation aux microprocesseurs avec le Kosmos CP1 (1983) », Elektor 07-08/2018** : www.elektormagazine.fr/160699

2] **Miroslav Cina, « Cours intensif d'assembleur (1) », Elektor 07-08/2015** : www.elektormagazine.fr/130483

3] **Flite Electronics International Limited** : http://www.flite.co.uk



# démarer en électronique… (7)

## …est moins difficile qu'on ne l'imagine ! Même lorsqu'il s'agit de condensateurs.

**Eric Bogers** (Elektor)

Après la présentation des résistances dans l'épisode précédent [1], passons aux condensateurs. Les condensateurs peuvent également être considérés comme des résistances, mais pour les signaux en courant alternatif. En effet, les condensateurs bloquent les signaux continus, mais offrent un passage aux signaux alternatifs. Outre leur usage comme résistances en alternatif, ils peuvent également servir à stocker de l'énergie électrique.

Un condensateur peut être considéré comme deux conducteurs (des plaques métalliques, par exemple, ou plus généralement des électrodes) placés à une très courte distance l'un de l'autre, mais sans aucun contact électrique entre eux. Leur symbole (**fig. 1**) traduit cette caractéristique : deux traits parallèles disjoints.

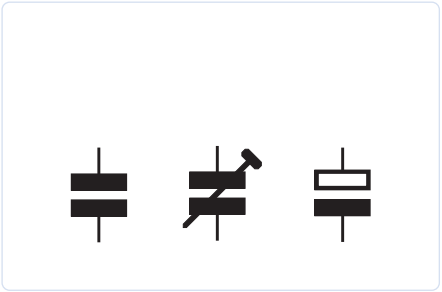


*Figure 1. Symboles utilisés pour les condensateurs. De gauche à droite : condensateur « ordinaire », condensateur ajustable (« trimmer »), condensateur électrolytique.*

Mais attendez ! Comment est-il possible qu'un composant bloque les signaux continus, mais pas les signaux alternatifs ? Pour le comprendre, rappelons-nous ce qu'est en réalité le « courant » : un déplacement de charges. Lorsqu'une charge est déplacée, un courant circule. Lorsque nous connectons un condensateur à une tension continue, il s'établit un courant initial – les électrons se déplacent d'une plaque du condensateur vers la borne positive de la source de tension continue et de la borne négative vers l'autre plaque. Mais une fois le condensateur chargé (la tension entre ses plaques est égale à celle de la source), le courant s'arrête, le condensateur se comporte comme un circuit ouvert (ou isolateur, ou résistance d'une valeur infiniment grande).

Il en va autrement lorsque le condensateur est soumis à une tension alternative : il est alors continuellement chargé, déchargé, chargé à l'envers, etc. Mais là aussi aucun électron ne *traverse* réellement le condensateur d'une plaque ou d'une électrode à l'autre !

### Capacité

Comme déjà indiqué, un condensateur peut stocker une charge (et donc de l'énergie électrique) et d'autant plus que sa *capacité* est grande.

Pour la capacité C d'un condensateur, on a la relation :

$$C = \frac{\varepsilon \cdot A}{d}$$

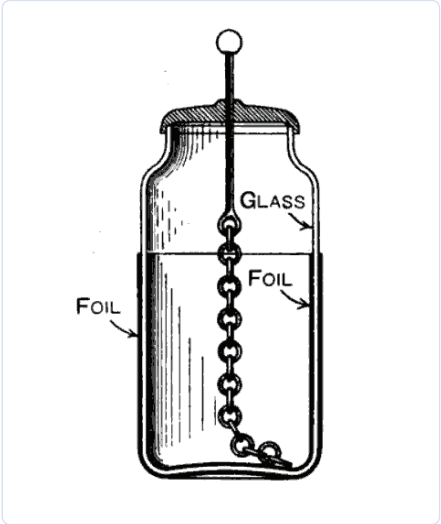



*Figure 2. Structure d'une bouteille de Leyde (source : Wikimedia Commons, http://bit.ly/wiki-leyden-jar).*

