ISSN 0181-7450

F : 15,50 €

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

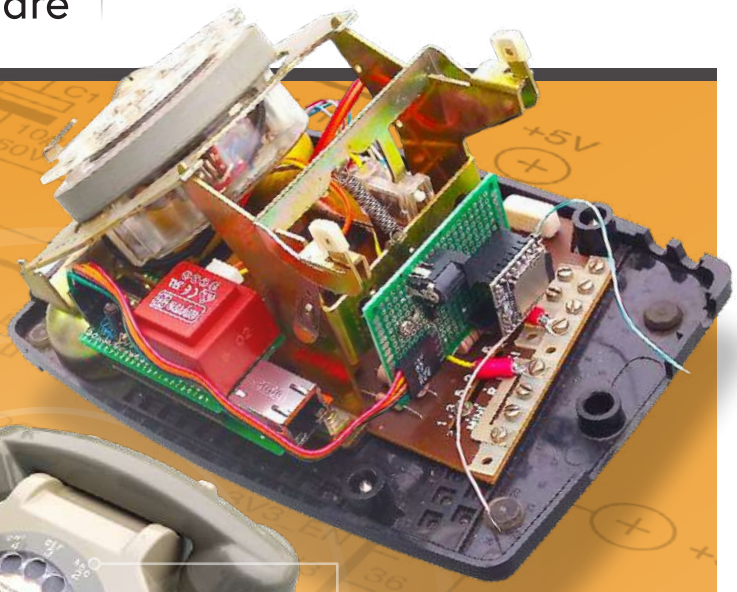
elektor 502

créer > partager > vendre

JUILLET & AOÛT 2023
ELEKTORMAGAZINE.FR

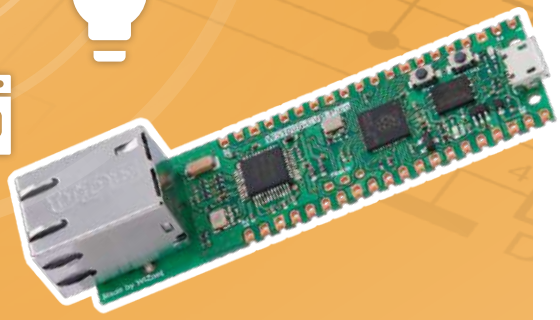
Basé sur le RP2040

Le téléphone à cadran rotatif comme télécommande



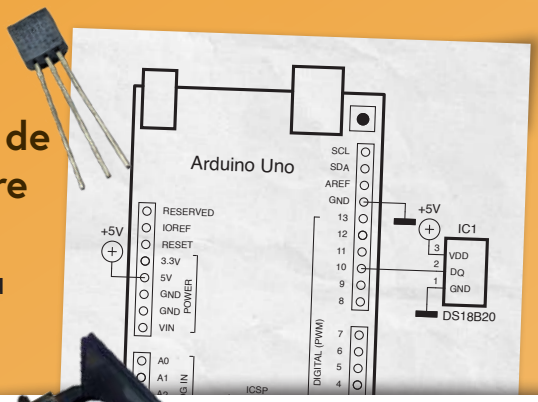
FOCUS SUR

IdO et capteurs



Le capteur de température DS18B20

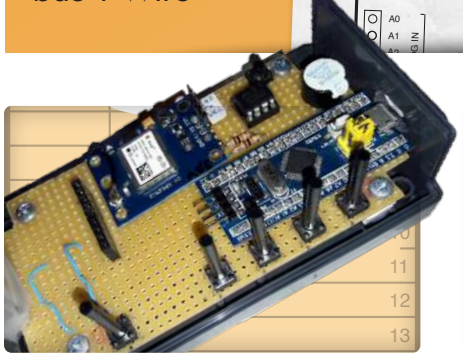
Connexion au bus 1-Wire

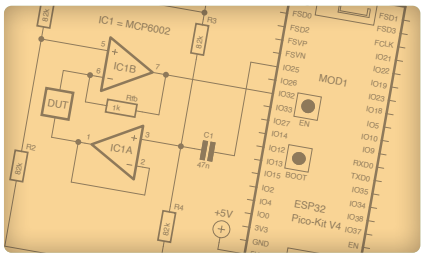


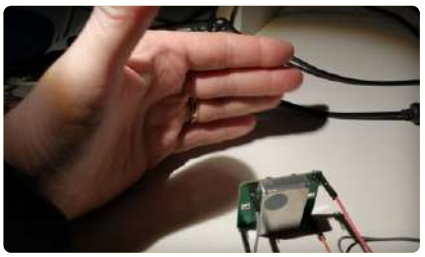
matter

Quels standards pour unifier la domotique ?

Matter et Thread se distinguent







L 19624 - 502 - F: 15,50 € - RD

PRÉPAREZ-VOUS À

un été plein de projets !

Appel à tous les passionnés d'électronique ! Le numéro spécial circuits du magazine Elektor paraîtra au mois d'août 2023 et regorgera de nombreux projets intéressants. Nous vous invitons à découvrir le monde fascinant des projets électroniques et à **STIMULER VOTRE CRÉATIVITÉ !**

RESTEZ À L'ÉCOUTE !

Tous nos membres recevront ce hors série. Il sera également disponible dans votre kiosque préféré.

www.elektormagazine.fr/special-circuits

elektor
design › share › earn

46ème année n° 502
juillet-août 2023
ISSN 0181-7450

N° de TVA Intracommunautaire : FR90319937454

Dépôt légal : mai 2023
CPPAP 1125 T 83713
Directeur de la publication : Donatus Akkermans

Elektor Magazine est publié 8 fois par an par
PUBLITRONIC SARL – c/o Regus Roissy CDG
1, rue de la Haye – BP 12910
FR - 95731 Roissy CDG Cedex

www.elektor.fr | www.elektormagazine.fr

Pour toutes vos questions :
service@elektor.fr

Devenez membre
www.elektormagazine.fr/abo

Publicité :
Ouafae Hassani
Tél. : +31 (0)6 41312932
ouafae.hassani@elektor.com
www.elektormagazine.fr/publicité

Tarifs Annuels :
France 1 an 129,95 € (8 numéros)

Imprimé aux Pays-Bas par Senefelder Misset, Mercuriusstraat 35, 7006 RK Doetinchem

Distribué en France par M.L.P. et en Belgique par A.M.P.

Jens Nickel

rédacteur en chef d'Elektor Magazine

IA terrifiante, IA amusante

C'est quoi *le contraire* de Marlon Brando ? Vous ne le savez pas ? Alors regardez la vidéo YouTube (flippante) à laquelle notre auteure Ilse Joostens renvoie dans son article à la page 46. Cette fois, la chronique porte sur l'intelligence artificielle, et plus particulièrement sur les innombrables outils qui génèrent des images, du texte et du code de programme. D'ailleurs, l'IA crée maintenant des vidéos entières, et là aussi, les résultats sont étonnants, parfois drôles, mais souvent un peu effrayants. On peut se demander pourquoi l'intelligence artificielle a tendance à générer si souvent des images d'horreur et d'apocalypse – même dans des publicités ordinaires pour des pizzas dans le style des années 1990 (*www.youtube.com/watch?v=MpvEXrhnoyw*).

Cependant, nous, dotés d'une intelligence naturelle, restons maîtres de la situation. Et tant que c'est le cas, nous devrions tirer le meilleur parti de l'IA. Les programmeurs et les électroniciens, en particulier, ont de nombreuses possibilités à cet égard, et c'est pourquoi chez Elektor, nous avons commencé à les explorer. Je suis certain que vous pourrez bientôt rechercher un certain article dans les immenses archives de notre magazine avec l'aide de l'IA. Il faudrait des années pour retrouver manuellement les innombrables liens croisés entre les projets et les articles de fond et pour transférer l'ensemble dans un corpus de connaissances complet (si nous parvenons même à imaginer un système viable pour cela).

Nous aborderons également la génération automatique du code de programme dans les prochains numéros. J'ai des amis qui utilisent maintenant ChatGPT pour écrire des fichiers batch afin d'éditer des fichiers audio. Je suis curieux de voir ce qu'il y a encore à découvrir, en particulier pour les électroniciens. Avez-vous déjà fait vos propres tests ? Avez-vous eu de bonnes ou de mauvaises expériences ? Écrivez-moi à l'adresse redaction@elektor.fr !

Proposez une contribution à Elektor!

Vos propositions sont les bienvenues ! Vous souhaitez proposer un article, un tutoriel vidéo ou une idée de livre ? Consultez le guide de l'auteur et la page de soumission d'Elektor :

www.elektormagazine.com/submissions.

Elektor Labs : idées et projets

La plateforme Elektor Labs est ouverte à tous. Publiez des idées et des projets électroniques, discutez des défis techniques et collaborez avec les autres.

www.elektormagazine.fr/labs

DANS CE NUMÉRO

Elektor n° 502
Juillet - août 2023

le téléphone à cadran rotatif comme télécommande

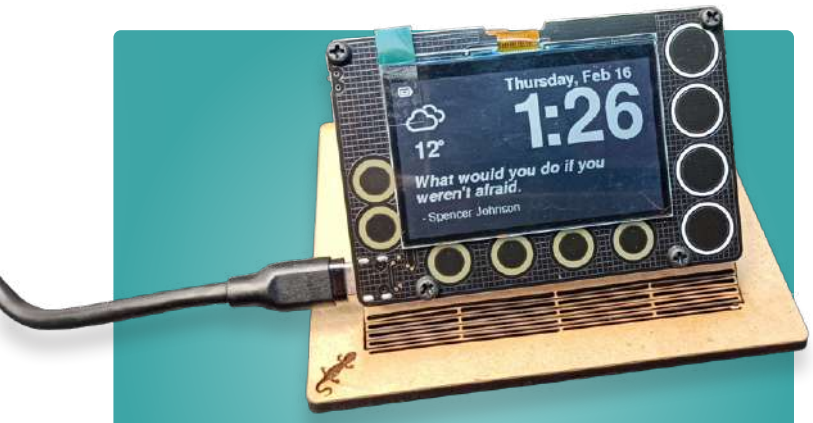
pour allumer la lumière, composez le 1 ;
pour la cafetière, composez le 2

6

Rubriques

Articles de fond

Industrie



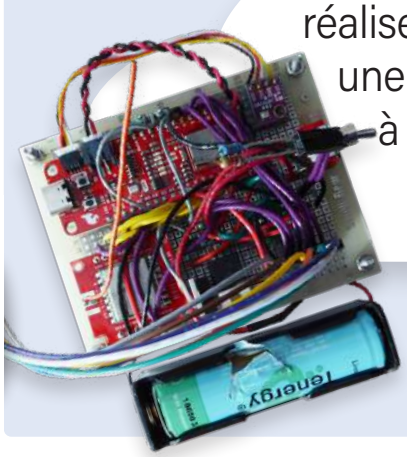
construisez un écran IdO sympa

avec le Phambili Newt

88

station météo LoRa à faible puissance

réalisez vous-même une station météo à longue portée



116

Projets

Bientôt dans ces pages

Le hors-série spécial circuits (août et septembre 2023)
Dans la tradition des circuits de vacances d'Elektor, le prochain numéro sera très épais et contiendra plus de 50 projets à réaliser soi-même, des circuits rétro, des trucs et astuces et bien d'autres choses encore !

Quelques-uns des points forts :

- redresseur actif
- un standard de fréquence à faible coût
- compresseur dynamique simple
- petite alimentation solaire
- générateur THD
- DAC programmable pour vidé
- grand chiffre RVB
- ChatGPT et Arduino
- boule de Noël à énergie solaire avec radio FM
- petit simulateur DCF77

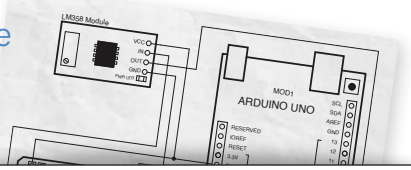
et bien d'autres choses encore !

Le hors-série spécial circuit du magazine Elektor sera publié aux alentours du 9 août 2023. La date d'arrivée du magazine papier chez les abonnés dépend des aléas d'acheminement. Le contenu et les titres des articles peuvent être modifiés.

détecteur de mouvement Doppler HB100

théorie et pratique

90



le téléphone à cadran rotatif **comme télécommande**

Clemens Valens (Elektor)

Même lorsque l'éclairage et les appareils sont contrôlés par un système domotique, il est souvent souhaitable de disposer d'une commande permettant de mettre en marche ou à l'arrêt un éclairage, un ventilateur ou quelqu'autre appareil. Le téléphone à cadran rotatif modifié présenté dans cet article remplit cette fonction en composant le numéro de l'appareil, tout en constituant un objet décoratif très apprécié.

Dans ce projet, nous transformons un vieux téléphone analogique à cadran rotatif en une télécommande et une alarme pour un système domotique. En plus d'être une télécommande (décorative), le téléphone modifié peut également être utilisé comme accessoire dans, par exemple, un jeu d'évasion. Je suis sûr que de nombreuses autres applications peuvent être imaginées.

Circuit basé sur le RP2040

Le cerveau de ce projet est un microcontrôleur RP2040 (MCU) monté sur une carte WIZnet W5100S-EVB-Pico (**figure 1**). Il s'agit essentiellement d'une carte Raspberry Pi Pico augmentée d'une puce « Internet » W5100S (et d'un connecteur Ethernet). Son brochage est donc compatible avec celui de la carte Pico, à ceci près que certaines de ses broches (GPIO16 à GPIO21) servent à communiquer avec la W5100S.

Câblé ou sans fil

Même si les réseaux sans fil semblent aujourd'hui être la norme, le câble n'a pas dit son dernier mot. L'un des avantages de l'Ethernet câblé est qu'il est possible d'alimenter les nœuds connectés, possibilité utilisée dans ce projet (voir ci-dessous). En outre, comme les téléphones à cadran rotatif ont toujours eu un fil à la patte, il serait curieux d'en voir un qui n'en a pas. La carte W5100S-EVB-Pico est donc un bon choix pour cette application.

Télécommande via MQTT

Le contrôleur domotique (HAC) contrôlé par ce téléphone est Home Assistant (HA), une plateforme d'automatisation populaire qui gagne du terrain chaque jour. Toutefois, étant donné que la télécommande utilise le protocole MQTT, elle peut facilement être intégrée à d'autres systèmes domotiques.

Connexion du microcontrôleur au téléphone

Le téléphone que j'ai utilisé, un modèle français classique (S63), produit des impulsions à une fréquence de 10 Hz. Le mécanisme de numérotation de ce vieux téléphone (**figure 2**) peut être considéré comme deux interrupteurs : l'un indique que la numérotation est en cours (actif/repos) tandis que l'autre se ferme un certain nombre de fois en

fonction du chiffre choisi. Un « 1 » produit une impulsion, un « 9 » neuf impulsions et un « 0 » dix impulsions.

Connecter ce mécanisme à la carte W5100S-EVB-Pico est facile ; le faire sans modifier le téléphone demande un peu plus de réflexion. J'ai voulu garder le téléphone aussi proche que possible de son état d'origine. J'y suis parvenu par un choix judicieux des points de connexion et des valeurs des résistances de rappel requises (**figure 3**).

Le combiné repose sur un interrupteur ouvert (NO) qui se ferme quand on décroche. Cela modifie le circuit électrique du téléphone, avec un effet sur l'impédance de certains points de connexion du mécanisme de numérotation, problème qu'il est possible de traiter en choisissant des valeurs relativement faibles pour certaines des résistances de rappel.

Alimentation de la sonnerie

La sonnerie du téléphone (**figure 4**) requiert une tension alternative relativement élevée, au moins 35 V_{AC} comme je l'ai constaté. Pour la produire, j'ai construit un simple générateur de faible puissance à partir de deux signaux carrés de 50 Hz en opposition de phase produits par le MCU, qui alimentent les secondaires d'un petit transformateur secteur de 230 V_{AC} dont le primaire est connecté à la sonnerie. Un petit transformateur 2× 9 V, 1 VA suffit pour obtenir un volume sonore appréciable. Le schéma complet de l'interface téléphone-microcontrôleur est représenté sur la **figure 5**.

Lecteur MP3

Un module de lecture MP3 a été ajouté pour écouter des fichiers audio avec l'écouteur du combiné. Ce module communique avec le MCU via un port série à 9 600 bauds. Le choix du fichier MP3 à lire peut se faire par envoi d'un message MQTT du HAC au téléphone ou être effectué par le seul MCU. Ce dispositif permet de jouer de la musique d'attente, de reproduire les tonalités de la ligne téléphonique analogique, de passer des messages préenregistrés ou de créer une horloge parlante. Les fichiers audio et musicaux sont stockés sur une carte microSD.

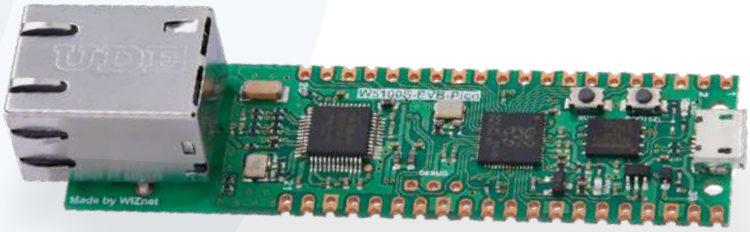


Figure 1. La carte WIZnet W5100S-EVB-Pico comporte un microcontrôleur RP2040 connecté à une puce Internet-sur-Ethernet W5100S. La carte a le même brochage que la carte Raspberry Pi Pico. Notez cependant que les broches GPIO16 à GPIO21 sont aussi connectés à la puce W5100S. (Source : WIZnet)

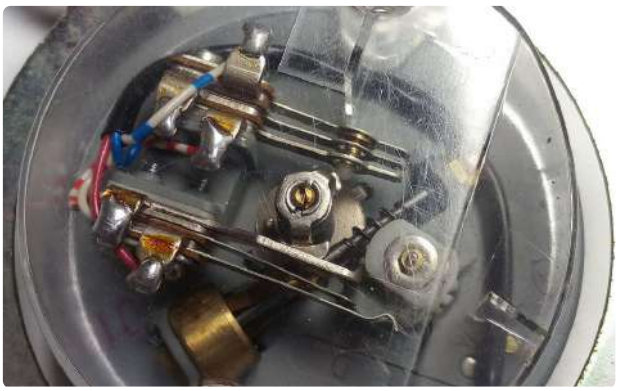


Figure 2. Le mécanisme de numérotation du téléphone S63 produit des impulsions à une fréquence de 10 Hz.

▼

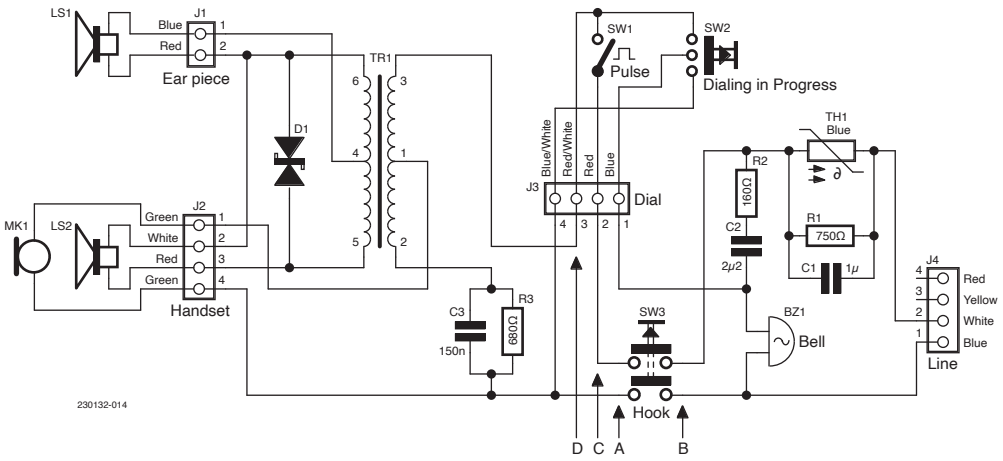


Figure 3. Pour éviter de modifier le téléphone, j'ai connecté la carte microcontrôleur aux points A (GND), B (décroché), C (impulsions de numérotation) et D (numérotation en cours).

Figure 4. Le mécanisme de la sonnerie de l'ancien téléphone.

En raison des contraintes d'espace, le lecteur MP3 est monté sur une deuxième carte d'essai avec le connecteur d'alimentation. Les deux cartes sont reliées par J3 et J4 (**figure 6**). L'écouteur du combiné est connecté directement à la sortie du lecteur MP3.

Alimentation électrique

La carte W5100S-EVB-Pico nécessite une alimentation de 5 V_{DC}. Le générateur de haute tension de la sonnerie a besoin d'au moins 10 V_{DC}. Les deux paires de fils libres d'un câble Ethernet normal sont utilisées pour alimenter le téléphone **(figure 7**). Mon câble de cinq mètres de long introduit une chute de tension notable, même avec deux conducteurs en parallèle pour chaque connexion d'alimentation. Le téléphone est donc alimenté par une prise murale de 12 V CA/CC. À l'intérieur du téléphone, un petit convertisseur CC/CC fournit les 5 V pour la carte MCU.

Logiciel

La carte W5100S-EVB-Pico est supportée par des bibliothèques destinées à être utilisées avec l'environnement de développement officiel du Raspberry Pi RP2040 MCU. Je n'aime pas trop cet environnement, car je le trouve complexe et encombré par tout le fouillis Cmake. Heureusement, l'EDI Arduino possède un gestionnaire de cartes à base de RP2040, et j'ai donc porté les bibliothèques WIZnet vers cet EDI. Cela rend les choses beaucoup plus simples et, je pense, plus accessibles et plus faciles à partager. Le programme d'application complet consiste maintenant en un croquis Arduino et une bibliothèque facile à installer [1].

MQTT

J'ai composé le programme pour la télécommande en me basant sur l'exemple MQTT fourni par WIZnet. Cela m'a permis d'être rapidement opérationnel. Le seul problème que j'ai rencontré était un appel manquant et non documenté à `MilliTimer_Handler` (voir le fichier *mqtt_interface.h*). Une fois que je l'ai ajouté au callback de mon timer milliseconde, MQTT a fonctionné correctement.

mDNS

Comme mon installation de Home Assistant utilise DHCP pour se connecter au réseau, une nouvelle adresse IP peut lui être attribuée à tout moment. Pour trouver ses périphériques et communiquer avec eux, HA utilise le DNS multicast, alias mDNS. Comme je ne voulais pas

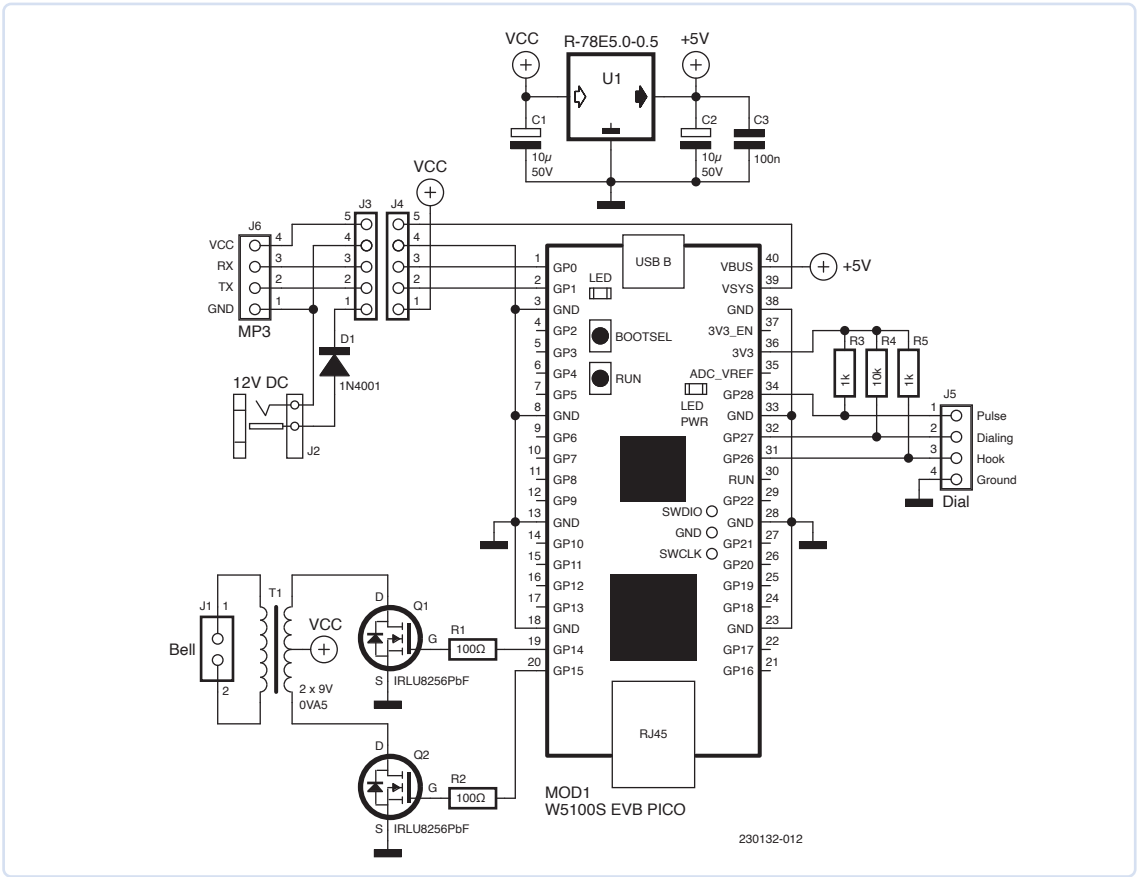


Figure 5. Il ne faut que quelques composants pour connecter le microcontrôleur au téléphone. Notez que les résistances de rappel pour les contacts de numérotation (R3, R4 et R5) ne sont pas toutes identiques.

coder en dur une adresse IP temporaire dans mon programme et le recompiler à chaque changement d'adresse du HAC, je lui ai ajouté le support mDNS. Pour cela, j'ai adapté la bibliothèque DNS de WIZnet, car le mDNS est assez similaire à un simple DNS. Désormais, le programme émet une requête mDNS pour obtenir l'adresse IP du CAH avant d'essayer de s'y connecter. Cela rend le système beaucoup plus souple et fiable.

Détection du combiné et numérotation

La lecture des contacts du mécanisme de numérotation nécessite un traitement antirebonds effectué par le logiciel. Cela fait, le comptage des impulsions devient une tâche banale.

La numérotation avec le combiné raccroché produit des messages à un chiffre ; quand il est décroché, on obtient un numéro à plusieurs chiffres, qui est envoyé quand on raccroche. Le chiffre ou le numéro constitue la charge utile d'un paquet MQTT. Notez que les chiffres sont envoyés « moins un » pour que les 10 valeurs tiennent dans un seul caractère. Ainsi, 1 est envoyé comme 0, 9 comme 8, et 0 est envoyé comme 9.

Des chiffres et des lettres

Lorsque le combiné est décroché, il est possible de composer non seulement des numéros à plusieurs chiffres, mais aussi des lettres pour former des messages alphabétiques. Les 26 lettres de l'alphabet sont réparties sur les chiffres 2 à 9, à raison de trois caractères par chiffre, sauf le 6, qui n'en a que deux (« m » et « n »). Le chiffre 0 en a également deux (« o » et « q »), le chiffre 1 n'en a aucun. Il s'agit de la répartition par défaut imprimée sur le téléphone français S63 (**figure 8**). Pour une obscure raison, il manque le « z », que j'ai ajouté au chiffre 0.

Pour sélectionner une lettre, il faut composer jusqu'à trois fois le chiffre correspondant. Par exemple : « a » est 2, « b » est 22 et « c » est 222. Pour « aaa », la séquence de composition est 2-2-2, pour « abc », c'est 2-22-222. Ici, le tiret représente une pause d'au moins une seconde. Un délai de moins d'une seconde entre deux chiffres identiques sélectionne la lettre suivante attribuée au chiffre. Cette méthode est très similaire à celle utilisée sur les téléphones portables à partir de l'an 2000 pour composer des textos, sauf qu'il n'y a pas d'affichage, fonction dévolue à votre mémoire (bon exercice !).

Quand on raccroche, le message composé est envoyé, avec le numéro à plusieurs chiffres, au HAC en tant que charge utile d'un paquet MQTT. Le HAC peut alors le transmettre, par exemple, à un service d'envoi de SMS ou l'afficher sur son tableau de bord.

Quelques variantes possibles

L'interface présentée dans cet article a été conçue pour fonctionner avec un téléphone français de type S63 et construite pour tenir à l'intérieur de celui-ci (**figure 9**). L'usage d'un autre modèle peut exiger d'en modifier certaines parties.

Au lieu d'Ethernet, vous pouvez préférer le Wi-Fi ou une autre méthode de communication sans fil. C'est bien sûr possible. Le logiciel peut facilement être adapté à d'autres couches physiques, car le programme d'application utilise une API de pilote de réseau standard. La pile TCP/IP est entièrement gérée par la puce W5100S

Figure 8. Il est facile d'envoyer des textos avec ce vieux téléphone grâce aux lettres imprimées autour du cadran. Par exemple, pour «elektor» on compose le 33-555-33-55-8-0-77, le tiret représentant une pause d'une seconde.

◄

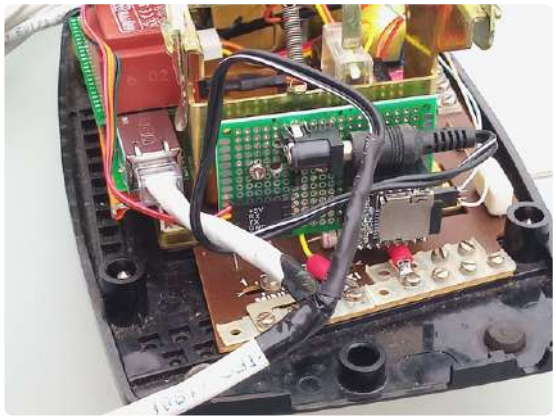


Figure 7. Réalisation pas très efficace de l'alimentation par Ethernet. Ça fonctionne tant que la tension d'entrée est suffisamment élevée pour compenser la chute de tension dans le câble.

◄

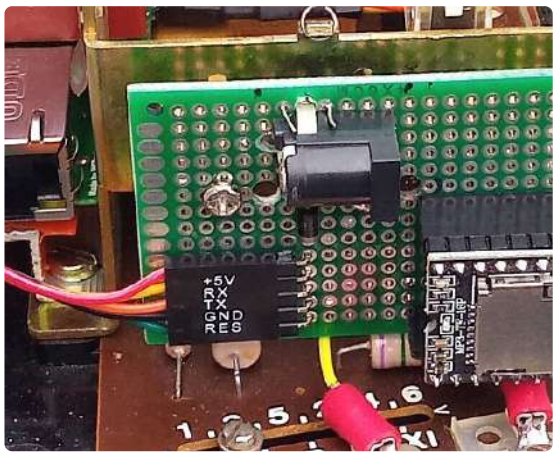


Figure 6. Le module de lecture MP3 et le connecteur d'alimentation partagent leur propre petite carte.

◄

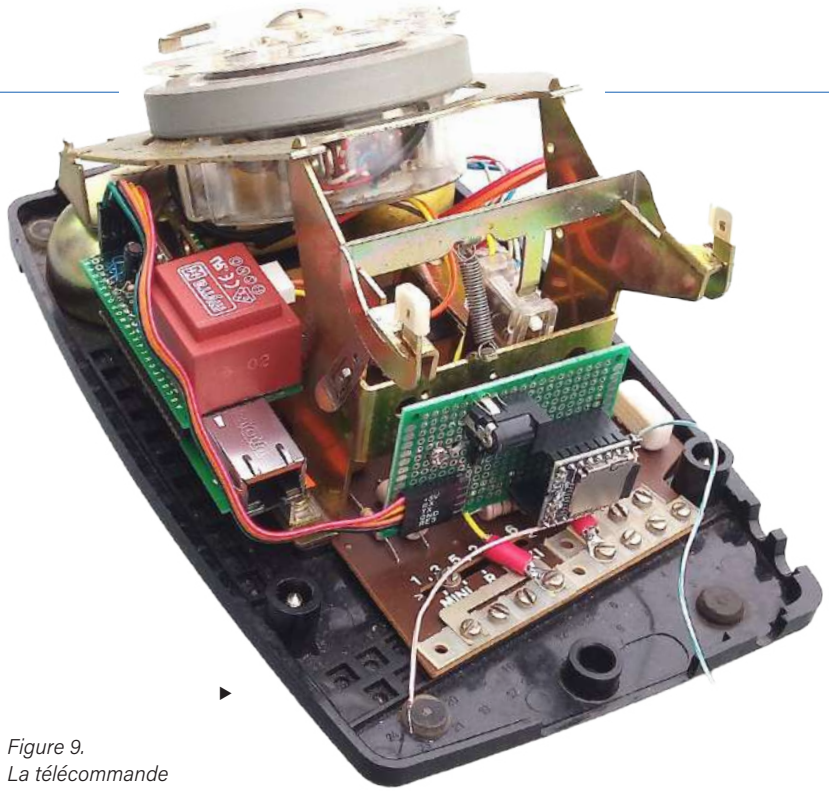


Figure 9. La télécommande par téléphone assemblée. Sur la gauche, on reconnaît le transformateur de l'alimentation haute tension de la sonnerie, monté sur la carte W5100S-EVB-Pico. Un câble relie cet ensemble au module du lecteur MP3 et au connecteur d'alimentation sur la droite. Les connexions au mécanisme de numérotation ne sont pas visibles, car situées de l'autre côté du téléphone.

et peut donc être remplacée par un autre module de communication.

L'ajout d'une bonne batterie (à recharger de temps à autre) permettrait de disposer d'une télécommande autonome. Si les circuits d'origine du téléphone peuvent être enlevés, il y a suffisamment d'espace pour en installer une.

L'alimentation à haute tension de la sonnerie peut être réalisée de différentes manières. J'ai utilisé un transformateur parce que j'en avais un sous la main, mais un convertisseur bon marché trouvé sur le web peut également faire l'affaire.

VoIP

Actuellement, le microphone du téléphone n'est pas utilisé, mais on pourrait imaginer de l'utiliser pour un service de voix sur internet ou VoIP. Le RP2040 prend en charge la norme I²S, ce qui permet d'utiliser des puces d'interface microphone standard.

L'application de télécommande consiste en un croquis Arduino et une bibliothèque Arduino facile à installer, basée sur le dépôt officiel de WIZnet. Elle est disponible sous [1].

Vf : Helmut Müller — 230132-04

Fonctions de la télécommande

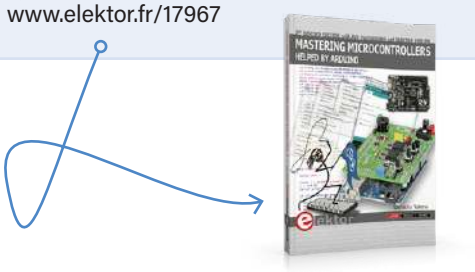
Le téléphone à cadran modifié a les fonctions suivantes :

- Lorsque le combiné est posé sur le téléphone et qu'un numéro est composé, l'appareil associé change d'état (arrêt / marche).
- Lorsque le combiné est décroché, le cadran peut être utilisé pour composer (laborieusement) un message texte, à la manière des téléphones portables d'il y a une vingtaine d'années. Le message est envoyé quand on raccroche.
- La sonnerie du téléphone est utilisable par le contrôleur domotique (HAC) comme alarme et peut, par exemple, être associée à la sonnette de la porte ou fonctionner comme minuterie de cuisine ou signal de réveil (désagréable).
- L'écouteur du téléphone est relié à un lecteur MP3 pour diffuser des messages préenregistrés ou de la musique, ce qui permet de réaliser, par exemple, une horloge parlante, fonction courante au siècle dernier.

La communication entre le téléphone et le HAC est basée sur MQTT, tandis que mDNS est utilisé pour établir automatiquement une connexion entre les deux.

Produits

- **Carte d'évaluation WIZnet W5100S-EVB-Pico à base de RP2040 (SKU 19971)**
  www.elektor.fr/19971
- **HAT Ethernet WIZnet pour Raspberry Pi Pico (SKU19970)**
  www.elektor.fr/19970
- **C. Valens, *Mastering Microcontrollers Helped by Arduino (3rd Edition)* (SKU 17967)**
  www.elektor.fr/17967



Des questions, des commentaires ?

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

LIENS

[1] Ce projet sur GitHub : https://github.com/polyvalens/rotary_dial_remote

contrôleur de vitesse par GPS

plus de contraventions pour excès de vitesse

Olivier Croiset (France)

Il existe de nombreux exemples sur l'internet montrant comment utiliser un module GPS dans un projet de microcontrôleur. Mais que faire des coordonnées GPS, si ce n'est les reporter sur une carte ? J'ai voulu faire quelque chose de plus utile et j'ai créé ce contrôleur de vitesse basé sur le GPS. Ce dispositif peut être utilisé dans n'importe quel véhicule, aussi bien une voiture qu'un bateau. Il vous permet de conduire sans avoir les yeux rivés sur le compteur de vitesse et d'éviter les mauvaises surprises par la suite dans le courrier.

L'appareil présenté ici n'est pas un limiteur de vitesse, qui nécessiterait une modification du véhicule, mais une alarme qui signale qu'une vitesse présélectionnée est atteinte (ou dépassée). Pour que vous ne quittiez pas la route des yeux, l'avertisseur émet deux longs bips si la vitesse sélectionnée est dépassée, tandis que deux bips courts indiquent que le véhicule est repassé sous la limite.

Quatre consignes de limites de vitesse

Le contrôleur de vitesse dispose de quatre valeurs programmables de limites de vitesse, ce qui devrait suffire pour les limites rencontrées sur la plupart des trajets. De toute façon, l'écran ne permet guère d'en afficher plus de quatre. Pendant la conduite, le conducteur doit être en mesure de choisir rapidement l'une des valeurs, si possible sans regarder l'appareil. Je n'ai pas voulu utiliser d'écran tactile ; le conducteur doit sentir physiquement les pressions exercées sur les boutons. Pour des raisons ergonomiques, les boutons sont placés au-dessus des quatre limites de vitesse affichées à l'écran.

Une bonne raison pour s'essayer au 32 bits

Mes premiers essais utilisant un microcontrôleur 8 bits ATmega328 associé à un module GPS m'ont rapidement convaincu que ses 32 ko de mémoire flash ne suffiraient pas pour une application plus évoluée, surtout avec un affichage graphique des données sur un écran TFT.
Par conséquent, après avoir épuisé les nombreuses qualités de l'ATmega328, j'ai décidé de passer au niveau au-dessus et d'essayer un microcontrôleur 32 bits STM32. Sa forme la plus adaptée aux bricoleurs de mon espèce est la carte BluePill avec son STM32F103C8. Cette petite carte facilite considérablement l'utilisation de ce microcontrôleur. Elle est facile à programmer via son port USB à l'aide de l'EDI Arduino (à condition que la carte soit programmée avec un chargeur adéquat au préalable), elle est compacte et ne coûte que quelques euros.

Tailles de la mémoire flash

Ce projet est une première tentative d'utilisation du STM32F103C8. Notez que, officiellement, le ***C8*** est équipé de 64 ko de mémoire flash, alors que la version ***CB*** en a 128 ko, mais il arrive que le C8 soit lui aussi équipé de 128 Ko (pièces de contrefaçon ?). Cela permettra, pour ceux qui seraient tentés, de poursuivre l'apprentissage en ajoutant d'autres périphériques, tels qu'une carte SD, un module SIM, la transmission de données par liaison radio, etc.

Le module GPS

Le module GPS utilisé dans ce projet est un module NEO-6 de u-blox. Son utilisation est facile, car la bibliothèque TinyGPS+ [2] se charge de décoder le flux de données au format NMEA 0183 émis par le module GPS. Les principaux paramètres que nous obtenons de cette manière sont les suivants :

- Date et heure UTC ;
- Longitude et latitude, en degrés décimaux ;
- Vitesse ;
- Direction (cap) ;
- Altitude ;
- Le nombre de satellites visibles ;
- La valeur de la précision horizontale (HDOP).

La précision horizontale, ou HDOP (horizontal dilution of precision) [2], dépend de la position des satellites à portée du récepteur GPS. Plus cette valeur est faible (proche de 1), meilleure est la précision des coordonnées. Une valeur HDOP de 10 indique que les coordonnées sont peu précises, voire invalides. Le contrôleur de vitesse n'utilise pas ce paramètre.

