F : 15,50 €

elektor 485
créer > partager > vendre

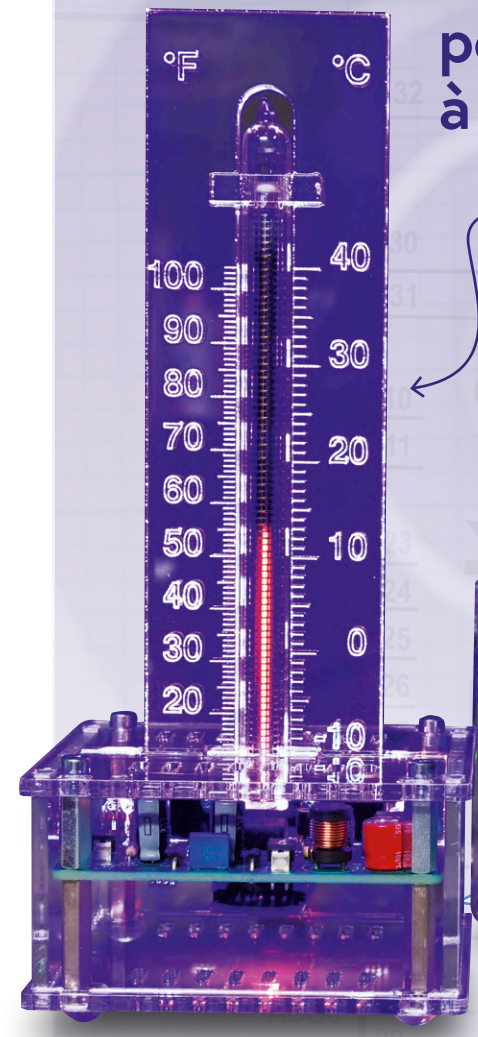
SEPTEMBRE/OCTOBRE
2020
ELEKTORMAGAZINE.FR

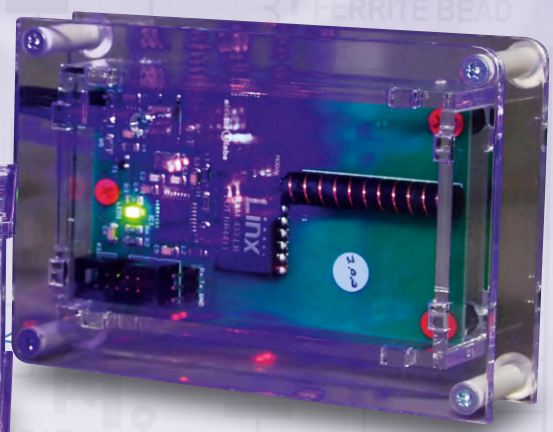
Elektor Helps
L'électronique aide quand ça va mal

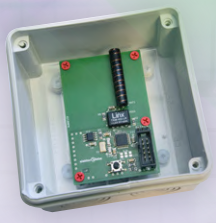
capteur sans fil

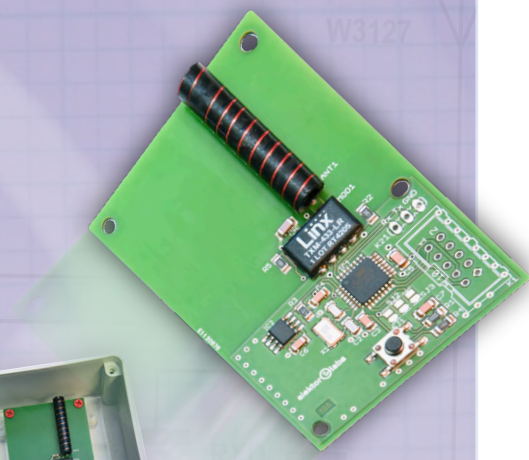
pour le thermomètre à bargraphe Nixie

p. 16







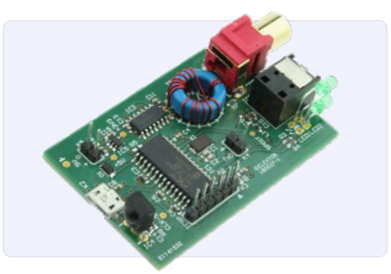


dans ce numéro :

- intelligence artificielle pour débutants
- multitâche en pratique avec l'ESP32
- minuterie pour ampli de casque
- conception de filtres analogiques
- alimentation HT avec traceur de courbes
- bidouiller une lampe IKEA
- dessine-moi un schéma avec EasyEDA52
- banc d'essai: multimètre OWON OW18E avec Bluetooth

et beaucoup d'autres !

la domotique, c'est facile avec...
ESPHome, Home Assistant & MySensors
p. 36



interface audio USB-S/PDIF
sortie audio numérique pour ordinateur, ordinateur portable, tablette etc.
p. 6

ISSN 0181-7450

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

L 19624 - 485 - F: 15,50 € - RD

NOTRE GAMME
PAR DES TECHNICIENS
POUR DES TECHNICIENS

The best part of your project: www.reichelt.com/assortiment

Uniquement le meilleur pour vous - provenant de plus de 900 marques.

Nos responsables produits sont employés par Reichelt depuis de nombreuses années et connaissent les exigences de nos clients. Ils rassemblent une large gamme de produits de qualité, à la fois parfaits pour les besoins dans les domaines de la recherche et du développement, la maintenance, l´infrastructure informatique et la production en petites séries et adaptés pour les fabricants.

Multimètre graphique avec Bluetooth®

pour le transfert de données vers un smartphone ou une tablette via une application Android ou iOS

PeakTech®

L'affichage couleur TFT éclairé affiche les différents menus de l'interface utilisateur graphique, les points de mesure détectés par l'enregistreur de données et les valeurs mesurées de la fonction multimètre en toute fiabilité et avec une grande précision.

- Mesure de la tension jusqu'à 1 000 VCA/CC
- Mesure du courant jusqu'à 10 ACA/CC
- Fonction d'enregistreur de données avec représentation de courbes
- 49.999 points
- Contenu de la livraison :

  sacoche, câbles de test, capteur de température type K, batterie li-ion 7,4 V, chargeur, clé Bluetooth et manuel

PETIT PRIX

Bestell-Nr.: PEAKTECH 3440

175,95 (151,68)

BACK LIGHT · APP · TRMS · EN 61010-1 CAT III 1000 V · EN 61010-1 CAT IV 600 V

Available on the App Store

ANDROID APP ON Google play

Vous trouverez de nombreux autres multimètres pour toutes les applications en ligne.

CHAUVIN ARNOUX · FLUKE · UNI-T · BENNING · METREL Mess- und Prüftechnik · BEHA-AMPROBE

Découvrez tout de suite ▶ www.reichelt.com/multimetres

Types de paiement : Sofort. · VISA · MasterCard · Maestro · PayPal · amazon pay

PRIX DU JOUR! Prix à la date du: 10. 8. 2020

- Excellent rapport qualité prix
- Plus de 110 000 produits sélectionnés
- Livraison fiable - depuis l'Allemagne dans le monde entier

Assistance téléphonique: +33 97 518 03 04 www.reichelt.com

reichelt elektronik – Tirer le meilleur parti de votre projet

43ème année
n° 485 – septembre/octobre 2020

ISSN 0181-7450
Dépôt légal : septembre 2020
CPPAP 1120 T 83713
Directeur de la publication : Donatus Akkermans

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

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

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

Jens Nickel

rédacteur en chef d'Elektor Magazine

Tout est une question de dosage

Son équilibre entre théorie et pratique vaut au magazine Elektor d'être apprécié par ses lecteurs du monde entier. Chaque numéro offre en plus un choix varié, depuis l'article de fond pour électronicien aguerri jusqu'au contenu accessible au débutant. Au fil de mes conversations avec nos lecteurs, j'ai perçu que bien des débutants lisent aussi les articles destinés aux experts. Lesquels ne dédaignent pas, à l'occasion, d'élargir ou de rafraîchir leurs connaissances dans des domaines moins familiers en lisant les articles pour débutants.

Dans ce numéro, vous en trouverez plusieurs, de ces articles pour « débutants », avec des petits circuits et même des instructions pas-à-pas pour dessiner vos premiers schémas sur votre ordinateur, grâce au programme de CAO simple EasyEDA.

Dans l'édition d'été d'Elektor, vous aviez sans doute déjà remarqué et apprécié la nouvelle mise en page. Celle-ci est désormais en place, et les hors-série *Elektor Industry*, les livres d'Elektor et nos sites ont eux aussi bénéficié d'un rafraîchissement. Que pensez-vous de cette nouvelle maquette ? Son créateur, notre graphiste Harmen Heida et moi-même serions heureux de recevoir vos commentaires par courriel (redaction@elektor.fr) !

Bonne nouvelle enfin pour tous les pros, surtout ceux qui s'intéressent, que ce soit pour leur profession ou à titre privé, à l'évolution récente et future de l'électronique. Tout a été prévu – sauf l'imprévisible, évidemment – pour que le salon **electronica** puisse ouvrir ses portes le 10 novembre.

Dans ce cadre, la compétition *Fast Forward Award*, où s'affrontent les meilleures jeunes sociétés de l'année, pourra également avoir lieu comme nous l'espérons. Nous nous réjouissons de vous y accueillir nombreux !
www.elektormagazine.fr/fastforward

Bonne lecture !

Jens Nickel

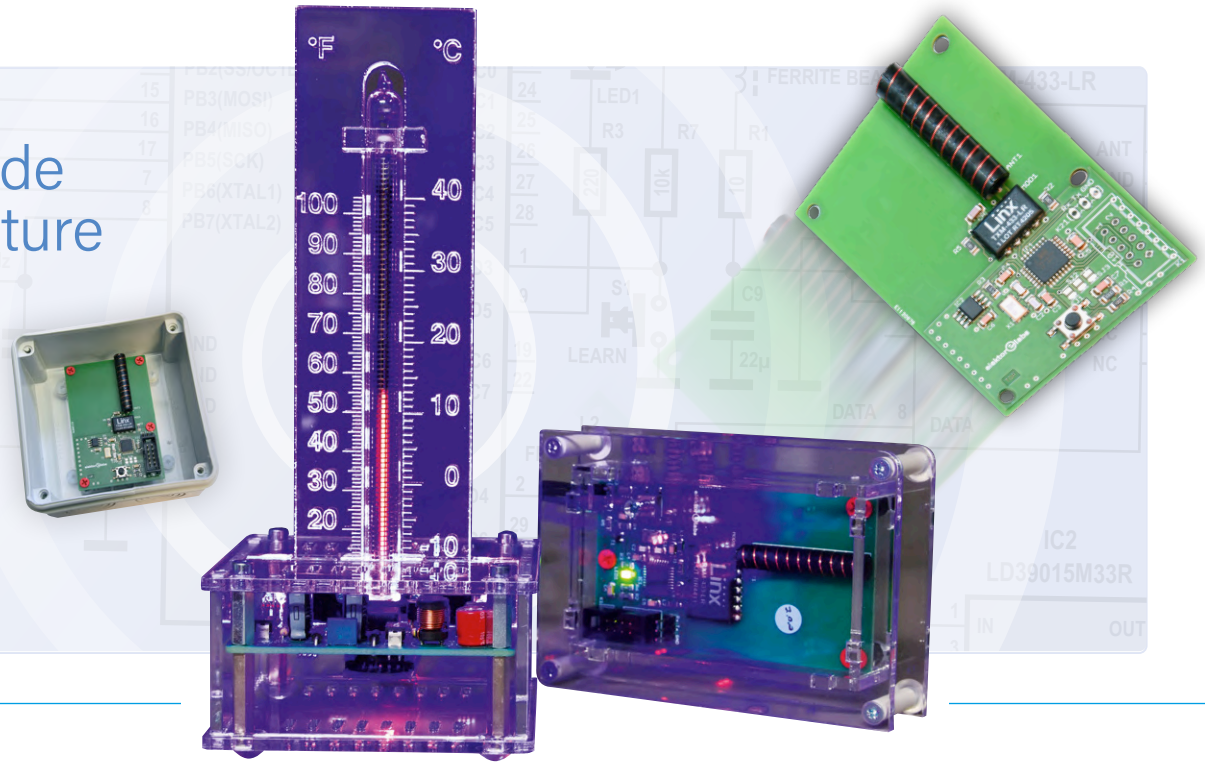
notre équipe

DANS CE NUMÉRO

43e année – n° 485
septembre - octobre 2020

projet du labo

capteur de température sans fil



16

rubriques

substantifique moëlle

projets

bidouiller une lampe IKEA

avec LED NeoPixel et réseau WiFi

56

banc d'essai

charge électronique

89

multitâche avec Raspberry Pi

23

bientôt dans ces pages

- LCR-mètre de précision
- traceur LoRa
- sonde de courant
- analyseur de puissance
- montre-bracelet à nixie
- domotique avec Home Assistant (2)
- FFT avec Maixduino
- Teensy 4.0 - la vitesse
- gestion de batteries : les bases

...et bien davantage !

Sous réserve de modification.
Le numéro d'Elektor de novembre-décembre 2020 paraîtra le 5 novembre.

190365-E-03

Découvrez le nouveau site Elektor Labs

interface audio **USB-S/PDIF**

Sortie audio numérique pour ordinateur, ordinateur portable, tablette etc.

Stephan Lück (Allemagne)

La plupart des PC, ordinateurs portables, tablettes et téléphones tactiles n'ont pas à proprement parler de sortie audio ; c'est au mieux un connecteur USB polyvalent et une sortie casque. Comment connecter un tel appareil à un ampli de qualité ou à un récepteur AV ? Voici une réponse sérieuse à cette question : une interface USB-S/PDIF de qualité.

INFOS SUR LE PROJET

Mots-clés
audio numérique, PC, ordinateur portable, tablette, téléphone tactile

Niveau
débutant – connaisseur – expert

Durée
environ 4 h

Outils
outils de soudure pour CMS et traversants, outils mécaniques

Coût
35 € à 40 €

CARACTÉRISTIQUES

- tension : +5 V
- courant : 36 à 43 mA
- µC : PIC32MX27F256B-I
- 3 fréquences d'échantillonnage : 44,1, 48 et 96 kHz
- format audio : 16, 20 et 24 bits
- sorties S/PDIF optiques et électriques
- télécommande IR (et RC5)
- compatible avec Windows 7/10, Linux, Android et Raspbian

S/PDIF ou S/P-DIF pour *Sony/Philips Digital Interface Format* est une interface de données audio numériques de courte distance entre modules voisins, dans un salon de cinéma ou de hi-fi. Cette interface unidirectionnelle sérielle est dépourvue d'horloge séparée ; l'horloge est extraite du signal lui-même. L'idée est de conserver les données audio dans le domaine numérique aussi longtemps que possible et de ne les convertir en analogique qu'au dernier moment.

S/PDIF [1] est basé sur la norme professionnelle AES3 ; le protocole des deux normes est compatible, mais leurs caractéristiques électriques divergent. Une interface AES3 professionnelle typique utilise un connecteur XLR à 3 voies et un câble blindé à paires torsadées symétriques d'une impédance de 110 Ω. Le niveau du signal varie de 3 à 10 $V_{càc}$ (AES/EBU 2 à 7 $V_{càc}$). Moins courants sont les connecteurs BNC avec un câble coaxial de 75 Ω. Les AES3 symétriques avec connecteurs XLR (paire torsadée blindée) peuvent être utilisés jusqu'à 1000 m de distance, 100 m pour la version coaxiale.

Deux types de connexion sont définis pour le S/PDIF. La plus simple (généralement utilisée dans les composants audio réputés «meilleurs») est un câble coaxial 75 Ω avec fiches RCA (généralement de couleur orange). Le niveau du signal est d'environ 0,5 $V_{càc}$.

La deuxième variante, sous le nom de Toslink (de *Toshiba Link*), est une fibre optique standardisée à LED émettrice rouge (659 nm). Normalement, c'est un simple câble POF (*Plastic Optical Fibre*). Elle peut être utilisée jusqu'à environ 10 m. Avec une fibre optique en verre de qualité, une distance maximale de 30 m est possible sans répéteur. Les signaux optiques ont la même logique que les signaux électriques, puisque le signal électrique ne fait qu'allumer et éteindre la LED.

Projet ambitieux

Le circuit décrit ici (**fig. 1**) est un convertisseur USB vers S/PDIF qui se connecte facilement à un PC et d'autres appareils par connecteur USB ; par sa sortie S/PDIF il se connecte à des récepteurs AV, des amplificateurs haut de gamme ou des DAC audio autonomes. La

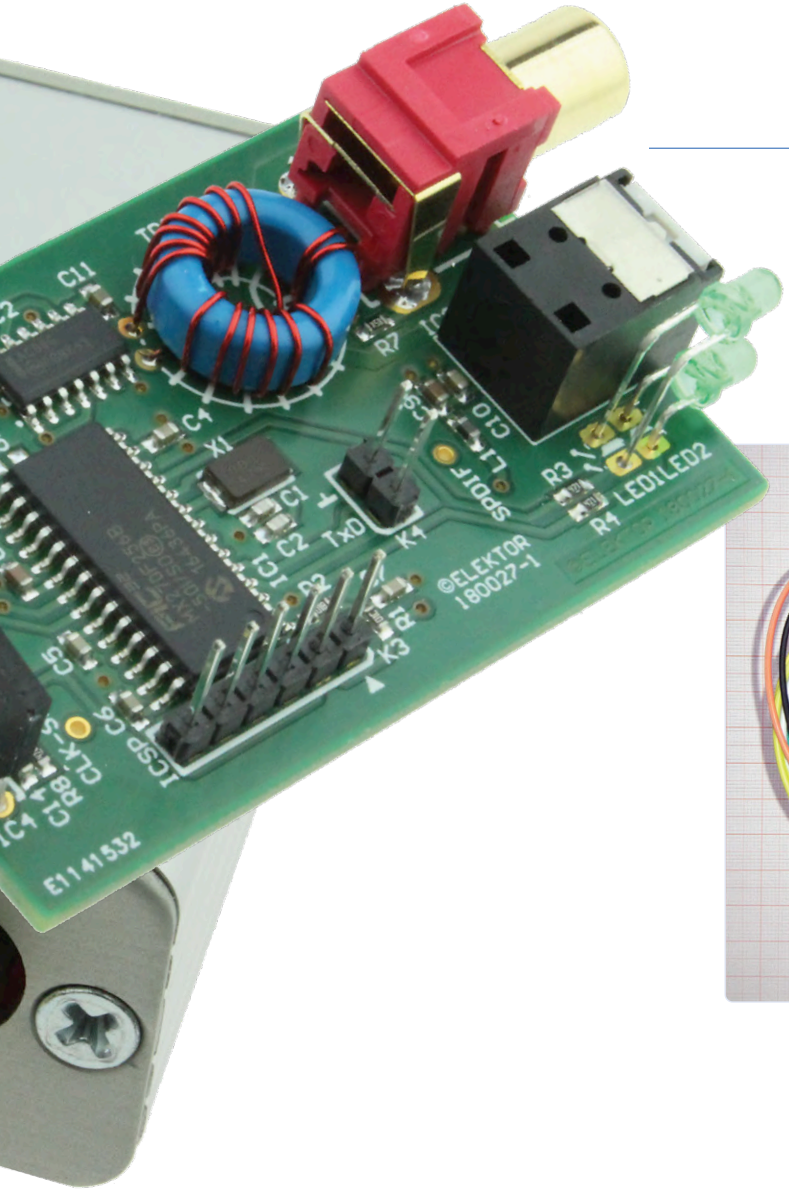


Figure 1. Interface USB-S/PDIF, élégante et compacte.

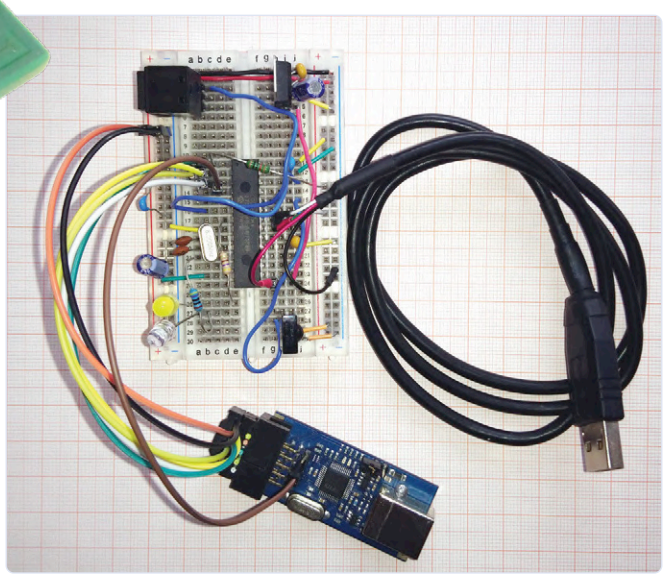


Figure 2. Prototype sur plaque d'expérimentation.

sortie S/PDIF existe sous forme électrique et optique ; la télécommande est possible grâce à un récepteur IR.

Le flux binaire S/PDIF est produit par le logiciel du µC chargé également de la communication USB. Cette solution à puce unique limite le matériel requis. Par rapport aux circuits intégrés audio USB spéciaux, cette solution est flexible et ouverte. Sa réalisation n'est pas trop critique, il peut être assemblé sur une plaque d'expérimentation (**fig. 2**).

Pour ce projet, nous avons choisi un µC PIC32MX270 doté des périphériques pour les applications audio USB, et de suffisamment de RAM pour stocker les trames S/PDIF codées. Il est disponible dans des variantes avec relativement peu de broches, ce qui simplifie le tracé du circuit imprimé.

La petite carte double face conçue pour ce projet contient le µC, l'alimentation et les sorties S/PDIF optiques et électriques, ainsi que le récepteur de télécommande IR, plus deux LED (qui indiquent la fréquence d'échantillonnage et l'activité de sortie). Tout cela tient dans un petit boîtier Hammond.

Section audio

La section audio (**fig. 3**) se compose d'une interface USB conforme à la classe audio USB [2][3], de l'implémentation logicielle du codeur S/PDIF et de la sortie S/PDIF, qui utilise la sortie SPI du µC. Un canal DMA copie en continu les trames S/PDIF du tampon circulaire vers le SPI.

Comme cette interface USB-S/PDIF utilise la spécification audio USB standard, l'installation de pilotes spéciaux sur l'hôte est inutile. La spécification de la *classe de l'appareil* est assez complexe. C'est pourquoi l'USB-IF (*USB Implementers Forum*) a publié une *USB Audio Device Class Specification for Basic Audio Devices* [4], qui contient un petit sous-ensemble de la spécification de *classe de dispositif audio* originale. L'interface audio USB de notre projet ressemble beaucoup à l'application pour casque d'écoute [4]. Nous avons ajouté quelques fréquences d'échantillonnage, omis la commande de volume, et le descripteur USB a été modifié de manière à décrire une sortie S/PDIF au lieu d'un haut-parleur. Notre interface audio USB présente les caractéristiques suivantes :

- trois fréquences d'échantillonnage (44,1 kHz, 48 kHz et 96 kHz)
- format audio à deux canaux avec 3 octets (24 bits) par canal (S24_3LE)
- possibilité de couper le signal audio

Ces caractéristiques sont spécifiées dans le descripteur de configuration USB (fichier `usb_descriptors.c` dans les archives du logiciel — informations complémentaires [3], [5] et [6].)

Comme nous ne visons pas une conformité formelle à l'USB, nous avons retenu des identifiants de fournisseur et de produit utilisables sans risque de conflit avec les produits USB officiels.

Codeur S/PDIF et sortie sérielle

Pour transférer des échantillons audio vers S/PDIF, l'interface USB utilise le point terminal USB isochrone 1. Cela signifie que chaque trame USB de 1 ms transfère un paquet de données USB. Chaque fois que le matériel USB du µC a reçu un nouveau paquet isochrone, la routine d'interruption est appelée et convertit les trames PCM individuelles du

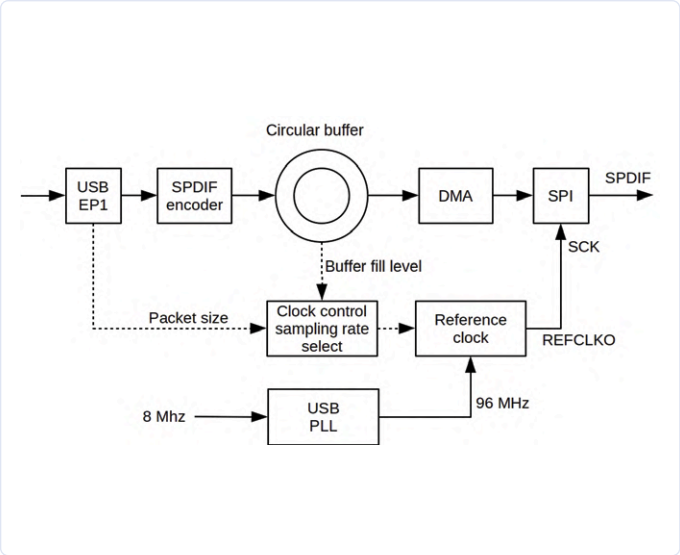


Figure 3. Schéma fonctionnel de l'interface.

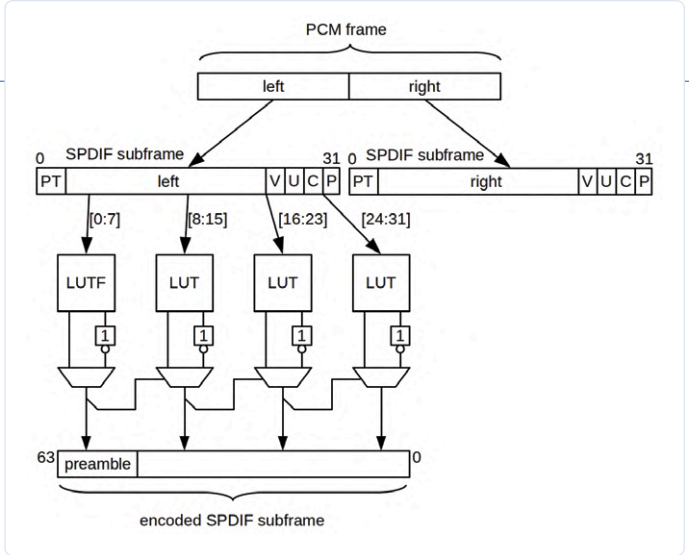


Figure 4. Principe du codeur S/PDIF logiciel.

paquet de données USB en trames S/PDIF correspondantes. Pour plus d'informations sur S/PDIF, consultez [7] et [8].

La **fig. 4** montre schématiquement comment fonctionne le logiciel de conversion S/PDIF. En haut, nous voyons la trame PCM de 48 bits ; chaque trame contient deux échantillons, l'un pour le canal gauche et l'autre pour le canal droit. Ces échantillons sont convertis en une représentation intermédiaire de 32 bits des sous-cadres S/PDIF en ajoutant une balise de préambule de 4 bits, qui spécifie le type de préambule S/PDIF (X, Y ou Z) qui doit être utilisé pour le codage final du sous-cadre S/PDIF. De plus, les quatre bits V, U C et P sont ajoutés à la fin. Les bits V et U correspondent respectivement aux bits de validité et de données utilisateur. Ici ces bits sont toujours à 0. Les bits C contiennent l'état du canal et les bits P sont les bits de parité pour les sous-cadres individuels. Ces bits de parité sont calculés pour chaque sous-trame à l'aide d'un algorithme optimisé [9].

Deux sous-trames S/PDIF successives forment une seule trame S/PDIF. 192 trames S/PDIF successives forment ensemble un bloc S/PDIF. L'état du canal est obtenu en mettant en chaîne tous les bits C d'un bloc l'un après l'autre - le bloc d'état du canal a donc un format de 192 bits (24 octets). Nous utilisons ici les bits d'état de canal pour indiquer au récepteur connecté à la sortie S/PDIF la fréquence d'échantillonnage actuelle.

Le S/PDIF utilise un codage de marque biphasé (également appelé codage *Manchester* différentiel) pour coder les données audio et les bits de préambule. Comme deux bits de code correspondent à un bit de données, la sous-trame S/PDIF codée comprend 64 bits. En utilisant la mise en œuvre la plus simple du codage de marque biphasée, chaque bit d'une sous-trame S/PDIF devrait être codé individuellement, ce qui, bien sûr, consommerait un temps de processeur précieux. C'est pourquoi nous utilisons deux tables de recherche différentes (LUT = *look-up table*) pour coder un octet à la fois. La table de recherche LUTF est utilisée pour coder le premier octet d'une sous-trame S/PDIF, tandis que la table de recherche LUT est utilisée pour coder les trois autres octets de la sous-trame. La table spéciale LUTF contient les modèles de bits du préambule.

La table de recherche LUT apparaît trois fois dans la **fig. 4**, car elle est utilisée trois fois pour coder une sous-trame S/PDIF. En réalité, il n'existe en mémoire qu'une seule instance de cette table. Les tables de consultation sont produites par une fonction d'initialisation exécutée une fois après réinitialisation du µC. Dans le code de marquage biphasé, il doit y avoir une transition (flanc) entre toute paire de bits de code qui représente un bit de données. Pour assurer cette transition, les mots de code de 16 bits des tables de recherche sont inversés ou non en fonction du dernier bit du mot de code précédent.

Les tables de recherche remplissent une deuxième fonction : l'inversion de l'ordre des bits. Dans la norme S/PDIF, les sous-trames sont transmises en commençant par le bit de poids le plus faible (LSB), mais le SPI le fait dans l'ordre inverse (MSB d'abord). C'est pourquoi les LUT fournissent les bits dans l'ordre inverse, afin qu'ils soient correctement transmis par le matériel SPI.

Les mots de code S/PDIF de 64 bits ainsi obtenus sont placés dans une mémoire tampon circulaire (dans la mémoire SRAM du µC) et envoyés par le port SPI en utilisant le DMA.

Horloge de bits S/PDIF

Un diviseur de fréquence fractionnaire programmable intégré au µC dérive l'horloge binaire S/PDIF d'un signal d'horloge interne de 96 MHz. La fréquence d'horloge S/PDIF requise s'élève à

$$f_{SPDIF} = 128 * f_S$$

où f_S est la fréquence d'échantillonnage audio. La spécification audio USB ne permet pas d'indiquer explicitement la fréquence d'échantillonnage f_S utilisée par l'hôte du périphérique USB. Cela signifie que le logiciel du µC la détecte d'après la base du nombre de trames PCM dans les paquets USB isochrones. Notez que le descripteur USB contient trois fréquences d'échantillonnage autorisées. L'hôte n'est pas autorisé à utiliser d'autres fréquences d'échantillonnage que ces 44,1 kHz, 48 kHz ou 96 kHz. L'avantage est que le nombre de trames MIC dans un paquet USB permet au µC de deviner la fréquence d'échantillonnage choisie par l'hôte.

Périls de l'horloge

Conformément à la spécification *USB Basic Audio Devices* [4], c'est le type de synchronisation «synchrone» qui a été sélectionné pour le point terminal audio isochrone. Autrement dit, la vitesse d'échantillonnage du flux audio dépend du domaine de l'horloge USB, c'est-à-dire de la fréquence des jetons SOF exécutés par l'hôte. L'horloge de bits S/PDIF est en revanche dérivée de l'oscillateur local. Pour éviter tout dépassement ou la vacance du tampon circulaire, le logiciel modifie fréquemment la partie fractionnaire du diviseur de fréquence, en fonction du niveau moyen de remplissage du tampon circulaire.

Bien que la note d'application AN1422 [10] de

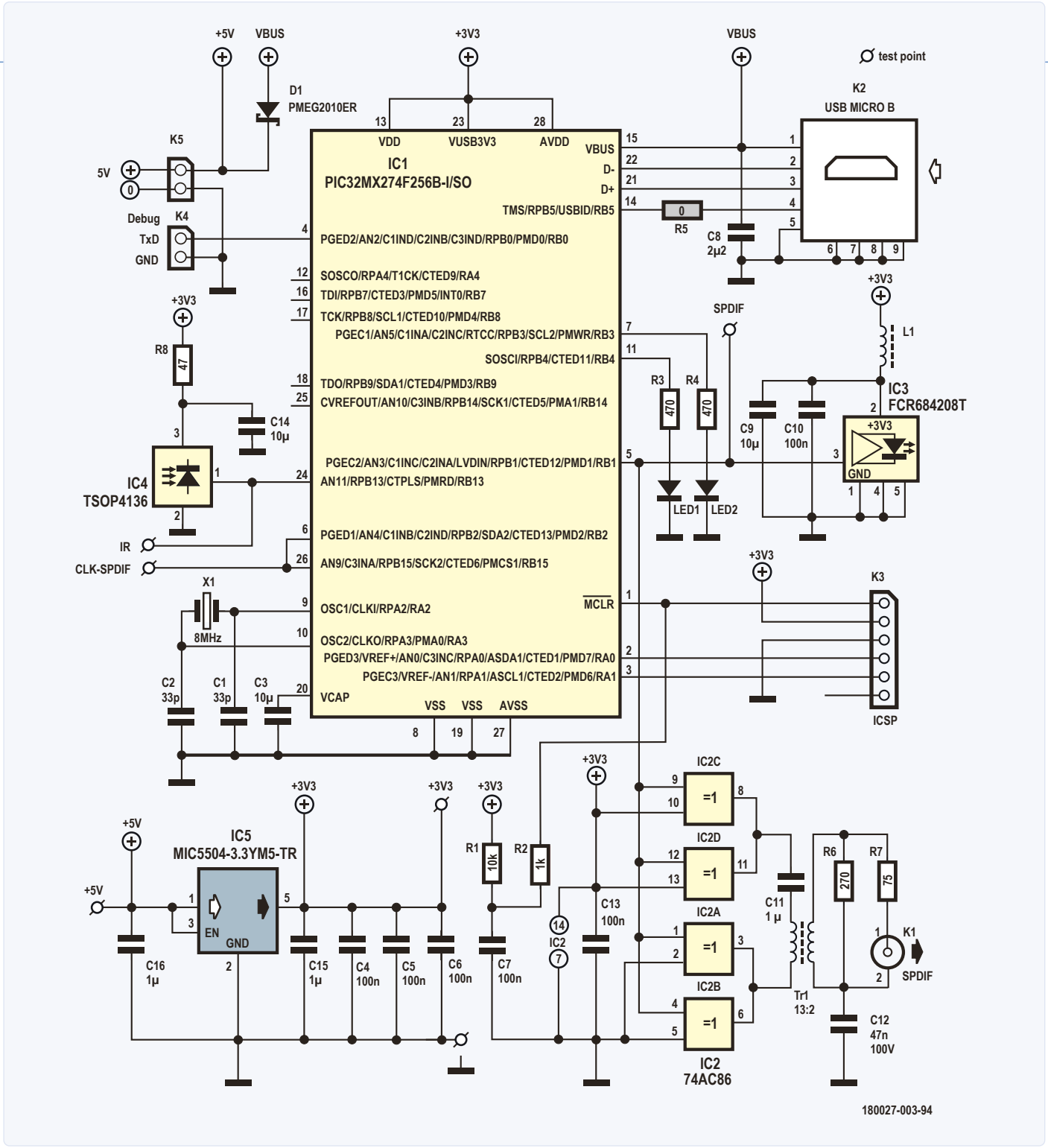


Figure 5. Schéma complet de l'interface USB-S/PDIF.

Microchip indique explicitement que l'horloge de référence peut être ajustée en cours de fonctionnement, nous avons constaté des problèmes lors du changement du rapport de division lorsque ce dernier était faible (c'est-à-dire avec la fréquence d'échantillonnage de 96 kHz). Chaque modification du rapport de division se traduisait par une courte interruption du signal d'horloge et des artefacts audibles.

Pour résoudre ce problème, nous relié la sortie de l'oscillateur d'horloge de référence REFCLKO (broche 6) à l'entrée d'horloge SPI SCK (broche 26), de façon à contourner le générateur de débit en bauds du SPI.

Télécommande IR

Le récepteur de la télécommande IR est indépendant de la section audio du circuit. Tout ce que le récepteur reçoit est transmis tel quel à l'hôte sans influencer la section audio. Pour décoder le signal IR de la télécommande, on utilise la bibliothèque libre IRMP [11] qui permet de décoder de nombreux signaux IR différents. Le logiciel du µC comprend une implémentation USB HID [12] qui permet d'envoyer des codes comme le spécifie le document *HID Usage Tables* [13].

La connexion entre l'IRMP et l'implémentation HID est obtenue à l'aide d'une table de

LISTE DES COMPOSANTS

Résistances (CMS 0603, 1%, 0,1 W)
R1 = 10 kΩ
R2 = 1 kΩ
R3,R4 = 470 Ω
R5 = 0 Ω **omise**
R6 = 270 Ω
R7 = 75 Ω
R8 = 47 Ω

Condensateurs (CMS 0603)
C1,C2 = 33 pF, 50 V, 5%, C0G/NP0
C3,C9,C14 = 10 µF, 16 V, 20%, X5R
C4...C7,C10,C13 = 100 nF, 50 V, 10%, X7R
C8 = 2,2 µF, 10V, 10%, X7R
C11,C15,C16 = 1 µF, 50 V, 10%, X5R
C12 = 47 nF, 100 V, 10%, X7R

Inductances
L1 = 600 Ω @ 100 MHz, 0,15 Ω, 1,3 A, CMS 0603 (Murata, BLM18KG601SN1D)
TR1 = noyau torique, 12,5 x 5 mm, matériau T38, Epcos B64290L0044X038

Semi-conducteurs
LED1,LED2 = LED verte, 3 mm, traversante
D1 = PMEG2010ER, CMS SOD-123
IC1 = PIC32MX274F256B-I/SO, CMS SO-28
IC2 = 74AC86, CMS SO-14
IC3 = FCR684208T, Toslink (Cliff Electronic Components)
IC4 = TSOP4136
IC5 = MIC5504-3.3YM5-TR, CMS SOT-23-5

Divers
K1 = prise RCA encartable, traversante (Pro Signal, PSG01545)
K2 = micro-USB type B, femelle, CMS (Molex, 47346-0001)
K3 = barrette1x6, verticale, pas 2,54 mm, traversante
K4,K5 = barrette 1x2, verticale, pas 2,54 mm, traversante
X1 = quartz 8 MHz, 5 x 3,2 mm, CMS (Abracon, ABM3-8.000MHZ-D2Y-T)
TR1 = 0,3 m de fil de cuivre verni (CuL), Ø 0,5 mm
Boîtier Hammond 1455D601, 60 x 42,5 x 23 mm

circuit imprimé 180027-1 v2.01 (e-choppe Elektor)

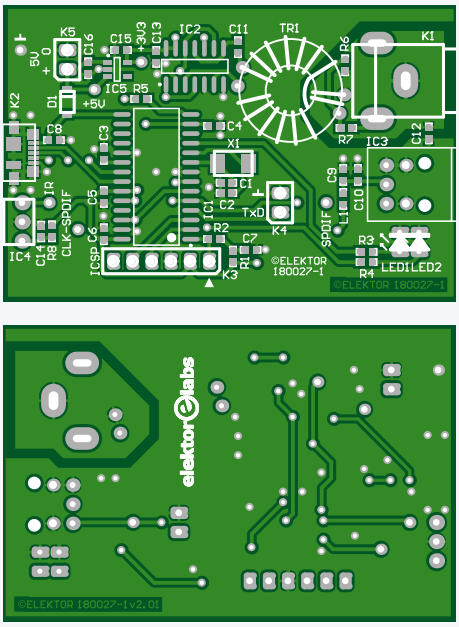


Figure 6. Dessin du circuit imprimé compact conçu pour l'interface.

correspondance de clés qui couple les codes détectés par l'IRMP aux ID d'utilisation [13]. Nous avons décidé d'ajouter les codes de la norme RC5, certes de moins en moins utilisée mais disponible sur de nombreuses télécommandes universelles. Elle fonctionnera bien dans de nombreuses situations, sans interférer avec d'autres appareils tels que les téléviseurs. Outre les codes RC5 énumérés [14], le tableau des codes contient également des codes pour télécommande Denon.
La carte des clés se trouve dans le fichier source `irhid.c` sous le nom `irhid_keymap`, et peut facilement être modifiée selon vos besoins en ajoutant ou en supprimant des lignes et en recompilant ensuite le logiciel. Pour ajouter une touche, vous devez trouver les codes IRMP pour le protocole concerné, l'adresse et la commande. Cela peut se faire en connectant un émulateur de terminal (115200 bits/s) à la sortie S/PDIF et en appuyant sur le bouton correspondant. Cela devrait suffire pour voir apparaître les codes correspondants dans l'émulateur de terminal. Si cela ne fonctionne pas, vous devrez peut-être activer le code de la télécommande appropriée en éditant le fichier `irmpconfig.h`. Vous avez également besoin de l'identifiant d'utilisation USB HID correspondant qui doit être couplé à la touche. Vous trouverez cet ID dans [13].

Assemblage

Après toutes les explications déjà données, le tour du schéma (**fig. 5**) sera vite fait. IC1, le microcontrôleur, peut être programmé en circuit via le connecteur ICSP K3 ; si vous préférez ne pas le faire vous-même, vous pouvez acheter le µC préprogrammé dans l'e-choppe d'Elektor. La source d'alimentation externe +5 V est connectée à K5.
Le signal USB entre par K2 et va directement au µC. La résistance de 0 Ω R5 ne doit pas être installée, car la version actuelle du logiciel n'utilise pas l'USBID.
IC4 est un récepteur IR standard ; IC3 s'occupe de la sortie optique (Toslink) ; le signal de sortie électrique arrive sur K1 via le quadruple tampon IC2. Nous arrivons ainsi au composant le moins banal de ce circuit : le transformateur TR1. Sa fonction n'est pas tant d'assurer une séparation galvanique que d'éviter les boucles de terre. Pour réduire le bruit RF, la sortie doit être découplée, ce que fait C12. Les opérateurs logiques EXOR d'IC2 qui forment un pont, présentent l'avantage d'augmenter la tension primaire et offrent un meilleur couplage à l'enroulement secondaire. Le rapport de transformation du transformateur est de 13:2 et cela donne une tension qui est pratiquement exactement de 0,5 $V_{càc}$ pour une charge de 75 Ω.
Un circuit imprimé (double face), **fig. 6**, a été dessiné pour ce circuit. La plupart des composants sont montés en surface, mais l'assemblage ne devrait pas poser de problème pour un électronicien expérimenté.
Le circuit imprimé tient dans un boîtier Hammond en alu de type 1455D601 (60 x 42,5 x 23 mm) où il peut simplement être glissé comme le montre la **fig. 7**. Selon les éventuelles tolérances de fabrication, il faudra peut-être passer une lime sur les bords de la carte.
Si vous utilisez le boîtier Hammond de la liste des composants, il est préférable de supprimer l'une des deux collerettes en plastique noir : à cause d'elle, le connecteur micro-USB n'affleure pas à la surface du boîtier, et le contact de la fiche mâle correspondante est mauvais. Les panneaux latéraux nécessitent bien sûr le perçage et la découpe d'ouvertures appropriées pour les connecteurs et les LED. Les broches des LED peuvent être pliées de telle manière qu'elles «regardent» vers l'extérieur, l'une au-dessus de l'autre (**fig. 7**).

Une remarque sur le transformateur : vous devrez le bobiner vous-même sur le noyau torique spécifié dans la liste des composants. La **fig. 8** montre que l'enroulement primaire (13 tours) est effectivement enroulé en deux moitiés sur le tore. Cela présente l'avantage que les connexions pour les enroulements primaire et secondaire sont opposées l'une à l'autre. Le secondaire n'a que deux tours, ce n'est pas vraiment pas sorcier, il suffit d'une trentaine de centimètres de fil de cuivre verni de 0,5 mm.

Quelques remarques pratiques

Nous avons bien sûr testé à fond l'interface USB-S/PDIF dans notre labo et essayée avec différents systèmes d'exploitation ; cela a bien fonctionné avec Windows 7 et Windows 10, Linux (Lubuntu et Kubuntu), Android et même Raspbian sur une Raspberry Pi 3 B+ en utilisant *2019-09-26-raspbian-buster-full.img*.

Windows

Windows a automatiquement reconnu l'interface ; la fréquence d'échantillonnage peut être réglée dans le panneau de configuration sous l'option Son. Sélectionnez «Propriétés de l'interface SPDIF» ; sur la page d'onglet Avancé, vous pouvez sélectionner le format, comme indiqué sur la **figure 9**.

Lubuntu

Sous Linux (nous avons utilisé Lubuntu), les fréquences d'échantillonnage dépendent du fichier. En utilisant le simple GNOME MPlayer, un fichier de 44,1 kHz apparaît comme 44,1 kHz sur la sortie S/PDIF. Mais à chaque fois, nous avons dû sélectionner «USB_SPDIF Stereo (IEC958) (PulseAudio)» comme sortie audio – même si nous l'avions déjà fait auparavant. C'est ennuyeux. Vous pouvez éviter cela en éteignant tous les autres appareils audio (ici : Sound & Video - PulseAudio Volume Control - Configuration). Un fichier de 48 kHz et un fichier de 96 kHz sont tous deux lus en 48 kHz. Une autre possibilité est d'utiliser ALSA directement, sans PulseAudio, dans un terminal. Si vous n'entendez rien, démarrez d'abord *alsamixer* dans un terminal (Ctrl+Alt+T et entrez `alsamixer`) ; appuyez sur F6 pour sélectionner la carte son 'USB_SPDIF' (il n'y a qu'un petit élément de contrôle qui affiche 00 et PCM, car l'interface n'a pas de réglage de volume). Ensuite, fermez le mélangeur et entrez (`//.../` est le chemin d'accès réel au fichier audio) :

```
aplay -D plughw:CARD=USBSPDIF
   //.../
```

Vous ne pouvez l'utiliser que pour lire des fichiers wav non compressés. Pour lire des fichiers mp3 à partir de la ligne de commande, vous pouvez utiliser p. ex. mpg123, mais il existe de nombreuses autres possibilités.
Il existe des CN/A audio USB depuis de nombreuses années, cependant la méthode standard de traitement des fichiers audio sous Linux reste perfectible. Le réglage de la fréquence d'échantillonnage de sortie est également possible en éditant les fichiers de configuration pour PulseAudio (`/etc/pulse/daemon.conf`). Des informations à ce sujet peuvent être trouvées en ligne. Faites d'abord une sauvegarde du fichier original, il est plus facile de faire une erreur que de ne pas en faire...

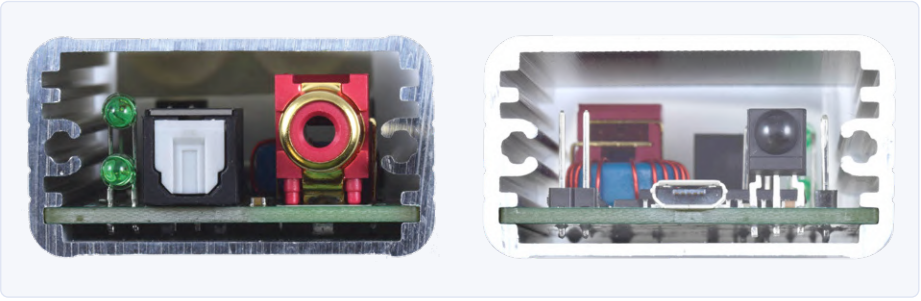


Figure 7. Circuit imprimé assemblé glissé dans le boîtier en alu.

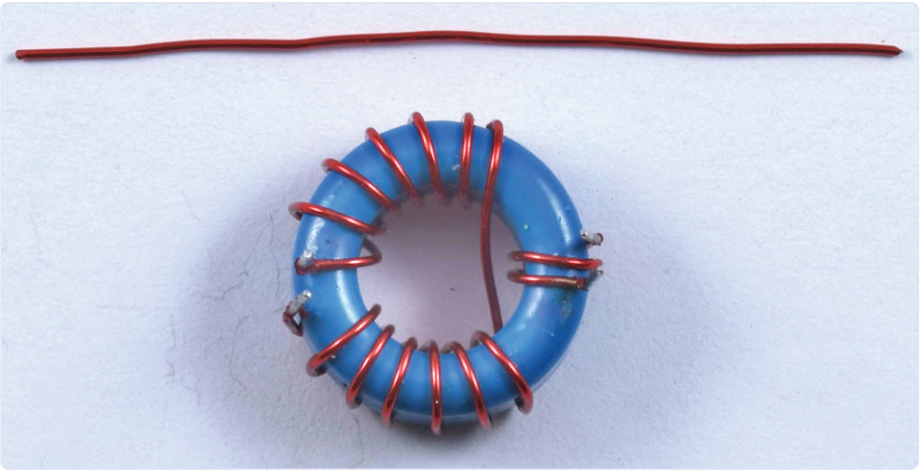


Figure 8. Bobinage du transfo torique. L'enroulement primaire est divisé en deux.

Android

Connectez l'interface à l'aide d'un câble OTG. L'installation de l'application «USB Audio Player Pro» est probablement la façon la plus simple d'utiliser notre interface. Cela permet de contourner les limitations audio d'Android. Les trois fréquences d'échantillonnage de l'interface sont prises en charge. Une fois que l'application a démarré, l'interface est immédiatement reconnue.

Raspbian

Notre dernier test portera sur l'interface avec Raspbian. Tout comme Lubuntu, Raspbian est un système d'exploitation libre également basé sur Debian ; il n'est donc pas surprenant que le test ait suivi le même schéma. La lecture des fichiers wav non compressés se fait ainsi :

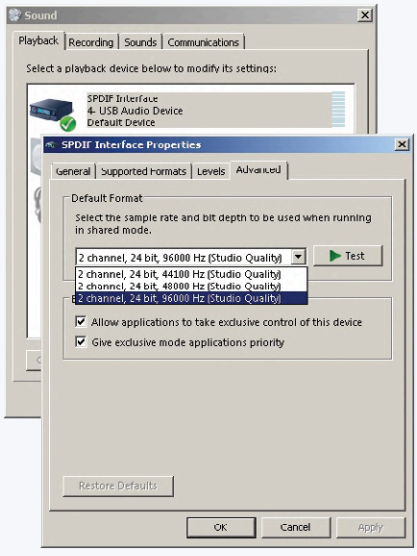


Figure 9. Paramètres de configuration pour Windows 7.

PRODUITS

- **Interface USB-S/PDIF, carte nue :** www.elektor.fr/180027-1
- **Interface USB-S/PDIF, microcontrôleur programmé :** www.elektor.fr/180027-41

```
aplay -D plughw:CARD=USBSPDIF //.../
```

Appuyez sur Ctrl+c pour arrêter la lecture.
Lors de la lecture de fichiers mp3 en ligne de commande, mpg123 ne fonctionnait (s'il n'est pas déjà installé : `sudo apt-get install mpg123`) avec notre nouvelle installation de Rasbian qu'après que PulseAudio et son contrôle de volume aient également été installés. Installez PulseAudio comme ceci :

```
sudo apt-get install pulseaudio
```

Installation du réglage de volume associé :

```
sudo apt-get install pavucontrol
   paprefs
```

Redémarrez le système après avoir tout installé.
Tout comme avec Lubuntu, vous devez désactiver l'audio embarqué dans la section *Sound & Video ➞ PulseAudio Volume Control ➞ Configuration*.
Le lecteur multimédia VLC préinstallé ne produisait aucun son avant l'installation de PulseAudio. Un fichier échantillonné à 32 kHz sera lu à 96 kHz.
Nous avons également mesuré les performances audio. Sur le site Elektor Labs [15], vous trouverez une description de ces mesures et de leurs résultats sur la page de ce projet.
Le logiciel mis à jour peut être téléchargé sur la page en ligne de cet article [16] ou sur GitHub [17].
Nous vous souhaitons beaucoup de plaisir lors de la réalisation de cette interface et bien sûr lors de son utilisation ! Vos observations et vos critiques sont les bienvenues.

180027-04

LIENS

[1] https://kompendium.infotip.de/spdif_toslink.html
[2] **spécification universelle des bus de série, révision 2.0** : www.usb.org/document-library/usb-20-specification
[3] **définition de la classe de dispositif de bus série universel pour les appareils audio, version 1.0** : www.usb.org/document-library/audio-device-document-10
[4] **spécification de la classe de dispositif audio du bus série universel pour les dispositifs audio de base, version 1.0** : www.usb.org/document-library/audio-device-class-spec-basic-audio-devices-v10-and-adopters-agreement
[5] **définition de la classe de dispositif de bus série universel pour les formats de données audio, version 1.0** : h www.usb.org/document-library/audio-data-formats-10
[6] **définition de la classe de dispositif de bus série universel pour les types de terminaux, version 1.0** : h www.usb.org/document-library/audio-terminal-types-10
[7] **AES3-2003** : Norme AES pour l'audio numérique - Interface entrée-sortie numérique - Format de transmission en série pour les données audio numériques à deux canaux représentés linéairement
[8] **Crystal Application None AN22: "Overview Of Digital Audio Interface Data Structures"** : https://statics.cirrus.com/pubs/appNote/an22.pdf
[9] **Sean Eron Anderson: "Bit Twiddling Hacks", section "Compute parity in parallel"** : http://graphics.stanford.edu/~seander/bithacks.html#ParityParallel
[10] **Microchip AN1422: "High-Quality Audio Applications Using the PIC32"** : http://ww1.microchip.com/downloads/en/AppNotes/01422A.pdf
[11] **Infrared Multi-Protocol decoder (IRMP)** : www.mikrocontroller.net/articles/IRMP
[12] **USB Device Class Definition for Human Interface Devices (HID), Version 1.11** : h www.usb.org/document-library/device-class-definition-hid-111
[13] **USB HID Usage Tables, Version 1.12** : www.usb.org/document-library/hid-usage-tables-112
[14] **protocoles de télécommande par IR :** www.elektormagazine.fr/magazine/elektor-200103/9101
[15] **page du projet sur Elektor Labs** : www.elektormagazine.com/labs/usb-spdif-interface-180027
[16] **page en ligne de cet article** : www.elektormagazine.fr/180027-04
[17] **logiciel sur GitHub** : https://github.com/kiffie/usb-spdif

multitâche en pratique avec l'ESP32 (4)

Sémaphores binaires

Warren Gay (Canada)

En multitâche dans FreeRTOS, il faut souvent synchroniser les tâches. Le sémaphore binaire est l'un des moyens de synchronisation. Cet article explore le sémaphore binaire et l'illustre par une métaphore de couples de danseurs.

Une métaphore du sémaphore ?

Un sémaphore est une fonction de multitâche qui permet à un appelant de bloquer la poursuite de sa propre exécution jusqu'à ce que soit *donnée* la permission de continuer. L'appelant tente de *prendre* le sémaphore. Si celui-ci est déjà *pris*, la tâche appelante attend et reste donc bloquée. Imaginez un bal populaire à l'ancienne. Plusieurs garçons souhaitent danser avec une fille elle-même en train de danser avec son cavalier ; elle est *prise*, les cavaliers intéressés doivent attendre qu'elle soit prête à se *donner* à l'un d'eux. Cela nous montre bien les deux propriétés d'un sémaphore *binaire* a deux états : il est pris ou donné.

Comment le sémaphore protège-t-il ?

Un sémaphore lui-même ne peut pas contrôler l'accès à une ressource quelconque. Ce n'est que par un *accord* que la protection des ressources peut être mise en œuvre avec un sémaphore. Si les danseurs ne respectent pas le principe *pris-donné*, plusieurs garçons s'arracheront la *fille* et se battront pour l'obtenir. Le sémaphore fonctionne selon un protocole : l'accédant accepte d'utiliser toute ressource offerte uniquement s'il réussit à *prendre* le sémaphore. Il n'utilisera pas non plus la ressource avant d'avoir *pris* le prochain sémaphore disponible.

Délais d'attente pour prendre

La patience d'un garçon qui attend n'est pas illimitée. Après p. ex. dix minutes d'attente, il essaiera de trouver une autre fille avec qui danser. Les sémaphores binaires permettent à l'appelant de spécifier une telle limite de durée d'attente, exprimée en unités de temps du système (tics). Si une tentative de prendre un sémaphore dure plus longtemps que le nombre de tics spécifié, l'appel sera renvoyé avec une notification d'échec. On peut aussi enjoindre à FreeRTOS d'attendre indéfiniment que le sémaphore soit *donné* (comme attendrait un cavalier évincé). Il est possible aussi de ne spécifier aucune limite, auquel cas la tentative échoue immédiatement si le sémaphore ne peut être *pris*. Pour résumer, les sémaphores peuvent :

- bloquer pour toujours, jusqu'à ce que le sémaphore puisse être *pris* par l'appelant ;
- bloquer pour un temps déterminé, à la fin duquel l'opération se solde par un échec ;
- échouer immédiatement si le sémaphore est *pris*.

Propriété et dons

Lorsqu'un sémaphore est *pris*, on dit qu'il *appartient* à la tâche. Le garçon qui danse avec la fille la *possède* effectivement. Après avoir dansé avec elle, il peut la *donner* soit directement à un autre danseur soit en la libérant : il n'y a jamais d'attente. De même, lorsqu'il *donne* un sémaphore, il n'y a aucun délai : le *don* d'un sémaphore *possédé* est immédiat.

Le fait de *donner* un sémaphore n'implique pas nécessairement qu'il sera *pris* immédiatement par une autre tâche. Une fois que le garçon a fini de danser avec la fille, elle redevient disponible, mais il n'y a pas nécessairement d'autres preneurs intéressés. Les autres garçons peuvent avoir renoncé à attendre et trouvé entre-temps des partenaires de danse différents.

État initial

Puisque notre analogie du bal vieillot fonctionne, tirons-en profit : la fille est créée à l'état *pris* par ses parents. Ce n'est que lorsque ses parents l'autorisent à aller au bal qu'elle est *donnée* et devient disponible. Le sémaphore binaire de FreeRTOS est lui aussi créé à l'état *pris* et diffère en cela des *mutex* de Linux ou de Windows auxquels vous pouvez le comparer, car ils sont créés dans un état *donné*. Nous reviendrons sur le mutex de FreeRTOS dans un prochain article.

xSemaphoreBinaryCreate()

La création d'un sémaphore binaire dans FreeRTOS est simple :

```
SemaphoreHandle_t h;

h = xSemaphoreCreateBinary();
assert(h)
```

Il n'y a pas d'arguments à fournir, et une poignée (*handle*) est rendue au sémaphore binaire créé. Il est possible que le *handle* soit renvoyé sous la forme `nullptr` (`NULL`) en cas d'épuisement de la mémoire disponible. Il est recommandé de vérifier la valeur retournée (la macro `assert()` de `assert.h` a été utilisée ici) pour s'assurer que le sémaphore a été créé.

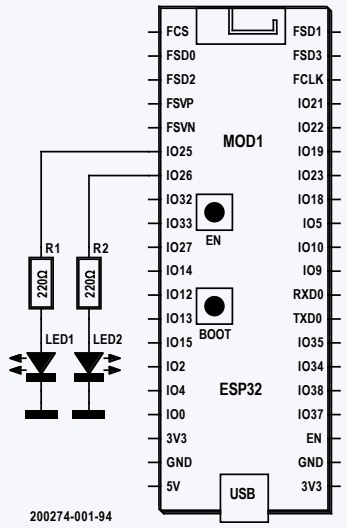


Figure 1. Schéma pour la démonstration de semaphores.ino.

xSemaphoreGive()

Donner un sémaphore est également simple :

```
SemaphoreHandle_t hMySemaphore ;
BaseType_t rc ; // code de retour
...
rc = xSemaphoreGive(hMySemaphore) ;
assert(rc == pdPASS) ;
```

L'opération *give* (= donner) peut échouer, en renvoyant `pdFAIL`, uniquement pour les raisons suivantes :
- poignée (`hMySemaphore`) invalide
- sémaphore déjà donné.

xSemaphoreTake()

Prendre un sémaphore est une opération potentiellement bloquante pour la tâche d'appel :

```
SemaphoreHandle_t hMySemaphore;
TickType_t wait = 30; // ticks
BaseType_t rc; // code de retour
```

Listage 1: Programme de démonstration semaphores.ino [1].

```
0001: // semaphores.ino
0002: // Practical ESP32 Multitasking
0003: // Binary Semaphores
0004:
0005: #define LED1_GPIO   25
0006: #define LED2_GPIO   26
0007:
0008: static SemaphoreHandle_t hsem;
0009:
0010: void led_task(void *argp) {
0011:   int led = (int)argp;
0012:   BaseType_t rc;
0013:
0014:   pinMode(led,OUTPUT);
0015:   digitalWrite(led,0);
0016:
0017:   for (;;) {
0018:     // First gain control of hsem
0019:     rc = xSemaphoreTake(hsem,portMAX_DELAY);
0020:     assert(rc == pdPASS);
0021:
0022:     for ( int x=0; x<6; ++x ) {
0023:       digitalWrite(led,digitalRead(led)^1);
0024:       delay(500);
0025:     }
0026:
0027:     rc = xSemaphoreGive(hsem);
0028:     assert(rc == pdPASS);
0029:   }
0030: }
0031:
0032: void setup() {
0033:   int app_cpu = xPortGetCoreID();
0034:   BaseType_t rc;  // Return code
0035:
0036:   hsem = xSemaphoreCreateBinary();
0037:   assert(hsem);
0038:
0039:   rc = xTaskCreatePinnedToCore(
0040:     led_task,   // Function
0041:     "led1task", // Task name
0042:     3000,       // Stack size
0043:     (void*)LED1_GPIO, // arg
0044:     1,          // Priority
0045:     nullptr,    // No handle returned
0046:     app_cpu);   // CPU
0047:   assert(rc == pdPASS);
0048:
0049:   // Allow led1task to start first
0050:   rc = xSemaphoreGive(hsem);
0051:   assert(rc == pdPASS);
0052:
0053:   rc = xTaskCreatePinnedToCore(
0054:     led_task,   // Function
0055:     "led2task", // Task name
0056:     3000,       // Stack size
0057:     (void*)LED2_GPIO, // argument
0058:     1,          // Priority
0059:     nullptr,    // No handle returned
0060:     app_cpu);   // CPU
0061:   assert(rc == pdPASS);
0062: }
0063:
0064: // Not used
0065: void loop() {
0066:   vTaskDelete(nullptr);
0067: }
```

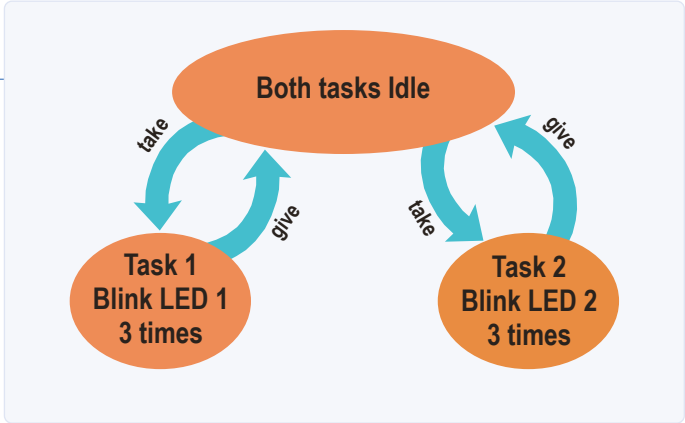


Figure 2. Les états des tâches d'exécution dans le programme semaphores.ino.

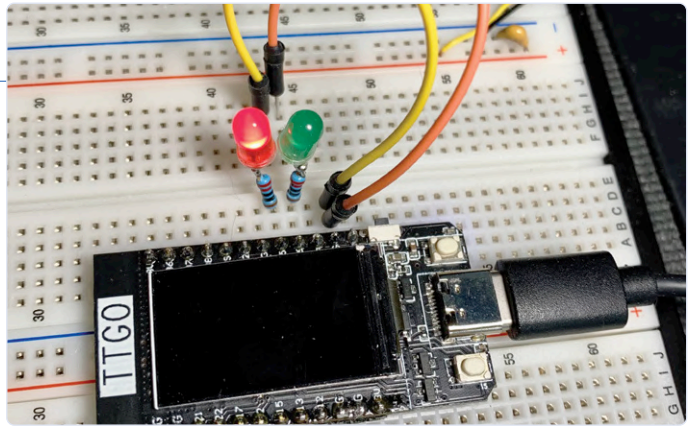


Figure 3. L'ESP32 TTGO pilotant deux LED à l'aide du programme semaphores.ino.

```
...
rc = xSemaphoreTake(hMySemaphore, wait);
// retourne pdPASS quand il est pris,
// sinon pdFAIL en cas de dépassement
```

L'opération de *prise* peut échouer si l'identifiant est invalide ou si l'appel a expiré (période définie par l'argument `wait`). Dans le cas contraire, l'appel bloquera l'exécution de la tâche appelante jusqu'à ce qu'elle ait *pris* le sémaphore.
Le paramètre `wait` peut avoir une des trois valeurs différentes selon les exigences de l'appelant :

- valeur macro `portMAX_DELAY` - attendre une éternité jusqu'à *pris*.
- valeur positive non nulle – attendre un nombre de tics correspondant à cette valeur.
- zéro - échec immédiat si le sémaphore ne peut pas être *pris*.

Démonstration

En guise de démonstration, deux tâches feront chacune clignoter leur propre LED (**listage 1**). En partageant un sémaphore binaire, une seule des tâches sera autorisée à faire clignoter sa LED à la fois. La tâche qui n'est pas en cours doit attendre que le sémaphore ait été *donné* par la tâche opposée en cours, ce qui permet de *prendre* le sémaphore une fois de plus. La **figure 1** illustre le schéma de branchement des LED pour tout ESP32 que vous choisirez d'utiliser.
La **figure 2** illustre les états de la paire de tâches s'exécutant dans le programme de démonstration. Lorsqu'une tâche prend le sémaphore, la tâche propriétaire est capable de s'exécuter et de faire clignoter sa LED, tandis que l'autre est bloquée en attente. Ce n'est que lorsque la tâche propriétaire rend le sémaphore que l'autre tâche peut le prendre et l'exécuter. La **figure 3** illustre un exemple de configuration de la carte d'expérimentation avec la carte de développement TTGO ESP32.
Dans la routine `setup()`, la ligne 36 crée le sémaphore et assigne la poignée à la variable statique globale `hsem`. Les lignes 39 à 46 créent la première tâche pour faire clignoter la LED1 (notez l'argument de la ligne 43 de l'appel de création de tâche). La broche GPIO à utiliser est passée comme un pointeur `void`. Elle est ensuite renvoyée à une valeur GPIO `int` dans la fonction de tâche `led_task()` (ligne 11), de la fonction de tâche. C'est abuser du pointeur que de l'utiliser pour passer une valeur, mais cela fonctionne tant que la valeur tient dans le même nombre de bits qu'une adresse de pointeur.
Après la création de la première tâche, nous *donnons* le sémaphore à la ligne 50. En faisant cela avant la création de la deuxième tâche, il est probable que la première tâche acquière le sémaphore en premier. Les lignes 53 à 60 créent ensuite la deuxième tâche. Les deux tâches exécutent le même code de fonction `led_task()`, mais avec des valeurs d'argument différentes spécifiant la broche GPIO de la LED à utiliser. Les lignes 14 et 15 configurent la LED utilisée dans le cadre de la tâche. La tâche entre alors dans une boucle sans fin à partir de la ligne 17. La première étape effectuée dans cette boucle est de *prendre* le sémaphore (ligne 19). Une seule tâche à la fois y parviendra.
La tâche qui réussit à prendre le sémaphore se poursuivra par la boucle des lignes 22 à 25. Cette boucle fait clignoter trois fois la LED de la tâche. Ensuite le sémaphore est *donné* dans la ligne 27, permettant à l'autre tâche de *prendre* le sémaphore. De cette façon, les tâches prennent à tour de rôle chacune le contrôle de l'autre.

Résumé

Le sémaphore binaire utilisé dans cette démonstration a fonctionné de deux manières différentes. Avant la première opération *give* de la fonction `setup()`, aucune des deux tâches ne peut être exécutée. Le sémaphore se comportait donc comme une barrière, car aucune tâche ne pouvait être exécutée. Après cette première opération *give* (= donne), l'une des deux tâches *prend* ce sémaphore, fait clignoter sa LED, puis *rend* le sémaphore. En opérant de cette manière, le sémaphore semble se comporter comme un *mutex*. Les deux tâches tentent continuellement d'exécuter leur code, mais, par l'exclusion mutuelle du sémaphore, une seule tâche à la fois est autorisée à faire clignoter sa LED.
Il est important de retenir que les sémaphores binaires sont créés à l'état *pris* (contrairement au mutex). Si le sémaphore de cette démonstration n'avait pas été *donné* initialement dans la routine `setup()` (ligne 50), les deux tâches seraient restées bloquées en attente sans fin. Dans FreeRTOS il existe d'autres primitives de synchronisation (y compris le *mutex*), mais le simple sémaphore binaire est parfois suffisant.