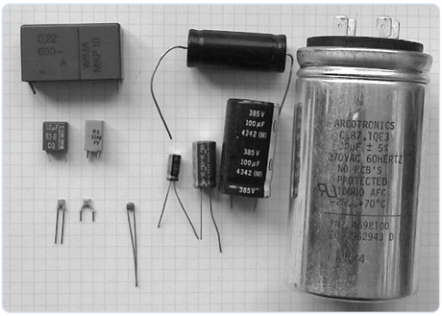


*Figure 3. Quelques types de condensateurs courants.*

où ε est la *constante diélectrique* de l'isolant entre les électrodes (« plaques »), *A* leur surface et *d* la distance entre elles. Il n'est guère utile de se souvenir de cette formule, elle n'est là que pour vous donner une idée des grandeurs qui déterminent la capacité d'un condensateur.

Il est évident que la capacité augmente avec la surface des électrodes (la charge est constante par unité de surface). Lorsqu'on rapproche les électrodes, la capacité augmente également. La constante diélectrique, qui a une grande influence sur la capacité, nécessite un peu plus d'explications. Entre les électrodes d'un condensateur se trouve un matériau isolant (le diélectrique). Pour la constante diélectrique ε, nous avons la relation suivante :

$$\varepsilon = \varepsilon_0 \cdot \varepsilon_r$$

où $\varepsilon_0$ est la permittivité du vide (une constante naturelle avec une valeur extrêmement faible de 8,85*10$^{-12}$ C2/N·m), et $\varepsilon_r$ est la constante diélectrique relative de l'isolant, qui est une caractéristique du matériau. Sa valeur est de 1 pour le vide et l'air, de 2 pour le papier et de 7 pour la céramique et le mica. On peut donc considérablement augmenter la capacité d'un condensateur par un choix judicieux du diélectrique.

L'unité de capacité est le *farad* (symbole F). Cette unité est beaucoup trop grande en pratique, où nous avons affaire à des valeurs de l'ordre du picofarad (10$^{-12}$ F), du nanofarad (10$^{-9}$ F) et du microfarad (10$^{-6}$ F). Il existe cependant des condensateurs d'une valeur de 1 à 10 F, voire plus : ce sont les « *supercaps* », disponibles auprès de différents fabricants, qui remplissent la fonction de batteries de secours (par exemple pour protéger un circuit d'horloge en temps réel des pannes de courant).

## Entrée en scène

Le condensateur a fait ses débuts en 1746 dans le laboratoire de Pieter van Musschenbroeck à l'université de Leyde (Pays-Bas). Ce condensateur était constitué d'un bocal en verre dont les parois intérieure et extérieure étaient recouvertes d'une feuille d'étain (**fig. 2**). Reliée à un générateur électrostatique qui fournissait la charge, cette *bouteille de Leyde* permettait de réaliser des expériences spectaculaires (et pas toujours sans danger).

Ceux d'aujourd'hui sont habituellement beaucoup plus petits. La **figure 3** montre les modèles plus courants. À l'extrême droite, un 'gros' condensateur métal/papier ; ces condensateurs sont souvent utilisés pour supprimer les parasites des moteurs. Un de leurs avantages est d'être très robustes ; un de leurs inconvénients est leur taille pas vraiment petite.

Les condensateurs électrolytiques sont considérablement plus petits (fig. 3, colonne centrale). Dans les condensateurs électrolytiques, le diélectrique est un *électrolyte ;* ils souffrent d'un inconvénient important : ils sont polarisés, c'est-à-dire qu'ils ont une borne positive et une borne négative. Ils doivent obligatoirement être montés dans le bon sens (en particulier ceux dits au tantale) sous peine de conséquences spectaculaires et extrêmement malodorantes. Il existe des condensateurs électrolytiques non polarisés, utilisables indifféremment dans les deux sens.

La colonne la plus à gauche de la figure 3 montre quelques condensateurs en polypropylène en haut et quelques condensateurs en céramique en bas. Ces derniers sont agréablement petits et peu coûteux, mais ne sont disponibles que pour des valeurs de capacité relativement faibles.