L'écran TFT

Pour l'affichage, j'ai utilisé un module d'affichage TFT standard de 2,2 pouces avec un pilote ILI9341 et une interface SPI. Il a une résolution de 320 × 240 pixels, ce qui est suffisant pour les données que nous voulons afficher (quatre limites de vitesse avec un bref message indiquant si la limite de vitesse a été atteinte ou non). Le contrôleur de vitesse dispose de deux modes d'affichage principaux en utilisation normale. Le mode d'affichage en cours est sauvegardé dans l'EEPROM et est restauré lorsque le système est remis en marche. Les préréglages des limites de vitesse sont également stockés dans l'EEPROM. Seuls six octets de cette EEPROM sont utilisés : quatre octets pour les limites de vitesse, un octet pour le dernier écran sélectionné et un octet pour la dernière limite de vitesse sélectionnée.

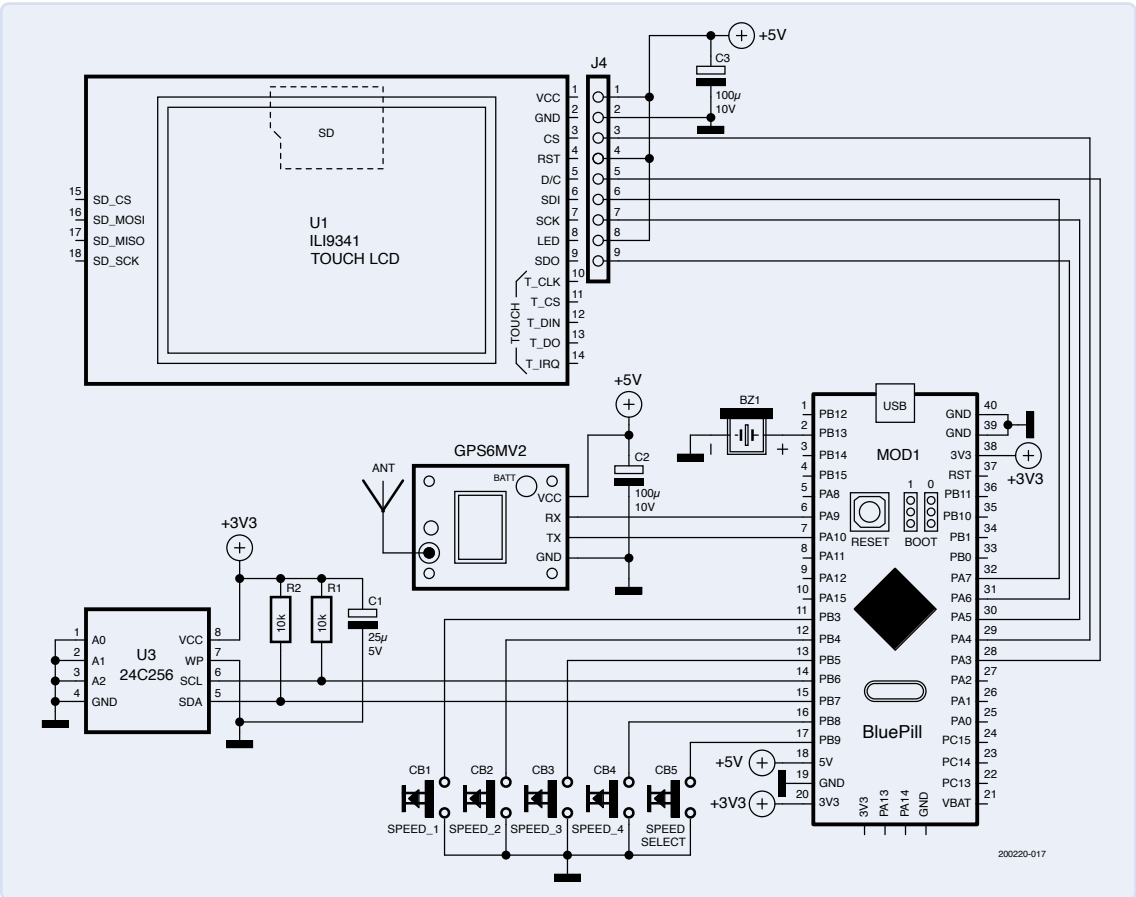


Figure 1. Le contrôleur de vitesse combine des modules facilement disponibles avec quelques composants électroniques.

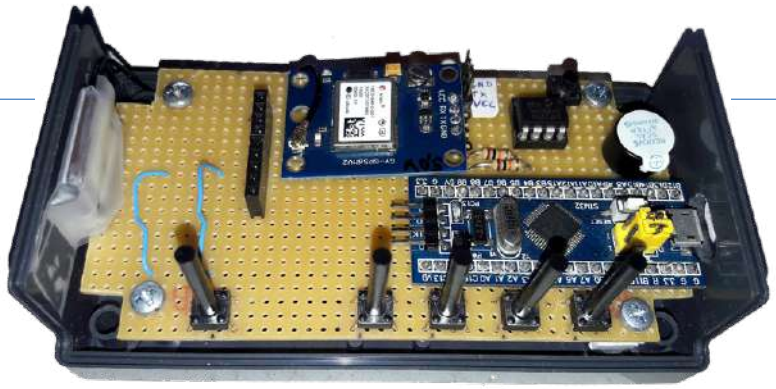


Figure 2. Le prototype a été construit sur une carte de prototypage. Les dimensions du boîtier sont de 12 x 6,5 x 4 cm.

Le schéma du circuit

Le schéma du contrôleur de vitesse est présenté à la **figure 1**. Le module BluePill dans le coin inférieur droit est le cœur du circuit.

L'alimentation électrique se fait par le biais du connecteur USB de la BluePill. Les véhicules récents sont souvent équipés d'un connecteur USB ; dans le cas contraire, un adaptateur 12 V vers USB est nécessaire (ou une batterie powerbank). Le module BluePill est alimenté par une tension de 5 V ; il possède son propre régulateur de 3,3 V, qui alimente l'EEPROM. L'écran TFT et le module GPS sont alimentés en 5 V. La consommation totale du circuit est d'environ 200 mA.

L'écran est contrôlé par un bus SPI, le module GPS est contrôlé par une liaison série et l'EEPROM est connectée au bus I²C (avec ses deux résistances de rappel). Pour mon prototype, j'ai utilisé une EEPROM 24C256 de 32 Ko, mais une 24C01 de 128 octets serait déjà largement suffisante.

Douze ports GPIO restent libres. Il est donc possible d'ajouter des périphériques, des éléments et des fonctions supplémentaires, tant que le programme d'application tient dans la mémoire flash. Le microprogramme compilé de ce projet occupe 64 020 octets, soit la quasi-totalité de la mémoire flash.

Construction d'un contrôleur de vitesse

Le critère le plus important à prendre en compte dans la réalisation de ce projet est l'ergonomie. Le conducteur doit pouvoir trouver les boutons sans se tromper. Les boutons-poussoirs ont une tige de 25 mm – la plus longue que j'ai pu trouver.

J'ai réalisé ce prototype sur une carte de prototypage (**figure 2**). Un circuit imprimé (PCB) permettrait une réalisation plus compacte dans un boîtier plus fin. Le connecteur USB doit rester accessible, car il sert à alimenter le circuit en utilisation normale. Il est également utilisé pour mettre à jour le logiciel.

L'antenne GPS doit être aussi éloignée que possible du connecteur de l'écran TFT (au moins quelques centimètres).

Notes sur le module GPS

Le module GPS est doté d'une minuscule batterie de sauvegarde des données. Si le GPS n'est pas utilisé pendant plusieurs jours d'affilée, la batterie se décharge et les données sont perdues. Lorsque vous rallumez le GPS, il se peut que vous deviez attendre quelques minutes pour restaurer les données (et recharger la batterie). La LED du module GPS clignote lorsqu'il réussit à décoder les données satellitaires. Quelques instants plus tard, le nombre de satellites s'affiche.

L'antenne GPS doit « voir » les satellites. La réception des signaux doit donc être entravée le moins possible. En zone urbaine, entre deux bâtiments, la réception peut être difficile. Néanmoins, le contrôleur de vitesse placé au niveau des pieds du passager dans ma voiture fonctionne correctement.

Réglage des limites de vitesse

Cette opération ne doit être effectuée que lorsque le véhicule est à l'arrêt ! Lors de la première mise sous tension du contrôleur de vitesse, les limites de vitesse sont toutes réglées sur 255. Cela correspond à FF en hexadécimal, ce que contient une EEPROM vierge.

Dans l'écran 3 (**figure 5**), qui affiche les données des satellites, appuyez sur le troisième bouton pour activer le réglage de la limite de vitesse. Ensuite, utilisez le premier et le deuxième bouton pour régler la limite souhaitée. Appuyez à nouveau sur le troisième bouton pour valider la limite de vitesse et passer à la suivante, et ainsi de suite. Veillez à ce que les limites soient dans un ordre croissant, car le logiciel ne les trie pas.

Figure 3. L'écran 1 montre les limitations de vitesse sous la forme de panneaux de signalisation ronds accompagnés d'un bref message.

Figure 4. Sur l'écran 2, la vitesse réelle est affichée en chiffres de style 7 segments.

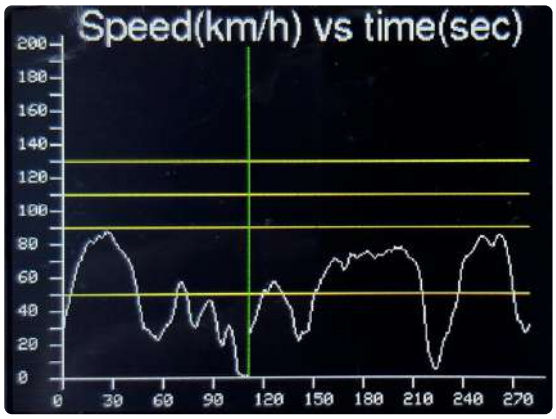


Figure 5. L'écran 5 montre la courbe de l'historique de la vitesse. Les quatre lignes horizontales jaunes indiquent les quatre limites programmées et la ligne verticale verte indique la vitesse actuelle.

Figure 6. Les données GPS sont disponibles sur l'écran 3. C'est également par cet écran qu'on accède à l'écran 4 de programmation des valeurs des limites de vitesse.

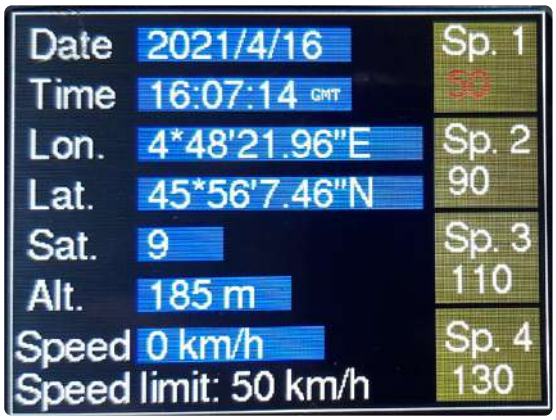


Figure 7. L'écran 4 permet de programmer les limites de vitesse. Ici, la première limite, « Sp. 1 », est en cours de réglage (en rouge, si vous regardez bien).

Utilisation du contrôleur de vitesse

Le contrôleur de vitesse est facile à utiliser. Toutefois, comme il est censé être utilisé dans un véhicule en mouvement, veillez à ce que l'appareil et surtout son câble d'alimentation ne soient pas gênants pendant la conduite. Une fois l'appareil placé à un endroit approprié, sélectionnez le mode d'affichage souhaité. L'écran 1 (**figure 3**) affiche des panneaux de signalisation ronds, l'écran 2 affiche la vitesse actuelle sous forme de chiffres à 7 segments (**figure 4**). Sélectionnez ensuite la limite de vitesse que vous souhaitez utiliser. L'avertisseur sonore retentit si vous roulez trop vite.

Votre passager peut visualiser la courbe de vitesse sur une période de 4,5 minutes (270 secondes) en sélectionnant l'écran 5 (**figure 5**). Sur cet écran, l'avertisseur sonore ne retentit pas. Les lignes horizontales indiquent les quatre limitations de vitesse. La ligne verticale «maintenant» indique la vitesse actuelle. Un graphique défilant aurait été plus joli, mais il nécessite une actualisation complète pour chaque nouveau point de données, ce qui est lent. Il est beaucoup plus facile de déplacer le curseur.

Écran des données GPS

L'écran 3 (**figure 6**) affiche les données GPS et vous permet de vous situer sur une carte à l'aide de marqueurs de coordonnées. La précision est d'environ 30 mètres, ce qui est tout à fait satisfaisant pour notre petit appareil bon marché. Dans ce mode d'affichage, l'avertisseur reste muet. L'écran 4 (**figure 7**) est identique à l'écran 3, sauf qu'il permet de programmer les limitations de vitesse. Notez que la date et l'heure du GPS sont des valeurs UTC, c'est-à-dire celles du méridien de Greenwich. Si vous souhaitez les utiliser, vous devez corriger ces valeurs en fonction de votre fuseau horaire local et de l'heure d'été ou d'hiver. N'étant pas une horloge, le contrôleur de vitesse n'inclut pas de menu de gestion de l'écart.

Tableau 1. Les différents écrans et comment naviguer.

Écran	Bouton 1	Bouton 2	Bouton 3	Bouton 4	Bouton 5
Ecran 1 (Panneaux ronds)	Limite de vitesse 1	Limite de vitesse 2	Limite de vitesse 3	Limite de vitesse 4	Aller à l'écran 2
Ecran 2 (Affichage 7 segments)	Limite de vitesse 1	Limite de vitesse 2	Limite de vitesse 3	Limite de vitesse 4	Aller à l'écran 3
Ecran 3 (Données GPS)	Limite de vitesse 1	Limite de vitesse 2	Aller à l'écran 4	Pas d'action	Aller à l'écran 1
Ecran 4 (Programmer les limites)	Diminuer la limite (-5 km/h)	Augmenter la limite (+5 km/h)	Valider et passer à la limite suivante, Retourner à l'écran 3 une fois terminé	Aller à l'écran 5	Pas d'action
Écran 5 (Courbe de vitesse)	Pas d'action	Pas d'action	Pas d'action	Aller à l'écran 2	Pas d'action

Logiciel

Le programme est facile à modifier et le code source peut être téléchargé à partir de [3]. Il s'agit d'un croquis Arduino, qui nécessite le STM32 Boards Package for Arduino (nous avons utilisé la version 2.3.0) [4]. La majeure partie du code est consacrée à l'interface utilisateur graphique, le reste se charge de la tâche simple consistant à recevoir les données GPS, à extraire la vitesse et à la comparer à une limite.

Comme d'habitude dans un sketch Arduino, la fonction `setup()` se charge d'initialiser tous les périphériques. Une fois cette opération terminée, elle affiche un écran de bienvenue pendant cinq secondes.

La fonction `loop()` commence par lire les boutons-poussoirs avant de vérifier si le GPS contient des données fraîches (dans la fonction `dataDecode()`). Ensuite, en fonction de l'écran actif, les données correspondantes sont affichées. Si la vitesse actuelle est supérieure à la limite de vitesse active, une alarme retentit.

Notez que des informations de débogage et d'état sont envoyées vers le port série (115200,N,8,1).

Pour aller plus loin

Voici quelques éléments que vous pouvez ajouter ou modifier :

- Utiliser d'autres unités de vitesse, pour un bateau, par exemple. La bibliothèque GPS vous permet d'utiliser d'autres unités de vitesse, telles que les nœuds ou les miles/heure (MPH). Il faudra pour cela modifier les écrans (par exemple, remplacer les km/h par des nœuds) et la taille des incréments de limite de vitesse (dans la fonction `SpeedSettings()`).

Si votre module BluePill dispose de 128 Ko de mémoire flash, des fonctions supplémentaires peuvent être ajoutées, telles que :

- Enregistrement sur carte SD. Au dos du module d'affichage TFT se trouve un emplacement pour carte SD. Il peut être utilisé pour enregistrer, par exemple, les données GPS afin de les consulter sur un PC.
- Ajouter un écran tactile pour de nouvelles fonctions. Dans ce cas, veillez à la sécurité de l'utilisateur.
- Réglage de l'heure locale.

Vf : Helmut Müller — 200220-04

Des questions, des commentaires ?

Si vous avez des questions techniques, n'hésitez pas à envoyer un courriel à l'équipe éditoriale d'Elektor à l'adresse redaction@elektor.fr.

Produits

- **OPEN-SMART GPS - Module GPS série pour Arduino (SKU 18733)**
  https://elektor.fr/18733
- **Module d'affichage TFT SPI 2,2 pouces ILI9341 (SKU 18419)**
  https://elektor.fr/18419
- **Majid Pakdel, *Advanced Programming with STM32 Microcontrollers* (SKU 19520)**
  https://elektor.fr/19520

LIENS

[1] Bibliothèque TinyGPS++ : http://arduiniana.org/libraries/tinygpsplus/
[2] Qu'est-ce que le POPD ? : https://fr.wikipedia.org/wiki/Geometric_dilution_of_precision
[3] Fichiers du projet sur Elektor Labs : https://elektormagazine.fr/labs/save-money-with-this-speed-monitoring-by-gps
[4] EDI Arduino : paquet de support des cartes STM32 : https://github.com/stm32duino/BoardManagerFiles/raw/main/package_stmicroelectronics_index.json

stroboscope RVB avec Arduino

un instrument utile, instructif et distrayant

Roel Arits (Pays-Bas)

Le dictionnaire nous apprend que le stroboscope (du grec *strobos*, tournant) est un instrument destiné à faire apparaître immobiles ou animés d'un mouvement lent des objets animés d'un mouvement périodique rapide. Une succession d'éclairs à une fréquence appropriée permettent de ne voir qu'une phase déterminée du phénomène : l'objet mobile paraît immobile ou il semble se mouvoir au ralenti.

Il y a quelques décennies, le stroboscope avait trouvé une application très courante, dans les garages automobiles. Pour qu'un moteur à combustion interne à essence fonctionne correctement, les bougies d'allumage de chaque cylindre doivent produire une étincelle à un moment précis. Cette distribution était assurée par un dispositif mécanique (sensible à l'humidité dans certaines marques de voitures), qui devait être synchronisé avec précision avec le moteur. Et c'est là que notre stroboscope entre en jeu.

Le nombre d'interrupteurs du distributeur correspondait à celui des cylindres du moteur. Chaque fois qu'un tel contact s'ouvrait, l'énergie électrique emmagasinée dans la bobine d'allumage était déchargée à travers la bougie d'allumage dans la chambre de combustion sous la forme d'une étincelle qui enflammait le mélange air-carburant. Et cela devait se produire au moment précis où le piston dans le cylindre venait de quitter son point mort haut (c'est-à-dire que le mélange avait atteint sa compression maximale). Les pistons du moteur sont solidaires du vilebrequin, au bout duquel est arrimée la poulie de la courroie trapézoïdale. La position de la poulie est synchrone avec la position du piston. Sur cette poulie se trouvait une marque blanche et à côté, sur un support métallique, une deuxième marque fixe.

Le stroboscope utilisé pour régler l'allumage était constitué d'un tube à éclats au xénon ou au néon et d'un câble de déclenchement relié à la bougie de référence par un capteur inductif, de sorte que chaque fois que la bougie de référence s'allumait, le stroboscope produisait un éclair.

Puisque le stroboscope était déclenché par la bougie de référence, les deux marques devaient apparaître exactement en face l'une de l'autre en raison de l'effet stroboscopique. En cas de décalage entre les deux marques, le distributeur devait être ajusté. Dans cette application, le stroboscope était synchronisé avec le mouvement de rotation, mais cela fonctionne aussi sans signal de déclenchement !

Sans déclenchement ?

Pour mesurer une rotation avec un stroboscope non synchronisé, il faut ajuster la fréquence des éclairs de sorte que la marque sur le disque tournant semble quasi immobile et qu'une seule marque soit visible. Si la roue tourne dans le sens horaire et que le marqueur semble se déplacer lentement dans le sens horaire, c'est que la fréquence de clignotement est un peu faible. L'éclair arrive un peu trop tard, et la marque semble se déplacer dans le sens de rotation. Si la roue tourne

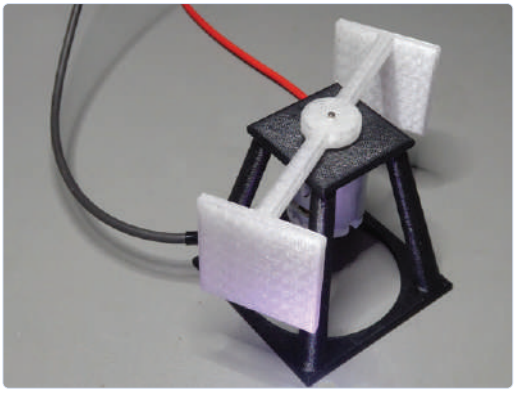


Figure 1. Le cadre avec les réflecteurs fixés à l'hélice.

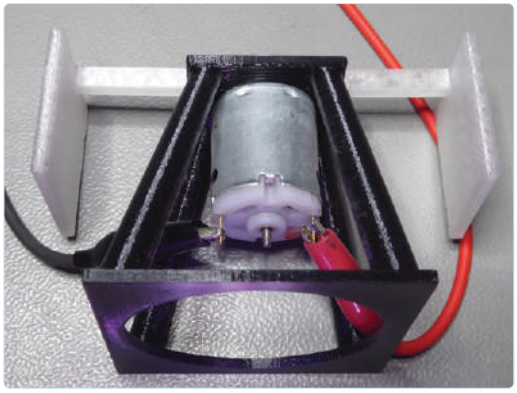


Figure 2. Gros plan du moteur.

dans le sens antihoraire et que le marqueur semble se déplacer lentement dans le sens horaire, c'est que la fréquence des éclairs est un peu trop élevée : chaque éclair arrive un peu trop tôt. Si plusieurs marques quasi fixes apparaissent, c'est que la fréquence des éclairs est un (sous-)multiple de la vitesse de rotation.
La vitesse peut être mesurée en ajustant la fréquence de clignotement du stroboscope de manière à immobiliser le marqueur. La durée de chaque éclair doit être assez courte pour obtenir une marque nette. Si la durée du flash est trop longue par rapport à sa fréquence, la marque bave.

Éclairs bariolés…

Et si notre stroboscope produisait plusieurs éclairs, de même fréquence mais de couleur différente, et avec un déphasage entre les différents éclairs ? Et si nous faisions tourner un objet blanc sur lui-même tout en variant fréquence, déphasage et largeur des éclairs colorés ?
Ce n'est pas difficile avec un microcontrôleur. Pour produire les éclairs de couleur, nous pouvons utiliser une LED RGB – ou mieux encore trois LED obtenir des éclairs plus lumineux. Pour commander la fréquence, le déphasage et la durée des éclairs, ce sera un Arduino Pro Mini, une carte qui dispose de suffisamment de ports d'entrée/sortie et largement assez rapide pour cette tâche. Les LED seront commandées par des signaux modulés en largeur d'impulsion (PWM).
Nous aurions pu utiliser trois modules PWM spéciaux de l'Arduino Pro Mini, mais ceux-ci utilisent trois temporisateurs différents. Il est donc plus délicat de les synchroniser et de programmer un déphasage précis. La fréquence des signaux PWM requis est relativement basse et une résolution de 16 bits hors de proportion avec nos besoins. Des signaux PWM produits par du logiciel dit *softPWM* feront amplement l'affaire. Le *softPWM* émet ses signaux sur des sorties numériques normales. Celles-ci sont commandées et initialisées à l'aide d'un compteur. Une interruption est émise lorsqu'un certain compte est atteint. De cette façon, on obtient un intervalle suffisamment court pour ajuster la largeur d'impulsion ou le déphasage du signal PWM avec une précision suffisante.

Pratique

Pour faire tourner l'hélice, nous avons un moteur 12 V à courant continu (MOT3N) pour lequel j'ai confectionné un petit support. L'hélice est prolongée par deux réflecteurs disposés verticalement. Ce dispositif (**fig. 1 et 2**) a été produit avec une imprimante 3D au moyen de fichiers téléchargeables [3].
Note : à vide, le moteur 12 V utilisé tourne à environ 11500 tr/min, ce qui s'est excessif pour l'hélice imprimée en 3D. Lors du premier essai, elle s'est brisée en mille morceaux sous la force centrifuge. Au cours de telles expérimentations, le port de lunettes de protection est recommandé !
Pour le ralentir, le moteur fonctionne donc sous seulement 3 V environ. La tension est réglable pour synchroniser la vitesse. Sous 3 V, la vitesse est d'environ 2100 tr/min, ce qui correspond à 35 Hz. Comme l'hélice est équipée de deux réflecteurs (ce qui en facilite l'équilibrage), la fréquence d'éclair est deux fois plus élevée, c'est-à-dire 70 Hz (durée de la période environ 14 ms).
Le circuit (**fig. 3**) est la simplicité même et peut facilement être construit sur une plaque d'essais (**fig. 4**)**.** Les trois LED RVB sont pilotées par de modestes BC547 sous les ordres de l'Arduino. La tension d'alimentation du circuit est de 5 V continus, la consommation est d'environ 500 mA.
Le stroboscope est commandé par une minuterie d'interruption appelée toutes les 100 µs. Cet intervalle de 0,1 ms est la résolution avec laquelle on peut faire varier la fréquence de l'éclair. Cela laisse assez de temps pour modifier l'image stable produite par l'hélice.

```
// Configuration d'une minuterie de 16 bits en
   fonctionnement normal avec interruption à 100 µs
// 16 MHz/1 = 6,25 ns. Pour obtenir 100us, nous
   devons donc laisser le minuteur compter 1600
   ticks.
TCCR1A = 0;
TCCR1B = _BV(CS10); //prédiviseur - diviser par 1
TCNT1 = 0xFFFF - 1600; // dépassemnt à 65535 = 0xFFFF
TIMSK1 = _BV(TOIE1); // interruption de dépassemnt
TCNT1 = 0;
sei();
}
ISR (TIMER1_OVF_vect)
{
TCNT1 = 0xFFFF - 1600; //100 us interruption
...
```

La période d'éclair est codée en dur dans le logiciel à 144 – c'est le nombre d'interruptions de la minuterie entre deux éclairs consécu-

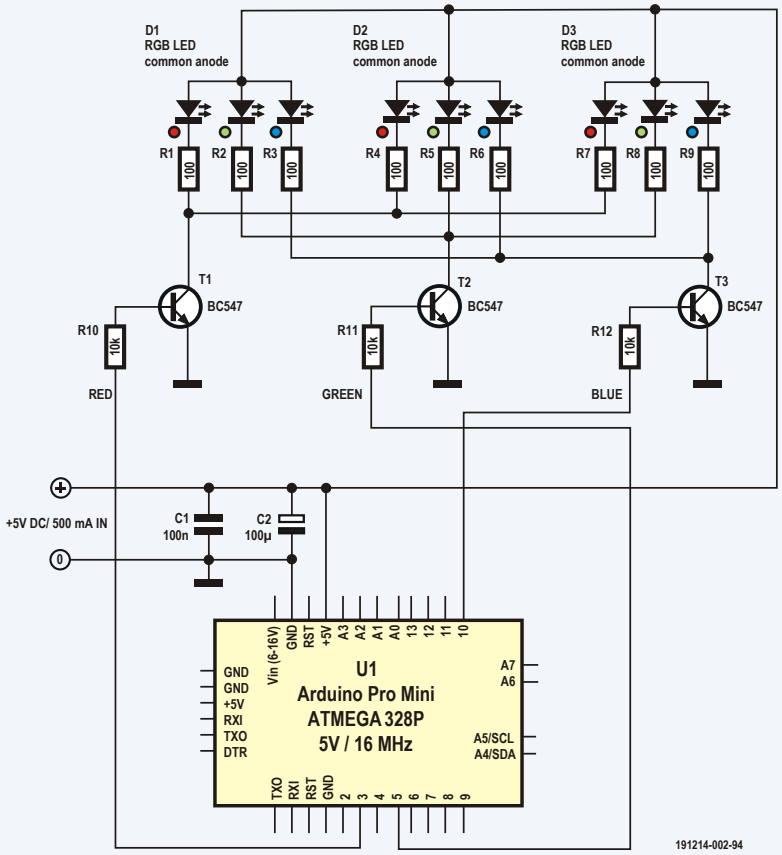


Figure 3. Pas trop compliqué, mon circuit ?

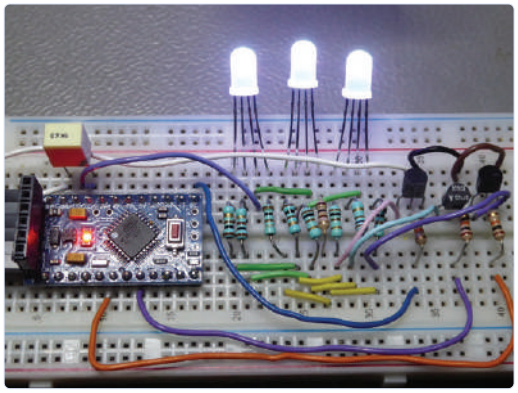


Figure 4. Pas trop critique, tout ça. Pour les expériences, une plaque d'expérimentation fera l'affaire.

tifs. La période est donc de 144 * 100 µs = 14,4 ms. La durée d'éclair de chaque LED est fixée à 8, ce qui correspond à 8 * 100 µs = 800 µs. Le cycle d'utilisation (rapport impulsion/pause) est donc de 800 µs / 14,4 ms = 5,5 %.

```
#define STROBE_PERIOD 144

TSoftPwm Pwm[] = {TSoftPwm(ID_RED, 0, 8,
   STROBE_PERIOD),
TSoftPwm(ID_GREEN, 0, 8, STROBE_PERIOD),
TSoftPwm(ID_BLUE, 0, 8, STROBE_PERIOD)};
```

Avec ce rapport cyclique , nous obtenons, une fois le régime moteur établi de manière optimale, une belle image stable.

Le circuit, y compris le logiciel téléchargeable sur la page de cet article [4], n'est pas un projet achevé. C'est pourquoi il n'y a pas (encore) d'interface confortable pour l'utilisateur. Cependant le logiciel montre bien les possibilités en jouant avec les paramètres : fréquence d'éclair, durée d'éclair et différence de phase. Le logiciel est commenté généreusement, inutile d'entrer dans les détails ici.

Un stroboscope non synchronisé n'est pas vraiment optimal. On pourrait monter une barrière lumineuse sur l'hélice, et en utiliser le signal pour déclencher le stroboscope. À vous l'initiative !
Mes deux vidéos sur Youtube permettent d'admirer le fonctionnement du stroboscope [1] et le rapport des signaux de commande des LED sur l'écran d'un oscilloscope [2] avec leur effet sur l'image stroboscopique perçue. Le projet peut également être consulté sur le site d'Elektor Labs [3].

191214-04

Produits

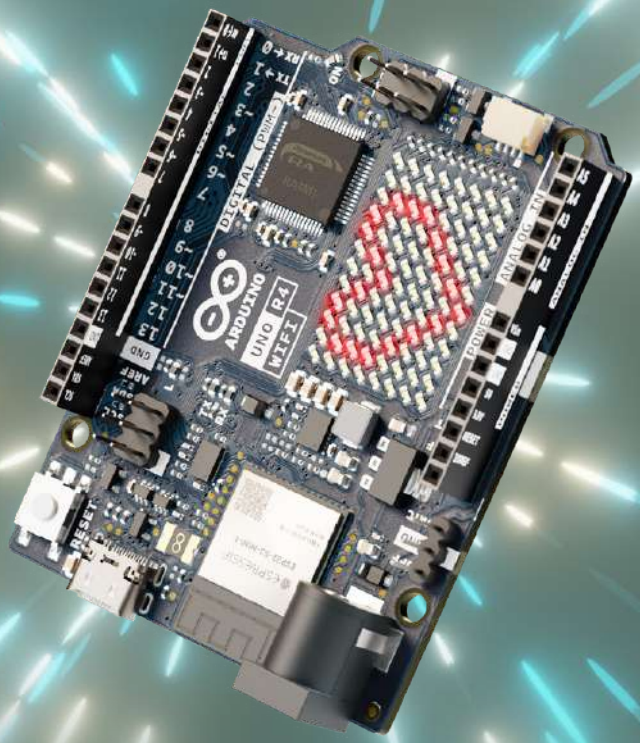
- **livre en français avec kit : Tuto Arduino Uno**
  www.elektor.fr/tuto-arduino-uno
- **livre en français : Arduino – Faites-le jouer au train**
  www.elektor.fr/arduino-faites-le-jouer-au-train
- **e-book en français :**
  36 Expériences de Physique avec Arduino
  www.elektor.fr/
  36-experiences-de-physique-avec-arduino-e-book

LIENS

[1] Vidéo de démonstration 1 : https://youtu.be/WHv6DCWM82k
[2] Vidéo de démonstration 2 : https://youtu.be/3tYqUin4-vQ
[3] Projet sur Elektor Labs : www.elektormagazine.fr/labs/arduino-rgb-color-stroboscope
[4] Page en ligne de cet article : www.elektormagazine.fr/191214-04

ARDUINO

UNO R4



La nouvelle dimension des Makers

Repoussez toutes vos limites grâce à un microcontrôleur convivial et accessible. Avec l'Arduino UNO R4, embarquez dans un univers en pleine expansion!

Mémoire plus rapide et plus grande - 48 MHz, 256 kB Flash, 32 kB RAM - parfait pour effectuer des calculs plus précis et gérer des projets plus complexes.

Nouveaux périphériques intégrés - Le RA4M1 (Arm® Cortex®-M4 Core avec FPU) comprend un DAC, un CAN-BUS, un support HID et un OpAMP incorporé permettant de réaliser des projets plus avancés.

Développez vos projets avec le shield UNO (compatible 5 V) et le vaste catalogue de nœuds *Qwiic compatibles.

Connectivité *Coprocesseur ESP32-S3 - connectivité Wi-Fi® et Bluetooth® Low Energy, laissant le microcontrôleur RA4M1 libre.

Robuste de par sa conception - avec une plus grande plage de tension, 6 - 24 V, vous pouvez maintenant utiliser l'UNO R4 en combinaison avec des moteurs, des bandes LED ou d'autres actionneurs à partir d'une seule source d'alimentation.

UNO R4 WiFi

UNO R4 Minima

*Note - ces fonctionnalités ne sont disponibles que sur le modèle UNO R4 WiFi.

bouton-poussoir d'urgence sans fil

sécurité renforcée avec LoRa

Somnath Bera (Inde)

Le bouton-poussoir d'urgence sans fil décrit dans cet article est un exemple de solution à un problème dans un environnement industriel complexe avec quelques modules électroniques bon marché et un peu de programmation Arduino. Le résultat est un système sans fil simple qui peut s'avérer utile dans de nombreuses autres situations.

Le besoin initial de développer le bouton-poussoir d'urgence (*Emergency Push Button* : EPB) sans fil présenté ici est assez spécifique et concerne le prélèvement d'échantillons dans les trains livrant du charbon à une centrale électrique. On doit prélever un échantillon au sommet d'un wagon de charbon pour déterminer son pouvoir calorifique avant de pouvoir l'utiliser. Il s'agit d'un paramètre important pour les centrales électriques. Avant de prélever un échantillon, l'échantillonneur appuie sur l'EPB existant pour signaler sa présence au conducteur du train. Cette action devrait immobiliser le train. Malheureusement, en fonction de la longueur du train, de la position d'échantillonnage et de la courbure de la voie, l'échantillonneur ne peut pas toujours voir son signal devant le train. Il est arrivé, lors de la collecte d'échantillons, que le train roule par inadvertance, provoquant des accidents et blessant la personne. Le système EPB devait être amélioré.

Ce qu'il faut savoir

Les trains de charbon peuvent être très longs, jusqu'à 500 meters, et il peut y en avoir jusqu'à quatre les uns à côté des autres pour le prélèvement et le déchargement en même temps. Pensez aux voies ferrées courbes, et vous réaliserez qu'il est difficile de voir ce qui se passe. De plus, l'environnement est bruyant à cause du déchargement des trains. Le retour d'information sonore est donc également problématique.

Le système EPB existant est un système câblé. Cependant, dans un environnement comme celui décrit ci-dessus, il est dangereux d'utiliser de longs câbles susceptibles de se rompre sans que personne ne s'en aperçoive. Le remplacement de ces câbles ou l'acheminement de plusieurs câbles dans le but d'obtenir un retour d'information visuel est compliqué et coûteux. Le fait de placer le mât de signalisation plus haut pour qu'il soit visible de loin n'est pas non plus une solution pratique, car il

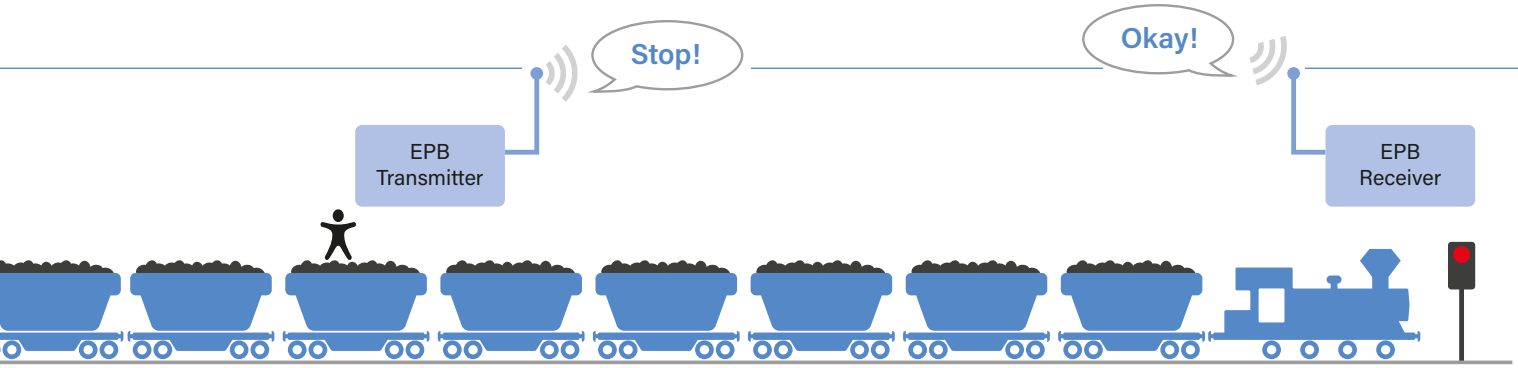


Figure 1. Aperçu du système du bouton-poussoir d'urgence sans fil (EPB).

faut avoir une bonne vision pour voir à quelle voie le signal s'applique.

Le sans-fil avec LoRa

La solution que nous avons trouvée est l'EPB sans fil (**figure 1**). Il se compose de deux unités : un émetteur EPB et un récepteur EPB. En réalité, les unités sont presque identiques et capables de transmettre et de recevoir, mais c'est ainsi que nous les désignerons. Vous pouvez le considérer comme un système maître-esclave ou client-serveur. Avant de prélever un échantillon de charbon, l'échantillonneur appuie sur un bouton de l'émetteur EPB pour signaler sa demande d'échantillonnage. Lorsque le récepteur EPB situé dans la cabine de contrôle reçoit cette demande, il active un relais pour appuyer sur le bouton du système EPB existant et renvoie un accusé de réception à l'émetteur EPB. L'échantillonneur sait alors que son signal est reçu et qu'il peut prélever un échantillon en toute sécurité. Ensuite, l'échantillonneur appuie sur un deuxième bouton de l'émetteur EPB pour débloquer le système. Cette fois, le récepteur EPB relâche l'EPB existant en activant un second relais et confirme à nouveau l'opération à l'émetteur EPB. Le système est alors prêt pour une nouvelle prise d'échantillon.

Si le système n'est pas disponible ou si le signal de l'EPB est interrompu pour des raisons opérationnelles, le signal de retour ne reviendra jamais à l'unité émettrice, ce qui permet d'alerter l'émetteur et d'éviter les erreurs de communication.