200274-03

LIEN

[1] https://github.com/ve3wwg/esp32_freertos/blob/master/semaphores/semaphores.ino

capteur de température sans fil pour le thermomètre à bargraphe Nixie

Solution polyvalente avec astuce de conception

Ilse Joostens (Belgique)

Après avoir lu l'article *thermomètre à bargraphe Nixie* d'Elektor [1], un lecteur du Midwest américain nous a gentiment fait remarquer que les températures dans sa région dépassent facilement 85 °F (29,5 °C) pendant plus de six mois de l'année. Dans ces circonstances, l'échelle de 50 à 85 °F (10 à 30 °C) de notre thermomètre est inutile la moitié du temps. C'est beaucoup. En voici donc une version remaniée qui couvre 15 à 105 °F (–10 à +40 °C). Au passage, nous avons transformé ce thermomètre en appareil sans fil. C'est parti.

INFOS SUR LE PROJET

Mots-clés
433 MHz, Linx, LPR, 1-Wire, Manchester, Nixie, capteurs

Niveau
débutant – connaisseur – expert

Durée
1,5 h environ

Outils
outils de laboratoire, fer à souder

Coût
75 € environ

SPÉCIFICATIONS (résumé)

- Alimentation du récepteur : 3,3 à 5 V
- Alimentation de l'émetteur : 3 V (pile CR2450)
- Fréquence radio sans licence : 433,92 MHz (USA/CAN : 315 MHz)
- Intervalle de TX-on : environ 2 mn, réglable
- Antenne demi-onde hélicoïdale imprimée
- Capteur de température : DS2438Z
- Plage de température: –20 °C à +70 °C (précision : ±2 °C)
- Modules RX-TX Linx Technologies OOK/ASK
- RX émule le capteur DS18B20 1-Wire mais sous 2,4 V
- Cartes RX et TX préassemblés

Au départ, nous nous attendions à une lecture plutôt difficile de la température en raison de l'imprécision de l'échelle du tube bargraphe IN-9. En pratique, cette crainte n'a pas été confirmée, la lecture est acceptable. Une telle échelle sur le thermomètre à bargraphe Nixie [1] est utile également pour afficher la température extérieure, du moins si vous vivez dans une région au climat plus tempéré que celui de notre correspondant américain. L'idée nous est venue de prévoir une connexion sans fil du capteur de température.

Émission & réception sans fil

Il existe de nombreux types de modules sans fil, entre quelques € l'unité à plusieurs dizaines, pour une paire composée d'un récepteur (RX) et d'un émetteur (TX). Par souci de simplicité et afin d'atteindre une portée raisonnable entre émetteur et récepteur, nous avons opté pour des modules à 433 MHz, homologués et sans licence.

Pour maintenir aussi bas que possible le prix du produit fini, nous avons d'abord testé les célèbres modules chinois FS1000A et MX-RM-5V, en commençant par trois paires achetées sur eBay. Puis nous avons fabriqué des prototypes de circuits imprimés pour les tester. Les modules fonctionnent, mais leur portée est décevante. Les principaux problèmes sont :

- l'émetteur exige 9 à 12 V de tension pour fournir suffisamment de puissance à l'antenne ;
- e récepteur est extrêmement sensible au bruit sur le rail d'alimentation.

Nous avons ensuite examiné d'autres modules (chinois) et constaté que la plupart, utilisant une simple modulation ASK ou OOK (*on-off keying / amplitude-shift keying*), provenaient de fabricants obscurs sans aucune garantie quant à leur disponibilité future. Une autre option consistait à utiliser des modules émetteurs-récepteurs avec interface SPI, mais il aurait fallu dans ce cas réécrire le microprogramme alors qu'il était presque terminé à ce moment-là.

Suit une autre mésaventure avec quelques modules FM 433 MHz, qui ont bien fonctionné. Malheureusement, **bien après l'acceptation de la commande et la livraison,** nous nous sommes aperçus (mieux vaut tard que jamais) que le fabricant en déconseillait l'utilisation pour de nouveaux projets. Finalement c'est la paire TXM-433-LR / RXM-433-LR de *Linx Technologies* qui a été retenue. Pas les moins chers, mais ils sont facilement disponibles en grandes quantités. Et vous économiserez de l'argent sur les piles, car l'émetteur en particulier est frugal. Les modules RF de Linx sont également disponibles en versions 315 MHz et 418 MHz, ce qui permet de les utiliser aux États-Unis, au Canada et dans d'autres pays où ces bandes ont une attribution ISM (industrielle/scientifique/médicale).

> « Astuce : DS2438Z+ = DS18B20 sous 2,4 V ! »

Émetteur - matériel

Au cœur du schéma de l'émetteur (**fig. 1**) se trouve un µC ATMEGA328P-AU cadencé à 8 MHz par un oscillateur à quartz. Oui, c'est le µC des cartes Arduino Uno et Nano.

Nous pensions à un capteur de température 1-Wire de type DS18B20, mais ce n'était pas le bon candidat pour un appareil alimenté par une pile de 3 V, car selon sa fiche technique [2], la tension d'alimentation minimale est … 3 V. Il existe bien d'autres capteurs de température qui fonctionnent bien avec une tension inférieure à 3 V, mais nous avons préféré un modèle 1-Wire doté d'un numéro de série unique de 64 bits, utilisé comme code d'identification de l'émetteur, ce qui permet de coupler le récepteur à un émetteur spécifique. Puis nous avons choisi le circuit intégré de commande de batterie 1-Wire de type DS2438Z+ avec son capteur de température intégré et sa tension d'alimentation jusqu'à 2,4 V.

À première vue, avec une erreur maximale de ±2 °C, la mesure du DS2438Z+ semble moins précise que celle du DS18B20. En pratique cependant, ses mesures sont acceptables.

Le module d'émission RF est du type TXM-433-LR de Linx Technologies [3], déjà mentionné. Il est constitué d'un VCO verrouillé par un synthétiseur de fréquence dont la référence est un quartz de précision. La sortie du VCO est amplifiée et tamponnée par un amplificateur de puissance interne capable de fournir +10 dBm, soit 10 mW dans une charge de 50 Ω. À pleine puissance avec un rapport cyclique d'émission de 50 %, la consommation est d'environ 5 mA, ce qui est faible par rapport à des modules RF similaires. Sauf l'antenne, il n'y a aucun composant RF externe. Nous avons utilisé une antenne demi-onde hélicoïdale sur circuit imprimé, montée perpendiculairement au plan de

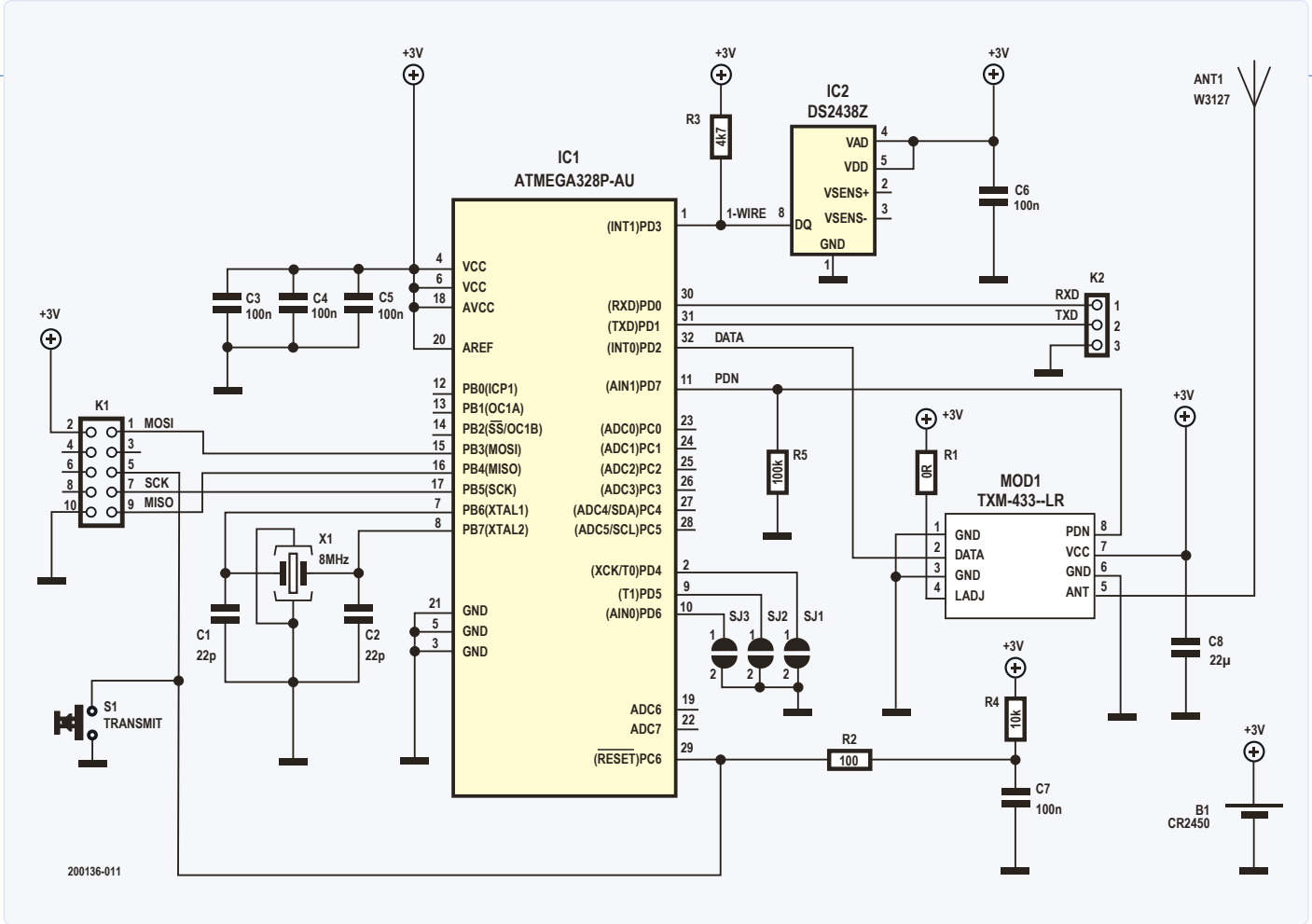


Figure 1. Schéma de l'émetteur de température. Pour que le circuit fonctionne jusqu'à 2,4 V, un capteur DS2438Z+ émule un DS18B20.

masse. La puissance rayonnée est réglable à l'aide de R1. Par défaut, une résistance de 0 Ω est montée pour une puissance de sortie maximale, mais il peut être nécessaire de réduire la puissance RF pour se conformer à la réglementation locale. Veuillez vous référer à la fiche technique du TXM-433-LR pour plus d'informations.
Les autres composants du circuit imprimé de l'émetteur sont :

- bouton de réinitialisation
- connecteur de programmation K1
- connecteur de débogage (série) K2
- cavaliers à souder SJ1, SJ2, SJ3
- pile CR2450 (alimentation)

Les connecteurs K1 et K2 n'ont aucune utilité en fonctionnement normal et ne sont pas montés. Les cavaliers à souder SJ1, SJ2, SJ3 permettent de régler l'intervalle de transmission entre 2 mn 8 s et 3 mn 4 s. Cela réduit le risque que plusieurs émetteurs envoient des données au même moment, ce qui pourrait entraîner une collision de données.

Nous avons opté pour une pile au lithium CR2450, pour sa capacité (mAh) plus grande que celle du modèle CR2032, à un coût légèrement supérieur (sur eBay). Avec notre prototype, la tension de la batterie était encore de 2,991 V après six mois d'utilisation continue. Lors d'un transfert de données, on a constaté que cette tension baissait brièvement à environ 2,93 V. L'intervalle de transmission utilisé était de 2 mn 24 s.

Émetteur - logiciel

La plupart du temps, le microcontrôleur est en veille (*Power-Down*) avec la minuterie du chien de garde en marche. La minuterie du chien de garde est réglée sur son délai d'attente maximum d'environ 8 s. En fin de temporisation, la minuterie produit une interruption pour réveiller le µC . Un compteur en RAM est alors décrémenté. Puis, si la valeur n'a pas encore atteint zéro, la minuterie du chien de garde est relancée et le µC repasse en mode de veille. Ce processus nécessite environ 20 instructions.
Si le compteur atteint zéro après une interruption par la minuterie du chien de garde, les registres de température et le numéro de série de 64 bits du DS2438Z+ sont lus. En cas d'échec, le logiciel tente à nouveau de lire le DS2438Z+ quatre fois de plus. Le logiciel prend également en charge les capteurs de température DS18B20, DS18S20 et DS1822. Cela peut être utile si l'émetteur est alimenté par une alimentation de 3,3 V au lieu d'une pile au lithium.
Lorsque le DS2438Z+ est présent, le code de famille 1-Wire est modifié pour correspondre au code de famille d'un DS18B20. Le CRC est également recalculé pour correspondre au nouveau code de famille.
Le numéro de série de 64 bits avec le nouveau code de famille, la valeur de température et le CRC sont ensuite transmis deux fois par le module RF précédé d'un préambule. Les modules RF utilisent la modulation OOK/ASK, ce qui signifie que l'onde porteuse est activée et désactivée en même temps que les données appliquées au module. Notez que vous ne pouvez pas simplement alimenter le module émetteur avec des données série et

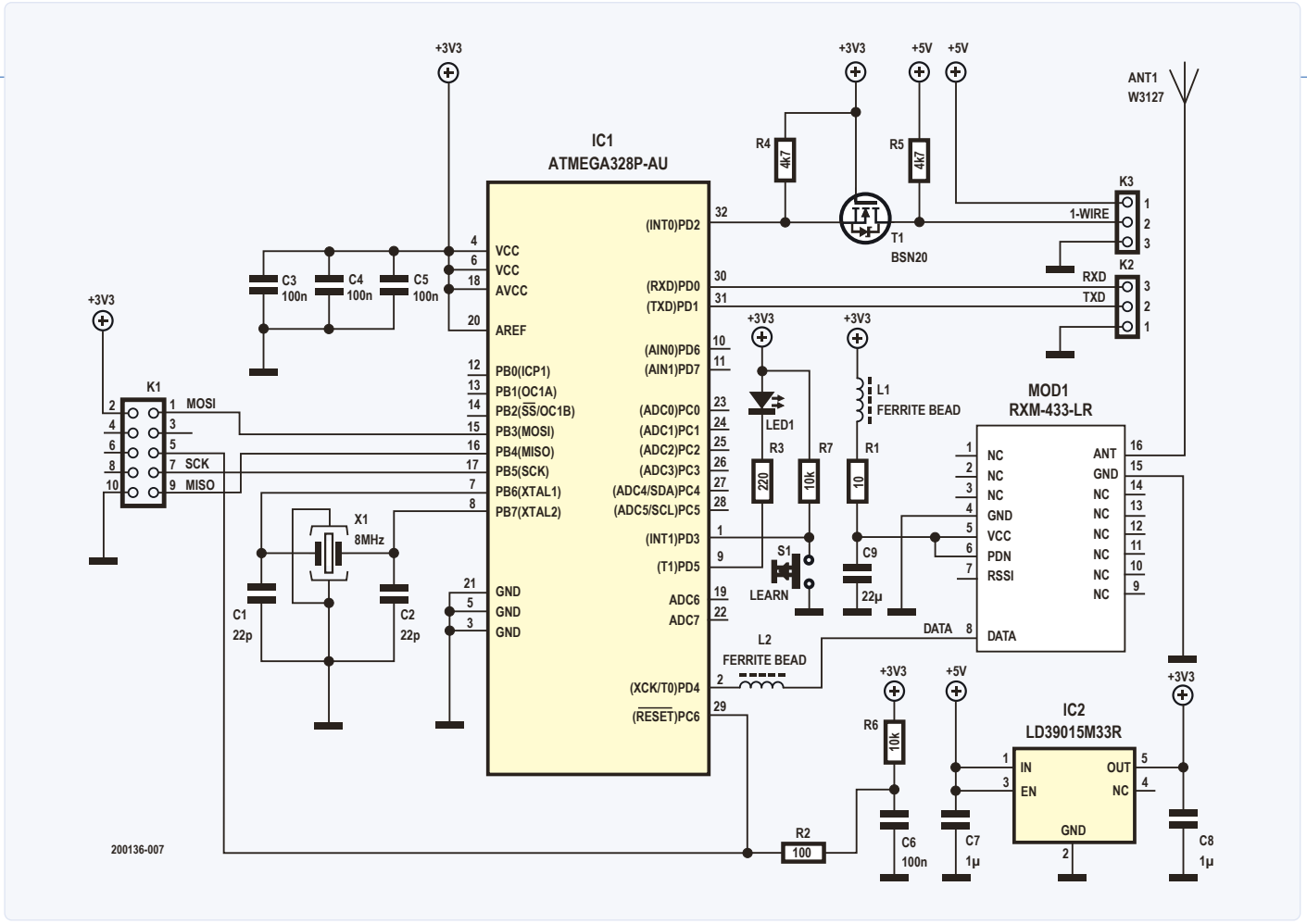


Figure 2. Schéma du récepteur de température. Les principales tâches du microprogramme sont le traitement des signaux radio démodulés et l'interprétation des commandes 1-Wire.

vous attendre à ce qu'elles arrivent parfaitement au récepteur. En l'absence de porteuse de 433 MHz, le récepteur peut capter des bruits de fond ou des interférences d'autres appareils fonctionnant également à 433 MHz. Pour atténuer ces phénomènes, les données doivent d'abord être codées et précédées par un «préambule» qui permet au récepteur de détecter un code connu et de se synchroniser avec lui avant de décoder les données. Parmi les normes de codage les plus populaires, on trouve *Manchester* [4] et la bibliothèque *RadioHead* disponible pour Arduino [5]. Les données ne sont pas cryptées pendant la transmission, car l'objectif principal est de mesurer la température extérieure, une information qui n'a rien de sensible.

Après la transmission, l'état des cavaliers SJ1-SJ3 est lu et le compteur du nombre de cycles de sommeil en RAM est réglé sur une valeur de 16 plus le nombre binaire représenté par les cavaliers (0-7, pour un nombre total de cycles de sommeil entre 16 et 23). La minuterie du chien de garde est alors lancée, le µC repasse en veille et le cycle se répète.

Après une réinitialisation, le cycle complet est également lancé. Ainsi, le fait d'appuyer sur le bouton de réinitialisation déclenche une transmission de données sans fil, ce qui est utile lorsqu'on couple un récepteur avec un émetteur.

Récepteur - matériel

Le schéma du récepteur (**fig. 2**) ressemble à celui de l'émetteur et est également basé sur un µC ATMEGA328P. Les seules connexions avec le monde extérieur sont Vcc, GND et DQ (données 1-Wire).

Le récepteur partage l'antenne avec l'émetteur. Un module récepteur RXM-433-LR de *Linx Technologies* est utilisé [6]. Ce module de AM/OOK est doté d'une architecture superhétérodyne à conversion unique avancée, qui se traduit par une sensibilité exceptionnelle et une portée remarquable.

Le bruit et l'ondulation de l'alimentation peuvent affecter considérablement la sensibilité du récepteur. Pour garantir que la tension d'alimentation fournie au récepteur soit aussi propre que possible, la perle de ferrite L1 et la résistance R1 de 10 Ω sont connectées en série avec la broche Vcc du module, et un condensateur de découplage C9 de 22 µF a été ajouté. Pour éviter les interférences du µC, une perle de ferrite L2 a également été ajoutée à la ligne de données.

Étant donné que la tension d'alimentation du RXM-433-LR est de 2,7 à 3,6 V, nous avons choisi de faire fonctionner tout le circuit sous 3,3 V. Le thermomètre à bargraphe fonctionne cependant sous 5 V. Pour permettre une connexion facile entre le récepteur et le thermomètre, un régulateur de tension à très faible chute de tension et à faible bruit de type LD39015M33R (IC2) a été ajouté ainsi qu'un convertisseur bidirectionnel de niveau logique autour du MOSFET T1.

Lorsque le récepteur est utilisé avec un bus 1-Wire de 3,3 V comme avec l'horloge Elektor/Axiris IV-22 VFD [7], la tension en aval du régulateur suffit pour que le circuit fonctionne correctement.

Outre un connecteur de programmation (non monté normalement) et de débogage, une LED et un bouton poussoir sont connectés

LISTE DES COMPOSANTS

Émetteur

Résistances

R1 = 0 Ω, 1206 *
R2 = 100 Ω, 1206
R3 = 4kΩ7, 1206
R4 = 10 kΩ, 1206
R5 = 100 kΩ, 1206

Condensateurs

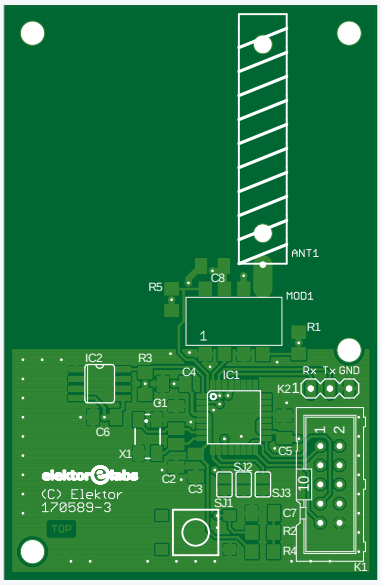
C1,C2 = 22 pF, C0G/NP0, 1206
C3-C7 = 100 nF, X7R, 1206
C8= 22 µF, X5R, 10 V, 1206, Kemet C1206C226M8PACTU

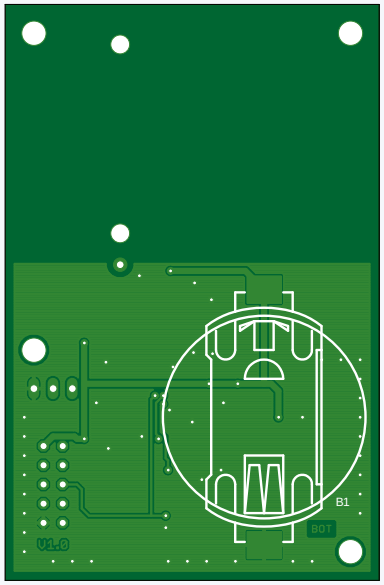
Semi-conducteurs

IC1 = ATMEGA328P-AU, programmé
IC2 = DS2438Z+
MOD1 = TXM-433-LR, Linx Technologies (USA/Canada : TXM-315-LR)

Divers

ANT1 = antenne hélicoïdale 433 MHz sur circuit imprimé, Pulse Electronics de type W3127 (USA/Canada : W3126)
B1 = support de pile CR2450, type Renata SMTU 2450N-1-LF, avec pile au lithium CR2450





K1 = connecteur à 10 broches (2x5), pas de 2,54 mm (facultatif)
K2 = barrette mâle à 3 broches, verticale, pas de 2,54 mm (facultatif)
S1 = bouton-poussoir SMT, TE Connectivity FSM6JSMA
X1 = quartz 8 MHz, Abracon ABMM2-8.000MHZ-E2-T ou Würth Elektronik # 830055663

* voir le texte

Remarque : l'émetteur est disponible en kit dans la boutique en ligne d'Elektor.

LISTE DES COMPOSANTS

Récepteur

Résistances

R1 = 10 Ω, 1206
R2 = 100 Ω, 1206
R3 = 220 Ω, 1206
R4,R5 = 4kΩ7, 1206
R6,R7 = 10 kΩ, 1206

Condensateurs

C1,C2 = 22 pF, C0G/NP0, 1206
C3-C6 = 100 nF, X7R, 1206
C7,C8 = 1 µF, X7R, 1206
C9 = 22 µF, X5R, 10 V, 1206, Kemet C1206C226M8PACTU

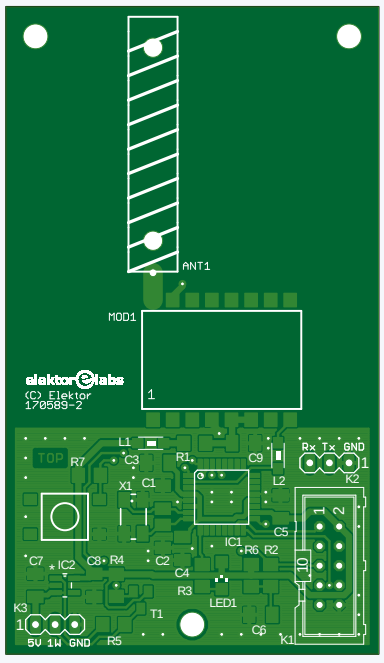
Semi-conducteurs

LED1 = verte, 1206, Broadcom Limited type HSMG-C150
IC1 = ATMEGA328P-AU, programmé
IC2 = LD39015M33R
MOD1 = RXM-433-LR, Linx Technologies (USA/Canada : RXM-315-LR)
T1 = BSN20

Divers

ANT1 = antenne hélicoïdale 433 MHz sur circuit imprimé, Pulse Electronics de type W3127 (USA/Canada : W3126)
K1 = connecteur à 10 broches (2x5), pas de 2,54 mm (facultatif)
K2 = barrette mâle à 3 broches, verticale, pas de 2,54 mm (facultatif)
K3 = connecteur à 3 broches, coudé, 2,54 mm avec connecteur femelle # 61900311621 pré-câblé avec 3 câbles # 619100126015 (Würth Elektronik)
L1,L2 = perle de ferrite, type Fair-rite 1206 2512067007Y3
S1 = bouton-poussoir SMT, TE Connectivity FSM6JSMA
X1 = quartz 8 MHz, Abracon ABMM2-8.000MHZ-E2-T ou Würth Elektronik # 830055663

Remarque : l'émetteur est disponible en kit dans la boutique en ligne d'Elektor.



au µC. Chaque fois qu'un paquet de données sans fil est reçu, la LED change d'état. Cela vous permet de vérifier si l'émetteur est toujours fonctionnel. Le bouton est utilisé pour coupler le récepteur avec un émetteur.

Récepteur - logiciel

Jusqu'à récemment, le bus 1-Wire de *Dallas Semiconductors* (*Maxim*) était très populaire dans des réseaux MicroLAN destinés aux systèmes domotiques par exemple. L'utilisation accrue des solutions sans fil a rendu obsolète le bus 1-Wire. Malgré cela, certains dispositifs 1-Wire tels que le DS18B20 (et les capteurs de température similaires) sont encore largement utilisés. Ils sont bon marché, précis, faciles à interfacer avec un µC et disponibles dans des boîtiers TO-92 faciles à souder ! Ce sont les principales raisons de leur utilisation dans les projets Elektor de thermomètre à bargraphe et d'horloge VFD IV-22, ainsi que pour simuler une DS18B20 avec notre module récepteur.

Le logiciel écoute le module de réception RF ainsi que le bus 1-Wire. Lorsqu'une transmission de données arrive au récepteur, les données codées sont décodées et leur validité est vérifiée. Si un paquet de données valide est reçu, le numéro de série 1-Wire reçu est ensuite comparé avec le numéro de série 1-Wire stocké dans l'EEPROM du µC. Lorsque les deux correspondent, le bloc-notes (contenant entre autres les octets de température) du capteur DS18B20 simulé est mis à jour, sinon le paquet de données est rejeté.
L'esclave 1-Wire simulé prend en charge les commandes 1-Wire suivantes (c'est-à-dire les commandes de fonction ROM et DS18B20) :

- `Read Rom` [33h]
- `Match Rom` [55h]
- `Search Rom` [F0h]
- `Convert T` [44h]
- `Write Scratchpad` [4Eh]
- `Read Scratchpad` [BEh]
- `Copy Scratchpad` [48h]
- `Recall E2` [B8h]
- `Read Power Supply` [B4h]

Il n'est pas possible d'alimenter le module récepteur en mode parasite. La commande `Read Power Supply` renvoie toujours l'état «alim externe» ('= *externally powered*').
Notez que les fonctions 1 Wire esclave du récepteur ont priorité sur les fonctions RF. Si une communication par bus 1 Wire est lancée alors qu'un paquet de données est reçu et décodé, la réception de données RF est interrompue. Comme le thermomètre et l'horloge VFD ne lisent la valeur de la température qu'une fois par minute, les chances que cela se produise régulièrement sont relativement faibles. C'est également la raison pour laquelle l'intervalle de transmission le plus court de l'émetteur est de 2 mn 8 s et non pas de 2 mn.

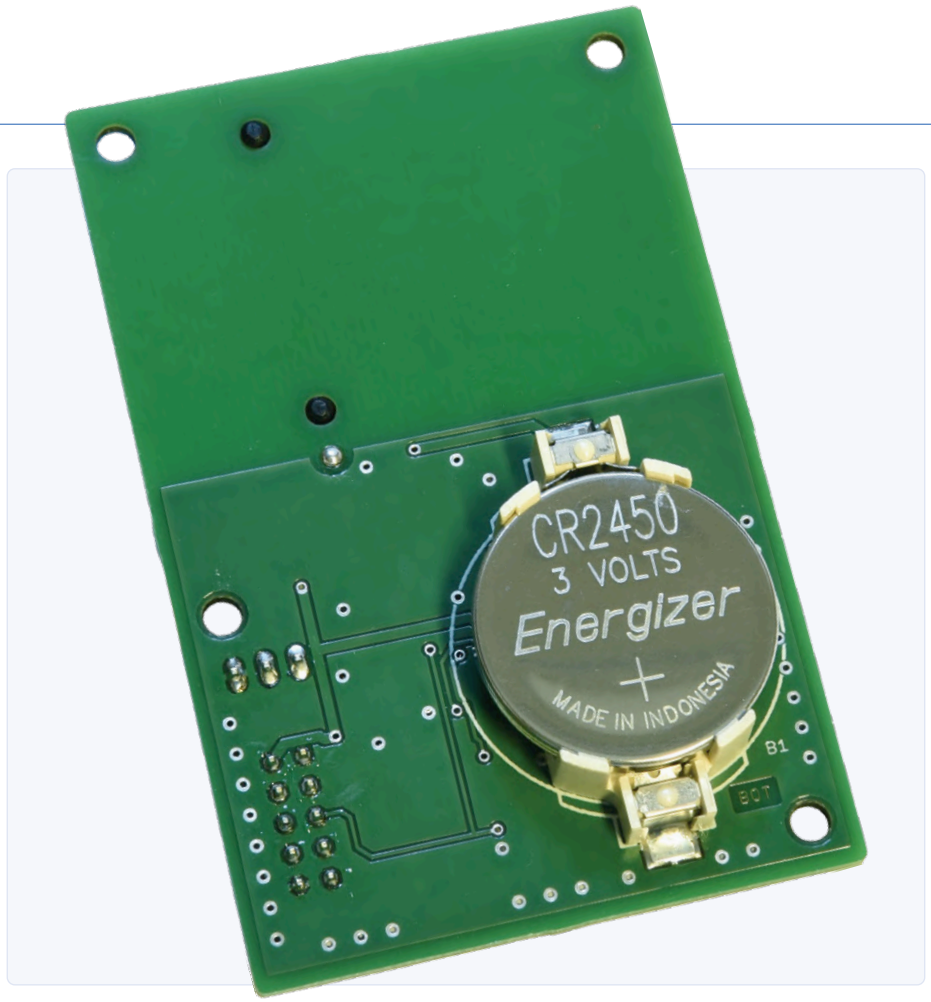


Figure 3. Une pile au lithium de 3 V de type CR2450 est suggérée pour l'émetteur. Le modèle 2450 est un peu plus cher que le CR2032 mais offre une plus grande capacité.

Construction et utilisation

Le module émetteur et le module récepteur sont tous deux disponibles chez Elektor sous forme de circuits imprimés assemblés et testés. Le plan d'implantation des composants pour les PCB et les listes de composants associées sont imprimés ici par souci d'exhaustivité et pour permettre un certain degré d'ouverture technique à ceux qui souhaitent acheter leurs composants eux-mêmes. De même, les fichiers Gerber du circuit imprimé et le micrologiciel ATmega du récepteur et de l'émetteur sont téléchargeables [8].
Quelques préparatifs sont nécessaires pour faire fonctionner les cartes TX et RX préassemblées. Montez K3, un connecteur à 3 broches coudé de 0,1 pouce et faites glisser les trois fils précâblés dans le connecteur femelle à 3 voies par l'arrière jusqu'à ce qu'ils s'enclenchent en place. Guidez les fils à travers le panneau latéral du coffret et soudez-en l'autre extrémité à l'embase de la carte de réception. Isolez ces connexions avec du tube thermorétractable (oui, il faut y penser avant !).

Vérifiez les connexions GND, DQ et 5 V, aussi du côté du thermomètre. Montez maintenant le récepteur dans son coffret, connectez-le au thermomètre et alimentez-le. Le thermomètre affiche alors la pleine échelle, car la température d'initialisation du DS18B20 simulé est de 85 °C, comme celle du DS18B20 réel. Si le récepteur n'a jamais été couplé à un émetteur, l'EEPROM peut ne pas contenir de numéro de série 1-Wire. Dans ce cas, les fonctions esclaves 1-Wire sont suspendues, le tube IN-9 n'affiche rien.
Placez maintenant une pile au lithium CR2450 dans le support de pile situé à l'arrière de

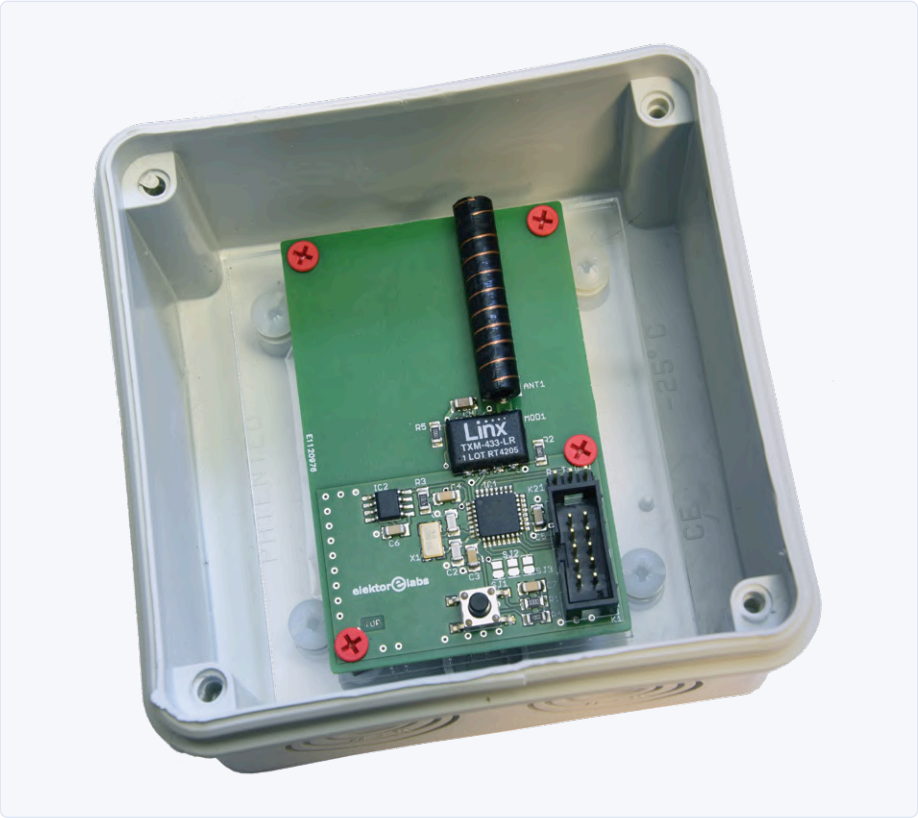


Figure 4. Pour l'extérieur, l'émetteur a été conçu pour tenir dans une boîte de dérivation électrique.

PRODUITS

- **Module récepteur avec échelle et câble en acrylique révisé**
  www.elektor.fr/170589-72
- **Module émetteur avec batterie et boîtier étanche**
  www.elektor.fr/170589-73
- **Thermomètre à bargraphes Nixie (Kit) (version originale)**
  www.elektor.fr/nixie-bargraph-thermometer-170589-71
- **Horloge à tubes VFD avec ESP32 DevKit-C, boîtier en acrylique inclus**
  www.elektor.fr/vfd-tube-clock-with-esp32-devkit-c-170537-71

l'émetteur (**fig. 3**). Placez l'émetteur à quelques mètres du récepteur. Appuyez sur le bouton du récepteur. La LED verte clignotera, indiquant le mode d'apprentissage du récepteur. Appuyez sur le bouton de l'émetteur. Si tout va bien, la LED du récepteur cessera de clignoter, indiquant qu'une valeur de température est reçue et que le récepteur est maintenant couplé à l'émetteur. Comme le thermomètre ne lit le capteur 1-Wire qu'une fois par minute, ça durera un peu avant que ne s'affiche la température réelle. Pour l'extérieur, installez l'émetteur dans un coffret étanche avec un petit sac de gel de silice pour le garder au sec. Une solution peu coûteuse consiste à utiliser une boîte de câblage électrique avec un indice de protection IP65 (**fig. 4**).
Quand vous allumez le thermomètre, il peut s'écouler quelques minutes avant que la température ne soit reçue et affichée. Pendant ce temps, le thermomètre affiche sa valeur à pleine échelle.

Il est possible d'utiliser plusieurs émetteurs et récepteurs en même temps. Comme l'émetteur remplace le code de famille du DS2438Z+ par celui d'un DS18B20, il y a une chance infime que le numéro de série du DS18B20 simulé ne soit pas unique lorsque l'on utilise un ou plusieurs récepteurs dans un réseau 1-Wire avec d'autres capteurs de température DS18B20 (réels) présents.

Au fil de nos essais, nous avons remarqué que certaines cartes Arduino Nano provoquent plus d'interférences (RFI) que d'autres. Ces interférences, ainsi que les matériaux de construction et les structures métalliques sur le trajet entre TX et RX, peuvent affecter la portée réelle.

200136-03

LIENS

[1] **Thermomètre à bargraphe Nixie, Elektor, juillet-août 2018** : www.elektormagazine.fr/magazine/elektor-201807/41755
[2] **DS2438 Smart Battery Monitor — fiche technique** : https://datasheets.maximintegrated.com/en/ds/DS2438.pdf
[3] **Linx LR Series Transmitter Module Data Guide** : www.linxtechnologies.com/wp/wp-content/uploads/txm-fff-lr.pdf
[4] **Manchester Data Encoding for Radio Communications** : www.maximintegrated.com/en/design/technical-documents/app-notes/3/3435.html
[5] **RadioHead Packet Radio library for embedded microprocessors** : www.airspayce.com/mikem/arduino/RadioHead/
[6] **Linx LR Series Receiver Module Data Guide** : www.linxtechnologies.com/wp/wp-content/uploads/rxm-fff-lr.pdf
[7] **horloge à tubes VFD avec ESP32, Elektor, mai-juin 2018** : www.elektormagazine.fr/magazine/elektor-201805/41576
[8] **Ressources en ligne de cet article** : www.elektormagazine.fr/200136-03

multitâche avec Raspberry Pi

Savoir-faire : commande de feux de circulation

Dogan Ibrahim (Royaume Uni)

Voici un exemple de projet extrait d'ouvrage publié récemment par Elektor. Intitulé *Multitasking with RPi*, ce livre est destiné aux étudiants, aux ingénieurs en exercice et aux amateurs intéressés par le les projets multitâches utilisant le langage Python3 sur l'ordinateur Raspberry Pi (RPI).

Le multitâche est devenu l'un des sujets les plus importants dans les systèmes à base de µC, notamment dans les automatismes. Avec l'accroissement de leur complexité, on attend plus des projets, ce qui, sur le même système, impose l'utilisation de plusieurs tâches interdépendantes et partageant l'unité centrale (ou de multiples UC) pour effectuer les opérations. D'où l'importance croissante du fonctionnement multitâche dans les applications à base de µC. Beaucoup de projets d'automatisme complexes font appel à un noyau multitâche. Dans le livre, on utilise le langage Python avec le Raspberry Pi (RPI). On peut aussi utiliser d'autres modèles de RPi sans rien changer au code.

Le livre repose sur des projets. Son objectif principal est d'enseigner les principes de base du multitâche avec Python sur RPi. Il présente de nombreux projets dûment testés avec les modules multitâches de Python. Chaque projet est décrit et commenté en détail. Les listages complets des programmes sont fournis pour chaque projet. Les lecteurs devraient pouvoir les utiliser tels quels, ou les adapter à leurs propres besoins.

Savoir-faire : commande de feux de circulation

Ce projet consiste à concevoir une commande simple des feux de circulation d'un carrefour, à l'intersection de deux routes : *East Street* et *North Street*. Il y a des feux à chaque branche du carrefour. Des boutons d'appel pour piétons se trouvent près des feux sur *North Street*. L'appui sur un bouton d'appel provoque le passage au rouge de tous les feux à la fin de leurs cycles. Un signal sonore est alors déclenché pour indiquer aux piétons qu'ils peuvent traverser sans danger. Un afficheur LCD indique le cycle *piétons* ou *circulation* en cours. La **figure 9.12** montre la disposition au carrefour.

Pour ce projet, les durées fixes suivantes sont affectées à chaque feu, ainsi que la durée du signal sonore pour piétons. Pour simplifier, on suppose que les deux routes du carrefour ont les mêmes durées (en secondes) :

- rouge : 19 s
- orange : 2 s
- vert : 15 s
- orange + rouge : 2 s
- piétons : 10 s

La durée totale du cycle des feux est fixée à 38 s.

La séquence adoptée pour ces feux de circulation est la suivante (qui n'est peut-être pas celle dont vous êtes familier dans votre pays) : Rouge ➞ Orange+Rouge ➞ Vert - Orange... Vert ➞ Orange ➞ Rouge ➞ Orange+Rouge...

La **figure 9.13** montre le diagramme du projet, la **figure 9.14** le schéma. Les LED rouge (R), orange (O) et verte (V) sont utilisées pour représenter les vrais feux de circulation.

Les connexions suivantes sont réalisées entre le RPi et les éléments de signalisation routière :

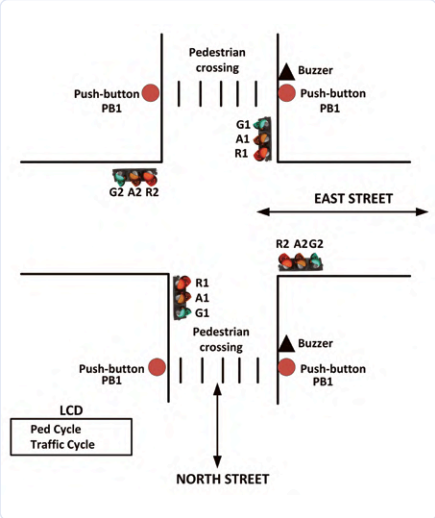


Figure 9.12. Signalisation au carrefour.

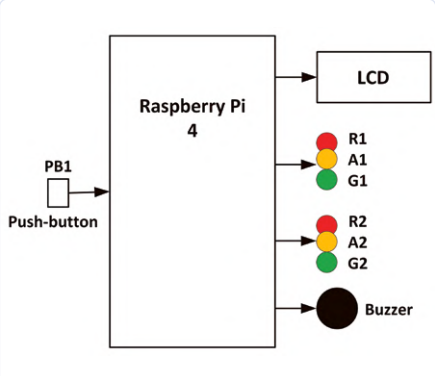


Figure 9.13. Diagramme du projet.

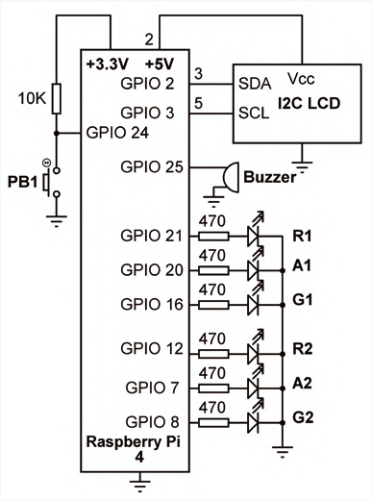


Figure 9.14. Schéma du circuit.

Raspberry Pi	Signalisation
GPIO 21	LED R1
GPIO 20	LED A1
GPIO 16	LED G1
GPIO 12	LED R2
GPIO 7	LED A2
GPIO 8	LED G2
GPIO 25	signal sonore
GPIO 2	LCD SDA
GPIO 3	LCD SCL
GPIO 24	PB1 (bouton d'appel)

La **figure 9.15** montre le listage complet (nom du programme : traffic.py ; téléchargement [1]). En début de programme, on importe les modules nommés *RPi, time, I²C LCD driver* et *multiprocessing* et on crée deux files nommées `pedq` et `lcdq`.
Deux fonctions nommées `ONOF` et `CONF_OUT` sont définies dans le programme. La fonction `ONOF` a deux arguments : `port` et `state`. Cette fonction envoie l'état (0 ou 1) au port GPIO spécifié. La fonction `CONF_OUT` a un paramètre nommé *port*. Cette fonction configure le port GPIO spécifié en mode sortie.
Il y a deux processus dans le programme : `Lights` (= feux) et `Pedestrian`.(= piéton) Au début du processus `Lights`, on définit les connexions entre le RPi, les LED (feux de circulation) et le buzzer puis ces ports sont configurés en sorties. De plus, tous ces ports sont mis à zéro pour éteindre toutes les LED et le buzzer. Enfin, les LED sont séquencées

Figure 9.15: Listage du programme traffic.py

```
#-------------------------------------------------------------------
#                    Commande de feux de circulation
#                    ===============================
#
# Il s'agit d'un projet de commande des feux de circulation d'un
# carrefour. On utilise 6 LED pour représenter les feux de
   circulation.
# On utilise de plus un bouton pour le passage des piétons, et un
# afficheur LCD indique en permanence l'état des feux.
#
# Auteur  : Dogan Ibrahim
# Fichier : traffic.py
# Date    : Mai 2020
#-------------------------------------------------------------------
import RPi.GPIO as GPIO                      # Importe RPi
import multiprocessing                       # Importe multiprocessing
import time                                  # Importe time
import RPi_I2C_driver                        # Bibliothèque I2C
LCD = RPi_I2C_driver.lcd()                   # Importe LCD
GPIO.setwarnings(False)
GPIO.setmode(GPIO.BCM)                       # GPIO en mode BCM
pedq = multiprocessing.Queue()               # Création de la file pedq
lcdq = multiprocessing.Queue()               # Création de la file lcdq

#
# Cette fonction envoie la donnée " state " (0 ou 1) au port spécifié
#
def ONOF(port, state):
   GPIO.output(port, state)

#
# Cette fonction configure le port spécifié en sortie
#
def CONF_OUT(port):
   GPIO.setup(port, GPIO.OUT)

#
# Processus de commande des feux
#
def Lights():                                # Processus Lights
   R1=21; A1=20; G1=16                       # Connexions des LED
   R2=12; A2=7;  G2=8                        #
   Buzzer=25                                 # Connexion du buzzer
   CONF_OUT(R1); CONF_OUT(A1); CONF_OUT(G1)  # Configuration en mode
   CONF_OUT(R2); CONF_OUT(A2); CONF_OUT(G2)  # sortie et mise à zéro
   CONF_OUT(Buzzer)                          # de tous les ports
   ONOF(R1,0); ONOF(A1,0); ONOF(G1,0); ONOF(R2,0); ONOF(A2,0); ONOF(G2,0)
   ONOF(Buzzer, 0)

   RedDuration = 15
   GreenDuration = 15
   AmberDuration = 2

#
# Commande de la séquence des feux de circulation
#
   while True:                               # Boucle infinie
      ONOF(R1,0); ONOF(A1,0); ONOF(G1,1); ONOF(R2,1); ONOF(A2,0);
    ONOF(G2,0)
      time.sleep(RedDuration)
      ONOF(G1,0); ONOF(A1,1)
      time.sleep(AmberDuration)
      ONOF(A1,0); ONOF(R1,1); ONOF(A2,1)
      time.sleep(AmberDuration)
      ONOF(A2,0); ONOF(R2,0); ONOF(G2,1)
      time.sleep(GreenDuration)
```

```
      ONOF(G2,0); ONOF(A2,1)
      time.sleep(AmberDuration)
      ONOF(A2,0); ONOF(A1,1); ONOF(R2,1)
      time.sleep(AmberDuration)

      while not pedq.empty():                # Si appel piétons
         lcdq.put(1)                         # Mise en file lcdq
         ONOF(G1,0); ONOF(R1,1); ONOF(A1,0)  # Feu 1 rouge seul
         ONOF(G2,0); ONOF(R2,1); ONOF(A2,0)  # Feu 2 rouge seul
         d = pedq.get()                      # Vide la file pedq
         ONOF(Buzzer, 1)                     # Active le buzzer
         time.sleep(10)                      # Attend 10 secs
         ONOF(Buzzer, 0)                     # Désactive le buzzer
         d = lcdq.get()                      # Vide la file lcdq


def Pedestrian():                            # Processus Pedestrian
   PB1 = 24
   GPIO.setup(PB1, GPIO.IN)                  # PPB1 en mode entrée

   while True:                               # Boucle infinie
      while GPIO.input(PB1) == 1:            # Attente PB1 appuyé
         pass
      pedq.put(1)                            # Mise en file pedq
      while GPIO.input(PB1) == 0:            # Attente PB1 relâché
         pass

#
# Création des processus
#
p = multiprocessing.Process(target = Lights, args = ())
q = multiprocessing.Process(target = Pedestrian, args = ())
p.start()
q.start()

#
# Commande de LCD. Affichage de " Ped Cycle " ou de  " Traffic Cycle "
#
LCD.lcd_clear()                              # Effacement du LCD
LCD.lcd_display_string("TRAFFIC CONTROL", 1) # En-tête

while True:                                  # Boucle infinie
   if not lcdq.empty():
      LCD.lcd_display_string("Ped Cycle    ", 2)
   else:
      LCD.lcd_display_string("Traffic Cycle", 2)
   time.sleep(1)
```

Figure 9.16. Exemple d'affichage.

dans le bon ordre avec les bonnes durées. Vers la fin de la fonction, on vérifie si le bouton *piétons* a été pressé. Si c'est le cas, la file `pedq` n'est pas vide. Pendant le cycle *piétons*, les feux rouges des deux rues sont allumés pour arrêter la circulation et céder le passage aux piétons.
Le buzzer est également activé pendant 10 s pendant le cycle *piétons* pour les avertir qu'ils peuvent traverser sans danger.
Le processus `Pedestrian` surveille en permanence le bouton PB1. Si le bouton est pressé, un « 1 » est envoyé à la file `pedq` pour que le processus *Lights* puisse facilement détecter cette action et commencer le cycle *piétons*.
Le programme principal commande l'afficheur LCD. Au démarrage, le message « TRAFFIC CONTROL » est affiché sur la première ligne.
La seconde vérifie en permanence la file `lcdq` et affiche soit « Ped Cycle » (cycle *piétons*) soit « Traffic Cycle » (cycle *circulation*).
La **figure 9.16** montre un exemple d'affichage sur l'afficheur LCD.

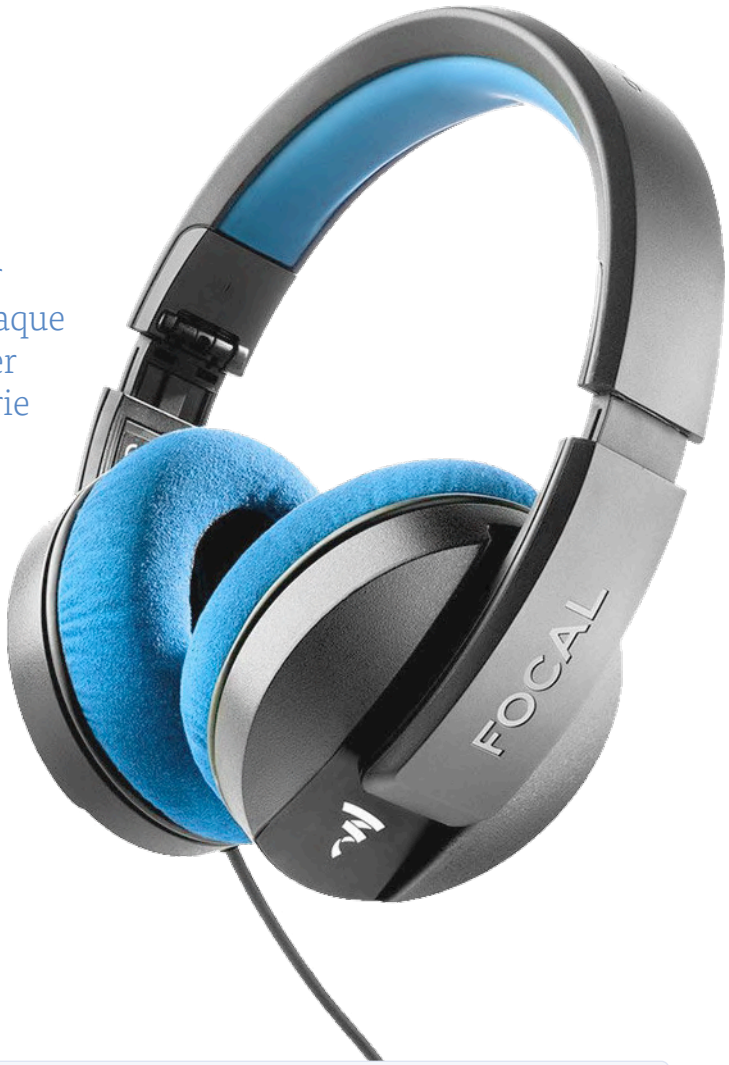
200381-03

LIEN

[1] **téléchargement du programme traffic.py** : www.elektormagazine.fr/200381-03

minuterie
pour ampli de casque ou autres charges

James Glyn



Après avoir construit un ampli audio à tubes pour casque alimenté par batterie, j'ai eu à déplorer, chaque fois que j'oubliais d'éteindre l'appareil, de retrouver la batterie déchargée. D'où l'intérêt d'une minuterie pour éteindre l'ampli automatiquement après un certain temps. En analysant mon utilisation habituelle, il m'est apparu que je ne gardais jamais le casque plus de 90 mn.

Pour la minuterie, pas question d'oscillateur en raison du risque d'interférences audibles et de la consommation d'énergie. Au lieu de cela, un réseau de résistances-condensateurs (RC), associé à un trigger de Schmitt et à un MOSFET, permet de réaliser la temporisation requise avec très peu d'énergie.

Le circuit **(fig. 1)** est construit autour d'un quadruple opérateur logique NON-ET (NAND) 4093 avec entrées à trigger de Schmitt (IC1). Celui-ci accepte jusqu'à 15 V de tension d'alimentation. Une paire d'opérateurs logiques (IC1a/b) est utilisée pour attaquer la grille d'un MOSFET de puissance IRL1404 (Q1) dans la branche haute duquel l'ampli pour casque est connecté comme charge (CN2). À l'entrée des opérateurs NON-ET se trouvent C1 et R3, calculés pour que mon ampli reste allumé pendant 100 mn environ avec une batterie de 12 V. L'inverseur est du type ON-OFF-ON SPDT à contacts fugitifs (SW1), préférable à deux poussoirs ordinaires susceptibles de court-circuiter la batterie s'ils sont actionnés simultanément.

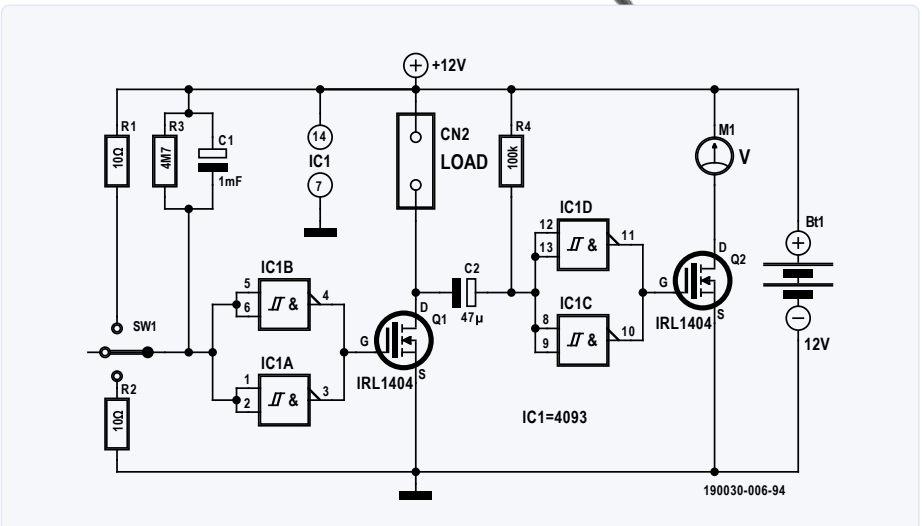


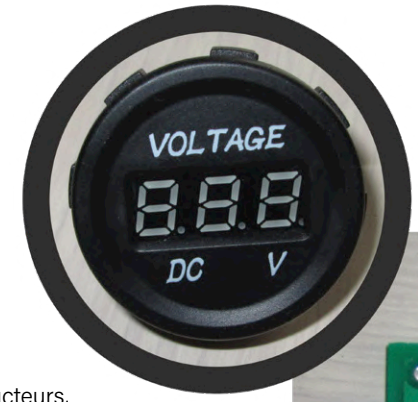
Figure 1. Lorsqu'elle est alimentée par une batterie de 12 V, la durée de fonctionnement de la minuterie est d'environ 100 mn. Une deuxième minuterie affiche la tension de la batterie pendant environ 3 s.

En appuyant brièvement dans le sens «marche» (vers R2), on charge C1 à travers R2 et on allume ainsi l'ampli. Environ 100 mn plus tard C1 finit par se décharger à travers R3, de sorte que Q1 se bloque et l'ampli est éteint. Une pression brève sur SW1 dans la direction opposée (vers R1) décharge rapidement C1 à travers R1, c'est l'extinction manuelle de l'ampli.

Avec les opérateurs NON-ET inutilisés (IC1c/d), j'ai fait un indicateur de l'état de la batterie pendant 3 s environ au moment d'allumer l'ampli. La temporisation est assurée par C2 et R4. Un autre MOSFET IRL1404 (Q2) est utilisé pour alimenter l'indicateur de tension proprement dit. Lorsque Q1 est conducteur, C2 se charge à travers R4, ce qui permet au voltmètre d'être alimenté par Q2. Dès que C2 est chargé, le voltmètre s'éteint. C2 se décharge à travers la charge (mon ampli) ou R4 si la charge est éteinte.

Le mini voltmètre (**fig. 2**) est un modèle à sept segments à LED, facile à trouver en ligne et bon marché. J'en ai testé deux dont la consommation variait entre 6 et 16 mA. L'impact de l'affichage bref de la tension est faible et n'affecte pas la charge de la batterie. Un modèle circulaire est plus facile à monter qu'un modèle rectangulaire.

Le MOSFET choisi répondra aux besoins en courant de toutes sortes de batteries puisque, selon sa fiche technique, il supporte jusqu'à 100 A. Sa tension grille-source peut atteindre 20 V, la tension de seuil de grille $VGS_{(th)}$ étant comprise entre 1,0 et 3,0 V. Grâce au trigger de Schmitt des opérateurs NON-ET, les deux MOSFET sont soit complètement conducteurs, soit bloqués, ce qui garantit une consommation de courant minimale. Le multimètre dont je dispose mesure des intensités aussi faibles que 100 nA. Comme je n'ai rien pu mesurer, la consommation au repos de cette minuterie est inférieure encore plus faible.



La résistance drain-source des MOSFET est inférieure à 6 mΩ, l'impact de la minuterie sur la tension d'alimentation de l'ampli est faible et la mesure du voltmètre correspond donc à la tension réelle de la batterie. Les tests ont montré l'absence d'échauffement des MOSFET jusqu'à 800 mA. Pas besoin de radiateur ! Si la minuterie est utilisée avec d'autres charges, il faudra vérifier l'intensité du courant et éventuellement rajouter un radiateur. Ici je n'ai pas prévu non plus de diode de roue libre, mais, en présence d'une charge inductive, il faudrait monter entre le drain et la source de Q1 une diode 1N4007 en polarisation inverse afin de protéger le transistor contre les courants inverses au moment de commuter la charge inductive.

190030-03

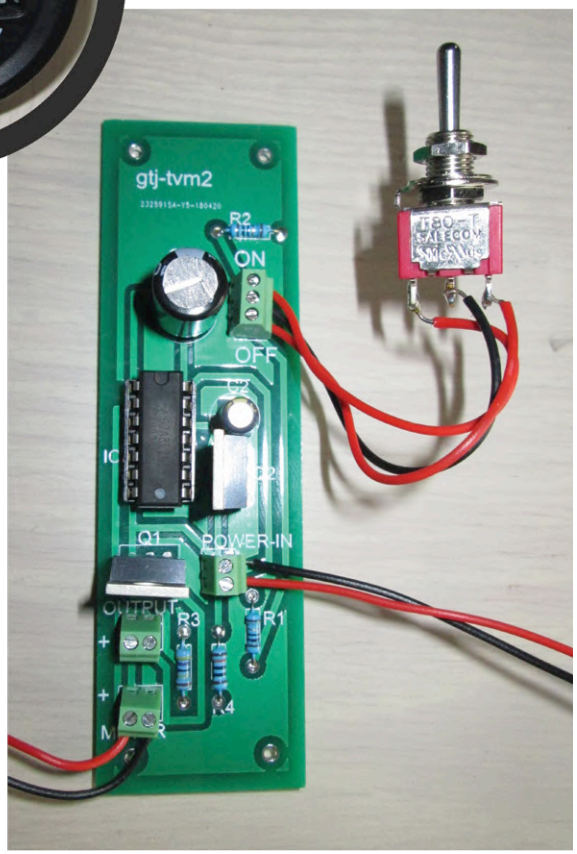
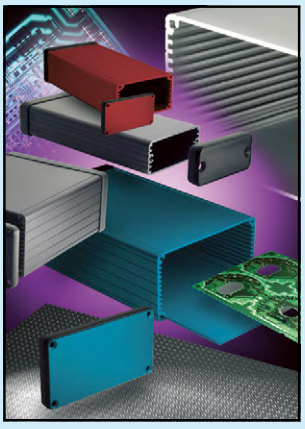
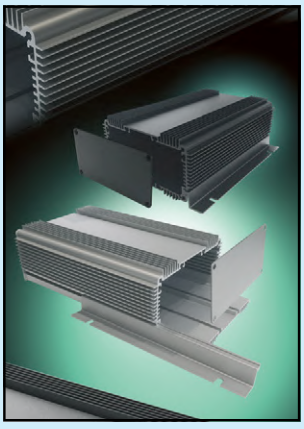


Figure 2. Le circuit assemblé, avec l'inverseur à contact fugitif ON-OFF-ON et le voltmètre.





station météo en réseau ouvert V.2

2ème partie : le logiciel

Luc Lemmens, Mathias Claussen (Elektor Labs)

Dans le numéro de mai/juin 2020 [1], le labo d'Elektor présentait sa nouvelle station météo en réseau, à code source ouvert. Le système se compose d'un kit mécanique (anémomètre et pluviomètre, coffret) disponible dans la boutique en ligne Elektor ainsi que d'une électronique autour d'un ESP32, pour envoyer les données collectées à des services dématérialisés comme *openSenseMap* et *ThingSpeak*. Ce matériel n'est qu'une partie du projet. Ici vous apprendrez comment interagissent les blocs fonctionnels du logiciel modulaire. Grâce à un cartographe intelligent, les données des capteurs peuvent être reliées de manière flexible aux canaux de communication. Le serveur web, qui joue un rôle central dans le projet, est également examiné en détail.

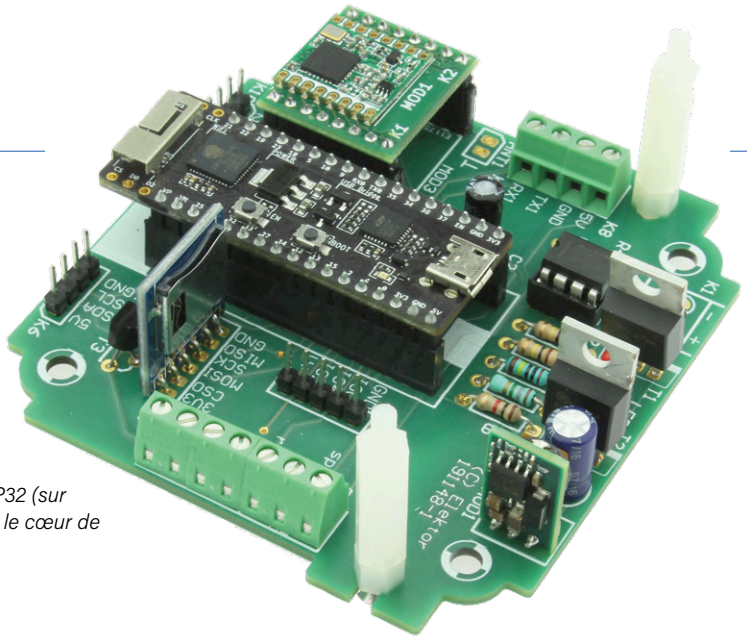


Figure 1. Le µC ESP32 (sur le module noir) est le cœur de la station météo.

Au labo d'Elektor, nous apprécions les suggestions de nos lecteurs pour améliorer ou enrichir nos projets. D'ailleurs, dès le début, nous prenons soin de programmer les logiciels de manière à pouvoir réaliser rapidement les extensions ultérieures. Le matériel de notre station météo [1] peut aussi être facilement étendu. Avec sa matrice E/S, l'ESP32 (**fig. 1**) offre une grande souplesse d'accès sur presque toutes les broches aux fonctions requises, qu'il s'agisse de SPI, I²C ou simplement d'une entrée numérique pour un poussoir.

Voyons les modules inclus dans notre logiciel et ce qui lui confère sa flexibilité, en commençant par les fonctions mises en œuvre par le logiciel [2][3].

Fonctions de base

Capteurs

Actuellement, les trois paramètres de base (précipitations, direction et vitesse du vent) sont mesurés par des broches GPIO. En outre, un bus I²C commande et lit entre autres le BME280 (température, humidité et pression atmosphérique). Les capteurs suivants sont pris en charge par le logiciel :

- VEML6070 (UVA)
- VEML6075 (UVA/UVB)
- TSL2561 (lumière)
- TSL2591 (lumière)
- WSEN-PADS (pression atmosphérique)

Pour la mesure de concentration de particules fines, le logiciel peut s'adresser par l'UART au SDS011 ou au Honeywell HPMA115SXXX.

D'autres capteurs peuvent être connectés, à condition d'adapter le micrologiciel.

Au démarrage, le microprogramme cherche les capteurs connectés et tente de sélectionner automatiquement le pilote approprié.

Services dans le nuage

Les services dématérialisés permettent de stocker périodiquement les données de la station météo, à condition d'être connecté. Les services supportés ici sont *ThingSpeak* et *openSenseMap*, ainsi qu'une connexion à un courtier MQTT pour le nuage local. Si vous utilisez un courtier MQTT comme *Mosquitto* sur *Raspberry Pi*, *Node-RED* [4] permet de traiter et de stocker les données.

Pour chaque service, on peut définir individuellement la périodicité du téléchargement des données, et spécifier quelles données transférer à quel service.

Carte SD, afficheur et bouton

Pour permettre le stockage de données hors connexion WiFi, la station météo est dotée d'un logement pour carte SD, qui accepte des cartes de plus de 4 Go. Les relevés de tous les capteurs sont stockés sur la carte SD au format CSV à des intervalles définis par l'utilisateur. Les données sont écrites directement dans le répertoire principal où un fichier nouveau est créé chaque jour.

L'afficheur de la station est utilisé pour des messages d'état (**fig. 2**). Après le démarrage, l'état du WiFi est affiché ici et un message indique si la carte SD est utilisée par le logiciel ou si elle peut être retirée en toute sécurité. Lorsque l'afficheur est actif, une pression sur un bouton permet d'éjecter ou de monter la carte SD. Si vous n'appuyez pas sur le bouton pendant 30 s, l'afficheur s'éteint pour ne pas gaspiller d'énergie, mais se rallume dès que vous appuyez à nouveau sur le bouton. Outre l'état de la carte SD, l'afficheur signale une connexion à un réseau WLAN existe et, si oui, avec quelle intensité de signal. L'adresse IP de la station est également indiquée.

Ce bouton est connecté aux mêmes broches que le bouton de démarrage de l'ESP32. Après le démarrage, il est mis à la disposition du logiciel utilisateur (voir ci-dessous).

Serveur web et site web

Comme de nombreux projets Elektor, celui-ci fait appel à un serveur web avec options de configuration. Voyez par exemple (**fig. 3**) le réglage de l'heure et de la date. Pour

Figure 2. L'affichage des messages d'état. NOSD indique l'absence de carte de mémoire.

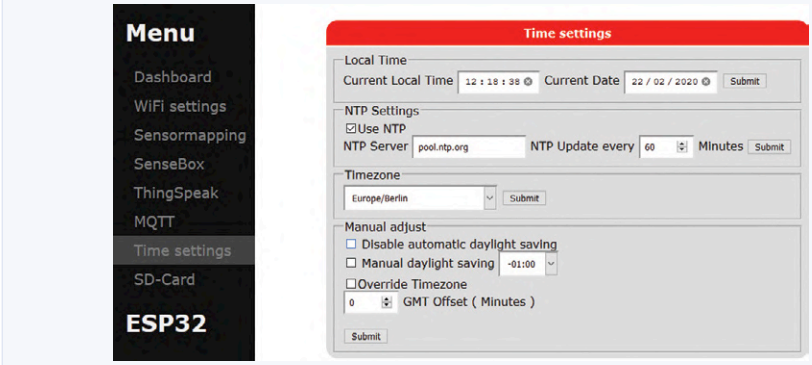


Figure 3. Le serveur web est central dans le logiciel. Il fournit les pages web pour la configuration de la station météo.