Il y a tant à dire sur les différents types de condensateurs disponibles et leurs caractéristiques spécifiques que nous pourrions facilement en remplir tout un numéro d'Elektor. Mais pour vos débuts en électronique, les types mentionnés ci-dessus devraient suffire. Au cours de votre pratique, vous l'aurez l'occasion de découvrir d'autres types – pas d'inquiétude !

## Le condensateur comme réservoir d'énergie

Un condensateur peut stocker une charge électrique, et d'autant plus que sa capacité est plus élevée et que la tension appliquée à ses bornes est plus élevée :

$$Q = C \cdot U$$

De là découle la définition officielle de l'unité de capacité, le farad : la capacité d'un condensateur est la charge qu'il contient quand il est soumis à une tension d'un volt. L'énergie stockée dans un condensateur est calculée comme suit :

$$W = \frac{1}{2} \cdot C \cdot U^2$$

Les condensateurs sont fréquemment utilisés comme réservoirs d'énergie dans les alimentations électriques : la tension alternative (abaissée) est redressée, ce qui donne une tension continue pulsée qui est appliquée à un condensateur (d'assez grosse capacité) qui fonctionne comme un réservoir ou un tampon. On obtient une tension filtrée et ce condensateur est appelé un condensateur filtrage. Quand nous aborderons le sujet des diodes et des circuits de redressement, nous y reviendrons plus en détail.

## Caractéristiques des charges et des décharges

Lorsqu'un condensateur est chargé ou déchargé, la tension à ses bornes dépend de la charge qu'il contient :

$$U = \frac{Q}{C}$$

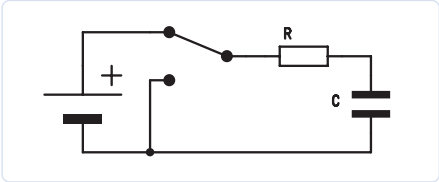



*Figure 4. Charge et décharge périodiques d'un condensateur à travers une résistance.*

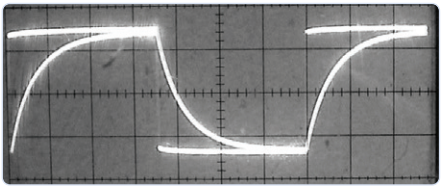


*Figure 5. Cycles de charge et de décharge observés sur un oscilloscope.*

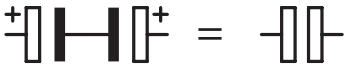


*Figure 6. Deux condensateurs électrolytiques montés tête-bêche forment un condensateur non polarisé.*

Si nous chargeons un condensateur avec un courant constant, la tension à ses bornes croît linéairement – et décroît linéairement s'il est déchargé avec un courant constant. Dans la pratique, c'est rarement le cas ; il est beaucoup plus fréquent d'utiliser une tension constante (**fig. 4**).
Comme la tension aux bornes du condensateur croît continuellement pendant la charge, la tension aux bornes de la résistance décroît continuellement (la somme des deux étant toujours égale à la tension de charge constante), le courant de charge décroît donc continuellement. La charge du condensateur devient donc de plus en plus lente ; la tension à ses bornes se rapproche asymptotiquement de la tension de charge (**fig. 5**).

Lorsque l'on décharge un condensateur à travers une résistance, il se produit exactement l'inverse : en raison de la tension élevée aux bornes du condensateur, il y a un courant de décharge initiale intense, mais comme cette tension diminue continuellement, il en est de même du courant de décharge.
Lorsqu'on charge un condensateur à travers une résistance, on a la même relation entre la tension et le courant (nous ne vous ennuierons pas avec des histoires de dérivées et vous pouvez tout aussi bien oublier cette relation, elle n'est présentée ici que par souci d'exhaustivité) :

$$U_C = U_0 \cdot \left(1 - e^{-t/RC}\right)$$

$$I_C = \frac{U_0}{R} \cdot \left(e^{-t/RC}\right)$$

Les formules suivantes sont valables pour la décharge :