Émetteur et récepteur EPB : le circuit

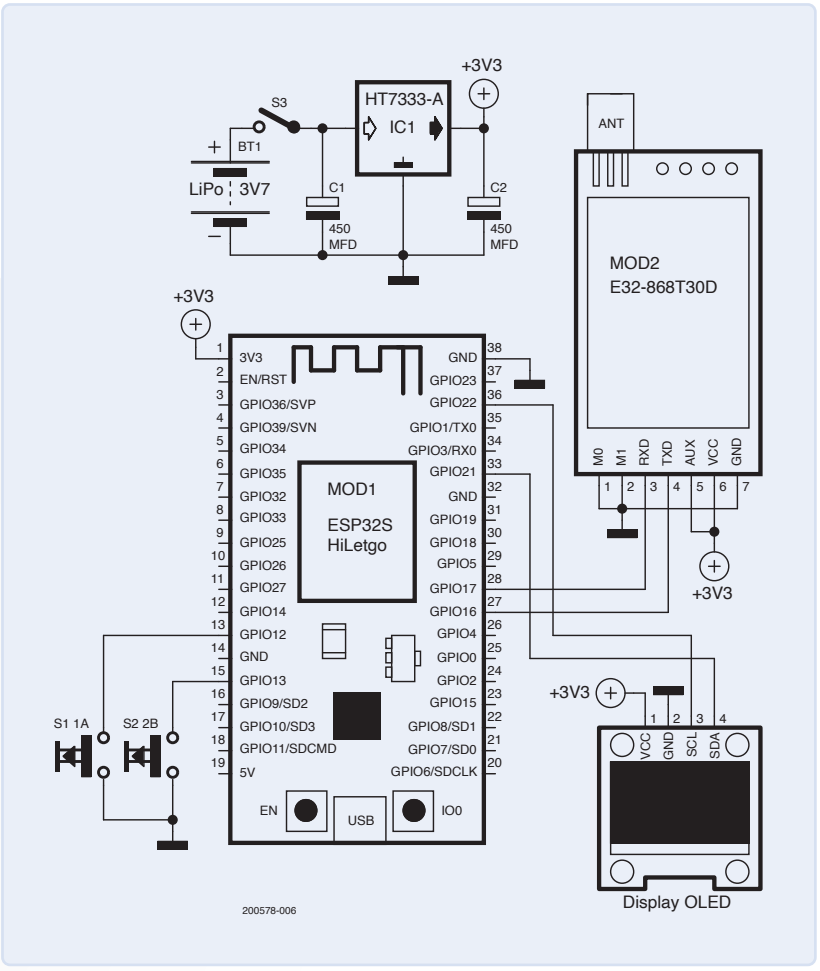
L'émetteur EPB est une réplique exacte de l'unité EPB existante, mais avec une petite antenne et un petit afficheur OLED. Il est alimenté par une batterie LiPo à une cellule. Il est doté de deux boutons poussoirs d'activation. À l'intérieur (**figure 2**) se trouve un module ESP32 qui commande l'afficheur OLED et un module émetteur-récepteur LoRa. Notez que le module ESP32 n'a été utilisé que pour des raisons de commodité, et non pour ses capacités sans fil. Vous pouvez utiliser n'importe quel autre module microcontrôleur bon marché, à faible consommation et doté des bonnes interfaces (I^2C, UART, 2× GPIO).

Le récepteur EPB est identique à l'unité émettrice, sauf qu'il possède deux relais à la place des boutons-poussoirs, et qu'il ne dispose pas d'afficheur (**figure 3**). Notez que l'un des deux relais doit être actif à tout moment. C'est le logiciel qui s'en charge.

Sécurité

Pour améliorer la sécurité et éviter que le système ne soit perturbé par un signal parasite, nous avons choisi les modules émetteurs-récepteurs LoRa à faible consommation d'énergie d'Ebyte. En plus d'utiliser une technologie à étalement de spectre résistante aux interférences, ils fournissent trois paramètres : la fréquence du canal, le débit d'air et un identifiant de 4 octets. La communication ne peut avoir lieu que si ces trois paramètres sont identiques aux deux extrémités de la liaison.

Figure 2. L'émetteur EPB se compose d'un écran OLED et de deux boutons poussoirs (en plus de l'émetteur-récepteur LoRa).



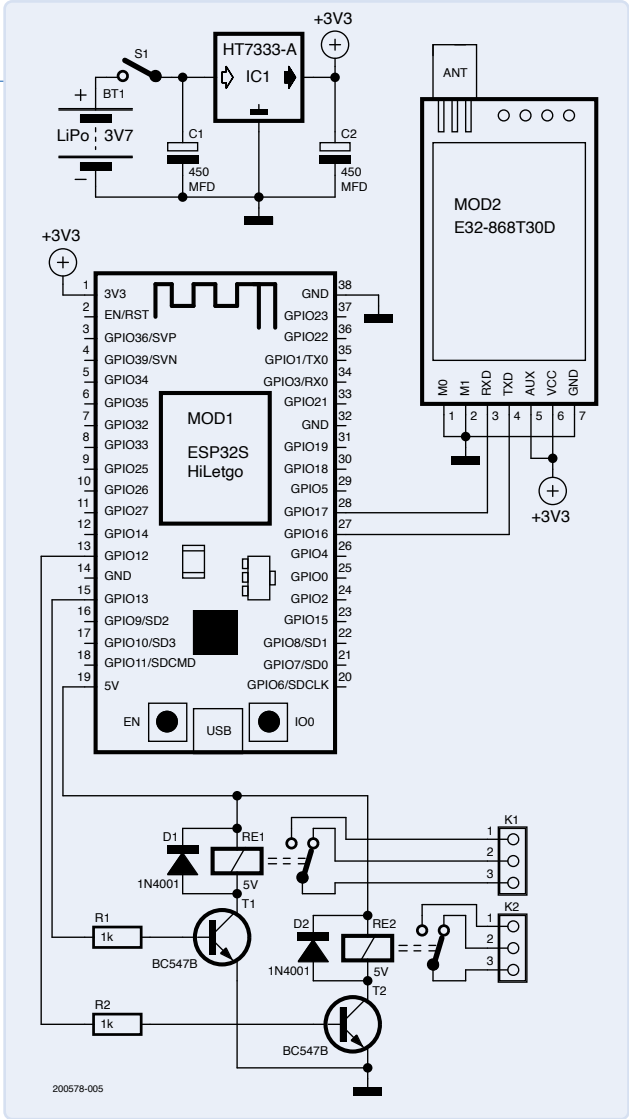


Figure 3. Les deux relais du récepteur EPB appuient sur les boutons du système EPB existant.

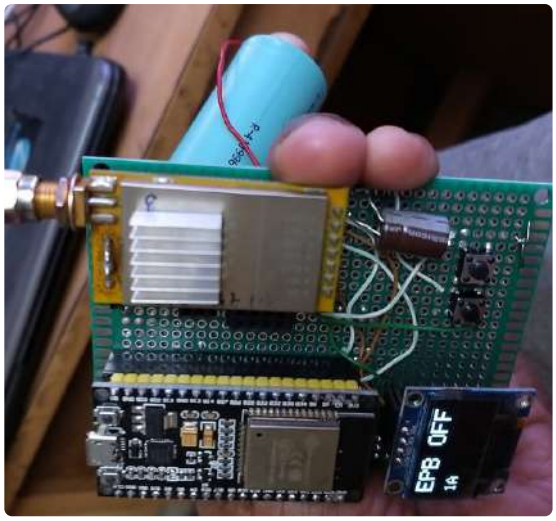


Figure 4. Prototype de l'émetteur EPB construit sur perfboard.

Le module comporte une interface série et délivre jusqu'à 500 mW (21…30 dbm) à 868 MHz (**figure 4**). Le signal peut facilement traverser 500 mètres le long de la voie ferrée avec une petite antenne portative et avec l'antenne de l'unité de réception placée sur le toit de la cabine/salle de contrôle, de sorte qu'il reste visible partout.

Logiciel

Tout comme le matériel des deux unités, les logiciels sont également similaires et se composent de deux courts croquis Arduino.

Les deux unités configurent le port série utilisé pour communiquer avec l'émetteur-récepteur LoRa (9600N81) dans la fonction `setup()`. Pour l'émetteur EPB deux broches E/S sont configurées comme entrées du bouton-poussoir, tandis que pour le récepteur EPB ces broches sont configurées en sorties pour commander des relais. L'émetteur EPB configure également l'afficheur OLED.

Dans la fonction `loop()`, l'émetteur et le récepteur vérifient d'abord si un message a été reçu de l'autre unité. Si c'est le cas, ils mettent à jour leur état. Le récepteur envoie un accusé de réception à l'émetteur et attend un nouveau message.

L'émetteur continue en vérifiant l'état de ses deux boutons-poussoirs. Selon le bouton appuyé, il envoie une courte chaîne ASCII à l'émetteur-récepteur LoRa via le port série. On ne peut appuyer que sur un seul bouton à la fois.

Le logiciel est disponible en téléchargement sur la page de ce projet sur Elektor Labs [1]. N'hésitez pas à le modifier selon vos besoins. Nous avons choisi arbitrairement les chaînes de caractères échangées entre les deux unités et vous pouvez les modifier.

200578-04

Des questions, des commentaires ?

Contactez Elektor (redaction@elektor.fr).

Produits

- **ESP32 DevKitC (SKU 18701)**
  https://elektor.fr/18701
- **Afficheur OLED I2C 0,96 pouce 128×64 (SKU 18747)**
  https://elektor.fr/18747
- **Claus Kühnel, *Develop and Operate Your LoRaWAN IoT Nodes* (SKU 20147)**
  https://elektor.fr/20147

LIENS

[1] Fichiers du projets sur Elektor Labs : https://elektormagazine.fr/labs/wagon-top-coal-sampling-remote-epb-module

démarrer en électronique

...l'émetteur-suiveur

Eric Bogers (Elektor)

Dans le dernier épisode, nous avons brièvement abordé les transistors comme amplificateurs, plutôt que comme commutateurs. Nous nous intéressons maintenant à ce que l'on appelle l'émetteur-suiveur, qui est le circuit à transistors le plus simple que l'on puisse imaginer.

Montage collecteur commun

Le premier montage amplificateur que nous allons étudier est connu sous le nom de montage à collecteur commun. On l'appelle ainsi, parce que le collecteur du transistor est le point de référence commun pour les signaux d'entrée et de sortie. Cette désignation n'étant pas très claire, ce type de montage est souvent appelé émetteur-suiveur – ce qui décrit bien son fonctionnement : la tension de l'émetteur du transistor suit la tension de la base. Nous découvrons l'utilité de ce montage dans cet article.

Supposons que vous vouliez construire une table de mixage pour lumière – rien de trop compliqué, juste une solution *maison* fonctionnant à basse tension. Vous pourriez avoir un certain nombre de lampes (canaux) avec une luminosité réglable individuellement (de 0 V à 10 V dans notre exemple), ainsi qu'une commande principale supplémentaire pour que toutes les lampes puissent être atténuées simultanément. En tant que débutant, vous pourriez avoir l'idée d'utiliser un potentiomètre principal et de connecter des potentiomètres de sortie à son curseur pour chaque lampe. Cette méthode est illustrée dans la partie supérieure de la **figure 1**.

Cependant, ce n'est pas la meilleure méthode. Ensemble, les potentiomètres de sortie forment une charge importante et déformeront la courbe de contrôle du potentiomètre principal, et, pire encore, tous ces potentiomètres s'influenceront, ce qui empêchera d'obtenir un contrôle efficace.

Vous devez réduire autant que possible la charge sur le potentiomètre maître et de découpler les potentiomètres individuels des canaux. Cela est possible avec des émetteurs-suiveurs, comme montré dans la partie inférieure de la **figure 1**. Le fonctionnement est comme suit :

lorsqu'un transistor est conducteur, la tension sur son émetteur est inférieure d'environ 0,7 V à la tension sur sa base, de sorte que

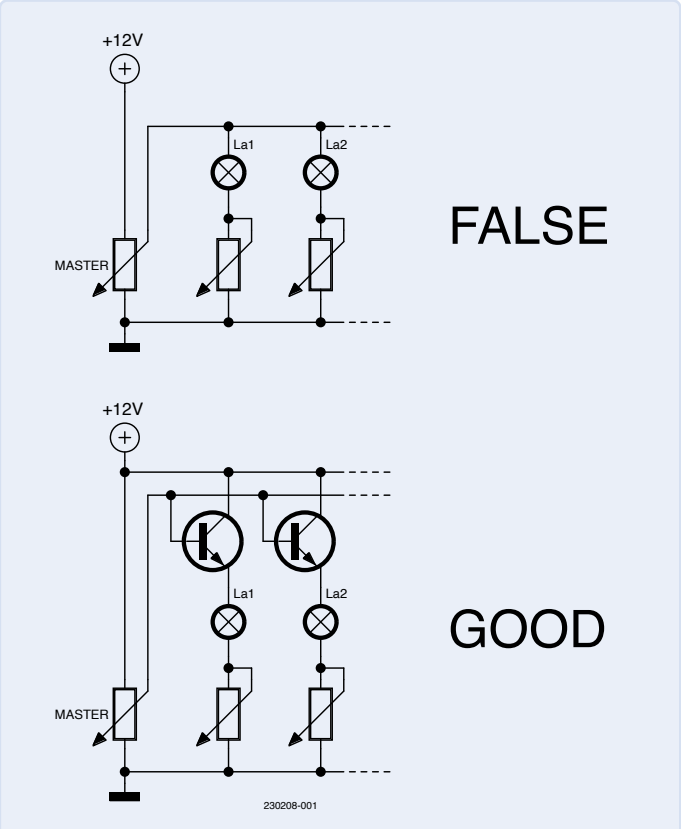


Figure 1. Une table de mixage pour lumière simplifiée : le mauvais montage en haut, le bon montage en bas.

la plage de contrôle va de 0 V à 11,3 V. Cela dépasse un peu les 10 V nominaux de cet exemple particulier, mais assure que lorsque les signaux de commande sont réglés au maximum, la luminosité des lampes est maximale. Lorsque la tension sur la base est inférieure à 0,7 V, le transistor est bloqué et la tension de sortie est de 0 V.
Entre ces deux limites, la tension sur l'émetteur suit bien la tension sur la base, mais avec un *offset* égal à la valeur de la tension base-émetteur. Le principal avantage de l'utilisation d'un émetteur-suiveur dans cette situation est que le courant de base n'impose qu'une très faible charge au curseur du potentiomètre maître, puisqu'il est beaucoup plus faible que le courant d'émetteur. Le rapport est égal au gain en courant continu (h_{FE}) du transistor. Dans la pratique, cela signifie que la charge sur le potentiomètre maître est inférieure d'un facteur h_{FE} au courant réglé par le potentiomètre de canal.
Ainsi, le montage à collecteur commun (émetteur-suiveur) n'offre pas de gain en tension, mais un gain en courant significatif, une impédance d'entrée élevée et une faible impédance de sortie. Un circuit émetteur-suiveur a une réponse très stable aux variations de charge. En général, on s'attend à ce que la tension de sortie diminue lorsque la résistance de la charge diminue, mais lorsque cela se produit, la tension base-émetteur U_{BE} augmente, de sorte que le courant de base I_B augmente également. Cela entraîne une augmentation du courant du collecteur I_C et, et par conséquent, une augmentation du courant d'émetteur I_E, ce qui compense la diminution de la tension de sortie.
Les formules suivantes s'appliquent à l'impédance d'entrée (en haut) et de sortie (en bas) d'un émetteur-suiveur :

$$Z_{in} = (h_{FE} + 1) \cdot Z_{load}$$

$$Z_{out} = \frac{Z_{source}}{h_{FE} + 1}$$

Alimentation stabilisée

Dans un épisode précédent, nous avons fait quelques calculs pour montrer que l'utilisation d'une diode Zener et d'une résistance en série pour stabiliser une tension présente des inconvénients importants. Ce type de circuit de stabilisation fonctionne mieux si vous y ajoutez un émetteur-suiveur, comme le montre la **figure 2**.
Faisons les calculs sur ce circuit avec les valeurs de l'exemple précédent (une tension de la diode Zener de 48 V et un courant de charge maximal de 14,1 mA). Il est possible d'utiliser ce circuit comme alimentation fantôme pour un microphone. Le courant de base est alors donné par :

$$I_B \approx \frac{I_E}{\beta} = \frac{14.1\text{mA}}{100} = 0.14\text{mA}$$

Le gain en courant continu s'applique au courant du collecteur. Stricto sensu, le courant d'émetteur est égal à I_C plus I_B – mais dans le cas d'un fonctionnement à petits signaux avec un gain de courant

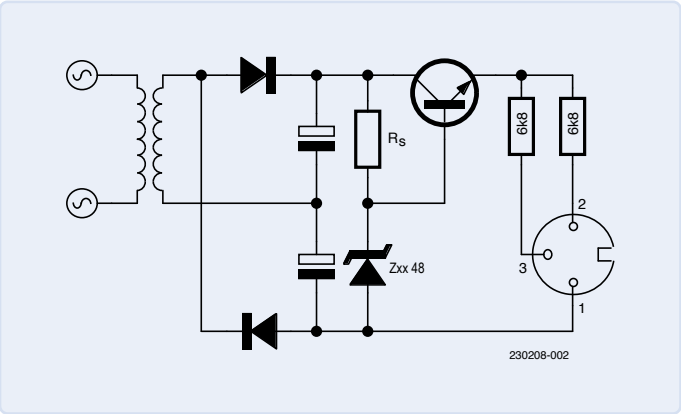


Figure 2. Alimentation stabilisée (alimentation fantôme pour un microphone).

supérieur à 100, nous pouvons considérer que I_C est presque égal à I_E. Nous ignorons la valeur exacte du gain en courant du transistor. Supposer un facteur de gain de 100 pour les transistors à petits signaux nous permet de rester sur le côté sécuritaire.
Pendant un cycle de la tension alternative d'entrée, la tension au niveau des condensateurs électrolytiques de filtrage s'établit à environ 60 V, ce qui signifie que la tension au niveau de la résistance en série de la diode Zener est de 12 V. Nous avons donc :

$$R_{series} = \frac{U}{I_B} = \frac{12\text{V}}{0.14\text{mA}} = 85\text{k}\Omega$$

Ici, nous utilisons la valeur standard 82 kΩ pour la résistance en série.
La dissipation maximale dans la diode Zener est obtenue lorsqu'aucune charge n'est connectée, de sorte que le courant de base du transistor est nul. Dans ces conditions :

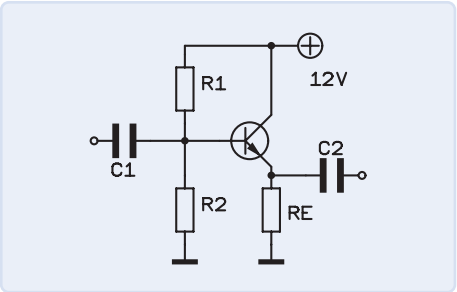
$$P_{zener} = U \cdot I = 48\text{V} \cdot \frac{63\text{V} - 48\text{V}}{82\text{k}\Omega} = 8.78\text{mW}$$

Bien entendu, la puissance est également dissipée dans le transistor :

$$P_{transistor} = U \cdot I = (63\text{V} - 48\text{V}) \cdot 14.1\text{mA} = 211.5\text{mW}$$

La somme de la dissipation dans la diode Zener et de celle dans le transistor est déjà moins élevée qu'en utilisant uniquement la diode Zener, puisque le courant sans le transistor était un peu supérieur à 900 mA. Le circuit comportant le transistor a un autre avantage : sans le transistor, la dissipation maximale est atteinte lorsqu'aucune charge n'est connectée et la dissipation minimale est atteinte lorsque la charge est 0 Ω, ce qui, bien sûr, n'arrive jamais dans la pratique. En revanche, dans le circuit à transistor, la dissipation

de puissance maximale se produit avec une résistance de charge de 0 Ω, alors que la dissipation est plus faible dans les conditions normales du fonctionnement.



Figuur 3. Amplificateur CA.

Montage émetteur-suiveur comme amplificateur CA

Jusqu'à cette étape, nous n'avons utilisé l'émetteur-suiveur qu'en tant qu'amplificateur de courant continu. Passons maintenant à l'amplification des signaux alternatifs (voir **figure 3**).
Lorsqu'une tension alternative est appliquée au condensateur C1, elle modifie la tension sur la base (par référence à la masse), ce qui fait varier le courant de l'émetteur de telle sorte que la tension sur l'émetteur est toujours inférieure d'environ 0,7 V à la tension sur la base. Là encore, le circuit n'offre pas de gain en tension, mais offre un gain en courant.
Pour obtenir une amplitude maximale avec ce circuit, la tension de repos de l'émetteur doit être approximativement égale à la moitié de la tension d'alimentation, soit, dans cet exemple, 6 V. Par souci de clarté : la tension de repos et le courant de repos sont des valeurs mesurées lorsqu'aucune tension de signal (CA) n'est appliquée.
Si nous supposons que le courant de repos est de 1 mA, la valeur correspondante de la résistance d'émetteur est :

$$R_E = \frac{U_R}{I_R} = \frac{6\,V}{1mA} = 6k\Omega$$

Cette valeur (6 kΩ) est comprise entre deux valeurs de E12, nous choisissons 5,6 kΩ. La tension de base du transistor est supérieure de 0,7 V à la tension d'émetteur, elle est donc de 6,7 V par rapport à la masse. Cette tension est obtenue par un diviseur de tension. Nous choisissons arbitrairement la valeur de R2 (100 kΩ) ce qui nous permet de calculer la valeur de R1. Nous laissons ce calcul comme exercice pour vous ; nous avons obtenu une valeur de 79,1 kΩ et avons choisi une résistance de 82 kΩ. En conséquence, les tensions de base et d'émetteur sont inférieures d'environ 0,1 V aux valeurs prévues, mais ce n'est pas un problème.
Nous devons proportionner les valeurs des deux capacités. Nous voulons que le circuit ait une réponse en fréquence aussi linéaire que possible dans la plage de fréquences allant de 20 Hz à 20 kHz, nous fixons donc la fréquence de la partie inférieure dans les deux cas à 10 Hz. Le condensateur C1, les résistances R1 et R2 en parallèle, et l'impédance d'entrée du transistor, forment un filtre passe-haut et, comme nous le savons déjà, l'impédance d'entrée du transistor dépend de la résistance de la charge. Si nous supposons que la résistance de charge ne sera pas inférieure à R_E, la résistance de charge et de R_E en parallèle s'élève à 3 kΩ. En supposant que le gain en courant est de 100, l'impédance d'entrée (R_{in}) du transistor est d'environ 300 kΩ. La résistance parallèle de R1, R2 et R_{in} s'élève donc à 39,2 kΩ (vérifiez le résultat vous-même !).
Nous utilisons la formule suivante pour déterminer la valeur de C1 :

$$C_1 = \frac{1}{2 \cdot \pi \cdot f \cdot Z} = \frac{1}{2 \cdot 3.14 \cdot 10Hz \cdot 39.2k\Omega} = 0.4\,\mu F$$

Avec une valeur de 0,47 µF, nous sommes en tout cas du bon côté. Pour C2, nous faisons le calcul en supposant une impédance de charge minimale de 6 kΩ :

$$C_2 = \frac{1}{2 \cdot \pi \cdot f \cdot Z} = \frac{1}{2 \cdot 3.14 \cdot 10Hz \cdot 6k\Omega} = 2.6\,\mu F$$

Dans ce cas, on choisit une valeur de 3,3 µF (ou 4,7 µF si c'est plus facile à se procurer).
Une remarque pour nos lecteurs attentifs : dans le calcul de R1, nous n'avons pas tenu compte du fait que l'impédance en courant continu du transistor (qui est égale au produit du gain en courant par R_E) est en fait en parallèle avec R2.
Si nous voulons le prendre en compte, nous devons connaître le gain en courant du transistor avec une précision suffisante. Si par précaution nous supposons que la valeur est 100, le point de fonctionnement sera fortement décalé ; une valeur de 300 à 500 est plus raisonnable pour les transistors à petit signal. Cependant, dans ce cas, nous pouvons ignorer l'influence du gain en courant sur le calcul de R1...
Et voilà, c'est terminé pour cet épisode. Le prochain article sera vraiment passionnant, car nous étudierons l'amplification de la tension avec des montages à émetteur commun (ce qui est tout à fait différent des émetteurs-suiveurs).

230208-04

Note de la redaction : la série d'articles « démarrer en électronique » est basée sur le livre « Basiskurs Elektronik » de Michael Ebner, publié par Elektor.

Des questions, des commentaires?

Contactez Elektor (redaction@elektor.fr).

Produits

- **B. Kainka, *Basic Electronics for Beginners* (Elektor, 2020) (SKU 19212)**
  www.elektor.fr/19212
- **B. Kainka, *Basic Electronics for Beginners* (Elektor, 2020) (E-Book, SKU 19213)**
  www.elektor.fr/19213

comparateur à hysteresis à niveaux indépendants

simulations, feuilles de calcul et algèbre

Stephen Bernhoeft (Royaume-Uni)

Souvent, on a besoin de niveaux d'hystérésis à la fois précis et indépendants. Pour les comparateurs ordinaires, c'est tout à fait possible. Voici comment.

La sélection des valeurs de résistance nécessaires pour atteindre précisément les niveaux d'hystérésis pour les montages comparateurs n'est pas couramment abordée dans la documentation disponible, sauf dans le cas où les seuils de basculement sont équidistants des rails d'alimentation. Dans cet article, nous utilisons quelques notions d'algèbre pour dériver des formules pour les circuits à « entrée -ve « et à « entrée +ve « couramment utilisés. Les feuilles de calcul, les dérivations et les schémas *LTSpice* correspondants sont disponible sur [1]. Les résultats présentés s'appliquent à :

- montages à sortie *rail-to-rail*
- montages à collecteur ouvert où $R_{PULLUP} << R_{FEEDBACK}$

Pour réaliser un montage à collecteur ouvert, nous devons d'abord choisir R_{PULLUP}, puis choisir $R_{FEEDBACK}$ (noté « *F* » dans la suite de cet article) au moins 10 × R_{PULLUP}. Les autres résistances sont calculées en fonction de *F*.

Paramètres

Pour réaliser le circuit, il faut choisir les seuils de basculement requis et noter les niveaux de sortie. Les valeurs sont ratiométriques et vous pouvez les adapter, mais pour les composants à collecteur ouvert, vous devez tenir compte de la valeur de la résistance de tirage.

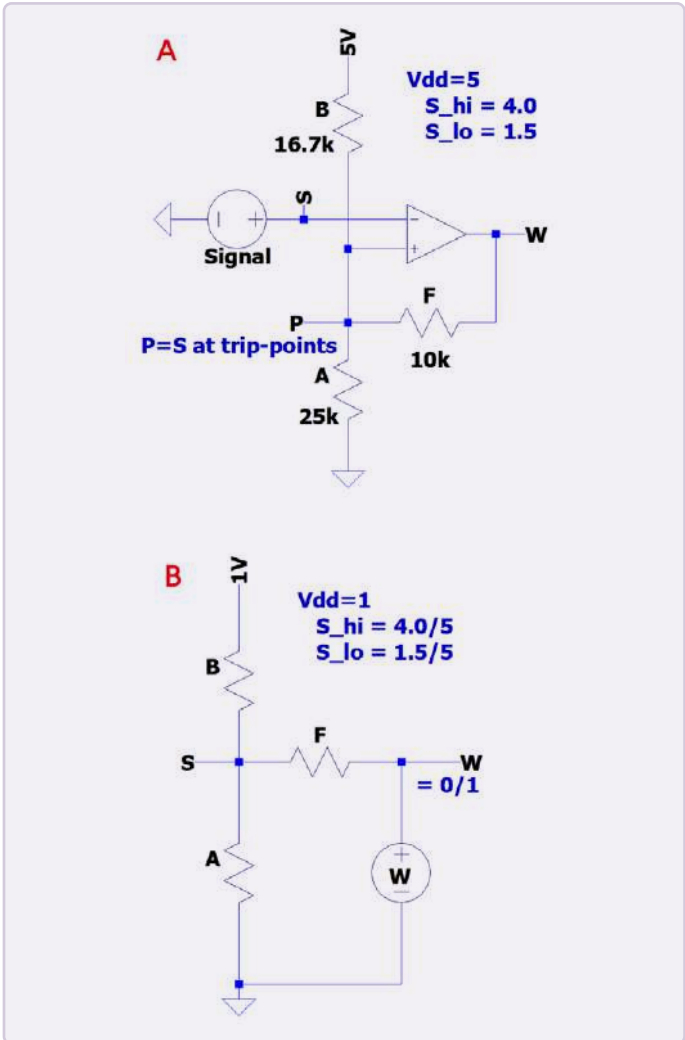


Figure 1. En haut, le circuit réel (A) ; en bas, une version simplifiée pour l'analyse (B).

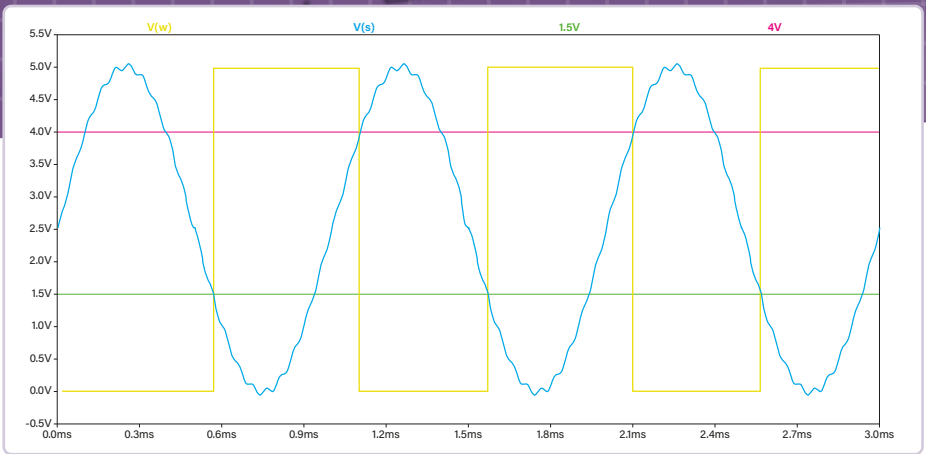


Figure 2. Simulation des signaux du circuit (A) de la figure 1.

Montage inverseur

La **figure 1** montre le premier exemple. Ce circuit se caractérise par une impédance d'entrée élevée et peu de composants et il ne nécessite pas une référence séparée. L'inconvénient est le caractère inverseur, qui peut être indésirable, en particulier lorsqu'on essaie de réduire le nombre de composants.

Dans le montage simplifié de la **figure 1B**, *F* est effectivement en parallèle avec *B* (si *W* = 1) ou *A* (lorsque *W* = 0). Pour simplifier les calculs, la tension d'alimentation est de 1 V. Vous pouvez modifier cette valeur. Par exemple, une alimentation de 5 V avec un niveau de basculement de 4 V est équivalente à une alimentation de 1 V avec un niveau de basculement de 4/5 V, soit 0,8 V.

W change d'état lorsque *P* = *S*. Supposons que *P* = *S* à l'un des deux niveaux de basculement. Dans la **figure 1A**, le nœud *P* change en fonction de la sortie du comparateur. Nous pouvons renommer *P* en *S* dans le schéma de la **figure 1B** car ils sont égaux.

Si nous définissons :

- S_{LO} = seuil de basculement inférieur
- S_{HI} = seuil de basculement supérieur

et nous avons :

$$k_{BF} = \frac{S_{HI} - S_{LO}}{S_{LO}} = \frac{B}{F}; \qquad k_{AB} = \frac{S_{LO}}{1 - S_{HI}} = \frac{A}{B}$$

alors :

1er Case : *A* fixe

$$B = \frac{A}{k_{AB}}; \qquad F = \frac{B}{k_{BF}}$$

2e cas : *B* fixe

$$A = B \cdot k_{AB}; \qquad F = \frac{B}{k_{BF}}$$

3e cas : *F* fixe

$$B = F \cdot k_{BF}; \qquad A = B \cdot k_{AB}$$

Nous avons maintenant établi les rapports *A* : *B* et *B* : *F* en fonction des niveaux de basculement.. Le choix d'une résistance (*A*, *B* ou *F*) permet de calculer les deux autres valeurs. La **figure 2** montre la simulation des signaux du circuit (*A*).

Considérations

Nous devons choisir les niveaux de basculement. en tenant compte de plusieurs facteurs :

- Pour le *timing* des circuits RC : évitez les extrémités de la « partie lente » de la courbe de charge du circuit RC, car de petites variations de tension entraînent d'importantes variations de temps.
- Les seuils de basculement doivent être indépendants des tensions de décalage du comparateur. Par conséquent, le seuil de basculementt inférieur doit être beaucoup plus supérieur à la valeur (absolue) de la tension de décalage.

En examinant la feuille de calcul [1], nous constatons que si nous choisissons une valeur très faible pour S_{LO}, nous obtenons plusieurs valeurs possibles de résistance requises pour *A*, *B* et *F*. Ce n'est évidemment pas l'idéal en termes de précision relative des valeurs. Nous obtenons la meilleure précision relative (ratio) lorsque les valeurs de *A*, *B* et *F* sont proches.

Applications

Pour donner un exemple réel, l'auteur a travaillé sur un système de détection de fumée. Nous avons d'abord creé un circuit analogique basé sur un capteur à métal-oxyde (nous doutons de nos compétences en programmation pour une telle application).

Supposons que nous avons besoin de 3 secondes de détection de fumée avant que la sortie d'alarme ne se déclenche, mais de 6 secondes sans fumée avant que l'alarme ne retentisse. La **figure 3** montre un circuit réalisable, la **figure 4** montre une simulation du signal de sortie.

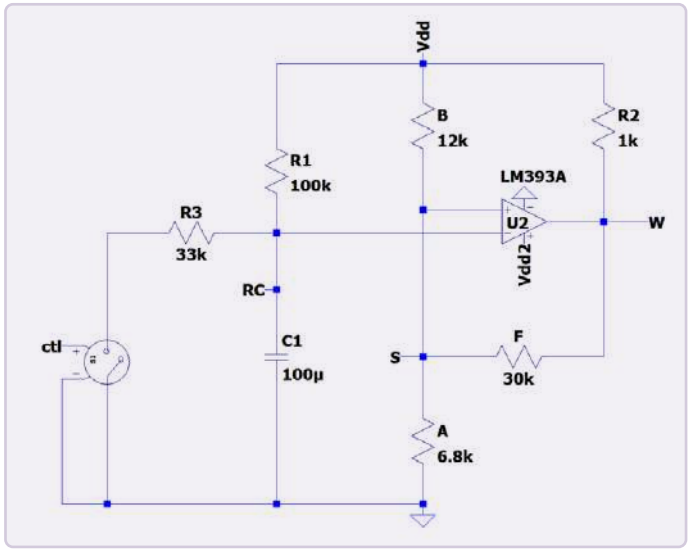


Figure 3. L'étage d'entrée d'un circuit de détection de fumée. Lorsque l'interrupteur est fermé, R_{EQ} = 25 kΩ, V_{EQ} = 1.25 V. Lorsque l'interrupteur est ouvert, R_{EQ} = 100 kΩ, V_{EQ} = 5 V. ΔV = (5-1.25) = 3.75 V.

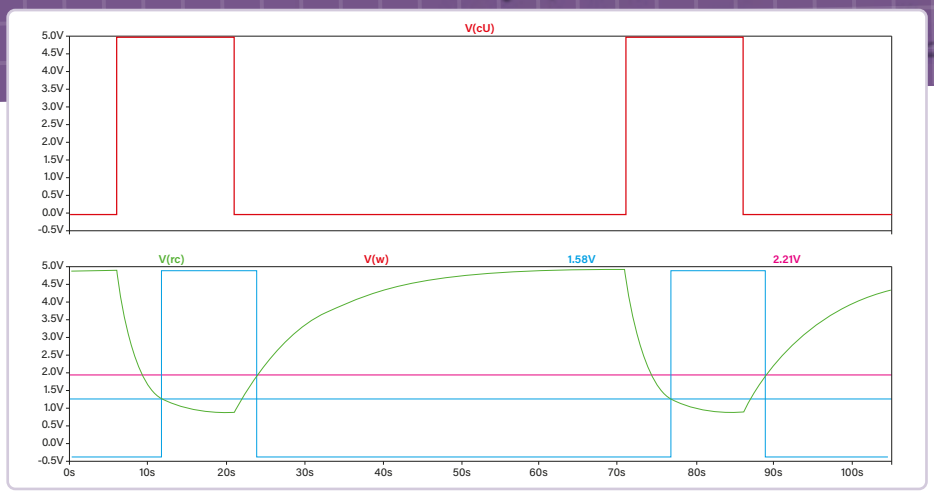


Figure 4. Simulation de la sortie du circuit de la figure 3.

Si aucune fumée n'est détectée, l'interrupteur est fermé et le condensateur se charge à un niveau de 5/4 V à travers une résistance efficace REQ = R1|R3. Si de la fumée est détectée, l'interrupteur s2 s'ouvre par le signal *ctl*, chargeant le circuit RC de 1,25 V à VDD = 5 V.
Les niveaux de basculement requis sont indiqués dans une feuille de calcul, tandis qu'une autre feuille de calcul montre les conversions de ces tensions en valeurs de résistance pour *A*, *B* et *F*.

Montage non-inverseur

Examinons maintenant le circuit à entrée positive (ou non inverseur), comme le montre la **figure 5.** Un tel circuit présente une impédance d'entrée de *A*+*F* et est un peu plus difficile à analyser.

Alors :

$$\Delta S = S_{HI} - S_{LO}\ ; \qquad \Delta W = W_{HI} - W_{LO}$$
$$\Sigma S = S_{HI} + S_{LO}\ ; \qquad \Sigma W = W_{HI} + W_{LO}$$

Alors :

$$\frac{A}{F} = \frac{\Delta S}{\Delta W}$$

$$M = \frac{A \cdot \Sigma W + F \cdot \Sigma S}{2 \cdot (A + F)}$$

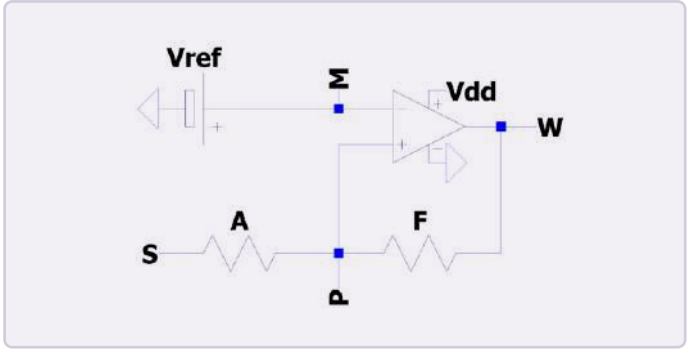


Figure 5. Circuit non inverseur.

Il est possible d'exprimer *M* indépendamment de *A* ou de *F* :

$$M = \left(\frac{1}{2}\right) \cdot \left(\frac{1}{\Delta W + \Delta S}\right) \cdot \left(\Delta S \cdot \Sigma W + \Delta W \cdot \Sigma S\right)$$

Exemple

Supposons que le seuil inférieur S_{LO} = 0.5, le seuil supérieur S_{HI} = 3.0, alors que les niveaux de sortie sont
W_{LO} = 0 et W_{HI} = 5. Alors, A/F = (3-0.5)/(5-0) = 0.5 et
M = 0.5 × 1/(5+2.5) × (2.5 × 5 + 5 × 3.5) = 2 V.

La figure 6 montre les résultats.

Hystérésis du courant d'entrée

Si nous connectons la résistance *A* à la masse, le circuit de la **figure 3** se transforme en un comparateur dont le signal d'entrée est un courant, comme le montre les **figures 7** et **8**. La valeur de la résistance *F* est obtenue directement à partir de *ΔW* et *ΔI*. Ensuite, nous pouvons déterminer le rapport du diviseur *β*. Enfin, nous calculons la valeur de A. Une feuille de calcul est utile ici car *β* et *A* doivent être aussi précis que possible en utilisant des valeurs de résistance standard.

$$\beta = \frac{A}{A+F}\ ; \qquad R_P = \frac{A \cdot F}{A+F}$$

$$P = \beta \cdot W_H + I_L \cdot R_P = \beta \cdot W_L + I_H \cdot R_P$$

$$\beta \cdot \Delta W = R_P \cdot \Delta I\ ; \qquad \frac{R_P}{\beta} = F = \frac{\Delta W}{\Delta I}$$

$$W_L = \frac{P - I_H \cdot R_P}{\beta} = \frac{P - I_H \cdot F \cdot \beta}{\beta} = \frac{P}{\beta} - I_H \cdot F$$

$$\frac{P}{\beta} = W_L + I_H \cdot F \quad \rightarrow \quad \beta = \frac{P}{W_L + I_H \cdot F}$$

$$A = \frac{F \cdot \beta}{1 - \beta}$$

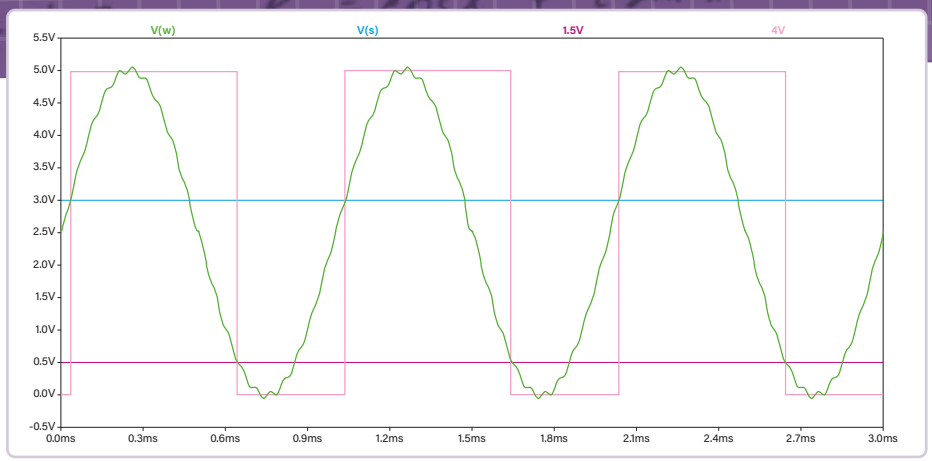


Figure 6. Les signaux simulés du circuit de la figure 5

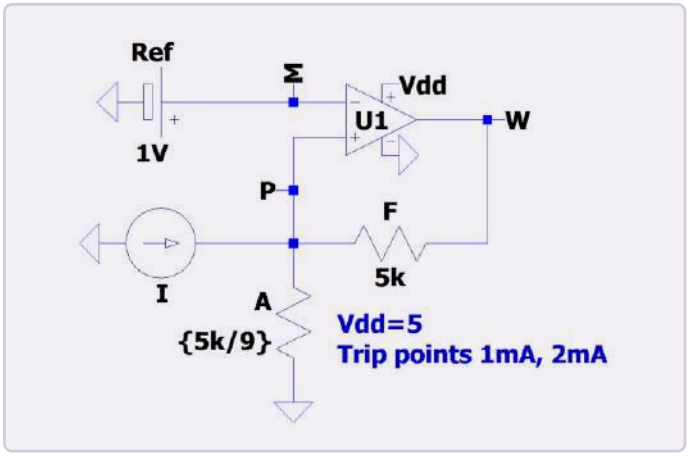


Figure 7. Le circuit de la figure 3, adapté à une entrée de courant.