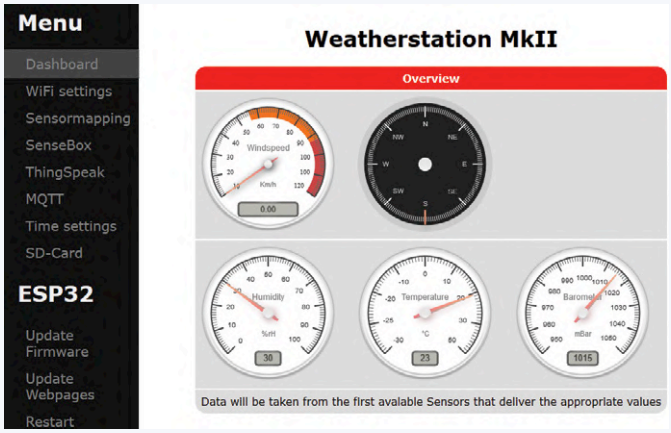


Figure 4. Le serveur web fournit également les principales valeurs de mesure, affichées ensuite dans un navigateur (p.ex. sur un téléphone) sous la forme d'un élégant graphique.

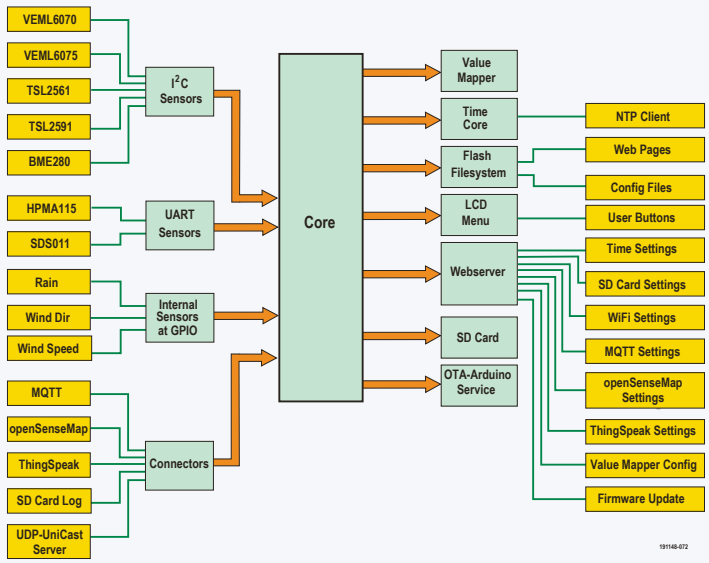


Figure 5. Aperçu des modules logiciels.

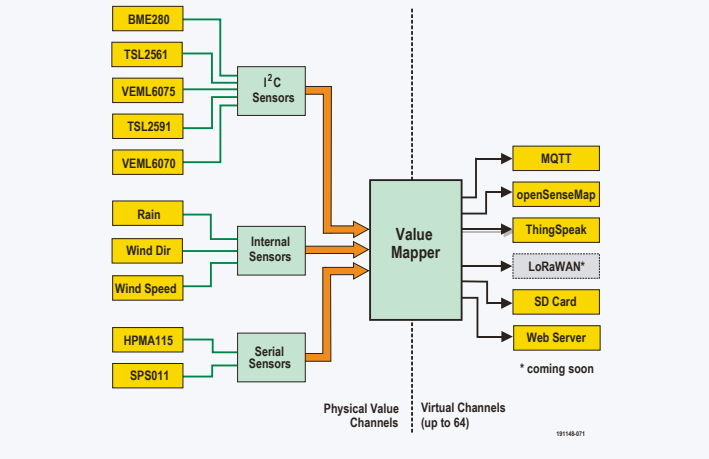


Figure 6. Le cartographe relie les sources de données (capteurs) aux puits de données. Les connecteurs vers les plateformes dématérialisées et au serveur web pour l'affichage). Il fournit 64 canaux de mesure virtuels dans le système.

afficher les données dans un navigateur, il faut plusieurs éléments emboîtés. Un serveur web ESP32 ne fournit pas seulement des pages web affichables dans un navigateur, il envoie également des données à la demande au navigateur. JSON est utilisé pour faciliter l'analyse des données par le navigateur. Les pages web ne se composent pas seulement de HTML, mais aussi de plusieurs lignes de JavaScript.

Cela permet de transférer partiellement la puissance de calcul nécessaire à l'affichage de ESP32 vers le navigateur. Comme le tableau de bord (**fig. 4**), qui permet d'afficher les valeurs mesurées de la station météo dans tel navigateur de votre choix (p. ex. sur le téléphone). Les graphiques ne sont pas transmis, mais dessinés dans le navigateur avec Javascript.

Dans le microprogramme du ESP32

Toutes les fonctions mentionnées nécessitent un logiciel. Voici les modules logiciels (**fig. 5**). Tout d'abord, nous nous penchons sur le traitement des valeurs mesurées. Nous avons ici trois parties principales : le chauffeur ou pilote, pour nous adresser aux capteurs, le connecteur, pour envoyer les données de la station aux services dématérialisés et les écrire sur la carte SD, et enfin, comme médiateur, un cartographe.

Mapper

Le cartographe fournit 64 canaux de mesure virtuels dans le système, qui peuvent être interrogés par les connecteurs. Un canal virtuel dans le cartographe peut être attribué à la valeur mesurée d'un capteur ; notez que les capteurs produisent parfois plusieurs valeurs mesurées (p. ex. le BME280 délivre trois valeurs : température, humidité et pression atmosphérique). Si l'un des connecteurs veut avoir la valeur pour le canal virtuel 0, le cartographe effectue une translation selon une table interne et obtient la valeur correspondante par le pilote de capteur approprié. Ces relations sont illustrées par la **figure 6.**

C'est un peu compliqué au début, mais l'avantage est que le connecteur lui-même n'a pas besoin de savoir comment s'adresser au capteur. C'est le boulot du cartographe. Si un autre bus ou un autre type de capteur est ajouté plus tard, seuls le cartographe et les conducteurs doivent être adaptés, les connecteurs eux-mêmes continuent de fonctionner comme avant. Belle équipe !

Le cartographe est bien sûr également connecté au serveur web. Il ne se contente

PROGRAMMATION DE TÂCHES

Dans le logiciel, les **connecteurs** sont des tâches distinctes, de sorte qu'ils peuvent lire et traiter les données des capteurs indépendamment les uns des autres. Cette solution engendre des risques à prévenir à résoudre, à savoir un accès éventuellement simultané au cartographe et donc aux capteurs sur le bus. appelés Appelons Tom et Jerry deux connecteurs actifs, cherchant chacun des données dans le cartographe toutes les minutes et accédant ainsi aux capteurs. Tom serait le premier à lire les données. Il devra peut-être attendre que les données à jour soient disponibles ou converties, et renoncera donc à son temps de calcul par l'unité centrale pour le céder à une autre tâche active. Supposons que ce soit Jerry. Celui-ci doit à son tour chercher des données dans le cartographe et donc dans les mêmes capteurs connectés à un bus. Si c'est le bus I²C, il est possible que Tom vienne de commencer la conversion des données d'un capteur et attende que le capteur soit prêt ; si Jerry commence maintenant à lire aussi les données du bus I²C, surtout à partir du même capteur, c'est la foire !
Dans le système d'exploitation d'ESP32, FreeRTOS, plusieurs mécanismes permettent de résoudre ce problème dont le plus simple est un *mutex*. On peut considérer un mutex comme la clé d'une zone bouclée par une clé unique. Quiconque veut entrer dans cette zone a besoin d'une clé, ouvre l'accès et entre dans la zone qui se referme sur lui. Si d'autres veulent entrer, ils doivent attendre que la clé soit sortie de la zone avec son propriétaire du moment. Le suivant peut alors recevoir la clé. Avec FreeRTOS, le développeur doit s'assurer que la zone est d'abord protégée par un mutex (clé) et qu'une fois ressortie de la zone protégée, la clé (c'est-à-dire le mutex) soit remise en place. Si cela ne se produit pas – ce qui arrive plus souvent que ne le voudraient les développeurs – le système s'arrêtera parce que la clé a été égarée.
Voilà pour l'accès réglementé aux données. Reste à veiller à ce que le temps de calcul soit cédé quand une tâche est oisive. Avec un logiciel sans système d'exploitation, tout fonctionne en super-boucle et il faut des mécanismes de gestion sensée du temps, mais avec un RTOS, une tâche peut être mise en sommeil. Le système d'exploitation cherche alors d'autres tâches qui ne dorment pas et les exécute. Avec les connecteurs (par exemple celui d'*openSenseMap*), le temps entre deux intervalles d'envoi n'est pas passé dans une boucle avec des instructions `delay()`. On utilise un autre mécanisme très similaire au mutex : le sémaphore binaire. Dans FreeRTOS, un mutex et un sémaphore binaire ont, à quelques petites différences près, une fonction très similaire. Si cela vous intéresse, voyez la série d'articles FreeRTOS [5]. Vous pouvez, dans FreeRTOS, spécifier le temps d'attente pour un sémaphore ; si un sémaphore se libère pendant le temps d'attente, système d'exploitation le signale. De cette façon, un temps d'attente ou un intervalle défini peut être créé et le temps de calcul peut être cédé automatiquement.
Pourquoi utiliser un sémaphore et ne pas simplement indiquer au système d'exploitation, avec la commande `vTaskDelay()`, après quel délai continuer la tâche à exécuter ? La réponse réside dans la méthode utilisée ici pour informer la tâche au sujet de changements de configuration que l'utilisateur peut demander par l'interface web. En cas de dépassement du délai lors de l'attente du sémaphore, la modification de la configuration serait perdue. Si en revanche l'attente d'un sémaphore se produit sans erreur, le nouveau réglage est validé. Les paramètres sont alors rechargés complètement et appliqués, et une nouvelle attente commence.

pas d'interroger les valeurs mesurées des différents canaux virtuels afin d'afficher les données elles-mêmes, mais exige également des informations sur les capteurs connectés, pris en charge ou manquants.
Chaque valeur de capteur a une adresse définie dans le logiciel. Il se compose du bus, du type de valeur et de l'identifiant du capteur approprié pour l'accès. Par exemple, «*BME280.0.Pressure.I2C*» signifie : *BME280* est l'identifiant du capteur, un nom lisible pour les humains, *0 est l'*identifiant de la valeur du capteur, *Pressure* est le type de valeur mesurée. L'expression *I2C indique que* le capteur est connecté au bus I²C de l'ESP32. Le cartographe voit ainsi qu'il s'agit d'un périphérique I²C et qu'il faut rechercher le pilote approprié pour l'accès.

Chauffeur

Le chauffeur reçoit du cartographe des informations sur le type de valeur concerné et sur son identifiant. À partir de ces informations, le chauffeur sait à quel capteur s'adresser et quelle valeur extraire de ce capteur.
Pour s'assurer que cela fonctionne pour le bus I²C p. ex., le chauffeur I²C scrute le bus une fois pour tous les capteurs connus après le démarrage. Cela peut se faire par un *START* dans le bus et en vérifiant si un appareil réagit. Si un appareil répond, il est marqué comme disponible et un *STOP est* envoyé. Cette séquence est répétée pour tous les capteurs connus. À la fin, on dresse une liste de ce qui est connecté et actif dans le bus. Avec cette liste, le chauffeur décide si une erreur (*capteur inexistant*) doit être renvoyée directement en réponse à une demande d'accès à ce capteur, ou si un accès à ce capteur doit être tenté. Si la valeur a pu être lue avec succès à partir du capteur, elle est renvoyée au cartographe, sinon c'est une erreur qui est renvoyée.

Connecteurs

Un connecteur est le lien entre les valeurs mesurées et un endroit où les données doivent être stockées (*puits de données*). Les connecteurs sont conçus comme des tâches distinctes dans le micrologiciel ESP32, de sorte qu'ils peuvent lire et traiter les données des capteurs indépendamment les uns des autres. Fondamentalement, chaque connecteur dispose d'une liste de canaux virtuels qui doivent être lus et d'une logique qui garantit que les valeurs, par exemple pour *ThingSpeak*, sont préparées de manière appropriée.
En raison de la programmation des tâches, les connecteurs sont indépendants les uns des autres. Ils posent certains problèmes qui doivent être résolus comme le montre l'encadré **Programmation des tâches**.

Interface web

Nous savons comment les données des

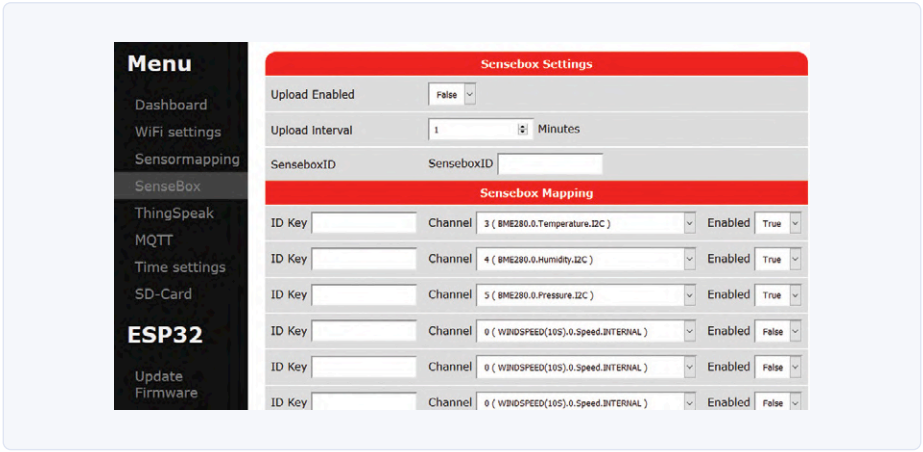


Figure 7. Configuration de l'accès à openSenseMap…

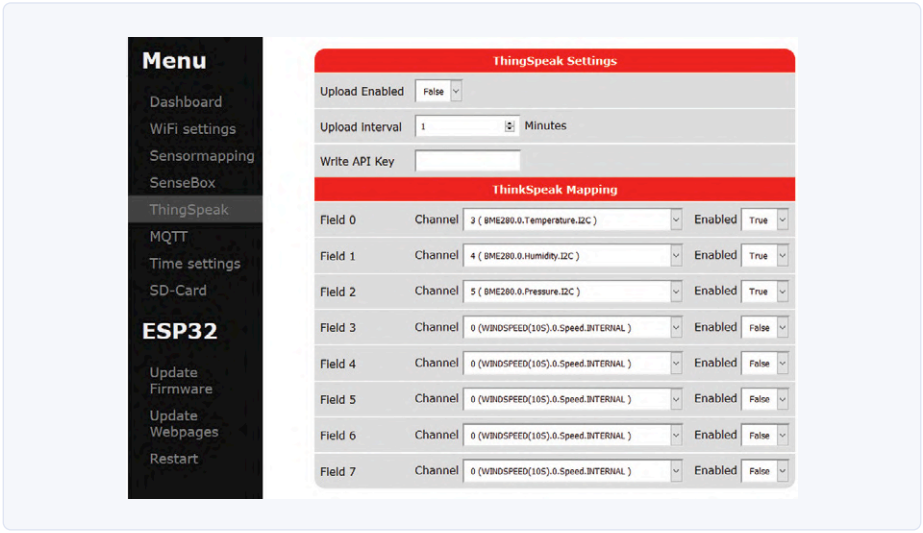


Figure 8. … à ThingSpeak et…

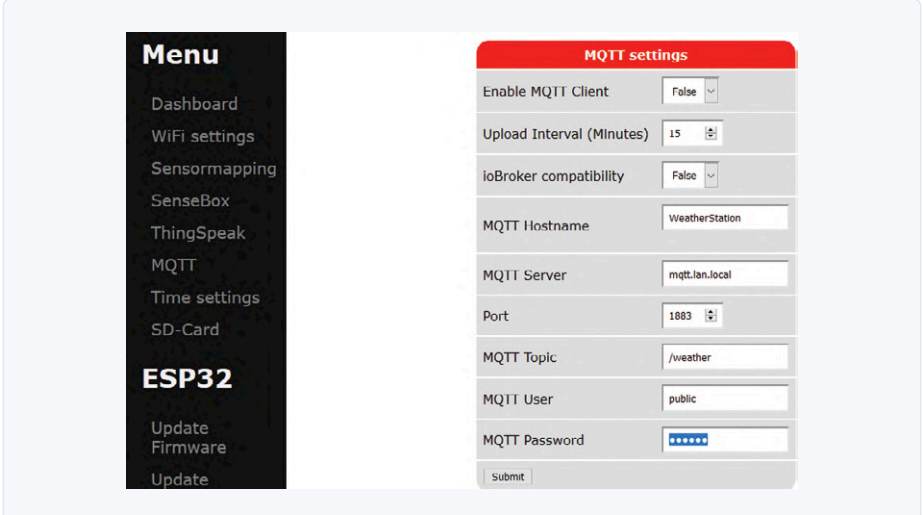


Figure 9. … au courtier MQTT. Chaque page de configuration comporte des éléments de commande différents.

capteurs sont transférées aux services dans le nuage. Nous arrivons à l'interface utilisateur de la station, divisée en plusieurs parties. Pour le cartographe et chaque connecteur, il existe une partie modulaire séparée dans l'interface web, de sorte que les nouveaux connecteurs peuvent être connectés assez facilement au serveur web existant.

Le serveur fournit les pages web et les données dont le navigateur a besoin pour restituer l'interface. Les fichiers idoines se trouvent dans une partie de la mémoire flash de l'ESP32, le SPIFFS. Si vous connaissez cette mémoire, la plupart ne l'auront utilisée dans leurs propres programmes que pour fournir des pages web par le biais d'un ESP32. Or, les SPIFFS peuvent également être utilisés pour écrire des fichiers que le serveur web peut livrer directement, comme un fichier JSON, tout comme une carte SD. Toutefois, cette méthode présente un inconvénient. Les fichiers journaux ne doivent pas être écrits dans le SPIFFS, car la mémoire flash a une durée de vie limitée et ne peut être remplacée aussi facilement qu'une carte SD.

Le microprogramme utilise une partie du SPIFFS pour une partie des fichiers de configuration, principalement pour les connecteurs. Dans les **figures 7, 8** et **9**, vous pouvez voir les pages web qui permettent à l'utilisateur de configurer les connecteurs vers les plateformes dématérialisées *ThingSpeak* et *openSenseMap* ainsi que le client MQTT. Ces pages ont des éléments de contrôle différents. Les pages doivent donc être produites de façon dynamique en fonction du connecteur. Nous avons déjà utilisé un tel mécanisme au labo d'Elektor pour d'autres serveurs web ESP32. En fonction de l'URL que le navigateur envoie au serveur web, celui-ci remplit sa propre fonction. Il est utilisé pour transférer les configurations et les paramètres du système au serveur.

JavaScript

Ce n'est pas seulement le serveur web qui est impliqué dans l'interface web, mais aussi le navigateur. En déplaçant la charge de travail du serveur vers le navigateur, on soulage l'ESP32. Notre site web en fait usage, p. ex. en ne transférant pas les cercles du tableau de bord sous forme de graphiques. Au moment de l'exécution, ils sont dessinés dans le navigateur par des scripts (JavaScript). De même, la préparation des données et des tableaux ne se fait plus dans l'ESP32, mais directement par les scripts de la page web. Pour que cela soit possible, ESP32 doit créer un moyen pour la page web d'accéder aux

ESP32 ET JAVASCRIPT

Par défaut, les navigateurs modernes chargent plusieurs fichiers à la fois. Selon le navigateur, cela peut être 4, 6, 8 ou davantage. Pour éviter à l'utilisateur d'avoir à attendre la fin du chargement de tous les fichiers, le navigateur commence à exécuter des scripts, p. ex. pour afficher les valeurs, les tableaux et les éléments d'affichage de la page. Si les scripts sont distribués dans plusieurs fichiers par souci de clarté du code, il est possible que des parties du code ne soient pas encore chargées et que l'exécution se plante. Un script a donc regroupé tout le code dans un seul fichier pour éviter le problème du morcellement. Ce problème ne s'est manifesté que sur l'ESP32, alors que le serveur web du système de développement était lui capable de fournir les données assez rapidement.
Un autre obstacle à négocier a été, dans JavaScript, le chargement de fichiers multiples effectué lors de l'exécution du script. Leur chargement (ou non) est signalé par un rappel (*Callback*). Si plusieurs fichiers sont nécessaires au même endroit, le chargement devra être enchaîné, c'est-à-dire que le fichier 1 devra être chargé et qu'un rappel pour le fichier 2 devra être donné, dans lequel le fichier 3 sera chargé. Pour contourner ce problème, la tâche a été automatisée à l'aide d'un tableau d'URL et une mémoire pour les réponses. Dans le fichier *basic.js,* vous pouvez voir comment fonctionne la fonction `loadMultipleData.`

Ne sous-estimez pas l'interaction entre interface web, serveur web et microprogramme dans vos propres projets. Des obstacles surgissent non seulement dans le microprogramme, mais aussi dans les navigateurs qui doivent fonctionner sur des appareils différents. Chaque navigateur a ses particularités et il faut souvent des scripts adaptés pour afficher la page correctement.

données dans le navigateur. C'est là qu'entrent en jeu les fichiers crées dynamiquement. Voici ceux qui peuvent être demandés par le navigateur web :

/setWiFiSettings
/getSSIDList
/mapping/mappingdata.json
/devices/supportedsensors.json
/devices/connectedsensors.json
/mapping/{}/value
/mqtt/settings
/sdlog/settings.json
/sdlog/sd/status
/sensebox/settings.json
/sensebox/mapping/{}
/sensebox/mapping.json
/thingspeak/settings.json
/thingspeak/mapping/{}
/thingspeak/mapping.json
/timesettings

Un fichier JSON est toujours livré comme réponse, de sorte que le JavaScript exécuté dans le navigateur peut l'analyser et le traiter très facilement.

L'expression {} est utilisée comme paramètre d'une fonction dynamique. Par exemple, */mapping/0/value* renvoie la valeur du capteur du premier canal logique, */mapping/1/value* la valeur du second canal logique. */sensebox/mapping/0* renvoie le premier canal logique dont la valeur est transférée à *openSenseMap*. Un coup d'œil sur le code source montre comment l'ensemble a été mis en œuvre. Tout d'abord la commande qui attribue la fonction dans le serveur web :

```
server->on(«/sensebox/mapping/{}», HTTP_GET, GetSenseboxChMapping ) ;
```

Cela indique au serveur web que lors de l'accès à *sensebox/mapping/* la dernière partie de l'URL doit être considérée comme un paramètre et la fonction `GetSenseboxChMapping` doit être appelée. Dans la fonction elle-même, le paramètre est ensuite traité comme une chaîne de caractères avec `String IDs = server->pathArg(0) ;.`

Un problème possible, né de la combinaison de l'ESP32 et de JavaScript, est signalé dans l'encadré **ESP32 et JavaScript**.

Démarrage du système

Un LCD pour l'affichage des données est parfois très utile, et il faut au moins un bouton pour le faire fonctionner. On s'efforce bien sûr de réduire le nombre de trous dans le boîtier d'une station météo. Voici donc un petit aperçu de l'affichage et de la fonction du bouton de démarrage. En appuyant sur la broche de démarrage, l'ESP32 peut être mis en mode *bootloader* au démarrage, après quoi il est disponible pour le logiciel.
Lançons le micrologiciel et appuyons sur le bouton, la station passe en mode AP (une note s'affiche à l'écran). Un nouveau réseau WLAN appelé *ESP32-xx-xx-xx* apparaît, qui peut être connecté à un appareil. Si la connexion est réussie, la page de configuration de la station météo est accessible à l'URL *http://192.168.4.1.* La page pourra vous paraître familière, elle a déjà été utilisée dans d'autres projets. Une fois saisis le réseau WiFi de votre routeur et les données d'accès, l'ESP32 redémarre et doit se connecter au réseau WiFi. Afin d'évaluer la qualité de la connexion, la valeur RSSI est également fournie.
L'afficheur est commandé avec la même touche. Pour économiser l'énergie, l'écran est éteint au bout de 30 s. Pour le réactiver, appuyez une fois brièvement sur le bouton : le rétro-éclairage se rallume et les informations d'état s'affichent. Une fois l'écran réactivé, une nouvelle pression sur le bouton permet, selon l'état affiché, de retirer ou d'insérer la carte SD en toute sécurité. La première ligne de l'écran affiche «_SD_» si la carte est insérée dans le système, ou «NOSD» si aucune carte n'est insérée ou détectée.
Dans le logiciel, il a fallu résoudre le problème des rebonds mécaniques du bouton. Ici, en raison de l'architecture du contrôleur ESP32 et GPIO, le bouton déclenche une interruption, la même sur le front montant et le front descendant. Cela implique qu'au cours de l'interruption, il faut déterminer s'il s'agissait d'un front montant ou descendant. La suppression du rebond se fait en mesurant le temps entre l'appui sur la touche et son relâchement ; si ce temps est de moins de 100 ms, la touche est rejetée. La signalisation d'une frappe est assurée par un sémaphore à partir de l'interruption, de sorte qu'une tâche distincte peut l'interroger.
L'affichage LCD est réalisé de manière similaire. Une tâche distincte met à jour l'affichage toutes les secondes pendant 30 s, puis s'endort et attend une nouvelle frappe.

Paramètres

Une fois le micrologiciel installé, la station météo créera une configuration de base

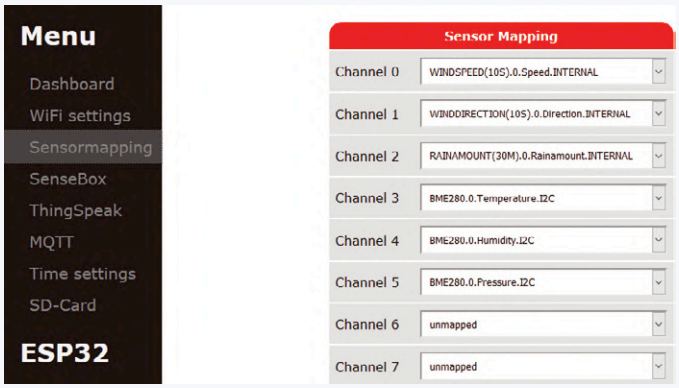


Figure 10. Affectation des valeurs des capteurs aux canaux virtuels.

basée sur un pluviomètre, un anémomètre et un BME280 connectés. Si aucun BME280 n'est connecté, ce n'est pas un problème, les canaux correspondants ne fourniront pas de valeur.
Lors de la configuration, les canaux virtuels doivent être réglés en premier, à raison d'une valeur de capteur affectée à un canal virtuel (**fig. 10**).
Ces canaux sont ensuite interrogés par les connecteurs, qui établissent la connexion avec des puits de données comme *openSenseMap*, *ThingSpeak* ou la carte SD.
Les paramètres eux-mêmes affectent également le tableau de bord. Le tableau de bord utilise les valeurs de la direction et de la vitesse du vent, de la température, de l'humidité et de la pression atmosphérique des canaux virtuels.

Les connecteurs ne sont pas seulement le lien entre station météo et services dématérialisés. Le stockage sur la carte SD est également commandé par un connecteur. Pour la carte SD elle-même, les paramètres sont assez simples : l'intervalle dans lequel les données doivent être écrites et si l'écriture doit être active. Sur le site web, la carte SD peut également être retirée du système (ou montée) en toute sécurité. La capacité et l'espace utilisé de la carte SD sont indiqués à titre d'information. Les données elles-mêmes sont stockées sous forme de fichier CSV dans le répertoire racine de la carte SD, dans un fichier journalier séparé. Pour que l'horodatage des données soit correct, l'heure doit être définie dans les *Paramètres* (fig. 2).

Pour les services dématérialisés, d'autres réglages sont possibles et nécessaires. Pour *SenseBox* et donc *openSenseMap*, l'*ID de la SenseBox et* pour chaque capteur, une clé séparée sont nécessaires (fig. 6). Ces informations peuvent être facilement extraites de l'interface web d'*openSenseMap*. Un canal virtuel peut alors être attribué à chacune de

PRODUITS

- **Circuit imprimé principal, nu**
www.elektor.fr/191148-1
- **Circuit imprimé des connecteurs, nu**
www.elektor.fr/191148-2
- **Circuit imprimé du capteur de particules, nu**
www.elektor.fr/191148-3
- **2×16-Zeichen-LCD mit I2C-Erweiterung:**
www.elektor.de/2x16-character-lcd-blue-white-120061-77
- **Afficheur LCD bleu-blanc 2 x 16 avec I²C (120061-77)**
www.elektor.fr/2x16-character-lcd-blue-white-120061-77
- **Mini module pour carte micro-SD**
www.elektor.fr/19251
- **BoB BME280 , version I²C (160109-91):**
www.elektor.fr/bme280-mouser-intel-i2c-version-160109-91
- **Station météo pro WH-SP-WS02**
(kit avec anémomètre, capteur de pluie, thermomètre hygromètre et support)
www.elektor.fr/professional-outdoor-weather-station-wh-sp-ws02

LIENS

[1] **station météo en réseau ouvert V.2 – 1e partie**: www.elektormagazine.fr/magazine/elektor-148/58640
[2] **page du projet sur Elektor Labs** : www.elektormagazine.com/labs/remake-elektor-weather-station
[3] **la page en ligne de cet article** : www.elektormagazine.fr/191148-B-03
[4] **prise en main de Node-RED** : www.elektormagazine.fr/articles/prise-en-main-de-nodered
[5] **multitâche en pratique avec l'ESP32 (1)** : www.elektormagazine.fr/magazine/elektor-145/57063
[6] **multitâche en pratique avec l'ESP32 (2)** : www.elektormagazine.fr/magazine/elektor-142/57175
[7] **multitâche en pratique avec l'ESP32 (3)** : www.elektormagazine.fr/magazine/elektor-148/58643

ces clés de capteur. Au total, jusqu'à 16 valeurs peuvent être transmises à *openSenseMap*. Pour *ThingSpeak* (fig. 7), les paramètres sont similaires, mais ici, une seule *clé API* est nécessaire pour l'écriture et les champs à transmettre doivent être réaffectés aux canaux virtuels.

Le MQTT offre ici beaucoup moins de paramètres ; cependant, l'utilisateur a besoin de plus d'informations pour traiter les données (fig. 8). Outre l'intervalle auquel les données sont envoyées (et la décision d'envoyer les données), il y a aussi la question de la compatibilité *ioBroker*. Si vous traitez les données, par exemple avec *Node-RED*, vous n'avez pas besoin d'utiliser ce mode. Dans ce cas, les données sont transportées via MQTT afin que le courtier puisse les en extraire. On part du principe d'un transport générique via MQTT, au bout duquel se trouve un système tel que Node-RED [4], capable de traiter une chaîne JSON,

```
// the JSON produced by the
   Station looks like this:
{
 Data:[
    ,
     …


       ]
}
```

Dans le tableau, ne sont transmis que les canaux de `Data` également affectés à une valeur de capteur. Les chaînes sans affectation n'apparaissent pas.

Valeurs moyennes

En déterminant les trois valeurs mesurées de la quantité de pluie, de la direction et de la vitesse du vent, des valeurs moyennes sont formées ; pour la direction et la vitesse du vent sur 10 s, 60 s et 1 h. Pour la quantité de pluie, sont émises les valeurs pour 30 mn, 60 mn, 720 mn et un jour (1440 mn). Ces valeurs sont des moyennes mobiles sur le temps passé. Cette fonction est actuellement encore stockée dans les pilotes des valeurs, mais pourrait être déplacée ultérieurement par le noyau lui-même vers un «enregistreur», qui reçoit les données comme un connecteur et les rend disponibles dans des canaux virtuels. D'autres fonctions mathématiques pourraient également être réalisées avec ce principe.

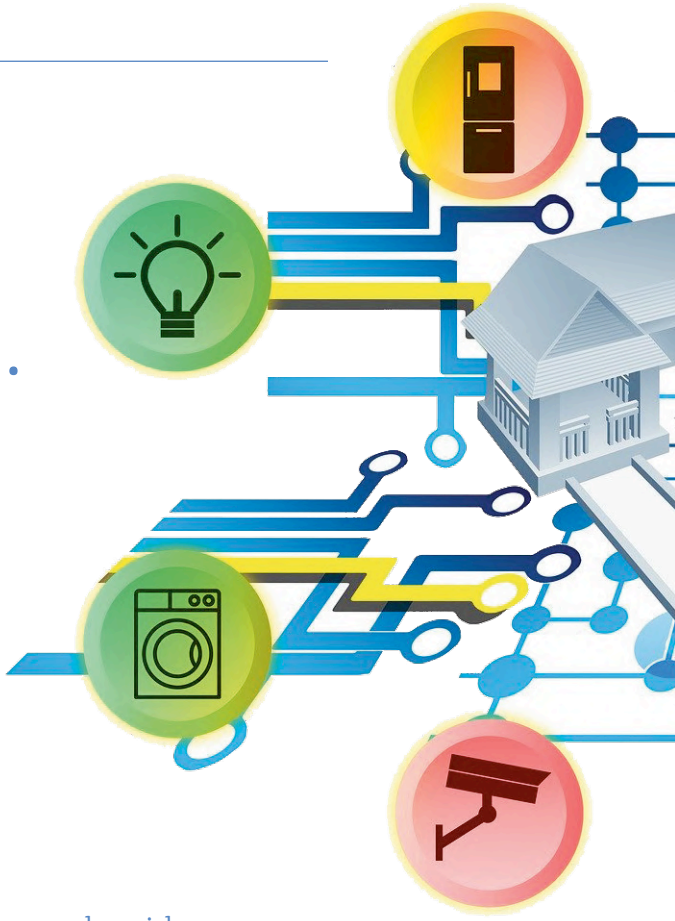
Possibilité d'extension

Cette station météo est utilisable telle quelle, mais il serait fort surprenant qu'on ne nous propose pas d'étendre ses fonctions. Au labo, quelques idées ont déjà germé, comme le support de capteurs *onewire*. Il serait intéressant de disposer d'un navigateur de fichiers pour l'interface web afin de supprimer ou de télécharger directement les données de la carte SD. Nous attendons vos suggestions, que vous pouvez faire en laissant un commentaire sur la page Elektor Labs [2] !

191148-B-03

la domotique, c'est facile avec…

ESPHome, Home Assistant et MySensors



Clemens Valens (Elektor Labs)

Qui n'a jamais rêvé d'équiper sa maison avec des éclairages ou des rideaux télécommandés ou des volets roulants qui s'ouvrent ou se ferment automatiquement ? La domotique, voilà comment cela s'appelle !

J'ai fait un rêve

Des rêves, j'en ai fait, mais ça n'est pas allé plus loin. Dès qu'il s'agissait de passer à la réalisation, je me suis invariablement heurté à des obstacles variés et ma motivation a vite molli.

Un capteur de température sans fil sur piles pour mesurer la température du salon, c'est facile à faire, y ajouter un relais télécommandé pour commander un radiateur aussi. Mais après… il faut un contrôleur qui contrôle : *"En semaine, allumer le chauffage à 7 h le matin, mais en fin de semaine pas avant 9 h"*. Et il faut arriver à reprendre la main de telle sorte que tout occupant de la maison puisse, sans avoir suivi d'abord un cours de Python, régler la température à son goût. Tout ça, ce n'est pas encore de la domotique, c'était juste la description d'un thermostat programmable. Pour passer d'une lampe télécommandée à un véritable système domotique extensible, il en faut bien plus. Voilà pourquoi je me suis contenté de rêver.

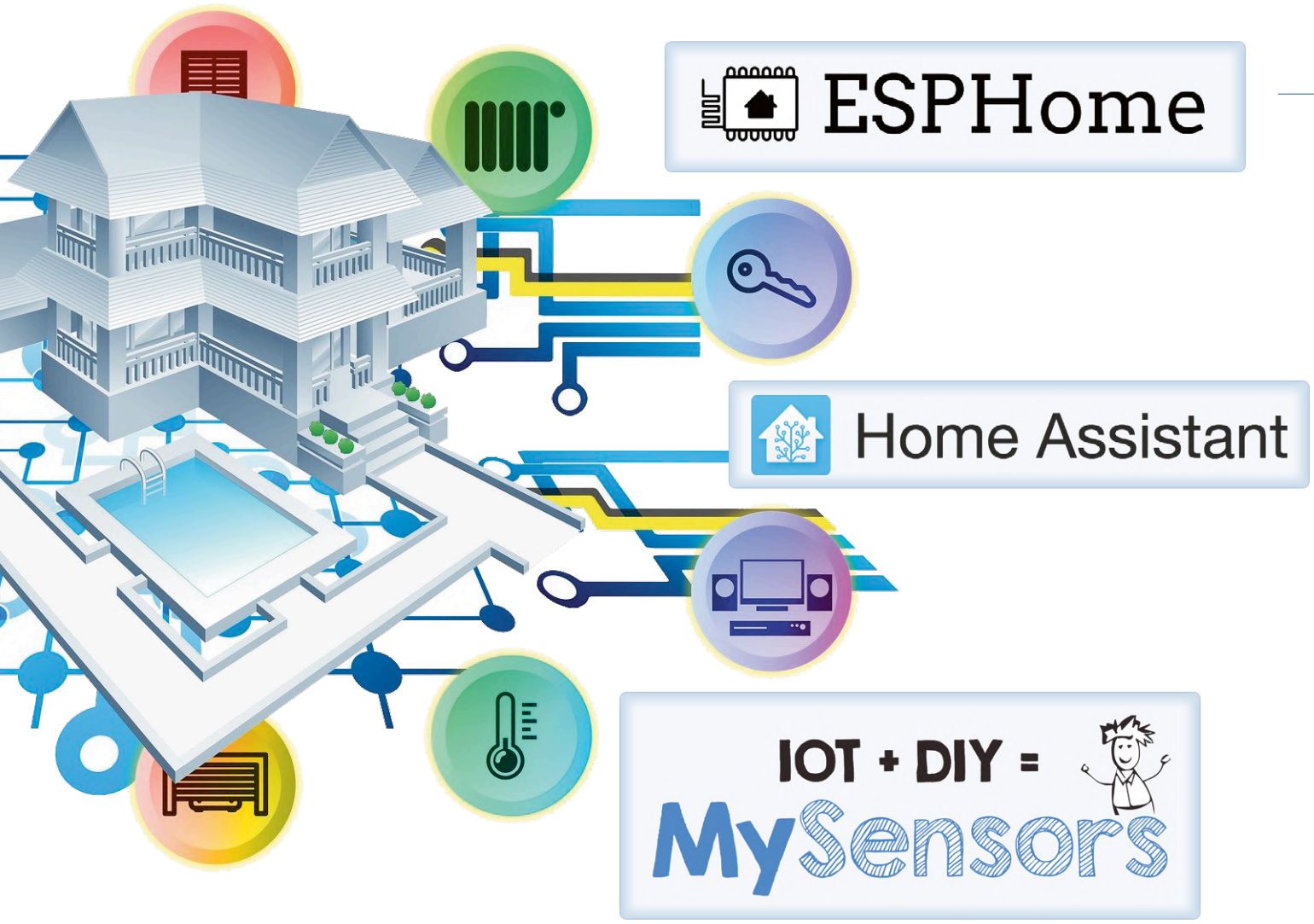
Espurna et ESPHome

Très récemment, en cherchant sur l'internet le manuel de l'utilisateur d'une prise de courant commandée par Wi-Fi achetée il y a quelques années, mais jamais utilisée, je suis tombé sur *Espurna* [1]. Même si finalement je ne m'en suis pas servi, je le mentionne, car cet excellent projet à logiciel ouvert mérite d'être connu. Avec *Espurna*, il est facile de créer des capteurs et des actionneurs Wi-Fi (surtout ESP8266) qui peuvent communiquer entre eux. *ESPHome* [2] (**fig. 1**) est un projet voisin d'*Espurna*.

Et vous trouviez Arduino facile ?

Espurna et *ESPHome* fournissent tous deux un moyen simple de programmer les puces ESP8266 et ESP32 d'*Espressif*. En fait, c'est tellement simple qu'*Arduino* vous paraîtra une jungle inextricable. En un mot, une fois qu'on a installé le logiciel, il n'y a plus qu'à écrire un fichier texte qui précise quel type de capteur est relié à quelle(s) broche(s) du module ESP et, après compilation et flashage du micrologiciel, vous disposez d'un périphérique intelligent avec une interface web, la programmation

Figure 1. La page ESPHome*, plutôt longue, mentionne tous les composants prêts à l'emploi utilisables par un appareil* ESPHome*.*



à distance (*Over-The-Air*, OTA), la communication MQTT et autres. C'est déjà impressionnant, mais ce n'est pas tout.

Commandez vos prises intelligentes sous Wi-Fi

L'internet déborde de prises intelligentes et pas chères ainsi que de cartes à relais avec interface Wi-Fi incluse ainsi que d'autres gadgets sous Wi-Fi. La plupart utilisent un ESP8266 (**fig. 2**) et ont en commun l'obligation de passer par un service dématérialisé, dans le nuage, c'est-à-dire quelque part sur l'internet. C'est de là que vient l'application de télécommande à installer sur votre portable. C'est généralement inconfortable, et personne ne s'en sert bien souvent. Avec *Espurna* et *ESPHome*, c'est différent, car ils offrent la possibilité de reprogrammer ces appareils avec un micrologiciel modifiable à volonté. Nul besoin de connexion internet, adieu les applications en anglais approximatif et les services douteux dans le nuage. Vous reprogrammez l'appareil comme bon vous semble et vous l'intégrez à votre système de domotique contrôlé par *moi@chezmoi.*

De la télécommande à l'automatisation

ESPHome et *Espurna* comportent des fonctions d'automatisation qui permettent à l'utilisateur de spécifier des règles de commutation en fonction d'événements ou de données de capteurs. Avec l'un ou l'autre, vous réaliserez facilement un système domotique élaboré à partir de composants du commerce à Wi-Fi intégré ou bien réalisés par vous-même.

Cet article se serait arrêté ici si je n'avais pas trouvé exagérément compliquée la domotique avec *Espurna* ou *ESPHome*. Mes capacités intellectuelles sont limitées, certes, mais la documentation de ces projets est médiocre. Copieuse peut-être, mais surtout disparate et plutôt obscure.

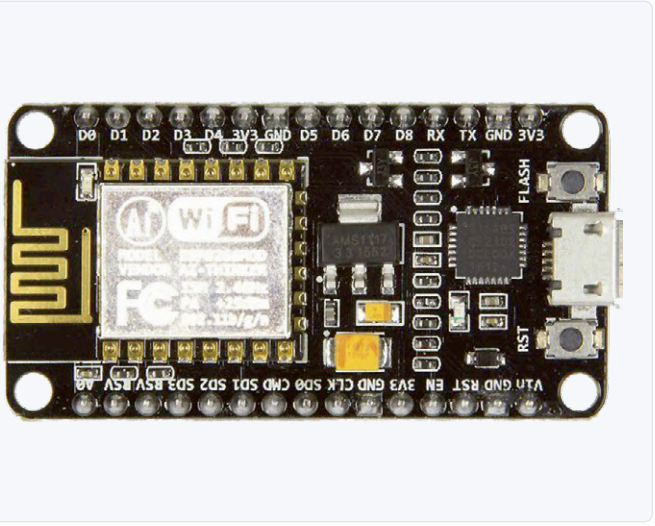


*Figure 2. **ESPHome** tourne sur les modules basés ESP8266 tels que **NodeMCU**, qui est répandu et peu coûteux. On peut aussi utiliser des modules ESP32.*

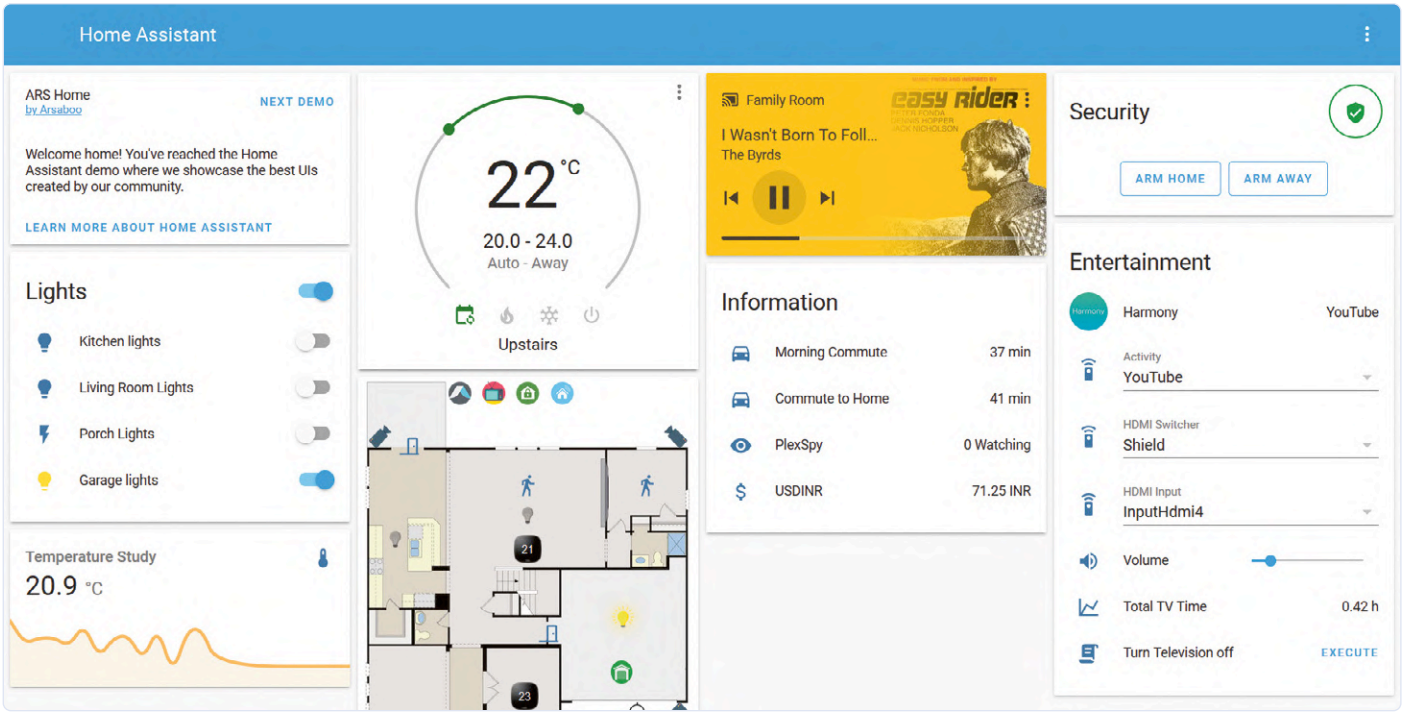


Figure 3. Le tableau de bord de Home Assistant est entièrement personnalisable. Selon votre goût, il sera touffu ou dépouillé.

L'assistant domestique

ESPHome et *Espurna* proposent tous deux une intégration à *Home Assistant* [3], ce qu'on appelle un concentrateur d'automatisation ou *contrôleur*. C'est un appareil qui permet à l'utilisateur de combiner des capteurs et des actionneurs (de fabricants différents) en un système unique. Ici, les mots *capteurs* et *actionneurs* doivent être pris dans un sens très général. Ça va du thermomètre, au GPS ou aux services internet en passant par la commande de moteurs et l'envoi de textos. *Home Assistant* (aussi 'HA', 'Hass' ou 'Hass.io') est compatible avec – pour ne citer que les plus connus - *Alexa* et *OK Google*, le *Trådfri* d'*Ikea*, le *Hue* de *Philips*, la domotique de *Z-Wave* et de *Zigbee*, et *Sonoff* de chez *iTead* [4]. Au moment où j'écris, HA en est à 1574 intégrations et va bien au-delà de ce que proposent *Espurna* et *ESPHome*.

Vous utilisez HA pour définir les règles de fonctionnement des appareils à l'intérieur et autour de votre maison. « *Si le soleil se couche dans vingt minutes et si le portable de l'occupant A est présent dans la maison et s'il n'a pas plu les trois dernières semaines, alors ouvrir le troisième gicleur à partir de la droite* ». Voilà une règle typique (et même ordinaire) de HA, en supposant que le matériel en question soit installé et fonctionne comme prévu. HA propose aussi une interface graphique utilisateur (GUI) entièrement personnalisable (**fig. 3**). HA est un logiciel ouvert, en *Python*, sur *Raspberry Pi*, mais qui s'accommode aussi d'autres systèmes d'exploitation. Si je l'ai installé sur un RPi 3, c'est parce que c'était tellement facile. Oui, mon truc, c'est la facilité.

L'intégration dans *Home Assistant*

C'est aussi pour *Home Assistant* que j'ai continué avec *ESPHome* plutôt qu'avec *Espurna*. Pour *ESPHome* il y a un greffon (**fig. 4**) qui l'intègre à HA de telle manière que la détection des appareils basés *ESPHome* soit automatique. « Flashez et jouez ! » Que demander de plus ? L'intégration est telle que vous n'avez même plus besoin d'ordinateur : du fond de votre canapé, vous (re)programmez et (re)configurez vos capteurs et vos actionneurs sur votre portable.

Internet des objets à faible consommation

La solution Wi-Fi, c'est bien pour connecter rapidement des appareils à un réseau, mais gare à la consommation. Elle n'est donc pas la meilleure pour des nœuds alimentés par de l'énergie glanée ou une pile bouton censée tenir des années. Ces appareils sont dormants la plupart du temps et, lorsqu'ils se réveillent, ils expulsent leurs données aussi vite que possible, faute d'énergie suffisante pour de verbeux échanges protocolaires.

MySensors [5] est excellent pour ce genre d'appareils. Ce projet libre de domotique et d'IdO basé sur la bande Radio ISM, en particulier le nRF24 de *Nordic Semiconductor* (**fig. 5**) et le RFM69 de *HopeRF*. On peut utiliser aussi la plateforme nRF5, plus récente, comme sur le *micro:bit* de BBC. Le site de *MySensors* est un peu fouillis, mais une fois familiarisé, vous y découvrirez des choses intéressantes.

MySensors utilise surtout (mais pas seulement) *Arduino* comme plateforme microcontrôleur et il gère lui-même un réseau arborescent (ou en étoile). Comme *ESPHome*, il s'intègre en (pas tout à fait la même) douceur à *Home Assistant*.

Le projet se présente sous la forme d'une bibliothèque *Arduino* incluse dans le gestionnaire de bibliothèques *Arduino*. Une fois installée, vous pouvez créer votre application à partir de l'un des exemples. Très souvent, il suffit de changer les numéros de broches des périphériques connectés.

Les mains dans le cambouis

Fin du préambule, passons à la pratique. Je suggère d'installer *Home Assistant* sur un *Raspberry Pi*. Un bon guide pas-à-pas est disponible [3] (onglet GET STARTED). Il est suggéré d'utiliser un RPi 4,

mais chez moi, ça marche sur un RPi 3. Une carte microSD d'au moins 32 Go est recommandée, mais il faut se rappeler que le RPi ne sait pas gérer les cartes SDXC formatées en exFAT (c'est-à-dire de plus de 32 Go), donc, si vous voulez utiliser une carte de 64 Go ou plus, n'oubliez pas de la reformater en FAT32. Pour améliorer la fiabilité du système, on peut utiliser un disque statique (SSD) au lieu d'une carte microSD fragile.

DNS multipoint

Home Assistant s'appuie sur le protocole DNS multipoint (mDNS) pour découvrir les appareils et communiquer avec eux, mais ce protocole n'est pas très bien supporté par *Android* et *Windows* (cf. encadré). Les appareils *Apple* et le RPi fonctionnent bien et je suppose que c'est le cas d'autres versions de *Linux*. C'est pour cette raison qu'il est bon d'attribuer une adresse IP statique au HA. Mon système HA utilise DHCP et j'ai observé des problèmes de connexion avec l'application HA pour *Android* quand le serveur DHCP attribuait une nouvelle adresse IP à l'ordinateur HA.
Notez que l'installateur HA démarre très tôt un serveur web qui permet de suivre la progression de l'installation ; connecter un écran au RPi n'est guère utile ici. Pour obtenir l'adresse IP de ce serveur, consultez votre routeur de réseau.

Accès au fichier de configuration

Le pas suivant de l'installation consiste à configurer *Home Assistant*. Il y a un assistant pour cela, je ne m'y attarde pas. Intéressons-nous plutôt à l'installation de quelques extensions après la configuration de HA. On l'effectue à partir de l'onglet *Add-on Store* sous le menu *Supervisor*. Pour une raison obscure, il n'est pas possible d'éditer le fichier de configuration principal de HA (*configuration.yaml*) , alors qu'on y est fréquemment obligé, en particulier quand on expérimente avec le système. C'est pourquoi j'ai installé les extensions *File Editor* (précédemment nommée *Configurator*) et *Samba share*. La première permet d'éditer des fichiers de bas niveau directement dans HA, la seconde d'accéder à certains répertoires HA sur le réseau, ce qui vous permet d'éditer des fichiers avec votre éditeur de texte favori. Pour des raisons de sécurité, vous préférerez peut-être installer l'extension *Terminal & SSH*.
Samba vous permet de transformer HA en un serveur de fichiers si vous créez un répertoire 'www' dans le répertoire 'config'. L'installation d'extensions est simple : cliquer sur la fiche de l'extension puis sur *Install*. Une fiche ouverte donne les instructions de configuration de l'extension.

Installation de l'extension *ESPHome*

Cela nous amène à *ESPHome* parce qu'il nous faut également installer l'extension *ESPHome*. Pour ce faire, commencez par ajouter le lien de son dépôt au store. Ensuite, trouvez le champ *Add new repository by URL* (= Ajoutez l'URL d'un nouveau dépôt) et collez-y l'URL suivant :
`https://github.com/esphome/hassio`
Vous pouvez maintenant installer l'extension *ESPHome*. Il y en a trois versions : ordinaire (= *plain*), beta et dev. Nous utiliserons l'ordinaire. La seule configuration à effectuer pour cette version est l'activation des options *Démarrer à l'initialisation* (= *Start on boot*), *Mise à jour automatique* (= *Auto update*) et *Afficher dans la barre latérale* (= *Show in sidebar*). Cette dernière facilite l'accès à l'extension.

Intégration de *MySensors*

Au moment où j'écris (été 2020), il n'existe pas de *Home Assistant* pour *MySensors*. Pour activer cette intégration il faut ajouter quelques lignes au fichier *configuration.yaml* de HA (voir l'encadré) :

```
mysensors:
  gateways:
  - device: '192.168.1.100'
    persistence_file: 'mysensors/Wi-Fi_gateway.json'
    tcp_port: 5003
    optimistic: false
    persistence: true
    retain: true
    version: '2.0'
```

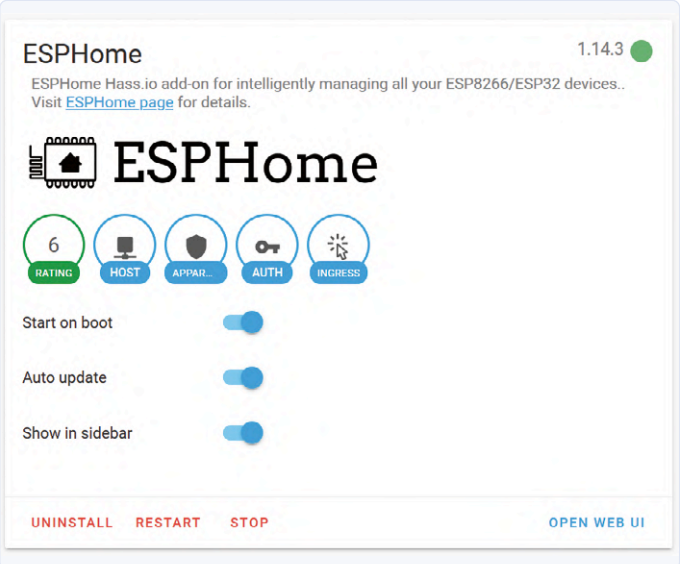


*Figure 4. J'aime bien l'option 'Afficher dans la barre latérale' activée pour l'extension **ESPHome** dans le **Home Assistant**.*

Figure 5. Les modules nRF24 se présentent sous différentes formes, dont la plus commune a 8 broches ; celle à 10 broches n'en diffère que par le brochage.

COMMENT ACTIVER **MDNS** SOUS WINDOWS

Dans une configuration de réseau DHCP (le réseau attribue des adresses IP aux nœuds connectés), les utilisateurs de *Windows* peuvent se heurter à des difficultés pour se connecter à HA parce qu'ils ne disposent pas du protocole DNS multipoint (mDNS) actif. La disponibilité de ce protocole permet de se connecter à HA en utilisant le lien http://hassio.local:8123 (sous [3] on trouve d'autres liens, mais ils n'ont pas fonctionné pour moi). Voici le moyen que j'ai trouvé de régler ce problème :

1. Ouvrez l'éditeur du registre (touches 'Windows'+R, puis tapez 'regedit') et trouvez la clé **Ordinateur\HKEY_LOCAL_MACHINE\SOFTWARE\Policies\Microsoft\Windows NT\DNSClient**
   Cliquez dessus avec le bouton droit de la souris et choisissez *New* pour créer un DWORD 32-bit nommé *EnableMulticast* et de valeur 0 (zéro).
2. Depuis le site d'*Apple*, téléchargez *iTunes*, mais ne l'installez pas. Ça demande un peu de recherche. La dernière fois où je l'ai fait, j'ai choisi *iTunes for Windows*, mais pas celui du *Microsoft Store*, mais en continuant à dérouler le menu jusqu'à tomber sur *Looking for other versions?*. Ce lien-là vous conduira au fichier désiré. Une fois le téléchargement terminé, changez le type du fichier de .EXE en .ZIP puis extrayez-en le fichier *bonjour.msi* (ou *bonjour64.msi*, selon la version d'*iTunes* téléchargée).
3. Installez *bonjour* et redémarrez votre ordinateur.

https://www.apple.com/itunes/

UN MOT À PROPOS DE L'INDENTATION DE **YAML**

Home Assistant et *ESPHome* utilisent tous deux YAML (*Yet Another Markup Language*) pour leurs fichiers de configuration. Comme *Python* et d'anciens langages assembleur, sa structure est liée à l'indentation (une horreur, si vous voulez mon avis). L'indentation doit être la même pour toutes les lignes de même niveau, mais peut différer d'un niveau à l'autre. Pour les fragments de YAML dans cet article, chaque niveau d'indentation est de deux (2) espaces.

Comme déjà mentionné, il faut utiliser une adresse IP statique pour la passerelle *MySensors* que vous allez réaliser (si, si, vous allez le faire, voyez ci-dessous) et vous devez choisir un nom et un emplacement pour le fichier JSON où HA pourra stocker l'information réseau *MySensors*.
Important : Chaque fois que le fichier *configuration.yaml* de HA est modifié, le système doit être redémarré pour prendre en compte ces modifications. Ceci peut s'effectuer à partir de l'onglet *Système* du Superviseur. Tant qu'à éditer le fichier de configuration, profitez-en pour ajouter les lignes suivantes (ça vous épargnera un redémarrage) :

```
binary_sensor:
  - platform: workday
    country: [country code]
```

Ces lignes permettent d'écrire des règles d'automatisation actives seulement les jours de semaine ou pendant les week-ends, ce genre de choses. Remplacez **[country code]** par le code du pays dans lequel travaille votre système. Pour trouver ce code (et d'autres informations utiles), reportez-vous à la page d'aide du *Workday Binary Sensor*.

Mon premier appareil *ESPHome*

Si vous avez suivi les instructions ci-dessus, vous êtes prêt à vous mettre à la programmation d'appareils à base d'ESP8266 et d'ESP32. Un nouveau système n'est pas immédiatement compatible avec *ESPHome*, il devra donc être programmé initialement par le port série. Selon l'appareil, il vous faudra peut-être installer un pilote USB-vers-Série sur l'ordinateur HA pour que ça marche. Les cartes *NodeMCU* équipées d'une puce USB CP2102 de *Silabs* (*Silicon Laboratories*) que j'ai utilisées ont fonctionné aussitôt déballées.
Après avoir connecté l'appareil à l'ordinateur hôte de HA, ouvrez le tableau de bord d'*ESPHome*, soit à partir de la barre latérale (si vous avez activé cette option) soit en cliquant sur *Open Web UI* sur la fiche des extensions dans le tableau de bord du superviseur. Vérifiez que le port série est disponible dans la liste déroulante de la fiche de l'extension dans le coin supérieur droit, *OTA (Over-The-Air)* par défaut. Si ce n'est pas le cas, redémarrez l'extension *ESPHome* (en retournant à la fiche *ESPHome* sur le panneau de contrôle du superviseur) ; là, ça devrait marcher.
Ensuite, cliquez sur le bouton rosâtre '+' sur le panneau de commande de l'*ESPHome*, ce qui ouvre un assistant qui va vous guider à travers la première partie. J'avoue n'avoir utilisé aucun mot de passe pour aucun appareil, ce qui témoigne peut-être de mon irresponsabilité. Pour le téléchargement, indiquez le port série auquel l'appareil est connecté.

Édition du fichier YAML

L'assistant a fait son travail et une fiche devrait maintenant exister pour votre appareil. À ce moment, le bouton *Edit* ne fonctionne pas encore (chez moi) ; il s'active quand on recharge la page. Cliquez dessus, vérifiez vos paramètres d'accès au réseau Wi-Fi et assurez-vous de la présence des lignes 'api:' et 'ota:'. Si vous téléchargez cette configuration vers votre appareil, il sera détecté par *Home Assistant* après son redémarrage. On peut maintenant déconnecter l'appareil de l'ordinateur, car la programmation sans fil (*Over-The-Air*) a été activée. Toutefois, l'appareil ne fera rien, car vous n'avez pas encore configuré ses périphériques. Alors, continuez votre lecture avant de télécharger une configuration.
Si vous utilisez un module *NodeMCU*, vous pouvez le démarrer en ajoutant les lignes ci-dessous à la fin du fichier de configuration (en dessous de 'ota:' juste par souci de clarté, car leur position dans le fichier n'a pas d'importance) avant de le télécharger. Cela va donner à HA l'accès à la LED embarquée et au bouton *Flash*.

```
output:
   - platform: gpio
     id: «led»
     pin:
       number: GPIO16
       inverted: True

 light:
   - platform: binary
     name: «LED»
     output: «led»

 binary_sensor:
   - platform: gpio
     name: «Flash pushbutton»
     pin:
       number: GPIO0
       inverted: True
```

Notez que dans ce fragment de code l'indentation de tous les niveaux est de deux (2) espaces, c'est-à-dire que les lignes '**number: GPIOx**' et '**inverted: True**' commencent avec six espaces (car elles sont au troisième niveau d'indentation, voir l'encadré).
Cette configuration peut aussi fonctionner avec un module ESP-01 en remplaçant GPIO16 par GPIO3 et en connectant une LED en série avec une résistance de 470 Ω (environ) entre la broche GPIO3 (RXD) et la masse. Il faut aussi connecter un bouton-poussoir entre la broche GPIO0 et la masse et une résistance de rappel de 10 kΩ (environ) entre GPIO0 et le 3,3 V. N'appuyez pas sur ce bouton au démarrage de l'appareil.
Téléchargez cette configuration vers l'appareil *ESPHome* et attendez qu'il redémarre. *Home Assistant* devrait le voir – le panneau de commande de *ESPHome* devrait indiquer *En service* et, sans intervention de votre part, créer un affichage pour le voyant et un indicateur pour le bouton-poussoir.
Si vous commencez par télécharger la configuration vide, il vous faudra sans doute le chercher dans la liste *Devices* du menu de configuration de HA.
Vous pouvez maintenant ajouter des fiches pour les commandes dans la vue d'ensemble *Overview* de HA (cliquez sur les trois points puis sur *Configure UI*) et créer des automates dans l'éditeur d'automates (*Automations*) du menu de configuration. Tout cela étant à peu près évident, je vous laisse faire sans plus d'explications ; c'est aussi une bonne occasion de farfouiller dans le *Home Assistant*. De plus, *MySensors* nous attend.

Réalisation d'une passerelle Wi-Fi *MySensors*

Pour réaliser un réseau *MySensors*, une passerelle est nécessaire. Vous en aurez aussi besoin pour vous connecter à d'autres réseaux Wi-Fi. Pour cela, j'ai pris un *NodeMCU* basé ESP8266 parce qu'il possède un port SPI dont nous allons avoir besoin. Une autre possibilité est d'utiliser un module ESP32. En fait, il y a de nombreuses possibilités, mais dans cet article, nous nous occupons de Wi-Fi. Connectez un module nRF24L01+ au port SPI (**fig. 6**). Il est recommandé d'ajouter un condensateur électrolytique (4,7 à 47 µF) en parallèle avec un condensateur céramique (de 100 nF environ) entre les broches VCC et masse du module nRF24. C'est tout ce dont vous aurez besoin.
Côté logiciel, il vous faut un ordinateur avec l'EDI *Arduino* installé. Ajoutez à l'EDI le noyau ESP8266 (ou ESP32) pour *Arduino* – vous trouverez tous les détails sous [6] – et la bibliothèque *MySensors*

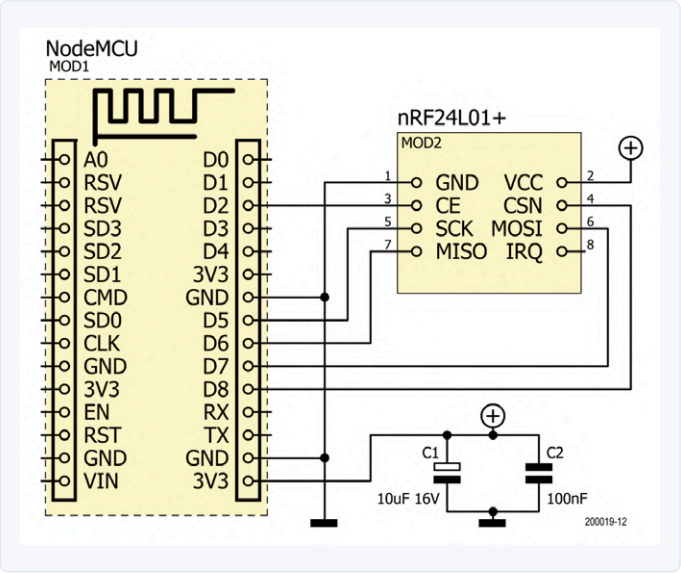


*Figure 6. Une passerelle Wi-Fi **MYSensors** peut être réalisée en utilisant un module **NodeMCU**.*

(Croquis > Inclure une bibliothèque > Gérer les bibliothèques...). Chargez l'exemple passerelle (Fichier > Exemples > MySensors) et saisissez le SSID, le mot de passe et l'adresse IP statique que vous avez indiquée plus tôt dans le fichier *configuration.yaml* de HA. Téléchargez le croquis vers votre appareil et voilà, votre passerelle est prête.
N'oubliez pas qu'une passerelle (ou des nœuds répétiteurs, non traités dans cet article) doivent toujours être alimentés et ne peuvent être mis en sommeil. Les nœuds de capteurs, par contre, peuvent faire ce qu'ils veulent.

Mon premier nœud *MySensors*

La création d'un nœud *MySensors* ressemble à la réalisation d'une passerelle, sauf que le module ESP est remplacé par une carte compatible *Arduino* équipée de capteurs et d'actionneurs. Connectez un module nRF24L01+ au port SPI de la carte et ajoutez les condensateurs mentionnés plus haut (**fig. 7**).
Téléchargez un croquis de la bibliothèque d'exemples de *MySensors*, de préférence un qui ressemble à ce que vous voulez réaliser. Notez qu'il y a d'autres exemples de croquis sur le site web de *MySensors*. Le croquis doit être édité pour se conformer à vos périphériques et numéros de broches, mais c'est à peu près tout ce qu'il y a à faire. Téléchargez le croquis vers votre appareil et laissez-le redémarrer ; il va automatiquement rejoindre le réseau MySensors. Oui, vraiment.