$$U_C = U_0 \cdot \left(e^{-t/RC}\right)$$

$$I_C = \frac{-U_0}{R} \cdot \left(e^{-t/RC}\right)$$

## Connexions en série et en parallèle des condensateurs

Pour conclure cet épisode, nous allons examiner brièvement les connexions en série et en parallèle des condensateurs. La connexion en parallèle est en fait très simple : lorsque nous connectons des condensateurs en parallèle, nous pouvons imaginer que les « plaques » du condensateur résultant sont devenues plus grandes et donc que la capacité qui en résulte est également plus grande. Cela est vrai :

$$C_{total} = C_1 + C_2 + \ldots + C_n$$

Remarquable ! Pour les condensateurs connectés en parallèle, la relation est la même que pour la mise en série des résistances ! Y aurait-il quelque chose de similaire pour la connexion en série des condensateurs ? En effet : comme pour la mise en parallèle des résistances, la règle de l'inverse de la somme des inverses s'applique :

$$C_{total} = \frac{1}{\frac{1}{C_1} + \frac{1}{C_2} + \ldots + \frac{1}{C_n}}$$

Enfin, une remarque sur les condensateurs électrolytiques : en les connectant tête-bêche, on obtient un électrolytique non polarisé (**fig. 6**).
Nous en restons là pour cet épisode. Dans le prochain, nous examinerons de près les condensateurs fonctionnant comme des résistances en alternatif. Il y a beaucoup à dire à ce sujet, car avec les condensateurs, il y a un déphasage entre la tension et le courant – ce n'est pas le cas avec les résistances..

(200680-04)

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

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

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


**PRODUITS**

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

**LIEN**

1] **E. Bogers, « démarrer en électronique… (6) », Elektor, 01-02/2021 :**
www.elektormagazine.fr/200551-02

elektorstore

elektor.fr

# e-choppe Elektor

## des produits et des prix surprenants

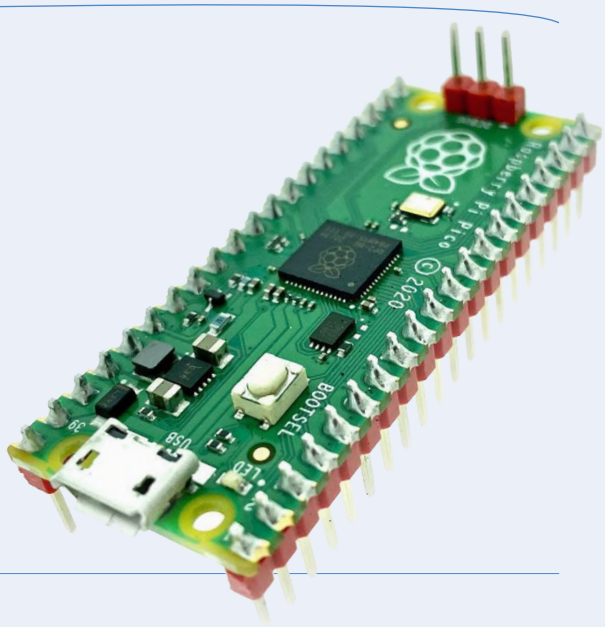
L'e-choppe Elektor a évolué. D'abord simple commerce de proximité pour nos productions (livres, magazines, kits et modules), c'est désormais une boutique en ligne bien rodée qui propose des produits surprenants à des prix très étudiés. Ce sont les produits que nous aimons et testons nous-mêmes. Si vous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

## Raspberry Pi Pico

(avec connecteurs soudés)