Simulations, feuilles de calcul et algèbre

Les simulations, les feuilles de calcul et les calculs complets des formules présentées dans cet article sont disponibles en téléchargement sur la page web de cet article [1].

200559-04

Des questions, des commentaires ?

Contactez Elektor (redaction@elektor.fr).

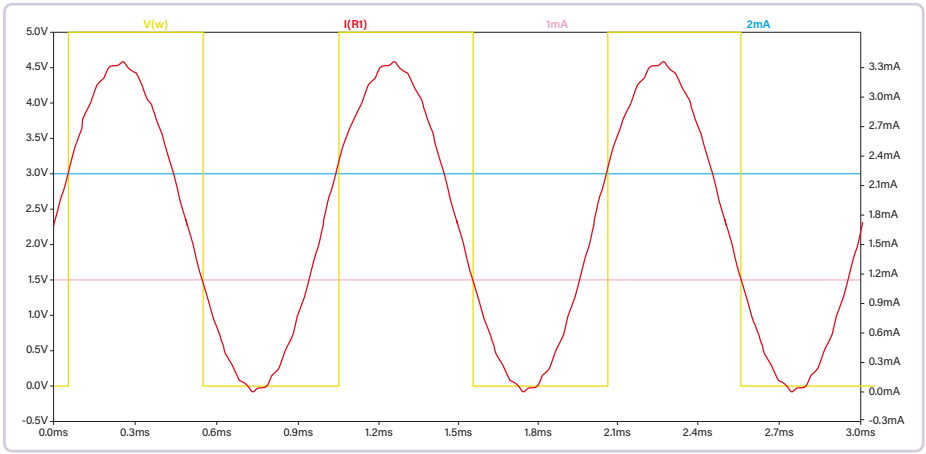


Figure 8. La sortie du simulateur pour le circuit de la figure 7.

Produits

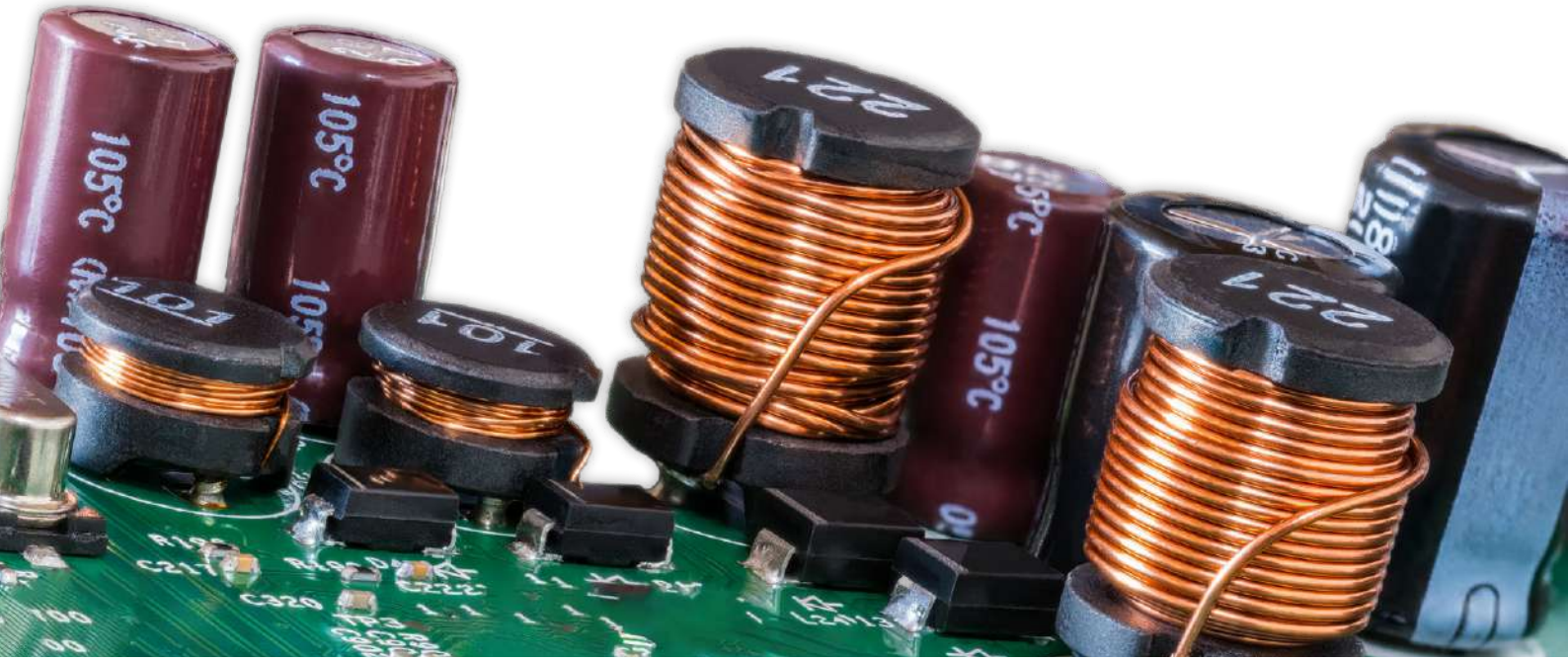
- **Paul Horowitz, Winfield Hill, *The Art of Electronics (3rd Edition)* (SKU 17167)**
  https://elektor.fr/17167
- **Gilles Brocard, *The LTspice XVII Simulator* (SKU 19741)**
  https://elektor.fr/19741

LIEN

[1] Téléchargements pour cet article : https://elektormagazine.fr/200559-04

analyseur d'impédance **basé sur un ESP32**

simple, comportant peu de composants et de faible coût !



Volker Ziemann (Suède)

Un ESP32 analyseur d'impédance, pouvant également vérifier la fréquence de résonance d'un réseau LC ? Oui, ce microcontrôleur peu couteux, associé à une poignée de composants externes peut le faire, tout en offrant une interface basée sur le Web.

Commençons par un peu de théorie. L'impédance d'un composant électronique $Z = U/I$ est le rapport de la tension U aux bornes du composant par l'intensité I qui le traverse. La loi d'Ohm écrite sous la forme $R = U/I$ est un exemple particulier où la résistance R est la forme la plus simple d'une impédance. Pour un condensateur de valeur C, l'impédance dépend de la fréquence f dont la pulsation est $\omega = 2\pi f$, est donnée par $-i/\omega C$; l'unité imaginaire i est une forme compacte indiquant que la phase de la tension est décalée de 90° par rapport au courant. De même, l'impédance d'une inductance L est donnée par $i\omega L$. Ici encore, l'unité imaginaire i indique que la tension précède le courant. Nous pouvons donc déterminer l'impédance d'un composant en examinant la dépendance en fréquence de celle-ci et la relation de phase entre la tension et le courant qui le traverse.

Ces mesures deviennent intéressantes dans le cas de plusieurs composants interconnectés. Si une inductance et un condensateur sont reliés en parallèle, le circuit à l'impédance la plus élevée à la fréquence de résonance pour laquelle les résistances des deux composants en courant alternatif sont égales. Pour une connexion en série, l'impédance est la plus faible à ce point. Avec l'analyseur d'impédance basé sur un ESP32 d'Espressif, on peut enregistrer la courbe de l'impédance

en fonction de la fréquence d'un réseau RLC et ainsi déterminer ses caractéristiques.
Pour cela il est nécessaire de disposer d'un moyen de produire un signal sinusoïdal d'amplitude constante, de faire varier sa fréquence dans une plage préférablement importante, puis de mesurer rapidement le courant qui traverse le(s) composant(s) que l'on qualifie de DUT (*Device Under Test*). J'ai remarqué que le microcontrôleur ESP32 que j'avais reçu de la part d'Elektor pour ma participation au concours de design de 2018 pouvait réaliser la plupart de ces opérations. Il possède un générateur de fréquence intégré et un convertisseur Numérique Analogique (CNA ou DAC (*Digital Analog Converter*) qui génère des tensions de sortie dont la fréquence peut atteindre plusieurs centaines de kilohertz. De plus, il intègre un Convertisseur Analogique Numérique (CAN ou ADC *Analog Digital Converter*) qui peut lire des tensions à une rapidité pouvant atteindre un million de fois par seconde, certes en trichant un peu ; mais nous y parviendrons. L'ESP32 ne pouvant lire qu'un seul canal CAN à haute vitesse et les convertisseurs CAN et CNA fonctionnant indépendamment, on ne pourra déterminer que l'amplitude de l'impédance avec un seul ESP32, mais pas la phase. La possibilité de déterminer l'écart de phase produit par le circuit en test nécessite un dispositif spécial, ce sera le sujet d'un article séparé.

Circuit analogique d'entrée

Nous devons nous assurer que l'amplitude de la tension du CNA est constante et ne dépend pas de l'impédance du circuit en test. C'est le rôle de l'amplificateur opérationnel de la **figure 1**. Il est utilisé comme tampon de gain 1x pour le signal issu de la broche 25 de l'ESP32. Sa faible impédance de sortie pilote le DUT, circuit en test. L'autre terminaison du circuit en test est reliée à l'entrée négative d'un second amplificateur opérationnel qui est configuré en amplificateur à transimpédance. Un courant fourni à son entrée négative est converti, grâce à la résistance de contre-réaction R_{fb} en une tension de sortie qui est appliquée à la broche 32 du CAN de l'ESP32. Ce CAN n'accepte qu'une tension positive. Ainsi, la composante continue du signal est portée à la moitié de la valeur de la tension d'alimentation. Grâce à ce dispositif, le circuit en test est alimenté par une tension alternative d'amplitude constante ; le courant est mesuré à l'aide du CAN.
La **figure 2** montre l'interface d'entrée réalisée sur une plaque d'essais avec une petite platine perforée munie d'un condensateur comme circuit en test (DUT) en bas et à gauche. La résistance R_{fb} à droite de l'amplificateur opérationnel est facile à changer. Les quatre fils reliant ce petit circuit à l'ESP32 sont, en noir la masse GND, en rouge, la tension d'alimentation 3,3 V, en vert la sortie du CNA (broche 25), et en bleu la broche 32 du CAN.

CAN rapide

Le mode rapide du CAN de l'ESP32 fait partie du sous-système I2S qui est normalement utilisé pour le traitement des signaux audio. En outre, l'ESP32 supporte un mode dans lequel les mots générés par le CAN sont copiés dans une mémoire tampon à accès direct (DMA) qui ne nécessite pas l'utilisation du processeur. Le CAN fonctionne de façon asynchrone (librement) et la mémoire DMA collecte les données à un rythme déterminé. Tant que ce rythme est inférieur à celui de la rapidité maximum de conversion du CAN, soit environ 250 kC/s, toutes

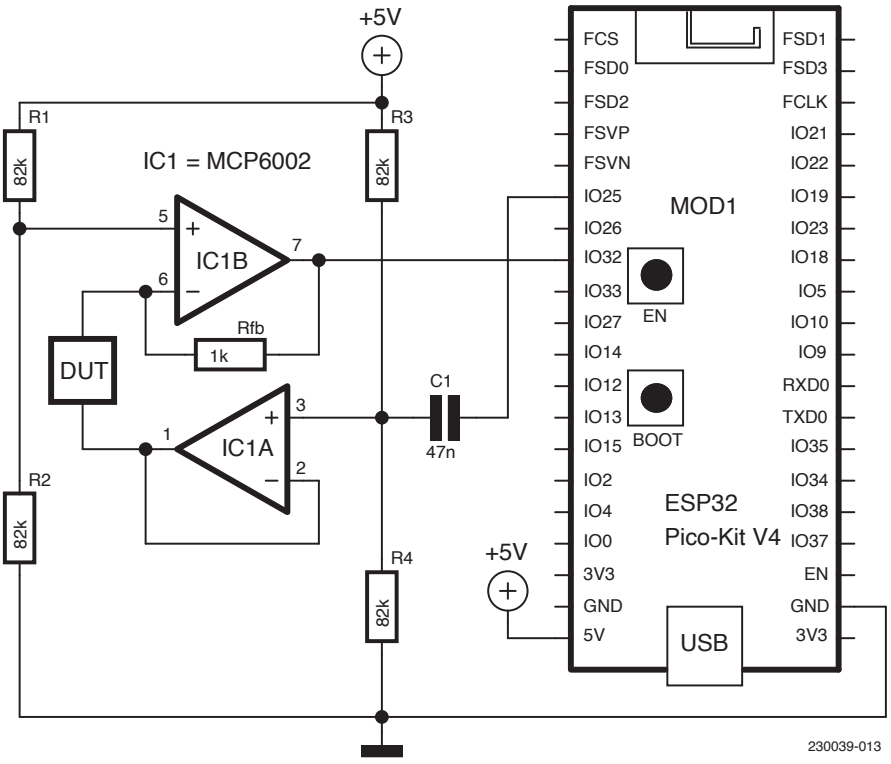


Figure 1. Interface analogique d'entrée.

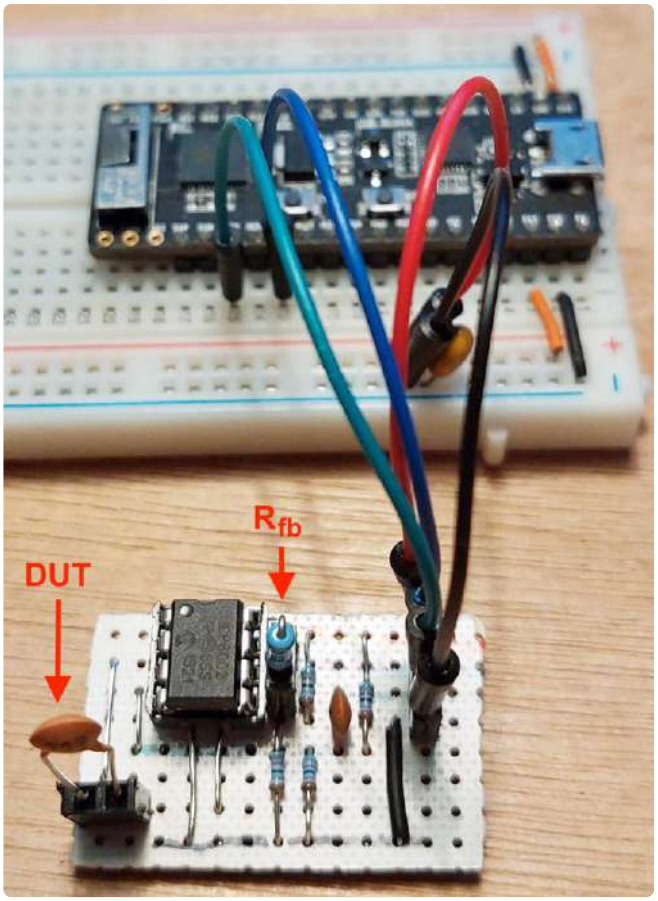


Figure 2. Seuls quelques composants externes sont nécessaires pour permettre à l'ESP32 de mesurer l'impédance d'un circuit en test (DUT).

les données sont « fraîches ». En revanche, si le taux est trop élevé, des échantillons identiques sont copiés dans la mémoire tampon, ce qui se traduit par des escaliers dans les données capturées.
La vitesse d'échantillonnage du CAN est paramétrée par la fonction `i2sinit()`, qui est dérivée du croquis *HiFreq_ADC.ino* présent dans la bibliothèque de support de l'ESP32 de l'EDI Arduino. La majeure partie de la configuration consiste à entrer dans la structure `i2s_config`, le mode d'opération, le taux d'échantillonnage et quelques paramètres indicateurs qui ont été trouvés après quelques recherches en [1]. Avant de quitter la fonction, on doit régler le niveau d'atténuation de façon à ce que l'amplitude du canal du CAN corresponde à la valeur de la tension d'alimentation de 3,3 V, choisir le canal du CAN et autoriser la sortie.

Génération de fréquence

L'ESP32 possède un bloc fonctionnel interne qui délivre les valeurs successives d'une fonction cosinus à un rythme qui peut être ajusté. La sortie de ce bloc peut être dirigée, en positionnant plusieurs bits de configuration, vers l'entrée du registre du CNA, ce qui produit une tension de sortie de forme sinusoïdale. La fréquence de ce générateur est directement contrôlée en paramétrant les registres `SENS_SAR_DAC_CTRL1_REG` et `SENS_SAR_DAC_CTRL2_REG`, qui sont décrits en [1] et plus en détail en [2]. En suivant les explications fournies en [2], la fonction `cwDACinit()` est paramétrée pour remplir ces registres avec les valeurs permettant de configurer la fréquence de sortie et l'amplitude. Après avoir inclus les fichiers d'en-tête qui permettent l'utilisation des noms mnémoniques, la fonction `SET_PERI_REG_MASK(reg,bits)` est utilisée pour définir les bits du registre `reg`. Par exemple, la première des commandes suivantes :

```
SET_PERI_REG_MASK(SENS_SAR_DAC_CTRL1_REG,
  SENS_SW_TONE_EN);
SET_PERI_REG_BITS(SENS_SAR_DAC_CTRL1_REG, SENS_SW_FSTEP,
  frequency_step, SENS_SW_FSTEP_S);
```

active le fonctionnement du générateur à haute fréquence interne. La seconde détermine la fréquence du signal de sortie en spécifiant l'entier `frequency_step`, dont la valeur est dérivée de la rapidité de l'horloge interne. La fréquence de sortie désirée est déterminée approximativement par l'équation figurant en [1].

Croquis ESP32 contrôlé par l'interface série

J'ai programmé l'ESP32 en utilisant la version 1.8.19 de l'EDI Arduino disponible en [3], et suivi les instructions d'installation disponibles. À la date d'écriture de cet article, la nouvelle version 2 de l'EDI ne supportait pas encore un supplément fonctionnel nécessaire par la suite. J'ai également dû installer les fonctions de support de l'ESP32 depuis [4] en suivant les instructions d'installation.
Les croquis suivants sont basés sur [5] où se trouve une discussion détaillée sur de multiples aspects. L'objectif est de contrôler la fréquence générée et l'acquisition des données à partir de l'interface série selon un protocole simple dans lequel une seule chaîne est échangée de façon bidirectionnelle entre l'ESP32 et un programme fonctionnant sur un ordinateur PC pouvant supporter une communication série ; cela peut être Python, Octave ou LabView. Ce protocole ressemble au protocole SCPI utilisé dans les oscilloscopes ou d'autres appareils de mesure, dans lequel un point d'interrogation final indique une commande qui attend une réponse de l'ESP32. Par exemple, l'envoi de `STATE?`, déclenche l'envoi de la réponse `STATE fmin fmax fstep`, dont les trois valeurs numériques indiquent la gamme et le pas de variation du balayage en fréquence.
Analysons maintenant les différentes parties du sketch. En premier, deux fichiers en-têtes sont inclus ; l'un pour les fonctions de support du CAN et du CNA que j'ai développées, l'autre pour l'accès au système de fichiers *SPIFFS* de la mémoire flash de l'ESP32. Ce dernier est utilisé pour mémoriser les données de calibration. Les valeurs par défaut de la gamme de fréquences balayées et quelques autres valeurs pertinentes sont également spécifiées. La fonction `setup()` du sketch initialise la communication série, fixe la fréquence de sortie et l'amplitude de sortie du CNA, et lit les données de calibration à partir des fichiers SPIFFS s'ils sont disponibles.
La fonction `loop()` vérifie si une demande de la part de l'ordinateur hôte est reçue par l'appel à `Serial.available()`, lit une ligne (terminées par le caractère retour à la ligne `\n`) et convertit ce qui est reçu dans la chaine de caractères `line`. Quelques vérifications sont alors faites, selon le contenu spécifique du début de la chaine. Par exemple, si `line` contient `FREQ 20000`, la valeur numérique est extraite par `atoi(&line[5])`. La conversion de la chaine commençant à la cinquième position de `line` en un entier, est effectuée par la fonction intégrée `atoi()`. Cette valeur est ensuite affectée à la variable `dac25freq` et initialise le générateur de fréquence par `cwDACinit()`. De façon identique, toutes les variables importantes sont accessibles depuis l'ordinateur hôte.
`SWEEP?` est la commande la plus intéressante. Après avoir reçu cette commande et initialisé les variables utilisées dans cette section, une boucle `for` incrémente la variable `f` de `fmin` à `fmax` avec un pas de

valeur `fstep`. À l'intérieur de la boucle, la valeur de la fréquence du CNA est tout d'abord définie, puis, après un court délai, les valeurs mesurées par le CAN sont récupérées à l'aide de `is2_read()`.

```
for (float f=freqmin; f<freqmax;f+=freqstep) {
 dac25freq=f;
 cwDACinit(dac25freq,dac25scale,0);
 delay(50);
 i2s_read(I2S_NUM_0, &buffer,
     sizeof(buffer), &bytes_read, 15);
...
}
```

La taille de son argument d'entrée `buffer`, définie au début du sketch, est utilisée pour déterminer le nombre d'échantillons, ici 1024. La fonction renvoie également la valeur `bytes_read` indiquant le nombre d'octets lus. Chaque échantillon contenant deux octets, nous devons diviser par deux lors du prélèvement des échantillons afin de déterminer les valeurs minimum et maximum. Rappelons-nous que la tension mesurée par le CAN est proportionnelle au courant qui traverse le circuit en test, nous utilisons donc les variations des valeurs de pointes pour déterminer l'amplitude du courant.

Calibration

En parcourant les échantillons, les sommes `q1` et `q2` sont également accumulées. Utilisées conjointement avec les variables `S0`, `S1` et `S2`, elles permettent de définir un segment de droite joignant les points des données, ces paramètres sont nécessaires à la calibration. Les détails de l'algorithme utilisé sont indiqués dans l'annexe B de [5]. La commande `SAVECALIB` sauvegarde ces paramètres dans la mémoire persistante SPIFFS. La commande `GETCALIB?` les récupère depuis le PC. Les constantes de calibration `calib_slope` et `calib_offset` sont nécessaires pour prendre en compte les variations de certains facteurs d'atténuation du système inconnus, par exemple, due au condensateur de couplage CA (transmission des tensions alternatives) de l'entrée de l'amplificateur opérationnel inférieur. Cette ambiguïté est résolue en calibrant le système à l'aide d'une résistance de valeur résistive connue, insérée comme DUT. Toutes les autres impédances sont alors déterminées en référence à cette résistance de calibration. Celle-ci devrait idéalement être insensible à la fréquence utilisée, mais de faibles variations systématiques sont prises en compte en utilisant la pente comme fonction de la fréquence, en plus de l'écart (offset). La valeur de la résistance de calibration devrait avoir sensiblement la même valeur que la résistance de contre-réaction de l'amplificateur à transimpédance afin de correspondre à la gamme opérationnelle du CAN.

Avant d'utiliser l'analyseur, le système de fichiers SPIFFS de l'ESP32 doit être initialisé en créant un répertoire vide nommé `data` contenu dans le répertoire où est stocké le sketch. Ensuite, le plugin de chargement du système de fichiers Arduino ESP32 doit être installé depuis [6] en suivant les instructions. Quand cela est fait, un nouveau choix *ESP32 sketch Data Upload* apparait dans le menu *Outils* de l'EDI Arduino. Après s'être assuré que le moniteur série est fermé, un clic sur ce menu va créer le système de fichiers de l'ESP32.

Contrôle par Octave ou Python

Le contrôle de l'acquisition de données depuis le PC nécessite l'ouverture d'un port série dont le nom diffère selon le système d'exploitation utilisé. Il est habituellement nommé `COMn` sous Windows, `/dev/tty.usbserial-n` avec un MAC ou `/dev/ttyUSBn` sous Linux. L'envoi d'une commande, par exemple pour définir la fréquence de départ du balayage, se fait simplement en envoyant `FMIN 10000` par le port série. De même, la lecture des valeurs des paramètres se fait en envoyant la commande `STATE?`, puis en attendant un court instant avant de lire les caractères qui sont retournés, jusqu'à la réception du caractère retour à la ligne (*line feed*) `\n`. La réponse contient `STATE fmin fmax fstep`. Le déclenchement du balayage de la fréquence est initialisé par l'envoi de la commande `SWEEP?` et la réception des valeurs se fait ensuite ligne par ligne. Lorsque toutes les données ont été mises à disposition, on peut fermer le port série et préparer le tracé des courbes.

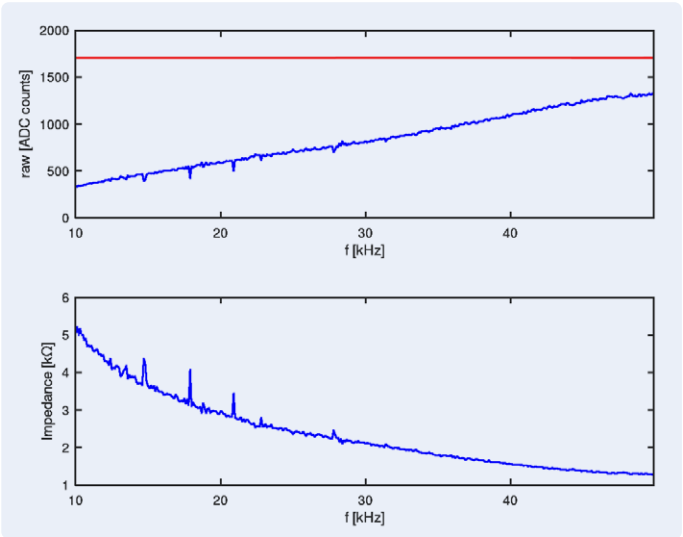


Figure 3. Valeurs brutes et impédance d'un condensateur utilisé en test.

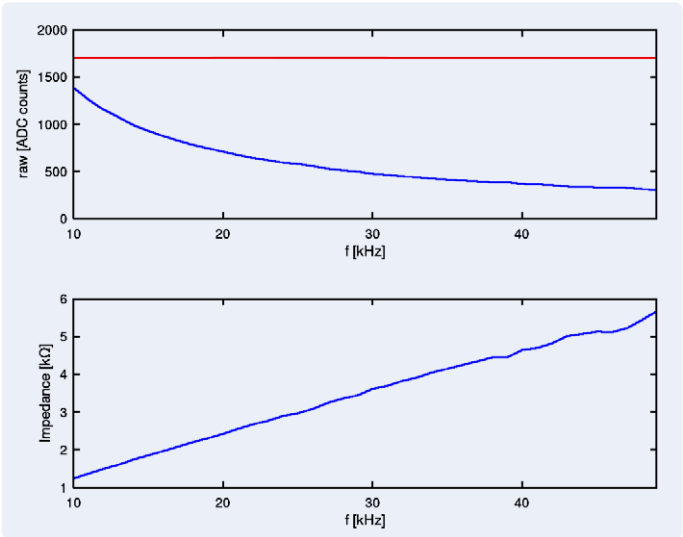


Figure 4. Données brutes et impédance dans le cas de test d'une inductance.

Les procédures scripts pour Octave sont incluses dans la *toolbox instrument* qui doit-être installée et pour Python dans le *Python-serial package* se trouvent dans l'archive logiciel accompagnant cet article. Notez que pour réaliser les tests préliminaires, un simple programme terminal tel que Putty peut être utilisé.

Utilisation du système

Pour commencer à utiliser le système, une résistance de calibration de 1 kΩ est utilisée comme composant de test et le balayage en fréquence est exécuté. Ensuite, la commande `SAVECALIB` qui est décrite en détail dans le script Octave `nwa.m` sauvegarde les données de calibration dans l'ESP32. Lorsque la commande `SAVECALIB` est exécutée, la résistance est remplacée par un condensateur de 2,2 nF, et `nva.m` est à nouveau exécuté, le tracé des données correctement calibrées de la **figure 3** apparait. Le tracé montre la dépendance inverse de la fréquence de laquelle on déduit la capacitance à l'aide de la commande Octave :

```
capacitance_nF=
     mean(1./(2*pi*Zabs*1e3.*xx*1e3))*1e9
```

Selon la formule :

$C = 1/(2\pi f Z_{abs})$

La moyenne de toutes les valeurs est réalisée par la fonction `mean()`. Les puissances de dix sont nécessaires pour prendre en considération le multiplicateur kilo de kΩ et de nano dans nF. Par exemple, `1e9` à la fin de la formule convertit la valeur du condensateur de Farad en nanoFarad.
Le tracé de la **figure 4** est obtenu en réitérant le balayage après avoir remplacé le condensateur by une inductance de 22 mH. Comme on s'y attendait, l'impédance de l'inductance sur le tracé inférieur augmente avec la fréquence, l'inductance peut être déterminée par la formule :

$L = Z_{abs}/2\pi f$

Ou obtenue en Octave par :

```
inductance_mH=
     mean(1e3*Zabs./(2*pi*xx*1e3))*1e3
```

qui réalise également la moyenne de tous les points obtenus, à l'aide de `mean()`. Les puissances de dix sont nécessaires pour tenir compte des multiplicateurs kilo de kΩ et khz et milli de mH.

Interface utilisateur Web

Jusqu'à présent, les mesures d'impédance ont été faites depuis Octave ou Python, nous allons maintenant utiliser un navigateur web pour contrôler et afficher les mesures. Pour cela, l'ESP32 est configuré afin d'exécuter un serveur web qui crée la page web de la **figure 5**. Quand le navigateur reçoit cette page web, un code javascript intégré ouvre un deuxième canal de communication série de l'ESP32, basé sur les websockets. Un websocket assume le rôle d'une communication série, il est ici utilisé pour assurer la transmission et la réception de messages. L'envoi de ces messages est déclenché par un clic sur les boutons situés en haut de la page web. Ils provoquent un balayage de la fréquence, l'effacement des courbes affichées et sauvegardent les données de calibration. Sur la ligne en-dessous, les paramètres de contrôle de l'ESP32 incluant la gamme du balayage, le taux d'échantillonnage du CAN peuvent être modifiés à partir des menus de sélection. Les boutons de la troisième ligne réalisent le calcul de la capacitance, de la résistance et de l'inductance, et en indiquent le résultat sur la ligne en-dessous des tracés. La ligne inférieure affiche les valeurs reçues de l'ESP32. Sur le tracé, l'axe horizontal indique la fréquence selon la gamme spécifiée dans les menus en haut de la page. L'axe vertical indique Z_{abs}, relativement à la résistance de calibration dont la valeur est affichée par la ligne horizontale bleue. En cliquant sur le tracé, les valeurs de la fréquence et de Z_{abs} apparaissent dans la ligne d'état.

Croquis de l'ESP32

Après avoir spécifié le nom du réseau wifi (SSID) et son mot-de-passe, les fichiers de support pour le wifi, websocket et webserver doivent être inclus, on déclare alors le serveur web `server2` afin de communiquer avec le réseau par le port 80, et par le port 81 pour le `websocket`. Les messages sous forme de texte échangés par les navigateurs sont formattés selon JSON, sous la forme `{"INFO":"Yada yada"}`. L'analyse grammaticale de ces messages et la génération de messages plus complexes sont réalisées par la bibliothèque *ArduinoJson*, tandis que le support du CNA et du CAN (ADC et DAC) est fourni par *ESP32_I2Sconfig.h* comme précédemment.

```
const char* ssid     = "YOUR_SSID";
const char* password = "YOUR_PASSWORD";
#include <WiFi.h>
#include <WebSocketsServer.h>
WebSocketsServer webSocket = WebSocketsServer(81);
#include <WebServer.h>
#include <SPIFFS.h>
WebServer server2(80);
#include <ArduinoJson.h>
#include "ESP32_I2Sconfig.h"
```

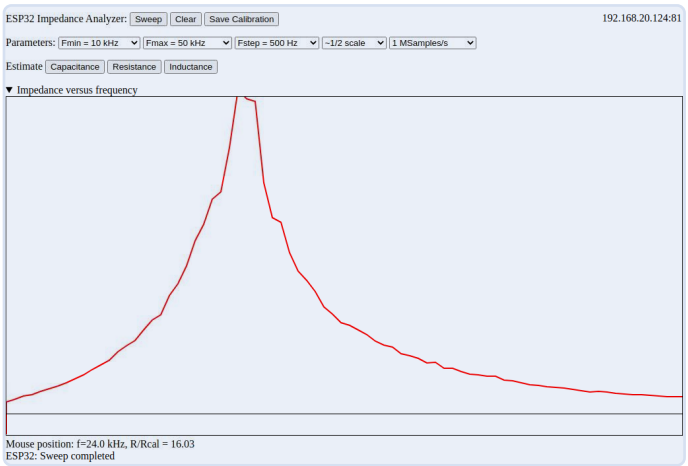


Figure 5. La page web générée par l'ESP32 montre la résonance obtenue sur la courbe d'impédance d'un condensateur de 2,2 nF, d'une inductance de 22 mH et d'une résistance de 33 kΩ connectés en parallèle.

Après avoir spécifié diverses variables, les fonctions de support sont définies, parmi lesquelles `webSocketEvent()` est la plus importante. Elle est appelée à chaque fois qu'un message envoyé par le navigateur arrive. S'il s'agit d'un message au format JSON, la fraction de code suivante extrait la commande `cmd` et la valeur `val` à l'aide des fonctions de la bibliothèque *ArduinoJson*. Selon la commande reçue, les actions appropriées sont alors exécutées. Par exemple, la réception de la commande `SWEEP` renverra des informations au navigateur par la fonction `sendMSG()` et positionnera la variable `mmode` à la valeur 1 qui est utilisée dans la boucle principale. `webSocketEvent()` étant appelée de façon asynchrone, elle interrompt les autres activités et doit être maintenue le plus court possible ; par conséquent, le balayage en fréquence est confié au programme principal. D'autre part, la définition de la fréquence de départ du balayage `freqmin` n'utilise que très peu de cycles de la CPU et se trouve dans la fonction `webSocketEvent()`.

```
DynamicJsonDocument root(300);
deserializeJson(root,payload);
const char *cmd = root["cmd"];
const long val = root["val"];
if (strstr(cmd,"SWEEP")) {
 sendMSG("INFO","ESP32: Received Sweep command");
 mmode=1;
} else if (strstr(cmd,"FMIN")) {
 freqmin=val;
 Serial.printf("FreqMin = %g\n",freqmin);
} else

...
```

Une suite de commandes similaires à celles utilisées précédemment est traitée de façon identique.
L'exemple de code suivant, extrait de la fonction `setup()`, connecte d'abord le réseau WLAN selon les paramètres fournis précédemment. Elle envoie des points sur la liaison série jusqu'à ce que cela réussisse, puis elle affiche l'adresse IP, démarre la communication avec le websocket, et paramètre `WebSocketEvent` de façon à traiter les messages provenant du websocket. Notez que la ligne de communication série n'est pas absolument nécessaire à ce stade, mais elle permet de mieux observer ce qui se passe dans l'ESP32, en particulier durant la phase de développement du système.

```
WiFi.begin(ssid, password);
while(WiFi.status() != WL_CONNECTED)
     {Serial.print("."); delay(500);}
Serial.print("\nConnected to ");
Serial.print(ssid);
Serial.print(" with IP address: ");
Serial.println(WiFi.localIP());
webSocket.begin();
webSocket.onEvent(webSocketEvent);
```

Ensuite, toujours dans `setup()`, on vérifie que le système de fichier SPIFFS a bien été créé, comme dans la première partie puis on en extrait les données de calibration. Cependant, à cet instant, le système SPIFFS contient également le fichier esp32_impedance_analyser.html qui décrit le contenu de la page web. Après avoir démarré `server2`, ce fichier HTML est utilisé par défaut lorsqu'un navigateur web se connecte comme indiqué par le premier argument dans le code suivant :

```
server2.serveStatic()
server2.begin();
server2.serveStatic("/",SPIFFS,
     "/esp32_impedance_analyzer.html");
```

La suite du code de la fonction `setup()` ressemble beaucoup au contenu du sketch précédent.
Dans la fonction `loop()`, ce qui concerne les serveurs http et websocket est tout d'abord traité, avant la vérification de la variable `info_available`. Cela est réalisé par `sendMSG()` qui informe que `info_buffer` contient un message au format JSON qui est traité immédiatement transmis au navigateur par un appel à `webSocket.sendTXT()`.

```
server2.handleClient();   // handle http server
webSocket.loop();         // handle websocket server
if (info_available==1) {
 info_available=0;
 webSocket.sendTXT(websock_num,
     info_buffer,strlen(info_buffer));
}
```

La valeur de `mmode` est ensuite vérifiée, et l'action appropriée est initialisée. Par exemple, `mmode==1` active le balayage de la fréquence par un code très similaire à celui précédemment utilisé. Dans ce cas, un message au format JSON nommé `WFO` (pour waveform ou forme d'onde) est créé et enregistré dans la variable `doc` par la commande :

```
doc["WF0"][nn-1]=floor((512/16)*Z);
```

dans laquelle la variable `nn` permet de reboucler sur toutes les valeurs de la fréquence et `Z` est la valeur absolue de l'impédance à cette fréquence. Notez que la variable est calibrée pour une couverture de 0 à 16 kΩ sur 512 pixels et ses valeurs sont converties en un entier par le fonction `floor()`. Lorsque la boucle est terminée, la courbe est envoyée au navigateur par :

```
serializeJson(doc,out);
webSocket.sendTXT(websock_num,out,strlen(out));
sendMSG("INFO","ESP32: Sweep completed");
```

et informe de la fin du balayage par `sendMSG()`. Les autres valeurs de `mmode` permettent de sauvegarder les données de calibration et envoient au navigateur les valeurs estimées du condensateur, de la résistance, ou de l'inductance.
Le fichier *esp32_impedance_analyser.html* décrit la page web et contient le code javascript qui la rend dynamique.

La page web

La plupart des pages web interactives utilisent les balises `<style>` pour décrire les aspects génériques de l'apparence des objets affichés par la page, suivies par des commandes HTML décrivant la page web et les instructions javascript qui assurent son interactivité.

Les instructions de style suivantes qui apparaissent au début du fichier *esp32_impedance_analyzer.html* définissent un cadre entourant la zone affichée et l'affichage de deux tracés rouge et noir. La dernière instruction permet l'affichage de l'adresse IP en haut et à droite de la page :

```
<style>
#displayarea { border: 1px solid black; }
#trace0 { fill: none; stroke: red; stroke-width: 2px;}
#trace1 { fill: none; stroke: black; stroke-width: 1px;}
#ip {float: right;}
</style>
```

La partie principale de la description de la page est insérée entre les balises `<BODY>` et `</BODY>`. Au début de cette section se trouvent la définition des boutons. La définition du premier bouton encadrée par des balises `button` est la suivante :

```
<button id="sweep" type="button"
     onclick="sweep();">Sweep</button>
```

`Sweep` est inscrit sur le bouton, un clic provoque l'exécution de la fonction `sweep()`. Ce type d'actions liées à un évènement sont baptisées fonctions de rappel. L'assignation de l'identificateur `sweep` au bouton permet de changer ultérieurement ses propriétés, par exemple le texte affiché ou l'action déclenchée. La définition de l'autre bouton suit les mêmes règles.
Le menu de sélection de la deuxième ligne utilise une syntaxe légèrement différente. La définition de ce menu est encadrée par des balises `SELECT`. Si l'une des entrées est choisie, elle appelle `setDataFreqMin(thisvalue)`, où `thisvalue` est la valeur spécifiée dans les différentes balises `OPTION`. Les balises `OPTGROUP` sont uniquement présentes pour agrémenter la clarté du code.

```
<SELECT onchange="setDacFreqMin(this.value);">
<OPTGROUP label="Sweep start frequency">
<OPTION value="1000">Fmin = 1 kHz</OPTION>
<OPTION value="2000">Fmin = 2 kHz</OPTION>
<OPTION value="5000">Fmin = 5 kHz</OPTION>
<OPTION selected="selected" value="10000">
    Fmin = 10 kHz</OPTION>
<OPTION value="20000">Fmin = 20 kHz</OPTION>
<OPTION value="50000">Fmin = 50 kHz</OPTION>
  </OPTGROUP>
</SELECT>
```

Pour terminer, la zone affichant l'impédance est de type SVG (*Scalable Vector Graphic* c'est-à-dire affichage graphique vectoriel adaptable), elle possède une taille de 1024 x 512 pixels et affiche deux courbes dont les identificateurs sont `trace0` et `trace1`. L'identificateur `id` permet leur modification ultérieure. La propriété d décrit la forme d'onde. L'initialisation est faite en déplaçant le curseur sur le pixel (0,200) avec `M0 200`. Par la suite, `d` est redéfinie par la courbe `WFO` en provenance de l'ESP32.

```
<svg id="displayarea" width="1024px" height="512px">
<path id="trace0" d="M0 200" />
<path id="trace1" d="M0 200" />
</svg>
```

À la fin, les deux lignes d'état sont définies par :

```
<div id="status">Status window</div>
<div id="reply">Reply from ESP32</div>
```

Elles sont identifiées par l'identificateur `id`, qui permet la modification ultérieure du texte affiché. De façon identique, la zone affichant l'adresse IP est identifiée par l'identificateur `ip` en haut à droite.