Attention à la version du protocole

Arrivé là, ça a coincé : Home Assistant ne reconnaissait pas mon appareil, un relais distant. J'ai fini par découvrir que c'était en rapport avec la version de l'interface de programmation (API) utilisée. En inspectant le fichier JSON de persistance (voir la section 'Intégration de *MySensors*'), j'ai remarqué l'indication que la passerelle utilisait l'API ou le protocole version 2.3.2. Ma page *MySensors* [7] présente un exemple avec une indentation spécifique à l'utilisation de l'API version 2.x. Quand je l'ai essayé, mon nœud est apparu dans la liste *Entities* du menu de configuration de HA et j'ai pu créer des fiches UI pour lui. Selon [7], pour qu'un nœud basé V2.x fonctionne sous HA, il

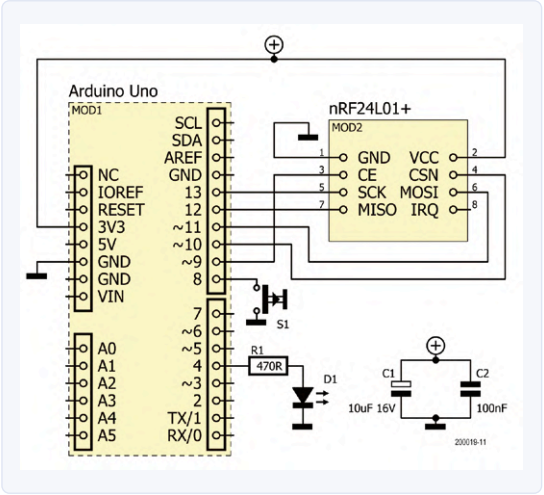


*Figure 7. Schéma d'un nœud **MySensors** avec un bouton-poussoir pour un capteur binaire et une LED pour un actionneur.*

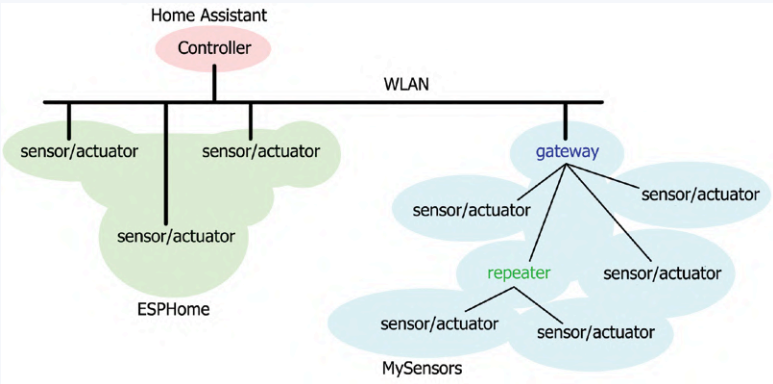


Figure 8. Si vous combinez tous les éléments décrits dans cet article, vous aurez réalisé un système domotique tel que celui-ci.

doit envoyer une valeur initiale depuis la fonction **loop()**, mais seulement pendant la phase de démarrage. Les croquis exemples ne le font pas. Mais en approfondissant, j'ai conjecturé qu'un nœud *actionneur* doit requérir de HA (et peut-être aussi lui envoyer) une valeur initiale pour être reconnu. Un nœud *capteur* est repéré aussitôt qu'il envoie des données. La demande n'a pas besoin de provenir de la fonction **loop()**, elle doit simplement être effectuée à un moment quelconque de la phase de démarrage.

Fonctions spéciales : *before()* et *presentation()*

Pour distinguer les croquis V2.x de types plus anciens, cherchez la fonction **presentation()**. Si le croquis en contient une, c'est un V2 ou plus récent. L'inverse n'est toutefois pas vrai, car un croquis récent peut ne pas la contenir. Les instructions **present** habituellement contenues dans cette fonction ne peuvent pas être omises et doivent être déplacées vers une autre fonction, par exemple **setup()**.

Les croquis *MySensors* peuvent comporter une fonction **before()**. Les deux fonctions **before()** et **presentation()** sont appelées avant **setup()** et dans cet ordre, c'est-à-dire **before(), presentation(), setup()**, et enfin **loop()**.

Adapter du code existant à votre matériel est devenu très facile. Il y a beaucoup d'exemples pour toutes sortes de capteurs dans la bibliothèque *Arduino* aussi bien que sur l'internet, prenez le temps de chercher.

J'espère vous avoir intéressé

Voici la fin de mon topo sur la domotique facile (**fig. 8**). Il est incomplet et on trouvera sans doute plus simple ou mieux. Je suis encore en phase d'apprentissage moi-même.

Il existe beaucoup de projets de domotique qui font des choses similaires, probablement aussi de meilleurs ou de plus beaux ; tous ont des points forts et des faiblesses.

Comme ils sont innombrables, le choix est difficile. J'en ai présenté ici quelques-uns que j'ai testés puis gardés pour réaliser mon système domotique. Je les utilise encore et leurs possibilités m'émerveillent par leur variété. Je n'ai pas d'ailleurs l'intention de m'en tenir là.

Sur le site Elektor Labs, vous trouverez des exemples de configurations pour *ESPHome* [8] et des composants *MySensors* [9], et quelques vidéos.

200019-02

LIENS

[1] **Espurna** : https://github.com/xoseperez/espurna

[2] **ESPHome** : https://esphome.io/

[3] **Home Assistant** : www.home-assistant.io/

[4] **Sonoff** : www.itead.cc/

[5] **MySensors** : https://www.mysensors.org/

[6] **ESP8266 core pour Arduino** : https://github.com/esp8266/Arduino

[7] **MySensors in Home Assistant** : www.home-assistant.io/integrations/mysensors

[8] **ESPHome sur le site Elektor Labs** : www.elektormagazine.com/labs/how-to-home-assistant-esphome

[9] **MySensors sur le site Elektor Labs** : www.elektormagazine.com/labs/mysensors-home-assistant-howto

le tube cathodique de stockage

Encore un drôle

Neil Gruending (Canada)

Les oscillos traditionnels à tube cathodique (CRT en anglais) sont l'outil par excellence pour observer les signaux périodiques. Pour les phénomènes lents ou sporadiques, ce n'est pas l'idéal. Les oscillos numériques modernes s'en tirent grâce à la mémorisation d'échantillons numériques pour reconstituer toutes formes d'onde, ce dont les premiers oscillos analogiques étaient incapables. Une astuce très répandue consistait à utiliser le tube cathodique lui-même comme mémoire de stockage. On les appelait d'ailleurs *tubes de stockage* ou *bistables*.

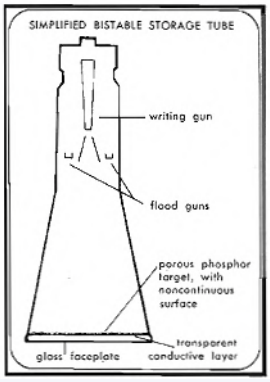


Figure 1. Schéma simplifié d'un tube cathodique de stockage [1].

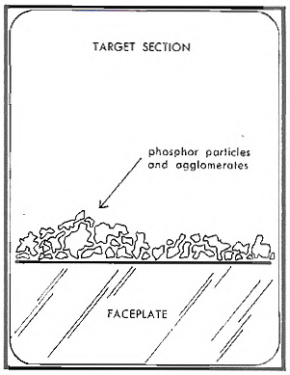


Figure 2. Couche cible de phosphore poreuse [1].

Un tel tube (**fig. 1**) tire parti du fait que la vitesse des électrons circulant entre deux surfaces est commandée par la différence de tension entre elles. Il comporte deux canons à électrons : un canon d'écriture et un canon dit d'inondation (*flood gun*). Le canon d'écriture est celui du tube ordinaire, il inscrit la forme d'onde sur le luminophore du CRT. La différence avec un oscillo classique est sa tension d'alimentation suffisamment élevée pour créer une charge positive dans le phosphore. Cette charge positive est créée lorsque les électrons se déplacent suffisamment vite pour que, lorsqu'ils frappent le phosphore, ils libèrent plus d'électrons qu'ils n'en frappent.

Une fois la forme d'onde écrite, elle reste visible grâce aux canons à inondation. Ceux-ci frappent le phosphore en envoyant un flux d'électrons de faible énergie sur toute la surface. Les électrons frappent la surface non écrite trop lentement pour libérer des électrons nombreux, de sorte que ces zones sont légèrement chargées négativement et que le flux de courant s'arrête. Mais les zones chargées positivement, où la forme d'onde a été écrite, attirent les électrons et les accélèrent. Lorsqu'ils touchent la surface du phosphore, ils libèrent des électrons en nombre égal à ceux qui l'ont frappée. Cela maintient la charge positive et crée un flux de courant pour éclairer la trace.

Cependant, cette méthode pose un problème, car le tracé de la forme d'onde sauvegardée gonfle et finit par baver comme l'encre sur du papier mouillé. Le *Memoscope 104* de *Hughes* utilisait plusieurs grilles de maillage dans la cible pour limiter l'effet, mais il manquait de contraste et de netteté. Sa fabrication était également coûteuse. En 1959, Bob Anderson a entrepris de chercher pour Tektronix une solution meilleur marché. Sachant que le phosphore devait avoir une surface de cible discontinue ou brisée, il a d'abord essayé d'utiliser des motifs de points dans le phosphore. Cette approche, malheureusement incompatible avec les exigences d'une production constante, l'a conduit à mettre au point néanmoins une couche poreuse de particules de phosphore dispersées sur la cible comme support de stockage (**fig. 2**)… et un nouveau tube, utilisé pour la première fois dans l'oscillo Tek 564 (1963), suivi rapidement par l'oscillo 549 dont l'écran stockait deux formes d'onde différentes. Ce procédé est apparu sur les grands écrans des terminaux d'ordinateur de la série 4000, difficiles à trouver de nos jours. Si vous en croisez un, mettez la main dessus, il contient des trésors d'ingénierie cachés.

190383-E-04

LIEN

[1] **Vintage Tek - Histoire du stockage** : https://vintagetek.org/wp-content/uploads/2011/10/The-Storage-Story3.pdf

intelligence artificielle pour débutants (3)

Créez votre propre réseau de neurones

Walter Trojan (Allemagne)

Dans les deux articles précédents de cette série, j'ai présenté la carte Maixduino et montré comment avec MicroPython mettre en œuvre un réseau de neurones prêt à l'emploi pour reconnaître les visages dans les images. Ce troisième et dernier épisode vous propose de créer votre propre réseau neuronal. Il traite également de l'ESP32 qui, en tant que coprocesseur sur la carte, prend en charge les tâches de communication.

J'ai fait jusqu'ici sur le Maixduino des démonstrations d'intelligence artificielle (IA) déjà formée. Si vous souhaitez créer vos propres réseaux neuronaux (que, pour faire court, nous appellerons NN pour *neuronal network*), vous devrez passer par le stade du développement et de la *formation* de votre intelligence artificielle sur un PC ou un service dématérialisé, puis transférer vers Maixduino votre NN une fois formé. Qu'il faille passer par cette procédure n'est pas un inconvénient, car la formation de votre NN nécessite de toute manière à la fois de grandes quantités de données et une grande puissance de traitement. Une fois votre NN formé, il sera utilisable sur une plateforme mobile, avec une faible consommation d'énergie, p. ex. une commande de robot.
Je décrirai donc le chemin qui mène de la conception d'un NN à son utilisation sur le Maixduino. Tout d'abord, cela nécessite sur le PC un environnement de développement typique sous Linux (**fig. 1**).

Kit d'IA

Comme interface avec le développeur, nous aurons l'infrastructure d'intelligence artificielle Keras, le Lego de l'intelligence artificielle. Elle s'appuie sur l'infrastructure bien éprouvée *TensorFlow*, qui fait tout le travail. Il existe aussi un système alternatif comme Theano. Si le PC est équipé d'une carte graphique Nvidia, vous pouvez utiliser les nombreux cœurs GPU via la bibliothèque CUDA, sinon vous mettez les CPU du PC en action avec BLAS.

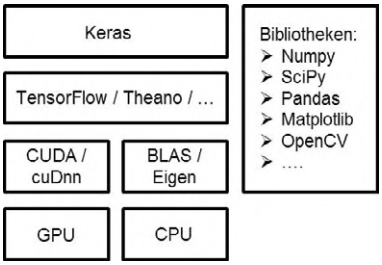


Figure 1 : Structure d'un environnement populaire de développement de l'IA.

Voici un récapitulatif des principaux éléments constitutifs :
- *Keras* : infrastructure d'IA, basée sur *TensorFlow* ou autre infrastructure
- *TensorFlow* : infrastructure d'IA puissante et répandue, conçue et fournie par Google
- *CUDA / cuDnn* : bibliothèque de support des GPU Nvidia
- *BLAS / Own* : bibliothèque de programmes pour des opérations élémentaires d'algèbre linéaire comme la multiplication de vecteurs et de matrices.

Les bibliothèques suivantes sont intégrées selon les besoins :
- *Numpy* : fournit des fonctions très efficaces, en particulier pour les opérations matricielles
- *SciPy* : bibliothèque basée sur *Numpy* avec des fonctions supplémentaires, p. ex. des équations différentielles
- *Pandas* : tableur utilisable dans le programme
- *Matplotlib* : dessine des diagrammes et des illustrations
- *OpenCV* : traitement d'images puissant
- *HDF5* : bibliothèque pour le traitement et le stockage d'ensembles de données hétérogènes
- *Graphviz* : visualisation d'informations structurées
- *pydot-ng* : utilitaire pour Graphviz
- *Jupyter* : environnement de développement pour Python, les sections de programme sont structurées en cellules exécutables individuellement, comme le préconisent les professionnels

D'autres infrastructures d'IA sont disponibles, p. ex. Theano, Torch, Caffe, Pytorch, Yolo avec bien sûr d'autres bibliothèques. Cela témoigne de la richesse du monde Linux/Python et justifie des recherches sur Google. L'environnement (fig. 1) peut être configuré sur un PC sous Linux à l'aide des commandes de terminal suivantes :

Mise à jour :
```
sudo apt-get update
sudo apt-get upgrade
```

Python et pip :
```
sudo apt-get install python3-pip python3-dev
pip install --upgrade pip
```

OpenBlas :
```
sudo apt-get install build-essential cmake git unzip
pkg-config libopenblas-dev liblapack-dev
```

SciPy, Numpy :
```
sudo apt-get install python-numpy python-scipy python-yaml
```

matplotlib :
```
sudo pip3 install matplotlib
```

HDF5 :
```
sudo apt-get install libhdf5-serial-dev python-h5py
```

Graphviz, pydot-ng :
```
sudo apt-get install graphviz
sudo pip3 install pydot-ng
```

OpenCV :
```
sudo apt-get install python3-opencv
# vers. 3, la procédure de la v.4 est complexe
```

Tensorflow 2.0 :
```
pip3 install tensorflow==2.0.0b1
```

Keras :
```
sudo pip3 install keras
```

Pandas :
```
sudo apt-get install python3-pandas
```

Thonny :
```
sudo apt-get install thonny
```

Jupyter :
```
pip3 install jupyter
```

Appel :
`jupyter notebook` (commande du terminal)

Nous avons là nos éléments de base. S'il faut des composants supplémentaires, spécifiques au projet, ils seront installés avec `pip` ou `apt-get`. L'éditeur Python choisi, *Thonny*, se contente de fonctions minimales mais suffisantes. Les pros en utilisent d'autres comme Jupyter, qui permet de structurer les programmes en cellules exécutables individuellement. Tout est prêt pour passer au développement d'un NN à partir de zéro puis pour le transférer sur Maixduino. Je propose comme exemple la reconnaissance de chiffres manuscrits. La base de données MNIST (*Modified National Institute of Standards and Technology database*) en est la base, avec 60.000 échantillons d'images disponibles pour la formation et 10.000 pour les tests. Toutes les images au format 28 x 28 pixels ont une étiquette avec la valeur numérique correcte (**fig. 2**).

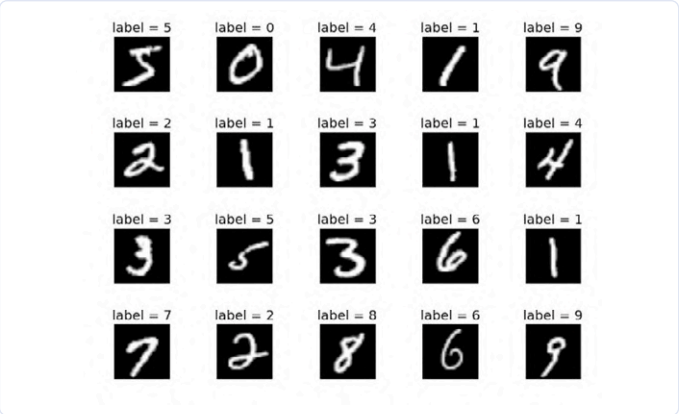


Figure 2. Extrait de la base de données du MNIST.

À petits pas vers le NN

Le NN que nous obtiendrons avec Keras/Tensorflow est destiné à classer correctement des chiffres manuscrits encore inconnus de lui. En principe, le développement d'un NN se déroule comme suit :

- **préparation des données** : convertir les données de formation et de test en un format approprié
- **définition** : définition de la structure NN avec le type et le nombre de couches
- **compilation** : traduction des instructions Keras en instructions Tensorflow
- **entraînement** : reprise répétée des données d'entraînement pour ajuster la pondération NN
- **test** : vérifiez l'exactitude de la reconnaissance à l'aide des données du test.
- **reconnaissance** : présentation de nouveaux chiffres et classification

Préparation des données

Les données du MNIST disponibles dans la bibliothèque Keras sont importées dans le format approprié.

Définition

Le modèle suivant, mis au point par des professionnels, présente un excellent taux de détection de plus de 99 %.

```
model = Sequential()
     # modèle séquentiel, il en existe de
     # fonctionnels
model.add(Conv2D(32, kernel_size=(3, 3),
     # première couche de convolution
     # bidimensionnelle
       avec 32 sorties
                activation='relu',
                   # fonction d'activation
                input_shape=input_shape))
                   # format d'entrée 28*28
model.add(Conv2D(64, (3, 3), activation='relu'))
     # deuxième couche de convolution avec 64 sorties
       fonction d'activation relu
model.add(MaxPooling2D(pool_size=(2, 2))
     # densification
model.add(Dropout(0.25))
     # renforcement de la robustesse
model.add(Flatten())
     # convertir de deux à une dimension
model.add(Dense(128, activation='relu'))
     # troisième couche NN avec 128 sorties et
```

ESP32 - AIDE POUR LES TÂCHES SPÉCIALES

La puce d'intelligence artificielle K210, par ailleurs bien dotée, manque d'entrées analogiques et ne communique ni par WiFi ni par Bluetooth. Une brèche que comble l'ESP32 qui, outre ces fonctions, dispose d'un double processeur avec horloge de 240 MHz. Le K210 et l'ESP32 sont couplés par une connexion SPI rapide◦ ; six entrées analogiques sont acheminées vers le connecteur femelle Arduino. Une connexion série à ESP32 est possible via le port *ttyUSB1*, et l'ESP32 est programmé via le K210 en utilisant l'EDI MaixPy.

Voici un premier exemple pour l'acquisition de signaux analogiques :

```
import network
   # importation des modules requis
import utime
from Maix import GPIO
from fpioa_manager import *

#iomap at MaixDuino
   # enregistrement des GPIO pour l'interface ESP32
fm.register(25,fm.fpioa.GPIOHS10)   # cs
fm.register(8,fm.fpioa.GPIOHS11)    # rst
fm.register(9,fm.fpioa.GPIOHS12)    # rdy
fm.register(28,fm.fpioa.GPIOHS13)   # mosi
fm.register(26,fm.fpioa.GPIOHS14)   # miso
fm.register(27,fm.fpioa.GPIOHS15)   # sclk

# definition of network interface
nic = network.ESP32_SPI(cs=fm.fpioa.GPIOHS10,rst=fm.fpioa.GPIOHS11,rdy=fm.fpioa.GPIOHS12,
mosi=fm.fpioa.GPIOHS13,miso=fm.fpioa.GPIOHS14,sclk=fm.fpioa.GPIOHS15)

adc = nic.adc( (0,1,2) )
    # résultats de ADC0 ADC1 ADC2
print('ADC 0,1,2')
    # impression des résultats de ADC 0-2
print(adc)
print()
print('ADC 0,1,2,3,4,5')

while True:
    try:
        adc = nic.adc()
    # résultat ADC0-5
    except Exception as e:
        print(e)
           # en cas d'erreur imprime la cause
        continue
    for v in adc:
        print("%04d" %(v), end=" ")
           # impression des résultats formatés de ADC 0-5
    print()
    utime.sleep_ms(1000)
           # pause 1000 ms
```

Au début du programme, les modules Python requis sont importés, puis les GPIO pour la communication avec ESP32 sont enregistrés et affectés à l'objet `nic.` Pour enregistrer les entrées analogiques, la fonction `nic.adc` reçoit un tuple avec les canaux

```
    # activation relu
model.add(Dropout(0.5))
    # renforcement de la robustesse
model.add(Dense(num_classes, activation='softmax'))
    # quatrième couche NN avec 10 sorties et
    # activation softmax
```

Le NN dispose de quatre couches, deux convolutives et deux classiques (*Dense*) avec un nombre de nœuds (32/64/128/10) qui reste intelligible.

L'activation de la dernière couche avec `softmax` limite la somme des valeurs initiales à la probabilité 1. Dans les deux premières couches, les détails tels que les lignes, les arcs et les points sont filtrés dans des champs de 3 x 3 pixels chacun. Entre les couches sont insérés divers filtres :

- `MaxPooling` compresse le résultat d'une couche et ne transfère que les valeurs moyennes d'un champ (ici 2 x 2). `Dropout` néglige

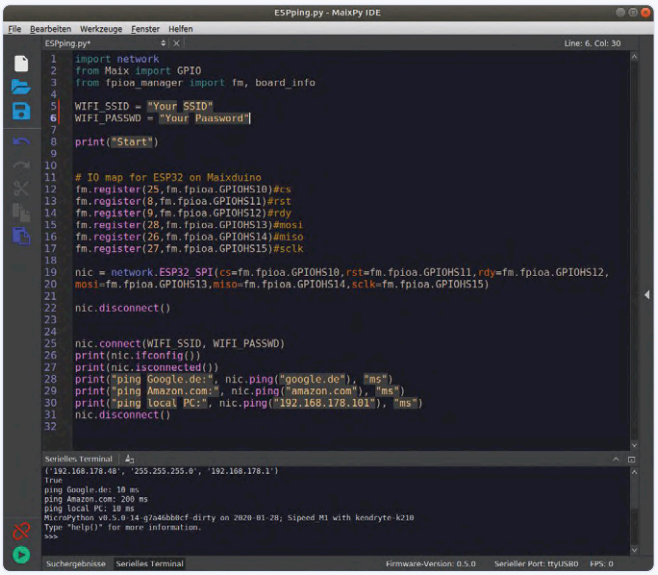
souhaités, ici ADC0 à ADC2. Après l'affichage de ces résultats, les six entrées disponibles sont même interrogées dans la boucle `while` sans nommer les numéros de canaux. Si des erreurs se produisent lors de l'acquisition, le code d'erreur est donné par la fonction d'exception.

Pour vérifier le fonctionnement de l'acquisition analogique, j'ai alimenté un niveau d'entrée variable au canal ADC0 via un potentiomètre. Veuillez noter que l'entrée analogique native ne couvre que la plage de 0 à 1,0 V. Les valeurs affichées par le terminal confirment le bon fonctionnement :

```
ADC 0,1,2
(0, 1786, 73)

ADC 0,1,2,3,4,5
0000 1469 0082 0521 0120 0001
    # potentiomètre à 0 V
0000 1813 0160 0606 0291 0000
2622 2135 0210 0630 0323 0000
    # potentiomètre à mi-course
4095 2421 0259 0575 0261 0000
4095 2472 0274 0615 0315 0000
    # potentiomètre à 1,0 V
```

Les autres canaux n'étaient pas connectés et affichent des valeurs aléatoires dans les colonnes 1 à 5.



Un deuxième exemple illustre la communication sur l'nternet◦ : deux portails bien connus, Google et Amazon, ainsi qu'un ordinateur local sont interrogés (voir la capture d'écran).

L'importation des modules ainsi que l'enregistrement des ports GPIO et la définition de l'objet `nic` se font comme dans le premier exemple. Pour utiliser l'internet, les données d'accès au routeur WiFi sont bien sûr nécessaires. Après la connexion au réseau, l'adresse IP de l'ESP32 peut être demandée avec `nic.ifconfig()` et l'état de la connexion est affiché avec `nic.isconnected()`. Avec la commande `nic.ping()`, seule l'adresse de destination doit être spécifiée. Outre les adresses internet, les ordinateurs locaux peuvent également être *pingés* à leur adresse IP.

Il existe de nombreux autres exemples d'application, par exemple un serveur web pour la transmission de l'analyse IA, une collection d'objets provenant d'Internet, et bien d'autres encore. Laissez-vous inspirer par les exemples présentés sous le lien [5] et n'hésitez pas à expérimenter. Veuillez noter que Maixduino est un encore jeune et dans une phase de développement assez mouvementée, c'est-à-dire qu'en plus du micrologiciel du K210, une mise à jour est également nécessaire de temps en temps pour la connexion de l'ESP32. Cela peut être fait avec la séquence de commandes suivante après avoir téléchargé le fichier binaire mentionné ci-dessous [4] :

```
pip install esptool
esptool.py --chip esp32 --port /dev/ttyUSB1 erase_flash
esptool.py --chip esp32 --port /dev/ttyUSB1 --baud 1500000 write_flash -z 0x0000
   maixduino_esp32_firmware_v1.4.0.bin
```

des nœuds choisis au hasard pendant la formation et attribue ainsi plus d'importance aux nœuds voisins, ce qui augmente la robustesse du NN. `Flatten` transforme une sortie multidimensionnelle de CNN en sortie unidimensionnelle.

Compilation

Dans cette phase a lieu la conversion des instructions Keras en instructions Tensorflow.

```
model.compile(loss=keras.losses.categorical_
   crossentropy,   # attribution de la fonction de
   perte
       optimizer=keras.optimizers.Adadelta(),
           # sélection de l'optimiseur
              metrics=["accuracy"])
                   # définition de la précision
```

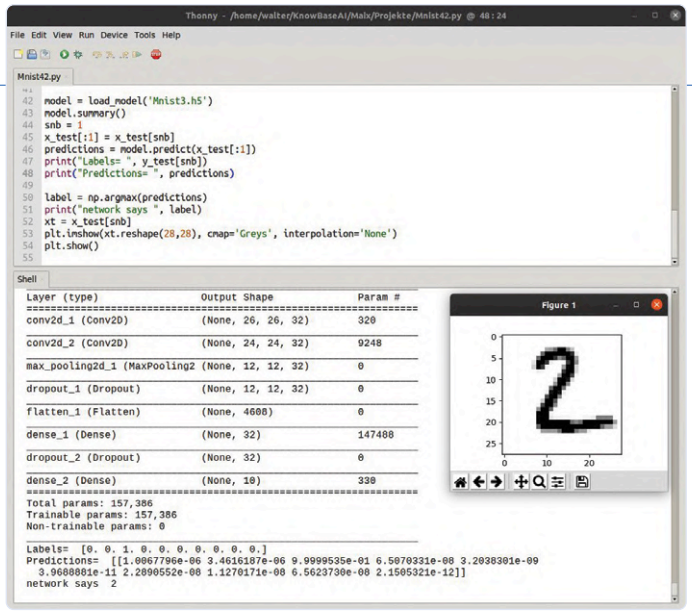


Figure 3. Reconnaissance du chiffre 2.

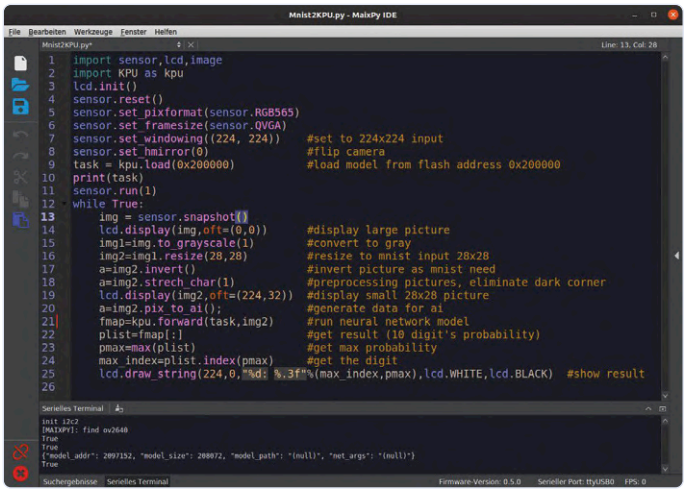


Figure 4. Reconnaissance de chiffres avec MicroPython.

Formation

Ici l'entraînement du NN avec 60.000 symboles numériques se joue en 12 manches (époques).

```
model.fit(x_train, y_train,
      # x_train contient des images, y_train contient
      # des étiquettes
            batch_size=batch_size,
               # la formation se fait par lots,
               # p. ex. par paquets de 16
            epochs=epochs,
              # nombre d'epochs, manches avec toutes
              # les données
            verbose=1,
              # résultat de la formation
            validation_data=(x_test, y_test))
              # validation du modèle avec les données
              # de test
```

Test

Enfin, le taux de détection est déterminé à l'aide des 10.000 données du test.

```
score = model.evaluate(x_test, y_test, verbose=0)
model.save("Mnist3.h5")
```

Le modèle peut maintenant être sauvegardé avec la structure et les poids formés au *format h5* et utilisé plus tard – également sur une autre plateforme – pour la reconnaissance.

Détection

La détection se fait localement, sur Maixduino après le transfert, mais vous pouvez déjà la faire sur le PC sous Linux avec l'éditeur Thonny (**fig. 3**).
On charge le NN formé pour qu'il reconnaisse de nouveaux chiffres manuscrits ; ensuite sera produite la structure du modèle. Le lecteur attentif aura remarqué que dans certaines couches, le nombre de nœuds a été réduit pour que le NN tienne dans la mémoire plus exiguë (max. 5,9 Mo ici). Ensuite, le deuxième élément des données d'essai est reconnu, par hasard un «2», comme le confirme l'étiquette de cet ensemble de données. Le NN voit les choses de la même manière, car sa reconnaissance donne une probabilité de 0,9999 pour cet indice, alors que tous les autres chiffres obtiennent des valeurs inférieures et négligeables. En ne retenant que la valeur la plus grande du tableau, *numpy,* avec la fonction `argmax` contribue à produire une évaluation compréhensible. Par ailleurs, l'image à reconnaître est dessinée grâce à la bibliothèque *matplotlib*. La formation et le test sur le PC sous Linux sont achevés. Il reste le transfert vers le Maixduino, avec les étapes suivantes :

- créer le modèle avec formation et test à l'aide de Keras, le résultat est un modèle d'IA au format *h5* (comme indiqué ci-dessus)
- convertir le *modèle h5* en TensorflowLite au format *tflite*
- compiler le modèle *tflit* avec le compilateur *nncase* en un format pour la KPU, le *format kmodel*
- flasher ce modèle dans Maixduino en utilisant Kflash à l'adresse 0x200000
- créer un script Python en utilisant l'IDE MaixPy et le télécharger sur Maixduino pour l'exécuter et le tester

Vous trouverez une documentation supplémentaire à ce sujet dans le dossier du projet [3], où tous les programmes et fichiers utilisés dans l'article sont annotés et commentés.
La figure 4 montre l'application sur le Maixduino.

Après l'importation des modules requis, l'appareil photo et l'écran sont initialisés et ajustés. Ensuite, le NN de l'adresse 0x200000 est chargé et est disponible avec l'objet `task`. Dans la boucle `while` une image est d'abord capturée puis affichée sur l'écran. Ensuite, l'image est préparée pour le NN : Conversion en gris, conversion en 28 x 28 pixels et inversion pour le MNIST. Après avoir ajusté les valeurs des pixels à un format utilisable par l'IA (généralement division par 255), l'image est transférée au NN pour reconnaissance. Celui-ci renvoie son résultat dans la variable `fmap`. Ensuite apparaissent sur l'écran le chiffre reconnu et la probabilité de succès de cette reconnaissance. Vite faite bien faite, une reconnaissance de chiffres en 25 commandes ! Le test pratique (**fig. 5**) confirme le bon fonctionnement de notre NN et, avec la reconnaissance de quelques images par seconde, la performance de la KPU.

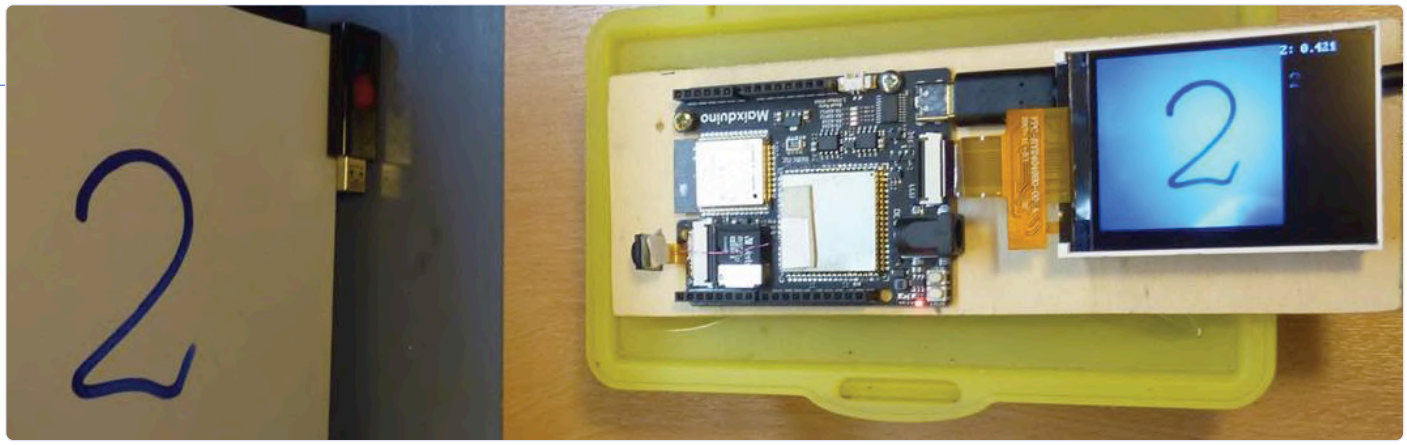


Figure 5. Configuration de test de la reconnaissance de chiffres avec Maixduino.

Ce n'est que le début !

Le matériel puissant et l'environnement logiciel aujourd'hui disponible montrent à quel point Maixduino est commode pour se frotter à l'intelligence artificielle. Grâce à sa faible consommation, il peut être utilisé dans des appareils mobiles avec des réseaux neuronaux déjà formés. Pour faire du développement, de la formation et de l'exécution sur une même plateforme, on préférera des systèmes plus trapus, avec Linux installé, p. ex. le Nvidia Jetson Nano avec sa flopée de logiciels et un support solide. Consultez éventuellement les nouvelles versions comme Rock Pi N10 et d'autres, mais avant d'acheter assurez-vous de la disponibilité du logiciel et d'une documentation de qualité. Et n'oubliez pas que pour commencer un PC sous Linux, avec ou sans carte graphique GTX, fait tout aussi bien.

Ça bouge dans l'IA, et ce n'est que le début. Cette petite série n'avait pour ambition que de présenter une partie de la partie visible de l'iceberg. Car outre la méthode d'*apprentissage supervisé* présentée ici, dans laquelle un réseau neuronal est formé avec des données connues dites *étiquetées*, il existe bien d'autres solutions d'IA.

Dans l'*apprentissage non supervisé,* des données inconnues sont introduites dans le réseau neuronal et celles-ci sont divisées en grappes, c'est-à-dire en groupes similaires les uns aux autres, en fonction de leurs caractéristiques. Cette méthode est utilisée, par exemple, pour l'analyse de grands ensembles de données afin d'identifier certains points communs. Ou, dans le cas des *autocodeurs,* pour la suppression du bruit dans la reconnaissance des caractères.

Vous êtes-vous déjà demandé comment il est possible que les NN se révèlent meilleurs que les humains dans certains domaines et puissent même les battre au jeu de Go alors que les données d'apprentissage leur ont été fournies par des humains ? Il existe pour cela un autre type d'IA, l'*apprentissage renforcé*, dans lequel le NN est motivé par un système de récompense dans lequel les résultats sont améliorés en combinant plusieurs étapes d'analyse. Cette méthode est utilisée dans le secteur financier et pour les jeux informatiques, entre autres. Un des points faibles de l'IA est à l'heure actuelle l'architecture de la «boîte noire» : après la saisie des données, l'IA effectue une classification sans la justifier. On ne sait toujours pas, par exemple, pourquoi l'IA suggère à un cabinet de recrutement tel candidat plutôt que tel autre. On a ainsi constaté que tel système préférait les hommes, simplement parce dans les données de formation de l'IA les hommes étaient plus nombreux que les femmes. Ces faibles sont l'objet de recherches approfondies pour tenter de rendre *explicable* l'*IA :* bientôt les NN justifieront leurs décisions qui deviendront ainsi (plus) compréhensibles pour les utilisateurs.

Vous aussi pouvez participer à ce vaste mouvement, en vous aidant des vidéos, des tutoriels et de la documentation technique disponibles. Vous pourrez bientôt ajouter une bonne part d'intelligence à vos propres projets. Un dernier conseil, car ça ne manquera pas de vous paraître complexe et même difficile : n'abandonnez jamais !

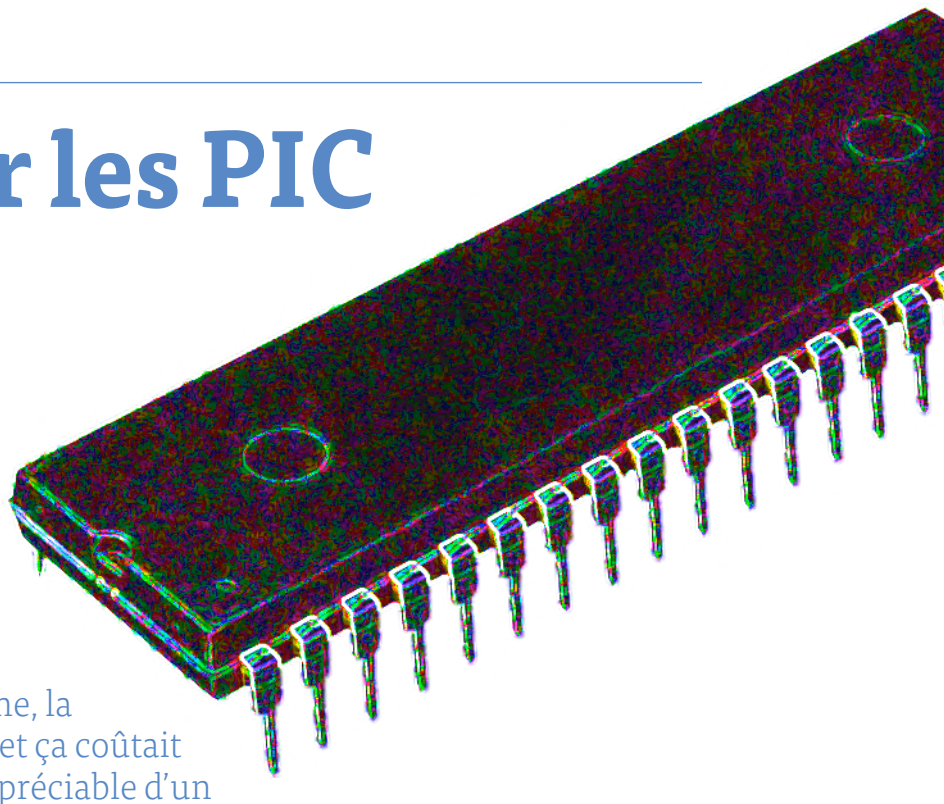
200023-C-02

LIENS

[1] **intelligence artificielle pour débutants (1), Elektor mai/juin 2020** : www.elektormagazine.fr/200023-01
[2] **intelligence artificielle pour débutants (2), Elektor juillet-août 2020** : www.elektormagazine.fr/200023-B-02
[3] **intelligence artificielle pour débutants (3) - cet article** : www.elektormagazine.fr/200023-C-02
[4] **progiciel ESP32** : https://github.com/sipeed/Maixduino_esp32_fimware/releases/tag/v1.4.0
[5] **programmes de démonstration ESP** : https://github.com/sipeed/MaixPy_scripts/tree/master/network

programmer les PIC à petits pas

Produire des sinusoïdes en assembleur



Tam Hanna (Slovénie)

À l'époque où les 8 bits étaient la norme, la capacité des mémoires était modeste et ça coûtait cher. Pour obtenir quelque chose d'appréciable d'un µP à moins de 5 MHz, il fallait un code costaud sans une once de gras. On n'avait donc pas d'autre choix que d'écrire en assembleur qu'il fallait transformer en un code hexécutable. Je propose ici d'écrire une routine en assembleur pour un PIC d'aujourd'hui, et produire une sinusoïde avec son CN/A intégré.

Pour le logiciel, nous utiliserons MPLAB X l'environnement intégré (IDE) de *Microchip,* et un PIC 16F18877. La programmation proprement dite passe par l'interface ICSP. Vous gardez donc la liberté de choisir la carte de développement ou de prototypage qui vous plaît.
Pour produire une onde sinusoïdale avec un CN/A, nous devons lui fournir à intervalles réguliers des valeurs numériques correspondant aux niveaux de tension successifs d'une sinusoïde. Par période, nous utiliserons ici 32 valeurs discrètes. La fonction `sin()` du tableur Excel permet de créer le tableau (**fig. 1**) qui donne dans la colonne de gauche les intervalles de temps de 0 à 31. C'est la formule `=A2*((2*PI())/32)` qui nous fournit les valeurs pour une période de 0 à 2π radians. La troisième colonne contient les valeurs sinusoïdales des radians de la colonne 2 `=SIN(B2)`. Ces valeurs sont comprises entre –1 et +1, or le CN/A n'accepte que des valeurs positives. Nous décalerons donc toutes les valeurs `=1+C2` de la plage de 1 à +1 dans la plage positive de 0 à 2. Ces valeurs ajustées sont ensuite mises à l'échelle à l'aide de `D3*(255/2)` pour les situer dans la plage de 0 à 255 qui correspond aux valeurs d'entrée codées sur 8 bits attendues par le CN/A. Enfin, les valeurs sont arrondies : `ARRONDI(E10)`.

Tableaux en stock

Les valeurs d'une fonction sinusoïdale sont des constantes qui peuvent être stockées en mémoire sous forme de table(au). Pour récupérer une valeur dans la table, nous pouvons utiliser l'instruction (mnémonique) `RETLW`. Celle-ci retourne avec la valeur de la table chargée dans W (l'accumulateur).
En assembleur, nous pouvons assigner une zone de mémoire à cette table. Dans les langages de niveau supérieur, on n'a pas à se soucier de tels détails, le compilateur se débrouille. Il est possible de commencer cette zone de stockage presque n'importe où, mais nous verrons plus tard qu'il y a des avantages à préférer certaines adresses. Ici, l'adresse de départ du tableau sera `0x200` :

```
TABLE_VECT  CODE  0x0200
  dt  127, 152, 176, 198, 217, 233, 245, 252, 255,
   252, 245, 233, 217, 198, 176, 152, 127, 102, 78,
   56, 37, 21, 9, 2, 0, 2, 9, 21, 37, 56, 78, 102
END
```

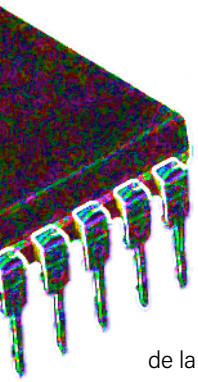
La directive `dt` (*define table*) stocke les valeurs fournies sous forme de tableau. Ce fragment de code peut déjà être passé en assembleur. L'assembleur MPLAB ignorera bien des situations critiques, mais lorsqu'il lance un avertissement tel que :

```
Warning[202] C:\USERS\TAMHA\MPLABXPROJECTS\CH6-
   DEMO1.X\NEWPIC_8B_SIMPLE.ASM 72 : Argument out of
   range.  Least significant bits used.
```

il faut le prendre au sérieux et résoudre le problème ; les erreurs sont une partie essentielle du processus d'apprentissage. C'est la façon de définir le tableau qui est incorrecte ici.

Détour : assemblage et désassemblage

Il existe un lien direct bien défini entre les mnémoniques (le nom des instructions) et le code machine sous-jacent. Le code machine est créé *par assemblage* des mnémoniques. Le processus inverse, du code machine aux mnémoniques, est le désassemblage.
Double-cliquez sur l'onglet *Usage Symbols disabled* **(fig. 2)** en bas à droite. Une fenêtre de réglage s'ouvre. Cochez *Load Symbols when programming or building for production (*c'est-à-dire *Charger les*



symboles lors de la programmation ou de la construction pour la production) afin d'activer les outils d'analyse durant la compilation.

Après avoir cliqué sur *Apply, relancez* le compilateur afin d'afficher l'utilisation de la mémoire. L'option *Window > Debugging > Output > Disassembly Listing File Project indique* au compilateur d'afficher une version désassemblée de la section de code originale. Cela peut être utile car cela nous montre comment le compilateur traite le code **(fig. 3)**.

La sortie se compose de six colonnes, avec à gauche l'adresse *logique* du mot dans la mémoire de programme du PIC. La colonne suivante est l'équivalent décimal de la commande. La troisième colonne contient la version désassemblée obtenue à partir du code machine hexadécimal (sans toutefois les noms de constantes ou de variables et les commentaires du fichier assembleur original). La quatrième colonne indique le numéro de ligne dans le fichier .asm et, après les deux-points, se trouve la ligne responsable du mot binaire à gauche.

Nous voyons maintenant que, par défaut, l'assembleur suppose que les valeurs stockées dans le tableau sont hexadécimales, et n'a donc utilisé que les deux derniers chiffres de chaque nombre (codée sur 8 bits, une valeur hexadécimale est constituée de deux caractères, son équivalent décimal en compte trois) :

```
0200  3427  RETLW 0x27     73:     dt  127, 152, 176,
   198, 217, 233, 245, 252, 255, 252, . . .
0201  3452  RETLW 0x52
0202  3476  RETLW 0x76
0203  3498  RETLW 0x98
0204  3417  RETLW 0x17
0205  3433  RETLW 0x33
. . .
```

Ces tableaux admettent toutes sortes des valeurs (hexadécimales, binaires, octales, décimales, etc.). Il faut donc, avant chaque valeur, en préciser sa «base». Ainsi, un point décimal indiquera à l'assembleur que la valeur est décimale. Pendant l'assemblage, le compilateur prendra la valeur décimale maximale (8 bits) de 255 et la convertira en valeur hexadécimale maximale FF pour l'utiliser dans le code. Maintenant le tableau se présente comme ceci :

```
TABLE_VECT  CODE  0x0200 dt  .127, .152, .176, .198,
   .217, .233, .245, .252, .255, .252, .245, .233,
   .217, .198, .176, .152, .127, .102, .78, .56, .37,
   .21, .9, .2, .0, .2, .9, .21, .37, .56, .78, .102
END
```

Accès aux tableaux

Pour récupérer les informations stockées dans la table, nous y accédons avec l'instruction `RETLW`. Au retour, le registre W (l'accumulateur) contient la valeur de la table. Pour comprendre, convertissons en binaire l'adresse de départ `0x0200` : `0000 0000 0000 0000 0010 0000 0000 0000 0000 0000`.

Les appels vers des emplacements dans la mémoire du programme peuvent être effectués avec l'instruction `CALLW`, (**fig. 4**).

La mémoire du PIC compte 32.768 mots, pour laquelle il faut donc un pointeur d'adresse de 15 bits. Les instructions comme `CALLW` utilisent le registre W pour transmettre la valeur du compteur ordinal, lequel pointe vers l'emplacement de la mémoire où commence le sous-programme. Comme le format de W n'est que de 8 bits, on utilisera le

Schritt	Laufwert	Sinuswert	Bereinigt	DAC-Wert	Abgerundet
0	0,0000	0,0000	1,0000	127,5000	127
1	0,1963	0,1951	1,1951	152,3740	152
2	0,3927	0,3827	1,3827	176,2921	176
3	0,5890	0,5556	1,5556	198,3352	198
4	0,7854	0,7071	1,7071	217,6561	217
5	0,9817	0,8315	1,8315	233,5124	233
6	1,1781	0,9239	1,9239	245,2946	245
7	1,3744	0,9808	1,9808	252,5501	252
8	1,5708	1,0000	2,0000	255,0000	255
9	1,7671	0,9808	1,9808	252,5501	252
10	1,9635	0,9239	1,9239	245,2946	245
11	2,1598	0,8315	1,8315	233,5124	233
12	2,3562	0,7071	1,7071	217,6561	217
13	2,5525	0,5556	1,5556	198,3352	198
14	2,7489	0,3827	1,3827	176,2921	176
15	2,9452	0,1951	1,1951	152,3740	152
16	3,1416	0,0000	1,0000	127,5000	127
17	3,3379	-0,1951	0,8049	102,6260	102
18	3,5343	-0,3827	0,6173	78,7079	78
19	3,7306	-0,5556	0,4444	56,6648	56
20	3,9270	-0,7071	0,2929	37,3439	37
21	4,1233	-0,8315	0,1685	21,4876	21
22	4,3197	-0,9239	0,0761	9,7054	9
23	4,5160	-0,9808	0,0192	2,4499	2
24	4,7124	-1,0000	0,0000	0,0000	0
25	4,9087	-0,9808	0,0192	2,4499	2
26	5,1051	-0,9239	0,0761	9,7054	9
27	5,3014	-0,8315	0,1685	21,4876	21
28	5,4978	-0,7071	0,2929	37,3439	37
29	5,6941	-0,5556	0,4444	56,6648	56
30	5,8905	-0,3827	0,6173	78,7079	78
31	6,0868	-0,1951	0,8049	102,6260	102

Figure 1. Table des données prête à l'emploi.

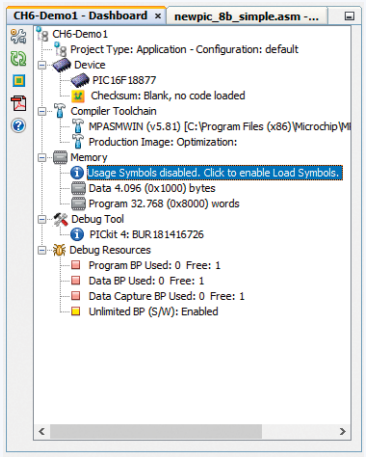


Figure 2. Double-cliquez sur cette ligne pour ouvrir la fenêtre d'options.

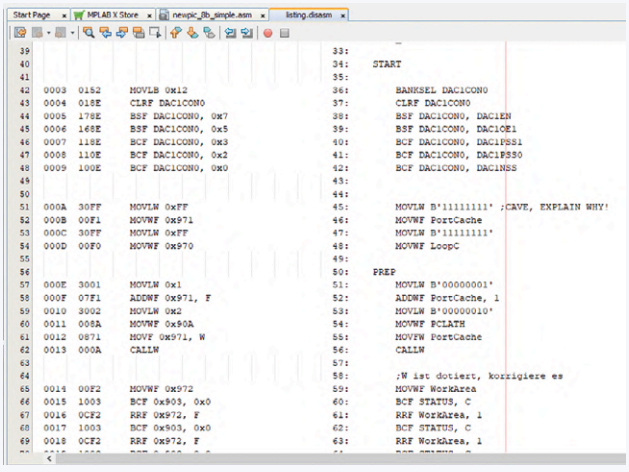


Figure 3. Le code machine brut.

CALLW	Subroutine Call With W
Syntax:	[*label*] CALLW
Operands:	None
Operation:	(PC) +1 → TOS, (W) → PC<7:0>, (PCLATH<6:0>) → PC<14:8>
Status Affected:	None
Description:	Subroutine call with W. First, the return address (PC + 1) is pushed onto the return stack. Then, the contents of W is loaded into PC<7:0>, and the contents of PCLATH into PC<14:8>. CALLW is a 2-cycle instruction.

Figure 4. La commande mnémonique CALLW calcule l'adresse par un processus relativement complexe.

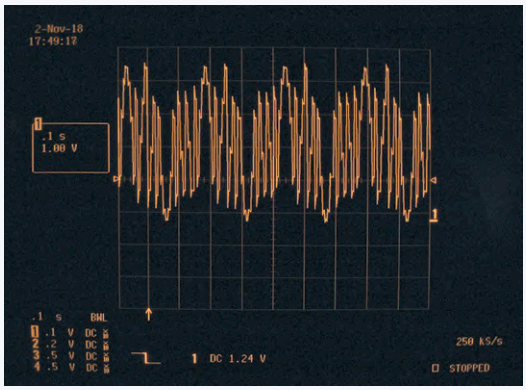


Figure 5. Voilà qui ne ressemble pas du tout à une sinusoïde.

registre PCLATH pour y stocker avant l'appel la partie supérieure du compteur ordinal. Pour réduire la charge et les éventuels conflits de chronologie, le PIC n'utilisera la valeur de PCLATH que si elle est nécessaire. Nous pouvons assembler les valeurs des pointeurs dans les registres avant d'effectuer l'appel. Les écritures sur PCLATH n'affectent pas la valeur actuelle du pointeur ordinal. L'initialisation du programme commence par l'incrémentation du compteur :

```
WORK
    BANKSEL DAC1CON1
    MOVLW B'00000001'
    ADDWF PortCache, 1
```

Dans l'adresse binaire complète ci-dessus, nous pouvons voir quelle valeur doit être chargée dans la partie supérieure du compteur ordinal. Cette valeur est transférée à PCLATH via le registre W :

```
    MOVLW B'00000010'
    MOVWF PCLATH
```

Nous sommes prêts. En exécutant `CALLW`, la valeur de la table est renvoyée dans le registre W. Cette valeur est ensuite transmise au CN/A pour produire le niveau de tension analogique correspondant :

```
    MOVFW PortCache
    CALLW
    MOVWF DAC1CON1
    CALL WAIT
```

Si nous exécutons le programme, ça fonctionne jusqu'à ce que MPLAB interrompe l'exécution à un moment donné parce que le tableau se compose de 32 valeurs alors que le programme en attend 255. Ces emplacements non prévus de la mémoire peuvent contenir des valeurs erratiques provenant de code installé précédemment. À mesure que le pointeur s'incrémente, le programme se retrouve en territoire inconnu. La section suivante du code limite la portée en soustrayant 31, en détectant si le drapeau zéro est levé par l'opération, puis en utilisant `CLRF` avec `PortCache` :

```
    MOVFW PortCache
    SUBLW .31
    BTFSC STATUS, Z
    CLRF PortCache
```

Enfin, nous revenons au début de la boucle :

```
    GOTO WORK        ; loop forever
```

Nous pouvons maintenant exécuter le programme. Le résultat de la **fig. 5** semble chaotique et ne ressemble en rien au sinus attendu. Le CN/A du processeur n'accepte en effet que des valeurs comprises entre 0 et 31, alors que nous lui fournissons des valeurs entre 0 et 255.

Tableaux en mémoire

Nous pourrions recourir à Excel pour réduire les valeurs requises pour la sinusoïde et les utiliser dans le tableau. Nous pourrions aussi utiliser le processeur pour effectuer la division nécessaire des valeurs dans le logiciel. Si nous utilisons l'assembleur, nous avons un accès direct aux valeurs binaires dans les registres. La méthode la plus simple pour effectuer la division par 2 d'une valeur binaire consiste à décaler son motif de uns et de zéros dans le registre d'une position vers la droite. Cette opération ne tient pas compte des bits de retenue et met à zéro le bit de poids le plus fort dans le registre. Pour convertir des valeurs comprises entre 0 et 255 en valeurs comprises entre 0 et 31, nous devons diviser chaque valeur par huit. Cela revient à la décaler trois fois vers la droite.

Lorsque les valeurs sont copiées dans la mémoire, nous devons calculer les adresses cibles. Pour ce faire, nous examinons les registres de base. PCLATH est déjà connu, mais nous nous intéressons également à INDF et FSR. Notre PIC possède deux registres de sélection de fichiers (FSR) de 16 bits. Ceux-ci sont capables d'accéder à tous les registres de fichiers et à la mémoire du programme, ce qui permet d'avoir un seul pointeur de données pour toute la mémoire. Les registres indirects de fichiers (INDF) ne sont pas des registres physiques. Une instruction qui accède à un registre INDFn accède au registre à l'adresse spécifiée par le registre de sélection de fichier (FSR). Les opérations d'écriture dans la mémoire de programme ne peuvent pas être effectuées via les registres INDF.

Une particularité du PIC est le fait que le choix entre mémoire de programme et mémoire de travail se fait par le bit 7 du registre d'adresse supérieur FSRxH. S'il est à 1, l'adresse est celle d'un emplacement dans la mémoire de programme ; sinon c'est dans la mémoire de données. On recommence en stockant les variables. Cette fois, nous avons besoin de 32 octets de mémoire en tout – c'est trop pour une utilisation comme mémoire partagée.

Nous accédons à la mémoire dans une banque que sélectionne l'assembleur et utilisons l'adressage relatif. Comme aucun paramètre supplémentaire n'est spécifié, MPASM laisse donc le libre choix dans

la position du `DataBuffer` :

```
udata_shr
   LoopC res 1
   PortCache res 1
udata
   DataBuffer res 32
```

Les parties haute et basse de l'adresse restent incertaines. Heureusement, le *linker* nous facilite le travail avec deux opérateurs :

```
START
  MOVLW high DataBuffer
  MOVLW low DataBuffer
```

Sachant cela, copions les informations de la mémoire du programme dans la mémoire principale. Les opérations de décalage ne fonctionnent pas directement dans W, il faut donc une variable supplémentaire :

```
udata_shr
   LoopC res 1
   PortCache res 1
   WorkArea res 1
```

Lorsqu'on travaille avec des tableaux de données, la condition initiale est cruciale. Nous chargeons 1111.1111 dans PortCache car la boucle s'incrémente *avant* utilisation. Avec la première incrémentation, elle passera donc à 0. Si nous avions chargé 0, la première incrémentation nous ferait écrire à l'emplacement 1 :

```
START
. . .
   MOVLW B'11111111'
   MOVWF PortCache
   MOVLW B'11111111'
   MOVWF LoopC
```

Il y a deux boucles : la boucle PREP prépare et écrit le tableau des valeurs des sinusoïdes dans un tableau ; la boucle `Work` envoie les valeurs au CN/A afin que la forme d'onde du signal puisse être sortie. PREP commence par un accès au tableau ::

```
PREP
   MOVLW B'00000001'
   ADDWF PortCache, 1
   MOVLW B'00000010'
   MOVWF PCLATH
   MOVFW PortCache
   CALLW
```

Maintenant, avec la valeur dans W, nous devons la déplacer vers le registre F et exécuter trois instructions de rotation à droite pour diviser sa valeur par 8 :

```
   MOVWF WorkArea
   BCF STATUS, Z
   RRF WorkArea, 1
   BCF STATUS, Z
   RRF WorkArea, 1
```

Pour éviter les erreurs causées par le bit de retenue dans le registre d'état, `BCF STATUS, Z` est utilisé pour effacer le drapeau de retenue dans le registre avant chaque opération de décalage. Le registre contient deux petites erreurs intéressantes à corriger.
Nous devons maintenant nous assurer que l'INDF indique le bon emplacement de mémoire : Pour ce faire, les registres FSR0H et FSR0L sont chargés avec des données d'adresse. Le registre H (haut) contient la partie supérieure et le registre L (bas) la partie inférieure :

```
   MOVLW high DataBuffer
   MOVWF FSR0H
   MOVLW low DataBuffer
   MOVWF FSR0L
```

INDF0 indique maintenant le début de la zone de mémoire. Nous devons ajouter le décalage qui identifie chaque emplacement. Il n'est pas certain que le début de la zone se trouve au début d'une page. S'il y avait une retenue lors de l'ajout du décalage dans la partie L, la partie H ne le remarquerait pas. Comme solution à ce problème, le statut du bit C est vérifié et la valeur de FSR0H est incrémentée s'il y a une retenue :

```
   MOVFW PortCache
   CLRC
   ADDWF FSR0L
   BTFSC STATUS, C
   INCF FSR0H
```

La commande `CLRC` (*Clear Carry*) est une macro qui efface le bit C (retenue) dans le registre d'état. Cela permet de s'assurer que INDF0 est correctement configuré. Nous devons charger et écrire la valeur stockée temporairement dans la zone de travail :

```
   MOVFW WorkArea
   MOVLW INDF0
```

Enfin, nous devons nous assurer que la boucle continue de tourner :

```
   MOVFW PortCache
   SUBLW .31
   BTFSS STATUS, Z
   GOTO PREP
   CLRF PortCache
```

Comme ça se complique, il serait utile d'utiliser certains des outils de l'EDI pour examiner de plus près l'exécution du programme et en vérifier le bon fonctionnement.

Débogage en assembleur

Théoriquement, il est facile de placer des points d'arrêt ; cliquez sur les numéros de ligne dans l'IDE pour placer un point d'arrêt rouge. Comme le µC ne reconnait qu'un seul point d'arrêt matériel, il y aura un message d'erreur.
MPLAB utilise des ressources de débogage pour pouvoir fonctionner par étapes. Nous n'avons pas besoin dans le logiciel d'émulation de point d'arrêt pour le moment, nous pouvons donc nier le message en cliquant sur *No*. Le PIC supporte lui les points d'arrêt logiciels, vous

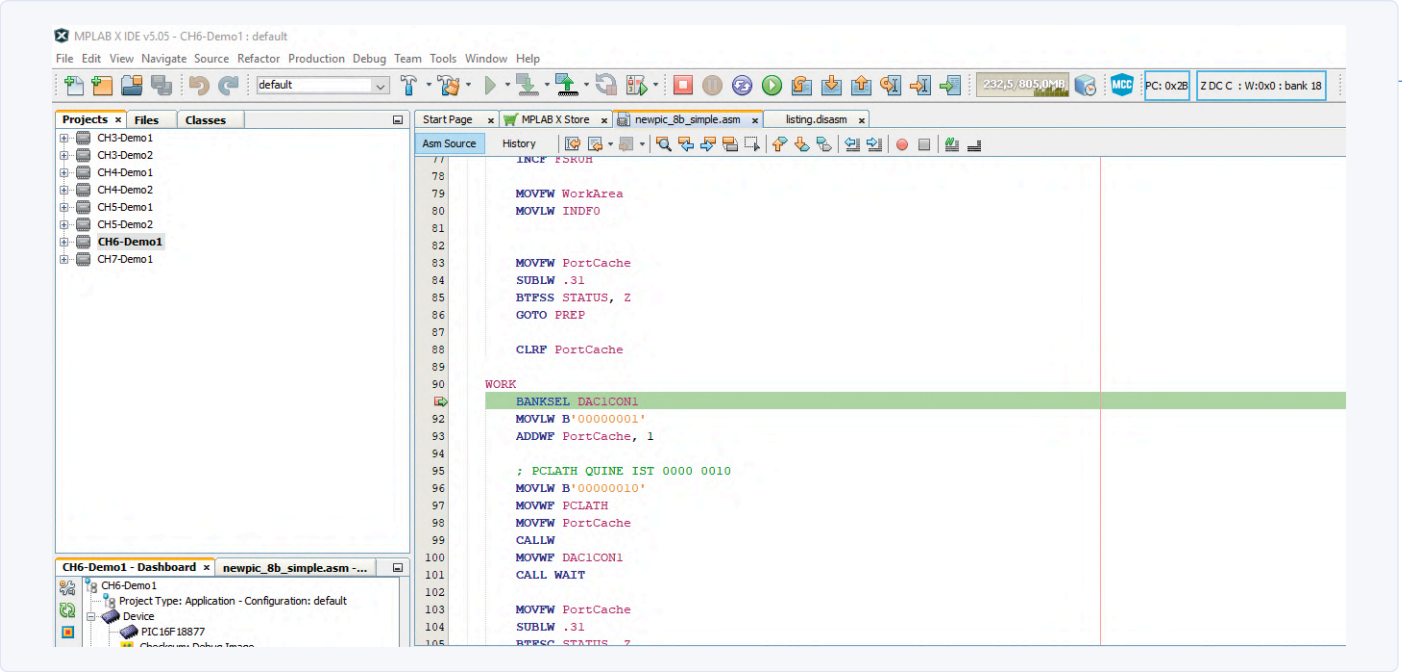


Bild 6. Le débogueur a interrompu l'exécution du programme.

pouvez confirmer avec *Yes.* Dès que vous placerez plus d'un point d'arrêt dans le fichier assembleur, *Microchip* activera cette fonction automatiquement.

Puisque notre PIC ne supporte qu'un seul point d'arrêt matériel, cliquez sur la flèche vers le bas à côté du symbole du débogueur dans la barre d'outils et sélectionnez l'option *Debug main project.* MPLAB ouvre une fenêtre de désassemblage que nous pouvons ensuite fermer. Après avoir atteint le point d'arrêt, l'EDI affiche l'état (**fig. 6**).

La ligne avec le fond vert et la flèche vers la droite dans la colonne des numéros de ligne est l'instruction en cours. Les symboles de la barre d'outils, tels que *stop* et *saut*, permettent à l'utilisateur d'interagir avec le programme. *Window Target Memory Views File Registers* ouvre une fenêtre pour visualiser l'espace mémoire du PIC. Placez le curseur comme le pointeur de la souris sur la déclaration du *DataBuffer*. Cela ouvre une info-bulle avec l'adresse du premier octet et sa valeur. Sur mon poste de travail , cette position est `0xD20`.

Il est plus pratique de cliquer sur la flèche bleue (*GoTo*) dans la fenêtre *File Registers*. Dans la case *GoTo What*, sélectionnez *Symbol* et *DataBuffer*.

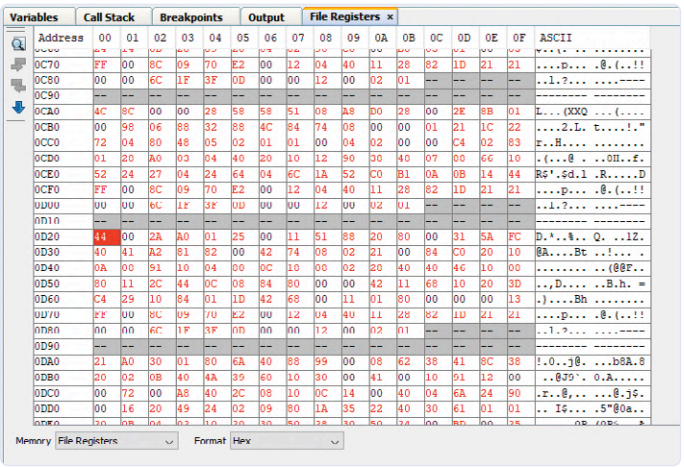


Figure 7. Ces valeurs ont une drôle d'allure.

Fermez le popup après avoir cliqué sur le bouton GoTo pour voir le résultat (**fig. 7**). La case surlignée en rouge représente le premier octet du tableau. Il est évident qu'il y a une erreur car d'autres valeurs figurant dans le tableau, comme FC, sont en dehors de la plage autorisée.

Pour enquêter, nous pouvons remplir la zone de mémoire avec un modèle de bits facilement reconnaissable. Un bon exemple serait d'écrire la valeur 11111111 dans le registre indirect :

```
MOVFW PortCache
CLRZ
ADDWF FSR0L
BTFSC STATUS, C
INCF FSR0H
MOVFW B'11111111'
MOVLW INDF0
```

En visualisant le code dans le débogueur, vous verrez une séquence des mêmes valeurs. Dans la plupart des cas, elles ne doivent pas avoir la valeur FF. Le code comporte une petite erreur. Nous pouvons utiliser l'instruction `MOVLW` pour charger la valeur d'un emplacement mémoire dans le registre W :

```
MOVFW B'11111111'
MOVLW INDF0
```

Du point de vue de MPLAB, le littéral INDF0 est un nombre : après compilation, les valeurs comme PORTA ne sont que des nombres. Notre programme copie donc l'adresse du registre dans les 32 emplacements de mémoire. Néanmoins, nous avons fait un pas de plus puisque nous avons déjà vérifié les données d'adresse. Une version corrigée du programme ressemble maintenant à ceci :

```
MOVLW B'11111111'
MOVWF INDF0
```

Le calcul de l'adresse mémoire fonctionne. Nous pouvons éradiquer l'erreur de calcul réelle. Le premier problème était d'utiliser l'instruc-

tion `MOVLW` au lieu de l'instruction `MOVWF`, ce qui rendait impossible l'écriture dans l'INDF :

```
MOVFW WorkArea
MOVWF INDF0
MOVFW PortCache
SUBLW .31
BTFSS STATUS, Z
GOTO PREP
```

L'examen des résultats révèle que les valeurs ne sont pas correctes. La cause de ce problème est que l'instruction `RRF` peut positionner le bit de retenue. Notre code n'a cependant pris en charge que le drapeau Z. Corrigeons :

```
MOVWF WorkArea
BCF STATUS, C
RRF WorkArea, 1
BCF STATUS, C
RRF WorkArea, 1
BCF STATUS, C
RRF WorkArea, 1
```

Le programme est prêt à fonctionner et le tableau des valeurs des sinusoïdes apparaît dans la fenêtre du débogueur. Pour terminer, nous devons nous assurer dans la boucle de travail que les valeurs sont extraites de la mémoire de données. Cela nécessite un incrément de la variable d'exécution afin d'obtenir un index continu :

```
WORK
   BANKSEL DAC1CON1
   MOVLW B'00000001'
   ADDWF PortCache, 1
```

L'adressage indirect convient pour la lecture et l'écriture. Nous chargeons les deux parties de l'adresse du tampon dans FSR0H et FSR0L. Ensuite, nous ajoutons le décalage et vérifions s'il y a un dépassement de capacité. Le cas échéant, nous incrémentons le registre supérieur :

```
MOVLW high DataBuffer
MOVWF FSR0H
MOVLW low DataBuffer
MOVWF FSR0L
MOVFW PortCache
CLRZ
ADDWF FSR0L
BTFSC STATUS, C
INCF FSR0H
```

Ce qui est nouveau, c'est que nous lisons le registre INDF0. La valeur est chargée dans le registre DAC1CON1 du CN/A :

```
MOVFW INDF0
MOVWF DAC1CON1
CALL WAIT
```

Der Rest des Programms ist eine gewöhnliche Schleife, die unter anderem die Inkrementierung vornimmt:

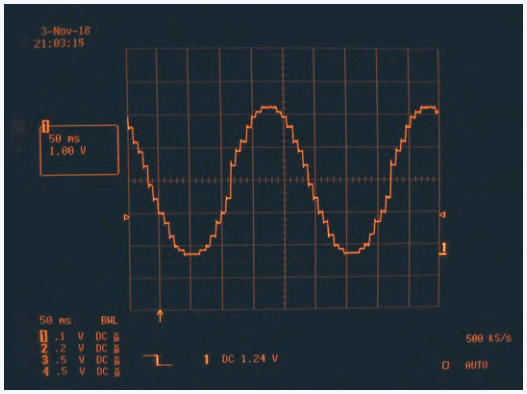


Figure 8. Le signal de sortie du CN/A correspond approximativement à un sinus de 4 Hz.

```
MOVFW PortCache
SUBLW .31
BTFSC STATUS, Z
CLRF PortCache
GOTO WORK
```

Le programme est prêt, il produit le signal de sortie sinusoïdal de la **fig. 8**.