**Prix : 9,95 €**

**www.elektor.fr/19568**

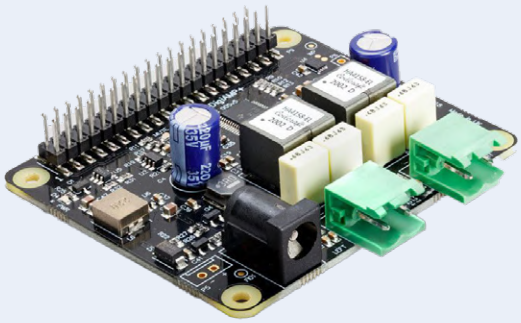


## JOY-iT JT-RD6006 alim de labo (360 W)


Prix : 89,95 €

**Prix (membres) : 80,96 €**

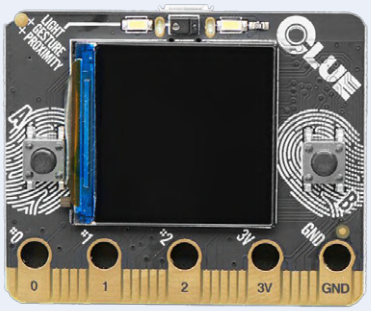
**www.elektor.fr/19564**



## IQaudIO DigiAMP+ DAC & amplificateur de classe D pour Raspberry Pi

Prix : 29,95 €
**Prix (membres) : 26,96 €**

**www.elektor.fr/19538**




## Adafruit CLUE – nRF52840 Express avec Bluetooth LE

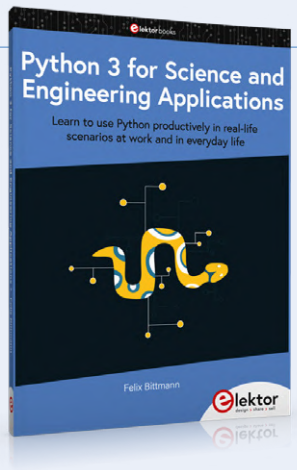
Prix : 49,95 €
**Prix (membres) : 44,96 €**

**www.elektor.fr/19512**

## IoTize TapNLink Primer NFC-BLE-WiFi Evaluation Kit

Prix : 59,95 €
**Prix (membres) : 53,96 €**

**www.elektor.fr/19494**




## Python 3 for Science and Engineering Applications

Prix : 29,95 €
**Prix (membres) : 26,96 €**

**www.elektor.fr/19441**

# 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 **14 juin 2021** à l'adresse **hexadoku@elektor.fr**

## Les gagnants

La solution de la grille du numéro de mars/avril 2021 est **29BF4.**

La liste des gagnants est publiée ici : www.elektormagazine.fr/hexadoku

Bravo à tous les participants et félicitations aux gagnants !

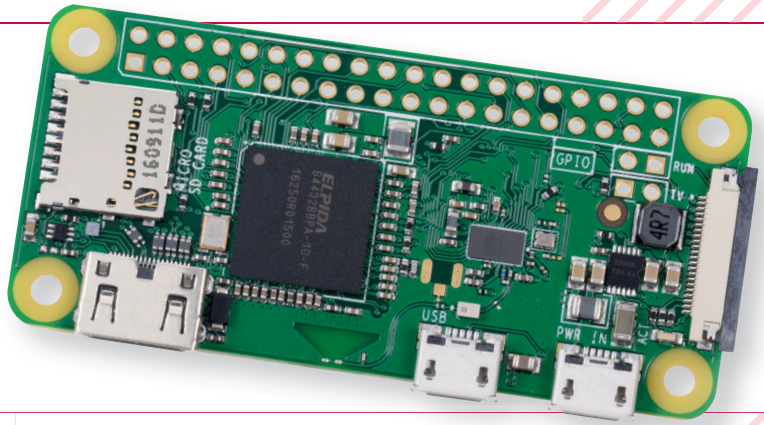
| | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | 7 | | 1 | F | 6 | | 2 | | | 0 | | 8 | | | 5 |
| A | 4 | 0 | | | | | | 5 | C | 2 | D | 7 | | | |
| | 9 | | | 0 | | | 5 | | | 1 | 7 | | A | | |
| D | | | 5 | | | B | | | 4 | 8 | 9 | | | 0 | 1 |
| 3 |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

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

*Tout recours est exclu, de même que le sont, de ce jeu, les personnels d'Elektor International Media et leur famille. Un seul gagnant par foyer.*

# 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**

elektor MAG
sixty > years > young

# CONTEST

Postez une photo de votre **première** couverture Elektor ou de votre couverture **préférée** sur les médias sociaux.

1. **Taguez @ElektorLabs**
2. **Mentionnez #Elektor60**

Participez pour avoir une chance de gagner l'un de nos superbes prix !

Plus d'informations :
www.elektor.fr/contest-Elektor60


1961 1971 1981 1991 2001 2011 2021