JavaScript

Toutes les commandes JavaScript sont insérées entre des balises `SCRIPT`. Après la balise d'ouverture, plusieurs variables sont définies, et l'adresse IP de l'ESP32 est déterminée par :

```
var ipaddr=location.hostname + ":81";
document.getElementById('ip').innerHTML=ipaddr;
```

Ici, le numéro de port du websocket est ajouté à l'ESP32 et le texte de la balise, identifié `ip`, est immédiatement mis à jour. L'espace alloué à l'affichage du texte est accédé à la deuxième ligne par une commande de construction assez longue, dans laquelle document se réfère à la page web elle-même. La partie suivant le point donne accès à l'élément nommé `ip` et `innerHTML` se réfère au texte affiché qui est alors modifié pour afficher `ipaddr`. Cette construction de commande est utilisée de façon extensive pour accéder à des éléments nommés et modifier leurs propriétés. Les fonctions `toStatus()` et `toReply()` suivent cette partie pour copier le texte dans les lignes d'état en bas de la page web.
Connaissant maintenant l'adresse du websocket de l'ESP32, ce dernier est ouvert par la commande new `WebSocket()` puis les fonctions de rappel sont ajoutées aux évènements `onopen`, `onclose` et quelques autres. Un court message est souvent envoyé à la console JavaScript par la commande `console.log ()`. La console est accessible par les outils de développement du navigateur, en utilisant par exemple, le raccourci clavier Ctrl-Shift dans Chrome.

```
var websock = new WebSocket('ws://' + ipaddr);
websock.onopen = function(evt)
  {console.log('websock open');};
```

La fonction de rappel la plus intéressante réagit à l'arrivée des messages provenant de l'ESP32. La fonction raccourcie `websock.onmessage()` analyse le message au format JSON dont le contenu est enregistré dans `event.data` et place la paire de valeurs de la commande dans la structure `stuff`. Si `stuff` contient la commande INFO, la valeur associée est mémorisée dans `val` et affichée sur la page web par `toReply()`. Si la commande est `WFO`, elle contient la courbe des données d'impédance. Le nombre de points transmis est déterminé par `val.length`. Pour s'y adapter, l'axe horizontal est mis à l'échelle afin d'utiliser la totalité des 1024 pixels. La propriété `d` du tracé est initialisée et les points y sont ajoutés, un pour chaque entrée de `val`. Le pixel correspondant à la valeur `0,0` étant en haut à gauche, la position verticale doit être inversée en affichant le pixel `512-val[i]`.

Pour terminer, la ligne de référence de calibration par la résistance, de couleur noire, est affichée.

```
websock.onmessage=function(event) {
var stuff=JSON.parse(event.data);
var val=stuff["INFO"]; // info
if (val != undefined) {toReply(val);}
var val=stuff["WF0"]; // waveform0
if (val != undefined) {
nstep=Math.floor(0.5+1024/val.length)
pixmax=nstep*val.length;
var d="M0 511";
for (i=0; i<val.length; i++)
{d += ' L' + (nstep*i) + ' ' + (512-val[i]);}
document.getElementById('trace0').setAttribute('d',d);
d="M0 480 L1024 480";
document.getElementById('trace1').setAttribute('d',d);
}
}
```

La suite du code JavaScript est principalement constituée de l'affectation des fonctions de rappel aux boutons menus. La fonction `sweep()`, déclenchée par le bouton correspondant est la suivante :

```
function sweep() {
websock.send(JSON.stringify
     ({ "cmd" : "SWEEP", "val" : -1 }));
}
```

La fonction `JSON.stringify()` encapsule ses arguments dans un message correctement formaté et `websocket.send()` le transmet à l'ESP32. Toutes les autres fonctions de rappel utilisent une structuration identique.
Juste avant la fin de la section JavaScript, `showCoordinates()` est définie pour afficher la fréquence et l'impédance correspondant au pixel du graphique dans la zone que l'on peut choisir en cliquant. Cette fonction répond à un évènement `mousedown` en attachant `addEventListener()` à `displayarea`:

```
document.getElementById("displayarea")
 .addEventListener('mousedown', showCoordinates, false);
```

Cette possibilité est très pratique pour déterminer directement les fréquences et les impédances correspondantes depuis l'affichage du tracé.

Sur la **figure 5**, on peut voir que la résonance d'un circuit parallèle RLC utilisé en test, indique un pic de résonance proche de la fréquence 24 kHz où cette impédance est supérieure à 16 kΩ.
Le microprogramme et les autres fichiers peuvent être téléchargés depuis [7]. L'analyseur d'impédance est maintenant autonome dans le sens où aucun programme de contrôle n'est nécessaire à son fonctionnement. Tous les échanges se font entre l'ESP32 et un navigateur, même dans le cas d'un smartphone.

VF : Jean Boyer — 230039-04

Des questions, des commentaires ?

Contactez Elektor (redaction@elektor.fr).

À propos de l'auteur

L'intérêt de Volker Ziemanns pour l'électronique a commencé avec l'amplificateur Edwin de 40 W (Elektor au milieu des années 1970), mais il a suivi une orientation différente et étudié la physique, puis a travaillé sur des accélérateurs de particules – au SLAC aux États-Unis, au CERN à Genève et maintenant à Uppsala en Suède. L'électronique ayant un rôle primordial dans le contrôle et l'acquisition de données des accélérateurs, son intérêt précoce lui a été utile au cours de sa carrière. Il enseigne maintenant à l'Université d'Uppsala. Un de ses livres traitant de l'acquisition de données avec Arduino et Rapsberry Pi traite du sujet de cet article.

Produits

- **ESP32-DevKitC-32D (SKU 18701)**
  www.elektor.fr/18701
- **OWON HDS1021M-N oscilloscope à 1 voie (20 MHz) + multimètre (SKU 18778)**
  www.elektor.fr/18778

LIENS

[1] ESP32 Technical Reference Manual (Version 4.7), disponible à l'adresse : https://www.espressif.com/sites/default/files/documentation/esp32_technical_reference_manual_en.pdf
[2] Descritpion du générateur sinusoidal ESP32 : https://github.com/krzychb/dac-cosine
[3] Site web Arduino : https://www.arduino.cc
[4] Fonctions de support Arduino : https://github.com/espressif/arduino-esp32
[5] V. Ziemann, A Hands-on Course in Sensors Using the Arduino and Raspberry Pi, 2nd ed., CRC Press, Boca Raton, 2023:
[6] ESP32fs Plugin : https://github.com/me-no-dev/arduino-esp32fs-plugin
[7] Code source de ce project sur GitHub : https://tinyurl.com/4awvvemc

visite à **domicile**

encourageons les réalisations personnelles

Figure 1. Le laboratoire personnel de Ruud bien rangé.

Ruud van der Meer (Pays Bas) et Eric Bogers (Elektor)

Après une longue et brillante carrière en électronique, lorsqu'ils atteignent l'âge de la retraite, de nombreuses personnes portent leur attention sur des sujets complètement différents, comme par exemple la culture des tomates ou l'élevage des cochons d'inde, pourvu que cela n'ait rien à voir avec l'électronique. Ruud van der Meer qui réside à Roelofarendsveen, est une des exceptions. Après avoir passé plus de 40 ans chez Siemens, son enthousiasme est toujours intact.

Actuellement, il n'y a pas assez de réalisations électroniques personnelles.

Après avoir étudié dans une école professionnelle et suivi une formation en mesures et techniques de contrôle, puis un cursus supérieur de technicien en électronique, Ruud commença sa carrière chez Siemens en 1970, dans le département des équipements de mesure et contrôle, instruments de mesures et calibration. Selon ses propres mots :
« Il y avait des tas d'opportunités. En un temps très court, j'ai développé de nombreux projets, en particulier des afficheurs à LED, des contrôleurs PID (Proportionnel, Intégrale et Dérivée) et des transducteurs de mesures. En développant des équipements et systèmes de test, j'ai pu améliorer de nombreux processus de travail. Après plusieurs années, je suis devenu leader d'équipe, et en coopération avec le Siemens Hobby Computer Club, j'ai développé notre propre PC alors appelé SUMO80. »

Ruud remarqua très tôt que certaines personnes avaient besoin d'un peu d'aide pour se frayer un chemin dans les technologies nouvelles, c'est pourquoi il a alors commencé à enseigner des formations dans sa compagnie ainsi qu'au centre de formation régional (Regional Training Center), dans les domaines de l'ingénierie électrique et électronique, les méthodes numériques, les technologies de communication et la mécatronique.
« Par la suite, j'étais à la recherche d'un nouveau challenge qui se présenta quand on me demanda de prendre la direction de la formation professionnelle chez Siemens. Il y avait alors environ 60 étudiants dans le programme d'ingénierie électrique. L'un de mes objectifs (en plus de la partie théorique du programme), était d'améliorer l'intégration des nouveaux embauchés dans la compagnie, processus qui avait décliné au cours du temps. En plus d'autres approches, j'y suis parvenu en étendant le programme de formation à l'électronique, la technologie PLC (*Programmable Logic Controller*), les technologies de communication ainsi que la mécatronique la dernière année, en coopération avec des conférenciers de l'université de Delft. »

Le programme de formation professionnelle de la compagnie Siemens a été clos en 2005, Ruud dût alors rechercher un job dans un autre domaine (mais toujours chez Siemens). Il a finalement trouvé un poste dans le département responsable de la domotique et de la protection incendie.
« Pour faire court, en termes de processus de travail, on en était à l'âge de pierre, mais rapidement, je me suis consacré à la remise en ordre du département. Puis vint l'heure de la retraite en 2016. »

Il était alors, à nouveau, temps de faire quelque chose de différent. Ruud est devenu le premier conseiller en énergie de la municipalité de Kaag en Braassem, tout en formulant des recommandations concernant la durabilité. En partie pour cela, il a développé de nombreux systèmes de domotique basés sur Arduino.
« À la suite de multiples présentations concernant le microcontrôleur Arduino au Hobby Computer Club, il m'a été demandé de le faire pour le Leidershop Adult Creation Center (LVU) il y a de cela six ans. Ensuite, j'ai également enseigné des formations sur le microcontrôleur Arduino dans les centres d'apprentissage des adultes d'Alphen aan de Rijn, Lisse et Hillegom. Nous avons maintenant réuni un groupe important de personnes enthousiastes et nous organisons mensuellement un Café Arduino, sans oublier la journée mondiale Arduino. Il y a également beaucoup d'intérêt pour le cours de réalisations électroniques. »
« Il n'y a pas suffisamment de réalisations électroniques personnelles (*DIY*), bien que ce soit devenu si facile grâce à Arduino. »

Mon laboratoire personnel

Ruud a développé de nombreux supports de formation (schémas, circuits imprimés et logiciels) pour l'ensemble des cours et sessions pratiques. La plupart ont été créés dans son laboratoire personnel (mais également, au début, diverses réalisations pour Siemens). La **figure 1** donne une impression de son laboratoire personnel.

Figure 3. La zone électronique et ses équipements de test et mesures.

« Mon laboratoire (ou plus simplement l'atelier de mon hobby) existe depuis 1985. Ce que vous voyez sur la photographie est le local que j'utilise depuis 2005. La partie informatique est sur la gauche, l'ensemble électronique et mesure au milieu, sur la droite, l'atelier de mécanique. C'est ici que j'ai développé mon robot de nettoyage, exemple d'exercice pratique de mécatronique (**figure 2**). »
« J'ai développé de nombreux équipements de test et de mesures. Je n'ai malheureusement plus mon oscilloscope à tube de réalisation personnelle, mais de nombreux dispositifs, dont certains élaborés par des collègues, sont toujours présents. La **figure 3** en illustre l'envergure. »
« L'un de mes projets récents est basé sur le kit calculateur *maison* de KKmoon. À mon avis, il ne possédait pas assez de fonctionnalités dans sa forme originale, de plus, il était difficile à reprogrammer. C'est malgré tout une excellente réalisation au niveau matériel, c'est pourquoi, j'ai décidé de lui développer un nouveau cœur, basé évidemment sur une carte Arduino. Nous construisons actuellement un grand nombre de ces calculateurs *maison* avec un groupe d'étudiants Arduino. Les fonctionnalités du calculateur (en plus des fonctions de calcul de base) ont été conçues pour les personnes qui sont fréquemment menées à programmer d'autres activités techniques. La **figure 4** en montre l'aspect extérieur et une vue interne. »
Ce multi calculateur Arduino fera l'objet d'une présentation plus détaillée dans une des prochaines éditions du magazine Elektor.

VF : Jean Boyer — 230035-04

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

Figure 2. Le robot de nettoyage.

Figure 4. Vues externe et intérieure du multi calculateur.

la carte d'apprentissage MCCAB pour Arduino Nano

plateforme pour le cours « Microcontrollers Hands-On Course »

Wolfgang Trampert (Allemagne) et Jan Buiting (Elektor)

L'assemblage de petits circuits d'extension pour Arduino consiste à placer des composants sur une plaque d'essai et de les connecter avec quelques fils de connexion colorés. Cependant, toutes ces connexions prennent souvent plus de temps que l'écriture du programme. De plus, la disposition des composants, par exemple, pour un feu de circulation à 11 LED et autant de résistances en série est sujette à des erreurs et à des heures précieuses consacrées au débogage du matériel ! Dans de tels cas, une carte d'apprentissage dédiée, comme la nouvelle « MCCAB » d'Elektor, est plus pratique, surtout qu'elle est livrée avec un excellent manuel utilisateur.

Note de la redaction.*cet article est un extrait du livre Microcontrollers Hands-on Course for Arduino Starters (Elektor, 2023), formaté et légèrement modifié pour correspondre aux normes éditoriales et à la mise en page du magazine Elektor. L'auteur et l'éditeur ont fait de leur mieux pour l'éviter et seront heureux de répondre aux questions – pour les contacter, voir l'encadré* **« Des questions, des commentaires ? »**

La carte d'apprentissage pour Arduino Nano « MCCAB » (*Microcontroller Crash Course for Arduino Beginners*) a été spécialement conçue par l'auteur et fabriquée pour accompagner le guide d'Elektor *Microcontrollers Hands-on Course for Arduino Starters*. La carte MCCAB, le guide (en anglais ou en allemand) et une carte Arduino Nano sont disponibles en offre groupée dans l'e-choppe Elektor [1].

Sur la carte d'apprentissage – appelée MCCAB dans la suite de cet article – de nombreux composants sont déjà connectés au module de microcontrôleur branché ou peuvent être connectés avec de simples fils de connexion. Ces composants incluent des interrupteurs/boutons, des LED, des potentiomètres, des buzzers, un écran LCD, des interfaces et des convertisseurs de niveau pour les connexions séries, ainsi que des étages pilotes pour les appareils externes.

Il est possible de connecter des modules externes au MCCAB via un connecteur femelle, ou vous pouvez les relier au microcontrôleur de la carte via les interfaces série. Cela permet d'éviter l'assemblage fastidieux de circuits expérimentaux et de se concentrer sur l'essentiel, c'est-à-dire sur le logiciel, ou « le programme ». Nous chargerons donc les programmes créés dans nos exercices dans le microcontrôleur ATmega328P du MCCAB, où ils seront exécutés (comme code de programme).

Certaines des 32 broches du microcontrôleur ATmega328P sont utilisées pour son alimentation – un quartz est connecté à l'oscillateur intégré pour la génération du signal d'horloge – ou comme entrée

RESET pour le bouton-poussoir du module de microcontrôleur connecté au MCCAB (l'appui sur ce bouton réinitialise le microcontrôleur à l'état initial défini et redémarre le programme). Cependant, la majorité des broches sont des entrées/sorties à usage général (GPIO), qui peuvent être utilisées pour connecter le microcontrôleur au monde extérieur. Sur le MCCAB, ces broches – sauf quelques exceptions réservées à des fins internes – sont accessibles grâce à des barrettes de connecteurs.

La **figure 1** montre une vue du MCCAB avec ses blocs fonctionnels codés en couleur. Le mode d'emploi du MCCAB, avec une description détaillée de tous les composants, est téléchargeable gratuitement sur le site web d'Elektor [2].

Blocs fonctionnels du MCCAB

Sur la figure 1, on peut distinguer les blocs fonctionnels et les modules suivants sur le MCCAB.

(1) **Module microcontrôleur Arduino Nano** avec bouton RESET (flèche 1a), LED (flèche 1b) et prise mini USB pour la connexion au PC de l'utilisateur.

(2) Connecteurs SV5 et SV6 pour **les entrées/sorties du microcontrôleur**. En utilisant des câbles Dupont, il est possible de connecter les broches d'E/S (GPIO) du microcontrôleur aux composants internes de la carte d'apprentissage ou à des modules externes via ces deux connecteurs.

(3) **11 LED**, LD10-LD20 (indicateurs d'état pour les entrées/sorties D2-D12 du microcontrôleur). On peut connecter chaque LED aux broches affectées, D2-D12, via un cavalier sur le connecteur JP6.

(4) **Matrice LED 3×3** LD1-LD9 (9 LED rouges). Les colonnes sont connectées en permanence aux sorties D6, D7 et D8 du microcontrôleur, et les lignes peuvent être connectées aux broches D3, D4 et D5 avec des cavaliers sur le connecteur JP1.

Si les lignes de la matrice LED 3×3 sont connectées aux broches D3, D4 et D5 via les cavaliers sur le connecteur JP1 ou à d'autres broches du microcontrôleur avec des câbles Dupont, ces lignes, ainsi que les colonnes D6, D7 et D8, ne doivent jamais être utilisées pour d'autres tâches dans un croquis.

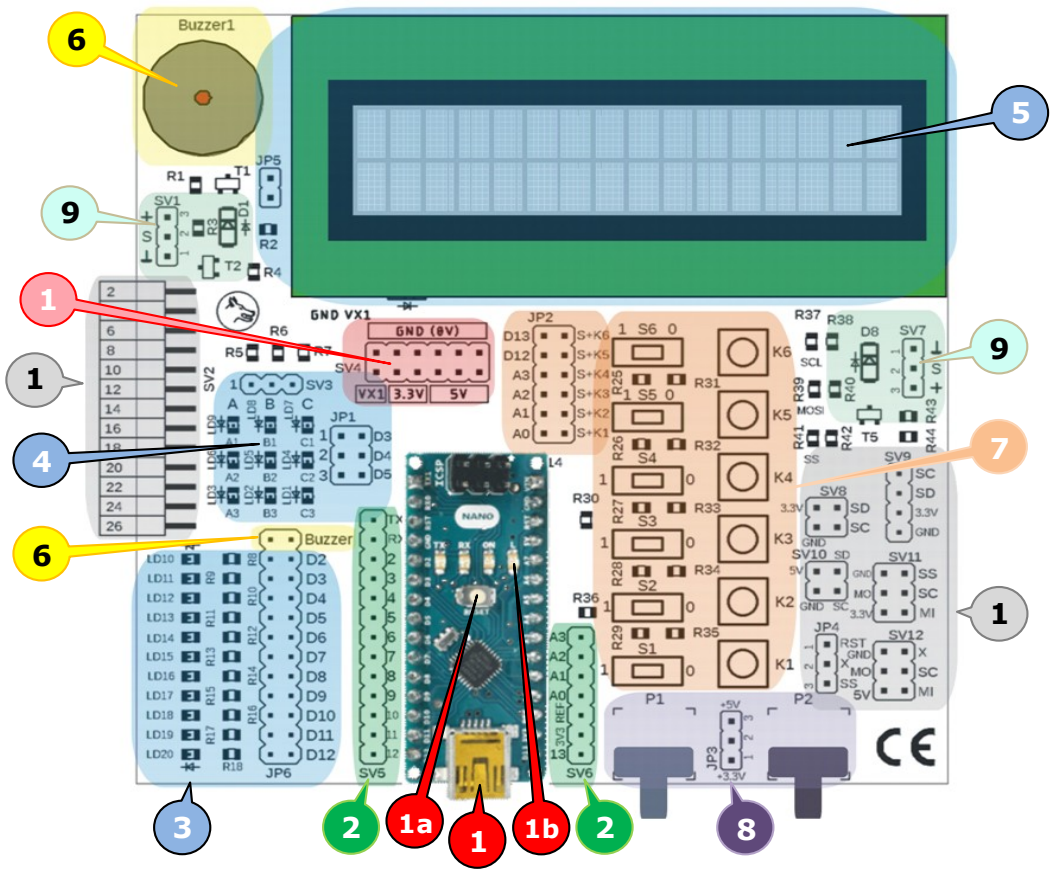


Figure 1. Schéma fonctionnel de la carte d'apprentissage pour Arduino Nano « MCCAB » d'Elektor.

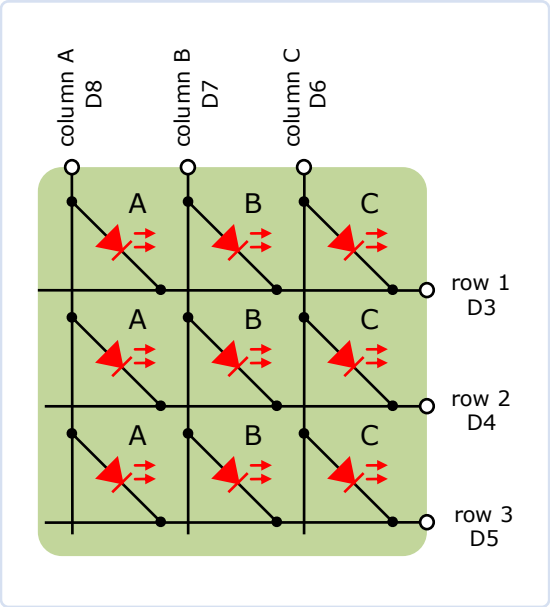


Figure 2. Matrice LED 3×3.

Une double affectation des broches de la matrice risque d'entraîner des dysfonctionnements, voire d'endommager le MCCAB !

(5) **Écran LCD** de 2×16 caractères, connecté via le bus I²C aux broches A4 et A5 du microcontrôleur. Il est possible de couper la tension qui alimente l'écran LCD en tirant le cavalier JP5, pour les essais ou les tests ou l'écran n'est pas utilisé.

(6) Il est possible de connecter **le buzzer piézoélectrique** Buzzer1 à la broche D9 avec un cavalier sur la position « Buzzer » du connecteur JP6.

(7) **6 interrupteurs à glissière, S1-S6**, connectés en parallèle à 6 boutons, **K1-K6**. Il est possible de les connecter aux entrées A0-A3 et D12-D13 du microcontrôleur via des cavaliers sur le connecteur JP2.

(8) **Potentiomètres P1 et P2**, dont les curseurs sont connectés aux broches d'entrée analogique du microcontrôleur, A6 et A7. L'alimentation 3,3 V ou 5 V peut être appliquée aux potentiomètres via le connecteur JP3.

Attention : les broches A6 et A7 de l'ATmega328P sont définies comme des entrées analogiques par l'architecture interne de la puce. Les configurer avec la fonction `pinMode()` n'est pas autorisé et peut conduire à un comportement erroné du programme.

(9) Les broches SV1 et SV7 sont **des sorties de commutation pour les appareils externes.**

(10) Connecteurs pour la liaison série des modules **SPI et I²C** externes.

(11) Barrette de connexion SV2 avec 2×13 broches pour **la connexion de modules externes.**

(12) Le connecteur SV4 est le **distributeur des tensions de fonctionnement de la carte.** Ces tensions peuvent alimenter des composants internes de la carte d'apprentissage ou des modules externes avec des câbles Dupont.

Les deux unités fonctionnelles un peu plus complexes, la matrice 3×3 LED et l'écran LCD (voir **figure 1**) seront décrites plus en détail dans la suite. Pour les parties restantes, veuillez vous référer à la description détaillée dans le document *MCCAB Operating Instructions* [2].

La bibliothèque *MCCAB_Lib* est disponible pour piloter la matrice LED 3x3, les LED LD10-LD20, les boutons K1-K6 et les interrupteurs S1-S6, ainsi que Buzzer1. La bibliothèque est disponible gratuitement par les propriétaires de MCCAB et intégrée dans l'EDI Arduino.

Pour contrôler l'écran LCD, nous utilisons la bibliothèque *LiquidCrystal_I2C*, que vous pouvez télécharger gratuitement et ajouter à l'EDI Arduino.

Matrice LED 3×3

Le MCCAB contient neuf diodes électroluminescentes placées dans une matrice (voir **figure 1**). La **figure 2** montre leur circuit de base.

La matrice se compose de trois colonnes et de trois lignes. Les colonnes sont nommées A, B et C, tandis que les lignes sont numérotées 1, 2 et 3. À chacune des neuf intersections colonne/ligne, une LED est connectée – son anode à la colonne et sa cathode à la ligne. Les neuf LED sont étiquetées en fonction de leurs positions colonne/ligne respectives, par exemple, « B2 » est connectée à la colonne B et à la ligne 2. Pour qu'une LED s'allume, sa colonne doit être au niveau logique 1 (+5 V) et sa ligne au niveau logique 0 (0 V).

Le manuel d'utilisation du MCCAB indique que les colonnes sont connectées en permanence aux broches D6-D8 du microcontrôleur. Vous pouvez connecter les lignes aux broches D3-D5 avec des cavaliers sur le connecteur JP1.

Si les lignes de la matrice LED 3×3 LED sont connectées aux broches D3, D4 et D5 via les cavaliers sur le connecteur JP1 ou aux autres broches du microcontrôleur avec des câbles Dupont, les lignes et les colonnes D6-D8 ne doivent pas être utilisées pour d'autres tâches dans votre croquis. Une telle connexion des broches de la matrice entraînerait des dysfonctionnements ou, dans le pire des cas, endommagerait le MCCAB ! **Si la matrice n'est pas utilisée dans un croquis, vous devez enlever les cavaliers sur le connecteur JP1 du MCCAB.**

Avantages de l'arrangement matriciel

Les matrices LED sont souvent utilisées, par exemple dans les stades, avec des LED colorées de forte puissance pour générer des images animées.

Si nous devions piloter 9 LED individuellement, nous aurions besoin de 9 broches du microcontrôleur. En organisant les LED dans une matrice, nous aurons besoin de six broches seulement. Plus il y a de LED à piloter, plus l'avantage de l'utilisation d'une matrice est grand : avec une matrice composée de huit colonnes et

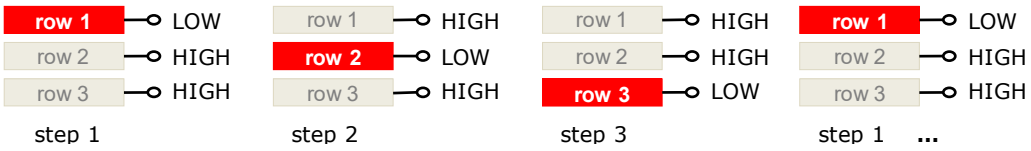


Figure 3. Activation progressive et cyclique des trois lignes qui composent la matrice.

huit lignes, nous pouvons piloter 64 LED, alors qu'avec des lignes de contrôle individuelles, 48 broches supplémentaires seraient nécessaires !

Pilotage de la matrice en mode multiplex

La **figure 2** montre que chaque colonne et chaque ligne de la matrice sur le MCCAB est connectée à trois LED. Par conséquent, le contrôle simultané de toutes les lignes et colonnes ne fonctionnerait pas, car les LED qui devraient en fait être éteintes seraient allumées involontairement. Au lieu de cela, comme le montre la **figure 3**, une seule ligne peut être activée à la fois et les colonnes doivent appliquer la séquence de bits de la ligne activée. Les deux autres lignes doivent être ouvertes pendant ce temps ou désactivées avec un niveau logique haut afin qu'aucun courant ne puisse les traverser.

Si les trois lignes sont rapidement activées les unes après les autres comme dans la **figure 3**, et à cause de la persistance rétinienne, on voit une image statique des neuf LED. Le programme utilisateur contrôle la matrice avec une boucle infinie, dans laquelle l'une des trois lignes, 1, 2 ou 3, est mise à un niveau logique bas cycliquement, tandis que les deux autres lignes sont mises à un niveau haut. Les connexions de colonne de toutes les LED à allumer dans la ligne active sont mises au niveau haut. Les connexions de colonne des LED qui doivent être éteintes dans la ligne active sont mises à un niveau logique bas.

Par exemple, pour allumer deux LED, A3 et C3, la ligne 3 doit être au niveau bas et les colonnes A et C au niveau haut, tandis que les deux lignes de la rangée, 1 et 2, sont au niveau haut et la colonne B au niveau bas.

LCD

Le MCCAB d'Elektor est équipé d'un afficheur LCD qui permet d'afficher du texte, des valeurs numériques et même des caractères spéciaux définis par l'utilisateur. L'afficheur utilisé comporte deux lignes de 16 colonnes. Il est possible d'afficher un caractère dans chaque colonne. Chaque caractère est formé des points d'une matrice 5×8, comme le montre la **figure 4**. Les séquences de bits de la matrice 5×8 points nécessaires pour chaque caractère individuel sont stockées à l'intérieur du contrôleur LCD selon la table ASCII.

L'ASCII est un code de 7 bits, alors que notre microcontrôleur ATmega328P est une « unité de 8 bits », ce qui signifie qu'il peut traiter un octet à la fois et stocker ses données sous la forme d'un octet. Par conséquent, le huitième bit est généralement mis à 0 lors du stockage des codes ASCII. Cependant, les développeurs du contrôleur d'affichage HD44780 (qui est standard dans cette catégorie et que l'on trouve sur presque tous les modules LCD) n'ont pas voulu laisser ce huitième bit inutilisé, et ont étendu le jeu de caractères ASCII de 128 codes de caractères supplémentaires (où le huitième bit est à la valeur logique 1). Pour cette raison, notre LCD utilise également les codes de caractères 128 à 255 (de manière plus ou moins contiguë). Outre les 95 caractères imprimables correspondant aux codes 32-126, l'ASCII compte également 33 caractères de contrôle, à savoir 0-31 et 127, qui sont « non imprimables ». Comme ces caractères sont destinés à des fins de contrôle, l'écran ne les affiche pas. Le contrôleur d'affichage HD44780 est disponible en deux versions : soit avec le code ROM A00, soit avec le code ROM A02. La version A00 affiche des caractères vides pour les codes 106-31 ainsi que 128-159. La version A02 affiche des caractères spéciaux pour ces espaces d'adresses (voir **tableau 1**). Il est impossible de prévoir laquelle des deux versions de ROM est installée dans l'écran LCD du MCCAB, car c'est le fabricant du module LCD qui fait ce choix.

L'utilisateur peut définir jusqu'à huit caractères

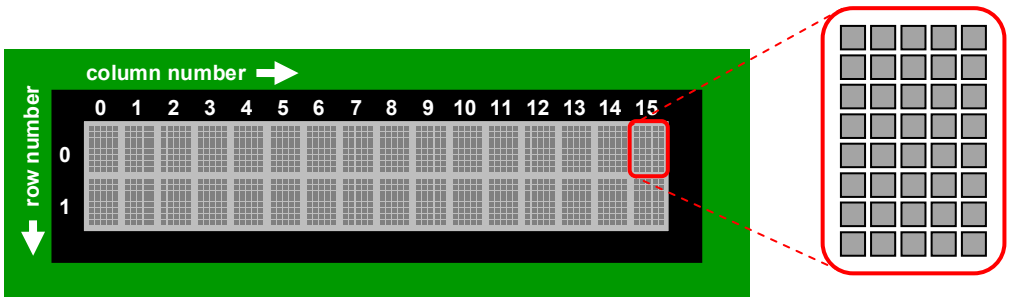


Figure 4. Illustration de l'afficheur LCD du MCCAB avec ses 16×2 caractères affichables.

Table 1. Allocation de mémoire de caractères pour les deux variantes de HD44780

Code de caractère	Code ROM A00	Code ROM A02
0–7	caractères définis par l'utilisateur	caractères définis par l'utilisateur
8–15	caractères définis par l'utilisateur	caractères définis par l'utilisateur
16–31	(sans affichage)	caractères visibles
32–127	caractères visibles	caractères visibles
128–159	(sans affichage)	caractères visibles
160–255	caractères visibles	caractères visibles

spéciaux et les afficher sur l'écran LCD. Les codes ASCII 0 à 7 sont attribués à ces huit caractères spéciaux. En option, il est possible d'adresser ces mêmes huit caractères spéciaux par l'intermédiaire des codes ASCII 8 à 15.

La **figure 4** montre également la numérotation des lignes et des colonnes de l'afficheur, qui, dans les deux cas, commence par 0. En spécifiant ces données, on peut écrire un caractère à une certaine position. À cette fin, l'écran LCD dispose d'un curseur qui détermine cette position. La bibliothèque LCD décrite dans le guide contient des méthodes pour positionner ce curseur.

Bien que l'afficheur ne puisse afficher que 16 caractères dans chaque ligne, la mémoire dans laquelle les caractères sont stockés dans le contrôleur d'affichage dispose en fait de 40 emplacements mémoire pour chaque ligne. Il y a un écart de 24 emplacements mémoire entre la dernière adresse d'affichage, 39, de la première ligne et l'adresse de début, 64, de la deuxième ligne (voir **figure 5**).

Grâce aux fonctions de décalage (*shift*), qui sont incluses dans la bibliothèque LCD, la zone de mémoire visible à l'afficheur peut être déplacée sur l'ensemble de la zone de mémoire.

La figure 5 comporte trois parties :

- en haut : la zone visible de l'afficheur dans le réglage de base, sans opérations de décalage préalables.
- au milieu : le contenu de l'afficheur par défaut après décalage à gauche.
- en bas : le contenu de l'afficheur par défaut après un décalage à droite.

Réglage du contraste de l'écran LCD

Le MCCAB ne dispose pas d'un réglage de contraste pour l'afficheur LCD. Pour cette raison, et parce que le contraste de l'écran peut varier en fonction des conditions environnementales (telles que la température) ou du vieillissement, le LCD est doté d'un potentiomètre de réglage qui permet d'ajuster le contraste. Ce petit potentiomètre est accessible par le dessous de la carte et est marqué d'une flèche. Réglez le contraste avec un petit tournevis lors de la première utilisation ou lorsque vous en avez besoin.

Astuce : Si aucun caractère n'est affiché lors de la première utilisation de l'afficheur LCD, c'est probablement parce que le réglage du contraste n'a pas été effectué correctement !

Transmission de données du microcontrôleur à l'écran LCD

Chaque module LCD est équipé d'un contrôleur d'affichage HD44780, qui reçoit les codes (ASCII) des caractères envoyés par le microcontrôleur via une interface, génère les caractères correspondants de la matrice à

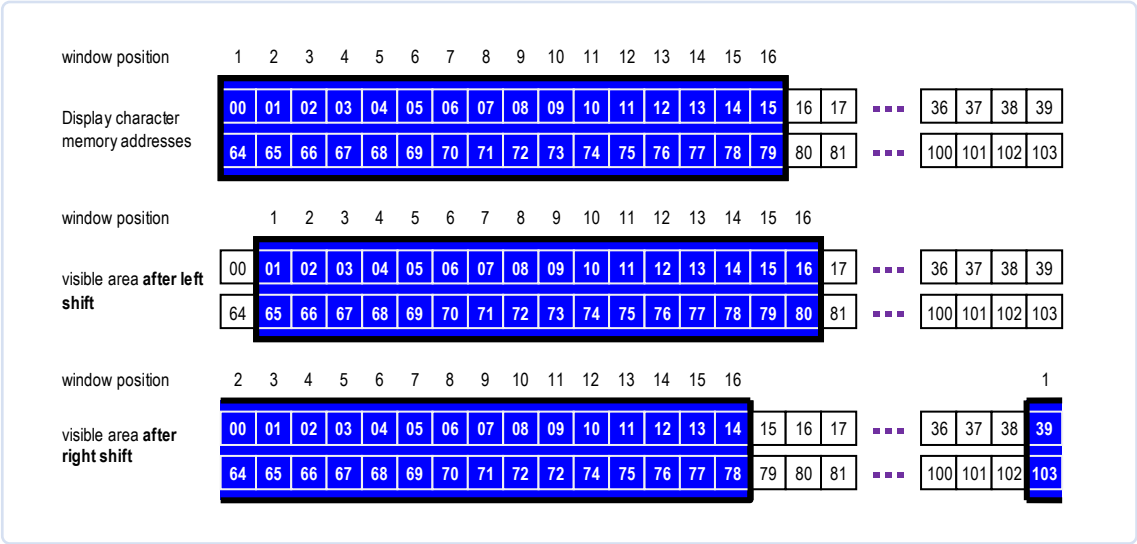


Figure 5. Contenu de l'afficheur avant et après le décalage.

points 5×8 mentionnée ci-dessus à partir des codes et les affiche.

Le contrôleur LCD HD44780 ne dispose que d'une interface parallèle pour recevoir les données à afficher, c'est-à-dire que le microcontrôleur écrit un octet composé de huit bits, D0-D7, dans le registre d'entrée du contrôleur HD44780, puis sur le LCD avec trois signaux de commande : RS, RW et E, comme le montre la **figure 6**. Cette méthode de transmission de données en parallèle limiterait considérablement les capacités du MCCAB car l'écran LCD occuperait déjà 11 des 16 broches disponibles du microcontrôleur ATmega328P !

La deuxième option pour la transmission des données au contrôleur HD44780, dans laquelle les huit bits de données sont transmis en deux paquets de quatre bits chacun, n'est pas réellement utile, car, même dans ce cas, sept broches du microcontrôleur seraient encore utilisées par le LCD, puisque les trois signaux de contrôle, RS, RW, E, sont également nécessaires dans ce cas.

Pour cette raison, le module LCD sur le MCCAB est contrôlé via le bus I²C, un bus série synchrone qui consiste en une seule ligne de données (SDA), une ligne d'horloge (SCL), et il transmet les données bit par bit. Le trafic de données via l'interface I²C est réalisé grâce à deux lignes du microcontrôleur ATmega328P, A4 (SDA) et A5 (SCL) (voir **figure 7**).

Un adaptateur supplémentaire situé sous le module LCD convertit les signaux I²C en signaux parallèles. Cet adaptateur facilite la méthode mentionnée ci-dessus. Il permet de transmettre deux ensembles de quatre bits à la suite via les lignes de données D4-D7, c'est-à-dire que les bits de données D4-D7, puis les bits de données D0-D3 sont transmis sur les mêmes lignes, D4-D7.

Étant donné que les lignes A4 et A5 du microcontrôleur sur le MCCAB sont de toute façon réservées à l'interface I²C, aucune ressource n'est perdue avec ce type de transmission à l'écran LCD car, en principe, plusieurs participant peuvent être connectés au bus I²C en même temps. Puisque chaque participant connecté au bus occupe sa propre adresse I²C, il ne « voit » qu'il est adressé que lorsqu'un paquet de données arrive avec l'adresse de ce participant. Le LCD du MCCAB occupe généralement l'adresse I²C 0x27. Si l'adresse diffère en raison du fabricant, cela est indiqué sur l'écran.

230215-04

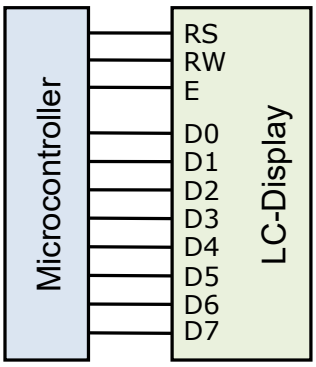


Figure 6. Pilotage de l'afficheur LCD avec huit bits de données et trois lignes de commande.