Conclusion

Ce projet montre l'intérêt des expérimentations avec des µC à 8 bits. Vous trouverez plus d'informations sur ce sujet dans mon nouveau livre «*Microcontroller Basics with PIC*». Si vous avez apprécié cet article, faites-le moi savoir. Les commentaires constructifs sont toujours les bienvenus !

200154-03

bidouiller une lampe IKEA

Améliorer une lampe IKEA bon marché avec des LED NeoPixel et une interface WLAN

Hans Henrik Skovgaard

Les lampes multicolores télécommandées et connectées par Wi-Fi sont des luminaires attrayants. Pourquoi en construire une soi-même si on les trouve toutes faites auprès de plusieurs fournisseurs ? Parce que ça revient moins cher, ça permet de personnaliser l'objet, ça procure une satisfaction et parce que c'est instructif en plus.

Ce qui me motive le plus, c'est de trouver moi-même des solutions et de relever des défis inattendus. Mon fils m'a passé une carte NeoPixel *Jewel 7* [1] d'Adafruit rescapée d'un projet sur lequel il avait travaillé à l'université. Ce petit circuit imprimé rond à CMS (**fig. 1**) comporte sept LED RBG NeoPixel programmables. À quoi diable pourrais-je bien l'utiliser ?

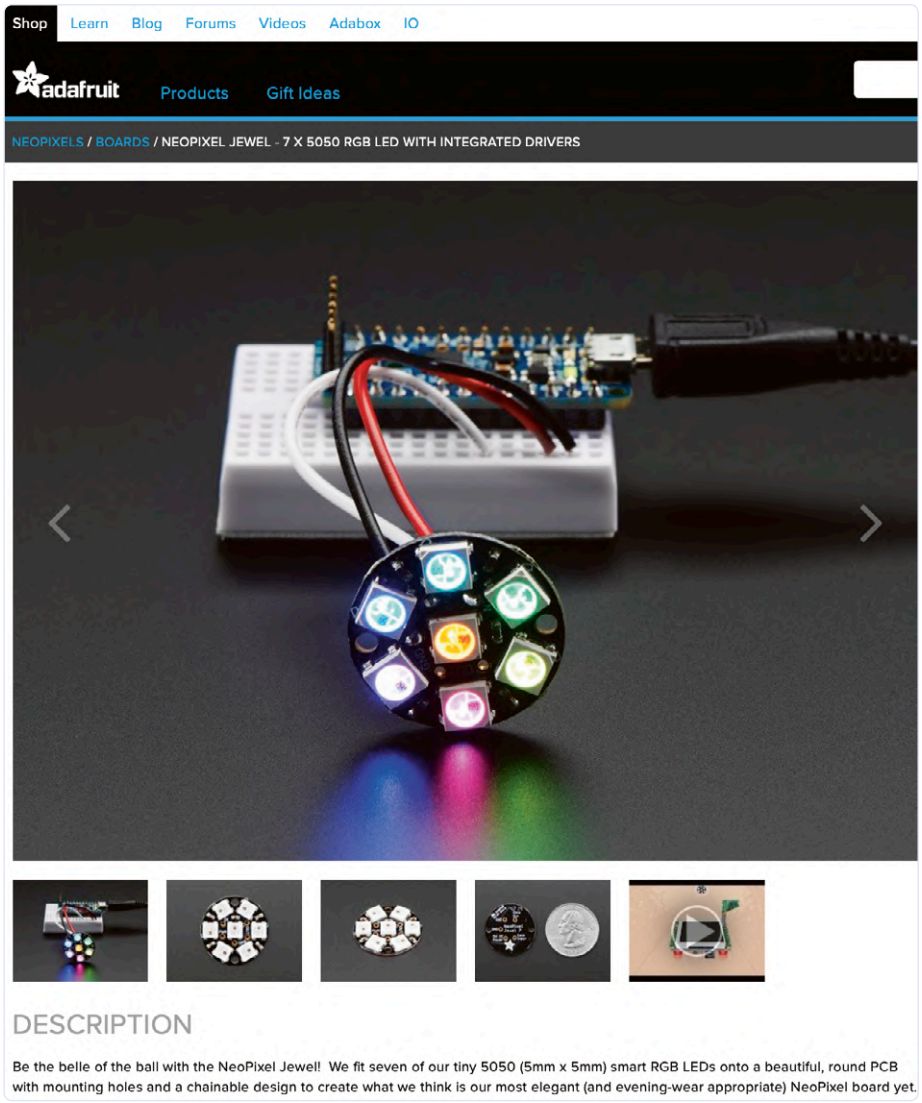


Figure 1. NeoPixel ***Jewel 7****, connectée à une carte Arduino* **[1]**.

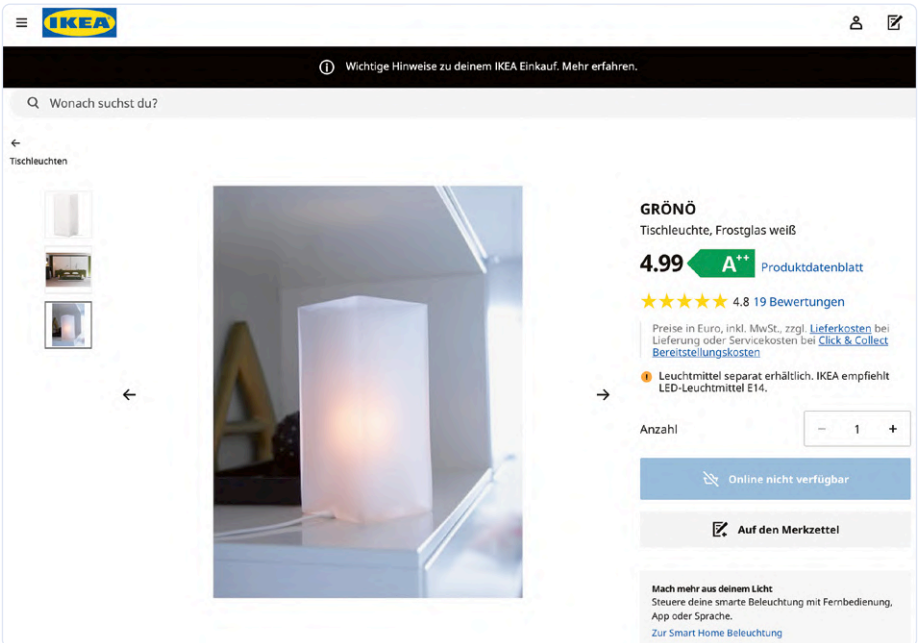


Figure 2. La lampe bon marché IKEA Grönö **[4]**.

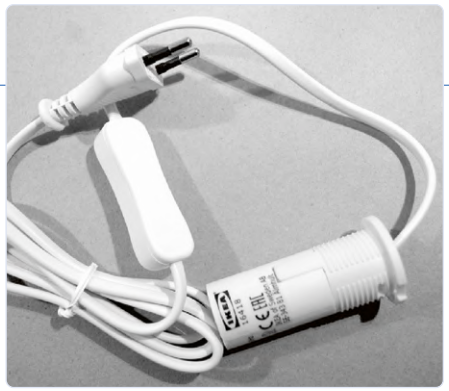


Figure 3. La lampe IKEA avant l'intervention.

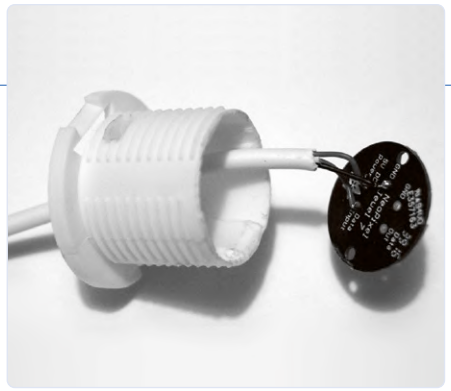


Figure 4. La carte LED fixée au sommet du support de lampe modifié.

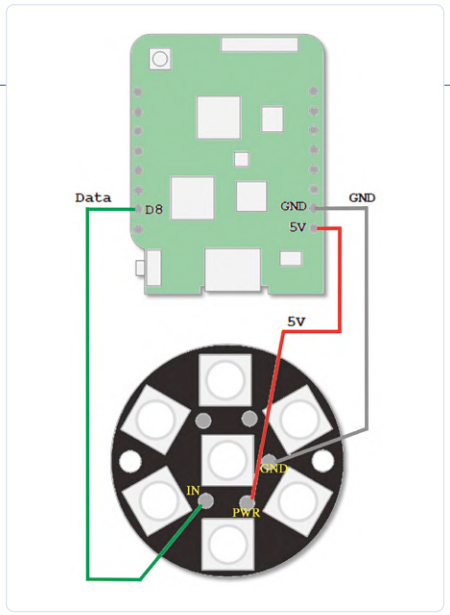


Figure 5. Le circuit de modification est très simple : carte à microcontrôleur + carte LED.

NeoPixel + Arduino

WorldSemi Co fournit depuis un certain temps déjà des LED RGB avec commande intégrée, dotées d'une interface à trois fils, qui permet de les enfiler facilement pour des applications plus ambitieuses. Chaque LED RVB peut être adressée individuellement par un seul signal de données en série. Avec les mots-clés «NeoPixel» ou «WS2812B» vos recherches sur eBay seront fructueuses. Comme j'avais déjà travaillé sur une série de projets Arduino, je savais qu'il serait facile de commander ces LED à l'aide de la bibliothèque appropriée d'Adafruit [2]. Mes premières impressions avec ces LED ont laissé sur ma rétine des traces hélas durables. À pleine puissance, leur luminosité est si violente qu'il faut éviter de les regarder directement !

Les cartes Arduino sont une bonne solution pour réaliser une commande à usage général et même pour des installations expérimentales, sauf s'il vous faut l'internet. Dans ce cas, il faut une autre solution. Comme je travaillais déjà par ailleurs avec un ESP8266 d'Espressif sous la forme du «D1 mini Pro» de WeMos [3], j'ai pensé que cette carte, associée à la carte LED NeoPixel, ferait un bon tandem. Il reste à bien intégrer ce projet dans un environnement domestique, plus précisément dans un boîtier approprié.

Modification de la lampe

Je propose de modifier une lampe standard IKEA Grönö [4] qui ne coûte que 6 € (**fig. 2**), ce qui en fait le cobaye idéal pour expérimenter : si ça ne marche pas, la perte est limitée. On voit l'intérieur de la lampe sur la **fig. 3** : un culot E14 et un cordon d'alimentation avec un interrupteur. Pour connecter l'ESP8266 à la carte NeoPixel, trois fils suffiront. La base du support de la lampe peut être modifiée et le cordon d'alimentation mis au rebut. La carte est alors fixée sur la base du support de lampe découpé avec de la colle. Enfin, le cordon d'alimentation est remplacé par trois fils courts (**fig. 4**).

Même Ikea pourrait difficilement faire plus simple que le circuit de commande de la lampe (**fig. 5**). En plus de la carte Neopixel et du ESP8266, il faut pour cette lampe une alimentation secteur de 5 V, 500 mA avec fiche micro USB. Le reste, c'est du logiciel. La carte Neopixel est équipée de sept LED RGB, avec chacune son propre µC intégré au format SMD 5050. Vous pouvez acheter ces LED à la pièce sous la référence WS2812B [5] pour réaliser d'autres arrangements plus complexes, en chaîne ou en matrice. Chaque LED RGB demande jusqu'à 60 mA à pleine puissance, et délivrer en échange un flux lumineux maximum de 20 lm. Vous pouvez les brancher en série et les adresser individuellement en injectant à une extrémité de la chaîne les données de luminosité RVB en série. Cela facilite la commande.

Logiciels

Une caractéristique importante de la carte mini Pro WeMos D1 est le support dans l'IDE Arduino. Ce dernier a ses avantages et ses inconvénients, mais il a le mérite de la rapidité de la mise en place aisée d'un projet et sans trop d'efforts. Avant de commencer à utiliser le logiciel pour s'attaquer à ce bidouillage IKEA, il y a les préliminaires suivants :

- installer l'IDE Arduino [6]
- inclure dans l'IDE le soutien au µC ESP8266 [7]
- installer la bibliothèque Adafruit-NeoPixel dans l'IDE Arduino [2]
- installer le téléchargeur de système de fichiers Arduino ESP8266 (pour le flash SPI) [8]

Le logiciel [12] paraît bien complexe pour la simple de la tâche de commande des LED, mais la raison de cette complexité est qu'un ESP8266 a un support Wi-Fi intégré. Si nous voulons l'utiliser, nous aurons besoin d'un code. Le logiciel offre les possibilités suivantes :

- création d'effets de lumière spéciaux

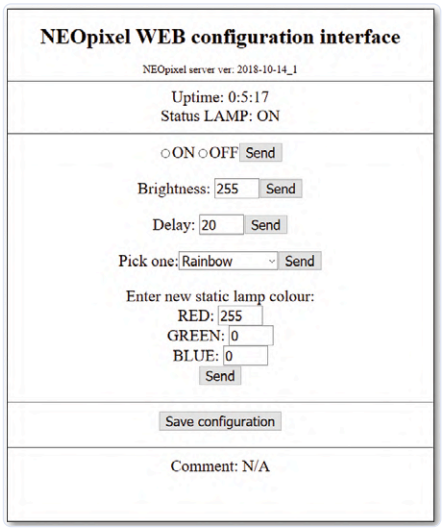


Figure 6. Interface de configuration simple du serveur web.

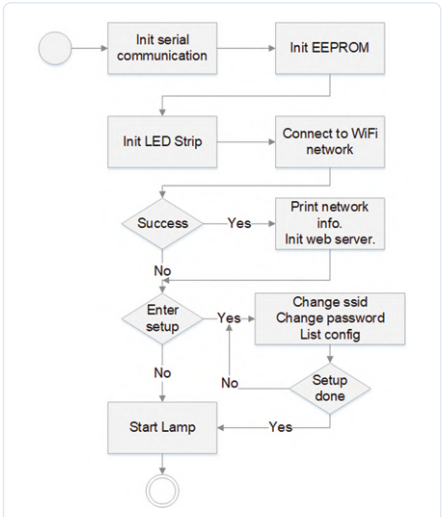


Figure 7. Schéma du processus de configuration.

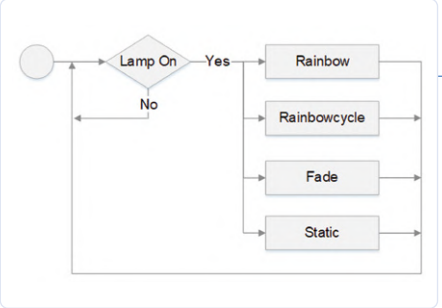


Figure 8. Ordinogramme du choix du mode de fonctionnement.

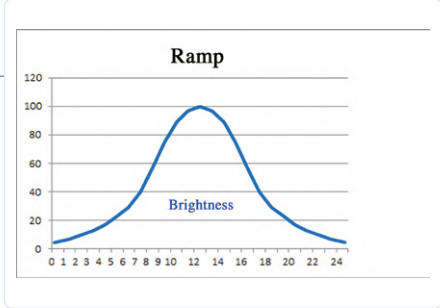


Figure 9. Courbe de luminosité optimisée pour la gradation.

- connexion à l'internet par Wi-Fi
- serveur web pour la configuration des lampes

La lampe peut également être utilisée hors ligne ou en mode autonome. Dans ce cas, elle parcourt les couleurs de l'arc-en-ciel à un rythme prédéfini (p. ex. 20 ms pour chacun des niveaux de couleur 8x8x8). Si vous souhaitez configurer la lampe en fonctionnement, vous devez activer la liaison Wi-Fi pour communiquer avec la lampe Neopixel via votre réseau local. Les paramètres nécessaires pour le Wi-Fi peuvent être codés en dur dans le code source. Cela signifie que toute modification des paramètres du réseau nécessite une recompilation du micrologiciel WeMos. Sinon, ils peuvent être définis au démarrage de la lampe. Une fois la lampe connectée au réseau Wi-Fi, les opérations suivantes sont possibles :

- allumer et étendre la lampe
- commander la luminosité
- modifier le délai entre les cycles de couleur
- sélectionner les cycles de couleurs (arc-en-ciel, cycles arc-en-ciel, statique, scintillement de la bougie et décoloration).
- choisir une couleur fixe.

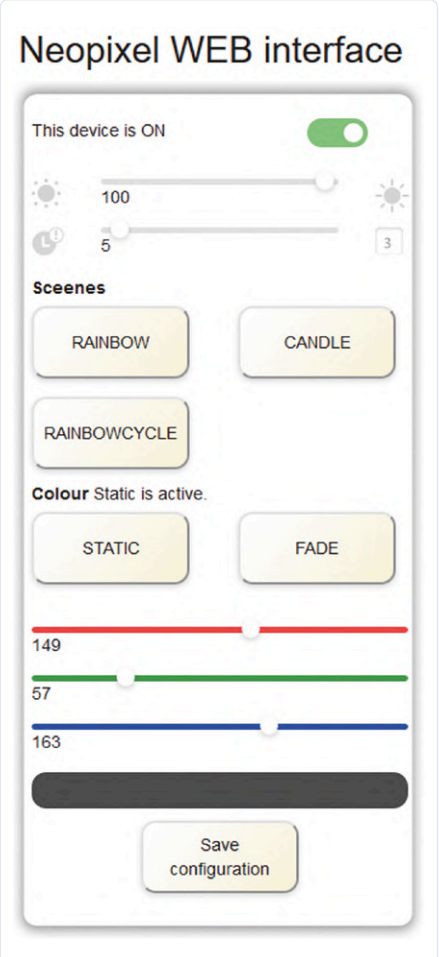


Figure 10. Une interface web plus fluide.

Ces opérations peuvent être effectuées via une interface web très simple (**fig. 6**). Le code HTML de la page web est obtenu à l'aide de la fonction `getPage(string str).` Tout cela est géré dans le logiciel et, de ce fait, il n'est pas si facile d'y apporter des modifications. Cependant, il fonctionne suffisamment bien pour nos besoins. L'interface de configuration web met à jour la lampe NeoPixel via une requête HTTP POST. La procédure de base est décrite dans mon livre [9]. Le logiciel est structuré à peu près de la même manière que pour d'autres de mes projets similaires. La **fig. 7** montre l'ordinogramme du logiciel pendant la configuration et la **fig. 8** montre la même chose pour le fonctionnement ou la sélection du mode.

J'ai rapidement découvert que la fonction de fondu (augmentation et diminution de la luminosité) nécessite un peu plus de considération. Si vous commandez la gradation de la lampe au moyen de valeurs qui correspondent à une fonction de rampe linéaire, le changement de luminosité n'est pas satisfaisant. Comme le montre la **fig. 9**, il est préférable d'utiliser une fonction sinusoïdale pour produire un changement de luminosité plus doux. Cette courbe de luminosité a été simulée dans Excel et les valeurs ont été implémentées sous la forme d'un tableau (*array*) `byte fadeInterpolation[]`. Ces valeurs peuvent bien sûr être modifiées. L'article sur les gradateurs à LED dans Elektor de septembre 2018 [10] décrit cet effet. La relation entre le changement réel d'un stimulus physique et le changement perçu est une propriété psychophysique régie par la loi de Weber-Fechner [11].

Utilisation de la lampe

Dès que la carte WeMos programmée est mise sous tension, le logiciel démarre et entre dans son cycle de mise en marche.

Connexion Wi-Fi (bleu)

Dans cette phase, une connexion avec le réseau Wi-Fi est tentée en utilisant les paramètres de réseau programmés dans le logiciel. Les LED clignotent en bleu l'une après l'autre.

Attendre (rouge)

Après avoir essayé de se connecter (avec ou sans succès) par le Wi-Fi, la carte recherchera l'interaction de l'utilisateur via le port USB. Pendant cette période, les LED clignotent en rouge l'une après l'autre. Lorsqu'une interaction est détectée, toutes les LED s'éteignent. Si après 10 s aucune interaction n'est détectée, la lampe commence à fonctionner selon son schéma d'éclairage préconfiguré. Après l'allumage et avec une connexion Wi-Fi active, il est possible de commander l'effet d'éclairage ou le mode de fonctionnement de la lampe Neopixel.

Une interface web différente

Mon fils, titulaire d'une maîtrise en design interactif, n'a pas été impressionné du tout par mon interface web. Il a donc retroussé ses manches (**fig. 10**), et pondu sa propre version combinant HTML, CSS et Javascript. C'est là que le système de fichiers flash SPI entre en jeu. Le µC ESP8266 fournit au moins 14 Mo de mémoire flash utilisables via le système de fichiers SPIFFS. Dans mon livre [9], vous trouverez des informations à ce sujet :

- téléchargement de fichiers vers SPIFFS
- stockage de fichiers dans un PC pour téléchargement
- installation des logiciels nécessaires à l'EDI Arduino

Si vous l'avez installé correctement, votre répertoire de logiciels Arduino devrait ressembler à la **fig. 11**. La **figure 12** montre les fichiers pour l'interface contenus dans le dossier *data.* La dernière version du logiciel peut être téléchargée gratuitement sur le site d'Elektor [13]. Vous remarquerez que le fichier *NEOPixel_new_20191222_load.js n'*est pas répertorié dans le dossier *data.* Ces fonctions sont codées directement dans le logiciel ESP8266. Cela permet au µC ESP8266 de configurer l'interface lors de la mise sous tension en utilisant les paramètres stockés dans l'EEPROM.

L'interface web a été conçue pour l'orientation portrait des écrans de téléphones tactiles. Comme la lampe NeoPixel peut être configurée par des requêtes HTTP POST, il est également possible de commander la lampe avec la domotique OpenHAB, mais c'est une autre histoire.

Travailler avec la carte NeoPixel *Jewel* m'a amené à explorer de nombreux domaines différents que je ne connaissais pas avant le début du projet. Le résultat est satisfaisant ; ce luminaire IKEA, avec sa carte NeoPixel, illumine mon salon depuis maintenant plus d'un an.

200165-03

Name	Date modified	Type	Size
data	25/12/2019 22.54	File folder	
New_IKEAHack_20191221_1.ino	08/02/2020 21.38	Arduino file	48 KB

Figure 11. Le répertoire des logiciels Arduino.

Name	Date modified	Type	Size
brightnessHigh.svg	19/12/2019 15.39	SVG Document	3 KB
brightnessLow.svg	19/12/2019 15.40	SVG Document	5 KB
delayLeft.svg	20/12/2019 22.16	SVG Document	2 KB
delayRight.svg	20/12/2019 22.17	SVG Document	7 KB
main_background.svg	25/12/2019 22.07	SVG Document	13 KB
NEOPixel_new_20191222.css	25/12/2019 22.52	Cascading Style Sheet Doc	2 KB
NEOPixel_new_20191222.html	18/01/2020 22.32	Firefox HTML Document	8 KB
NEOPixel_new_20191222.js	26/12/2019 23.29	JavaScript File	4 KB
sliderCircle.svg	18/12/2019 20.19	SVG Document	2 KB

Figure 11. Le répertoire des données.

WEBLINKS

[1] **NeoPixel Jewel 7** : www.adafruit.com/product/2226

[2] **Guide de la bibliothèque NeoPixel** : https://learn.adafruit.com/adafruit-neopixel-uberguide/arduino-library-use

[3] **Wemos D1 mini Pro** : hwww.wemos.cc/en/latest/d1/d1_mini_pro.html

[4] **Lampe de table d'Ikea, modèle Grönö** : www.ikea.com/fr/fr/p/groenoe-lampe-de-table-verre-givre-blanc-20373225/

[5] **Fiche technique WS2812B** : www.world-semi.com/DownLoadFile/111

[6] **EDI Arduino**: www.arduino.cc/en/Main/Software

[7] **Support ESP8266** : https://arduino.esp8266.com/stable/package_esp8266com_index.json

[8] **Téléchargeur pour le système de fichiers** : https://github.com/esp8266/arduino-esp8266fs-plugin#arduino-esp8266-filesystem-uploader

[9] **Mon livre « IoT Home Hacks with ESP8266 »** : www.elektor.de/iot-home-hacks-with-esp8266-e-book

[10] **Article Elektor « variateurs pour LED (1) »** : www.elektormagazine.de/magazine/elektor-59/41891/

[11] **Loi de Weber-Fechner** : https://de.wikipedia.org/wiki/Weber-Fechner-Gesetz

[12] **Logiciel** : www.elektor.de/amfile/file/download/file/2134/product/9476/

[13] **Version à jour du logiciel** : www.elektormagazine.fr/200165-01

N'abandonnez plus vos projets, gérez-les avec rigueur et plaisir

Gestion du temps disponible et conception en spirale

Tessel Renzenbrink (Pays-Bas)

Comme moi, tu connais forcément ce cycle de calamités : d'abord tu te lances dans un projet avec enthousiasme, il avance bien, tu commandes les composants, et le projet trône sur ta paillasse pendant des jours, des semaines, des mois… puis c'est la débandade. Un jour, tu te rends compte de la couche de poussière et, dépité, tu mets tout ça de côté. L'hiver passe, l'été aussi, tu soupires et finis par glisser ton projet dans un tiroir où il rejoint dix autres projets orphelins.
Une façon d'augmenter nos chances de réussite et de renforcer le plaisir que nous procurent nos projets, en combinant deux méthodes : la *gestion du temps disponible (ou supply-time management)* et la *conception en spirale (ou spiral development)*.

Gestion du temps disponible

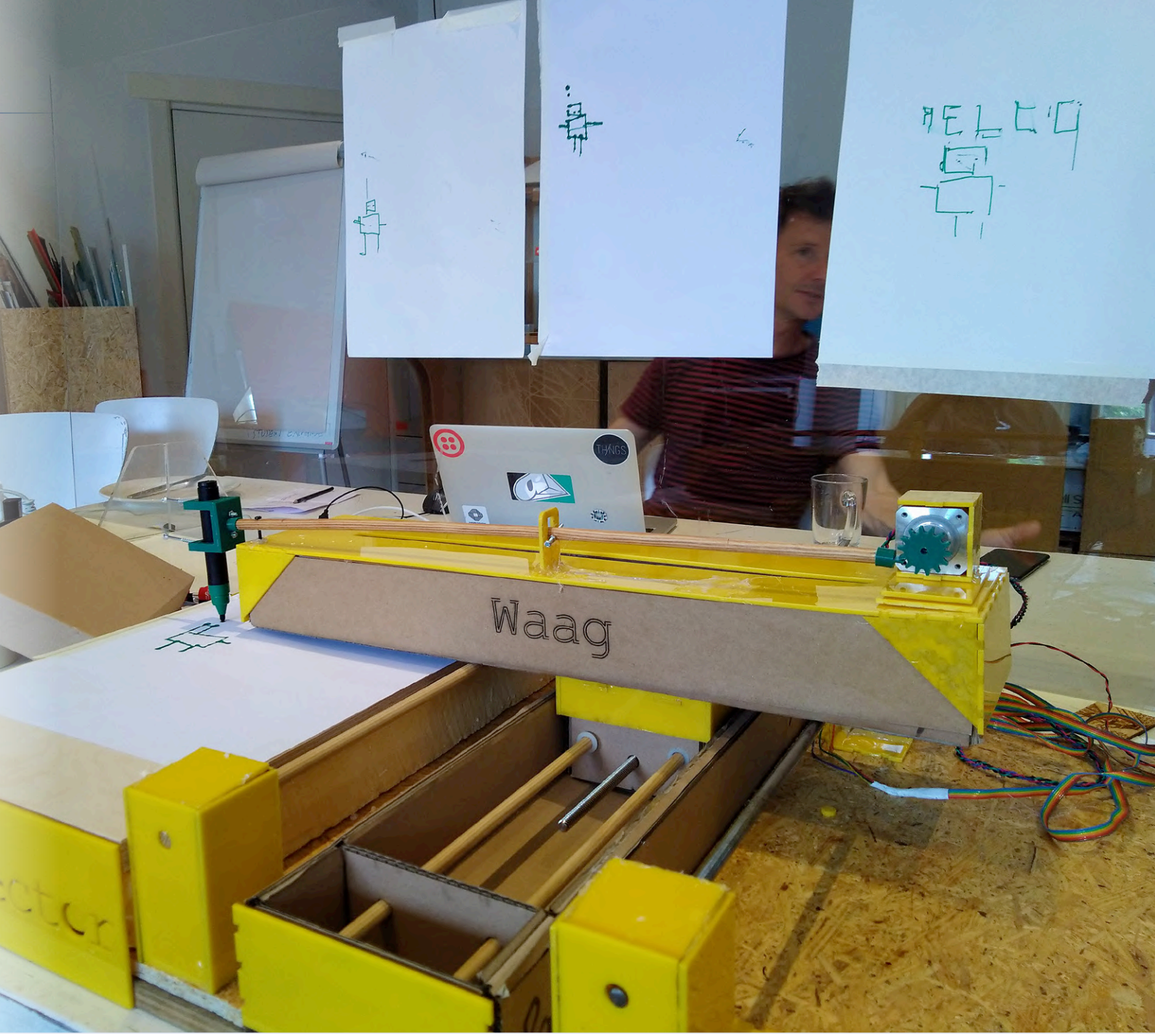
Lorsque tu évalues le temps requis pour un projet, tu commences en fait par *le problème* : tu décides de quelque chose de grand et d'ambitieux avant d'envisager le temps que ce *fantasme* prendra. Une première estimation te fait espérer que 7 ou 8 dimanches devraient suffire. Bien, ce sera terminé dans deux mois. Dès la 2e semaine, la météo enfin fantastique te convainc de faire cette promenade en forêt longtemps reportée. Pas de problème, tu avais une semaine de marge et le délai initial reste tenable.
Au cours de semaine 4, tu tombes sur un os. Ta carte refuse de fonctionner. Même après une journée de débogage, ça coince. L'enthousiasme flétrit. À la semaine 5, plus aucune motivation. Aucun résultat tangible. Tu n'y crois plus. Ton projet est sur le chemin qui mène aux oubliettes.
Cet exemple illustre la gestion du temps requis : tu laisses le projet déterminer le temps qu'il prendra et tu grappilles désespérément les heures disponibles pour y arriver. L'inverse est la gestion de l'offre : tu détermines le temps disponible et tu dimensionnes ton projet en conséquence. Si tu sais que tu auras du temps pour l'électronique dimanche prochain, tu te mets au travail sur un projet réalisable en un dimanche que tu divises en épisodes, un pour chaque phase du projet. Deux heures pour la conception, trois heures pour le prototypage, etc. Tu t'en tiens à ce plan et évites ainsi de passer tout le dimanche à la conception à essayer de régler un petit détail… qui te paralyse.

Conception en spirale

Planifier en fonction du temps disponible n'implique pas de ne s'attaquer qu'à des projets de courte durée. C'est là qu'entre en jeu la conception en spirale, qui divise un projet en cercles qui s'appuieront les uns sur les autres. Chaque cercle est une tâche achevée, de la conception au prototype fonctionnel. Chaque cercle commence là où le précédent s'est terminé. Une spirale se forme ainsi, dans laquelle le projet se complexifie avec chaque cercle ajouté.
La conception en spirale permet d'atteindre l'objectif beaucoup plus fréquemment que la conception en série. Celle-ci ne procure de sentiment de satisfaction qu'une fois l'ensemble du projet terminé. Grâce à la spirale, après chaque cercle, tu as quelque chose de tangible entre les mains. Et, si tu combines spirale et gestion du temps disponible, tu auras quelque chose à fêter chaque dimanche.
Un deuxième avantage de cette méthode est de vérifier, à la fin de chaque cercle, que le projet fonctionne bel et bien. De cette façon, on ne risque pas d'arriver à la fin d'un projet de longue haleine pour constater que ça ne fonctionne pas. De plus, le débogage d'un petit (morceau de) projet est forcément beaucoup plus facile que celui d'un projet complet et complexe.

Genèse d'une machine en sept jours

Mes camarades de classe et moi avons utilisé ces méthodes pour construire une machine en une semaine dans le cadre d'un cours de la *Fab Academy* où les processus de fabrication assistés par ordinateur, tels que l'impression en 3D, et la production électronique sont enseignés à un rythme effréné. Pour ce projet de groupe, nous avons construit une machine à dessiner télécommandée. Notre camarade, étudiant en France, peut maintenant faire des dessins (de chats et de robots) qui apparaissent sur le papier dans notre laboratoire d'Amsterdam. Pour y parvenir en sept jours, nous avons utilisé la gestion de temps disponible et la conception en spirale. L'impact positif du chronométrage a été perçu dès le début. Lorsque cinq personnes travaillent sur un projet, on risque de perdre pas mal de temps à délibérer sur ce qu'il faut faire ou sur les exigences à respecter. Grâce au tic-tac incessant de l'horloge, l'accord a



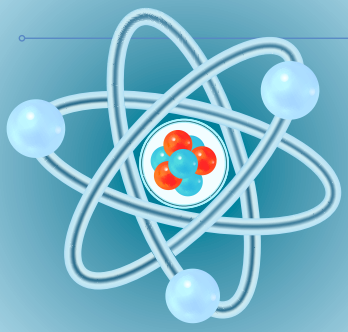
La machine à dessiner est commandée via l'internet (photo : T. Renzenbrink).

été trouvé rapidement. L'objectif de la première spirale était de commander localement les moteurs pas à pas avec le moniteur série de l'EDI Arduino, et de construire un boîtier simple pour piloter les axes X et Y.

Cette première phase a pris plus de temps que prévu. Pour respecter le temps imparti, nous avons procédé au *triage* qui consiste à fixer des priorités. Le procédé est courant dans les services d'urgence saturés quand la capacité de traitement est réduite. Les patients sont répartis en trois catégories : *non traitable*, *traitement immédiat* et *peut attendre un peu*. Nous avons fait de même avec notre projet. Notre patient « mortellement blessé » a été l'ajout d'interrupteurs de fin de course pour les moteurs pas à pas. La tâche semblait simple mais, après quelques heures de forums et d'avance rapide sur les tutoriels Youtube, nous étions au point… mort. La compétence consiste alors à ne pas s'accrocher, mais à lâcher prise. Il suffit de choisir : laisser cette exigence de côté, ou la pousser vers un cercle ultérieur de la spirale.

Le nombre de raccourcis et de solutions *créatives* a augmenté à mesure que la ligne d'arrivée se profilait : au début, nos assemblages étaient encore très soignés, vers la fin tout se faisait au pistolet à colle. La phase finale d'une spirale est excitante parce qu'on y accomplit beaucoup en peu de temps. La gestion du temps disponible et la conception en spirale rendent les projets beaucoup plus intéressants parce qu'ils vous incitent à continuer. C'est moins frustrant, puisque vous n'avez pas le temps de vous attarder sur quelque chose qui ne réussira pas. Cela n'implique pas qu'il faille bâcler votre projet, mais cela vous aide à établir vos priorités et à les respecter. Pour notre objectif global, qui est de faire la démonstration en classe d'une machine qui fonctionne, les interrupteurs de fin de course n'étaient pas essentiels. Si en revanche vous mettez la machine à dessiner à la disposition d'autres utilisateurs, il serait préférable de les inclure. Dans ce cas, ce sera l'objectif d'une spirale ultérieure.

200300-04



démarrer en électronique (4)

...est moins difficile qu'on ne l'imagine !

Eric Bogers (Elektor Pays-Bas)

Où en étions-nous avant que la pandémie ne chamboule tout et que nous nous mobilisions pour préparer l'édition spéciale pour l'été ? Ah oui, les résistances montées en série et en parallèle. Nous avons terminé l'épisode du numéro de mai par un casse-tête : le circuit en H.

Cet épisode est consacré à la transformation étoile-triangle ou Y-Δ, et si on part en sens inverse à la transformation triangle-étoile (**fig. 1**). Il s'agit de convertir la connexion en étoile (en haut) en connexion en triangle (en bas) de telle manière qu'aux points 1, 2 et 3 les deux se comportent de manière absolument identique pour le monde extérieur.
Pour la conversion de l'étoile en triangle s'appliquent les formules ci-dessous (dont nous nous contenterons sans les expliquer, Elektor est une revue d'électronique et non de mathématiques) :

$$R12 = \frac{R10 \cdot R20}{R30} + R10 + R20$$

$$R23 = \frac{R20 \cdot R30}{R10} + R20 + R30$$

$$R13 = \frac{R10 \cdot R30}{R20} + R10 + R30$$

Et pour la conversion dans l'autre sens :

$$R10 = \frac{R12 \cdot R13}{R12 + R23 + R13}$$

$$R20 = \frac{R12 \cdot R23}{R12 + R23 + R13}$$

$$R30 = \frac{R23 \cdot R13}{R12 + R23 + R13}$$

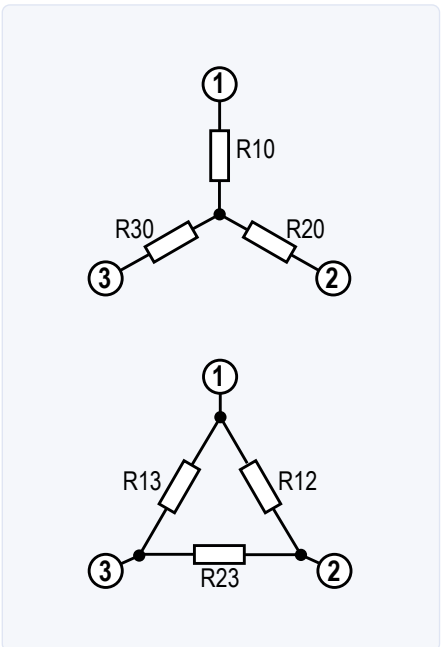


Figure 1. La transformation étoile-triangle.

Attention aux désignations des différentes résistances !
Nous pouvons maintenant nous occuper du circuit en H (**fig. 2**). Regardez bien et vous verrez un circuit en triangle avec deux résistances supplémentaires (R4 et R5) à la base. Nous soumettons le triangle de R13, R12 et R23 à la transformation. Pour simplifier, les valeurs de résistance seront : R13 = 10 Ω, R12 = 20 Ω, R23 = 30 Ω, R4 = 40 Ω et R5 = 50 Ω. Avec la transformation, on obtient :

$$R10 = \frac{R12 \cdot R13}{R13 + R12 + R23} = \frac{200}{60}\Omega = 3{,}3\Omega$$

$$R20 = \frac{R12 \cdot R23}{R13 + R12 + R23} = \frac{600}{60}\Omega = 10\Omega$$

$$R30 = \frac{R23 \cdot R13}{R13 + R12 + R23} = \frac{300}{60}\Omega = 5\Omega$$

Ce qui reste est une simple combinaison de connexions en série et en parallèle ; nous vous laissons le soin de continuer à calculer. Le résultat est 29,05 Ω. Ça colle ?

En règle générale

Souvent, on peut faire l'économie d'une grande partie de ces calculs en examinant deux situations extrêmes et en réfléchissant un peu. Dans notre circuit en H de la fig. 2, nous enlevons la résistance R23 sans la remplacer. La résistance totale du réseau augmenter forcément, car, souvenez-vous que si nous augmentons une résistance dans un réseau ohmique, la résistance totale ne diminue jamais. Or ici nous augmentons R23 à l'infini.
Le calcul de la résistance totale du réseau est facile ; avec les mêmes valeurs de résistance que dans l'exemple, vous obtenez 29,16 Ω.
Remplaçons maintenant R23 par l'autre extrême : un pont qui réduit la résistance à 0 Ω. Vous n'avez pas oublié que si nous réduisons une résistance dans un réseau de résistances ohmiques, la résistance totale de ce réseau n'augmente jamais. Là encore, le calcul est simple, le résultat est de 28,8 Ω.
Quelle que soit la valeur réelle de R23, la résistance totale équivalente de notre circuit en H doit se situer entre ces deux valeurs extrêmes. Et si, comme dans cet exemple, ces valeurs ne diffèrent pas de beaucoup plus qu'environ 1 %, nous pouvons en pratique nous contenter de la moyenne des deux valeurs extrêmes. Les fans de décimales

s'amuseront à les calculer, en vain car l'erreur résultante est de toute façon offusquée par les tolérances des composants.

Dans certains cas, il faut faire le calcul car l'écart peut être tel que l'on ne peut pas se contenter du pifomètre pour la résistance totale !

Résistance des lignes

Avant d'en arriver (enfin !) au courant alternatif, nous devons parler brièvement de la résistance des lignes. Jusqu'à présent, nous avons négligé de prendre en compte la résistance des câbles de connexion, alors que tout câble et toute piste de circuit imprimé présente une certaine résistance, qui ne doit en aucun cas être négligée !
Pour la résistance d'un câble :

$$R = \frac{l \cdot \rho}{A} = \frac{l}{A \cdot \gamma}$$

Ici, ρ (*rho* en grec) est la résistivité et γ (*gamma* en grec) est la conductivité spécifique. Ce sont deux constantes matérielles. Avec les conducteurs courants (cuivre, aluminium, argent), il est un peu plus facile de calculer la conductivité spécifique.

Résistance spécifique et conductivité

matériau	ρ (Ω·mm²/m)	γ (m/Ω·mm²)
cuivre	0,017	56
aluminium	0,0287	34,8
argent	0,016	62

La formule pour le conducteur spécifique contient deux variables supplémentaires : la longueur *l du* conducteur et sa section *A*. Calculons cela pour un enrouleur de câble de 50 m avec une section de 1,5 mm². Le câble a deux fils, il faut donc compter le double de la longueur :

$$R = \frac{l}{A \cdot \gamma} = \frac{2 \cdot 50m}{1{,}5mm^2 \cdot 56m/\Omega^2} = 1{,}19\Omega$$

Faisons passer un courant de 16 A par ce câble. La chute de tension sur ce câble sera de :

$U = R * I$ = 1,19 Ω * 16 A = 19 V

Et ce n'est pas fini... La résistivité d'un conducteur comme le cuivre varie selon sa température : la résistivité augmente avec la température. Ce qui donne :

$$\rho_T = \rho_{20}\left(1+\alpha\left(T-20°\right)\right)$$

Ici α (*alpha* en grec) est le coefficient de température, qui pour le cuivre est de 0,0038, et *T* est la température en °C La valeur de ρ est toujours donnée à 20 °C.

Chauffe Marcel...

Supposons que nous utilisions notre câble de rallonge par une chaude journée d'été. L'isolation devient agréable et chaude au soleil, et un courant de 16 A n'est pas rien non plus. Supposons que le conducteur en cuivre atteigne une température de 60°C. Si nous calculons cela, nous obtenons une perte de tension d'au moins 23,3 V (vous en doutez ? Eh bien faites le calcul !).
À température ambiante normale, le câble dissipe déjà une quantité considérable d'énergie convertie en chaleur :

$P_{dissipée} = I2 - R$ = (16 A)² – 1,19 Ω = 304,6 W

C'est beaucoup, et le signe positif du coefficient de température a ici un effet négatif : un câble fortement chargé devient chaud. Par conséquent sa résistivité augmente, ce qui augmente la résistance électrique, ce qui augmente la dissipation de puissance et fait chauffer encore plus le câble et ainsi de suite : une surchauffe qui finit par constituer un risque d'incendie. D'où l'importance d'un geste souvent oublier : toujours dérouler les câbles sur tambour lorsqu'ils sont utilisés. Vous voici prévenus !

200320-03

La série d'articles *démarrer en électronique* est basée sur le livre *Basic Electronics for Beginners* de Michael Ebner, publié par Elektor.

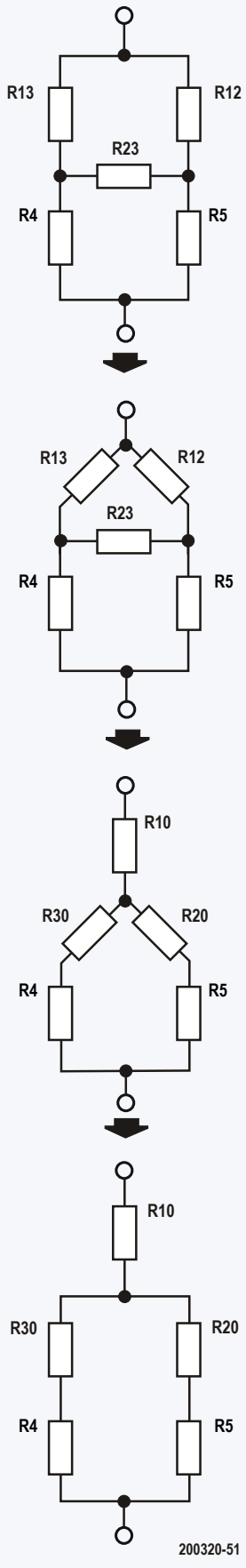


Figure 2. Voici comment traiter un circuit en H.

OHM SUITE OHM

machine de phonogravure faite maison

Quentin Therond (France), **Eric Bogers** (Pays-Bas)

À l'ère de l'audio numérique dématérialisé et de la quasi disparition du support CD, le disque vinyle analogique n'est pas mort, au contraire. Le nombre d'amateurs de platines tourne-disques et même de phonographes augmente sans cesse. Voici plus étonnant encore : le véritable aficionado n'achète pas seulement des vinyles pour les écouter, mais il grave ses vinyles lui-même !

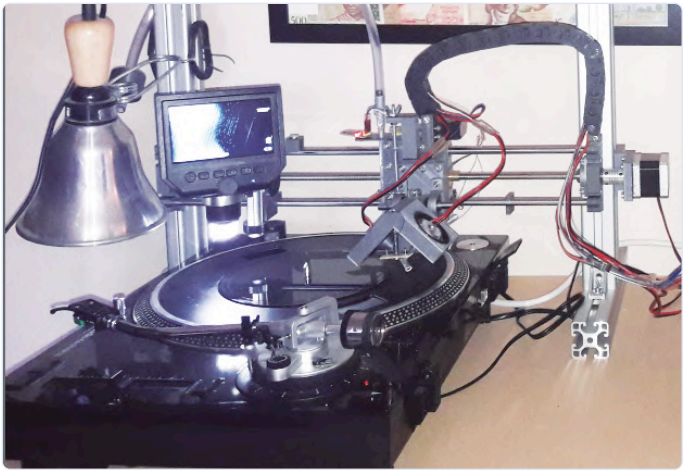


Figure 1. Au cœur du dispositif se trouve une platine de qualité.

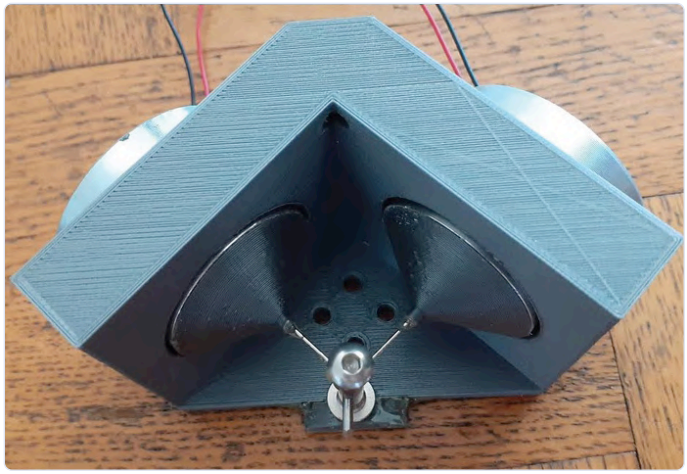


Figure 2. Tête pour le gaufrage

Dans une scène du road-movie *O Brother* des frères Coen apparaît un enregistreur Presto qui selon les connaisseurs est une merveille pour enregistrer de la musique (ou de la parole) directement sur un disque de phonographe ; c'est ainsi que Quentin à découvert la gravure analogique directe sur disque vinyle.
La figure 1 montre l'installation réalisée par Quentin pour graver lui-même des disques audio. Voici ce qu'il en dit en bref :

« Le but est de graver le signal audio sous la forme d'un sillon dans une matrice. Cela se fait en temps réel : un burin (ou aiguille) est relié mécaniquement à un haut-parleur ou à un transducteur acoustique qui reproduit le signal à enregistrer. Les vibrations du diaphragme sont inscrites par le burin dans la matrice ; le sillon est donc une reproduction théoriquement fidèle au signal original ».
« Auparavant, la courbe du spectre audio du signal doit être corrigée. Au cours de l'enregistrement, les fréquences hautes doivent être accentuées tandis que les basses fréquences sont atténuées. Lors de la lecture, c'est le contraire qui se produit. C'est la caractéristique bien connue de la RIAA [4], où la préaccentuation des aigus et l'atténuation des graves préviennent la déformation du signal analogique par la gravure sur disque ».

Presser ou couper ?

« Il y a deux façons de produire le disque (matrice) : gaufrage et gravure. Avec la première méthode, pour y enregistrer le signal, la matrice est comprimée en utilisant un burin extrêmement dur. Avec la seconde méthode, le sillon est taillé dans une couche de laque avec un burin en saphir ou en diamant. Ce procédé, plus cher, permet d'obtenir la meilleure qualité sonore ».

La figure 2 montre la tête de gaufrage en creux construite par Quentin (avec transducteurs et stylet), et la **figure 3** une tête de gravure construite aussi par lui, avec des haut-parleurs et un burin en diamant.

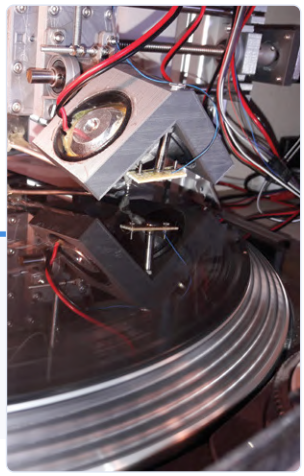
Mono ou stéréo ?

« Ma tête de gravure est construite pour la stéréo : deux haut-parleurs disposés à un angle de 90°. Le burin en diamant y est fixé. Ainsi, la déviation horizontale du sillon correspond à la somme des canaux gauche et droit tandis que la variation de la profondeur du sillon correspond à la différence de ces deux signaux ».

Combien de tours ?

« Selon la vitesse de rotation du disque (33 ou 45 tr/min), la translation de la tête de gravure de l'extérieur vers l'intérieur sera plus ou moins rapide. Ce déplacement devrait être d'environ 0,1 mm par tour pour une bonne qualité. Un disque de 30 cm à 33 tours par minute permet d'enregistrer environ 20 minutes par côté. Pour une commande fine du déplacement de la tête de gravure, il faut un chariot de précision (**fig. 4**) entraîné par un moteur aussi silencieux que possible. J'ai moi-même utilisé un moteur pas-à-pas qui vibre encore trop. Il reste des possibilités d'amélioration ».

200321-03



LIENS

[1] **machine à cocktail connectée**: http://www.elektormagazine.fr/180076-01
[2] **machine à graver toute faite**: https://phonocut.com/
[3] **vidéo de démonstration de l'auteur**: https://youtu.be/a7VAe0HyO1g
[4] **la courbe de correction RIAA**: https://fr.wikipedia.org/wiki/%C3%89galisation_RIAA

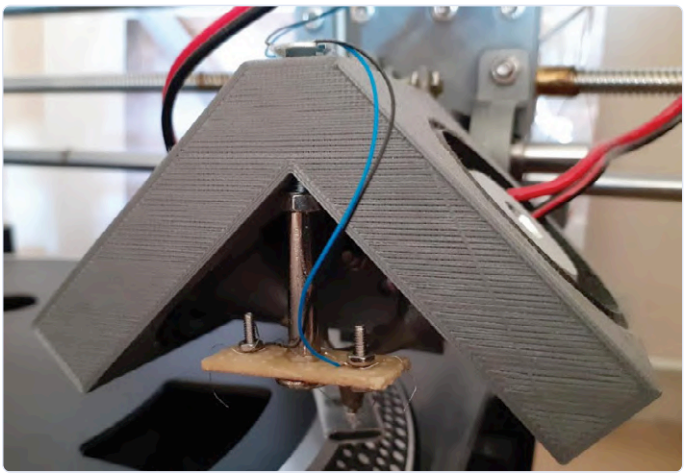


Figure 3. Tête de gravure faite maison.

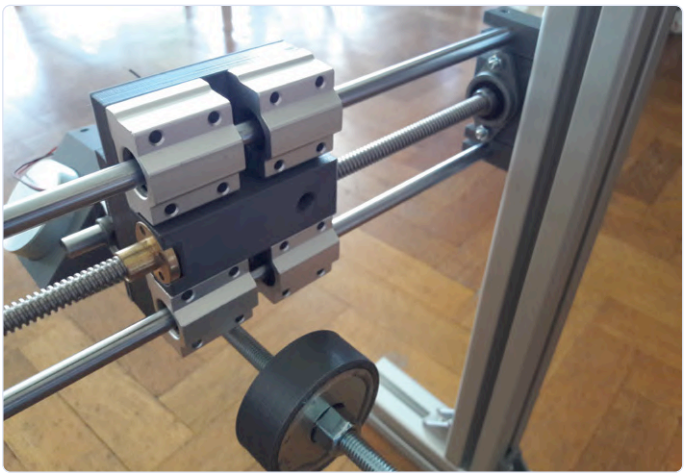


Figure 4. Mécanique de translation de la tête de gravure.

8 bits et au-delà

Entretien avec Tam Hanna

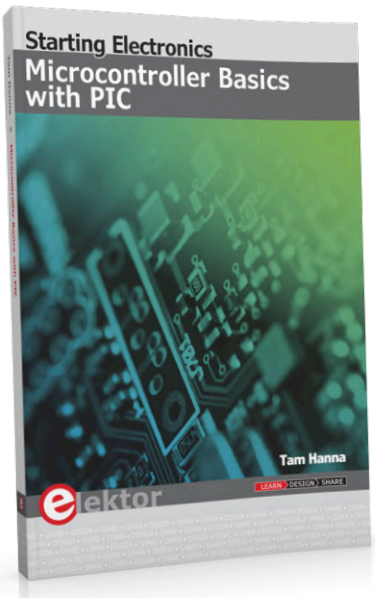
Elektor Team

Si vous voulez construire des applications pour l'internet des objets, il est bon de maîtriser les µC à 8 bits. Tam Hanna, auteur d'un livre sur ce sujet, publié par Elektor, nous fait part ici de ses réflexions sur les microcontrôleurs petits et grands et de son expérience de l'électronique.

Elektor : Votre nouveau livre sur *les bases des microcontrôleurs PIC* aide le lecteur à comprendre et à programmer un microcontrôleur à 8 bits. Pourriez-vous éclairer votre choix de ce sujet ?

Tam Hanna : Parce que les contrôleurs à 8 bits offrent une ouverture fascinante sur le monde de l'électronique programmée. Si vous comprenez ce qui se passe dans un microcontrôleur à 8 bits, la description de l'architecture d'un RISC-V sera plus facile à saisir. Gamin, j'ai commencé ma carrière par l'excellent livre *L'électronique - pas de panique* (hélas épuisé en français). Je suis honoré de pouvoir aujourd'hui rendre la pareille.

À bien des égards, la situation à laquelle sont confrontés les jeunes électroniciens aujourd'hui est différente. Certes, la disponibilité de systèmes et de langages comme Python leur facilite les choses, mais je rencontre des gens qui se heurtent à un mur, car il leur manque les bases de l'électronique. Avec son format de 8 bits, un microcontrôleur rudimentaire peut être entièrement compris au niveau de l'assembleur. Cette connaissance intime peut ensuite être transférée à des architectures beaucoup plus complexes. Comme j'utilise encore beaucoup le PIC pour des études de produits que je vends, j'ai eu l'impression qu'il fallait un livre.



Les bases incontournables de la programmation.

Elektor : Pour qui l'avez-vous écrit ? Pour profiter de la lecture de *ce livre*, faut-il savoir programmer en C ?

Tam Hanna : Je ne m'inquiète pas des compétences en programmation en C. Le C a une mauvaise réputation, car il est facile de se blesser avec. Comme avec un certain bombardier supersonique (TU-22) enclin à inverser les commandes lorsqu'il était piloté sans le doigté adéquat, mais je m'égare, excusez-moi.

Je me préoccupe plus du niveau de connaissances en génie électrique. Si vous ne connaissez pas au moins les bases de l'électronique analogique sur lesquelles repose l'électronique numérique, si vous ne comprenez pas la loi d'Ohm, vous aurez des problèmes avec les microcontrôleurs et leur programmation

.

Elektor : Qu'est-ce qui vous a le plus plu dans l'écriture du livre *Les bases du microcontrôleur avec le PIC* ?

Tam Hanna : Lorsque je me suis inscrit à *SourceForge*, le site m'a demandé de remplir un questionnaire d'auto-évaluation. C'est là que j'ai compris qu'il me fallait écrire un livre sur ce sujet. Écrire ce livre m'a permis de trier rigoureusement les connaissances recueillies au cours de ma carrière et m'a fourni l'occasion de recycler une part de cette expérience pour aider les autres.

Elektor : Quelle a été la partie la plus difficile de l'écriture du livre ?

Tam Hanna : Savoir m'arrêter. Je croyais m'adresser à une version plus jeune de moi-même, assis dans son labo, essayant d'acquérir ce supplément de compétences qui lui permettrait d'obtenir ce poste de travail rémunéré dans lequel il s'épanouirait professionnellement. J'avais tellement de choses utiles à lui dire, pas seulement sur les microcontrôleurs, aussi sur l'électrotechnique en général et sur l'équipement de son labo. Il a fallu m'imposer le silence, sinon je serais encore en train d'écrire, et aucun lecteur n'en profiterait jamais. Mon éditeur m'a habilement amadoué en faisant miroiter l'espoir d'une deuxième édition augmentée. C'est donc l'intérêt que témoigneront les lecteurs qui décidera de la suite.

Le microcontrôleur à 8 bits

Elektor : Pourquoi le microcontrôleur à 8 bits reste-t-il d'actualité ?

Tam Hanna : Les microcontrôleurs à 32 bits sont de moins en moins chers, d'accord. Mais avons-nous *toujours* besoin de la performance des 32 bits ? Regardez ce qui s'est passé lorsque Java est passé en 64 bits. Dans de nombreux cas, les pointeurs plus longs ont entraîné une dégradation des performances du système. Surtout quand on recherche une faible consommation, le fait de devoir alimenter trente-deux cellules de mémoire au lieu de huit pour stocker un pointeur fait la différence.

Elektor : Puisque les 8 bits restent d'actualité, qu'aimez-vous dans la famille des PIC ?

Tam Hanna : La simplicité. Ce n'est pas une coïncidence si Massimo Banzi a choisi l'AVR pour Arduino Uno, un 8 bits optimisé pour les besoins des compilateurs C. Le PIC a été conçu pour être programmé à la main. Son architecture interne est donc parfaitement adaptée à l'apprentissage et à l'enseignement.

Elektor : Certains diront qu'un microcontrôleur à 8 bits est difficile à programmer en C. Et vous ?

Tam Hanna : Aucun outil ne peut satisfaire tout le monde. Si vous voulez exécuter un algorithme complexe, un microcontrôleur à 8 bits n'est probablement pas idéal. Vous n'aimerez pas ça non plus si vous êtes enclin à allouer généreusement la mémoire.

L'ordinateur portable Palm VII avec moins de 2 Mo de mémoire totale !

Mais gardons les pieds sur terre. Pour beaucoup si ce n'est pour la plupart des applications, la tâche à accomplir est assez simple. Dans ce cas, une petite routine C est forcément plus facile à gérer que l'assembleur, j'en conviens. Que celui qui craint de se sentir à l'étroit prenne donc un contrôleur plus puissant ou fasse appel aux ressources d'un ordinateur. Chacun son truc ! Mon point de vue est marqué par ma propre expérience. Regardez ce Palm VII. J'ai programmé avec ce truc qui avait moins de 2 Mo de mémoire, alors forcément…

Électronique et cigares

Elektor : Assez parlé des 8 bits. Quand vous êtes-vous intéressé pour la première fois à l'électronique ? Avez-vous été inspiré par quelqu'un ?

Tam Hanna : Dans les années 1990 et 2000, mon intérêt pour la technologie était mal vu autour de moi. Je l'ai donc professionnalisé pour échapper à ce rejet : microcontrôleurs et PDA ne se soucient pas de savoir qui les utilise. Si le code et la configuration des circuits sont bons, ça fonctionne. C'est ainsi que je me suis évadé et que j'ai pu travailler avec plaisir et profit depuis lors.
J'ai beaucoup appris en expérimentant moi-même en plus de ce que j'ai appris à l'école d'ingénieurs TGM de Vienne, au labo notamment, où j'ai vu tant de choses que je n'aurais peut-être pas vécues ailleurs.

Elektor : Quels conseils donneriez-vous aux lecteurs d'Elektor ?

Tam Hanna : Au risque de donner l'impression que votre labo est situé à La Havane, n'hésitez pas à acheter du matériel d'occasion. Disposer de son propre analyseur de réseau vectoriel ou son propre analyseur de spectre vaut son pesant d'or.

Elektor : Les ingénieurs et les *makers* connaissent l'importance des leçons tirées de leurs propres erreurs. Quelle est votre erreur la plus instructive ?

Tam Hanna : Essayer de réparer en solo un atténuateur TDS754D. Il y a des fois où il vaut mieux demander de l'aide ou confier une tâche à un collègue. Ça peut vous faire gagner du temps et de l'argent.
Une autre erreur consiste, lors du démontage d'un appareil en vue d'une réparation, à ne pas coller religieusement sur la carcasse les

À chaque oscillo sa fonction de prédilection.

Fumisterie ? Non, mais humidificateur de nouvelle génération.

pièces spécifiques, difficiles à remplacer si elles venaient à manquer lors du remontage.
Osez aller là où personne n'est jamais allé. L'impression 3D pour des applications de grande consommation était jusqu'à présent considérée comme une folie. Je suis convaincu que les propriétaires d'*Humidors* apprécieront de pouvoir choisir la couleur personnalisée de leur humidificateur.

Elektor : Quelle est la prochaine étape pour vous ? Vous avez un nouveau livre, produit ou projet en cours de réalisation ?

Tam Hanna : Permettez que j'allume un cigare. Je crois qu'il faut oser aller là où personne n'est jamais allé. L'impression 3D pour des applications de grande consommation était considérée comme une folie il y a peine quelques années. Je suis convaincu que les propriétaires d'*Humidors* apprécieront de pouvoir choisir la couleur personnalisée de leur humidificateur. Un de mes prochains produits est l'*HygroSage*, humidificateur qui n'a *pas* besoin d'être calibré, dispose d'un écran couleur et présente une précision de 2 % garantie à vie.
Une autre nouveau produit est la série d'écrans de remplacement *Stinkely*. Je sais que *Danaher* fabrique des tubes cathodiques à un prix très intéressant pour la série de traceurs de courbes 57x - nous allons bientôt mettre un terme à cette mécanique. De plus, le fait de sauver des appareils de la décharge me procure toujours de fortes sensations et des douces satisfactions.
Je travaille aussi pour une start-up de mode, basée aux États-Unis, qui pour l'instant fonctionne en mode furtif pour des raisons juridiques. L'USPTO, c'est-à-dire le *United States Patent and Trademark Office*, autrement dit le bureau américain des brevets et des marques de commerce, est étrange et fascinant. Nous allons redéfinir les lunettes de soleil et… les cendriers. Patientez et laissez-nous vous surprendre.

Elektor : Parlez-nous de votre travail actuel.

Tam Hanna : D'accord, ce sera long. L'une des grandes choses dans ma vie, c'est que je ne s'ennuie jamais. Il se passe tellement de choses intéressantes.
Cet entretien a lieu à un moment zarbi. Mon entreprise est volontairement mise en quarantaine en ce moment, comme la plupart de nos fournisseurs. Mais ce n'est pas plus mal. J'ai récemment déménagé dans ce grand laboratoire souterrain où je suis bien. Je travaille sur toutes sortes de projets. Permettez-moi de vous donner mon compte Instagram : www.instagram.com/tam.hanna. Ainsi, en suivant mon *Instagram*, vous resterez informé de ce que fait le laboratoire *Crazy Electronics*.
Grâce à nos stocks, mon entreprise peut continuer de travailler normalement. L'objectif est de fournir à nos clients des services de conseil en ingénierie pour maintenir leurs processus en marche. En plus des nouveaux produits, je profite de l'occasion pour effectuer des tâches de maintenance. Je pense à réviser mon multimètre de table *Solartron*, à récurer les buses d'extrusion de mon imprimante 3D bouchées, à faire des rénovations intérieures et à installer de nouvelles étagères à la cuisine. Enfin, je devrais désencombrer le labo pour pouvoir enfin y passer la serpillière.

Vous aussi travaillez sur un projet comportant un microcontrôleur à 8 bits ? Vous pouvez en partager les détails et collaborer avec d'autres ingénieurs et électroniciens grâce à un compte Elektor Labs gratuit. Inscrivez-vous dès aujourd'hui !

200305-02

Si petit mais si utile et même irremplaçable

LIVRE

Livre : Microcontroller Basics with PIC
www.elektor.fr/microcontroller-basics-with-pic

Robert Lacoste :
grands et petits secrets de l'électronique

Le monde de l'électronique est à la fois vaste et tout petit ! Tout touche à tout, le plus petit détail peut avoir les plus grandes conséquences. L'objectif de R. Lacoste, l'auteur de

Percer les mystères de l'électronique,

est de vous donner des pistes pour comprendre et vous permettre ensuite de progresser seul. Repoussez vos propres limites et apprenez à détecter celles du matériel et du logiciel utilisés. Repassez par les notions de base et distinguez les véritables progrès techniques des laïus commerciaux.
Ce livre vous aide à y voir plus clair. Sans formule mathématique qui ne soit pas à la portée d'un lycéen, il balaye tout le champ de l'électronique. L'auteur explique comment ça marche, pourquoi parfois ça ne marche pas, et différentes techniques pour que vos projets marchent toujours.

Auteur de plus d'une centaine d'articles dans les revues techniques spécialisées Elektor et Circuit Cellar, Robert Lacoste a plus de trente ans d'expérience dans le domaine des signaux mixtes : acquisition et traitement du signal, radiofréquences et antennes, hyperfréquences, électronique ultra-rapide..

Sommaire

Prix : 33,75 € (membres)
www.elektor.fr/19080

nouveau livre en français
248 pages en couleur

le retour **des petits circuits**

… et des bonnes pépites d'Elektor

compilation par **Eric Bogers** (Elektor Pays-Bas)

Pour un dimanche après-midi pluvieux, ou pour une de ces journées de confinement où les murs semblent se rapprocher au fil des heures, voici quelques petits circuits qui peuvent être rapidement assemblés sur une carte d'expérimentation.

idée : Petre Petrov (Bulgarie)

Amplificateur audio simple à trois voies

Le schéma présenté n'a pas de prétentions de haute-fidélité mais il n'en présente pas moins quelques caractéristiques intéressantes.
Pour simplifier les choses, l'alimentation de cet amplificateur fait appel à un bloc d'alimentation externe d'un ordinateur portable. Il serait surprenant que vous n'en ayez pas (au moins) une qui traîne chez vous. Dans ce cas, vous en trouverez facilement un et pour un bon prix dans les circuits de récupération. Leur tension de sortie est généralement de 19 V_{CC}. Leur inconvénient majeur est la présence d'un signal de bruit non négligeable superposé à la tension de sortie continue. C'est pourquoi on insère un filtre LC qui réduira ce bruit autant que possible.

Pour l'amplificateur proprement dit, on utilise le célèbre circuit intégré LM380 (ou LM384) ; en fait, on en utilise trois, pour construire un amplificateur à trois voies. Il y a donc un chemin de signal séparé pour les sons des trois registres : grave, médium et aigu.

Diverses configurations possibles

Les filtres séparateurs peuvent être contournés grâce aux commutateurs S3 à S8.
L'amplificateur possède également des entrées séparées pour le grave, le médium et l'aigu, mais celles-ci peuvent aussi être interconnectés à l'aide des commutateurs S1 et S2.
Le circuit n'est pas critique et pourra être assemblé facilement sur une carte d'expérimentation.
Attention, ça chauffe ! La température des circuits intégrés amplificateurs peut atteindre des valeurs élevées ; pour éviter leur destruction, il faut sans faute les monter sur un radiateur. Selon l'impédance des haut-parleurs utilisés, il faudra adapter la valeur du condensateur de sortie comme indiqué dans le schéma.
Le LM380 et le LM384 ont tous deux un gain fixe de 50 avec une impédance d'entrée de 150 kΩ. Avec une tension d'alimentation de 19 V, la puissance de sortie maximale (pour un taux de distorsion de 10 %) atteint environ 3 W.
Pour faciliter la lecture du schéma, nous en proposons une version agrandie en téléchargement gratuit [1]. .

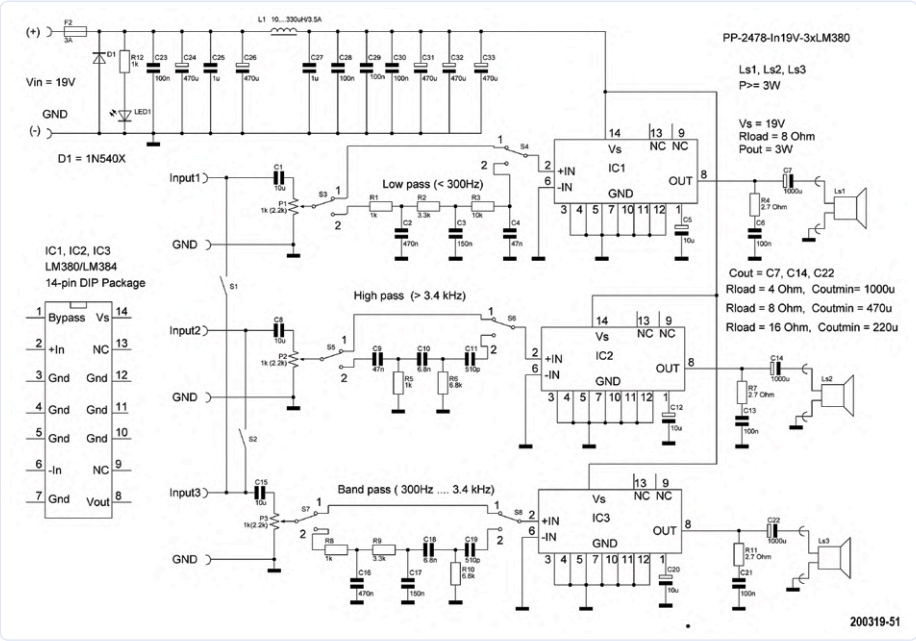


Figure 1. Compliqué, ce schéma? Non, regardez bien, ce n'est qu'une impression trompeuse.

idée : Elex

indicateur simple d'ions négatifs

On entend dire que les ions négatifs auraient un effet positif sur la santé et l'humeur des humains. Cet effet positif des ions négatifs est principalement revendiqué par les fabricants d'appareils appellés ioniseurs ; il n'existe aucune preuve irréfutable pour ces allégations. N'empêche que par temps de confinement certes justifié pour freiner ce fichu virus, il est important aussi de maintenir une atmosphère intérieure aussi saine que possible.
Grâce au circuit présenté ici, vous pourrez donc facilement vérifier la concentration d'ions négatifs dans l'air que vous respirez.
Le schéma montre qu'il suffit de quelques composants pour détecter dans l'air ambiant la présence d'ions négatifs. Ces ions sont capturés ou collectés, si vous préférez, par une petite plaque métallique (bloc grisé dans le schéma). En présence d'ions positifs, considérés comme «nocifs», il ne se passe rien. Si en revanche les «bons» ions négatifs prennent le dessus dans l'air ambiant, le potentiel électrique sur la base du transistor

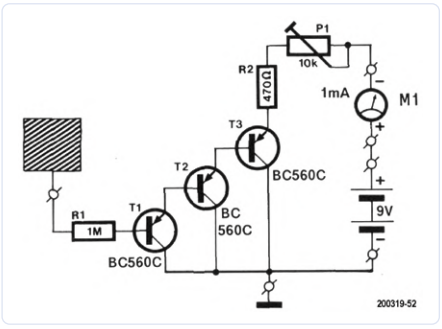


Figure 2. Quelques composants que vous devriez trouver dans votre tiroir ÇPTS.

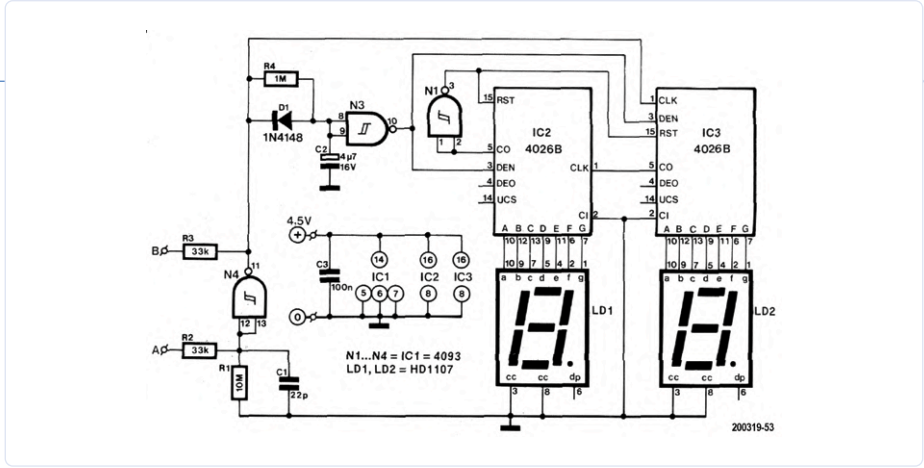


Figure 3. Un générateur de nombres aléatoires pour la loterie, facile à monter sur un morceau de carte d'expérimentation.

T1 deviendra négatif par rapport au potentiel sur l'émetteur : le transistor se met à conduire. Il en va de même pour le T2 et conséquemment aussi pour T3. Ces trois transistors forment une sorte de triple super-Darlington dont le gain total est considérable puisqu'il sera le produit de leurs trois facteurs d'amplification. Autrement dit, il suffit d'une faible charge négative sur la plaque métallique pour que T3 se mette à conduire ; le courant à travers ce transistor circule aussi à travers la bobine mobile du galvanomètre M1. Plus son intensité est élevée, plus l'angle de rotation de la bobine augmentera, et plus la déviation de l'aiguille couplée à la bobine sera forte. Ce détecteur indiquera donc non seulement la présence d'ions négatifs libres, mais, par l'ampleur de la déviation de son aiguille, il en indiquera également la concentration approximative.

Il est facile d'assembler ce circuit sur un morceau de circuit d'expérimentation à trous; mais le fil de connexion entre la plaque métallique et R1 devra être aussi court que possible. Le réglage du circuit avec le potentiomètre P1 dépend entièrement des circonstances. Testez le fonctionnement dans votre salle de bains : faites couler la douche, beaucoup d'ions négatifs seront libérés. Y aurait-il un rapport avec l'effet relaxant d'une bonne douche… ?

idée : Elex

Générateur de numéros de loterie

Avez-vous jamais fantasmé sur tout ce que vous pourriez faire après avoir gagné à la loterie ? Acheter une grande maison ou une nouvelle voiture, faire un tour du monde, ou dire ses quatre vérités à votre agaçant patron et démissionner dans la foulée… Ah, vous trouvez trop fatigant de remplir la grille, sans parler des risques de crise du genre « je t'avais pourtant dit de mettre le 22, pourquoi t'as mis le 33 !» etc.

Et si on laissait faire l'électronique ? Si la probabilité est la même pour toutes les combinaisons de nombres, nous pouvons utiliser un générateur aléatoire et un afficheur pour obtenir des nombres aléatoires réels (dans ce cas entre 0 et 49). Il ne restera plus aucun motif de dispute.

Au cœur du schéma du générateur (fig. 3) se trouve un oscillateur construit autour de N4. Deux petites plaques de métal (morceaux de circuit imprimé, punaises) sont connectées aux points A et B faire un bouton tactile. Tant que le bouton n'est pas touché, l'oscillateur est arrêté et sa sortie (broche 11 de N4) reste au niveau haut. En conséquence, la tension aux bornes de C2 est également élevée et la sortie de N3 est basse. Cette sortie commande les entrées d'activation de l'affichage de la paire de compteurs décimaux IC2 et IC3. Ces afficheurs à sept segments restent donc éteints.

Interlude : les compteurs décimaux de type CD4026B utilisés ici ont un décodeur et un pilote 7 segments intégrés. Cela permet de connecter directement aux sorties des compteurs les afficheurs à cathode commune LD1 et LD2.

Dès que l'on appuie sur le bouton, l'oscillateur se met à fonctionner à une fréquence de quelques kHz. À la sortie se trouve une onde carrée utilisée comme signal d'horloge pour IC3. La sortie CO de ce circuit intégré fournit le signal d'horloge pour IC 2.

Lorsque l'oscillateur fonctionne, C2 est rapidement déchargé lorsque la sortie de N4 est basse. La charge du condensateur via R4 est beaucoup plus lente. Cela signifie que la sortie de N3 est haute lorsque l'oscillateur fonctionne et que les afficheurs seront donc actifs. Comme le compteur va très vite, les segments paraissent allumés tous en même temps.

À l'instant où nous relâchons le bouton tactile fait maison, l'oscillateur s'arrête et l'état du compteur à ce moment est affiché. Celui-ci reste visible pendant quelques secondes, jusqu'à ce que C2 soit à nouveau suffisamment chargé. Les afficheurs s'éteignent et c'est le tour du nombre suivant.

La sortie de retenue des circuits intégrés compteurs est haute pour les valeurs de compteur de 0 à 4 et passe au niveau bas dès que le compteur atteint 5. Cela nous sert à remettre les deux compteurs à zéro via N1 dès que le compteur atteint 50.

Le circuit présente deux petits défauts cosmétiques : il est possible d'afficher «00», alors que cette valeur est invalide, le circuit ne vérifie pas si un nombre donné est déjà apparu. Si ces petits défauts vous gênent, cherchez donc une solution. Et si jamais vous gagnez le gros lot avec ce générateur, n'oubliez pas que c'est Elektor qui vous aura aidé à faire fortune !

200319-04

LIEN

[1] **page en ligne de cet article :** www.elektormagazine.fr/200319-04

PRODUITS

Livre (en anglais) "Electronic circuits for all"
www.elektor.fr/electronic-circuits-for-all

multimètre OWON OW18E
avec Bluetooth

Harry Baggen (Pays-Bas)

Tout doit être «connecté», même l'électricien. Il lui faut donc un multimètre qui transmette les données de mesure sans fil à son téléphone ou sa tablette. C'est donc un multimètre Bluetooth qu'il lui faut, et bien sûr un téléphone avec l'application idoine. Je teste ici le nouvel Owon OW18E, doté du Bluetooth, mais aussi de nombreuses autres fonctions. Il offre une grande précision pour un prix qui me paraît modéré. Je vous invite à lire mon banc d'essai.

OW18 est la plus récente série de multimètres portables d'Owon. À leurs OW18A et OW18B se sont joints les OW18D et OW18E qui offrent une plus grande précision et un affichage avec une résolution plus élevée (4 ½ contre 5 $^{5}/_{6}$ chiffres). Comme la version B, la version E de cette série est équipée de Bluetooth.

Quincaillerie

L'aspect de l'OW18E est celui des autres appareils de cette série. Le boîtier, assez grand, fait une impression de robustesse. Il est équipé d'une coque de protection (amovible) en plastique souple de couleur bleue. Le sélecteur rotatif fonctionne bien. Le grand afficheur donne au milieu la valeur mesurée sur quatre chiffres, sans le 1. La valeur maximale est donc 19999. Sur les bords, plusieurs indicateurs apparaissent en fonction du contexte de mesure. L'afficheur est (assez) facile à lire, mais vu de dessus, son contraste est limité. À l'arrière, il y a le pied escamotable et un compartiment pour une pile de 9 V.

Quatre boutons poussoirs servent à passer à la deuxième fonction d'une position donnée du sélecteur central, au réglage manuel du calibre, au rétroéclairage, pour le Bluetooth, et pour les modes *hold*, *relative* et *duty-cycle*.
Le multimètre est livré avec un jeu de cordons de mesure avec des pinces crocodiles assorties, un câble avec une sonde de température (typeK), une pile, un manuel, une fiche technique, une carte avec des instructions pour télécharger le logiciel et enfin un tournevis pour le compartiment de la pile ou le boîtier.

Possibilités

Le nombre de possibilités de mesure est considérable. Les gammes les plus utilisées sont bien sûr V et A. Pour les tensions alternatives, on mesure de la valeur efficace (True-RMS). Il existe une gamme de tension très sensible de 20/200 mV et une gamme de courant très sensible de 200 µA. On remarquera aussi que l'on peut mesurer jusqu'à 20 A (durant 10 s max.), alors que sur d'autres multimètres la limite est souvent de 10 A.

Mentionnons encore les modes ohmmètre, testeur de diode et testeur de continuité. La mesure de la capacité est possible jusqu'à 20 000 µF. La mesure de la fréquence va jusqu'à 20 MHz, soit une fourchette assez large pour cette gamme de prix ! La mesure du rapport cyclique est également possible dans ce mode. Avec la sonde fournie, la mesure de la température atteint 400 °C. Le mode NCV permet (en principe) de détecter sans contact si une fiche, une ligne ou une prise est sous tension.
Au-dessus de l'afficheur, un indicateur à LED clignote lorsque des mesures sans contact sont effectuées sur la tension du secteur. Cet indicateur s'allume également lors de l'utilisation de la plupart des fonctions. En plus du capteur NCV, une autre LED blanche sert de lampe de poche. Le multimètre est bien sûr équipé d'une fonction d'arrêt automatique (APO) qui le met en sommeil après 30 mn d'inactivité.

Résultats des mesures

Commençons par la sécurité. Selon le fabricant, le compteur répond à la catégorie de mesure CAT III 1000 V/CAT IV 600 V. Cela signifie une assez bonne isolation. Vous pouvez donc dans tous les cas mesurer en toute sécurité dans et autour de la maison.
La précision de base de l'OW18E est spécifiée à 0,1 % + 2 chiffres pour la gamme 2 V et plus. Comparé à un multimètre d'une précision dix fois supérieure, l'Owon a fait ses preuves : l'écart étant inférieur à 2 chiffres. Avec les mesures de courant continu, les résultats ont été comparables.

En courant alternatif, l'écart était de quelques millivolts à une valeur mesurée de presque 2 V. C'est bien, puisque les spécifications indiquent 0,5 % + 10 chiffres. Pour les mesures en courant alternatif, une gamme de fréquences de 40 à 1000 Hz est spécifiée. Lors de mes tests divers, la lecture est restée assez précise à environ 1,5 kHz.

Les mesures de résistance que j'ai effectuées ont aussi été plus précises que les 0,3 % promis. Au-dessus de 1 kΩ, l'écart constaté par rapport à une mesure de référence n'était que de 0,1 à 0,2 %. De même, dans la mesure de la capacité, la valeur mesurée ne s'écarte que de quelques chiffres du multimètre de référence. Ce sont tous d'excellents résultats dans cette gamme de prix.

Figure 1. L'OW18E et les accessoires fournis.

Figure 2. À côté du capteur NCV : une LED d'éclairage, pour faire lampe de poche. Il fallait y penser !

Pratique

Comment ce multimètre se comporte-t-il dans l'usage quotidien de l'électronicien ? L'affichage est clair, les chiffres faciles à lire sous presque tous les angles. La fonction de sélection automatique de calibre n'est pas très rapide, mais elle est fiable. J'ai bien apprécié l'astucieuse indication, sous la virgule, du calibre de mesure en très petits chiffres (**fig. 4**). Les fonctions des boutons poussoirs sont faciles à retenir. Les changements de fonction ne passent pas inaperçus : lorsqu'on actionne ces boutons, un signal sonore retentit et le voyant lumineux s'allume. Le temps de réaction du *bip* du testeur de continuité et de la LED est très court, celui de l'afficheur est long. Le capacimètre fonctionne bien, même avec des valeurs plus élevées. Pour un électrochimique de 2200 µF, il faut attendre environ 8 s pourvoir s'afficher la capacité.
L'éclairage de l'afficheur garantit une bonne lecture dans un environnement sombre ; d'ailleurs la lampe de poche à LED sur le dessus s'allume en même temps sans qu'on puisse intervenir pour les faire fonctionner séparément. Je n'ai pas aimé la fonction NCV de l'OW18E, elle m'a paru assez insensible. Il m'est arrivé de placer

Figure 3. La pile de 9 V est montée sur un support séparé qui fait couvercle à l'arrière.

Figure 4. Détail astucieux : le calibre apparaît sous la virgule.

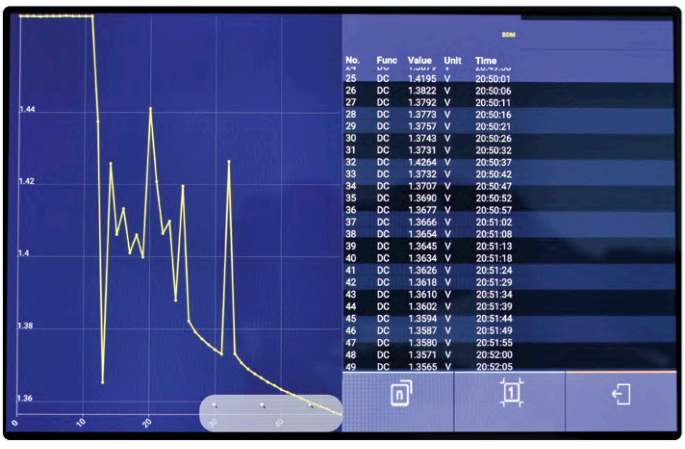


Figure 5. L'écran d'enregistrement dans l'application.

le capteur tout près d'une prise de courant sans que le détecteur s'allume. Manque de fiabilité ?

Bluetooth

En plus de sa grande précision pour cette gamme de prix, la principale caractéristique intéressante de l'OW18E est sa fonction Bluetooth. Il est doté d'un module BLE4.0, qui assure une connexion stable et une faible consommation d'énergie. J'ai activé la fonction Bluetooth sur le multimètre pendant plusieurs heures sans arriver à épuiser la batterie. L'application *MultimeterBLE* correspondante est disponible pour les utilisateurs d'Android et d'iOS.

Après avoir installé et démarré cette application, activez la fonction Bluetooth sur le multimètre, lequel apparaîtra dans l'application sous le nom «BDM». Il faut le sélectionner pour que l'application se connecte et affiche la valeur mesurée. Les fonctions des boutons poussoirs peuvent être commandées depuis l'application. Pour le commutateur rotatif, la commande est exclusivement manuelle. L'application offre une fonction d'enregistrement de données avec des options de réglage, la cadence d'enregistrement. Les valeurs enregistrées peuvent être stockées et envoyées dans un fichier. Depuis l'application, vous pouvez demander l'enregistrement (**fig. 5**) des mesures puis interrompre la connexion Bluetooth. Plus tard, les valeurs mesurées et enregistrées peuvent être récupérées via l'application. Celle-ci offre la possibilité de se connecter à deux multimètres en même temps.

Je regrette que l'application ne fonctionne qu'en mode paysage et affiche par défaut un écran pour deux multimètres. Pour l'instant, la plupart des utilisateurs n'auront qu'un multimètre Bluetooth à leur disposition. L'application fonctionne bien, la connexion est stable et les transferts de données sont rapides. Cette connexion sans fil est bien pratique chaque fois que vous ne pouvez ou ne voulez pas rester les yeux rivés sur l'afficheur du multimètre. Les informations fournies dans l'application sont rigoureusement les mêmes que celles qui s'affichent sur le multimètre lui-même. Il me semble que ce multimètre est une bonne solution de remplacement du très populaire *Mooshimeter* désormais indisponible.

Conclusion

Pour moins de 70 €, l'Owon OW18E est un excellent outil. Vous aurez un appareil assez solide qui non seulement offre de nombreuses possibilités de mesure, mais en affiche les valeurs avec une étonnante précision. Grâce à Bluetooth, vous pouvez les lire à distance et les enregistrer. Que vouloir de plus pour ce modeste prix ?

200322-02

@ WWW.ELEKTOR.FR

OWON OW18E Bluetooth Multimeter (20000 Counts)
www.elektor.fr/owon-ow18e-bluetooth-multimeter-20000-counts

électronique interactive

Corrections & mises à jour || Questions & réponses

compilé par **Clemens Valens** (Elektor Labs)

Mises à jour et ajouts aux projets publiés dans le magazine *Elektor*, pimentés de trucs et astuces, de conseils techniques et de réponses aux questions de nos lecteurs.

projet ancien, progiciel nouveau

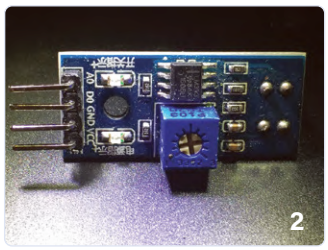
Mon intérêt a été attiré par l'article *générateur MLI universel* de l'édition de juillet/août 2010 de votre magazine, car il utilise un codeur rotatif pour le choix des options dans un menu. J'ai essayé à plusieurs reprises de programmer un PIC16F628A [comme celui de ce projet, *ndlr*], mais le code refuse de fonctionner. Je pense qu'il a été écrit en C/C++ avec probablement un compilateur CCS. Avec une carte EasyPICv7 et une licence mikroC, j'ai porté le logiciel original et constaté qu'il ne marchait pas non plus. Je l'ai donc analysé, j'en ai modifié quelques lignes et programmé un PIC16F88 avec le fichier HEX résultant (le PIC16F628A d'origine manque de RAM, la nouvelle bibliothèque LCD consomme probablement plus de ressources que celle utilisée par Alexander Ziemek, le concepteur). Ainsi, j'ai obtenu un fonctionnement normal, mais certainement perfectible.
J'ai ajouté de nombreux commentaires au code pour en faciliter la lecture et la compréhension. Je précise que je fais ça en amateur, je ne suis pas un concepteur professionnel.
Je vous offre mon travail [1] sous forme d'archive ZIP avec le schéma légèrement modifié, car je n'utilise pas la broche R/W du LCD, maintenant connectée à la terre. Cela libère la broche RA2 du processeur. L'esprit, les fonctions et les caractéristiques de l'original sont inchangés.

Philippe Le Guen

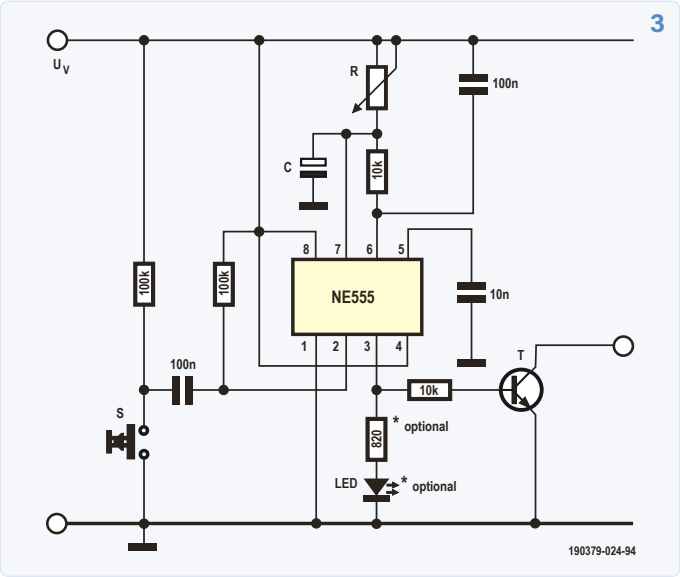
www.elektormagazine.com/magazine/190379-D-01

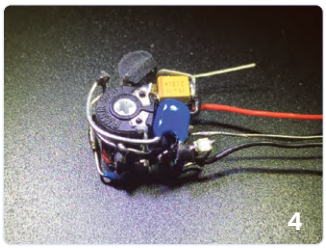
capteur amélioré pour le contrôle du débit d'eau

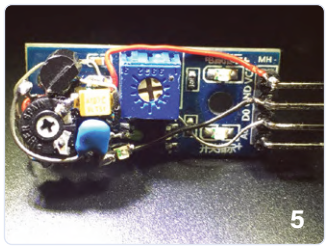
Après avoir lu l'article *surveillance de la consommation d'eau avec l'ESP32* de Denis Lafourcade de l'édition de juillet-août 2019 d'Elektor, j'ai immédiatement réalisé qu'un tel système me serait utile. Pour commencer, je ne trouvais pas de capteur adapté à mon compteur d'eau. Une demande de renseignements auprès du fabricant est restée sans réponse. J'en ai donc bricolé une.
Un premier essai avec un capteur à effet Hall a échoué : le compteur d'eau ne contient aucune pièce magnétique, pour – ai-je découvert plus tard – déjouer les tours de passe-passe frauduleux. Et cette surface brillante en demi-lune sur l'aiguille des litres (**fig. 1**), pourrait-elle servir à détecter la rotation de l'aiguille ? Comme j'avais dans mes réserves des capteurs optiques à infrarouge réfléchissant (**fig. 2**), j'ai cherché à en monter un sur le compteur. Sur Thingiverse [1], j'ai téléchargé des fichiers de modèles 3D pour un boîtier de capteur approprié, j'en ai imprimé un et j'y ai placé un capteur. Cet ensemble a été fixé au compteur d'eau à l'aide de ventouses. Puis j'ai rapidement assemblé le reste du moniteur de consommation d'eau.

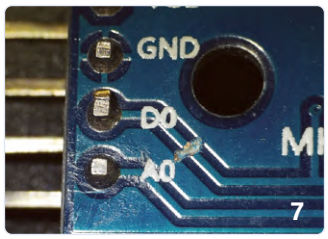


Le premier test a été décevant : les quantités mesurées étaient beaucoup trop élevées. En étudiant bien le schéma et en analysant le signal de sortie du capteur, j'ai compris que le circuit attend des impulsions de 500 ms, or ce n'est pas ce que délivre mon capteur IR.









Non seulement la durée des impulsions dépendait du débit de l'eau (ce *qui est laid comme le débit de lait*), mais il arrivait aussi assez souvent que la surface brillante s'arrête juste en face du capteur, ce qui fausse tout. Que faire ?
J'ai pensé au bon vieux 555, idéal puisque je voulais que tout reste analogique. Et comme j'avais quelques 555 à disposition, j'ai opté pour cette solution. J'ai utilisé le 555 dans une configuration de multivibrateur monostable (MMV) non redéclenchable (**fig. 3**). Cela signifie qu'il n'accepte d'être déclenché que lorsqu'il est au repos. Quand c'est le cas, il délivre une impulsion d'environ 500 ms. Cette méthode permet également d'éviter les faux déclenchements lors de la mise en marche.
La construction n'a pas été une mince affaire, tous les composants, y compris le condo de 100 µF, devaient tenir dans le boîtier du capteur. J'ai donc utilisé des CMS (**fig. 4, 5, 6**).
Pour insérer le circuit, il faut couper la piste entre le LM393 sur la carte du capteur et la sortie sur le connecteur Do (**fig. 7**). La broche 7 du LM393 est ensuite soudée à l'entrée du MMV dont la sortie est connectée à Do.
À ma demande, l'auteur a également ajouté de brèves instructions sur la page du projet sur Elektor Labs [2] pour la mise en place du service IFTTT.

Hans Schneider

www.thingiverse.com/thing:2749407
www.elektormagazine.com/labs/waterflow-monitor

construire un moniteur de vibrations portable

L'exposition prolongée de votre organisme à des vibrations peut entraîner des blessures. Comment mesurer cette exposition ? Ce projet pourrait vous aider à le découvrir. «*sixbacon*» est un physicien et un ingénieur actif sur Elektor Labs, et intéressé par la santé et la sécurité. Préoccupé par la nocivité des vibrations et désireux d'évaluer les outils de bricolage qui sont peu ou pas réglementés, *sixbacon* s'est lancé dans l'étude d'un moniteur de vibrations fait maison. Celui-ci est constitué dans son cas d'un support en bois, mais ça pourrait être un modèle imprimé en 3D . Sa forme en «H» lui permet de s'insérer entre les doigts d'une main gantée, le capteur placé contre l'outil. Un module ESP32 avec afficheur OLED et un accéléromètre LIS331 ont été fixés sur le dessus pour permettre à l'opérateur de lire les informations. L'alimentation est assurée par une petite batterie USB glissée dans la manche de l'opérateur pendant l'utilisation.
Bien que conçu pour mesurer les vibrations causées dans la main et le bras de l'utilisateur par les outils électriques portatifs (perceuse, meuleuse, ...), les domaines d'application de cet appareil sont vastes, puisqu'il suffit de placer le capteur au bon endroit.

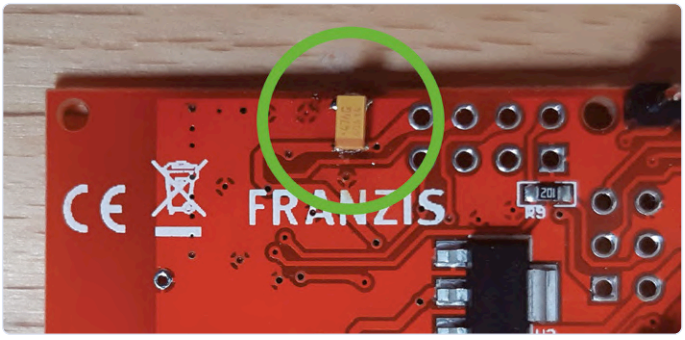
www.elektor-labs.com/1879

améliorer la carte Pretzel

En jouant avec la carte Pretzel [1], j'ai eu des problèmes, car elle ne faisait plus ce que j'attendais d'elle. Après quelques recherches en ligne, j'ai découvert que l'instabilité de l'alimentation de la partie WLAN peut se révéler critique, même sur des modèles récents.
Le remède suggéré était d'ajouter un gros condensateur tampon de 100 µF sur le connecteur de la carte pour améliorer la stabilité de la partie WLAN. Cela résout le problème, mais au prix du blocage du connecteur par le gros condo. Voici une méthode plus élégante. Le condensateur est un CMS de 47 µF, discrètement monté sous la carte, après voir soigneusement gratté au bon endroit le masque de deux pistes sur lesquelles j'ai soudé le composant (avec juste une goutte d'étain). Un modèle de 100 µF tiendrait aussi, mais je n'en avais pas.

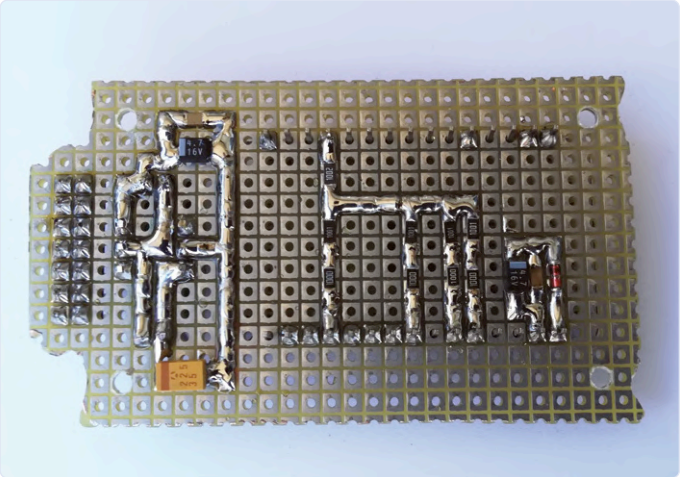
Uwe Werner

www.elektor.com/pretzel-board-iot-wifi-board



comment souder à la main ces mimuscules CMS

Il est devenu presque impossible de faire de l'électronique sans se frotter à ces minuscules composants montés en surface (CMS). Ils sont réputés difficiles à souder, mais on s'y fait et de nombreux



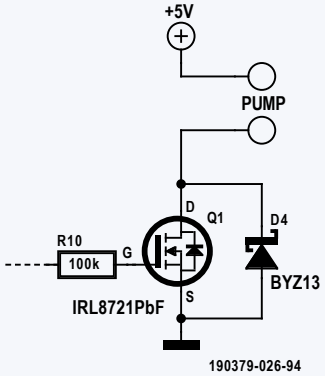
électroniciens ont trouvé des moyens de les utiliser pour le prototypage. Voici une astuce suggérée par Antoni Gendrau. Si vous en avez un autre, n'hésitez pas à l'ajouter en commentaire au projet pour la faire connaître au monde entier.

www.elektor-labs.com/1913

mise à niveau pour *l'alerte sécheresse*

Elektor 07/2020, p. 39 (200213)

Dans l'édition de juillet-août 2020 d'*Elektor*, nous avons décrit une *alerte sécheresse pour amateurs de plantes distraits*, un circuit qui avertit acoustiquement et optiquement si le terreau est desséché. C'est bien, mais tant qu'à faire, ne serait-il pas plus utile que la plante soit arrosée automatiquement ?



Nous sommes nombreux à y avoir pensé, le concepteur a retroussé ses manches. Et avec une petite pompe et quelques composants il propose une solution facile à réaliser. La pompe est commandée par un MOSFET (IRLB8721PbF), connecté à la sortie du circuit par une résistance de 100 kΩ (au lieu du buzzer) et protégé par une zener. Pour obtenir un arrosage digne de ce nom, la durée de l'impulsion produite par le circuit doit être multipliée par 10, c'est-à-dire de 100 ms à une seconde. Il suffit donc de remplacer C4 par un condensateur de 100 nF ; la période du multivibrateur astable doit être portée à trois secondes par exemple en utilisant une valeur de 2,7 µF ou de 3,3 µF pour C1.
Grâce à cette modification et à un réservoir d'eau de capacité appropriée, vos plantes pourront survivre sans surveillance pendant des mois.

www.sossolutions.nl/1150-peristaltic-liquid-pump-with-silicone-tubing

horloge *Stargate à LED* NeoPixel

Elektor 07/2020, p. 76 (200208)

Dans la description du laboratoire domestique de Peter Neufeld et de son horloge Stargate à LED NeoPixel (juillet-août 2020, p. 76), il était question de « nombreuses heures de travail ». En fait, il s'agit d'un kit de la société UGears - il n'y a qu'à détacher de la plaque de bois les pièces découpées au laser et de les assembler (sans colle !). C'est pas de l'horlogerie suisse.

https://ugearsmodels.com/de/date-navigator.html

limiteur de courant d'appel simplifié

Elektor 03/2020, p. 34 (200004)

Le concepteur de ce petit circuit pratique publié dans l'édition de mars/avril 2020 est le Suisse Kurt Kühni et non un membre de l'équipe d'Elektor (même si cela ne nous aurait pas déplu). Veuillez accepter nos sincères excuses pour cette involontaire usurpation de paternité.

190379-D-03

expérience vécue

Soudage sans plomb et zèle réglementaire de l'UE

Ilse Joostens (Belgique)

Au début des années 1970, nous vivions dans une maison dont les conduites d'eau étaient en plomb. L'année de mes cinq ans, ma famille a emménagé dans une construction neuve sans tuyaux de plomb, mais dont le revêtement de sol traité au *pentachlorophénol* et contenant de l'amiante. C'est épouvantable. À lire tout ce qu'on sait maintenant sur les risques que nous font courir des choses que nous aimions tant, c'est un miracle que je sois encore en vie et que je puisse écrire cette chronique. Depuis le temps de ma jeunesse, le monde est devenu beaucoup plus dangereux. Avec la réglementation RoHS (2006), l'Union européenne a lancé une croisade contre certaines substances dangereuses dans l'électronique grand public. Pour une raison qui me paraît mystérieuse, le plomb est devenu le principal coupable, au point d'être interdit même

Du plomb dans l'air

Si votre passe-temps est de souder, de faire des réparations ou de construire des circuits, inutile de lire la suite, tombez à genoux et remerciez les dieux qui vous autorisent à utiliser de la soudure au plomb et profitez-en pour souder beaucoup. Si, comme les miens, vos produits sont commerciaux, vous n'avez pas de chance ! Vous êtes condamné à souder sans plomb. Pour les prototypes, la soudure à base de plomb reste permise, mais pour éviter le risque de contamination, je n'en utilise plus.

La période qui a suivi l'entrée en vigueur du premier règlement sur la limitation des substances dangereuses (RoHS) en 2006 a été assez spectaculaire pour l'industrie électronique. Certains procédés sans plomb n'étaient pas encore au point, et l'expérience des nouvelles méthodes était limitée. Pendant cette période, mes cauchemars étaient peuplés de moustaches d'étain, acérées comme des rasoirs, qui poussaient et repoussaient sans cesse entre les broches des CMS, ce qui finissait toujours par des signaux de fumée. Aujourd'hui, pour le soudage automatisé, la plupart des problèmes ont été résolus. En pratique, les fameuses moustaches d'étain et autres catastrophes potentielles n'ont été que des problèmes mineurs. Je me demande quand même si l'électronique grand public moderne ne tombe pas en panne plus rapidement et plus souvent qu'avant. Si c'est le cas, la cause pourrait en être la moindre qualité des composants. Avec la hausse vertigineuse des importations à bas prix en provenance d'Asie, la qualité a systématiquement diminué.

Se passer de plomb pour la soudure manuelle reste difficile. Les photos qui circulent sur l'internet montrent les difficultés de la soudure manuelle sur les cartes mères des ordinateurs. La température de fusion plus élevée de la soudure sans plomb y est pour quelque chose. Pour vous en convaincre, demandez donc à votre moteur de recherche favori des photos avec pour mot clé «*solder fail*».

Avec mes anciens postes de soudage, il n'était pas vraiment facile de souder avec de la soudure sans plomb des composants traversants de grande taille sur des circuits imprimés à trous métallisés, surtout à proximité de grands plans de cuivre. Les fers étaient généralement réglés à une température de 420 à 450 °C, et dans certains cas, il fallait deux fers en même temps, et l'usure des pannes était accélérée.

Grâce aux stations modernes à pannes actives interchangeables, il est plus facile de souder sans plomb, mais la soudure sur de grandes surfaces de cuivre reste délicate, tout comme le dessoudage des composants traversants et la retouche des CMS. Tout cela se traduit par un plus grand nombre de composants défectueux et de produits inachevés. Les rares fois où il m'arrive de refaire des soudures à base de plomb, c'est toujours un soulagement.

Si vous avez l'intention de vous lancer dans l'électronique ou si vous soudez occasionnellement quelque chose dans le cadre de vos loisirs, évitez la soudure sans plomb. L'indispensable bonne station de soudure avec pannes actives serait chère et, en fait, il n'y a pas de risques spécifiques pour la santé à travailler à faire des soudures à base de plomb. Évidemment, il ne faut pas en manger. Le fil à souder sans plomb contient généralement un flux décapant, dont les vapeurs ne sont pas inoffensives. Il faut les aspirer et bien ventiler.

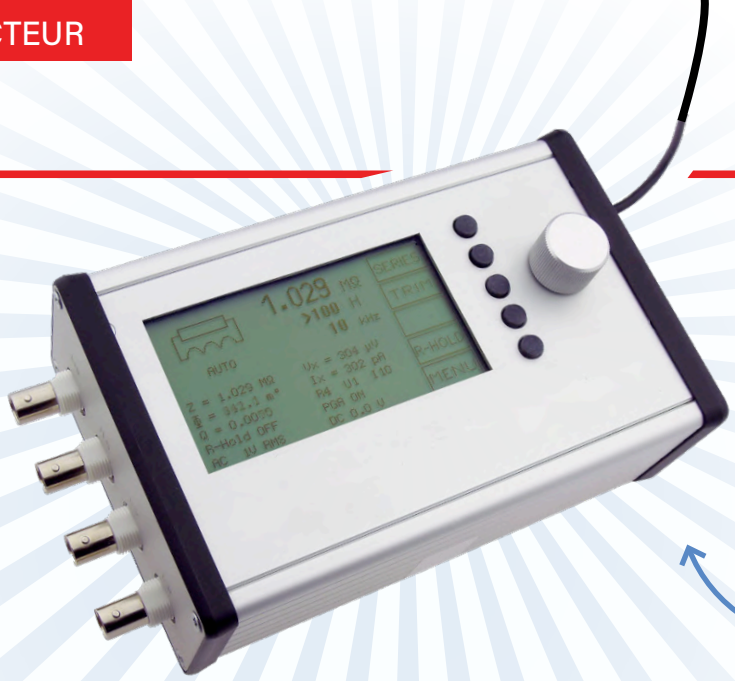
Recyclage

Je me demande si l'interdiction du plomb dans l'électronique est vraiment bénéfique pour l'environnement. Au lieu de l'électronique jetable bon marché actuelle, n'aurait-il pas été préférable de viser des produits de meilleure qualité et plus durables ? Ceci nous amène au recyclage, un sujet sur lequel les bureaucrates se montrent zélés. Dans la plupart des pays européens, il existe des organismes pour le recyclage des appareils mis au rebut et des déchets réutilisables. Chez moi, en Belgique, ils s'appellent *Recupel*, *Bebat* et *Fost* Plus.

Je suis favorable au recyclage, mais pas aux surcoûts ni à la paperasserie imposés par ces organismes. Heureusement, la gêne est mineure tant que je ne vends ni produits finis ni piles et que je limite l'utilisation d'emballages en plastique. À en juger par les témoignages de collègues et de concurrents en Allemagne, la situation en Belgique n'est pas trop mauvaise. La *Stiftung EAR*, équivalent allemand de *Recupel*, prend son rôle très au sérieux et inflige de lourdes amendes dans certains cas. Ce qui confirme qu'en Allemagne, les règles sont les règles.

Cependant, en dépit de toutes les réglementations et interdictions, des composants tels que les interrupteurs à mercure et les photorésistances au sulfure de cadmium (*horresco referens*) restent facilement disponibles auprès de sources asiatiques, sans parler de produits potentiellement mortels qui ne sont conformes à aucune norme, mais toujours en vente dans les magasins en Europe. Dois-je en déduire qu'il y aurait, pour nos décideurs politiques européens, deux poids deux mesures ?

200082-03



Projet prometteur :

nouveau LCR-mètre 50 Hz – 2 MHz

Précision et confort de mesure

Jean-Jacques Aubry

Il y a un peu plus de sept ans, Elektor publiait une série d'articles sur mon *LCR-mètre 0,05%*. Plus d'un demi-millier de lecteurs ont réalisé cet appareil avec succès à partir du kit proposé par Elektor. Content de ce succès, j'y ai aussi trouvé la motivation pour concevoir un nouveau LCR-mètre qui n'aurait pas pour ambition de faire mieux sur l'extrême voire l'excessive précision, mais sur le confort d'utilisation.

Elektor Kickstarter?

Ce projet est l'un des premiers à utiliser le nouveau module Kickstarter d'Elektor Labs en ligne. Notre objectif est de mesurer l'intérêt suscité par ce projet parmi nos lecteurs afin d'adapter au mieux le nombre de kits que nous allons produire..

Comment participer ?

1. Recherchez le projet en ligne sur Elektor Labs (www.elektor.com/lcr)
2. Cliquez sur «En savoir plus» dans la colonne de droite, à côté de la fusée.
3. La description des produits et les prix sont conformes à la situation actuelle.
4. Si vous souhaitez acheter ultérieurement : cliquez sur «Retour à ce projet».
5. Saisissez votre adresse électronique
6. C'est tout.

Il n'y a ni paiement préalable ni conditions à remplir : vous pouvez annuler à tout moment si vous n'êtes plus intéressé. Nous vous informerons par courrier l'avancement du projet, de la production et du délai de livraison.

Ah ! oui... j'oubliais...

Quand le produit sera disponible dans notre boutique, nos bailleurs de fonds bénéficieront tous d'une belle remise !

Changer de nom

Elektor Kickstarter : nous ne pouvons pas continuer d'utiliser ce nom. Participez au concours qui permet à chacun de nous aider à trouver un meilleur nom :
www.elektormagazine.fr/news/elektor-kickstarter-pledge-euh

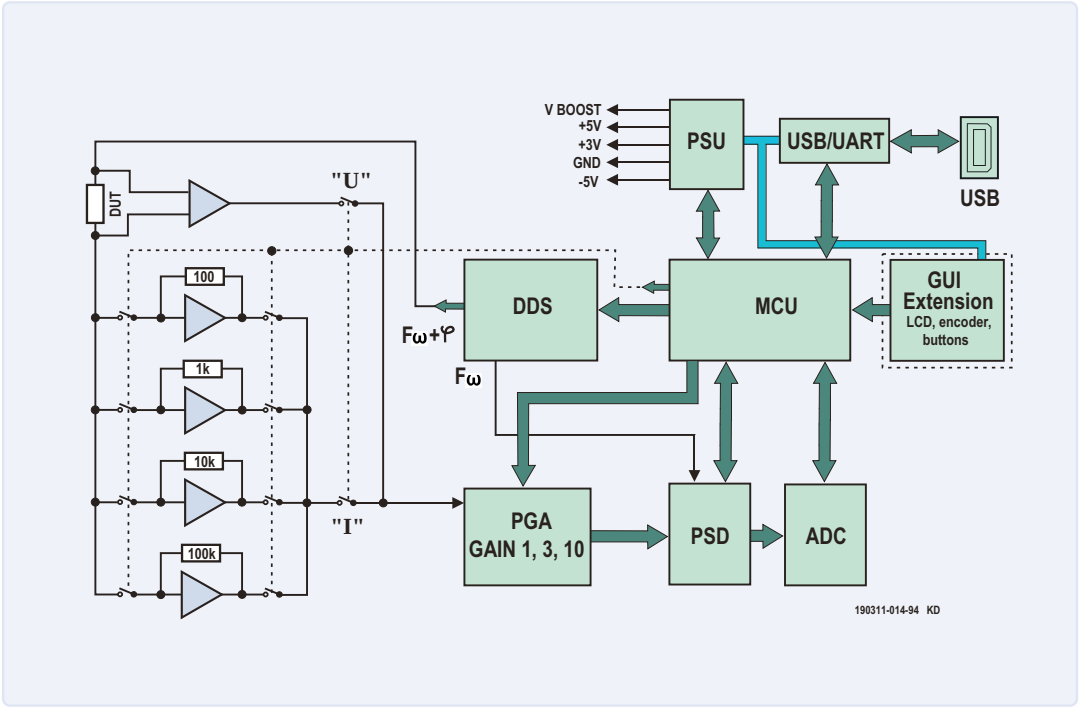


Figure 1. Synoptique du nouveau LCR-mètre. Toutes les fonctions sont sur la même carte, sauf l'afficheur, les organes de commande et les extensions possibles, réunis sur une deuxième carte (bloc GUI Extension entouré de pointillés).

Le défi que je me suis lancé portait sur le confort de mesure et des fonctions étendues :

- fréquence de test de 50 Hz à 2 MHz
- choix de 4 tensions de test : 100 mV, 200 mV, 500 mV ou 1 V efficace.
- polarisation continue additionnelle en tension jusqu'à 5 V pour les condensateurs, et en courant jusqu'à 50 mA pour les inductances.

Pour vaincre les réticences qu'un appareil de cette classe pourrait inspirer (à tort), j'ai apporté une attention particulière à la facilité de sa mise en œuvre (les calibrations sont automatisées) et à l'ergonomie : un codeur rotatif associé à 5 boutons poussoirs à fonctions multiples et un afficheur graphique LCD 240 x 128 pixels permettent de naviguer facilement et intuitivement dans les menus, de changer de fréquence et bien entendu d'afficher les mesures.

Repousser les limites

Pour caractériser les composants électroniques passifs (résistance, condensateur, inductance), l'impédance (Z) est un paramètre important. Pour déterminer cette impédance, il faut au moins deux valeurs (en grandeur et en phase), généralement la tension aux bornes du composant et le courant qui le traverse. Comme son prédécesseur de 2013, le nouveau LCR-mètre utilise la méthode du pont auto équilibré, dont le convertisseur courant/tension est un simple amplificateur opérationnel. La méthode est simple, la précision bonne, le coût raisonnable. Son principal inconvénient est la limitation par l'amplificateur opérationnel de la gamme de fréquences. La plupart des instruments de mesure similaires ont une fréquence haute limitée à 100 ou 200 kHz. J'ai réussi à repousser la fréquence haute à **2 MHz** sans augmentation exagérée du coût, et sans compromettre la simplicité de réalisation.
Ce projet sera décrit en détail dans le prochain numéro. Il comporte deux cartes.

Une carte principale dont les fonctions apparaissent sur le synoptique (**fig 1**) :

- **Circuit d'entrée :** le composant à tester (DUT) se voit appliquer un signal de test. Pour réduire l'influence des cordons, la mesure se fait avec une configuration à 5 connections (dont le blindage)
- **Générateur sinusoïdal :** la synthèse directe de fréquence (DDS) utilisée pour produire la fréquence de test, permet d'obtenir à la fois n'importe quelle fréquence dans la gamme de 50 Hz à 2 MHz, et un signal de même fréquence, mais à phase relative variable, pour le détecteur synchrone.
- **Amplificateur à gain programmable (PGA) : pour** procurer un gain de 1, 3 ou 10 afin d'attaquer le PSD dans les meilleures conditions.
- **Détecteur de phase (PSD) :** nécessite un signal parfaitement carré, à la même fréquence que le générateur, mais dont on puisse faire varier la phase relative. Cette variation de phase du signal de commutation par rapport à la sinusoïde permet de faire les mesures des composantes en phase ou en quadrature du signal d'entrée.
- Le **microcontrôleur**, un **CA/N** à 24 bits, la **passerelle USB** et **l'alimentation**

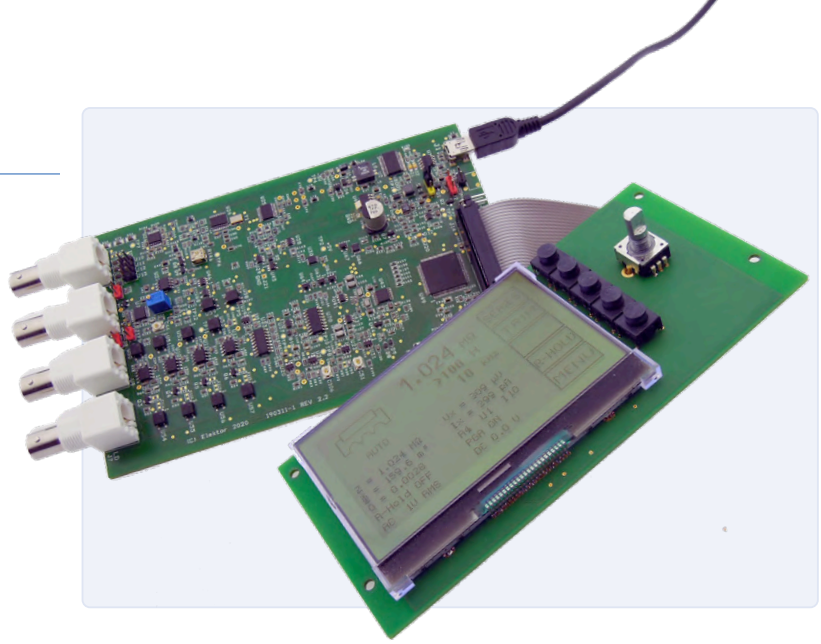


Figure 2. La combinaison carte principale + extension permet de réaliser un appareil autonome. Il est aussi possible d'utiliser la carte principale avec un PC.

Une carte d'affichage et d'interface avec l'utilisateur :

- Afficheur graphique
- 5 boutons à fonctions multiples (affichées sur l'écran) et un codeur rotatif

Sur l'honneur

Les caractéristiques du LCR-mètre sont résumées dans le **tableau** ci-dessous. Je vous recommande de suivre mon projet sur le site Elektor Labs : www.elektor.com/lcr. Avant d'engager la production du kit, Elektor lance une campagne de soutien : si vous êtes intéressé par la réalisation de ce projet et désirez en acquérir le kit, vous pouvez vous inscrire sans engagement formel contraignant, simplement sur l'honneur. La production du kit sera lancée à partir de 150 promesses. En échange de votre engagement, le kit vous sera proposé à prix réduit.

Le kit du LCR-mètre comprendra :

- **Carte principale** préassemblée avec tous les CMS soudés
- **Carte d'affichage** préassemblée avec tous les CMS soudés
- **Composants traversants** pour les deux circuits imprimés (afficheur LCD rétroéclairé, connecteurs, boutons poussoirs, codeur rotatif, bouton)
- **Câble en nappe** pour connecter l'afficheur et la carte principale
- **Câble mini-USB** pour la connexion au PC et les mises à jour du microprogramme **Coffret en alu Hammond, façades percées et fraisées**
- Visserie **Clip Kelvin avec câble de test à 4 fiches BNC**
- Manuel

200309-01

Caractéristiques techniques résumées

Affichage	• Valeurs des paramètres (principal et secondaire) • Circuit équivalent : série ou parallèle • Fréquence • |Z| • ϕ • Q ou D • Tension et courant du DUT (V_x et I_x) • Tension de test (AC) et valeur de la polarisation continue (DC) • État du gel de gamme (R_Hold) • (sur la droite) le label des boutons à fonction variable	
Domaine de mesure	**paramètres**	**valeur**
	L	10 nH ➞ 100 H
	C	1 pF ➞ 100 mF
	R, |Z|	10 mΩ ➞ 100 MΩ
	Q	0 – 5000 pour affichage
	Φ	– 90,00 ° / +90,00 °
Fréquences de test	50 Hz à 2 MHz en 54 pas prédéfinis + une fréquence quelconque dans la gamme	
Consommation	Avec l'extension afficheur/clavier	5 V / 420 mA
Logiciels PC	pour Windows, MacOSX	

Conditions du test :

Tension de test à vide	4 valeurs : 0,1 V_{eff}, 0,2 V_{eff}, 0,5 V_{eff}, 1 V_{eff} ± 5 %
Polarisation continue	Pour C : 0 à 5 V | Pour L : 0 à 50 mA

Précision du paramètre principal (R, L, C) :

Conditions*	Après calibration puis utilisation d'une fréquence de test en adéquation avec le DUT (aux hautes fréquences les composantes parasites prennent beaucoup d'importance)
Précision	Jusqu'à ± 0,1 % ±1 du dernier chiffre

Divers :

Connexions de mesure	4 fils Kelvin sur connecteurs BNC
Alimentation	5 V_{DC} ± 5 %, par le connecteur mini-USB

erreurs fécondes

Les seules vraies erreurs sont celles dont personne ne tire aucune leçon

compilation **par CJ Abate** (Elektor Labs)

Des perturbations électromagnétiques, un circuit imprimé grillé, une batterie qui explose et des relevés de capteurs non conformes : tout cela peut résulter d'une simple erreur technique. Les seules vraies erreurs sont celles dont personne ne tire aucune leçon. Pour s'améliorer, rien de tel que de tirer les leçons des erreurs des autres en plus de celles qu'on a faites soi-même. Sous le titre "erreurs fécondes" de cette nouvelle série, nous partageons les points de vue des membres de la communauté Elektor sur leurs erreurs et ce qu'ils ont tiré de leurs expériences. Avec aussi des conseils et des astuces qui vous aideront à diversifier vos compétences en matière de conception, de test et de programmation.

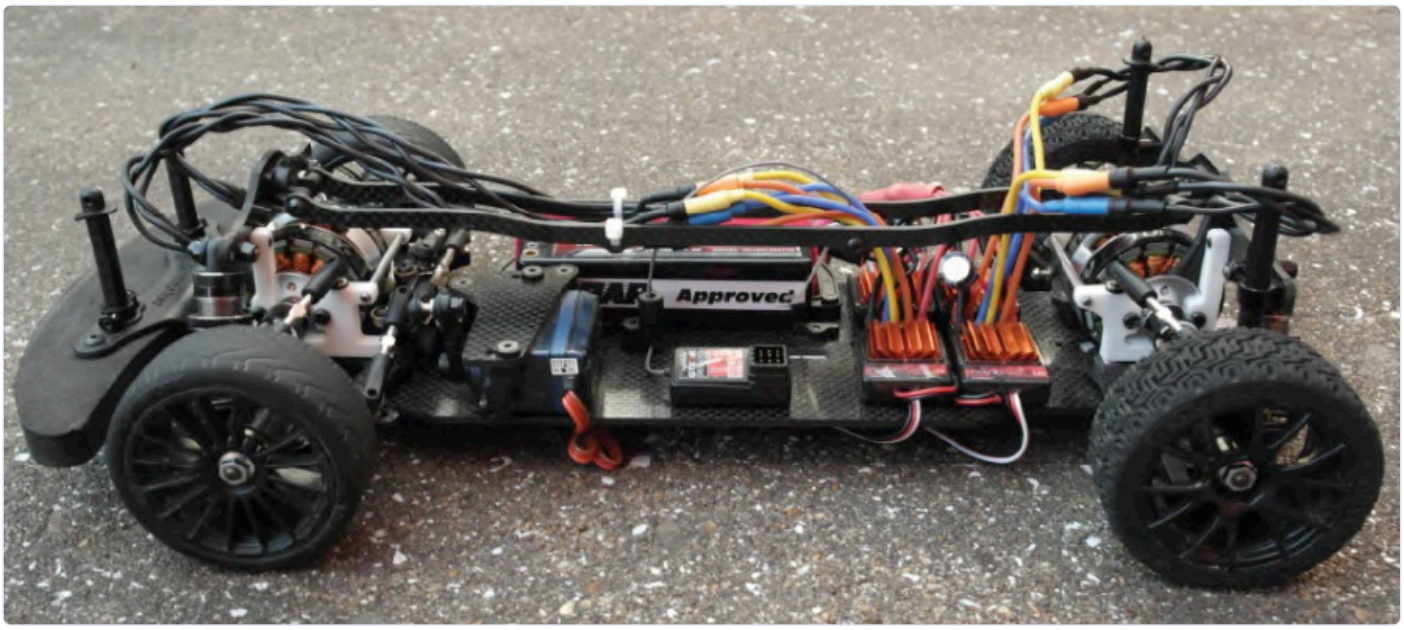


Figure 1: Rajesh Nakarja's RC project.

L'AMDE pour améliorer la fiabilité

Quelle que soit la complexité de votre projet, l'analyse des modes de défaillance et de leurs effets (AMDE) peut faire la différence. Rajesh Nakarja, ingénieur établi à Stockholm, le sait bien. Il nous raconte ici l'histoire d'un projet de radiocommande basé sur du logiciel Arduino et ce qu'il retenu de son expérience.

"Étudiant, j'adorais le modélisme RC. Ma formation en pilotage embarqué m'a amené à développer un contrôleur DSP personnalisé pour le vectoriage de couple sur des modèles réduits à l'échelle 1/10. Le véhicule à quatre roues motrices indépendantes et divers capteurs atteignait des vitesses élevées normalement réservées à la compétition sur piste (fig. 1). Grâce au traitement en temps réel, il maintenait la stabilité à pleine puissance tout en détectant la perte de traction, et se corrigeait avec des gyroscopes.

Le code déployé à partir du codeur Simulink sur une carte STM32F4 personnalisée comportait de nombreuses caractéristiques de sécurité : débordements et déficits de données, perte de données radio ou de capteurs, tous ces problèmes étaient résolus en toute sécurité et... arrêtaient le véhicule en cas de pépin. De nombreux tests et simulations ont montré que cela fonctionnait, et j'étais persuadé que la conduite du véhicule serait sans danger.

> Outils et méthodes modernes, telles que la conception modélisée, permettent l'automatisation des tests et le suivi de tout le processus.

Il ne pesait que quelques kilos, mais la vitesse et la puissance de quatre moteurs BLDC 40A impliquaient un risque d'accident dévastateur en cas de collision.

Pour commander la voiture, j'avais bricolé un contrôleur Xbox sur un pont Arduino-ZigBee, qui pilotait la voiture en temps réel et commandait l'accélérateur. Le contrôleur de la voiture était fiable et robuste, mais pas le contrôleur Arduino qui se contentait d'arrêter la voiture en cas de perte de la radio. La solution semblait assez raisonnable.
Un après-midi, lors d'un test en plein air, le contrôleur USB a perdu brièvement la connexion avec l'unité Arduino/ZigBee. Le pilote USB s'est alors planté, donnant – chose imprévue – le plein badin à la radio qui continuait d'émettre. Le véhicule s'est alors propulsé à pleine vitesse droit vers une rue très fréquentée.
Avec la longue portée des modules ZigBee utilisés, il faudrait que le véhicule soit hors de vue avant que la sécurité ne fasse son travail. Pendant que j'arrachais en toute hâte les batteries de l'Arduino, le véhicule est passé en trombe sous le nez d'une foule de gens avant de s'écraser sur un trottoir à une vitesse de plusieurs dizaines de km/h. Pas de blessé, mais sous l'impact, les batteries au lithium ont été éjectées. J'ai couru récupérer mon tas de fibre de carbone et suis rentré chez moi penaud (fig. 3).

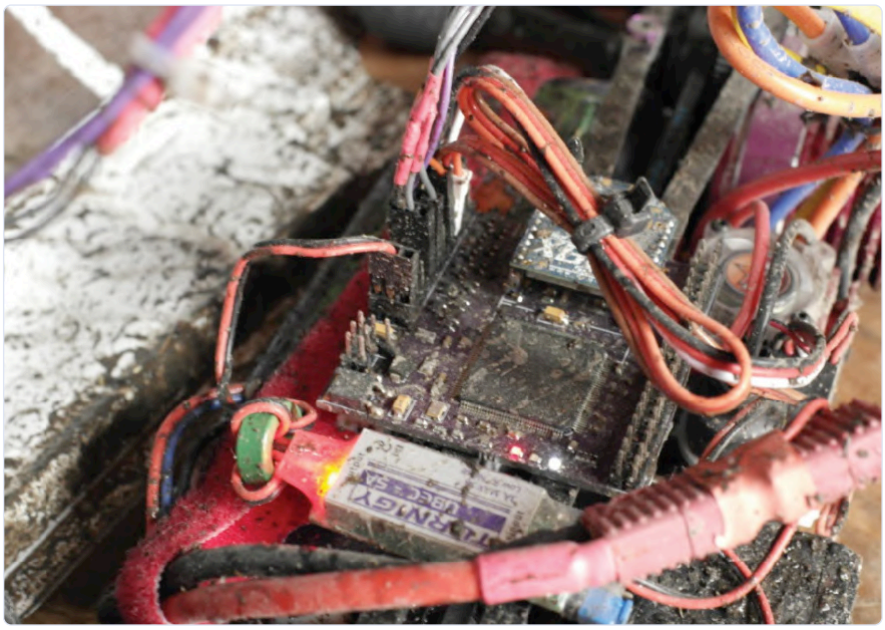


Figure 2. Les désastres ne sont pas tous inévitables.

Ce jour-là, j'ai appris que la plus simple des erreurs peut n'avoir rien d'anormal pour tous les dispositifs de sécurité d'un système. Il suffit d'un petit hic pour que tout finisse par un grand crac. Depuis, j'ai pris beaucoup plus au sérieux l'analyse des effets des modes de défaillance, et, par principe, je l'applique dès le départ à chaque projet sur lequel je travaille.
Outils et méthodes modernes, telles que la conception modélisée, permettent d'automatiser les tests. Utilisés en combinaison avec un plan de progression, ils permettent d'éviter les bourdes dès le début ou de les déceler rapidement au fil du processus. Les choses peuvent encore mal tourner, mais avec un bon plan de test on évitera toute reproduction des erreurs déjà commises. Les photos montrent le véhicule modifié et la carte de pilotage. L'électronique a été moulée pour éviter les dommages causés par la saleté ou l'eau. Une vidéo sur YouTube décrit le projet en détail [1]".

Régulation et freinage des courants forts

Lars Krüger est enseignant à Potsdam (Allemagne) et lit le magazine Elektor depuis huit ans. Il conçoit des vélos électriques depuis 1992. La régulation des courants de forte intensité, ça le connaît, de même que le test de circuits de vélos électriques, mais ça n'a pas toujours été le cas !
C'est avec des projets comme celui-ci que L. Krüger a commencé son apprentissage de la propulsion électrique dès 1994.
« Quand j'étais jeune, j'étais pressé. Une idée qui me venait le matin devait être concrétisée le soir même. Je ne m'embarrassais pas de trop de certains détails dysfonctionnels, pourvu que ça marche, à peu près... Il y a donc forcément eu beaucoup de casse, des semi-conducteurs grillés par des tensions et des courants indubitablement excessifs !
Mon premier vélo électrique date de 1992. J'avais lu le premier quart d'un livre qui m'a suffi pour me lancer dans une PWM à environ 100 Hz, une armada de 10 MOSFET "BUZ", une énorme batterie de voiture de 100 Ah et le démarreur de 2 kW d'une **Citroën CX 2.5D.**
Au premier coup d'accélérateur, le moteur a marché, je devrais plutôt dire le vélo *avait* marché. Les MOSFET étaient grillés et le vélo s'est changé en moto : il dépassait les 60 km/h, il était parfaitement inarrêtable. Inutile de couper la tension d'alimentation de la commande de largeur d'impulsion, les FET cramés ne se souciaient plus de leur tension de grille.
J'avais heureusement monté un fusible et surtout un frein assez solide

Figure 3. Avec ce modèle de vélo électrique de 1994 et quelques autres qu'il a conçus depuis, L. Krüger a beaucoup appris sur les courants de forte intensité.

pour finir par faire sauter le fusible. Étape suivante : je construis un gros interrupteur, fait de cuivre épais et d'un ressort, connecté au câble Bowden au lieu du potentiomètre. Les 10 premiers mètres sur ce vélo étaient amusants. Après il devenait incontrôlable dans les rues étroites de ma ville natale, mais j'adorais faire le pitre. Et j'ai appris que, quand on augmente la taille d'un moteur, la limitation du courant est o-bli-ga-toire. Et que cette commande doit être suffisamment rapide pour faire face au gradient de la montée du courant, au risque d'avoir des oscillations dans la boucle de régulation. Par après, j'ai étudié sérieusement les tenants et les aboutissants de la régulation, mais c'était bien moins drôle que la vie réelle. Plus tard encore, les outils de simulation comme PSpice, ou son successeur SIMetrix, m'ont permis d'acquérir une meilleure expérience. J'ai eu de moins en moins de problèmes avec la régulation des courants de forte intensité, mais dans mon labo, il n'est toujours pas interdit de griller des MOSFET de temps en temps à condition de porter des lunettes de protection.

Parfois, ça arrive par accident : une sonde de mesure dérape et provoque un court-circuit. Au fait, la photo montre mon deuxième *e-bike* de 1994. Le premier était si rapide que je n'ai même pas eu le temps de le photographier. »

Lars Krüger

MOSFET, BJT et PLL

Robert Owen à une riche expérience dans divers domaines de l'électronique. Il partage ici ce qu'il a appris un jour qu'il fallait désynchroniser périodiquement la commande de poursuite d'un projecteur vidéo à tube cathodique. Du fait de l'utilisation pourtant a priori évidente d'un transistor MOSFET, il s'est heurté à des problèmes. Heureusement, résolus sans trop de difficultés. Voyons comment.

« Il y a des années, dans une autre vie, j'ai eu à me pencher sur la séparation de la synchronisation et de la commande de suivi de vidéoprojecteurs à tube cathodique. Afin de plaire aux utilisateurs qui insistaient pour que l'image de leur magnétoscope à 200 $ soit parfaite sur nos projecteurs à 10.000 $, nous devions pouvoir interrompre la boucle de pilotage PLL pendant la commutation de la tête du magnétoscope. Heureusement, la puce de commande avait pour cela une entrée de pilotage idoine. J'ai donc inséré un transistor MOSFET comme résistance variable et assemblé une nouvelle carte. Pourtant, non seulement ce circuit n'a pas permis d'ouvrir la boucle, mais celle-ci essayait au contraire de suivre le signal de synchronisation de plus près encore ! Résultat : image déchirée en haut de l'écran. Nos projecteurs affichaient toutes les lignes de l'image, contrairement à un téléviseur, qui ne se gênait pas pour en couper bien 10 à chaque extrémité. J'ai fini par comprendre que l'entrée de pilotage réagissait à l'impulsion de courant issue de ce condensateur géant formé par la jonction grille-source du MOSFET. Il a donc suffi de remplacer ce MOSFET par un JFET ou, comme je l'ai fait, par un simple BJT pour résoudre le problème. »

Robert Owen

200303-03

PARTAGEZ VOS ERREURS

Êtes-vous prêt à partager l'expérience de vos erreurs ? Que vous soyez professionnel, familier de systèmes embarqués, ou un amateur, électronicien du dimanche, nous pensons que toutes les erreurs peuvent se révéler fécondes

www.elektormagazine.com/pages/error-analysis-submission

LIEN

[1] **"Independent Wheel Drive 1/10 Scale RC Car - Project Aelith", SiliconWitch Couture, YouTube, 2014 :** https://youtu.be/ET1HEyqEQJQ

bureau d'études – Zone D

D comme développement, débrouille et dur-à-cuire ! Trucs et astuces, bonnes pratiques et autres informations utiles

Clemens Valens (Elektor Labs)

DE L'IDÉE AU PRODUIT – 6E PARTIE

Dans la livraison précédente de cette série, nous sommes passés au stade de la production de l'électronique de notre nouveau produit. Entretemps vous avez reçu un lot de circuits imprimés. Il faut vérifier que ces cartes fonctionnent comme prévu, ce qui implique des tests. Comment s'y prendre ? Vous aviez bien sûr réfléchi à ce sujet délicat avant de lancer la production et vous y venez donc bien préparé.