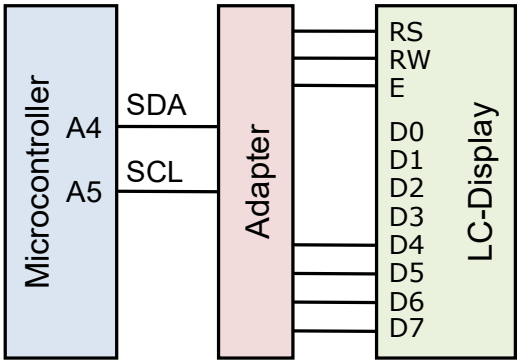


Figure 7. Pilotage de l'afficheur LCD via le bus I_2C plutôt qu'en parallèle (voir figure 6).

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

Produits

- **Microcontrollers Hands-on Course for Arduino Starters (offre groupée) (SKU 20440)**
  - 1 × Guide : *Microcontrollers Hands-On Course for Arduino Starters*
  - 1 × MCCAB Arduino Training Board
  - 1 × Arduino Nano

  https://elektor.fr/20440

LIENS

[1] Elektor Arduino Nano Training Board MCCAB and Companion Guide: https://elektor.fr/20440
[2] Manuel d'utilisation de la carte MCCAB d'Elektor : https://elektor.fr/20440

Source : Shutterstock / Metamorworks

sur le vif

luddisme moderne

Ilse Joostens (Belgique)

Il est vrai que les performances du matériel informatique sont en augmentation constante, ce qui permet également de développer des applications logicielles toujours plus intelligentes. Lorsque je travaillais comme administrateur système il y a une vingtaine d'années, j'étais passionné par les nouvelles technologies et ma mission personnelle consistait à motiver les utilisateurs habituels d'ordinateurs pour les nouvelles technologies. Cela n'a pas toujours été facile, c'est le moins que l'on puisse dire. Maintenant que je suis un peu plus âgé, et surtout plus sage, mon enthousiasme pour les nouvelles technologies s'est un peu refroidi, et j'ai un sentiment de malaise face aux récents développements de l'IA.

L'absurdité artificielle

À moins d'avoir vécu dans une grotte ces derniers mois (sans Internet), vous avez certainement entendu parler de la nouvelle (r)évolution technologique de l'intelligence artificielle, et plus particulièrement du chatbot d'OpenAI, ChatGPT [1]. L'intérêt est énorme, et Microsoft s'engage pleinement dans l'IA générative avec un investissement dans cette start-up à hauteur de 10 millions de dollars. Comme le veut la tradition, son rival, Google, ne veut pas rester à la traîne et travaille actuellement sur un bot similaire portant le nom mélodieux de *Bard*. Même Baidu entre dans la danse et a commencé à développer une version chinoise de ChatGPT appelée *Ernie*.

En tant que photographe amateur, j'étais déjà intéressé par DALL-E [2], StabilityAI [3] et Fotor. Ces applications permettent de convertir du texte en images, et de modifier ou combiner des images existantes. C'est devenu un peu une addiction, et j'ai passé des heures à les expérimenter, jusqu'à ce que de la fumée sorte de mon clavier (**figure 1**). Les résultats allaient de décevants à spectaculaires, et parfois même effrayants. En ce qui concerne ce dernier point, ma préférence pour le genre horreur peut offrir une explication sous-jacente. Cette technologie est également au cœur de la *creepypasta* autour de « Loab » [4][5], une femme démoniaque qui se cacherait au plus profond des cavernes de l'IA.

Naturellement, par curiosité, j'ai également essayé ChatGPT et, à première vue, le bot m'a rappelé « Evarist the computer », un personnage de la série flamande pour la jeunesse Merlina [6], à qui l'on pouvait demander n'importe quoi. Les réponses à mes questions étaient correctes d'un point de vue linguistique, mais, contrairement à Evarist, ChatGPT manquait souvent sa cible et certaines réponses étaient extrêmement hilarantes, dans le style de Xavier De Baere (un célèbre humoriste de la télévision flamande – NDLR). Bien que ce ne soit pas l'objectif initial, il semble que le robot puisse également générer du code source. Un simple projet Arduino n'a pas posé de problème (**figure 2**), mais une demande plus complexe d'écrire un programme C pour un AVR, afin d'initialiser un CI WIZnet W5500 Ethernet avec une adresse IP fixe, sans utiliser de bibliothèques, est restée bloquée à chaque fois. Peut-être que la version payante avec le GPT-4 d'OpenAI ferait mieux, mais

Figure 1. Création avec IA par Ilse Joostens.

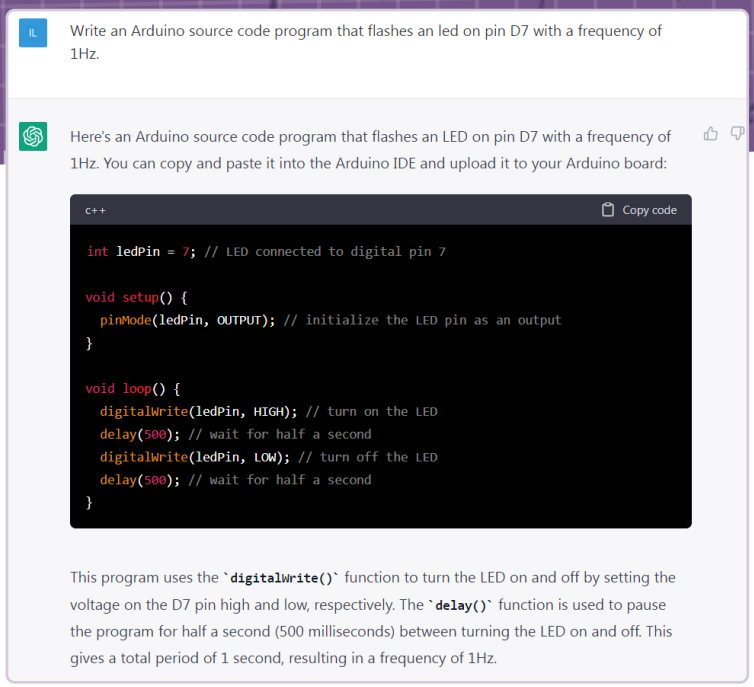


Figure 2. ChatGPT écrit du code...

ce n'est pas une possibilité pour une application pour laquelle je n'ai pas (encore) d'utilisation spécifique. Je peux faire fonctionner l'appareil moi-même, mais les questions les plus ordinaires de la vie quotidienne, comme la façon de préparer un dîner végétarien à trois plats avec un dessert au chocolat ou des idées de bricolage pour les tout-petits, ne m'intéressent pas particulièrement.

Tout comme « cloud » est un terme à la mode pour travailler et stocker des données via internet, et que le mot à la mode « IdO » désigne des appareils tels que des lampes « intelligentes » connectés à internet, « intelligence artificielle » est un terme global pour désigner des algorithmes et des modèles puissants liés au *big data*. Dans le cas de GPT-3, il s'agit de 570 Go d'entrée, tandis que DALL-E 2 se contente de plus de 650 millions d'images. Le terme « intelligence » n'est pas vraiment approprié, et les chercheurs qui entraînent les réseaux neuronaux à rechercher des connexions qui n'existent pas réellement créent en fait une « absurdité artificielle », si l'on peut l'exprimer ainsi [7].

Le côté obscur de l'IA

J'admets que l'IA générative inspire et apporte de nouvelles perspectives, mais je ne suis pas un grand fan, étant donné que les textes et les images générés sont essentiellement du plagiat parce qu'ils sont basés sur le travail de millions de personnes. Par ailleurs, toute IA formée sur la base de données historiques, qui contiennent inévitablement des préjugés humains, reprend ces partis pris, parfois sous une forme amplifiée. Je ne pense pas que ChatGPT écrira cette chronique dans le futur, mais les choses sont déjà assez difficiles pour les ventes en ligne de photographies et d'œuvres d'art. Avec l'amélioration constante des appareils photo, des smartphones, c'est un nouveau revers pour les photographes amateurs et les artistes qui cherchent à gagner un peu plus d'argent [8].

Il existe également de nombreuses possibilités d'utilisation abusive, allant des courriels d'hameçonnage parfaits aux discours haineux et aux contrefaçons graves. Grâce à ChatGPT, l'IA est aujourd'hui un sujet d'actualité, mais elle existe depuis bien plus longtemps et peut également être utilisée – ou détournée - à d'autres fins, par exemple par des agences gouvernementales comme en Chine, avec leur système de crédit social, ou aux États-Unis où la technologie de reconnaissance faciale est utilisée pour identifier les manifestants. En Flandre également, l'IA est appliquée aux photos aériennes pour déterminer si quelqu'un a coupé plus d'arbres dans son jardin que ce qui était autorisé. Personnellement, j'ai été plusieurs fois victime d'un délit dans le passé, mais on n'a jamais cherché à retrouver les auteurs. Par contre, tous les moyens sont bons pour constater les petits délits commis par de bons citoyens, avec, comme triste fait marquant, le cas d'un habitant de Heusden-Zolder qui rentrait d'un restaurant avec sa famille, et s'est brièvement arrêté pour jeter une couche pleine dans une poubelle publique venant de sa fille malade. À sa grande surprise, il a reçu un peu plus tard par la poste un rapport de police de vingt pages (!), photos à l'appui, pour dépôt illégal [9]. Apparemment, l'enfermement des vrais criminels ne fait pas rentrer assez d'argent dans les caisses.

C'est plus qu'il n'en faut pour faire de vous un luddite moderne.

VF : Laurent Rauber — 230196-04

LIENS

[1] ChatGPT : https://chat.openai.com
[2] DALL·E 2 : https://labs.openai.com
[3] StabilityAI's Stable Diffisuion : https://huggingface.co/spaces/stabilityai/stable-diffusion
[4] Loab, an AI-generated entity : https://loab.ai
[5] YouTube : The Disturbing Art of AI (spooky!): https://youtu.be/i9InAbpM7mU
[6] Merlina: Remembering the hit BRT youth series from the 1980s [en néerlandais]: https://merlina.info/evarist/uitvindingen
[7] EOS Wetenschap: Kunstmatige Domheid [Artificial Stupidity, en néerlandais]: https://eoswetenschap.eu/technologie/kunstmatige-domheid
[8] YouTube: AI Art Apocalypse : https://youtu.be/HDbu2rvhCk4
[9] Nieuwsblad: Heusdens gezin riskeert GAS-boete voor weggooien van pamper [en néerlandais] : https://nieuwsblad.be/cnt/dmf20220901_96655325

B.a.ba capteur : le capteur de température DS18B20

connexion au bus 1-Wire

Mathias Claussen (Allemagne)

Inclus dans de nombreux kits pour débutants, le capteur DS18B20 de Dallas Semiconductor facilite la découverte de la mesure de température. Avant de se lancer, il vaut mieux connaître un peu le bus 1-Wire et la façon d'y connecter le capteur. Nous allons rapidement explorer ici les bases de l'utilisation du DS18B20, pour vous permettre de débuter sereinement dans la mesure de température !

Si vous souhaitez enregistrer des températures à l'aide d'un microcontrôleur, vous avez le choix entre plusieurs capteurs et systèmes de bus. L'un des plus répandu est le DS18B20 de Dallas Semiconductor. Sa plage de mesure s'étend de -5 °C à 125 °C (-67° à +275 °F) avec une précision impressionnante de ±0,5 °C. Sa polyvalence lui permet de mesurer non seulement les températures ambiantes, mais aussi de surveiller les congélateurs et les chambres froides. Dans cet article, nous examinons de plus près son fonctionnement avec un microcontrôleur, avec des exemples comprenant le code source et les schémas de câblage pour montrer comment l'intégrer dans vos propres projets.

Le DS18B20

Le capteur DS18B20 utilise le système de bus 1-Wire, breveté par Dallas Semiconductor en 1989 et qui permet la connexion de plusieurs capteurs. Avec sa plage d'alimentation de 3,0 V à 5,5 V, on peut connecter directement le DS18B20 aux GPIO de nombreux appareils, de l'Arduino UNO au Raspberry Pi Pico. Le bus 1-Wire offre également un mode d'alimentation parasite, où l'énergie est prélevée sur la ligne de données qui alimente le capteur. Cela signifie que le bus 1-Wire ne nécessite qu'une seule broche sur un microcontrôleur (MCU) pour connecter plusieurs capteurs, ce qui justifie son choix, en particulier sur les cartes MCU les plus petites avec peu de GPIO. Bien qu'il existe d'autres capteurs 1-Wire, nous nous concentrerons ici sur le dispositif Dallas.

Figure 1. Le DS18B20 dans un boîtier TO-92.

Figure 2. Version étanche du DS18B20 avec câble.

Figure 3. Brochage du DS18B20.

Le capteur DS18B20 est disponible sous différentes formes, comme le montrent les **figures 1** et **2**. Grâce au bus 1-Wire, il est désormais très facile de connecter un plus grand nombre de capteurs. Bien que le protocole 1-Wire lui-même n'impose pas de limite au nombre de capteurs pouvant être connectés au bus, les limites sont déterminées par les propriétés électriques du bus.

Le bus 1-Wire

Pour connecter des capteurs au bus 1-Wire, trois fils sont nécessaires : la masse (GND), les données (DQ) et la tension d'alimentation (VCC). Le bus 1-Wire est bidirectionnel et fonctionne selon le concept contrôleur/cible (maître et esclave), où le contrôleur et la cible échangent des données via la ligne de données. Le brochage du capteur DS18B20 est présenté à la **figure 3**.

Tous les nœuds sur l'unique ligne de données utilisent un pilote à collecteur ouvert ; en cas de collision de données, lorsque deux nœuds essaient par erreur de parler en même temps, il en résulte simplement des données corrompues. Il ne peut jamais y avoir de conflit matériel lorsqu'un nœud essaie de forcer la ligne de données vers le haut alors qu'un autre essaie de la forcer vers le bas. Une résistance de rappel est nécessaire parce qu'aucun des nœuds ne peut forcer activement le bus vers VCC. La **figure 4** montre la connexion d'un capteur 1-Wire à un Arduino UNO.

Outre la connexion électrique, il faut un protocole pour le fonctionnement des appareils 1-Wire. Heureusement, la plupart des microcontrôleurs intègrent un UART qui peut être utilisé pour cela, sans nécessiter de matériel particulier. La **figure 5** présente un circuit approprié.

Pour la commande via UART, Analog Devices propose un article utile [1]. Alors que l'UART peut aider à réduire la charge du CPU, il n'est pas essentiel avec les MCU d'aujourd'hui. Même un petit ATtiny est assez puissant pour adresser les capteurs connectés via le bus 1-Wire grâce à une solution purement logicielle, ne nécessitant qu'une broche d'E/S. La plupart des bibliothèques qui fonctionnent avec le DS18B20 n'utilisent qu'un seul GPIO, ce qui rend l'utilisation d'un UART plutôt rare.

Identification du capteur

Lorsque plusieurs capteurs sont connectés à un fil de bus commun, chacun d'eux doit pouvoir être adressé individuellement afin qu'un seul capteur à la fois soit autorisé à écrire sur le bus et éviter que ses données ne soient brouillées ou corrompues par un autre capteur.

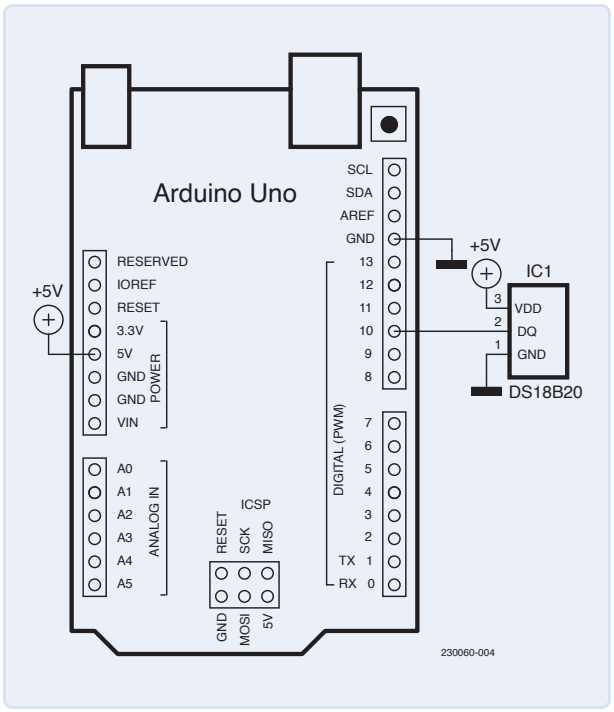


Figure 4. Schéma du circuit Arduino UNO avec un DS18B20.

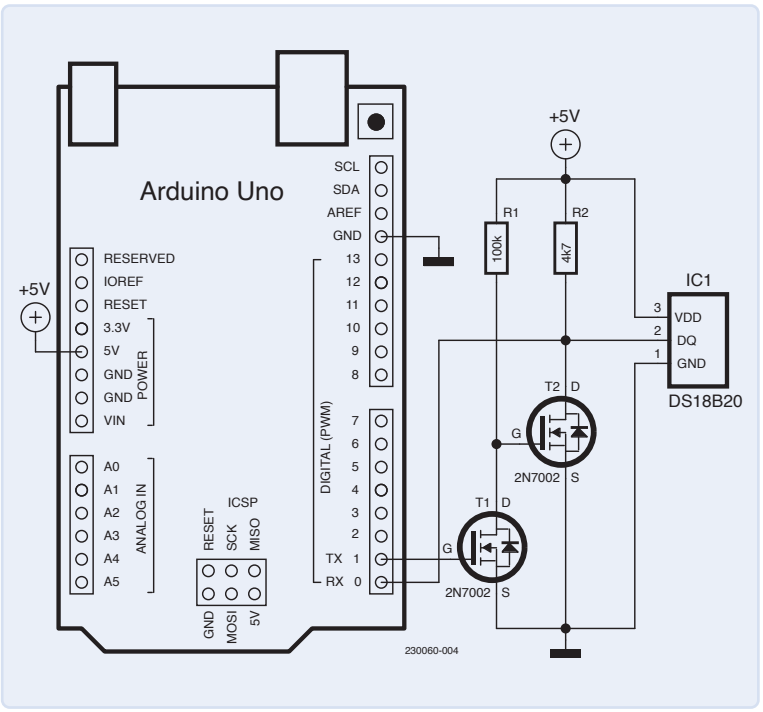


Figure 5. Schéma du bus 1-Wire du DS18B20 utilisant l'UART.

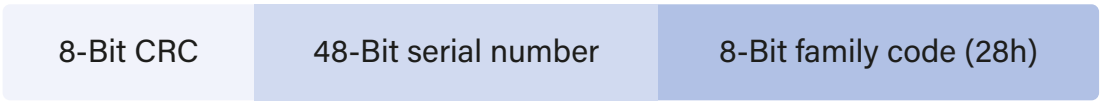


Figure 6. Format du message UUID du DS18B20.

Pour cela, chaque participant au bus 1-Wire dispose d'un code d'identification unique, d'une largeur de 64 bits (UUID). Il se compose d'un code sur 8 bits indiquant la famille du capteur, d'un numéro de série sur 48 bits et d'une somme de contrôle sur 8 bits (CRC) (**figure 6**). Le numéro de série ne figure toutefois nulle part sur le capteur, mais réside en interne, stocké sur 8 octets en ROM. Il peut être récupéré par logiciel à l'aide d'une fonction de recherche sur le bus 1-Wire. On peut trouver l'algorithme de recherche sur la page d'Analog Devices [2].

Exemple de code pour Arduino

À partir du schéma de la **figure 4**, la configuration reste très simple. Le capteur utilise une alimentation de 5 V comme l'Arduino UNO. Le GPIO de l'Arduino UNO, connecté à la ligne de données ou DQ, nécessite une résistance de rappel de 4,7 kΩ. La bibliothèque pour bus 1-Wire que nous utilisons ici est la bibliothèque *OneWire* de Paul Stoffregen [3]. Le code d'exemple de la bibliothèque montre aussi comment lire les données des capteurs DS18B20 (**listage 1**), ce que nous pouvons maintenant examiner plus en détail.

Pour utiliser la bibliothèque, il faut d'abord l'inclure avec `#include <OneWire.h>` au début du croquis. Ensuite, avec `OneWire ds(10)`, on affecte la broche 10 au fonctionnement de la ligne de données du bus 1-Wire. Avec `ds.search(addr)`, on recherche le capteur suivant sur le bus. La recherche est terminée lorsque tous les capteurs ont été identifiés. Les identifications de tous les capteurs trouvés sont stockées dans `addr`.

Maintenant que le capteur a été identifié, on peut interagir avec lui. Le premier octet, `addr[0]`, contient le code indiquant la famille. Il peut servir à déterminer si un capteur de température DS18B20, DS18S20 ou DS1822 a été trouvé, ou s'il s'agit d'un autre type de dispositif de bus 1-Wire.

La séquence suivante apparaît à la ligne 69 :

```
ds.reset();
ds.select(addr);
ds.write(0x44, 1);
```

Une réinitialisation de tous les participants au bus est effectuée une fois à l'aide de `ds.reset()`. Ensuite, en utilisant `ds.select(addr)`, le capteur découvert est adressé. Pour demander une lecture de température au capteur, il faut d'abord lancer une mesure. La commande correspondante est `ds.write(0x44, 1)`, où 0x44 est la commande reconnue par le capteur. Avec le deuxième paramètre (ici, `1`), la broche d'E/S est forcée au niveau haut pour alimenter les capteurs tels que le DS18S20 en mode parasite. Le résultat de la mesure de température est disponible après 750 à 1000 ms. Dans cet exemple, on utilise un délai pour attendre la fin du traitement. Lorsque la mesure de la température est terminée et que le résultat est prêt, on peut le récupérer. Le capteur est à nouveau adressé à l'aide de `ds.select(addr)` et on lui demande de lire la valeur stockée dans sa mémoire de travail (`ds.write(0xBE)`). Les neuf octets stockés ici, comprenant la valeur de la température, vont maintenant être envoyés.

Avec le DS18B20, la température est disponible après lecture sous forme d'une valeur sur 16 bits, composée de deux octets de registre. Si on utilise un capteur de type DS18S20 au lieu d'un DS18B20, les valeurs de registre ont un poids différent, si bien que leur position dans le registre doit être modifiée. L'exemple de code en tient compte et effectue les 3 décalages vers la gauche nécessaires sur la valeur brute.

Le DS18B20 fournit la température avec une résolution de 12 bits et prend 750 ms pour la conversion. Il peut, au choix, fournir des valeurs sur 11, 10 ou 9 bits, ce qui permet d'obtenir un temps de conversion plus rapide. Avec une résolution de 9 bits, la valeur des bits de poids faible n'est pas définie, ils peuvent donc être mis à zéro. La température relevée par le capteur est toujours exprimée en degrés Celsius ; votre logiciel devra effectuer la conversion en degrés Fahrenheit si nécessaire.

DS18B20 - La guerre des clones

Le DS18B20 est un capteur répandu, souvent inclus dans divers kits de capteurs et kits de début pour microcontrôleurs. Ces capteurs sont aussi très bon marché ; vous pouvez en trouver un pour seulement 30 centimes sur certains sites de vente en ligne. Les distributeurs tels que Mouser ou Farnell vous feront cependant payer plus de 4 euros pour un seul DS18B20.

La popularité de ce capteur a incité certaines usines malhonnêtes de fabrication de puces à produire leurs propres versions, qui ressemblent au capteur original et semblent se comporter comme lui, mais qui peuvent s'écarter considérablement des caractéristiques de l'original après un examen plus approfondi. Les distributeurs reconnus s'approvisionnent en composants auprès de fournisseurs certifiés.

Pour vous assurer que vous disposez d'un capteur authentique, la page GitHub de Chris Petrich [4] fournit des croquis Arduino pour tester votre capteur, ainsi que de plus amples informations sur chaque clone particulier et ses déviations par rapport à la conception originale.

Résumé

Il est facile de connecter un capteur de température à un microcontrôleur avec le DS18B20 et 1-Wire. Comme nous l'avons vu, en utilisant la plateforme Arduino, il y a tout un tas de bibliothèques

et d'exemples de code source pour alléger le processus. Avec cette configuration, une seule broche du MCU est nécessaire pour communiquer avec le capteur, et plusieurs capteurs peuvent être connectés, ce qui permet à un petit MCU avec un GPIO libre de prendre en charge plusieurs capteurs simultanément.

VF : Denis Lafourcade — 230060-04

Des questions, des commentaires ?

Contactez Elektor (redaction@elektor.fr).

Produits

- **Kit capteur Elektor 37-en-1 (SKU 16843)**
  https://elektor.fr/16843
- **Cytron Maker UNO (SKU 18634)**
  https://elektor.fr/18634
- **Kit de développement MakePython ESP32 (SKU 20137)**
  https://elektor.fr/20137

LIENS

[1] Using a UART to implement a 1-wire-bus-master, Analog Devices : https://analog.com/en/technical-articles/using-a-uart-to-implement-a-1wire-bus-master.html
[2] 1-Wire search algorithm, Analog Devices : https://analog.com/en/app-notes/1wire-search-algorithm.html
[3] OneWireLibrary par Paul Stoffregen sur GitHub : https://github.com/PaulStoffregen/OneWire
[4] Chris Petrich, "Your DS18B20 temperature sensor is likely a fake, counterfeit, clone...," GitHub : https://github.com/cpetrich/counterfeit_DS18B20

Listage 1. Exemple DS18B20 avec Arduino.

```
001        #include <OneWire.h>
002
003        // Exemple de température avec DS18S20, DS18B20, DS1822 OneWire
004        //
005        // http://www.pjrc.com/teensy/td_libs_OneWire.html
006        //
007        // La bibliothèque DallasTemperature peut faire tout ce travail pour vous !
008        // https://github.com/milesburton/Arduino-Temperature-Control-Library
009
010        OneWire ds(10);  sur la broche 10 (une résistance de 4k7 est nécessaire)
011
012        void setup(void) {
013          Serial.begin(9600);
014        }
015
016        void loop(void) {
017
018          byte i;
019          byte present = 0;
020          byte type_s;
021          byte data[9];
022          byte addr[8];
023          float celsius, fahrenheit;
024
025          if (!ds.search(addr)) {
026            Serial.println("No more addresses.");
027            Serial.println();
```

suite à la page suivante

```
028           ds.reset_search();
029           delay(250);
030           return;
031         }
032
033
034         Serial.print("ROM =");
035         for(i = 0; i < 8; i++) {
036           Serial.write(' ');
037           Serial.print(addr[i], HEX);
038         }
039
040         if (OneWire::crc8(addr, 7) != addr[7]) {
041             Serial.println("CRC is not valid!");
042             return;
043         }
044
045         Serial.println();
046         // le premier octet en ROM identifie le type
047         switch (addr[0]) {
048
049           case 0x10:
050             Serial.println("  Chip = DS18S20");  // ou l'ancienne DS1820
051             type_s = 1;
052             break;
053
054           case 0x28:
055             Serial.println("  Chip = DS18B20");
056             type_s = 0;
057             break;
058
059           case 0x22:
060             Serial.println("  Chip = DS1822");
061             type_s = 0;
062             break;
063
064           default:
065             Serial.println("Device is not a DS18x20 family device.");
066             return;
067         }
068
069         ds.reset();
070         ds.select(addr);
071        ds.write(0x44, 1); // début de conversion, avec alimentation en mode parasite à la fin
072         delay(1000);       // peut-être que 750ms suffisent, peut-être pas
073         // nous pourrions faire un ds.depower() ici, mais le reset s'en chargera.
074
075         present = ds.reset();
076         ds.select(addr);
077         ds.write(0xBE);         // Lecture de la mémoire de travail
078
079         Serial.print("  Data = ");
080         Serial.print(present, HEX);
081         Serial.print(" ");
082         for (i = 0; i < 9; i++) {           // nous avons besoin de 9 octets
083           data[i] = ds.read();
084           Serial.print(data[i], HEX);
085           Serial.print(" ");
086         }
087         Serial.print(" CRC=");
088         Serial.print(OneWire::crc8(data, 8), HEX);
```

```
089         Serial.println();
090         // Conversion des données en température réelle
091         // comme le résultat est un entier signé sur 16 bits, il doit
092         // être stocké dans un type "int16_t", qui est toujours sur 16 bits,
093         // même lorsqu'il est compilé sur un processeur 32 bits.
094
095         int16_t raw = (data[1] << 8) | data[0];
096         if (type_s) {
097           raw = raw << 3; // résolution 9 bits par défaut
098           if (data[7] == 0x10) {
099             // le résidu donne une résolution complète de 12 bits
100             raw = (raw & 0xFFF0) + 12 - data[6];
101           }
102         } else {
103           byte cfg = (data[4] & 0x60);
104          // à une résolution plus faible, les bits de poids faible sont indéfinis, alors mettons-les à zéro
105           if (cfg == 0x00) raw = raw & ~7;  // 9 bit resolution, 93.75 ms
106           else if (cfg == 0x20) raw = raw & ~3; // 10 bit res, 187.5 ms
107           else if (cfg == 0x40) raw = raw & ~1; // 11 bit res, 375 ms
108           // la résolution par défaut est 12 bits, le temps de conversion est de 750 ms
109         }
110
111         celsius = (float)raw / 16.0;
112         fahrenheit = celsius * 1.8 + 32.0;
113         Serial.print("  Temperature = ");
114         Serial.print(celsius);
115         Serial.print(" Celsius, ");
116         Serial.print(fahrenheit);
117         Serial.println(" Fahrenheit");
118       }
```

quels standards pour unifier la domotique ?

Matter et Thread se distinguent

Stuart Cording (Elektor)

L'index tapote l'écran, et voilà, les pièces sont éclairées. Enfin, presque toutes : entre applis trop nombreuses, incompatibilités matérielles et standards différents, la maison est parfois moins intelligente qu'annoncé. La solution ? D'autres standards ! Mais *Matter* et *Thread* y suffiront-ils ?

Vous voici enfin à la maison après une journée difficile. Le système audio vous accueille avec votre musique préférée, les pièces ont la température idéale, et votre four peaufine la cuisson de votre repas. Les stores s'empressent de tamiser l'ardeur du soleil lorsque vous entrez dans le salon. Vous vous enfoncez dans le moelleux de votre fauteuil, et aussitôt une lampe sur pied vous baigne d'une lumière idéale pour la lecture. Vous saisissez votre liseuse, qui bien sûr s'ouvre à la bonne page.

Ah, le rêve d'une maison intelligente ! Il existe pour le concrétiser une foule de protocoles, adaptés au sans-fil ou à la connexion filaire, chacun conçu pour un domaine particulier, avec ses propres applications et caractéristiques. Le fantasme d'une maison automatisée ne date pas d'hier. À la fin des années 1990, Microsoft a publié une vidéo mettant en scène une grande partie de la technologie disponible aujourd'hui. On y voit des serrures électroniques, des achats sur Internet, des assistants vocaux, et bien d'autres choses encore, le tout piloté par Windows XP et CE [1]. En 1950, Ray Bradbury avait lui aussi entrevu ce que pourrait être une maison automatisée dans la nouvelle *Viendront de douces pluies* [2]. Si j'évoque le passé, c'est qu'avant même de parler des protocoles *Matter* et *Thread*, il nous faut comprendre d'où vient la complexité actuelle.

Naissance du sans-fil à courte portée

Lorsque l'idée d'un Internet des Objets a pris forme, le sans-fil s'est imposé comme une évidence pour sa connexion. Mais lequel ? Le début des années 2000 ne laissait guère de choix : wifi, modem cellulaire GSM, ou Bluetooth. Les deux premiers étaient énergivores, et donc plutôt réservés aux dispositifs alimentés par batterie. Restait Bluetooth, une technologie adaptée aux besoins du marché naissant des téléphones mobiles et bon candidat au remplacement d'IrDA, la liaison infrarouge utilisée par les assistants numériques personnels (PDA) pour la synchronisation des données.

Parce qu'il opérait dans une bande ISM (Industrielle, Scientifique et Médicale) centrée autour de 2,4 GHz et régie par des directives internationales, Bluetooth semblait idéal pour l'IdO. Le protocole était déjà au service de passerelles et ports série, et même d'imprimantes – à l'aide d'adaptateurs. Il n'était pas encore pris en charge par Windows, mais on pouvait faire communiquer un PC et un téléphone à l'aide d'un *dongle* et d'un logiciel.

Côté audio, Bluetooth fournissait une interface robuste aux systèmes de téléphonie des automobiles. Ces dispositifs plutôt encombrants appartiennent désormais au passé, puisque de nos jours il suffit d'apparier nos smartphones avec un kit mains libres intégré à nos véhicules.

L'architecture de Bluetooth est par essence celle d'un réseau personnel (PAN). La spécification permet à sept appareils esclaves de se connecter à un maître pour former un picoréseau, ou *piconet* (**figure 1**). Elle autorisait aussi la création d'un *scatternet* (réseau dispersé) formé par plusieurs *piconets* (**figure 2**). Le débit théorique, de seulement 720 kbit/s à l'époque, chutait toutefois de façon significative à mesure que des *piconets* étaient ajoutés, et aussi parce que le maître devait commuter rapidement d'un esclave à l'autre pour donner l'illusion d'un fonctionnement instantané. La façon dont les *piconets* transféraient leurs données était en outre laissée à celui qui implantait le réseau. J'ai participé à plusieurs projets de *scatternets*, et aucun n'a semblé pouvoir être mis en pratique.

Wibree l'économe

La nécessité de maintenir l'émetteur-récepteur sous tension rendait Bluetooth inadapté aux capteurs alimentés par batterie. Pour y remédier, Nokia développa *Wibree*, une technologie opérant elle aussi

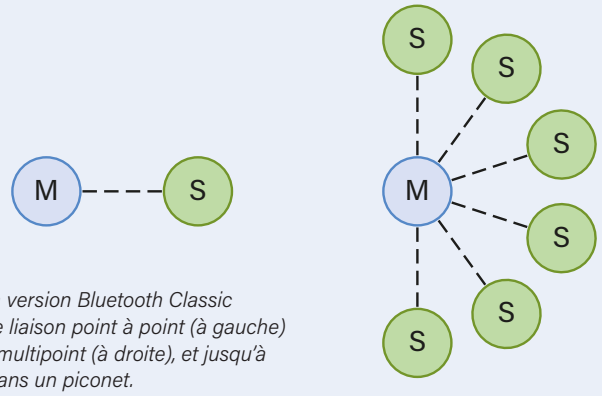


Figure 1. La version Bluetooth Classic permet une liaison point à point (à gauche) ou point à multipoint (à droite), et jusqu'à 8 nœuds dans un piconet.

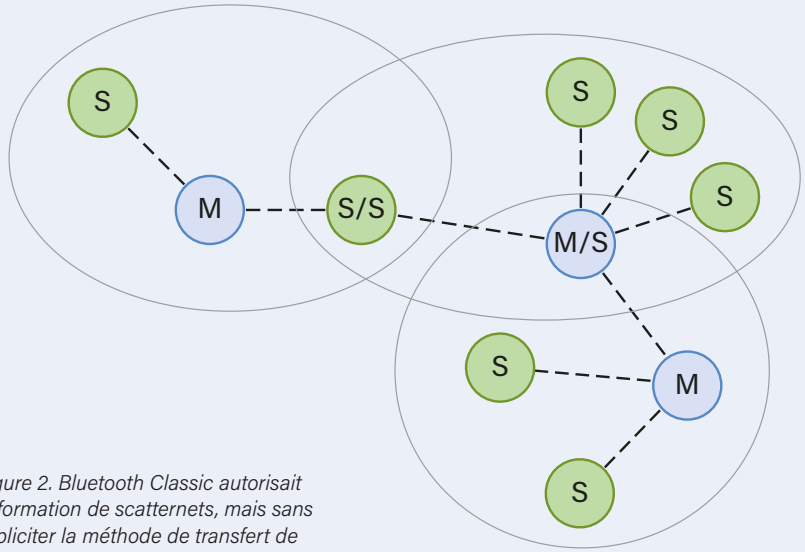


Figure 2. Bluetooth Classic autorisait la formation de scatternets, mais sans expliciter la méthode de transfert de données entre les piconets.

Figure 3. Un exemple d'objet connecté reposant sur Zigbee : une ampoule Trådfri d'Ikea.

sur la bande de 2,4 GHz [3]. Au milieu des années 2000 apparurent ainsi des puces hybrides Bluetooth/Wibree pour montres, tensiomètres, claviers sans fil, etc. Wibree fut adopté par Bluetooth SIG, l'organisme en charge de la spécification Bluetooth, et fut rebaptisé plus tard Bluetooth Low Energy (BLE). De nos jours les versions *Classic* et *BLE* résident sur la même puce, partagent antenne et émetteur-récepteur, ainsi que certains traits de leur pile logicielle. Ces versions restent toutefois incompatibles entre elles [4].

Malgré une consommation d'énergie adaptée aux capteurs IdO alimentés par pile, il aura fallu attendre 2017 pour être en mesure de créer un réseau maillé BLE.

Zigbee et Z-wave

D'autres standards pour l'IdO étaient disponibles. Ainsi Zigbee (**figure 3**), une technologie reposant sur la norme IEEE 802.15.4, apparue en 2003, révisée en 2006, œuvrant sur la bande de 2,4 GHz, mais pouvant aussi opérer sur 784 MHz (Chine), 868 MHz (Europe) et 915 MHz (Australie et États-Unis). Lancée également en 2003 mais par Zensys – rachetée plus tard par Silicon Labs - Z-wave opérait quant à elle dans la bande des 868 MHz.

Ces deux technologies offrent un mécanisme d'auto-réparation garantissant un chemin entre émetteur et récepteur en cas de défaillance d'un nœud du réseau. Le mécanisme agit en arrière-plan et n'implique pas l'utilisateur. Plusieurs milliers de produits actuels reposent sur Zigbee et Z-wave.

Plus de problèmes que de solutions

Ces technologies posent toutefois problème sur plusieurs points. D'abord, aucune ne se connecte nativement à l'internet. À l'exception d'un appareil Bluetooth capable d'interagir avec un service en nuage via un smartphone proche, la connexion à l'internet nécessite une passerelle matérielle reliée à un routeur. Ensuite, l'interopérabilité est inexistante. Vous pouvez opter pour Zigbee, et découvrir plus tard qu'un produit qui vous intéresse ne fonctionne qu'avec Z-Wave. Un outil comme Node-RED [5] peut arranger les choses, mais il ne convient guère à l'utilisateur lambda.

Autre point délicat, la commande des objets connectés : en général une seule application n'y suffit pas, et chacune a sa propre méthode de commande. Dernier écueil, la difficulté à faire dialoguer plusieurs appareils avec un même assistant vocal. *Alexa* (Amazon) en reconnaît plus que *Google Home*, et celui-ci plus qu'*HomeKit* (Apple) [6].

C'est pour répondre à certaines de ces préoccupations que l'industrie a fait ce qu'elle sait faire de mieux : introduire une nouvelle technologie, *Thread*.

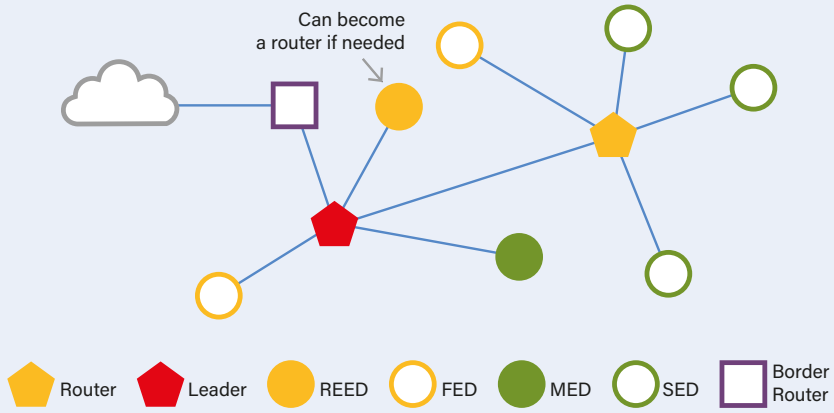


Figure 4. Thread repose sur le standard 6LoWPAN d'IPv6 et possède un mécanisme d'auto-réparation que n'ont pas ses concurrents.

Figure 5. Présentation de solutions Matter par Espressif lors du salon Embedded world 2023.

Figure 6. Présentation par Silicon Labs d'un cas pratique d'utilisation de Matter avec des réseaux Zigbee et Z-wave.

Thread ? Thread-t-on jamais…

Formée en 2014, l'alliance *Thread Group* s'était donnée pour objectif de créer une technologie de réseau sans fil à faible consommation et faible latence, capable de rivaliser avec les solutions de l'époque. Elle aussi opère sur 2,4 GHz, mais repose sur le protocole Internet (IP), plus précisément sur une version d'IPv6 appelée *6LoWPAN*. Cette version adapte le protocole IP aux besoins des appareils de faible puissance et aux réseaux sans fil IEEE 802.15.4 utilisés. Elle y ajoute des mécanismes inutiles à IPv6, comme la compression d'en-tête, la fragmentation et le réassemblage des paquets.