QUE TESTER ?

Chaque fonction ou caractéristique de la carte doit être testée. Comment ? Comment définir un test ? Concerne-t-il chaque carte ? Est-il automatisable ? Faut-il l'automatiser ? Un test automatisé aurait-il été possible si vous y aviez pensé plus tôt lors de la conception du produit ? *Aïe !* Désolé, ça fait mal ?

TESTS INTÉGRÉS

La carte idéale se teste elle-même grâce aux tests intégrés (*built-in tests* ou BIT). Même quand ça marche, il reste à activer la procédure de test et à en interpréter les résultats, puis à séparer des bonnes les cartes défaillantes. Un test intégré peut être mis en œuvre dans les logiciels (*n'oubliez pas de tester le test* !), mais toutes les cartes ne disposent pas de logiciels et les logiciels ne peuvent pas tout tester. Une routine de test qui par exemple allume un à un tous les pixels ou segments d'un écran est utile, mais seulement s'il y a un observateur. Au fait, si dans une forêt un arbre tombe mais que personne n'est là pour l'entendre, fait-il du bruit ?

RETOUR À LA PLANCHE À DESSIN

C'est récurrent dans cette série d'articles : à chaque pas en avant, vous revenez au stade du cahier des charges. C'est encore le cas ici. Le test des cartes doit faire partie des spécifications de conception ; si vous pensez aux tests dès le début, tout devient (plus) simple et (plus) cohérent. Plus la procédure de test est simple, mieux ce sera , car les tests sont généralement effectués par des personnes ou des machines qui n'ont aucune connaissance de l'appareil testé (*device under test* ou DUT).

CONNECTEURS ET POINTS DE TEST

L'ajout sur la carte d'un connecteur de test est une solution. Le JTAG, par exemple, est une méthode populaire pour tester les circuits logiques tels que les microcontrôleurs et les FPGA implantés.

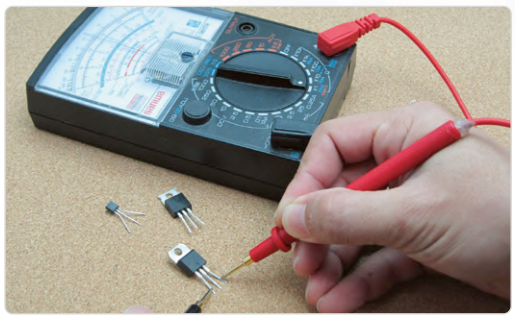
Un connecteur de test peut donner accès à des entrées et des sorties, analogiques, numériques, ou les deux, qui peuvent ainsi être excitées et lues par un testeur, humain ou machine. Si le testeur simule la pression sur un bouton et obtient un signal de réponse, cela ne lui dit pas cependant si le bouton lui-même est présent *et* fonctionne. Problème « immobilier » : le connecteur de test prend de la place et coûte de l'argent, sans oublier les composants supplémentaires nécessaires à son fonctionnement.

À la place, les concepteurs utilisent souvent des points de test. Des oscilloscopes et autres instruments similaires peuvent être programmés pour vérifier la conformité des signaux (complexes) sur les points de test. Le faible encombrement des points de test et leur liberté de placement sont un avantage évident, mais un gabarit de test spécial est nécessaire pour s'y connecter. Les instruments de test doivent être programmés et des gabarits de test doivent être mis au point et testés, ce qui n'est pas gratuit non plus.

La qualité de vos procédures de test peut faire toute la différence entre un séjour de rêve sur une île tropicale (prévenir) et la faillite (ne pas parvenir à guérir).

PROCÉDURE DE TEST APPROFONDI

Quelle que soit la manière dont les tests sont menés, il faut définir une procédure claire qui permette une décision fiable de réussite ou d'échec. La procédure doit couvrir tous les aspects de la carte et rester aussi facile que possible à appliquer, à comprendre et à exécuter, et à plus forte raison encore si les quantités de production augmentent. La mise au point d'une telle procédure n'est pas à prendre à la légère. Comme je l'ai déjà suggéré, pour réduire et les coûts et le risque d'erreurs des tests, il est primordial que le testeur n'ait besoin d'aucune ou seulement de peu de formation.



La méthode ne dépend pas seulement du nombre de produits à tester.

TEST D'ENDURANCE

Le test fonctionnel est une bonne chose, mais suffit-il ? Pensez aussi au test d'endurance. Pas forcément pour chaque carte, mais une inspection visuelle approfondie de quelques échantillons de chaque lot, puis des tests de cycles de température et d'humidité dans une chambre climatique pour vérifier s'ils survivent. Selon l'application du produit final, des tests de chute et de choc peuvent être nécessaires.

TEST DE CONFORMITÉ

Il faut s'occuper également des tests de conformité et s'y préparer. Là c'est plus facile, puisque ce n'est pas à vous d'élaborer les procédures, elles sont définies par les autorités (pensez à donner à l'organisme chargé des tests de conformité des instructions sur la manière de manipuler correctement votre produit). Le plus difficile est sans doute de rendre votre produit conforme. Là encore, cela coûte du temps et de l'argent. Il y a non seulement les frais du test de conformité lui-même, mais il faudra rémunérer quelqu'un qui déterminera quelles normes concernent votre produit et dans quel pays, et quelles sont les implications pour votre produit. Il peut être intéressant d'investir dans certains outils de test de préconformité afin de pouvoir détecter et corriger les problèmes potentiels à un stade précoce. Là encore, la préparation aux tests de conformité aurait dû être effectuée avant de commencer à faire des frais pour votre produit.

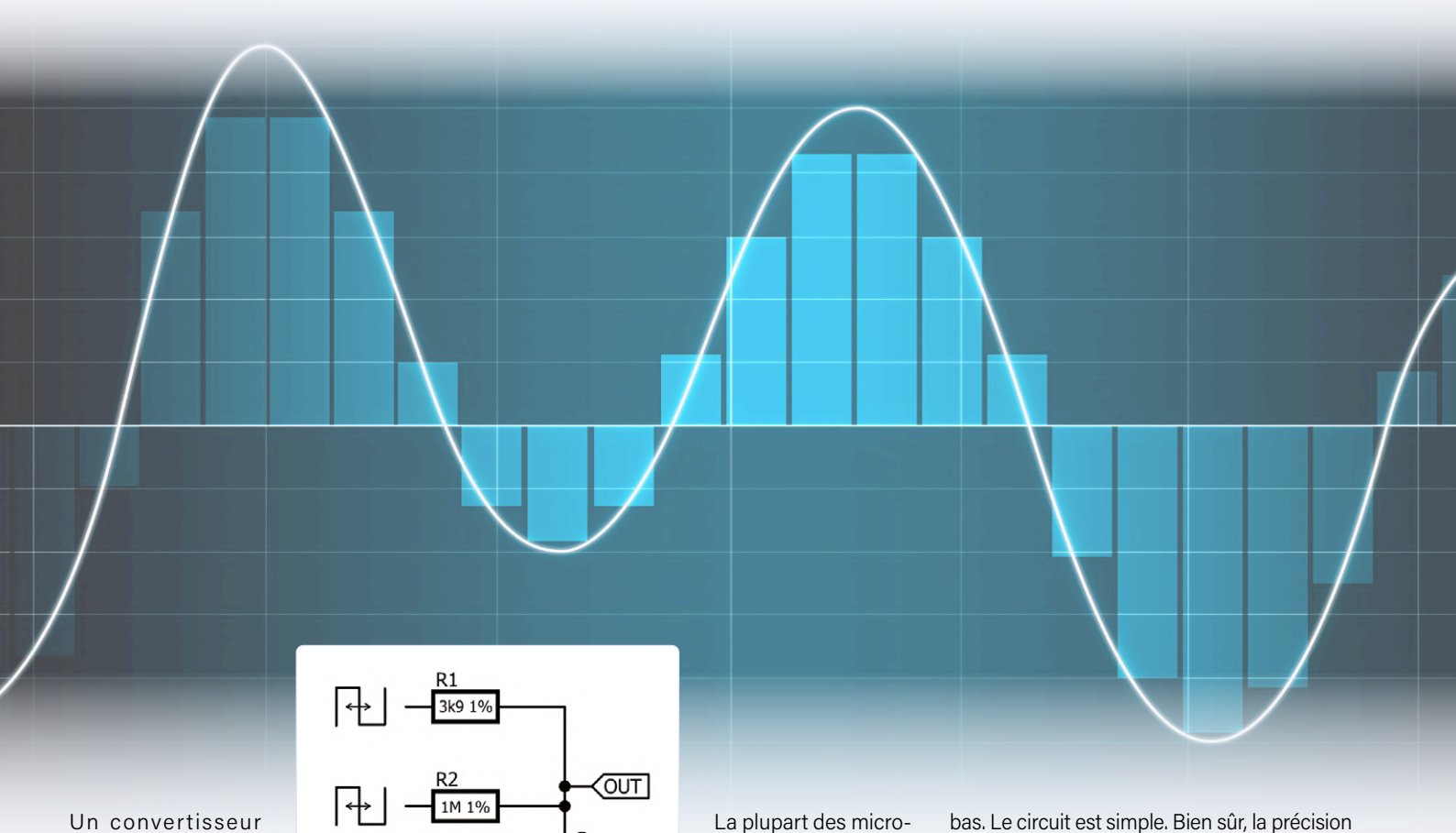
ÉCOUTER LES RÉACTIONS

Tenez compte des commentaires des personnes qui effectuent les tests que vous avez demandés, car cela peut contribuer à améliorer votre carte. Cela peut se traduire par exemple par une meilleure ergonomie, moins de réclamations, une production moins chère et/ou des tests accélérés.

LES TESTS ONT UN COÛT

Imaginer, mettre en œuvre et exécuter de bonnes procédures de test fait partie intégrante de la conception et de la fabrication de produits ; ce temps et ces efforts ont un coût.
À moins de vivre dans un monde parfait, les tests révéleront un flux de cartes défectueuses, que vous pouvez détruire ou essayer de réparer. Dans les deux cas, il y aura un coût que vous pouvez réduire en améliorant la qualité de votre carte et sa production. C'est le boulot des ingénieurs de qualité. Si vous escomptez une réduction du nombre d'appareils défectueux en simplifiant la procédure de test, vous serez mordu plus tard par le service après-vente.
L'intérêt des tests est d'éviter de mécontenter des clients qui renvoient votre produit et lui font une mauvaise réputation. Attendre l'après-vente pour s'en préoccuper coûte beaucoup plus cher que d'essayer de le faire dès le début. Cela peut même vous coûter votre entreprise.

la double MLI
améliore la qualité audio sur 8 bits



Un convertisseur numérique-analogique ou CN/A transforme des valeurs numériques en une tension analogique proportionnelle. La résolution de sortie est déterminée par le nombre de bits du CN/A. Plus son format est grand, meilleure est sa résolution. Une solution astucieuse consiste à jumeler deux DAC de faible résolution. Si nous utilisons un CN/A pour produire une valeur grossière, et un deuxième pour l'affiner («en comblant les lacunes») à condition de mettre sa sortie à l'échelle correctement. Théoriquement on obtient ainsi une résolution de 16 bits avec deux CN/A de 8 bits. En fait, la résolution sera moindre, car la mise à l'échelle sera imparfaite, à moins d'utiliser une électronique très précise, plus complexe et plus coûteuse que d'utiliser directement un CN/A à 16 bits. Cette technique n'est donc pas utilisée souvent.

La plupart des microcontrôleurs (µC) ne disposent pas de CA/N, et la pratique courante consiste à utiliser la MLI ou modulation de largeur d'impulsion (***pulse width modulation*** ou PWM) pour obtenir des signaux analogiques. Le filtrage passe-bas d'un signal MLI donne une tension proportionnelle à la largeur de l'impulsion. Ici la résolution est déterminée par la résolution du signal MLI. Arduino, par exemple, dispose de la fonction `analogWrite()` pour produire des signaux MLI avec une résolution de 8 bits. L'Arduino Uno peut produire jusqu'à six de ces signaux en même temps (sur les broches marquées d'un tilde ~).

Pour augmenter la résolution, nous recourons à la même astuce de jumelage que ci-dessus : une sortie MLI produira la valeur grossière, une autre l'affinera. La mise à l'échelle est assurée par les résistances du filtre passe-bas. Le circuit est simple. Bien sûr, la précision des résistances est cruciale, ce qui rend difficile à obtenir une résolution réelle de 16 bits, mais avec les résistances à 1 %, la résolution effective sera de presque 14 bits.

Mais, direz-vous, puisque le µC de l'Arduino Uno peut faire de la MLI sur 16 bits, pourquoi ne pas l'utiliser ? C'est une question de rapidité. Pour la même fréquence d'horloge, la MLI sur 16 bits aura la moitié du taux de sortie de la MLI sur 8 bits. Or, pour les applications audio, par exemple, c'est une fréquence de sortie ou une fréquence d'échantillonnage élevée qui est intéressante.

Alors, pourquoi ne pas mettre trois signaux PWM en parallèle, voire plus ? Essayez, mais vous finirez par vous heurter aux limites des logiciels et aux problèmes de complexité du matériel. En bref, l'effort requis n'en vaut probablement pas la peine. La double MLI sur 8 bits est un bon compromis. Vous trouverez une bonne explication et une analyse du double PWM à l'adresse suivante :

www.openmusiclabs.com/learning/digital/pwm-dac/dual-pwm-circuits/

charge électronique
Siglent SDL1020X-E

Harry Baggen (Pays-Bas)

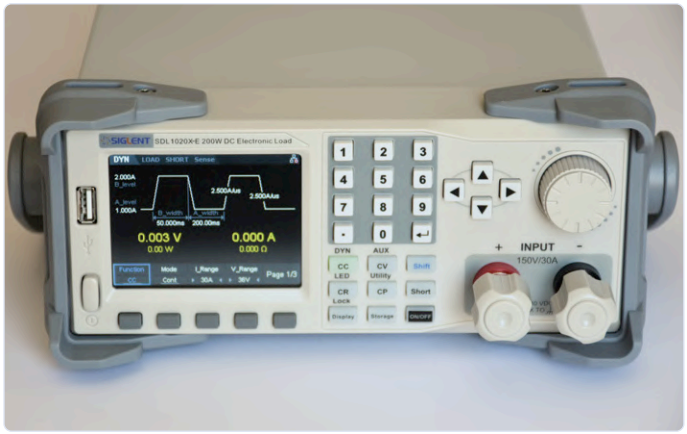
Une charge électronique programmable est l'outil par excellence pour tester les alimentations, les piles et les accumulateurs. Une telle charge est capable non seulement de se comporter comme une résistance fixe, mais d'agir également comme une charge à courant constant ou tension constante. Elle peut même commuter instantanément entre différentes valeurs de charge. Le Siglent SDL1020X-E offre toute une série de possibilités de ce type et convient pour des tensions jusqu'à 150 V et des courants jusqu'à 30 A, avec une dissipation maximale de 200 W.

Lorsqu'on teste un circuit d'alimentation, on veut savoir s'il reste stable sous différents types de charge, s'il réagit rapidement aux changements de charge et s'il produit de faibles niveaux d'interférence. Il semble que la demande de dispositifs capables de charger une source d'alimentation de différentes manières est en hausse à en juger par l'offre de charges électroniques des fabricants chinois d'appareils de mesure. Depuis moins d'un an, Siglent a également une série de charges électroniques dans son programme, la série SDL1000X. J'ai testé le SDL1020X-**E**. C'est la plus petite version dont la puissance maximale est de 200 W. Il existe une version de 300 W. Les deux appareils existent en version sans le suffixe -**E**, laquelle offre une plus grande précision de lecture.

Un boîtier solide

J'ai déjà eu l'occasion de signaler la solidité des boîtiers robustes de Siglent constatée aussi ici. Cette qualité d'autant plus frappante ici que le boîtier est assez long, à savoir 39 cm. Il vous faudra une étagère sur laquelle il reste encore pas mal de place.

Le boîtier est beaucoup plus profond que celui de beaucoup d'autres appareils Siglent.



Le panneau frontal avec l'affichage montrant le réglage des formes d'onde dynamiques.

Le panneau arrière comporte une série d'accessoires et une grille d'aération.

Les grandes bornes à vis sont robustes, mais excluent malheureusement l'utilisation de fiches bananes.

Le panneau frontal réunit tous les boutons, deux terminaux de connexion robustes et un grand écran LCD de presque 9 cm affichant toutes les informations. En dessous de l'écran se trouvent cinq boutons dont la fonction actuelle est affichée à l'écran.

À l'arrière, on trouve le cordon, un connecteur USB, LAN et RS232 et deux prises BNC qui fournissent un signal de mesure pour la tension et le courant de la charge (pour la connexion à un oscilloscope p. ex.). En bas se trouve un bloc de connexion avec une série de bornes dont les plus importantes sont les entrées de détection et l'entrée de déclenchement. À gauche, des fentes d'aération pour le refroidissement du grand tunnel de refroidissement qui s'étend à l'intérieur sur presque toute la longueur de l'appareil et, à l'avant, un ventilateur thermostaté. Pour les charges plus faibles, le SDL1020X-E est donc pratiquement silencieux.

Options

Les possibilités du SDL1020X-E sont nombreuses, elles dépassent le cadre de ce banc d'essai. Je me limiterai aux plus importantes. Pour approfondir, avant d'acheter, n'hésitez pas à consulter la fiche technique ou le manuel.

Tout d'abord les bases. Vous pouvez choisir entre courant constant (CC), tension constante (CV), puissance constante (CP) ou résistance constante (CR). L'écran affiche toujours la tension, le courant, la puissance et la résistance mesurés. Il existe deux plages de courant (5/30 A) et deux plages de tension (36/150 V) principalement liées à la précision des mesures. Une fonction séparée de test de la batterie permet, entre autres, de décharger avec un courant, une résistance ou une puissance fixes jusqu'à une tension de seuil définie, l'écran affichant alors la capacité et la puissance de décharge.

Les possibilités de test dynamique, qui ne sont pas si faciles à réaliser avec des méthodes simples, sont particulièrement intéressantes. Lorsque la fonction de test dynamique (DYN) est activée, une forme d'onde carrée apparaît sur l'écran. Cela montre que vous pouvez faire passer l'instrument d'une valeur à l'autre, et que vous pouvez régler tous les paramètres vous-même : la valeur de charge faible pour CC/CV/CR/CP, la valeur élevée, le temps faible, le temps élevé, la pente du front montant et du front descendant. Ceci est possible

La configuration du test.

jusqu'à 25 kHz, ce qui suffira certainement pour la plupart des tests. Le test dynamique peut également commencer sur un signal de déclenchement externe ou manuel.
Grâce aux fonctions *List* et *Program*, l'utilisateur peut composer des formulaires de chargement plus complexes en plusieurs étapes ou pas. Cela offre des possibilités extrêmement puissantes. Avec la fonction List (max. 100 pas par forme d'onde), le type de charge (par exemple courant constant) est identique à toutes les étapes ; avec la fonction Program (max. 50 pas par forme d'onde), vous pouvez également, à chaque étape, passer par exemple de courant constant à résistance constante. Avec les deux fonctions, vous pouvez saisir tous les paramètres par étape dans une sorte de tableau.

Par défaut, toutes les valeurs mesurées sont affichées à l'écran, en mode dynamique, l'onde carrée avec toutes les valeurs réglées apparaît. Si vous passez en mode DISPLAY, l'évolution dans le temps d'un paramètre sélectionné est affichée sous forme graphique.

Utilisation pratique

Après avoir branché une alim de lab, un oscillo et un multimètre précis, j'ai pu commencer à tester. La précision des valeurs affichées était très bonne, généralement dans la limite de 1 mA/mV. Les grandes bornes de connexion offrent suffisamment d'espace pour connecter fermement les câbles directement ou à l'aide de fourches. Pourquoi pas des fiches bananes, pourtant bien pratiques avec des courants jusqu'à quelques ampères ? Je l'ignore.
Après avoir essayé des choses simples, je me suis quand même mis à lire le manuel, car le nombre de réglages est énorme. Je n'en ai encore mentionné qu'une partie. Grâce au manuel, j'ai ainsi découvert que, pour qu'elles fournissent des signaux de mesure, les deux sorties de moniteur à l'arrière doivent être activées dans un sous-menu. Malheureusement, ce réglage est remis à zéro lorsque l'appareil est éteint et doit être réactivé la fois suivante.

J'ai également découvert que la pente maximale des flancs en mode de test dynamique dépend de la gamme de courant sélectionnée. En effet, dans la gamme 30 A, elle est de 2,5 A/µs, dans la gamme 5 A, elle n'est que de 0,5 A/µs.
Mis à part ces petits détails, auxquels il serait possible de remédier au moins partiellement par une mise à jour du micrologiciel, le SDL1020X-E est un appareil avec lequel on peut travailler agréablement, avec la qualité habituelle de *Siglent*. Il est également possible de commander cette charge électronique depuis votre PC, soit au moyen de commandes SCPI, soit par un pilote *LabView*. On m'a dit que *Siglent* prépare un programme sous Windows.

Conclusion

Si vous avez l'usage régulièrement d'une charge électronique, p. ex. pour tester des alimentations ou des batteries, un appareil tel que le **SDL1020X-E** complétera votre équipement de mesure. Il faut déjà bien réfléchir pour imaginer des options de test qu'il n'offre pas. L'écran affiche clairement toutes les informations nécessaires, au point que dans de nombreux cas on se passera même d'un oscilloscope et/ou d'un multimètre. L'instrument est bien pensé, bien conçu et proposé à un prix qui correspond à ses qualités et à son utilité !

200323-03

PRODUITS

› **charge électronique Siglent SDL1020X-E**
https://bit.ly/2Xf1ScT

alimentation haute tension
avec traceur de courbes

Des tensions régulées jusqu'à 400 V & des courbes caractéristiques de tubes

Rainer Schuster (Allemagne)

Cette alimentation HT universelle fournit non seulement les tensions d'alimentation élevées pour des circuits à tubes, mais peut également servir de traceur de courbes caractéristiques pour les tubes et les semi-conducteurs. Tout cela avec un Raspberry Pi et un écran tactile de 7 pouces !

Dans la liste des outils indispensables à l'électronicien, l'alimentation de labo arrive en deuxième, après le fer à souder et à égalité avec le multimètre. Pour la plupart des applications, les tensions dont vous avez l'usage restent sous le seuil des 60 V et pour mesurer en ampères l'intensité des courants, un seul chiffre suffit. Sauf si vous fricotez avec des tensions beaucoup plus élevées, comme c'est le cas si vous vous intéressez aux tubes.
Et tant qu'à réaliser une alimentation pour des tensions plus élevées, autant utiliser les moyens modernes pour obtenir quelques fonctions supplémentaires intéressantes. Au rang des moyens modernes se trouve le Raspberry PI. Cependant, pour cette application, les exigences en matière de puissance de calcul sont modestes. Il n'est donc pas

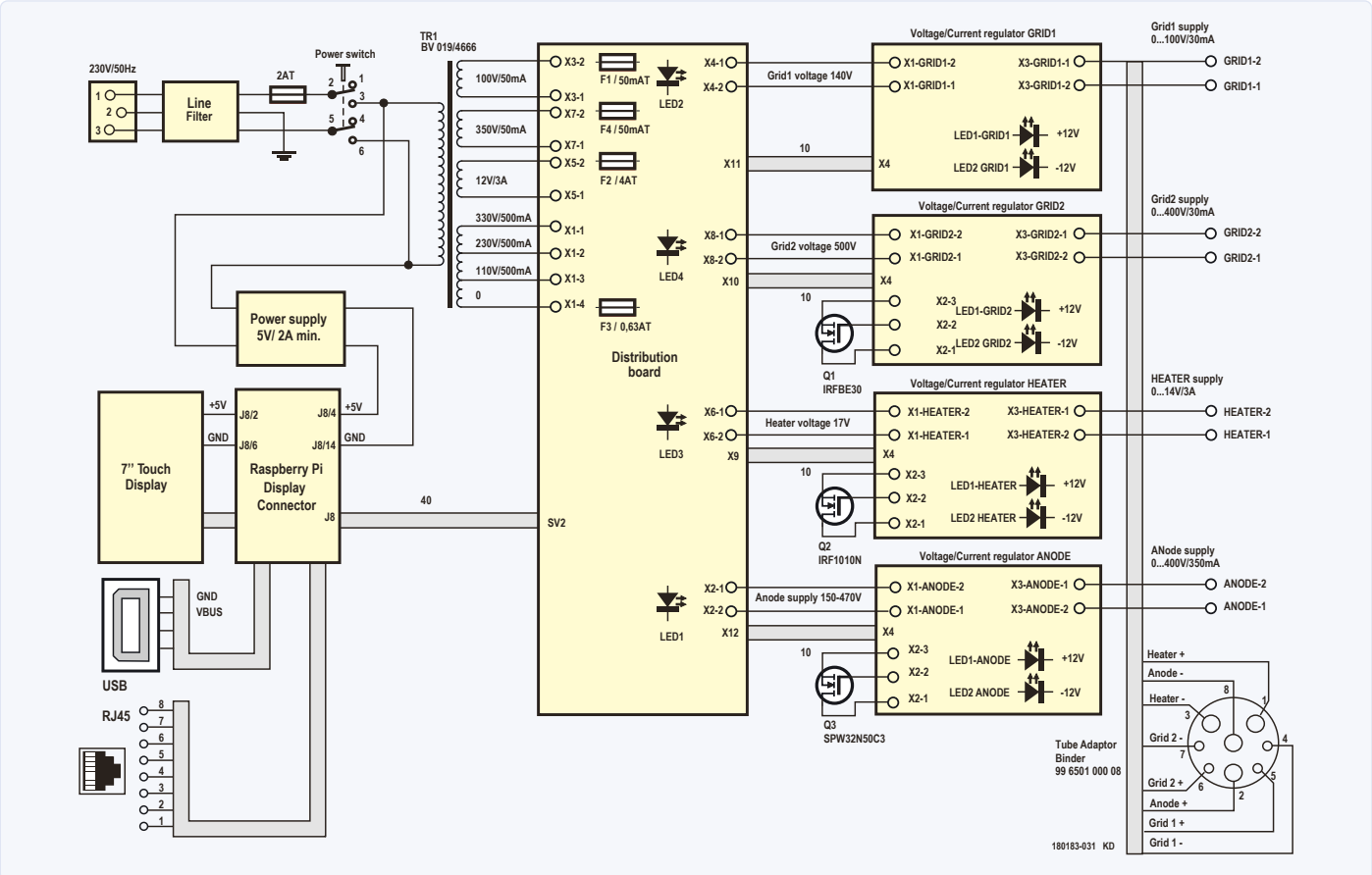


Figure 1. Synoptique de l'alimentation haute tension, avec l'interconnexion des différentes sections.

nécessaire que ce soit la dernière version 4. Un RPi 3 ou même un 2 feront l'affaire. Avec un écran tactile de 18 cm , vous serez parfaitement dans l'air du temps. Parmi les fonctions supplémentaires utiles, citons encore le traceur de courbes caractéristiques de tubes et de semi-conducteurs, facilement mises en œuvre par le logiciel.

Matériel

Bien sûr, on attend d'une telle alimentation qu'elle fournisse non seulement une tension élevée (qui n'est pas à proprement parler une *haute* tension, puisqu'elle n'atteindra ni ne dépassera 1,5 kV CC). Si vous voulez alimenter des tubes, vous aurez besoin d'une tension de chauffage en plus de la tension de l'anode. Et pour les tracés de courbes caractéristiques, vous lui demanderez des tensions auxiliaires (grilles de contrôle et grille-écran indispensables). Toutes ces tensions et tous ces courants sont réglables.
La commande de la tension et du courant des différentes alimentations subalternes est constituée de quatre unités indépendantes et galvaniquement isolées qui peuvent fournir les tensions ou courants suivants :

- anode : 0...400 V ; 0...300 mA
- grille de contrôle : 0...100 V ; 0...30 mA
- grille-écran : 0...400 V ; 0...30 mA
- chauffage : 0...14 V ; 0...3 A

La figure 1 montre le circuit global de l'alimentation, c'est-à-dire le câblage de ses différentes parties. Les différentes fonctions de l'alimentation sont situées sur la carte de distribution et les cartes auxiliaires.

Distribution

L'ensemble dit de distribution est composé des redresseurs, des condensateurs de lissage et des fusibles pour les différentes cartes de commande. Le connecteur SV2 à 40 broches reliera l'alimentation HT aux ports d'E/S du Raspberry Pi et distribue les signaux SPI par X9...X12 aux cartes auxiliaires de commande.
En fonction de la tension de sortie, les relais K1 et K2 commutent sur les enroulements du transformateur 110, 220 ou 330 V pour réduire la dissipation de puissance dans le transistor en série. La commutation est effectuée par un logiciel au moyen des broches GPIO 20 et 21 du RPi (**fig. 2**).

Régulation

Chaque carte de commande comporte des convertisseurs A/N et N/A pour le réglage et la mesure de la tension et du courant, qui communiquent avec le Raspberry Pi par l'interface SPI. Les signaux SPI sont isolés électriquement du mini-ordinateur.

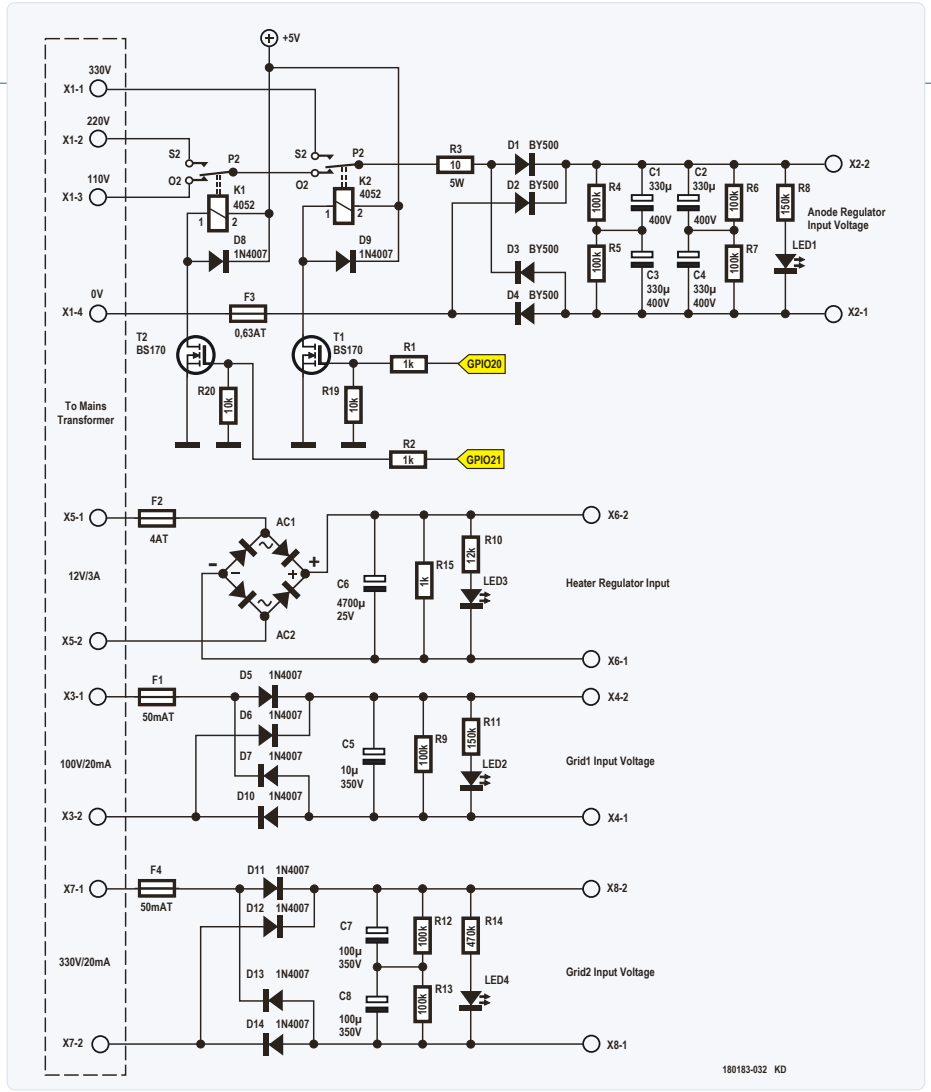


Figure 2. Le schéma du circuit de distribution.

Principe du circuit

Le synoptique (**fig. 3**) montre le MOSFET T2 connecté comme régulateur en série de la tension de sortie. Pour permettre la commande de T2 par des amplis op ordinaires sous les tensions élevées que l'on trouve ici, le pôle positif de la tension de sortie est relié à la masse (GND) de l'alimentation de l'ampli op, qui fournit une tension symétrique de ±12 V à cet effet.

Pour fixer les valeurs de consigne de courant et de tension, on fait appel au MCP4812, un CN/A à 2 voies avec une résolution de 10 bits. Ce CN/A a une tension de référence interne de 2,048 V, qui peut être doublée (4,096 V) par commande SPI. La tension de sortie est divisée par R17/R18 et la tension sur R18 est numérisée et envoyée au Raspberry Pi comme valeur réelle par le convertisseur A/N à 12 bits LTC1298.
Pour la commande du courant, la chute de tension sur R1 est amplifiée cinq fois par un ampli op dont la tension est ensuite transmise au Raspberry Pi comme intensité réelle du courant par le deuxième canal du convertisseur A/N.
Pour l'isolation galvanique des signaux SPI MOSI, MISO, SCLK et LDAC ainsi que des lignes de sélection de circuit intégré pour les convertisseurs A/N et N/A, on fait appel à des optocoupleurs rapides de type 6N137, pilotés par un 74HC04.
L'alimentation de toutes les cartes de commande doit également être isolée galvaniquement. À cet effet, la sortie 5 V du Raspberry Pi alimente un convertisseur CC/CC TMA0512D par carte, qui délivre les ±12 V nécessaires pour les amplis op.
Les valeurs réelles de tension et courant ne sont pas seulement présentes aux entrées du convertisseur A/N, mais aussi en retour aux deux amplis op pour la régulation de tension

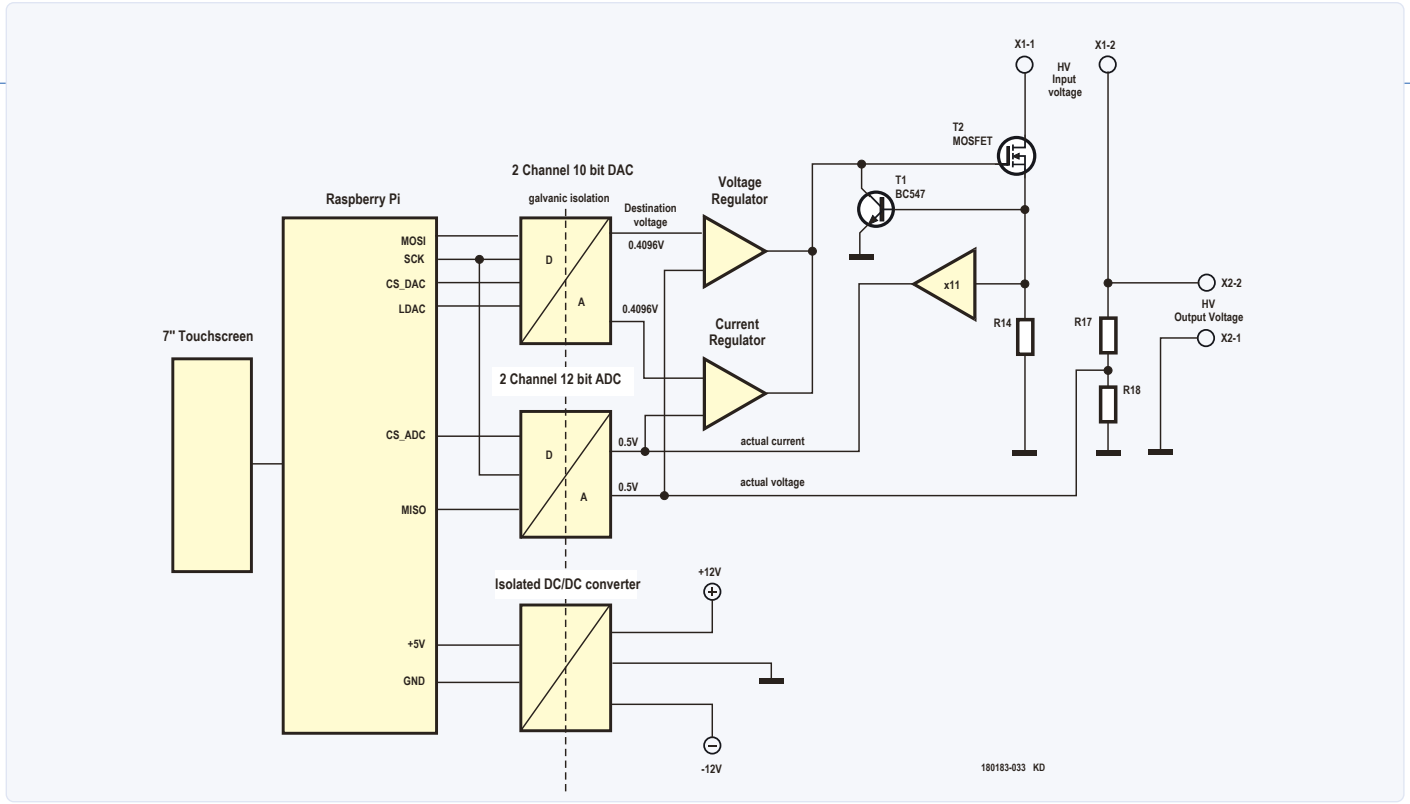


Figure 3. Le circuit principal de l'alimentation haute tension.

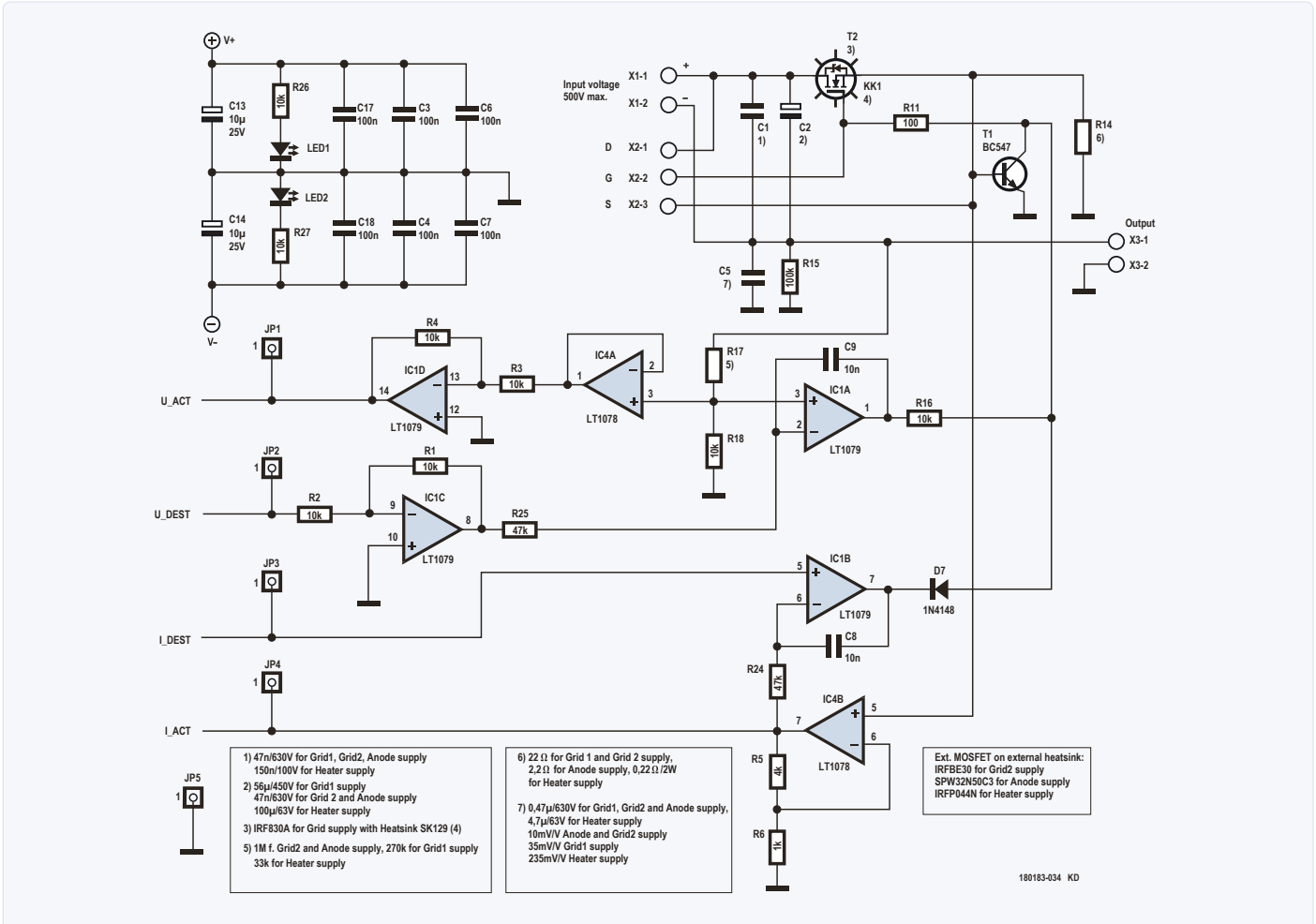


Figure 4a. Circuit de contrôle effectif de la carte de commande.

et de courant. Ces amplis op comparent les valeurs réelles avec les valeurs de consignes respectives des convertisseurs N/A. Le signal de correction de l'amplificateur du régulateur de tension qui en résulte commande alors la grille de T2. La régulation du courant entre en jeu si l'intensité mesurée dépasse la valeur de consigne. Il s'agit donc d'une commande à tension constante avec limitation de courant réglable.

En cas de court-circuit franc à la sortie, le comportement des régulateurs aura pour conséquence un bref dépassement de la valeur de consigne par le courant de sortie. Dans ce cas, la carte peut quitter la zone de sécurité dite SOA (*Safe Operating Area*) de T2 pour la tension anodique. Pour éviter cela, T1 limitera rapidement et radicalement le courant de sortie à un maximum de 0,7 V / R14 = 0,31 A. La puissance dissipée est alors brièvement portée à 465 V * 0,31 A = 144 W jusqu'à ce que la tension d'entrée soit commutée à 110 V par le logiciel. Selon sa fiche technique, T2 tolère une dissipation de puissance presque deux fois plus forte.

La **figure 4** montre le circuit des cartes de commande. La régulation du circuit a été simulée avec LTSpice. Le fichier *HVRegulator.asc est disponible* comme téléchargement gratuit sur la page de cet article [1] sur le site d'Elektor. Toutes les cartes de commande ont la même disposition, mais avec des valeurs différentes pour les différentes tensions et courants (voir **tableau 1**).

Raspberry Pi

Le Raspberry Pi qui avec son écran tactile de presque 18 cm assure la commande de régulation est alimenté lui-même par une toute petite

Tableau 1.

CARTE	C1	C2	C5	T2	R14	R17
anode	47 nF 630 V	47 nF 630 V	0,47 uF 630 V	SPW32N50C3 (KK ≤0,7 k/W)	2,2 Ω 1 W	1 MΩ
grille de contrôle	47 nF 630 V	56 uF 450 V	47 nF 630 V	IRF830A (KK SK129)	22 Ω	270 k
grille-écran	47 nF 630 V	47 nF 630 V	47 nF 630 V	IRFBE30 (KK ≤0,7 k/W)	22 Ω	1 M
chauffage	150 nF 100 V	100 nF 63 V	4,7 uF 63 V	IRFP044 (KK ≤0,7 k/W)	0,22 Ω 2 W	33 k

* R14 = 4,096 V / (5 * I_{max})
** R17 = U_{max} * 2,441 kΩ / V – 10 kΩ

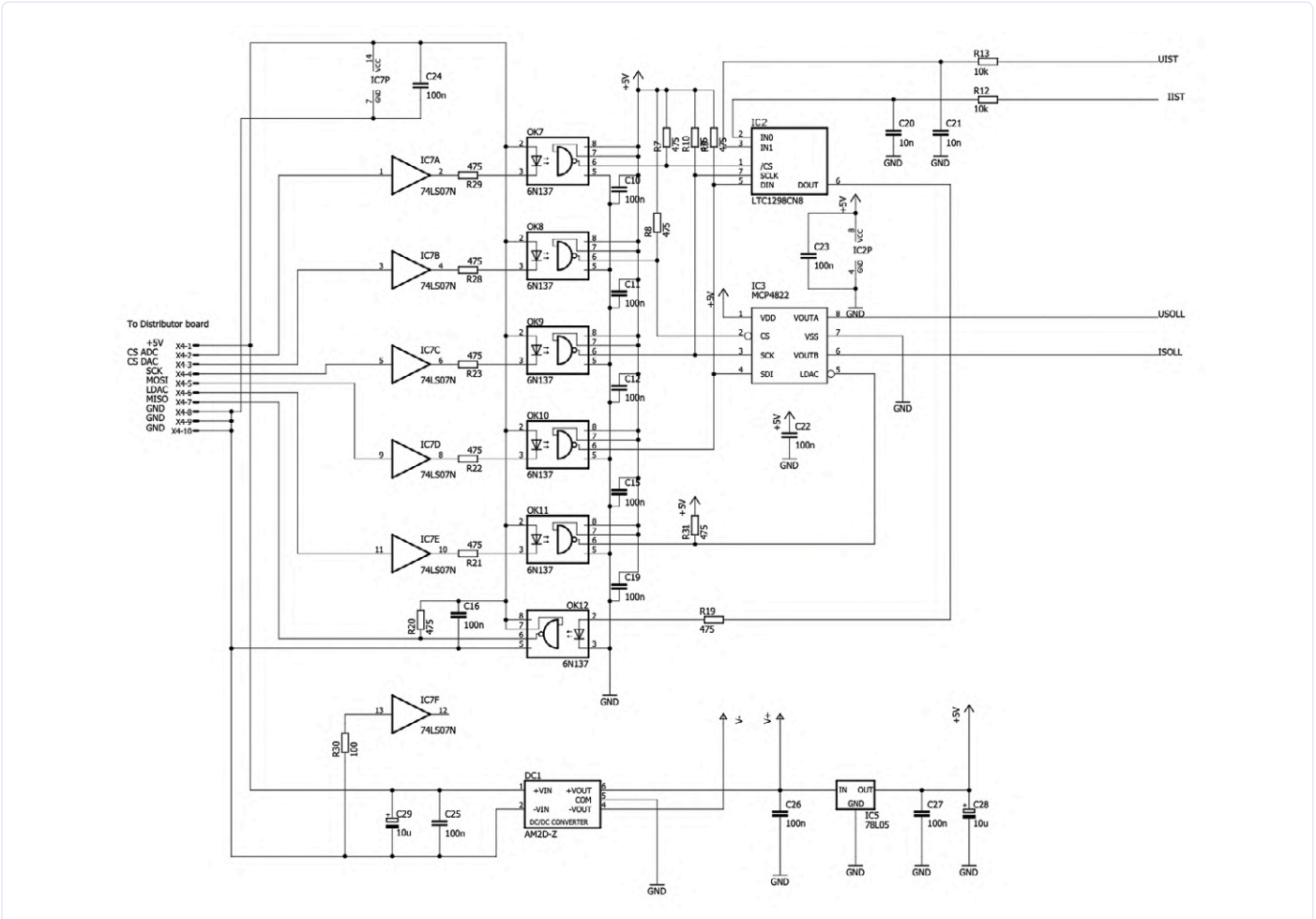


Figure 4b. Connexion de carte de commande à la carte de distribution.

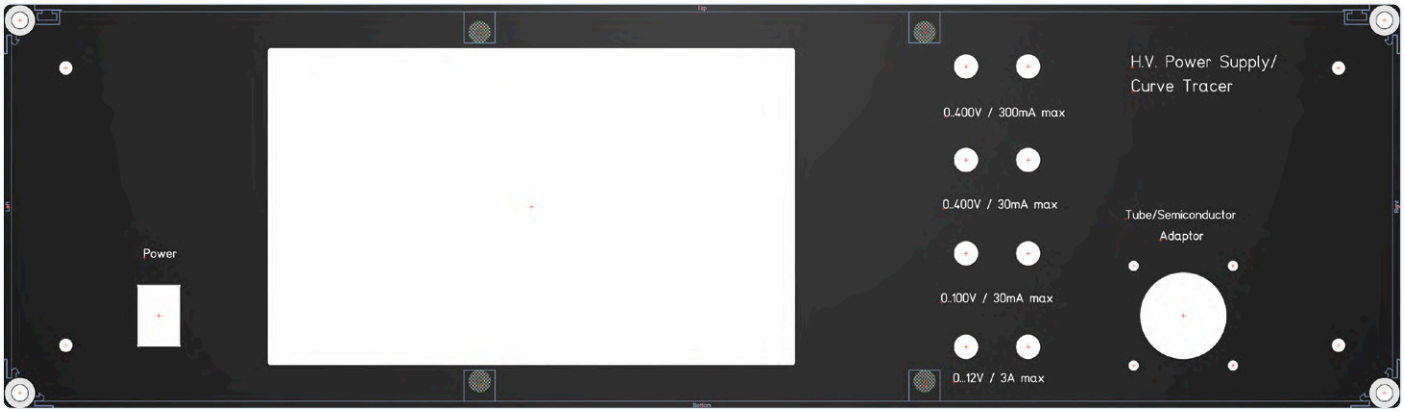


Figure 5. La face avant dessinée avec le programme FrontDesign.

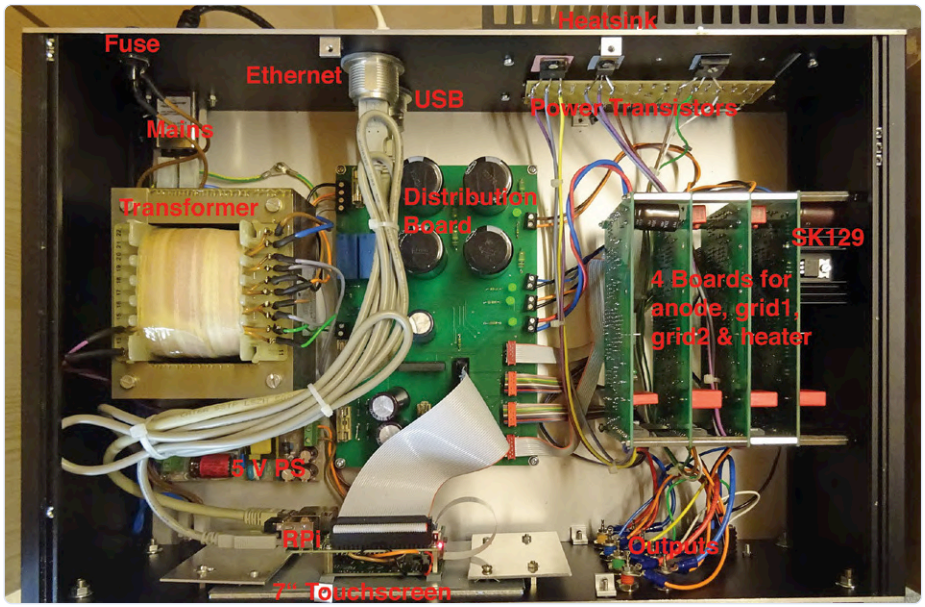


Figure 6. Les entrailles du prototype.

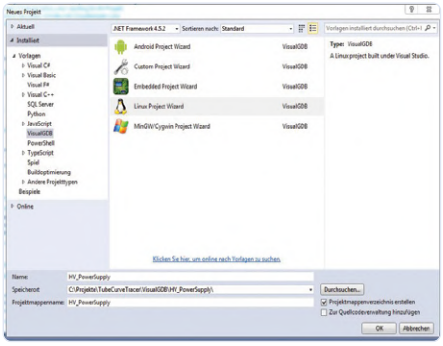


Figure 7. Paramétrage de VisualGDB en tant que projet Linux.

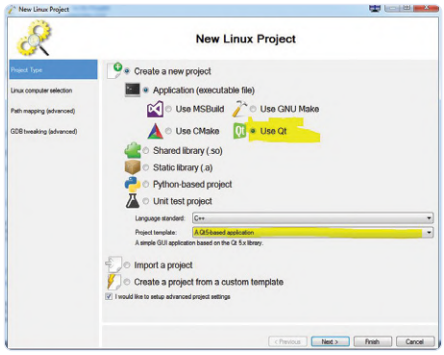


Figure 8. Il s'agit d'un système basé sur Qt5.

alimentation à découpage de 5 V – 2 A (3 A pour le RPi 4). La tension de sortie de cette alimentation doit être connectée directement au connecteur J8, car la chute de tension sur le connecteur micro-USB est trop forte.

Construction

L'appareil a été monté dans un boîtier profilé, dessiné avec le programme FrontDesign, disponible gratuitement auprès de la société Schäffer [2] pour Windows, MacOS et Linux. La **figure 5** montre la façade. Les fichiers correspondants sont téléchargeables [1]. La **figure 6** montre le prototype vu de haut. On distingue notamment les quatre cartes de commande similaires sur la droite.

Logiciels

Le logiciel Raspberry Pi a été créé en C++ pour Visual Studio de Microsoft [3]. Pour que le code fonctionne sur le Raspberry Pi, le compilateur croisé VisualGDB [4] est nécessaire, pour lequel il existe une version de test de 30 jours. Qt Designer [5] a permis de créer l'interface graphique.

Le fichier projet *HV_PowerSupply.sln* joint au téléchargement [1] contient les paramètres nécessaires pour VisualGDB (**fig. 7**). Ici, vous devez indiquer qu'il s'agit d'un projet Linux. Le nom du projet et le répertoire du projet doivent aussi être précisés.

L'étape suivante consiste à définir le système comme un système basé sur Qt5 (**fig. 8**). Suit le paramétrage selon que le logiciel est créé sur le PC local ou sur le système cible (Raspberry Pi) (**fig. 9**). J'ai choisi le PC local, sinon le Raspberry Pi doit être allumé et connecté au PC pendant l'exécution du compilateur de liens. Puisque la bibliothèque WiringPi [6] est utilisée, elle doit également être spécifiée (**fig. 10**).

Une fois le logiciel créé, *HV_PowerSupply* se trouve dans le dossier de débogage du projet sous la forme d'un fichier exécutable sur le Raspberry Pi. Le fichier doit maintenant être copié dans le répertoire */home/pi/HV_PowerSupply* du Raspberry Pi, p. ex. avec WinSC [7]. Le fichier HV-PowerSupply-Icon *Tube.png* doit également être copié dans ce répertoire.
Pour lancer l'application depuis le bureau, le fichier *HV_PowerSupply.desktop* doit être copié dans */home/pi/desktop.* Si le programme doit démarrer automatiquement après la mise en marche, ce fichier doit également se trouver dans le répertoire */home/pi/.config/autostart.*
Pour un affichage correct des icônes haut/bas sur les boutons utilisés, il est nécessaire de copier les fichiers *collapse-arrow.png* et *expand-arrow.png dans le* répertoire */home/pi.* Si vous souhaitez afficher l'icône du tube dans la barre des tâches, le fichier *Tube.png* doit également être placé dans */home/pi.*
Si vous souhaitez utiliser le logiciel sans modification, il vous suffit de copier les fichiers du téléchargement dans le Raspberry Pi comme décrit.

Fichiers source

Si vous souhaitez modifier le logiciel, vous devez installer les progiciels décrits ci-dessus et copier les fichiers source de ce projet dans le répertoire */VisualGDB/HV_PowerSupply/HV_PowerSupply.* Le projet se compose des fichiers source suivants :

- *HV_PowerSupply.cpp* : Programme principal (*MainWindow.cpp* est lancé).
- *MainWindow.cpp*, *MainWindow.h*, *ui_MainWindow.h* : Fichiers source pour l'alimentation
- *CurveTracerWindow.cpp*, *CurveTracerWindow.h*, *ui_CurveTracerWindow.h* : Fichiers source pour le traceur de courbes de tubes
- *TransistorCurveTracerWindow.cpp*, *TransistorCurveTracerWindow.h*, *ui_TransistorCurveTracerWindow.h* : Fichiers sources pour le traceur de courbes de semi-conducteurs
- *Hardware.cpp*, hardware.*h* : fonctions partagées pour contrôler les convertisseurs D/A et A/D, utilise le *câblage de la bibliothèquePi*
- *Qcustomplot.cpp*, *qcustomplot.h* : Bibliothèque pour la représentation graphique des courbes caractéristiques. Pour s'assurer que cette bibliothèque est correctement traduite, la ligne

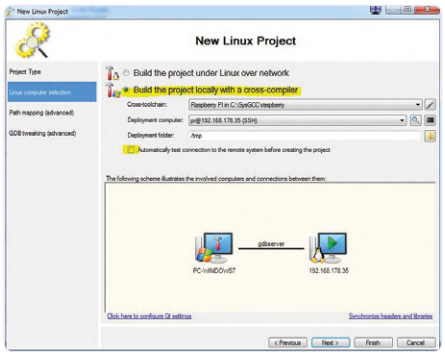


Figure 9. Logiciel pour PC ou système cible (RPi) ?

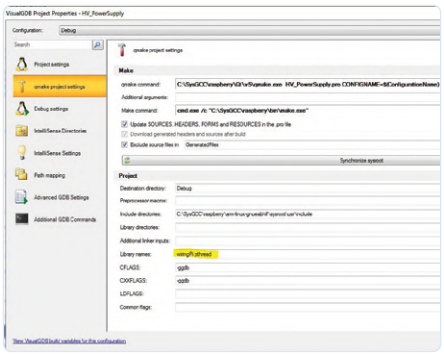


Figure 10. La bibliothèque WiringPi.

```
QT += core gui xml printsupport
```

doit être ajoutée au fichier *debug.pro.*

Les fichiers *ui_XXXX.h* sont produits par Qt Designer après que les interfaces graphiques des différents écrans ont été dessinées. La **figure 11** montre l'écran de l'alimentation. Le code source est produit en sélectionnant *Form-> Generate code* dans le menu. Il suffit ensuite de le stocker dans le répertoire source.

Déroulement du programme

Lorsque le programme démarre, l'écran de l'alimentation s'affiche et le matériel est initialisé. Toutes les tensions et tous les courants sont initialement réglés sur 0. Il est maintenant possible de régler les tensions et les courants des différentes sorties. Cela peut se faire avec les boutons haut/bas de l'écran tactile, par une combinaison clavier/souris externe via USB ou par un clavier virtuel tel que le *Matchbox Keyboard* [9] (**fig. 12**).

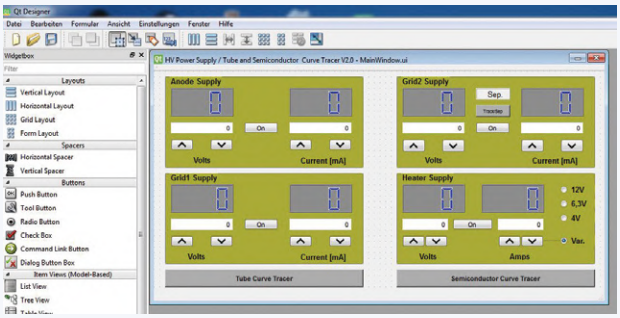


Figure 11. Maquette de l'écran de l'alimentation HT dans Qt Designer.

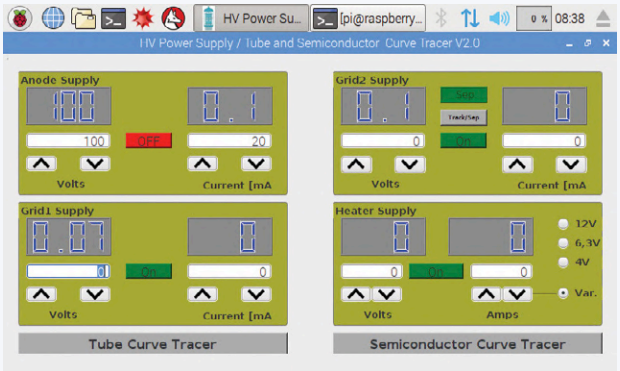


Figure 12. Interface graphique de l'alimentation HT.

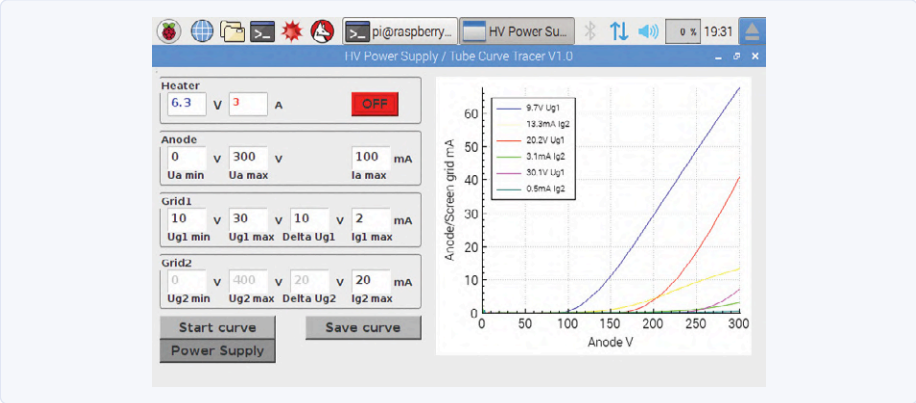


Figure 13. Interface graphique du traceur de courbes pour tubes.

Si la tension de la grille de l'écran doit suivre celle de l'anode, activez le mode de suivi avec le bouton *Track/Sep*. La tension de chauffage peut être réglée de 0 à 14 V avec *Var* ou être réglée sur les valeurs prédéfinies 4/6,3/12 V. Les boutons *On/Off* permettent *d'*activer ou de désactiver les différentes sorties individuellement.

Enregistreur de caractéristiques de tubes

Le bouton *Tube Curve Tracer* active l'écran du traceur de courbes de tubes. Au début, toutes les tensions et tous les courants sont réglés sur 0.

Dans ce mode, les courbes caractéristiques des tubes sont enregistrées et, si nécessaire, sauvegardées sous forme de tableaux avec l'extension *.csv,* afin qu'elles puissent être lues directement (p. ex. Excel). Des valeurs minimales et maximales pour l'anode et la tension de commande peuvent être spécifiées pour la courbe caractéristique. Des valeurs maximales peuvent être fixées pour le courant de l'anode, de la grille de contrôle et de la grille de l'écran. La tension de la grille de l'écran suit la tension de l'anode. Un maximum de cinq courbes caractéristiques peut être affiché.

La **figure 13** montre les courbes caractéristiques enregistrées d'un tube EL34. Les courbes contiennent le courant d'anode en fonction de la tension d'anode avec la tension du réseau de commande comme paramètre. Les courants de grille d'écran correspondants sont également enregistrés.

Les données de la courbe caractéristique peuvent être enregistrées en appuyant sur le bouton *Enregistrer la courbe.* Une boîte de dialogue permettant d'enregistrer le fichier de courbe s'ouvre alors et vous pouvez l'enregistrer dans un fichier portant l'extension *.csv.* Les fichiers peuvent être copiés sur le PC avec, par exemple, WinSCP et traités ultérieurement avec Excel. Le téléchargement [1] contient le fichier d'exemple *EL34.csv.*

Pour tester les tubes de l'étage de puissance EL34, j'ai construit un adaptateur (fig. 15) qui peut être facilement connecté au traceur de courbes par un câble (**fig. 14**).

Avec d'autres socles, vous pouvez construire des adaptateurs pour d'autres types de tubes selon le même principe. Vous pouvez aussi connecter le tube à tester directement aux bornes de l'alimentation.

Courbes caractéristiques de semi-conducteurs

Le bouton *Semiconductor Curve Tracer* affiche l'écran du même nom. La **figure 16** montre un exemple du comportement d'un transistor bipolaire bloqué à base ouverte.

Des adaptateurs appropriés sont également très utiles pour les semi-conducteurs. La **figure 17** montre le circuit de l'adaptateur pour semi-conducteur et la **figure 18** le dispositif fini.

Le connecteur SV1 représente la connexion au traceur de courbes. La tension de chauffage alimente les relais K1 et K2, qui effectuent la commutation PNP/NPN ou NMOS/PMOS. La tension du réseau de commande est utilisée pour K3. Ce relais commute le courant de base ou la tension de grille. La tension d'anode est utilisée pour alimenter le collecteur/émetteur ou le drain/source. Pour tester les diodes Z, l'anode sera connectée au pôle négatif et la cathode au pôle positif de la tension d'anode.

Pour tester les transistors bipolaires, la tension de grille de l'écran est utilisée pour produire le courant de base. Cette tension est limitée par R2 et R3 à 4 mA sous 400 V. Pour tester les MOSFET, R1 est connecté comme diviseur de tension pour créer la tension de grille. D3 et D4 limitent U_{GS} à un maximum de 19 V.

Diodes Z et $U_{ce\ max}$ des transistors bipolaires

Après avoir sélectionné *diode Zener,* le texte des champs $U_{ce\ max}$ et $I_{c\ max}$ devient U_z et P_{max} (fig. 16), car la puissance maximale est généralement spécifiée pour les diodes Zener. Après avoir appuyé sur le bouton *Start curve,* la tension d'anode est augmentée de 0...U_z et au maximum jusqu'à P_{max} (selon celui qui sera atteint d'abord). Si le courant traversant la diode Z à U_z *est* inférieur à 1 % du courant Z maximum, le message d'erreur de la **fig. 20** est affiché. Dans ce cas, c'est soit

Figure 14. Adaptateur de tube pour EL34.

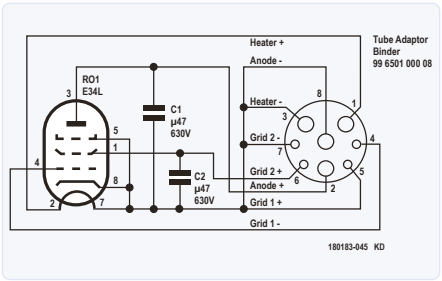


Figure 15. Schéma de l'adaptateur pour tube.

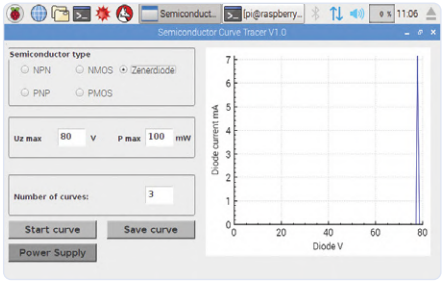


Figure 16. Tension maximale collecteur-émetteur d'un BC547C testé.

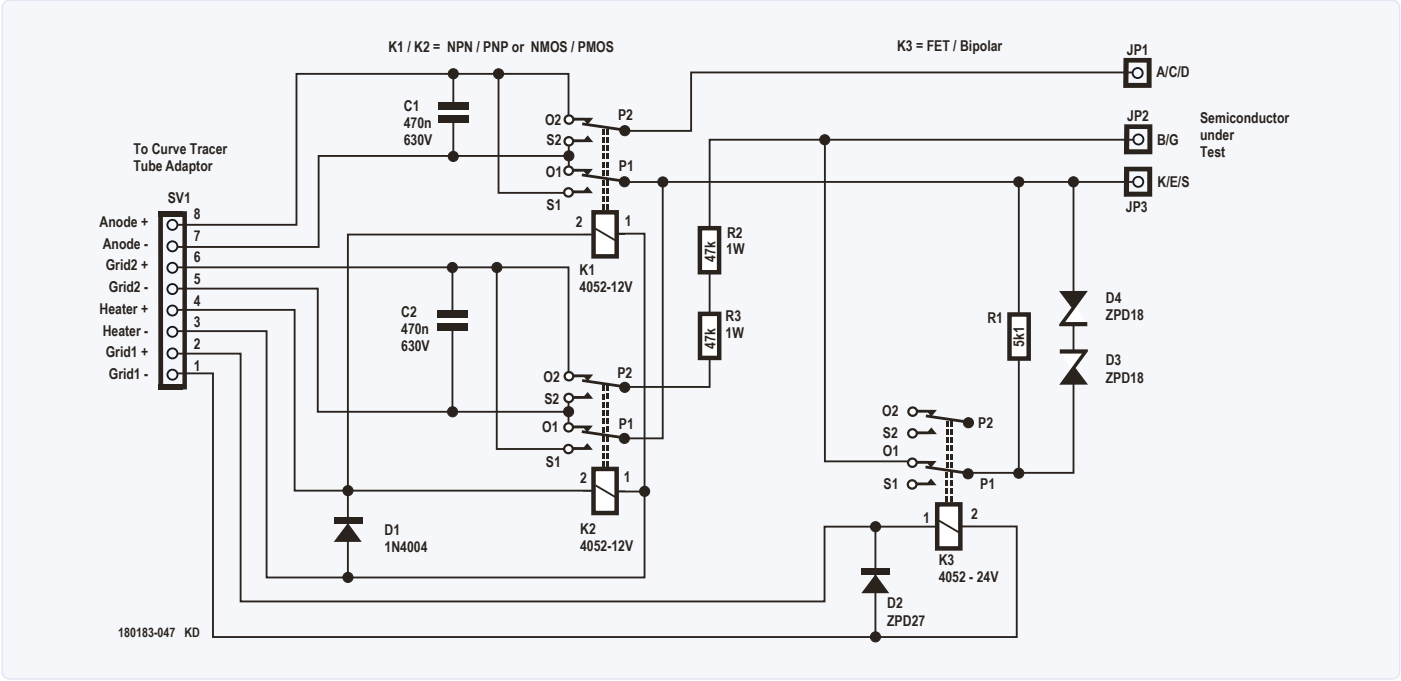


Figure 17. Circuit de l'adaptateur pour semi-conducteur.

la diode Z qui est défectueuse, soit il faut augmenter U_z.

En raison de la tension d'anode élevée possible, la tension maximale collecteur-émetteur de nombreux transistors bipolaires peut également être testée avec une base ouverte (= $U_{ceo\ max}$). La courbe de la figure 16 a été créée lors de l'essai d'un BC547C. Bien que la fiche technique n'indique que 45 V, ce transistor a atteint un U_{ceo} maximum de >75 V.

Test des transistors bipolaires

Pour les transistors bipolaires, il faut choisir : *NPN* ou *PNP*. $U_{ce\ max}$ et $I_{c\ max}$ et le nombre de courbes caractéristiques peuvent être définis. Après le démarrage, le logiciel vérifie si un courant de collecteur de forte intensité circule déjà, même sans courant de base. Dans ce cas, le test se termine par un message d'erreur. Sinon, $U_{ce\ max}$ est réglé sur la valeur sélectionnée et le courant de base est augmenté jusqu'à ce que $I_{c\ max}$ soit atteint. Si le courant de collecteur réglé ne peut pas être atteint avec le courant de base maximum de 4 mA, un message d'erreur le signale.

Les courbes caractéristiques $I_c = f(U_{ce})$ sont ensuite affichées avec les courants de base associés, qui sont indiqués dans la légende. Avec le bouton ***Enregistrer la courbe,*** les données des courbes caractéristiques de I_c, I_b et U_{ce} peuvent être enregistrées dans un fichier avec l'extension ***.csv.***

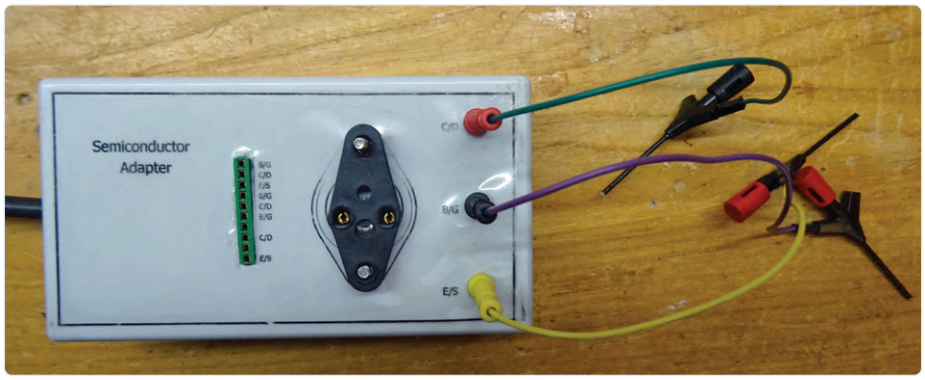


Figure 18. Adaptateur pour semi-conducteur de l'extérieur.

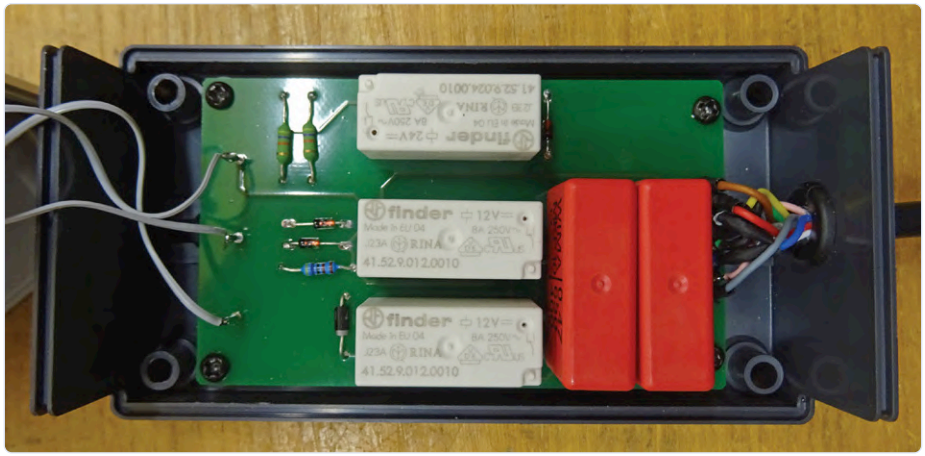


Figure 19. Adaptateur pour semi-conducteur de l'intérieur.

Figure 21. Prototype vu de face.

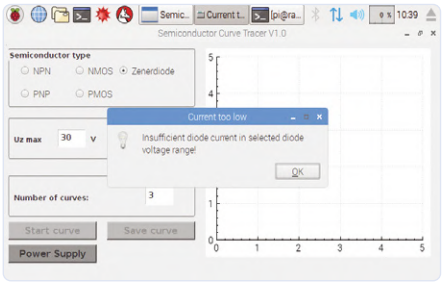


Figure 20. Message d'erreur lorsque l'intensité du courant à travers la diode Z est trop faible.

Test des MOSFET

Avec les MOSFET, vous devez choisir entre *NMOS* et *PMOS.* Les valeurs maximales pour $U_{ds\ max}$ et $I_{d\ max}$ et le nombre de caractéristiques peuvent être fixées. Après le démarrage, le logiciel vérifie si un courant de drain circule déjà à $U_{gs} = 0$. Dans ce cas, le test se termine par un message d'erreur. Sinon, la valeur de $U_{ds\ max}$ est fixée et U_{gs} est augmenté jusqu'à ce que $I_{d\ max}$ soit atteint. Si le courant de drain maximal ne peut être atteint à $U_{gs\ max} = 18$ V, un message d'erreur s'affiche également.

Les courbes caractéristiques $I_d = f(U_{ds})$ sont affichées avec les tensions de grille correspondantes, qui sont indiquées dans la légende. Le bouton *Enregistrer la courbe permet de* sauvegarder les données des courbes caractéristiques de I_d, U_{gs} et U_{ds} dans un fichier portant l'extension *.csv.*

Attention !

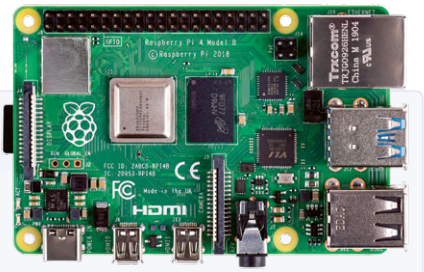
Veillez au réglage du courant maximal drain/collecteur et de la tension maximale drain-source ou collecteur-émetteur !

Un réglage de 400 V pour un courant de 300 mA entraîne une dissipation de puissance du transistor testé de 120 W ! Il est essentiel que les spécifications SOA de la fiche technique soient respectées, et le transistor en cours de test peut avoir besoin de refroidissement.

Voir aussi

Le téléchargement gratuit [1] contient les fichiers des cartes au format *Eagle*, y compris les listes de composants, les cotes pour le coffret, les fichiers LTSpice ainsi que le code source complet (disponibles pour un usage privé). Pour finir, la **fig. 21** donne une bonne impression de l'apparence du prototype.

180183-02



LIENS

[1] **Téléchargement associé à cet article** : www.elektormagazine.fr/180183-02
[2] **FrontDesign** : www.schaeffer-ag.de/fr/designer-de-faces-avant
[3] **Visual Studio** : https://visualstudio.microsoft.com/fr/downloads/
[4] **Compilateur croisé pour RPi** : https://visualgdb.com/download/
[5] **Qt Designer** : https://build-system.fman.io/qt-designer-download
[6] **WiringPi** : http://wiringpi.com/
[7] **WinSCP** : https://winscp.net/eng/download.php
[8] **QCustomPlot** : www.qcustomplot.com/index.php/download
[9] **Matchbox-Keyboard** : https://github.com/xlab/matchbox-keyboard

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 **10 octobre 2020** à l'adresse **hexadoku@elektor.fr**

Les gagnants

La solution de la grille du numéro de juillet/août 2020 est **8C5B2.**

Les cinq bons Elektor d'une valeur de **50 €** vont à Heinz Seitz, Allemagne – Kathryn Rangeley, Royaume-Uni – Mihails Sehurins, Lettonie – Martin Poelstra, Pays-Bas – Eric Poole, Australie

Bravo à tous les participants et félicitations aux gagnants !

		7	6	8	4							9			3
E				3	6	F	7		B		2	5	0	4	
					A			F		D			B		6
B			5	1	2	C		3					7	A	8
	3				5	2			1	0			F	7	
						1	3	9	F		8			C	
1			E				F		D	2			5		
	C						4			5		B			
	1						8			6		A			
D			9				0		2	4			1		
						7	9	D	5		A			8	
	6				1	A			8	E			9	D	
2			C	7	F	B		0					A	1	5
					E			6		7			D		C
4				A	0	3	2		C		D	8	6	B	
		A	F	5	C							7			0

6	D	7	A	B	9	3	5	C	E	4	0	2	F	8	1
F	9	0	5	4	7	C	1	8	A	D	2	6	B	E	3
4	1	C	2	6	E	8	D	7	F	3	B	5	9	A	0
3	B	E	8	F	0	A	2	9	1	5	6	7	C	D	4
0	A	D	1	7	B	4	3	E	9	C	5	8	6	F	2
B	6	2	9	5	F	0	C	A	7	1	8	E	3	4	D
C	5	3	4	1	6	E	8	B	D	2	F	0	7	9	A
E	F	8	7	9	D	2	A	0	3	6	4	B	1	C	5
7	4	6	0	C	1	5	E	D	B	8	3	F	A	2	9
A	3	9	F	D	4	B	7	2	0	E	1	C	5	6	8
8	C	5	B	2	3	6	0	F	4	9	A	D	E	1	7
2	E	1	D	8	A	F	9	5	6	7	C	3	4	0	B
D	2	A	6	0	5	7	F	1	C	B	9	4	8	3	E
1	0	B	C	A	8	D	4	3	5	F	E	9	2	7	6
9	8	4	E	3	C	1	B	6	2	0	7	A	D	5	F
5	7	F	3	E	2	9	6	4	8	A	D	1	0	B	C

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.

conception de filtres analogiques (1ère partie)

Étude de cas nº2 – 1 : la théorie du filtrage analogique

Alfred Rosenkränzer (Allemagne)

Voici une série d'articles sur la conception de filtres analogiques dont cette première partie traite des bases théoriques, mais avec l'œil sur l'application pratique visée. La deuxième partie portera sur la conception de filtres actifs. La troisième et dernière partie sera consacrée aux filtres passifs.

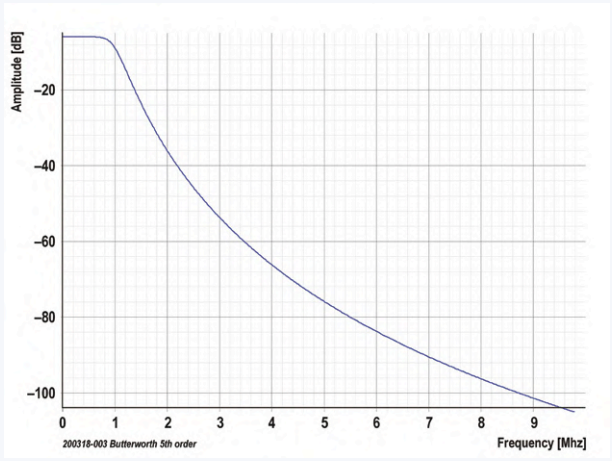


Figure 1. Réponse en fréquence d'un filtre passe-bas de 1 MHz avec courbe de fréquence et d'amplitude linéaire en dB.

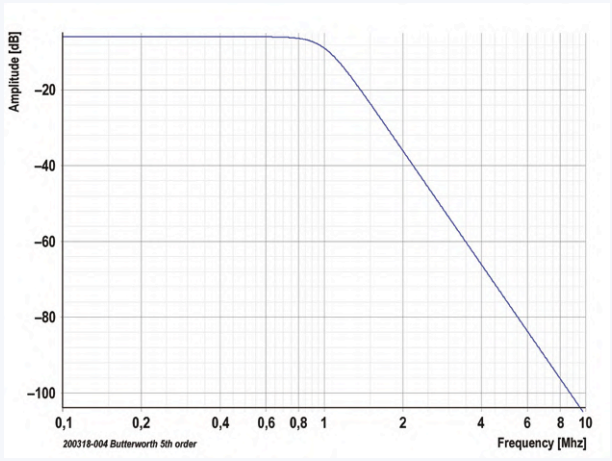


Figure 2. Réponse en fréquence dans le domaine audio avec échelle de fréquence logarithmique et amplitude en dB.

On obtient le filtrage analogique par une combinaison de résistances, de bobines, de condensateurs et d'amplificateurs opérationnels. Ils agissent sur les courbes de tension et de courant analogiques. Les filtres numériques, en revanche, sont basés sur des circuits logiques tels que des bascules, des additionneurs, des multiplicateurs (généralement sous la forme d'un DSP ou d'un FPGA) ainsi que des logiciels dans les microcontrôleurs et les SoC ; ils agissent sur les flux de données.

Propriétés des filtres

Les caractéristiques d'un filtre sont généralement représentées par des diagrammes XY, dont le principal montre l'amplitude du signal en fonction de la fréquence. C'est ce qu'on appelle la courbe de réponse en fréquence. Le rapport entre tensions d'entrée et de sortie est rarement représenté de manière linéaire sur l'axe Y en pourcentage, mais plutôt de manière logarithmique en dB. Sur l'axe des X, le tracé de la fréquence est souvent linéaire pour des bandes passantes plus étroites (**fig. 1**) ou, surtout pour la gamme audio, souvent

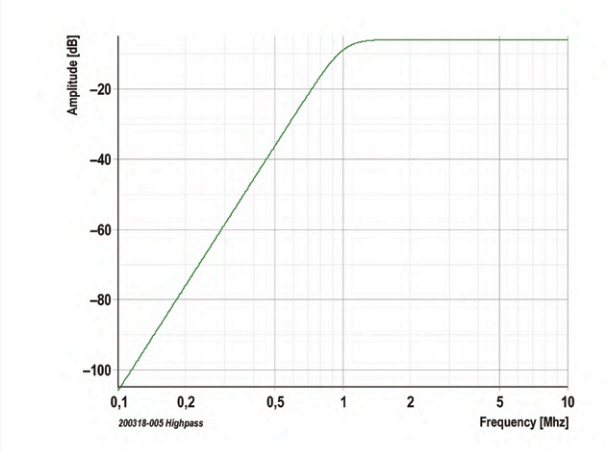


Figure 3. Réponse en fréquence d'un filtre passe-haut de 1 MHz avec double échelle logarithmique.

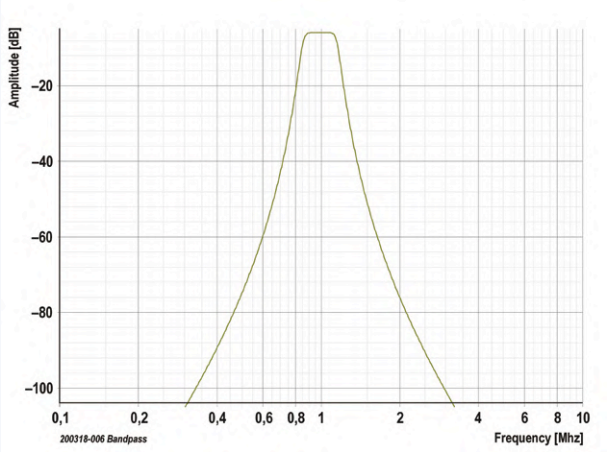


Figure 4. Réponse en fréquence d'une bande passante de 1 MHz avec double échelle logarithmique.

logarithmique (**fig. 2).** Mes diagrammes ont été créés avec le logiciel de simulation Simetrix [1].

Un filtre ne cause pas seulement une modification de l'amplitude, mais aussi de la phase du signal de sortie. Dans la réponse en phase, l'axe Y correspond au déphasage en degrés. Sur l'axe X, c'est la fréquence qui est indiquée. Pour moi, la réponse en phase n'est pas très significative, surtout dans le registre HF. Je trouve plus intéressant le retard de groupe sur la fréquence. Ce diagramme correspond au déphasage en fonction de la fréquence. Il permet de voir facilement combien de temps un signal d'une certaine fréquence met à traverser le filtre. Si la courbe du retard de groupe est plate, toutes les fréquences mettent le même temps. Je reviendrai en détail sur les raisons de cette importance à l'aide des schémas ci-dessous.

Dans le domaine temporel, la réponse impulsionnelle du filtre est une caractéristique importante. Elle montre la réponse d'amplitude linéaire du signal de sortie dans le temps lorsqu'il est excité par un front de signal – un peu comme l'affichage d'un oscilloscope. Ces propriétés seront également abordées.

Types de filtres

Un **filtre passe-bas** laisse les basses fréquences passer aussi librement que possible et atténue les fréquences élevées. Entre les deux se situe la gamme de transition. Le point auquel le signal subit une atténuation de 3 dB est considéré comme la limite entre bande passante et *bande d'atténuation*. Sa fréquence est donc la propriété importante du filtre. La bande d'arrêt commence à partir d'une certaine atténuation. Il n'y a pas de conventions pour cette atténuation, car elle dépend de l'application. Les figures 1 et 2 contiennent des exemples de la réponse en fréquence des filtres passe-bas. Un passe-bas laisse passer la tension continue.

Un **filtre passe-haut** laisse passer les fréquences élevées et atténue les fréquences basses. Ici aussi, le point –3 dB marque la limite de la bande de filtrage à la zone de transition (**fig. 3**). Un passe-haut supprime la composante continue du signal d'entrée.

Un **passe-bande** laisse passer une certaine bande de fréquences et atténue donc à la fois les fréquences basses et hautes. Ici, il y a deux points de –3 dB qui délimitent la bande passante (**fig. 4**). La branche passe-haut du filtre passe-bande bloque les tensions continues.

Le contraire du passe-bande est la **réjection de bande**. Une certaine bande de fréquences est atténuée, tandis que les fréquences plus élevées et plus basses passent (**fig. 5**). La composante passe-bas permet aux composantes de tension continue de passer.

Un **passe-tout** ne change pas l'amplitude, sa courbe de réponse en fréquence est plate, mais le retard de groupe ou la phase (**fig. 6**) du signal de sortie change selon la fréquence. En additionnant au signal original les signaux issus de (plusieurs) filtres passe-tout, on obtient un **filtre en peigne** dont la courbe d'amplitude du signal de sortie présente de profondes encoches causées par l'atténuation forte du signal à certaines fréquences (cette atténuation résulte de la superposition du signal original et du signal déphasé.

Approximations des filtres

Comment concevoir des filtres et définir leurs propriétés ? La question n'a rien d'anodin, surtout pour les filtres d'ordre supérieur. Des mathématiciens célèbres ont imaginé la modélisation des filtres selon des règles spéciales, toujours valables aujourd'hui. Leur calcul est facilité par les ordinateurs. Voici quelques-uns de ces filtres qui portent des noms célèbres :

- Bessel
- Butterworth
- Thebychev
- Cauer

De nombreuses autres variantes sont concevables et réalisables, de même que des combinaisons de caractéristiques. Cet article mettra en lumière les caractéristiques les plus importantes de ces filtres afin de vous faciliter le choix d'un filtre pour une application spécifique. Les **figures 7** et **8** montrent quatre passe-bas passifs différents et leur réponse en fréquence.

Les trois circuits du haut sont identiques à l'exception des valeurs des composants, mais leur réponse en fréquence n'est pas la même. La transition du filtre Bessel est très douce entre bande passante et bande d'atténuation. La réponse en fréquence du filtre Butterworth est déjà plus raide, celle du filtre Tchebychev encore plus. Le filtre Cauer utilise deux condensateurs supplémentaires au-dessus des bobines pour obtenir une réjection de bande à deux fréquences spécifiques, ce qui permet d'obtenir une transition encore plus raide dans la bande de réjection. La réponse en fréquence n'est donc pas continue et,

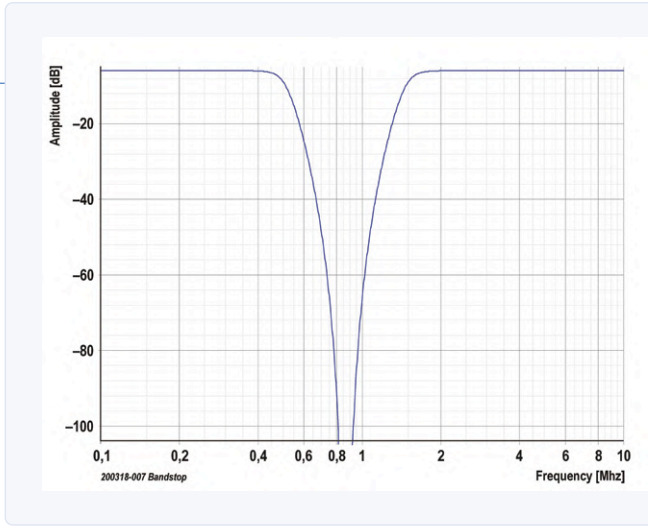


Figure 5. Réponse en fréquence d'un filtre à réjection de bande avec double échelle logarithmique.

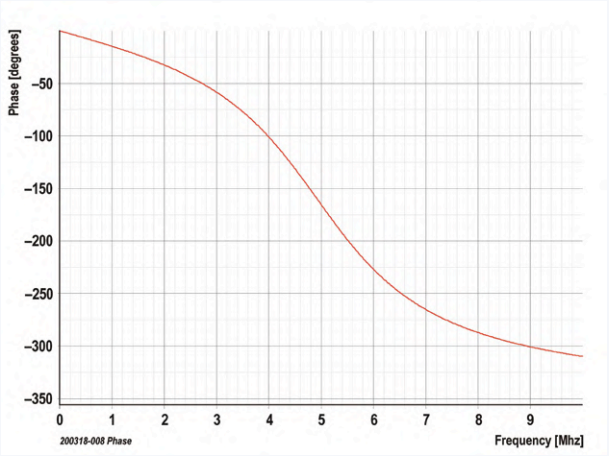


Figure 6. Réponse en phase d'un filtre passe-tout avec échelle de fréquence linéaire.

surtout pour les hautes fréquences, l'atténuation reste presque constante et plus faible qu'avec les trois autres types de filtres. La position des encoches sur la courbe de réponse est déterminée à la conception. Ces encoches permettent de supprimer des fréquences spécifiques. C'est aussi la conception qui détermine la bosse que fait la courbe entre deux encoches, de même que l'atténuation aux fréquences.

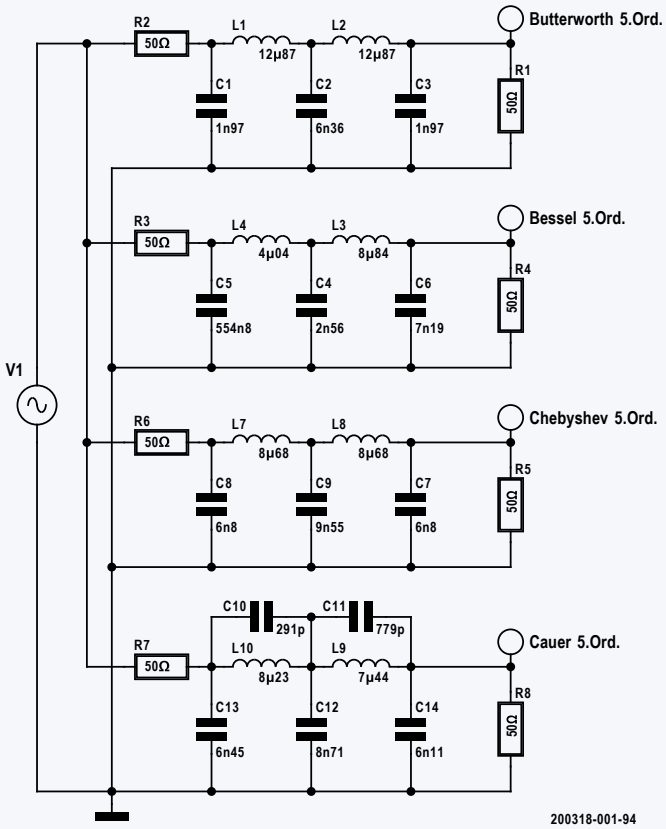


Figure 7. Passe-bas passifs de 1 MHz de 5e ordre (Butterworth, Bessel, Tchebychev, Cauer)

La **figure 9** nous invite à regarder de plus près les ondulations de la bande passante et la réponse d'amplitude autour du point à –3 dB. Bessel atténue assez tôt dans la bande passante, mais doucement. Butterworth fonctionne presque idéalement, comme un simple passe-bas RC, mais avec une pente moyenne sur les bords. Tchebychev et Cauer montrent une ondulation spécifique dans la bande passante avec une chute d'amplitude pouvant atteindre –1 dB. Leur réponse en amplitude ne recoupe pas le niveau de –3 dB à la fréquence nominale, mais atteint ici –1 dB. C'est précisément ce dernier point qu'il faut garder à l'esprit lors de la conception. Pour Tchebychev et Cauer, il en va de même : plus le filtre est raide, plus l'ondulation de la bande passante est forte.

La figure 10 montre que le retard de groupe du filtre Bessel est indépendant de la fréquence. Avec le filtre Butterworth, le retard de groupe augmente continuellement avec la fréquence. Le retard du groupe dans les filtres Tchebychev et Cauer présente aussi une ondulation.

Il est intéressant d'examiner le comportement des signaux de sortie lorsque les filtres sont excités par un front ou par une onde carrée (**fig. 11**). La réponse impulsionnelle du filtre Bessel (**fig. 12**) ne montre pratiquement aucun dépassement. Le filtre Butterworth (graphique bleu) montre un dépassement à dégradation rapide. Avec Tchebychev et Cauer, on voit des oscillations quasi amorties. Dans les quatre filtres, les fronts montants sont également décalés, ce qui est normal en raison des différents retards de groupe.

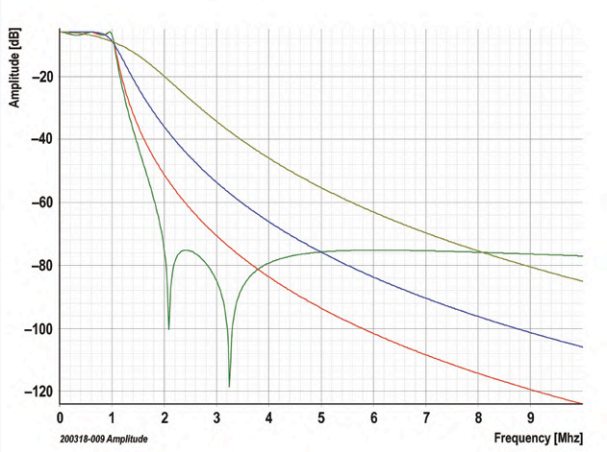


Figure 8. Réponses en fréquence des quatre filtres passe-bas de la figure 7 avec amplitude logarithmique et échelle de fréquence linéaire. Courbes : vert clair = Bessel, bleu = Butterworth, Tchebychev = rouge et Cauer = vert foncé.

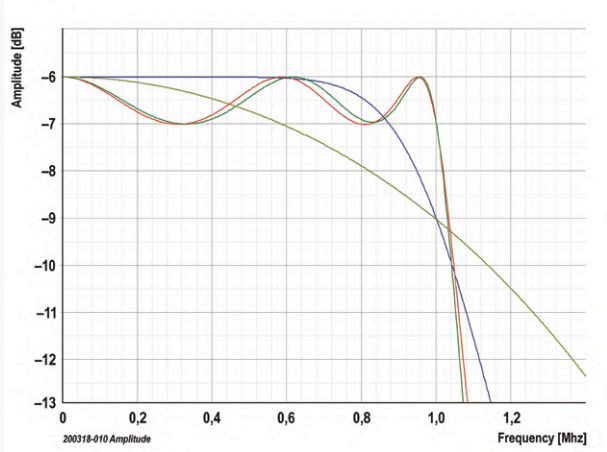


Figure 9. Les quatre réponses en fréquence de la figure 8 avec une résolution plus élevée. Vous pouvez voir le parcours dans la bande passante et autour du point –3 dB plus précisément.

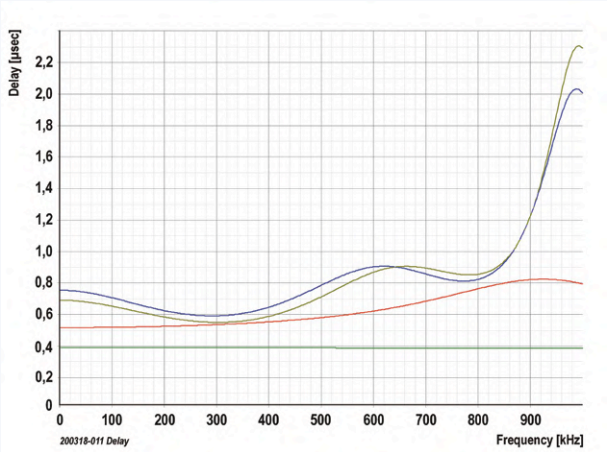


Figure 10. Retard de groupe dans la bande passante en fonction de la fréquence des quatre filtres.

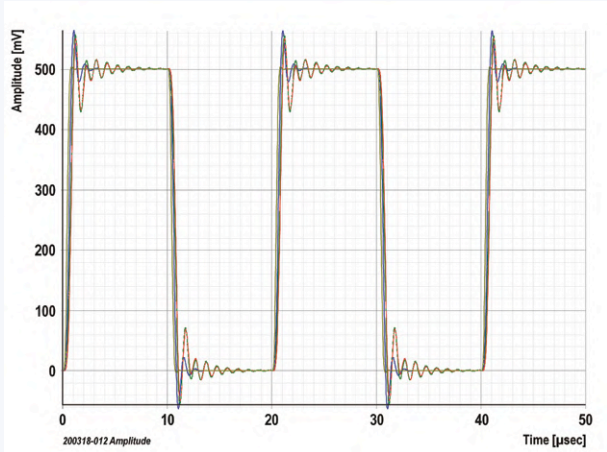


Fig. 11. Courbe des signaux de sortie des quatre filtres lorsqu'un signal d'onde carrée est appliqué. Courbes : vert clair = Bessel, bleu = Butterworth, rouge = Tchebychev et vert foncé = Cauer

En plus des considérations électrotechniques, les ondulations des caractéristiques du signal peuvent être dérivées spectralement : selon Fourier, un signal d'onde carrée peut être vu comme une combinaison de ses composantes sinusoïdales. Lorsqu'elles passent à travers le filtre, ces fréquences subissent, différemment les unes des autres, un changement d'amplitude et peut-être aussi un décalage temporel. Il n'est donc pas étonnant qu'à la sortie du filtre la courbe d'amplitude soit modifiée. Avec le filtre Bessel, les composantes de fréquence élevée sont atténuées, mais grâce au retard de groupe constant, elles ne sont pas décalées par rapport aux fréquences plus basses. Cela augmente le temps de montée et de descente et arrondit les angles du signal de l'onde carrée. En raison du retard de groupe fortement dépendant de la fréquence sur les filtres Tchebychev et Cauer, les dépassements et les ondulations observables se produisent sur un front raide , lors d'un changement rapide du signal d'entrée.

Ce comportement était redouté autrefois, par exemple dans les circuits de vidéo analogiques. En insérant des passe-tout, on obtenait un nivellement des retards de groupe. En réduisant les dépassements du signal, on rétablissait le front étroit. La conception de ces passe-tout de correction du retard de groupe est une science en soi. Le nombre de composants requis est plus élevé et tous les programmes de conception n'offrent pas de tels calculs.

Heuristiques pour la sélection des filtres

L'encadré **Caractéristiques** en donne les plus importantes pour les filtres les plus courants. Résumons leurs particularités :

Un **filtre Bessel** présente un comportement d'impulsion optimal et convient donc à la formation d'une impulsion à angles arrondis à partir d'un signal numérique à pente raide. Sa faible pente d'atténuation le rend impropre à la réjection de fréquences.

Caractéristiques de quelques filtres

type	bande passante	bande d'atténuation		pente
	courbe	courbe	atténuation	
Bessel	très progressive	très progressive	faible	faible
Butterworth	plate	pente rectiligne	moyenne	moyenne
Tchebychev	ondulante	pente rectiligne	large	forte
Cauer	ondulante	ondulante	large	forte

Le **filtre Butterworth** offre un bon compromis entre la pente du filtre et réponse impulsionnelle. Sa courbe d'amplitude ne présente aucune ondulation.

Les **filtres Tchebychev** et **Cauer** permettent d'obtenir une forte pente d'atténuation à la fréquence de coupure, au prix d'ondulations de la bande passante et de dépassements forts et plus longs (on dit du filtre qu'il résonne).

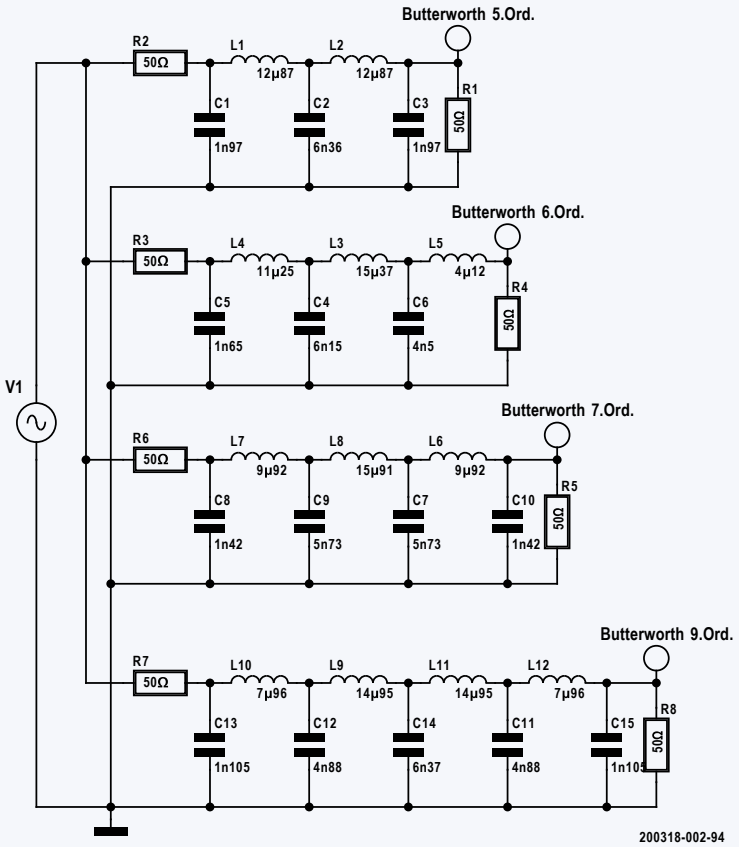


Figure 13. Filtres Butterworth des ordres 5, 6, 7 et 9.

Voyons comment les propriétés d'un filtre changent avec son ordre, c'est-à-dire avec le nombre de composants déterminant la fréquence. Les filtres Butterworth d'ordre 5, 6, 7 et 9 sont utilisés comme exemples dans la **figure 13**. Les valeurs des composants présentent une symétrie dans les ordres impairs. La **figure 14** montre comment la pente des filtres augmente avec l'ordre. La **figure 15** montre les rapports autour du point –3 dB agrandi : tous les filtres se

Terminologie

Technique	› analogique passif › analogique actif › numérique
Type	› passe-bas › passe-bande › passe-tout › réjection de bande
Configuration	› Bessel › Butterworth › Tchebychev › Tchebychev inversé › Cauer › Lehmann › Legendre › Gauss › en cosinus à racine surélevée
Impédance	› d'entrée › de sortie
Fréquence	› fr. de coupure › fr. d'atténuation › fr. centrale de bande passante › fr. de réjection
Atténuation	› atténuation de coupure › atténuation de réjection
Divers	› retard de groupe › réponse impulsionnelle › ordre › complexité › sensibilité aux tolérances des composants

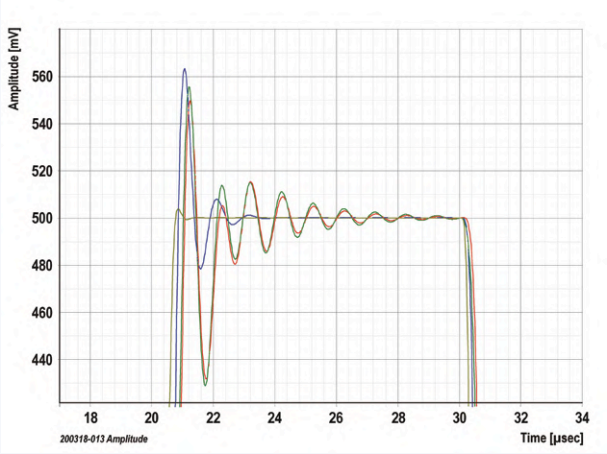


Figure 12. Réponses aux impulsions visibles dans un agrandissement détaillé de la figure 10. Courbes : vert clair = Bessel, bleu = Butterworth, rouge = Tchebychev et vert foncé = Cauer

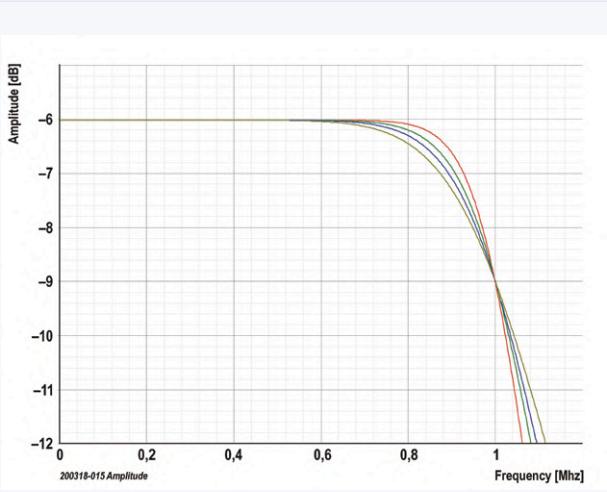


Fig. 15. Réponses en fréquence selon l'ordre des filtres, agrandies autour du point –3 dB par rapport à la fig. 14.

croisent exactement à ce point. Les filtres d'ordre supérieur n'atténuent le signal qu'à des fréquences plus élevées, mais le font alors avec une pente d'autant plus raide.
Plus l'ordre est élevé, plus le retard de groupe (**fig. 16**) et le dépassement (**fig. 17)** sont **prononcés**, en amplitude et en durée. La pente d'un filtre est donc déterminée par l'approximation choisie ou le type de filtre, **ainsi que** par son ordre. En adoptant un ordre élevé et donc un nombre de composants plus élevé, on peut concevoir un filtre Bessel plus raide.

200318-03

LIEN

[1] **Simetrix** : www.simetrix.co.uk

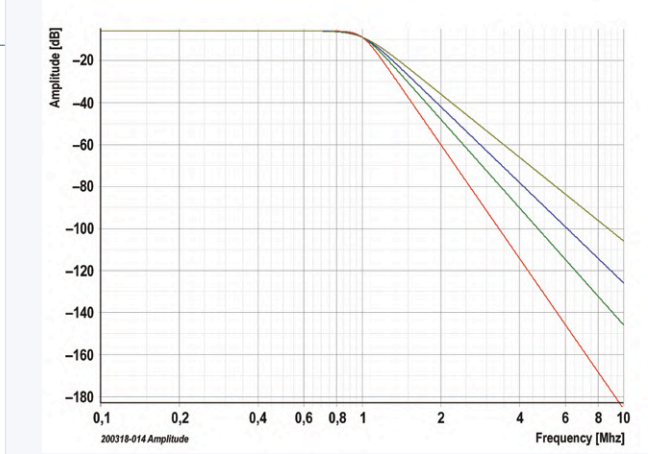


Fig. 14. Réponses en fréquence selon l'ordre du filtre avec double échelle logarithmique. Ordre de filtrage : 5 = vert clair, 6 = bleu, 7 = vert foncé et 9 = rouge.

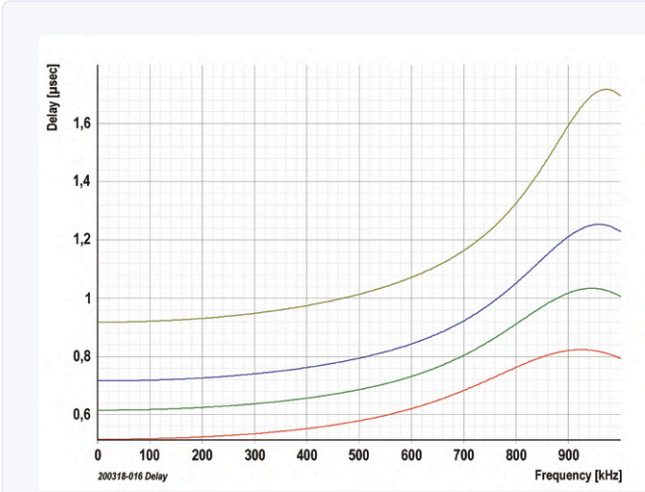


Figure 16. Retard de groupe en fonction de la fréquence pour les quatre ordres de filtres. Courbes : vert clair = 9, bleu = 7, vert foncé = 6 et rouge = 5.

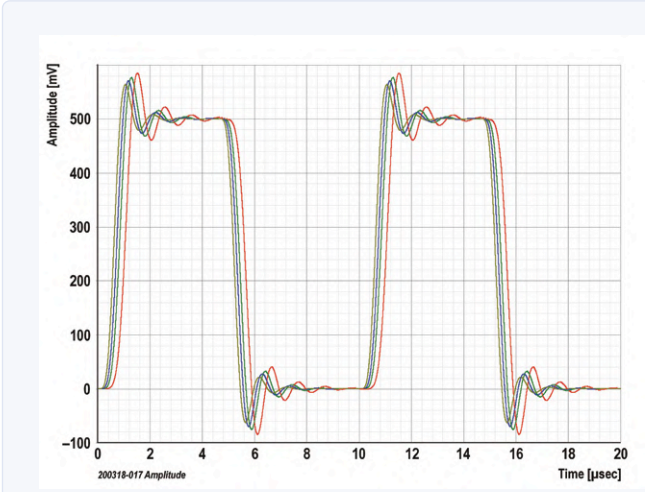


Figure 17. Réponse impulsionnelle des quatre types de filtres.

elektorstore

www.elektor.fr

e-choppe Elektor

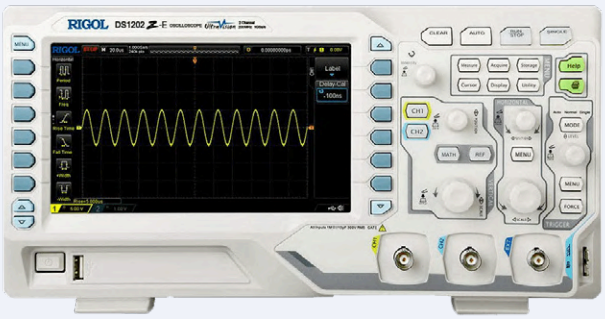
des produits et des prix surprenants

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

Oscilloscope Rigol à 4 voies S1054Z (50 MHz)

~~449,00 €~~
maintenant 349,00 €

www.elektor.fr/17821



Oscilloscope Rigol à 2 voies DS1202Z-E (200 MHz)

~~449,00 €~~
maintenant 349,00 €

www.elektor.fr/19344

RIGOL
Possibilities and More

Alimentation CC programmable Rigol DP832 3 sorties (0 à 30 V, 0 à 3 A, 195 W)

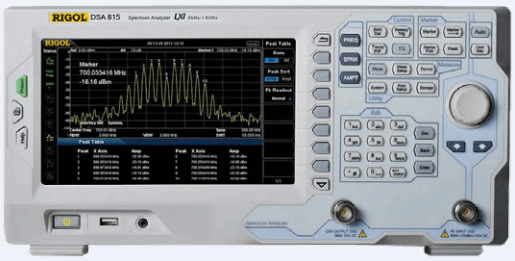
~~549,00 €~~
maintenant 449,00 €

www.elektor.fr/17823

Générateur de fonctions et d'ondes arbitraires Rigol DG2052 (50 MHz)

~~699,00 €~~
maintenant 579,00 €

www.elektor.fr/19345



Analyseur de spectre Rigol DSA815-TG (de 9 kHz à 1.5 GHz)

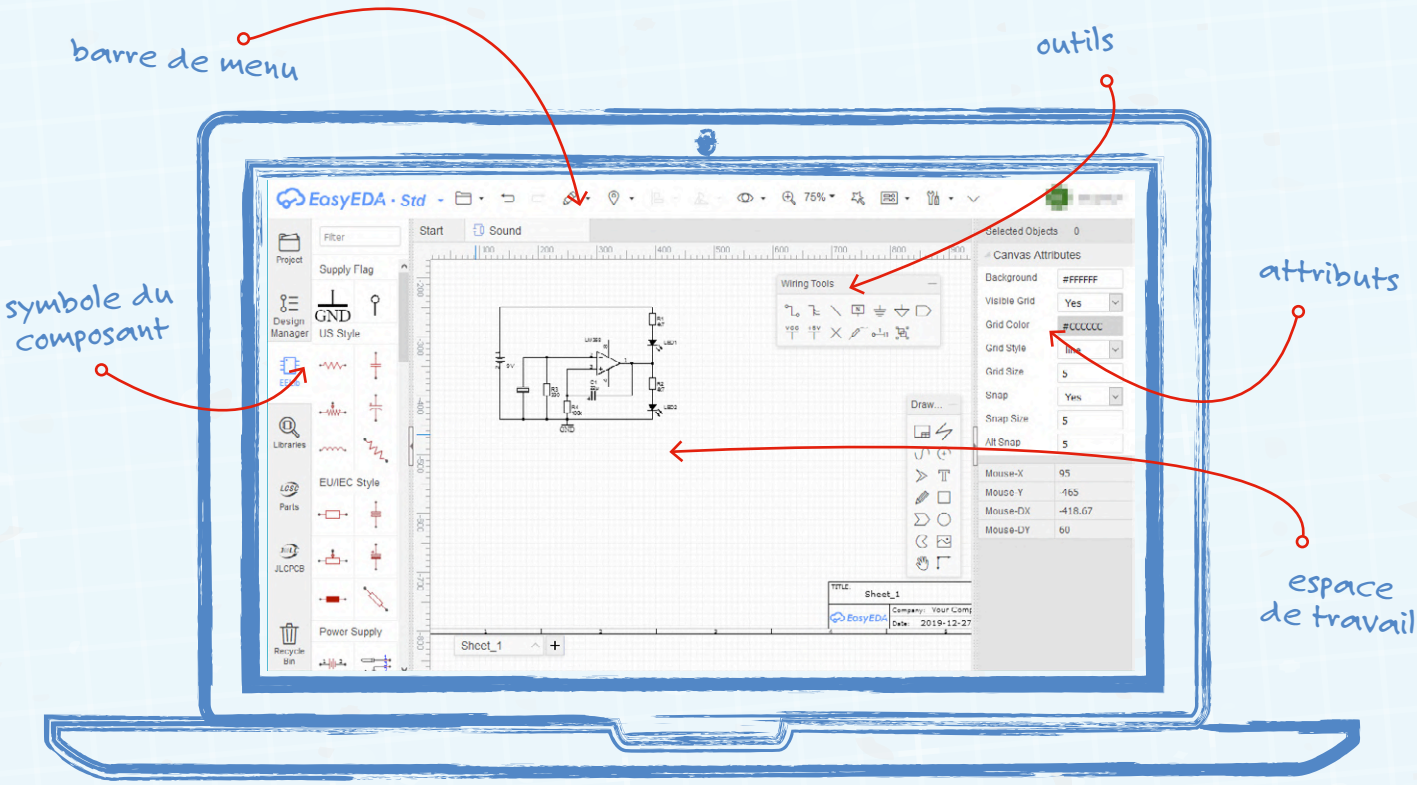
~~1 549,00 €~~
maintenant 1 299,00 €

www.elektor.fr/19349

Générateur de fonctions et d'ondes arbitraires Rigol DG2102 (100 MHz)

~~1 149,00 €~~
maintenant 949,00 €

www.elektor.fr/19347



dessine-moi un schéma

avec EasyEDA

Florian Schäffer (Allemagne)

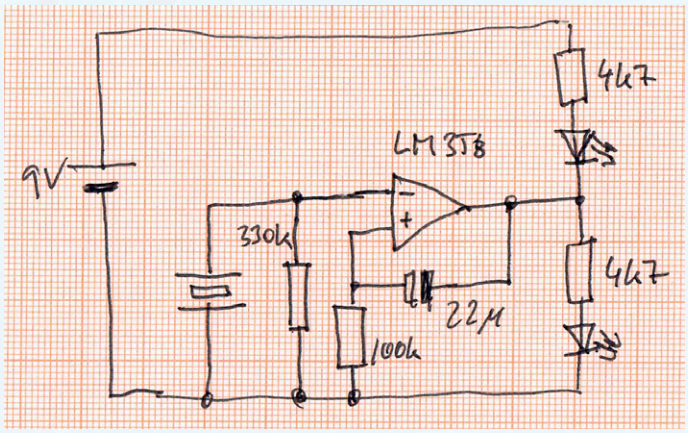
Si vous avez une bonne idée de circuit que vous souhaitez partager ou documenter pour vous-même, il est bon d'en faire un schéma sans tarder. À la main ? Pourquoi pas, mais avec EasyEDA, c'est presque plus facile que de dessiner à la main et c'est bien plus clair, surtout quand il faut modifier, ajouter ou supprimer…

N'importe quel logiciel graphique peut servir à dessiner des schémas électroniques, à condition de créer d'abord tous les symboles vous-même. Vous ne bénéficierez pas cependant des fonctions offertes par les programmes spéciaux d'assistance à la conception de circuits électroniques (CAO). Ceux-ci facilitent non seulement la conception, en offrant de vastes bibliothèques de symboles et de composants, mais ils peuvent également vérifier vos circuits, détecter des erreurs de logique et tester voire simuler le fonctionnement. À partir du schéma mis au propre, ils vous aident également à dessiner des cartes de circuits imprimés (PCB).

Le logiciel parfait ?

La question du logiciel idéal ne trouvera pas de réponse générale. L'EasyEDA présente l'avantage de pouvoir être utilisé gratuitement et dans un navigateur web sans installation locale. C'est pratique et facile pour une utilisation occasionnelle et en guise d'intro. Avec Fritzing, il est possible de créer non seulement les vues de la plaque d'expérimentation, mais aussi des schémas de circuit. Ces derniers sont hélas inesthétiques et les schémas ne répondent pas aux normes européennes. KiCad et gEDA sont gratuits, mais nécessitent une formation intensive. La prise en main n'est pas une partie de plaisir, les mécanismes d'installation ne semblent pas au point.
Target 3001 est un programme commercial pas cher, dont il existe une version (fortement bridée) pour les particuliers. Dans la cour des grands, le jeu est mené par Altium Designer et EAGLE, duquel il existe une version gratuite bridée.

Nous vous proposons dans ce petit atelier d'apprendre les bases d'EasyEDA, mais ne parlerons pas ici de l'étude de circuit imprimé possible en ligne à partir du schéma que vous aurez dessiné.

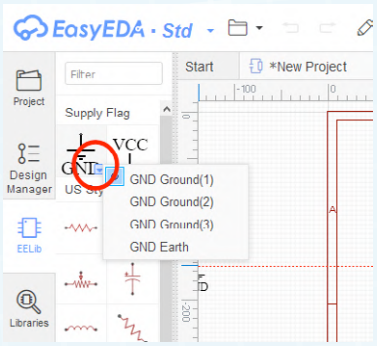


Un schéma que nous allons mettre au propre avec un logiciel de CAO (peu importe ici la fonction du circuit).

1. Inscrivez-vous sur le site de l'éditeur en ligne **https://easyeda.com/editor en** haut à droite avec un identifiant ou créez un nouveau compte si c'est votre première visite : *Register*. Vous pourriez dessiner un schéma sans même vous inscrire, mais ne pourriez, dans ce cas, ni le sauvegarder ni l'exporter.

2. Vous pouvez choisir la langue de l'interface, mais comme la traduction peut laisser à désirer, il vaut mieux s'en tenir à l'*anglais*. Créer un nouveau dessin avec *Document / New / Schematic*.

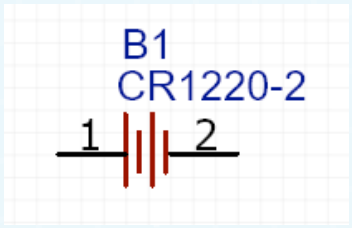
3. À gauche, on vous propose différentes catégories. Activez l'*EELib* pour les premières étapes pour afficher les symboles des composants les plus courants. Vous pouvez faire défiler la liste avec la molette de la souris. Il existe des symboles américains et européens selon la Commission électrotechnique internationale (CEI). Lorsque votre pointeur se trouve sur un symbole, un petit triangle apparaît pour la plupart des symboles, avec un menu contextuel. Vous pouvez l'utiliser pour sélectionner des styles spéciaux qui soit modifient l'apparence du symbole, soit sont importants pour la production de circuits imprimés — dans la plupart des cas, vous n'aurez pas (encore) besoin de ce paramètre tant que vous ne dessinez qu'un schéma.



4. Dans la section *Power Supply* (= alimentation), sélectionnez l'icône de la batterie en cliquant dessus. EasyEDA ne propose ici que des piles boutons, mais pour l'instant cela n'a pas d'importance.

5. L'icône est maintenant attachée au pointeur et vous pouvez la déplacer vers l'espace de travail. Avec la molette de la souris, vous pouvez effectuer un zoom avant et arrière sur la zone de travail.

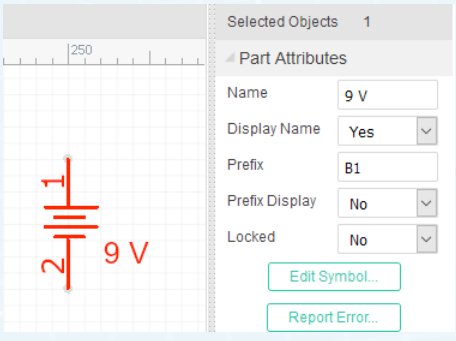
6. Cliquez dans l'espace de travail là où vous souhaitez placer le symbole. En cliquant derechef, vous pourriez poser une copie du même symbole ailleurs, mais s'agissant d'une batterie, cela n'aurait aucun intérêt. Terminez l'opération d'insertion en appuyant sur la touche Esc.



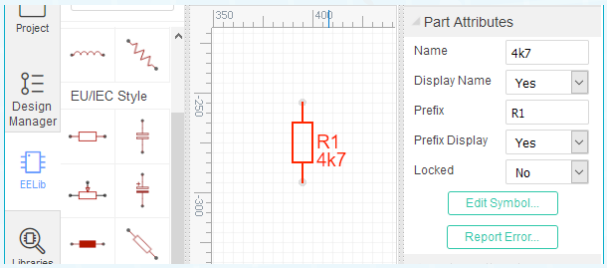
7. Vous pouvez à tout moment déplacer le symbole de circuit avec la souris ou le supprimer avec *Delete.* Veillez à cliquer sur le symbole et pas seulement sur l'étiquette.

8. Marquez le symbole de la pile en cliquant dessus et faites-le pivoter : Menu *Rotate* and *Flip / Rotate Right*.

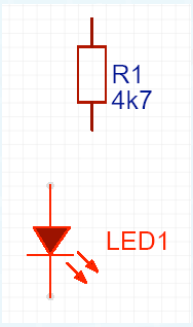
9. Sur le côté droit de la fenêtre, vous pouvez modifier les attributs du symbole tant qu'il est sélectionné. Replacer la valeur de *Name* par "9 V" et saisissez la valeur *No* pour *Display Prefix* (= *affichage du préfixe*).



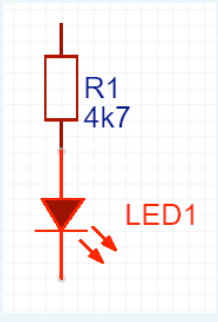
10. Insérez une résistance, faites-la pivoter de 90 ° vers la droite et changez sa valeur : "4k7" dans la section *Name* des attributs.



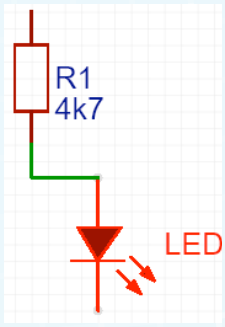
11. Insérez le symbole de diode électroluminescente (LED) dans le schéma près de la résistance que vous venez d'insérer. Faites pivoter la LED vers la droite et sélectionnez l'attribut de masquage de son nom : *Non* pour *Display name.*



12. Les deux composants doivent être reliés électriquement. Pour obtenir cette connexion, il suffit de pousser les deux symboles l'un vers l'autre pour que leurs broches se touchent.

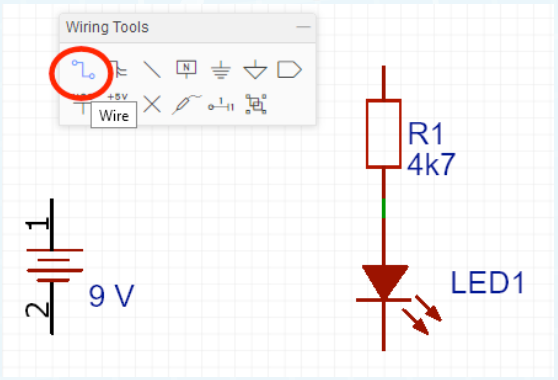


13. EasyEDA crée une connexion (invisible) et dès que vous éloignez à nouveau un peu l'un des deux symboles, vous verrez une ligne verte entre eux.

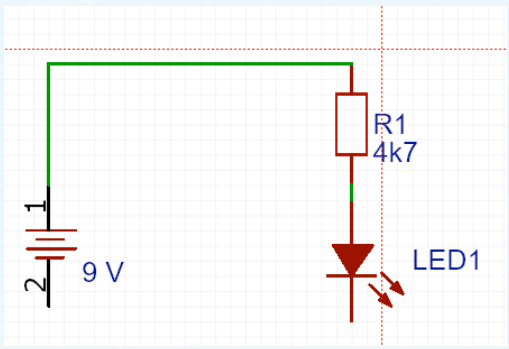


14. Vous pouvez également modifier cette ligne après avoir cliqué dessus pour la déplacer ou la supprimer. Une fois la connexion établie, elle sera maintenue même si vous déplacez les symboles — c'est l'un des grands avantages du logiciel EDA.

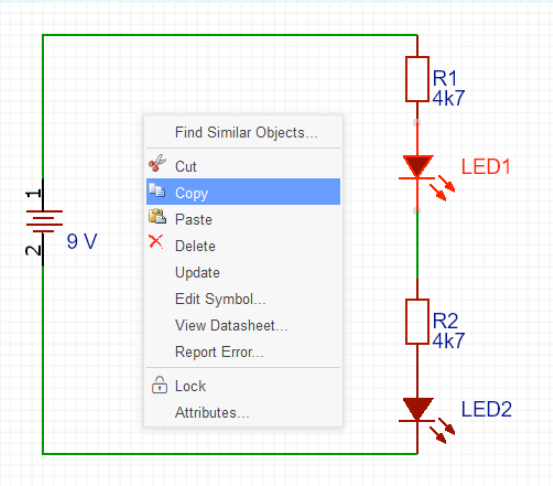
15. Vous pouvez établir des connexions manuellement, par exemple pour relier l'autre broche de la résistance à la batterie. Il y a deux palettes d'outils (si ce n'est pas le cas : *View / Wiring Tools*). Il est important de toujours établir les connexions électriques avec *Wire* (= *fil),* mais jamais avec Line (= *ligne*) *de* la palette *Drawing Tools*, car il s'agit de simples lignes graphiques dépourvues de fonction électrique.



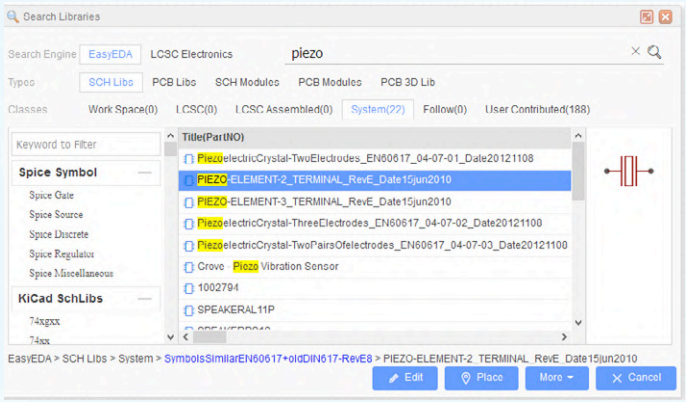
16. Activez *Wire* et cliquez sur la connexion de la batterie. De là, un lien vert sera établi avec la position de la souris. Vous pouvez conduire la ligne directement à la patte non connectée de la résistance. Pour cela, c'est le logiciel qui commande l'itinéraire de la ligne. Si vous n'êtes pas d\accord, vous pouvez fixer vous-même le trajet en cliquant sur des points intermédiaires. Dès que vous cliquerez sur une autre connexion ou un autre composant, la connexion sera établie.



17. Insérez la deuxième LED et la deuxième résistance puis connectez ces composants entre eux et à la batterie. Vous pouvez soit utiliser les nouveaux symboles de la sélection de gauche, soit travailler par *copier-coller* via le presse-papiers comme avec tout autre programme. Pour cela, il existe le menu *Edit.* Vous pouvez utiliser le menu contextuel qui s'ouvre par un clic droit.



18. Deux symboles de circuit ne figurent pas dans la liste standard : le buzzer piézo et l'amplificateur opérationnel. Pour insérer ce dernier, cliquez sur l'icône *Library* (= *bibliothèque*) à l'extrême gauche, de sorte que la fenêtre de recherche de composants s'ouvre. Dans le champ de recherche, saisissez la désignation (numéro de type ou description) : "LM358" et cliquez sur le symbole de la loupe à droite.



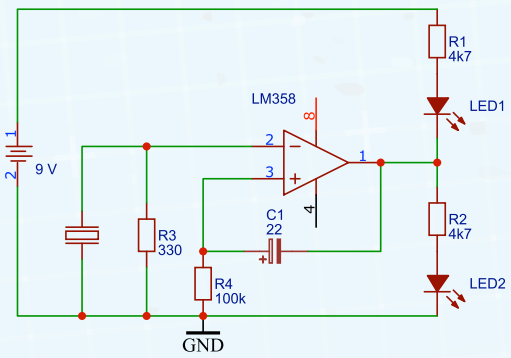
19. Patientez… une liste de résultats apparaît. Plusieurs entrées y figurent, qui diffèrent par la forme du symbole (à droite) et du composant réel. Pour l'exemple, c'est le composant LM358APWR qui convient. Ce modèle réunit dans un seul boîtier deux amplificateurs opérationnels (peu importe ici leur fonction). Sélectionnez la deuxième sous-entrée (*LM358APWR.2*) et cliquez sur *Place* : la fenêtre se ferme et vous pouvez simplement placer le symbole sur l'espace de travail.

20. Il faudra probablement faire de la place pour les autres composants et déplacer les symboles déjà présents. Pour l'instant on ne s'occupe pas des broches 4 et 8 (alimentation de l'amplificateur opérationnel).

21. C'est le moment de chercher l'élément : cliquez à nouveau sur *Library* et saisissez "piezo" comme mot-clé à rechercher. Dans la liste de résultats, le deuxième semble approprié : cliquez dessus et insérez-le dans le schéma avec *Place.*

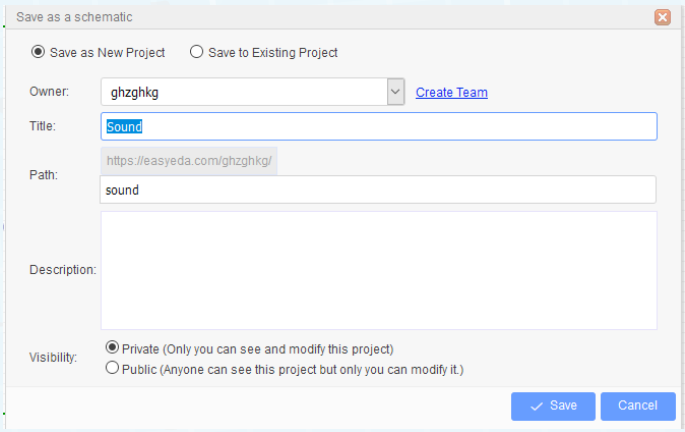
22. Le reste des symboles et des connexions à établir ne devraient plus poser de problème. EasyEDA connaît les règles de dessin des points de connexion. Quand deux lignes se croisent simplement, aucun point n'est dessiné, mais si vous cliquez sur un autre lien lors de la pose d'une connexion, un point est ajouté.

23. Par souci d'exhaustivité, il convient d'ajouter un symbole de masse. Sélectionnez *GND Ground(3) dans la* zone de *Supply Flag* comme symbole GND dans le menu contextuel afin d'obtenir le symbole UE pour la masse plutôt qu'un symbole de terre.



C'est ainsi que le schéma doit se présenter à la fin.

24. Sauvegardez le schéma. Sélectionnez *Document/Save As* et saisissez un nom de fichier de votre choix sous *Title* dans la fenêtre qui s'ouvre. Laissez les autres paramètres tels quels. Vous pouvez également publier votre projet afin que d'autres utilisateurs puissent y accéder sur l'internet en activant *Public*. Cliquez sur *Save* pour enregistrer votre projet.



25. En utilisant *Document/Export/…* vous pouvez enregistrer votre schéma localement sous différents formats de fichiers. Si vous souhaitez un affichage monochrome pour les formats graphiques, sélectionnez d'abord *Theme/Black on White*.

Note: *Cet article a été tiré d'un numéro hors-série d'Elektor en allemand, consacré à l'électronique avec Arduino, disponible dans la boutique en ligne d'Elektor : www.elektor.de/elektor-special-einstieg-in-die-elektronik-mit-arduino. Fin 2020, ce numéro hors-série sera disponible en anglais.*

200225-03

Rejoignez les électroniciens de la communauté Elektor

Devenez membre elektor GOLD membership maintenant !

- accès à l'archive numérique depuis 1978 !
- 6x magazine imprimé Elektor
- 9x magazine numérique (PDF) dont Elektor Industry (EN)
- 10 % de remise dans l'e-choppe et des offres exclusives pourles membres
- le DVD annuel d'Elektor
- accès à plus de 1000 fichiers Gerber, collaboration avec les milliers d'électroniciens d'Elektor LAB, et une ligne directe avec nos experts !
- possibilité de voir votre projet publié ou vendu par notre boutique en ligne

---

Également disponible

abonnement elektor GREEN membership sans papier !

- accès à l'archive numérique d'Elektor
- 10 % de remise dans l'e-choppe
- 6x magazine Elektor (PDF)
- offres exclusives
- accès à plus de 1000 fichiers Gerber

www.elektormagazine.fr/membres

elektor
design › share › sell

ABONNEZ-VOUS ET RECEVEZ

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

Vos avantages :

- Prix au numéro réduit
- Chaque numéro directement dans votre boîte aux lettres
- Tous les numéros disponibles sous forme numérique (PDF)
- Découverte de chaque nouveau numéro avant sa sortie en kiosque

SEULEMENT
54,95 €
PAR AN
(6 NUMÉROS)

ABONNEZ VOUS : WWW.MAGPI.FR