La sécurité est aussi plus robuste – et obligatoire. Comme dans tout réseau TCP/IP, les données sont cryptées de bout en bout, autrement dit tout dispositif servant de routeur ne peut pas lire les données le traversant. Enfin, le transfert de données se fait avec une latence deux fois moins moindre que celle de Zigbee, et sept fois moins élevée que celle de Bluetooth [7].

Nœuds Thread

Le nœud d'un réseau maillé Thread est un dispositif d'un des deux types suivants [8] : FTD (*Full Thread Device*) ou MTD (*Minimal Thread Device*). Un MTD ne peut être qu'un dispositif terminal (*End Device*, ED) et nécessite un nœud routeur pour accéder au réseau Thread. Un routeur peut prendre en charge 511 ED. Il existe deux types de MTD : le *Minimal End Device* (MED), dont l'émetteur-récepteur est toujours actif, et le *Sleepy End Device* (SED), dont l'émetteur-récepteur ne vérifie que périodiquement la présence de messages. Un nœud SED, p. ex. un capteur, est en général alimenté par pile.

Un FTD a son émetteur-récepteur toujours activé et possède deux variantes. La première est le *Router*. Il s'occupe des nouveaux nœuds qui rejoignent le réseau et achemine les paquets. Le premier nœud à prendre le rôle de Router est appelé *Leader*. Il prend en charge les nouveaux nœuds du réseau devenant Router. Il ne peut y avoir qu'un seul Leader et jusqu'à 32 Router. Si le Leader est défectueux ou supprimé, un autre Router prend le relais. Si le rôle de Router est inutile (car il y en a suffisamment dans le maillage), le FTD peut devenir REED (*Router Eligible End Device*). Il fonctionne comme un ED jusqu'à ce que sa fonction de routeur soit requise, auquel cas il est promu.

La seconde variante est le *Full End Device* (FED). Un FED agit comme un ED, mais garde trace de plus de données qu'un MED (comme la multidiffusion et le mappage des adresses IPv6). Ce profil convient aux appareils alimentés par le secteur.

Pour relier un réseau Thread à des services en nuage, un ou plusieurs nœuds dits routeurs de bordure (Border Router) sont nécessaires. Leur rôle est de relier les autres nœuds à un réseau local par Ethernet ou wifi (**figure 4**). Ils pourraient être intégrés à des appareils tels que des TV intelligentes, des routeurs wifi ou des assistants vocaux déjà reliés à l'internet.

L'atout d'un réseau Thread est que ses nœuds communiquent par IP sécurisé, là où Zigbee, Z-Wave et Bluetooth doivent

traduire toute communication internet en leurs protocoles propriétaires. Autre avantage, la couche d'application, celle sur laquelle s'appuient en général les applis et les interfaces web, est agnostique. L'interopérabilité entre appareils compatibles avec Thread et assistants domotiques devrait donc s'améliorer.

Thread prend également en charge une (autre) nouvelle technologie : *Matter*.

Matter, ou la promesse de l'interopérabilité

Matter a été initié fin 2019 par un groupe de travail appelé CHIP (*Connected Home over IP Project*). Un des membres en est l'alliance CSA (*Connectivity Standards Alliance*) [9], anciennement alliance Zigbee. Son rôle (parmi d'autres) est de délivrer une certification « Matter ». Le standard Matter repose sur la couche de transport TCP/IP, et peut donc également opérer avec Thread et n'importe quel appareil domotique Ethernet ou wifi.

Matter est une couche logicielle qui définit la façon dont les appareils communiquent entre eux afin d'autoriser pour leur commande n'importe quel système domotique ou assistant vocal. Parmi ses qualités figure *Multi-Admin*, une fonction qui permet d'utiliser différentes applications pour un même appareil. Une ampoule pourrait ainsi être commandée par un interrupteur mural, par l'appli fournie par le fabricant, et par l'assistant vocal auquel elle serait connectée. Un assistant vocal de Google devrait donc pouvoir commander des appareils d'Amazon ou d'Apple. Du moins en théorie. Car rien ne garantit que cette promesse de compatibilité sera tenue en pratique. Comme le note Simon Hill dans un article publié sur Wired, l'accès à certaines fonctions ou paramètres avancés pourrait encore nécessiter un appareil ou une application du même écosystème [10].

Plusieurs fabricants de puces, dont Espressif [11] et Silicon Labs [12], ont présenté des exemples pratiques de commande avec Matter, Thread et Wi-Fi (**figure 5** et **6**). La spécification Matter actuelle prend en charge la plupart des objets domotiques de base (ampoules, prises de courant, thermostats…) et quelques assistants vocaux. D'autres produits (appareils ménagers, caméras…) devraient figurer sur la prochaine spécification [13].

Qui l'emportera ?

Nous voici donc, après une décennie d'IdO pour la domotique, plongés au cœur d'une nouvelle bataille « VHS contre Betamax » [14], qui plus est de complexité croissante. Thread devrait jouer un rôle clé puisqu'il résout le problème de longue date de Zigbee, Z-Wave et Bluetooth Mesh, à savoir la nécessité d'une passerelle pour traduire le protocole IP en celui utilisé par ces réseaux sans fil. Avec Matter, ce ne seront plus quelques géants de l'électronique grand public qui décideront des applications et assistants vocaux à utiliser pour commander leurs produits, mais bien l'utilisateur.

Matter, c'est aussi la garantie qu'une installation ne deviendra pas obsolète du jour au lendemain. Certes, ce standard sert uniquement d'interface aux passerelles de Zigbee & Cie, et non aux réseaux eux-mêmes, mais cela devrait favoriser son succès, de même que sa prise en charge des appareils wifi et Ethernet. Les fabricants de biens coûteux et durables – comme les appareils électroménagers – savent que la quasi-totalité de leurs clients dispose du wifi, ce qui devrait leur faire préférer Matter. Autre argument en sa faveur, il est plus orienté vers l'utilisateur que les autres. Sera-t-il pour autant le standard de facto dans dix ans ? Seul l'avenir nous le dira.

VF : Hervé Moreau — 230226-04

Des questions, des commentaires ?

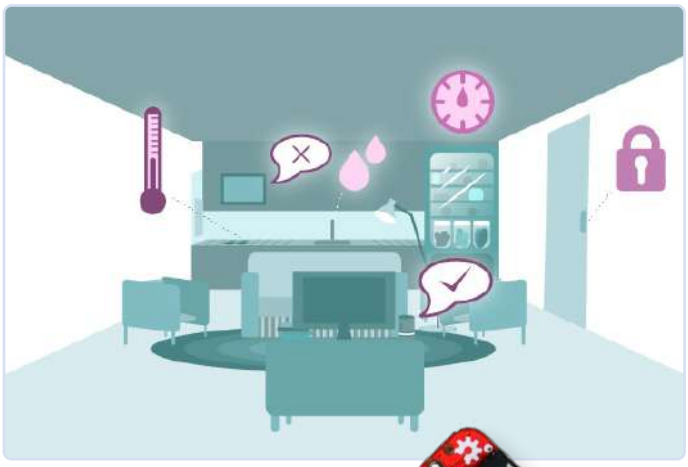
Envoyez un courriel à l'auteur (stuart.cording@elektor.com) ou contactez la rédaction d'Elektor (redaction@elektor.fr).

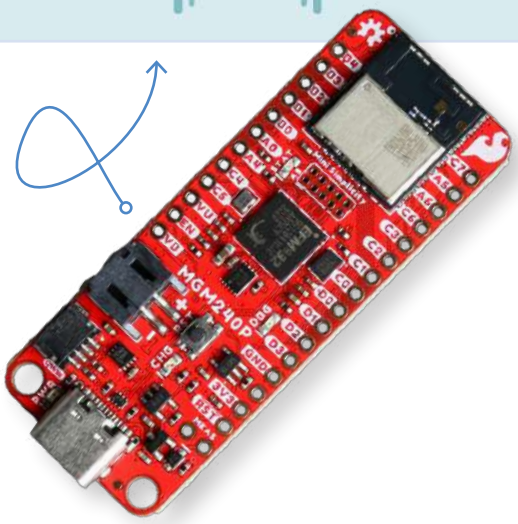
LIENS

[1] Microsoft Smart Home, 1999 [YouTube] : https://bit.ly/3zoDVCv
[2] Viendront de douces pluies [Wikipedia EN] : http://bit.ly/3ZR73x7
[3] E. Grabianowski, Is Wibree going to rival Bluetooth?, HowStuffWorks : http://bit.ly/3nHHTnb
[4] M. Afaneh, Bluetooth vs. Bluetooth Low Energy: What's the Difference?, avril 2022 : http://bit.ly/3KvmiaA
[5] R. Dullier, Control a Z-Wave plug using a Zigbee button!, mars 2021 : http://bit.ly/3nHRdHB
[6] M. Timothy, Amazon Alexa vs. Google Home vs. Apple HomeKit: What's the Best Smart Home System?, MakeUseOf, mars 2023 : http://bit.ly/40C3wDY
[7] Benchmarking Bluetooth Mesh, Thread, and Zigbee Network Performance, Silicon Labs : http://bit.ly/3Ge74UF
[8] Node Roles and Types, Google, février 2023 : http://bit.ly/3m5ZmVN
[9] Site de l'alliance CSA : http://bit.ly/3nF6qcm
[10] S. Hill, Here's What the 'Matter' Smart Home Standard Is All About, Wired, octobre 2022 : http://bit.ly/3zrpWf7
[11] Solutions Espressif pour Matter : http://bit.ly/3zsb545
[12] Vidéo de Silicon Labs : Matter over Wi-Fi and Thread Demo : http://bit.ly/3nFZ2gZ
[13] Vidéo de Silicon Labs : Matter over Wi-Fi and Thread Demo : http://bit.ly/410zh9D
[14] D. Owen, The Betamax vs VHS Format War, MediaCollege.com, mai 2005 : http://bit.ly/3U78JRD

Matter, ou la concorde des objets

Testez Matter avec la carte Thing Plus Matter et Simplicity Studio





Rob Reynolds (SparkFun)

Quel écosystème choisir pour un projet domotique ? Sempiternelle question, dont *Matter* entend être la réponse puisque ce nouveau protocole vise à unifier la communication entre tous les appareils de l'IdO. Découvrez son potentiel avec une application très simple créée avec la carte de développement *Thing Plus Matter* de Sparkfun et l'EDI *Simplicity* Studio de Silicon Labs.

Bien qu'il soit à vocation universaliste, l'Internet des Objets (IdO) reste aujourd'hui encore gouverné par de multiples protocoles de communication. Ce caractère disparate oblige les concepteurs, et par ricochet les utilisateurs, à opter pour l'un ou l'autre de ces protocoles et, par force, à s'y tenir. Cette contrainte est en passe d'être levée grâce au standard de connectivité open source *Matter*. Sa couche d'application permet aux développeurs et aux fabricants d'appareils de créer des écosystèmes fiables et sécurisés, et d'accroître la compatibilité entre les objets connectés d'un même système domotique.

Matter en bref

Le projet *Matter* a démarré en 2019 sous le nom *CHIP* (*Connected Home over IP Project*). Des acteurs majeurs tels qu'Amazon, Apple et Google, ainsi que l'alliance Zigbee et diverses entreprises comme Nordic Semiconductor, se sont réunis à l'époque pour élaborer un protocole de communication à même d'unifier l'Internet des Objets. *Matter*, le fruit de cette collaboration, est un protocole open-source, libre de droits, qui permet aux appareils de communiquer par wifi, Ethernet, Bluetooth Low Energy et Thread. Des appareils « certifiés Matter » pourront ainsi communiquer entre eux quelle que soit la technologie sans fil utilisée, autrement dit sans qu'il soit nécessaire de traduire un protocole en un autre.

Concrètement, concepteurs, fabricants et consommateurs n'auront plus à choisir entre des produits compatibles avec HomeKit (Apple), Alexa (Amazon) ou Weave (Google). Le fabricant voit son processus de conception simplifié, le consommateur y gagne en compatibilité.

Un des grands avantages de Matter est qu'il simplifie la configuration et la gestion d'un système domotique. Si ledit système comprend des appareils certifiés Matter, l'utilisateur pourra le configurer rapidement et facilement, sans avoir besoin de compétences techniques particulières. Et comme la sécurité est primordiale, le protocole prend en charge le cryptage de bout en bout, autrement dit sécurise la transmission de données entre appareils.

Autre avantage clé pour la plupart d'entre nous, Matter est open source. Tout ce qui a trait au protocole est disponible sur le dépôt GitHub de Matter [1] : code source, documentation, scripts,

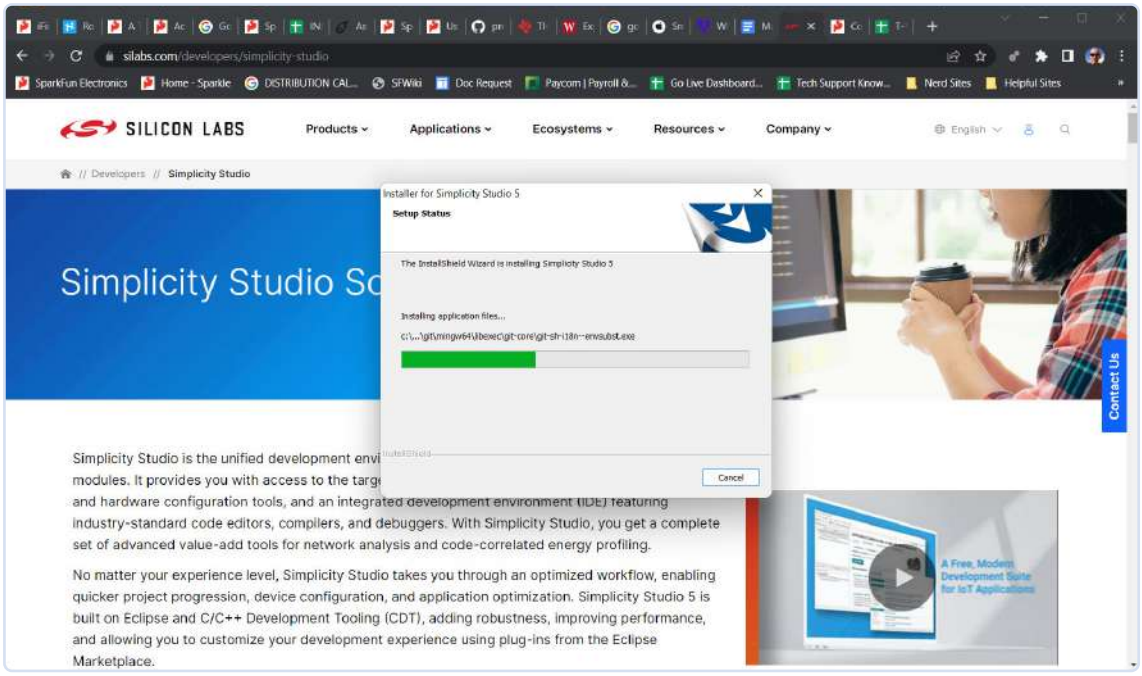


Figure 1. Installation de Simplicity Studio.

exemples, et plus généralement tout ce dont un concepteur a besoin pour créer un produit compatible avec Matter.
Tout cela est bien beau, mais bon nombre d'entre nous restent aujourd'hui encore relégués au rang de simple consommateur, sans possibilité de prototypage pour tester Matter, sinon en partant de zéro. Heureusement, les choses évoluent avec la sortie de la carte de développement *Thing Plus Matter* de *SparkFun Electronics* [2] – disponible dans l'e-choppe, cf. l'encadré **Produit**. Grâce à son écosystème Qwiic, *Thing Plus Matter* offre une méthodologie agile pour la conception et le prototypage de produits reposant sur Matter. Le module sans fil MGM240P de Silicon Labs fournit une connexion sécurisée pour les protocoles 802.15.4 à communication maillée (*Thread*) et Bluetooth Low Energy 5.3, et il s'intègre nativement au protocole Matter de Silicon Labs. Les cartes Thing Plus comprennent un connecteur Qwiic auquel peuvent être branchés sans soudure des circuits I²C de la famille *Qwiic Connect System*, et leur facteur de forme est compatible avec Feather.

Configuration de Simplicity Studio

Vous ferez vos premiers pas de développeur Matter avec l'enceinte *Nest Hub* de Google, mais avant cela il vous faut installer et configurer l'EDI *Simplicity Studio* de Silicon Labs. Je ne détaillerai pas toutes les étapes, reportez-vous au tutoriel de SparkFun [3] si vous avez besoin de compléments.
Rendez-vous sur le site de Silicon Labs [4] pour y télécharger Simplicity Studio (en version 5 au moment de la rédaction de cet article). Cliquez sur le bouton qui correspond à votre système d'exploitation. La page qui s'ouvre vous invite à vous connecter avec votre compte – créez-en un si vous n'en avez pas, c'est gratuit. Lancez l'installeur une fois le programme téléchargé (**fig. 1**).

Lorsque vous lancez Simplicity Studio pour la première fois, l'installeur cherche les mises à jour disponibles et, s'il y en a, vous les présente. Cliquez sur *Update All* pour les télécharger et les installer afin de disposer des derniers ajouts et améliorations. S'il n'y a aucune mise à jour supplémentaire, ou si votre système est configuré pour les faire automatiquement, l'installeur passe à l'étape suivante.
Après redémarrage (si celui-ci était nécessaire), l'installeur vous demande si vous souhaitez installer votre carte en la branchant, ou si vous souhaitez procéder à une installation par type de technologie (sans-fil, Xpress, microcontrôleur, capteurs). Sélectionnez *Install by connecting devices* (**fig. 2**), puis branchez votre carte *Thing Plus Matter* dans un port USB.
Le programme demande alors s'il doit installer les paquets requis. Cliquez sur Yes. Après installation, la fenêtre devrait indiquer *1 Device Found*, avec un identifiant tel que :

```
☑ J-Link (000449050174)(ID: 000449050174)
```

Cochez la case associée et cliquez sur *Next*. Deux options d'installation se présentent. L'option *Auto* installe pour nous tous les paquets nécessaires. L'option *Advanced* nous laisse choisir quels paquets nous souhaitons – mais si cette option vous semble le choix évident, il est fort probable que vous ne soyez pas en train de lire cet article. Sélectionnez donc Auto, puis *Next* (**fig. 3**). Acceptez les termes de la licence (*Master Software License Agreement*) pour que l'installation démarre. L'opération prend un certain temps et nécessitera un redémarrage une fois terminée. C'est généralement à cet instant que le développeur sérieux quitte son poste de travail et va se faire un café.

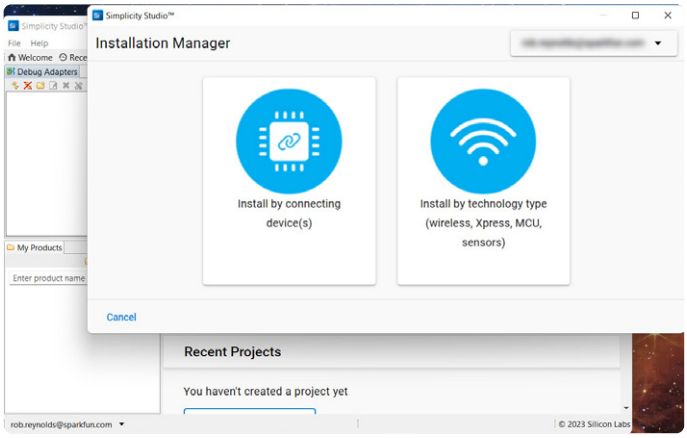


Figure 2. Installation par périphériques connectés.

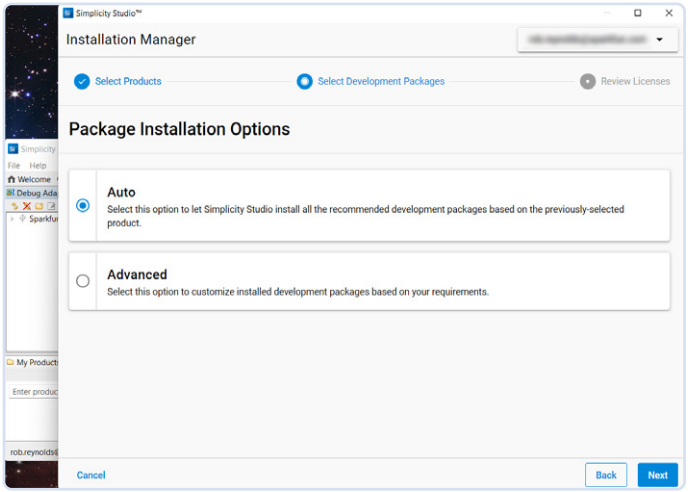


Figure 3. Les deux options d'installation des paquets.

Après redémarrage, Simplicity Studio vous invite cette fois-ci à accepter le contrat de licence d'utilisateur final, puis affiche – enfin ! – la fenêtre *Welcome to Simplicity Studio*. Vous devriez voir votre module MGM240P sous *Connected Devices*. Cliquez sur *Start* pour ouvrir la page d'information *Thing Plus* ; elle contient une présentation de la carte, des exemples et démos de projets, de la documentation, et divers outils. Ouvrez l'onglet *Example Projects and Demos*, entrez le mot-clé *Blink* dans la barre de recherche *Filter on keywords*, puis cliquez sur le bouton *Create* associé à *Platform - Blink Bare-metal* (**fig. 4**).

La fenêtre qui s'ouvre permet de changer le nom ou l'emplacement du projet. Donnez à votre projet un nom parlant, p. ex. *FirstBlink-Demo*, puis cliquez sur *Finish*. Une fois le projet compilé, Simplicity Studio affiche sur la gauche un explorateur appelé *Project Explorer*. Faites un clic droit sur le dossier principal du projet – qui devrait avoir pour nom *MatterBlinkExample* – puis sélectionnez *Run as/1 Silicon Labs ARM Program* (**fig. 5**).

Le menu *Run as* compile le script et l'écrit dans la mémoire de la carte. Vous devriez alors voir clignoter sa LED bleue toutes les demi-secondes. Ceci dit, il est fort probable que votre carte ait commencé à clignoter au moment même où vous l'aviez insérée dans le port USB de votre ordinateur. Une façon de s'assurer que le clignotement vient bien du code est d'en modifier l'intervalle dans le fichier *blink.c*. Ouvrez-le avec l'explorateur, et repérez la ligne ci-dessous, en théorie la 31 (**fig. 6**) :

```
#define TOOGLE_DELAY_MS 500
```

Remplacez 500 par une valeur qui ne laissera aucun doute sur l'origine du clignotement, p. ex. 100 pour un clignotement très rapide, ou 3000 pour un tempo beaucoup plus mesuré. Le résultat effacera tout doute. Pour le voir, faites un clic droit sur le dossier *MatterBlinkExample*, puis choisissez comme précédemment *Run as/1 Silicon Labs ARM Program*. Observez la fréquence de clignotement de la LED bleue, et concluez vous-même.

Pour aller plus loin : tutoriel Nest Hub

Félicitations, vous communiquez avec votre carte *Thing Plus Matter* depuis Simplicity Studio en utilisant le protocole Matter. Les choses intéressantes commencent maintenant. Être en mesure de dialoguer avec votre enceinte *Nest Hub* va en effet vous permettre d'intégrer à votre système domotique des dispositifs conçus par

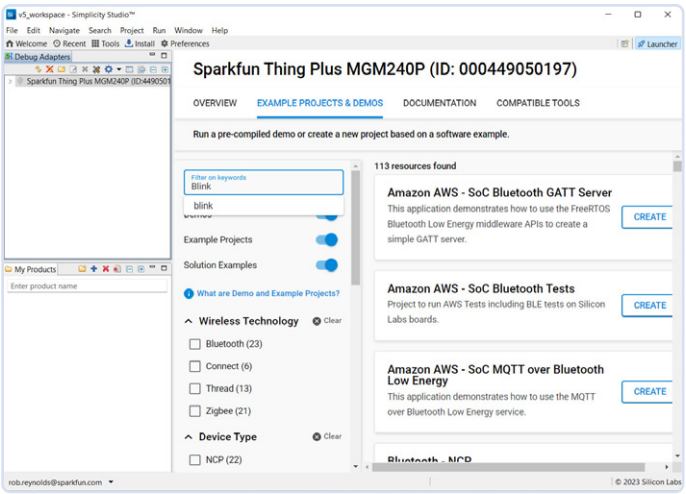


Figure 4. Onglets Example Projects et Demos.

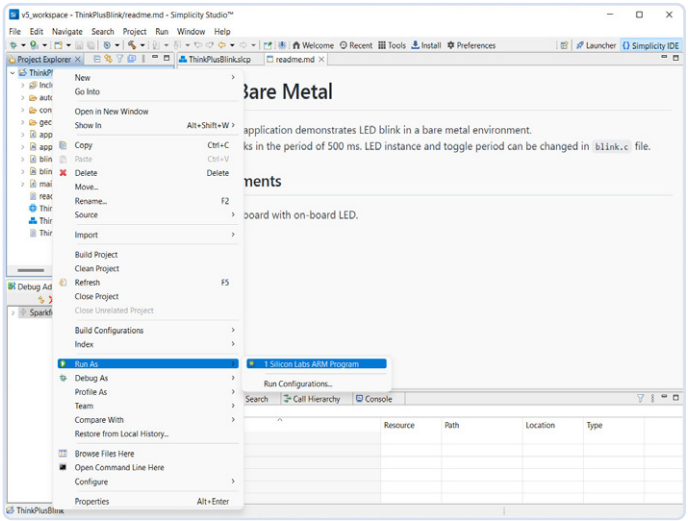


Figure 5. Compilation et exécution du script.

Relier Thing Plus Matter à Nest Hub (tutoriel)
https://learn.sparkfun.com/tutorials/connecting-thing-plus-matter-to-google-nest-hub

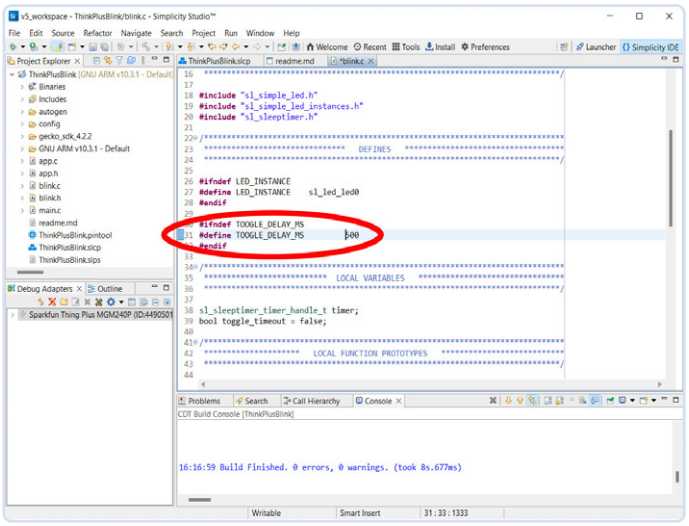


Figure 6. Cette valeur détermine la fréquence de clignotement de la LED.

vous-même à l'aide de cartes telles que la *Thing Plus Matter* (**fig. 7**). Drew, ingénieur chez SparkFun, et Mariah, technologiste créative, vous expliquent comment procéder dans un tutoriel et une vidéo publiés sur SparkFun [5].

Cette technologie étant récente, tutoriels et exemples sont encore peu nombreux, mais de nouveaux apparaissent chaque jour. Prenez une longueur d'avance en utilisant Matter dès aujourd'hui, ce protocole contribue à l'unification de la domotique et se répand sur toutes les plateformes.

VF : Herve Moreau — 230224-04

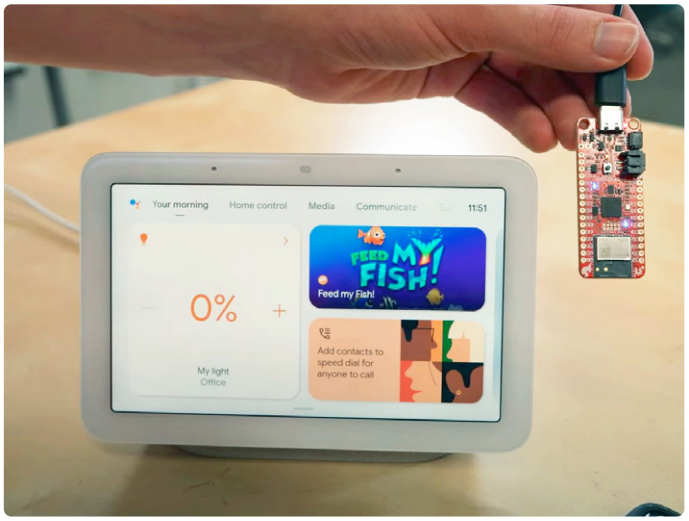


Figure 7. La carte Thing Plus Matter et l'enceinte Nest Hub.

Des questions, des commentaires ?

Envoyez un courriel au service d'assistance de SparkFun (support@sparkfun.com), ou contactez Elektor (redaction@elektor.fr).

À propos de l'auteur

Rob Reynolds a rejoint SparkFun en 2015, et depuis cinq ans y exerce comme technologiste créatif. Il s'appuie sur son expérience dans le domaine des arts pour créer des projets, des vidéos et des tutoriels qui sont le plus souvent à la fois instructifs et divertissants. Vous pouvez le suivre sur Twitter à @thingsrobmade.

Produits

- **Carte SparkFun Thing Plus Matter (MGM240P)**
  https://elektor.fr/20442

Plus d'infos sur la carte

LIENS

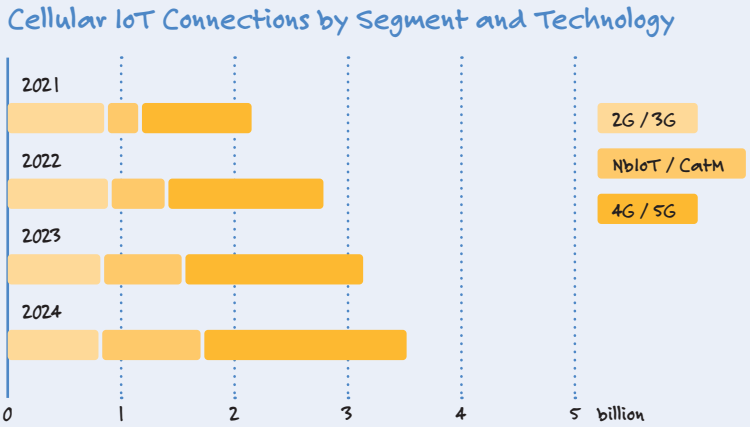
[1] Matter sur GitHub : https://github.com/project-chip/connectedhomeip
[2] Carte de développement Thing Plus Matter de SparkFun : https://www.sparkfun.com/products/20270
[3] Tutoriel détaillé de SparkFun : https://bit.ly/42NRVII
[4] Simplicity Studio : https://silabs.com/developers/simplicity-studio
[5] Tutoriel Thing Plus Matter et Nest Hub : https://bit.ly/3VRcQCl

Perspectives sur l'IdO

L'IdO est encore un sujet brûlant. En fait, on pourrait dire que l'innovation dans l'IdO n'en est qu'à ses débuts. Selon McKinsey, la valeur totale de l'IdO pourrait atteindre 12,5 billions de dollars d'ici 2030 [1]. Les capteurs sont des éléments clés de l'IdO, et ils sont présents dans d'innombrables applications conçues et déployées par les membres de la communauté d'Elektor. Le marché total des capteurs de l'IdO devrait passer d'environ 10,9 milliards de dollars en 2022 à 22,1 milliards de dollars en 2027 [2]. Cette croissance potentielle ouvre de nouvelles perspectives aux électroniciens professionnels et aux *makers*.

5 tendances technologiques des capteurs IdO

- Capteurs intelligents
- Capteurs à ultra-basse consommation
- Capteurs logiciels et virtuels
- Fusion de capteurs
- Biocapteurs



Source : Ericsson [3]

12,5 billions de dollars

Estimation de la valeur totale de l'IdO d'ici 2030 selon McKinsey.

Le protocole Matter

Le protocole Matter permet aux appareils domotiques de communiquer entre eux, indépendamment de leur fabricant. Il normalise la configuration des appareils de différentes marques. Selon la Connectivity Standards Alliance, « en se basant sur le protocole Internet (IP), Matter permettra la communication entre les appareils domotiques, les applications mobiles et les services cloud, et définira un ensemble spécifique de techniques de connexion basées sur le protocole IP pour la certification des appareils » [4]. Appareils pris en charge par Matter : ponts, contrôleurs, serrures de porte, commandes de chauffage, de ventilation et de climatisation, éclairage et électricité, appareils multimédias, capteurs de sécurité et de surveillance, volets roulants et stores.

37 %

Pourcentage de résidences connectées à l'Internet et possédant un appareil domotique. [6]

68 %

Pourcentage de propriétaires d'appareils (ou de propriétaires prévus) qui pensent que la certification Matter est importante. [6]

Source : CSA [5]

Technologie 5G

La 5G - la cinquième génération de technologie cellulaire sans fil – offre de nombreux avantages aux entreprises et aux consommateurs : une connectivité de plus en plus rapide et sécurisée, une latence réduite, une plus grande autonomie de la batterie, et bien plus encore. Depuis son lancement en 2019, elle est devenue l'une des technologies les plus importantes pour l'industrie.

2 trillions de dollars

L'augmentation potentielle du PIB mondial si la 5G est déployée dans les domaines commerciaux clés suivants : commerce de détail, industrie, soins de santé et mobilité. [8]

50 %

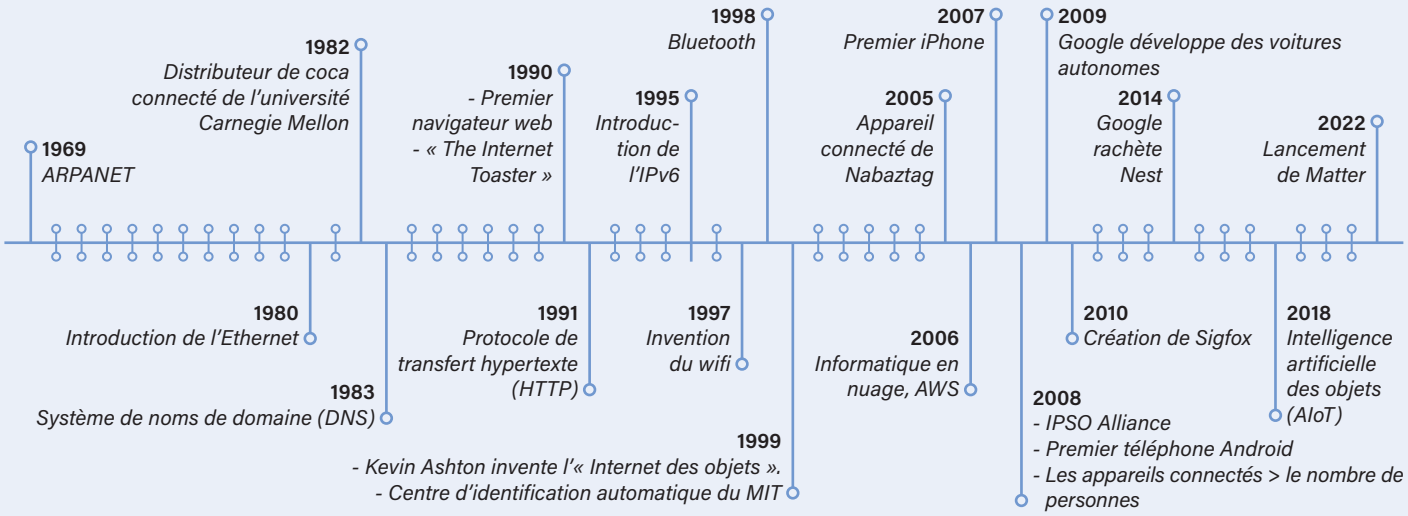
Pourcentage de personnes participant à une enquête de l'IEEE ayant déclaré que la 5G figurait parmi les 5 technologies les plus importantes en 2022. [7]

Où les participants à l'enquête de l'IEEE voient-ils la 5G jouer un rôle ? [7]

- Apprentissage à distance
- Télémédecine
- Loisirs
- Communications quotidiennes
- Transport et contrôle du trafic
- Production/assemblage
- Efficacité énergétique

Évolution de l'IdO

Kevin Ashton, pionnier de la technologie et auteur, a inventé le terme « Internet des objets » à la fin des années 1990. Consultez la frise chronologique suivante pour découvrir quelques moments clés de l'histoire de l'internet des objets.



LIENS

[1] McKinsey & Company, " What is the Internet of Things? ", 17 août 2022 : https://www.mckinsey.com/featured-insights/mckinsey-explainers/what-is-the-internet-of-things
[2] S. Sinha, " 5 IoT sensor technologies to watch " IoT Analytics, 4 janvier 2023 : https://iot-analytics.com/5-iot-sensor-technologies/
[3] Ericsson, " IoT Connections Outlook " : https://www.ericsson.com/en/reports-and-papers/mobility-report/dataforecasts/iot-connections-outlook
[4] CSA, " Matter: The Foundation for Connected Things " : https://csa-iot.org/all-solutions/matter/
[5] CSA, " Matter Executive Overview " : https://csa-iot.org/wp-content/uploads/2022/09/22-Matter-Executive-Overview-One-Pager.pdf
[6] C. White, " The Wait Is Over and Matter 1.0 Is Here ", Parks Associates, 6 octobre 2022 : https://www.parksassociates.com/blog/article/matter-is-here
[7] IEEE, " Advancing Connectivity in 2023 " IEEE Transmitter, 24 octobre 2022 : https://transmitter.ieee.org/advancing-connectivity-in-2023/
[8] McKinsey & Company, " What Is 5G? ", 7 octobre 2022 : https://www.mckinsey.com/featured-insights/mckinsey-explainers/what-is-5g

Q&A avec Amey Inamdar

directeur du marketing technique chez Espressif

Matter, ExpressLink, Rainmaker — de quoi s'agit-il ?

Questions de Tam Hanna (Hongrie) et Jens Nickel (Elektor)

Au salon *Embedded World* 2023 à Nuremberg, Espressif, bien connu pour son célèbre microcontrôleur ESP32, a présenté de nombreuses solutions d'automatisation destinées à faciliter la vie des développeurs (et des utilisateurs) d'applications IdO. Cependant, il n'est pas facile de comprendre ce qui se cache derrière de nouveaux termes comme ExpressLink ou RainMaker. Dans cet entretien, Amey Inamdar, directeur du marketing technique chez Espressif, nous éclaire à ce sujet. Il répond également à des questions d'ordre général sur la gamme de produits de son entreprise.

Elektor : Parlez-nous d'Espressif aujourd'hui. Quel est votre objectif ? Comment la société a-t-elle évolué depuis sa création en 2008 ?

Amey Inamdar : Espressif s'est attaché à démocratiser le segment de l'IdO avec des solutions de connectivité wifi innovantes, abordables, et centrées sur les développeurs. Nous avons veillé à ce que notre matériel soit facilement accessible et à ce que nos logiciels soient disponibles dans la communauté des logiciels libres. Cette philosophie est toujours au cœur des préoccupations d'Espressif. Cependant, notre objectif s'est élargi avec l'évolution des exigences du marché.

Espressif continue, non seulement à améliorer la connectivité wifi et BLE en ajoutant le wifi 6, la prise en charge bibande et le Bluetooth LE(5) à son portefeuille, mais aussi à répondre aux besoins émergents du marché en supportant la norme 802.15.4 comme base pour les protocoles *Thread* et *ZigBee*. Nous n'avons jamais cessé d'améliorer les aspects de la connectivité liés à la consommation d'énergie et aux performances RF. Nous avons également ajouté des périphériques d'interface de pointe à nos produits.

De plus, Espressif a été le pionnier des microcontrôleurs multicœurs avec ses puces ESP32 et ESP32-S3. La prise en charge de l'accélération de l'IA dans l'ESP32-S3 est bénéfique pour les applications d'apprentissage automatique en périphérie. Des accélérateurs matériels et des encodeurs multimédias ont également été ajoutés aux puces les plus récentes.

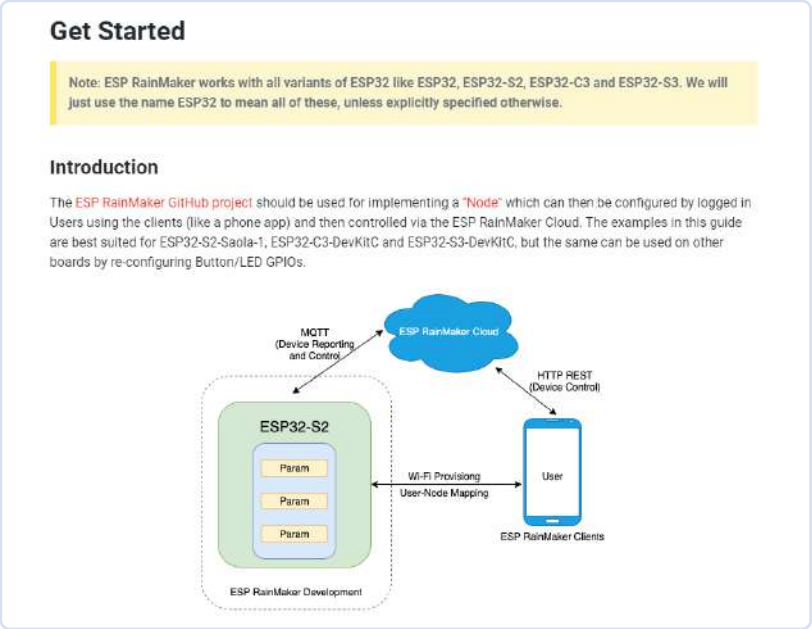
En outre, les *SoC* d'Espressif sont dotés de fonctions de sécurité qui garantissent que tous les appareils construits répondent aux exigences de sécurité. La plupart des puces sont également dotées de périphériques de sécurité innovants, tels que le périphérique de signature numérique, qui offre une fonctionnalité matérielle intégrée semblable à celle d'un élément de sécurité.

En plus de cela, Espressif a également évolué en tant que fournisseur de solutions complètes, où nous identifions les points faibles des clients et y répondons efficacement avec des solutions qui vont au-delà du matériel et des kits de développement logiciel (*SDK*). Les modules ESP RainMaker, ESP Insights et ESP ZeroCode en sont de bons exemples.

Elektor : Au salon embedded world, Espressif a présenté des solutions domotiques utilisant des cartes ESP32 connectées à Amazon Web Service (AWS). Nous avons entendu parler de « RainMaker » et « ExpressLink ». Pouvez-vous nous en dire plus sur ces deux solutions ? Sont-ils indépendants ou peuvent-ils fonctionner ensemble ?

Amey Inamdar : ESP RainMaker est une implémentation IoT cloud que vous pouvez déployer dans votre propre compte AWS. Elle dispose également d'un firmware SDK open-source, d'applications téléphoniques et de compétences d'assistant vocal. ESP RainMaker est basé sur l'architecture sans serveur d'AWS et utilise AWS IoT Core et les services connexes en interne. ExpressLink est un module de connectivité qui fournit une simple interface de commande AT au microcontrôleur hôte et offre une connectivité transparente à AWS IoT Core et aux services connexes, tels que la transmission à distance (*OTA*). ExpressLink réduit la complexité de la construction et de la gestion des appareils connectés.

ESP RainMaker et ExpressLink sont complémentaires. Les clients peuvent utiliser l'un ou l'autre, ou



▲

Figure 1. RainMaker connecte les clients (comme les smartphones) et les appareils basés sur ESP32 via un backend du cloud, qui est basé sur AWS.

les deux ensemble pour créer facilement des appareils connectés.

Elektor : Commençons par RainMaker [1]. Selon la documentation, les fonctions de RainMaker sont accessibles via ESP-IDF. Depuis l'automne, une interface Arduino est également disponible. Pour un usage professionnel, recommanderiez-vous toujours l'IDF ?
Amey Inamdar : ESP-IDF est un kit de développement logiciel (*SDK*) pour la création d'applications IdO. Ce n'est pas le seul cadre logiciel, mais ESP-IDF est le projet dans lequel nous introduisons d'abord le support pour les nouveaux produits que nous lançons. ESP-IDF est un projet open-source, et il constitue également la base de nombreux autres structures logicielles, applications et solutions d'Espressif.
Arduino fournit une interface simple et permet de tirer parti des bibliothèques et des pilotes de périphériques existants. ESP-IDF offre plus de flexibilité aux clients qui souhaitent développer des applications multitâches avec un accès à toutes les API natives du SDK. Les clients peuvent choisir entre l'interface Arduino et l'IDF en fonction de leurs domaines d'utilisation.

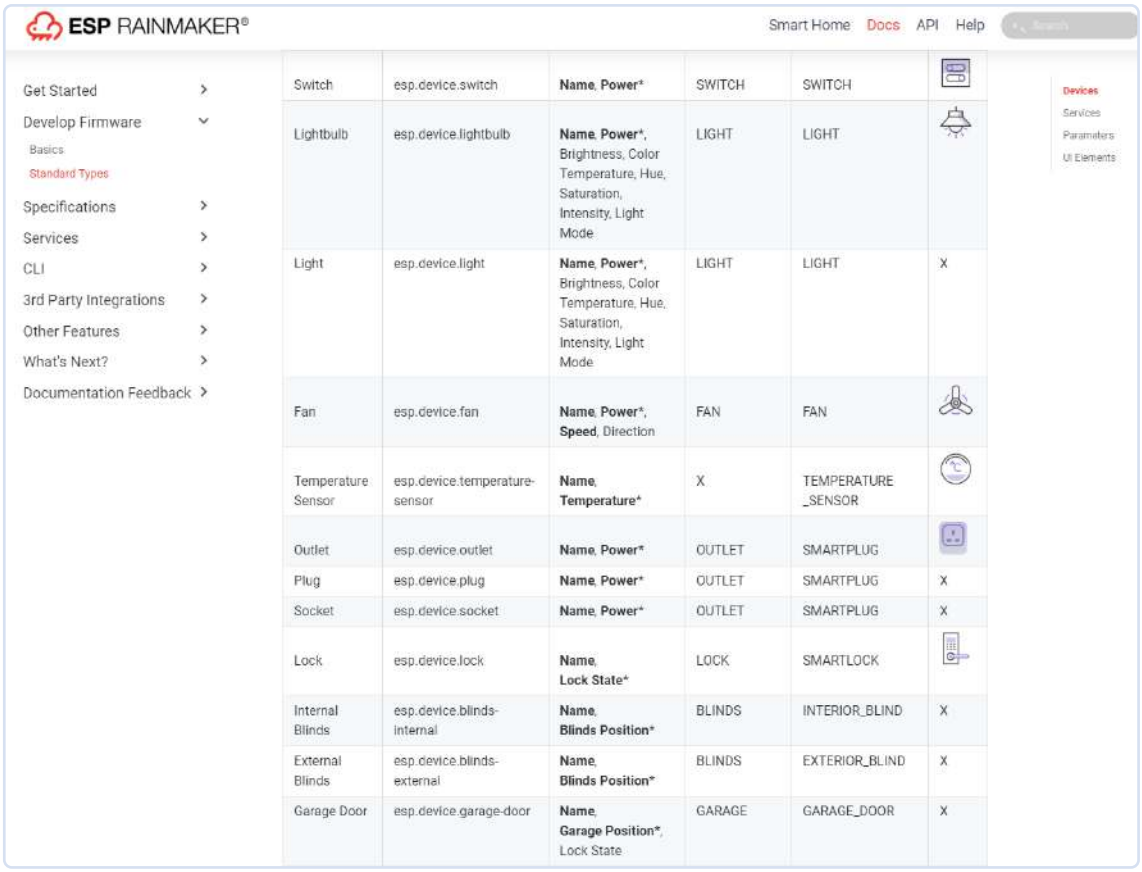
Elektor : De nombreux utilisateurs semblent apprécier RainMaker, mais ils sont réticents à l'héberger sur le cloud AWS. RainMaker peut-il être utilisé sans AWS ? Une telle solution est-elle prévue pour l'avenir ?
Amey Inamdar : Il existe plusieurs approches pour créer des plateformes IdO dans le cloud. Vous pouvez utiliser l'environnement *Platform as a Service* (PaaS) avec simplement des conteneurs ou des machines virtuelles d'entreprises de cloud. Cependant, ces implémentations dans le cloud nécessitent des efforts techniques pour maintenir la flexibilité et le coût. Très peu de clients peuvent gérer ce type de DevOps. C'est pourquoi nous avons décidé de baser ESP RainMaker sur l'architecture AWS Serverless qui fournit une solution sans maintenance avec une tarification en fonction de l'utilisation. Dans ce contexte, vous pouvez voir que, en raison de choix conscients faits en faveur des clients, il n'est pas facile d'utiliser RainMaker sans AWS.

Elektor : Les spécifications de RainMaker prévoient des types d'appareils standard, dont la plupart relèvent de la maison intelligente (stores, ventilateurs, alarmes antivol, etc.) [2]. Cependant, le système pourrait être utilisé pour d'autres applications telles que l'éclairage de studio et l'équipement vidéo, sans parler des équipements de production cinématographique. Envisagez-vous d'élargir le champ d'application ou voyez-vous RainMaker uniquement dans le domaine de la maison intelligente ?
Amey Inamdar : Le systeme de gestion du cloud (*backend*) de l'ESP RainMaker est totalement indépendant des types d'appareils. Il assure le transfert et le stockage de données temporelles. Aucun travail n'est donc nécessaire pour prendre en charge un type d'appareil particulier. Même les applications pour téléphone fournies dans les boutiques d'applications en ligne génèrent dynamiquement l'interface utilisateur sur la base de la description fournie par l'appareil. Par exemple, si l'appareil indique qu'il s'agit d'une table de mixage sonore dotée de huit faders de fréquences différentes, et que chacun d'entre eux est représenté par un curseur allant de la valeur minimale à la valeur maximale, les applications pour téléphone modifieront automatiquement l'interface utilisateur en conséquence, et aucune modification ne sera nécessaire du côté du cloud.

Elektor : Ne serait-il pas judicieux de laisser Rainmaker accessible à d'autres fabricants de microcontrôleurs, par exemple Microchip ?
Amey Inamdar : Le protocole ESP RainMaker et le SDK de l'appareil sont entièrement open-source, et nous n'avons aucun problème à utiliser ESP RainMaker avec des microcontrôleurs qui ne proviennent pas d'Espressif.

Elektor : Venons-en à ExpressLink [3]. On peut lire sur Internet que « les modules compatibles ExpressLink fournissent une simple interface série par laquelle le microcontrôleur hôte se connecte aux services AWS IoT, transformant ainsi n'importe quel produit hors ligne en un produit connecté au cloud ». Comme nous pouvons le voir sur une page web AWS [4] d'Espressif, il semblerait qu'il n'y ait qu'une seule carte compatible disponible.
Amey Inamdar : Oui, mais il s'agit simplement d'une carte de développement pour l'évaluation et le prototypage autour du module compatible ExpressLink (*qui est essentiellement un module ESP32-C3 avec un micrologiciel préprogrammé faisant toute la magie AWS ~ Ndlr*). Nous avons choisi un facteur de forme compatible avec Arduino pour cette carte de développement. La disposition des broches de cette ESP32-C3-AWS-ExpressLink-DevKit est compatible avec celle de la carte Arduino Zero et peut être directement branchée dessus.

L'industrie de la maison intelligente a besoin de normalisation.



Switch	esp.device.switch	**Name**, **Power***	SWITCH	SWITCH	
Lightbulb	esp.device.lightbulb	**Name**, **Power***, Brightness, Color Temperature, Hue, Saturation, Intensity, Light Mode	LIGHT	LIGHT	
Light	esp.device.light	**Name**, **Power***, Brightness, Color Temperature, Hue, Saturation, Intensity, Light Mode	LIGHT	LIGHT	X
Fan	esp.device.fan	**Name**, **Power***, **Speed**, Direction	FAN	FAN	
Temperature Sensor	esp.device.temperature-sensor	**Name**, **Temperature***	X	TEMPERATURE_SENSOR	
Outlet	esp.device.outlet	**Name**, **Power***	OUTLET	SMARTPLUG	
Plug	esp.device.plug	**Name**, **Power***	OUTLET	SMARTPLUG	X
Socket	esp.device.socket	**Name**, **Power***	OUTLET	SMARTPLUG	X
Lock	esp.device.lock	**Name**, **Lock State***	LOCK	SMARTLOCK	
Internal Blinds	esp.device.blinds-internal	**Name**, **Blinds Position***	BLINDS	INTERIOR_BLIND	X
External Blinds	esp.device.blinds-external	**Name**, **Blinds Position***	BLINDS	EXTERIOR_BLIND	X
Garage Door	esp.device.garage-door	**Name**, **Garage Position***, Lock State	GARAGE	GARAGE_DOOR	X

Figure 2. RainMaker est livré avec un SDK puissant et une longue liste de types d'appareils prédéfinis.

Elle peut également être facilement connectée à d'autres microcontrôleurs hôtes, tels que le Raspberry Pi.

Elektor : En ce qui concerne Matter [5], Espressif est l'un des premiers et des plus importants fournisseurs de solutions. Comment êtes-vous parvenu à cette position unique ? Qu'est-ce qui vous a fait prendre conscience de la valeur de cette norme ?

Amey Inamdar : L'industrie de la maison intelligente avait besoin d'une normalisation, les tentatives précédentes ayant été infructueuses. Cette expérience utilisateur fragmentée à fait en sorte que les consommateurs avaient du mal à utiliser les appareils connectés, et les fabricants avaient des difficultés à les construire. Cette fois-ci, les acteurs les plus importants de l'écosystème se sont réunis sous l'égide de la *Connectivity Standards Alliance* et ont fait preuve d'engagement pour assurer le succès de la normalisation. En outre, les considérations relatives à la conception du protocole ont largement contribué à son succès. Exemples notables : garantir la sécurité cryptographique pour toutes les communications, prendre en charge les transports wifi et Thread, utiliser la chaîne de blocs pour l'authentification et l'OTA sécurisé, et quelques autres.

Espressif est dans une position unique pour offrir la solution la plus complète pour Matter avec du matériel, des logiciels, des solutions prêtes à l'emploi et des services. Espressif propose des puces et des modules

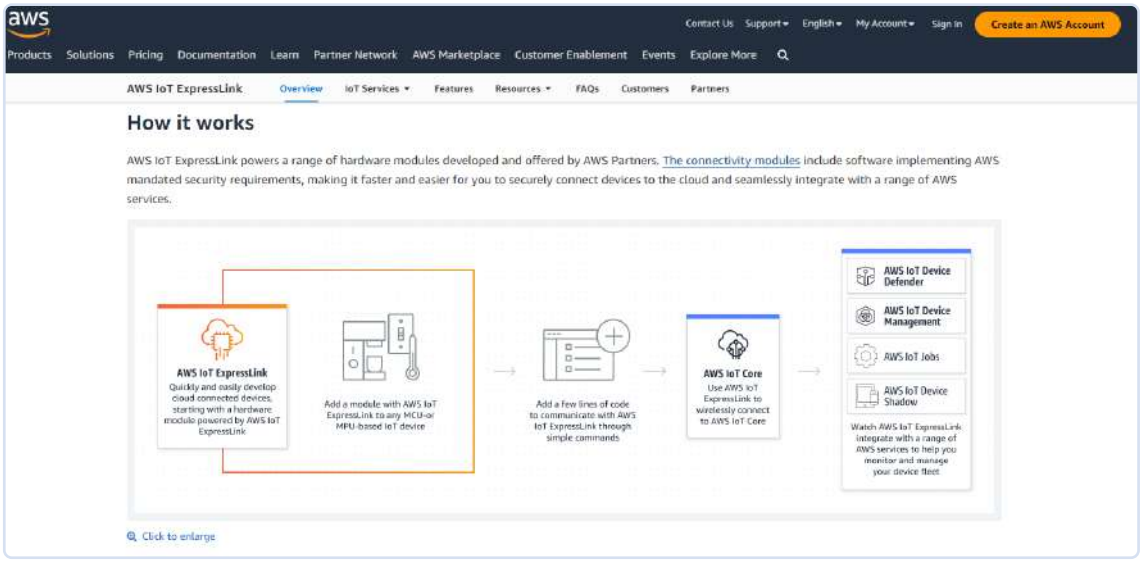


Figure 3. ExpressLink est un module de connectivité qui fournit une interface de commande AT simple au microcontrôleur hôte.

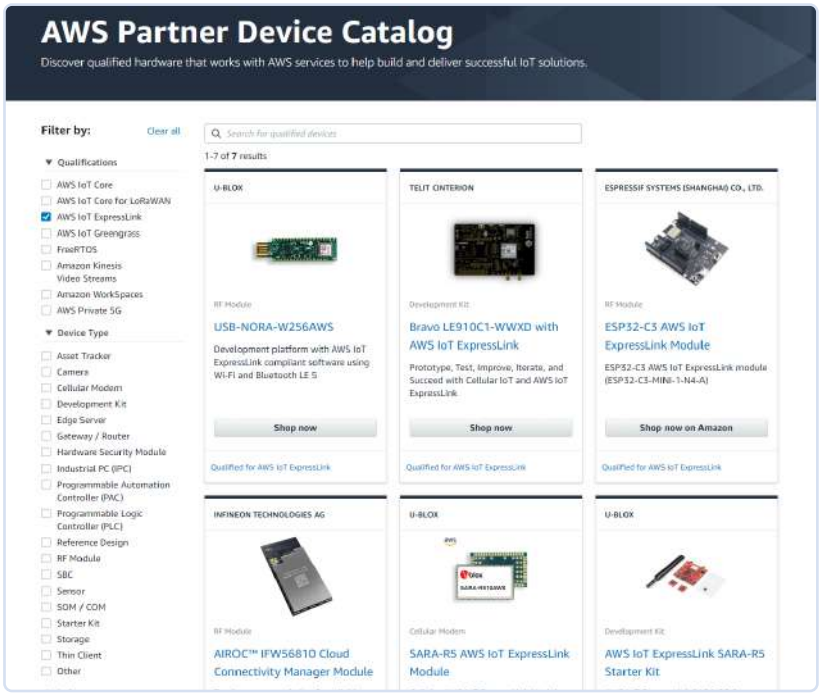


Figure 4. Actuellement, seules quelques cartes de développement ExpressLink d'Espressif et d'autres sociétés sont disponibles.

pour construire des accessoires *Wi-Fi Matter*, des accessoires *Thread Matter*, des routeurs Thread Border, ainsi que des passerelles Matter. Esp-matter est un SDK open-source qui fournit des outils et des exemples pour créer des accessoires Matter. Le service de provisionnement DAC d'Espressif fournit des modules provisionnés de manière sécurisée avec des certificats d'attestation d'appareil afin que les clients n'aient pas à se soucier d'une fabrication sécurisée et complexe. Le service d'assistance à la certification d'Espressif aide les clients à faire certifier leurs accessoires compatibles avec Matter. Et les modules ESP ZeroCode sont livrés avec des firmwares pré-certifiés pour des appareils simples, afin que les clients puissent construire directement des accessoires compatibles avec Matter sans avoir à développer quoi que ce soit de leur côté.

Elektor : L'ESP32-P4 nous montre qu'Espressif sort du marché traditionnel des modules de communication. Étant donné que des fournisseurs tels que ST et Microchip offrent à leurs clients une génération de code automatisée, quel est l'argument de vente unique que vous voyez pour le P4 ?
Amey Inamdar : L'ESP32-P4 dispose d'un ensemble de caractéristiques intéressantes, telles qu'une meilleure puissance de calcul, des périphériques améliorés et une architecture de mémoire. L'ESP32-P4 peut être associée à n'importe quelle autre puce de connectivité Espressif. Une fois jumelée, elle débloquera des possibilités d'utilisation intéressantes dans le segment des appareils IdO haut de gamme. Le principal atout de l'ESP32-P4 sera le support logiciel standardisé grâce auquel le même ESP-IDF supportera l'ESP32-P4, ce qui permettra aux développeurs de transférer leurs applications d'apprentissage et de portage de manière transparente des autres SoC Espressif vers l'ESP32-P4. En outre, le reste de l'écosystème, avec les interpréteurs de langage de haut niveau, le support RTOS, les SDK et les composants logiciels, restera utilisable sur l'ESP32-P4.

Elektor : Étant donné que de plus en plus de fournisseurs se lancent dans le domaine des modules, que comptez-vous faire pour rester à la pointe du marché à long terme ?
Amey Inamdar : Nous voulons continuer à faire ce qu'il faut pour nos clients, et nous pensons que cela nous aidera à maintenir notre position de leader à long terme. Je ne pense pas qu'il y ait une seule dimension à cela. Un niveau élevé d'intégration dans notre SoC, l'innovation et le choix de l'architecture du SoC et du sous-système de communication, l'ouverture en termes de logiciels et d'informations, l'absence de compromis sur les caractéristiques de sécurité, l'efficacité de la chaîne d'approvisionnement et la flexibilité de la fabrication sont quelques-uns de ces facteurs de différenciation.

Vf : Laurent Rauber — 230227-04

À propos de Amey Inamdar

Amey est directeur du marketing technique chez Espressif. Il a 20 ans d'expérience dans le domaine des systèmes embarqués et des appareils connectés, avec des rôles dans l'ingénierie, la gestion de produits et le marketing technique. Il a travaillé avec de nombreux clients pour construire des appareils connectés basés sur la connectivité Wi-Fi et Bluetooth avec succès.

LIENS

[1] Page internet d'ESP RainMaker® : https://rainmaker.espressif.com/
[2] Types d'appareils standards prédéfinis ESP RAINMAKER : https://rainmaker.espressif.com/docs/standard-types.html
[3] Page internet AWS IoT ExpressLink : https://aws.amazon.com/iot-expresslink/
[4] Produits ExpressLink sur AWS Partner Device Catalog : https://devices.amazonaws.com/search?page=1&sv=iotxplnk
[5] Solutions d'Espressif pour Matter : https://espressif.com/en/solutions/device-connectivity/esp-matter-solution

guide d'introduction à la sélection **de kits de développement de microcontrôleurs pour applications IoT et IIoT**

Mark Patrick (Mouser Electronics)

L'Internet des objets (IoT) fait désormais partie de notre environnement. Les ingénieurs en développement embarqué qui se lancent dans une nouvelle conception IoT doivent accorder une attention toute particulière à certains facteurs tels que la consommation d'énergie, les capacités de détection et la connectivité sans fil. Or, ce n'est pas chose facile étant donné la pression des délais de commercialisation. Les kits de développement IoT offrent une plateforme de prototypage viable et pratique sur laquelle baser une conception, mais tous sont loin de présenter les mêmes capacités. Il convient donc d'être attentif non seulement aux exigences de l'application, mais surtout aux fonctionnalités et aux capacités du kit. Dans cet article, nous aborderons quelques-uns des nombreux éléments à prendre en compte lorsque l'on choisit un kit de développement IoT pour une nouvelle conception.

L'ère du tout connecté

Il ne fait aucun doute que nous sommes entrés dans l'ère du tout connecté, tant les appareils connectés sont omniprésents. Nous en portons certains sur nous, d'autres servent à surveiller avec précision notre consommation d'électricité ou nous avertissent lorsqu'un visiteur se présente à notre porte… ce ne sont pas les exemples qui manquent. Pour ce qui est des processus de production industrielle, l'avènement de l'Internet industriel des objets (IIoT) transforme la manière dont les usines fonctionnent et contribue au gain d'efficacité général des équipements. En seulement une décennie, nous avons changé notre façon d'interagir avec le monde qui nous entoure et de contrôler notre environnement. Depuis que nous nous sommes habitués à avoir instantanément accès à toutes les informations relatives à certains aspects de notre vie professionnelle ou privée, nous nous demandons comment nous faisions auparavant pour vivre sans téléphone mobile.

En voiture aussi, nos habitudes ont changé depuis que nous recevons en temps réel les dernières informations sur le trafic ou des avertissements en cas de ralentissement sur notre trajet. Grâce aux appareils de surveillance médicale connectés à Internet, des patients peuvent aujourd'hui se rétablir dans le confort de leur foyer tout en ayant l'assurance qu'un personnel médical

s'occupe de la surveillance et se tient prêt à intervenir en cas de besoin.
L'industrie a rapidement adopté l'IoT, une technologie tout juste émergente sous l'impulsion d'initiatives gouvernementales comme l'Industrie 4.0 visant à soutenir l'automatisation, à améliorer l'efficience des processus et à rationaliser les opérations. Désormais, une légion de capteurs surveille et rapporte l'avancement de chaque étape d'un processus en envoyant les données collectées au système de contrôle et d'analyse de l'automatisation.
Le déploiement de l'IoT et de l'IIoT apporte des avantages considérables, mais, sur le plan de l'ingénierie électronique, le développement d'un dispositif IoT rencontre aussi son lot de difficultés.

Tour d'horizon des exigences liées à la conception d'un dispositif IoT

Bien qu'il existe une grande variété d'applications IoT, leur conception est régie par un ensemble d'exigences fonctionnelles de base globalement identiques, qu'il s'agisse de concevoir un capteur de pression pour un processus industriel ou un détecteur de présence dans un bureau.
Lors de la collecte de données initiale en vue d'établir les spécifications techniques générales d'un nouveau dispositif IoT, il convient de tenir compte de chacun des aspects énumérés ci-dessous, car ce sont eux qui définissent l'architecture fonctionnelle et la conception du dispositif.
La détection : les capteurs sondent tous les aspects de notre environnement, de la température à la pression atmosphérique en passant par les mouvements des personnes. Par exemple, les données collectées par une caméra peuvent servir à alimenter une application d'apprentissage automatique pour la détection d'objets dans le but de confirmer qu'une étiquette a été correctement apposée sur une bouteille. Plusieurs choix techniques dépendent de ce qui doit être détecté et à quelle fréquence. D'autres paramètres à prendre en compte sont le coût, la taille et la complexité du capteur. Une thermistance utilisée pour mesurer la température nécessitera des composants supplémentaires dans le domaine analogique et un traitement logiciel avant la conversion en une forme numérique. Le nombre de capteurs nécessaires et la fréquence à laquelle ils doivent être interrogés ont également leur importance.
La connectivité : comment le dispositif IoT doit-il interagir avec un système de contrôle hôte ? Une communication sans fil fiable est-elle disponible dans chaque cas d'utilisation ou est-il préférable de recourir à une méthode de communication filaire ? Le volume de données à transférer ainsi que la fréquence de transmission de ces données dépendent aussi du type de capteur utilisé. Un réseau maillé sans fil peut offrir un mode de communication plus robuste dans le cas d'un déploiement à grande échelle, mais cela exige que tous les dispositifs IoT fonctionnent de cette manière. Si l'on opte pour un mode de communication sans fil, les décisions techniques porteront sur la création d'une conception discrète ou sur le choix d'un module homologué.
La source d'alimentation : quel serait le profil de consommation d'énergie de votre dispositif IoT ? Certaines applications, certaines fréquences de communication et certains protocoles sans fil impliquent une consommation d'énergie qui dépasse la capacité d'une petite batterie. Pour certains scénarios de déploiement, un branchement au secteur est-il disponible ? Certains capteurs IoT de dernière génération utilisent des technologies de collecte d'énergie pour se passer complètement de l'alimentation par batterie. Il s'agit alors de capter l'énergie de diverses sources d'énergie ambiante (énergie solaire, vibratoire, thermique…) pour charger un supercondensateur.
L'interface utilisateur : est-il prévu qu'un utilisateur puisse interagir avec votre dispositif IoT ? Est-il toujours possible d'installer et de connecter le dispositif au système hôte s'il n'est pas en cours d'utilisation ? Faut-il prévoir un dispositif d'affichage ou un quelconque dispositif d'indication comme des voyants ?
Les applications d'analyse et de contrôle dans le cloud : la raison d'être de l'IoT est de permettre à des appareils de se connecter à un système de contrôle hôte. Les exigences logicielles du capteur et la façon dont il interagit avec le système hôte varient en fonction du mode de connexion et des protocoles y afférents. Doit-on disposer d'une liaison de données permanente pour diffuser un flux de données ou les données peuvent-elles être envoyées par lot à intervalles réguliers ?

Conseils et astuces pour bien choisir son kit de développement IoT

Les kits de développement offrent aux ingénieurs embarqués un moyen pratique et rapide de prototyper une conception. Dans cette partie de notre guide, nous passerons en revue quelques-uns des paramètres à prendre en compte lorsqu'il s'agit de choisir un kit approprié. Les principaux fournisseurs de microcontrôleurs proposent un large choix de kits de développement et d'évaluation de dispositifs IoT. Le mieux est donc de guider son choix en fonction des exigences de l'application précédemment exposées. Voici une liste non exhaustive de fonctionnalités à considérer au moment de sélectionner une plateforme de kit de développement.

La source d'alimentation électrique :
- Comment la carte est-elle alimentée ? USB à partir d'un poste de travail hôte ? Batterie ? La source d'alimentation envisagée suffit-elle à son alimentation électrique ? La carte dispose-t-elle d'un circuit de gestion de puissance (PMIC) auquel vous auriez accès pour essayer d'autres sources d'alimentation ?
- Est-il possible de placer une sonde de courant en ligne pour mesurer la consommation d'énergie en temps réel à des fins de profilage ? Si oui, cela inclut-il tout ce qui se trouve sur la carte et tous les modules supplémentaires, capteurs, etc. ?

Les capteurs :
- La carte est-elle équipée des types de capteurs que votre application utilisera ?
- Est-il possible d'ajouter des capteurs supplémentaires ? Soit en utilisant une connexion périphérique ou un format complémentaire conforme aux normes de l'industrie comme le mikroBUS Click ?
- Quelles interfaces périphériques sont accessibles ? I2C, UART, SPI, GPIO ?
- La carte ou le microcontrôleur dispose-t-il d'un convertisseur analogique-numérique (CAN) que vous pourriez utiliser ? Des composants de conditionnement de signal supplémentaires sont-ils nécessaires ?

La connectivité :
- De quelles options de connectivité

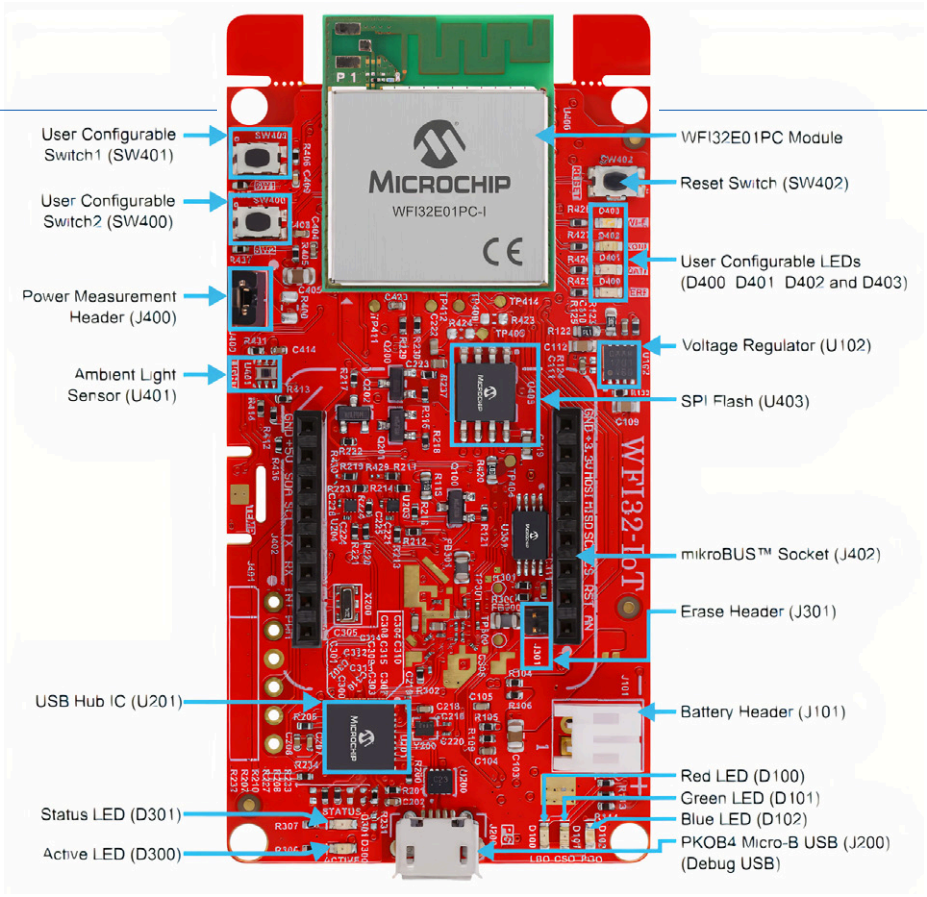


Figure 1. Le kit de développement IoT EV36W50A de Microchip (Source : Microchip).

filaire/sans fil la carte dispose-t-elle ? Ethernet, Wi-Fi, LoRa, BLE, ISM, etc.

- En l'absence de connectivité embarquée, est-il possible d'en ajouter facilement une ? Le fabricant recommande-t-il et prend-il en charge un module sans fil approprié ? Une option d'interface tierce (mikroBUS Click, etc.) est-elle présente ?
- Le micrologiciel de la carte peut-il être mis à jour par un mode de communication sans fil ?

Les ressources de calcul :

- La carte est-elle dotée du microcontrôleur que vous avez l'intention d'utiliser ? L'avez-vous déjà utilisé et disposez-vous déjà de chaînes d'outils de développement adaptées ?
- Les ressources de calcul de la carte sont-elles suffisantes pour exécuter l'application IoT, les protocoles hôtes et des piles de protocoles de connectivité ?
- Si le microcontrôleur est doté d'un émetteur-récepteur sans fil intégré, pouvez-vous contrôler indépendamment ses modes de veille à des fins d'économie d'énergie ?
- De quelles fonctionnalités de sécurité intégrées le microcontrôleur dispose-t-il et ces fonctionnalités sont-elles utilisables avec votre application ?

Le contrôle utilisateur :

- La carte est-elle équipée de boutons, de curseurs tactiles ou d'autres fonctionnalités matérielles pour le contrôle utilisateur ?
- Un dispositif d'affichage est-il présent ? Ce dispositif est-il nécessaire dans l'application finale ?
- Des voyants peuvent-ils être commandés depuis votre code ? Sont-ils disponibles en quantité suffisante ou

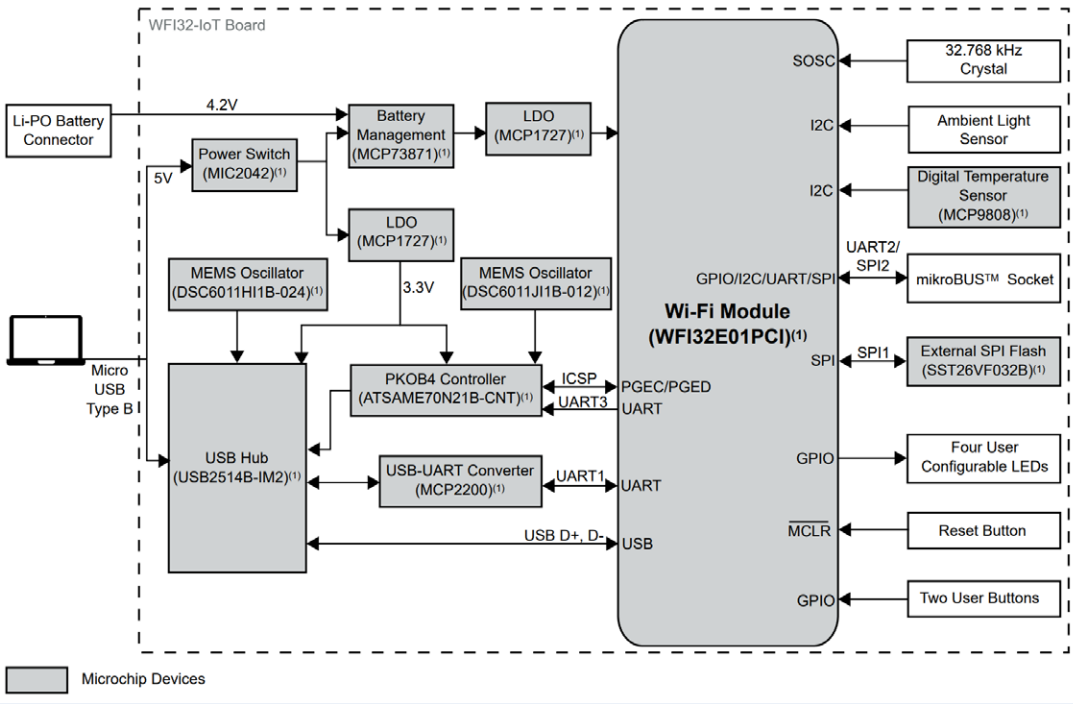


Figure 2. Le schéma fonctionnel de la carte de développement Microchip WFI32-IoT EV36W50A (Source : Microchip).

Figure 3. Le kit de développement pour suivi d'actifs STEVAL-ASTRA1B (Source : ST).

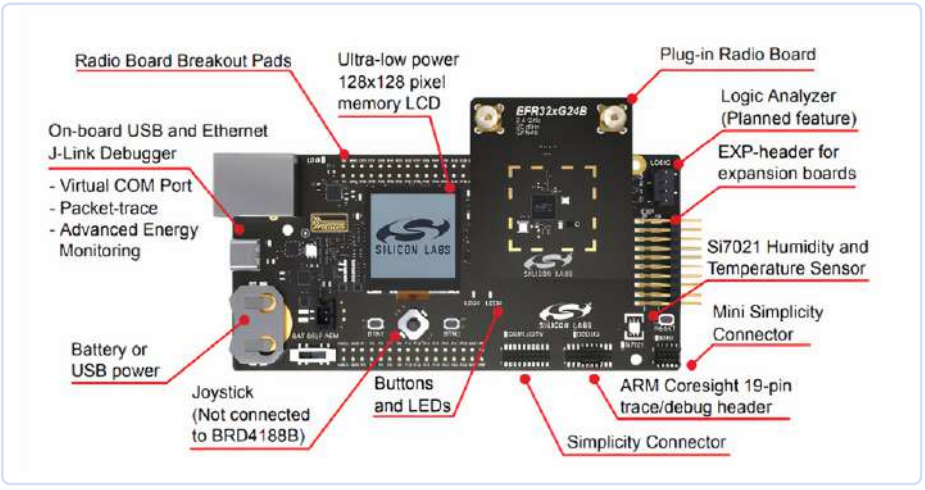


Figure 4. Le module de diversité d'antennes xG24-RB4188A de Silicon Labs monté sur une carte mère Silicon Labs Wireless Kit Pro (Source : Silicon Labs).

pouvez-vous en ajouter rapidement à l'aide d'un port GPIO de rechange ?

Le support logiciel :

- Quelle chaîne d'outils de développement est recommandée pour cette carte ? En disposez-vous déjà ?
- Un pack de support de carte complet (BSP) est-il inclus ?
- Quels pilotes, bibliothèques et micrologiciels supplémentaires sont nécessaires ? Sont-ils libres de droits ?
- Vérifiez quelles sont les exigences relatives à la licence du micrologiciel et de l'intergiciel auprès du fabricant de la carte.
- La carte est-elle fournie avec une démo préinstallée pour présenter les fonctionnalités de la carte ? La communication avec des fournisseurs de services populaires tels que Microsoft Azure ou Amazon AWS est-elle comprise ?
- D'autres exemples de démo et de code sont-ils disponibles pour cette carte ? Existe-t-il un écosystème de bibliothèques et de partenaires de développement ?

Présentation de diverses cartes de développement IoT

La carte de développement WFI32-IoT de Microchip

La Microchip WFI32, référence EV36W50A [1], est une carte de développement IoT complète, entièrement intégrée et autonome **(figure 1)**. Le WFI32-IoT intègre un module sans fil Microchip WFI32E01PC Wi-Fi 802.11 basé sur la famille de microcontrôleurs PIC. Côté capteurs embarqués, nous y trouvons un circuit intégré de température I²C numérique Microchip et un circuit intégré de lumière ambiante numérique. La présence d'une prise mikroBUS permet aux développeurs d'ajouter des capteurs ou des périphériques supplémentaires. Le module de microcontrôleur sans fil est également équipé d'une antenne intégrée. La carte peut être alimentée par un poste de travail hôte ou une batterie LiPo. Un PMIC intégré permet en outre de recharger la batterie par l'hôte USB.

La figure 2 présente le schéma fonctionnel de la carte WFI32-IoT et met en évidence les composants Microchip intégrés à la carte. La carte est préchargée avec une image de démonstration prête à l'emploi (OOB) capable de lire les capteurs embarqués et d'envoyer les données dans le cloud d'Amazon AWS. Le code de démonstration et les instructions complètes sont disponibles dans un dépôt GitHub [2].

La conception de référence pour suivi d'actifs multiconnectivité STEVAL ASTRA1B de STMicroelectronics

La figure 3 présente le kit de développement et la conception de référence STEVAL ASTRA1B [3]. Ce kit spécifiquement conçu pour le prototypage et l'évaluation des applications de suivi d'actifs intègre deux modules de connectivité sans fil : un module de microcontrôleur STM32WB-5MMG [4] sans fil à courte portée 2,4 GHz BLE/ZigBee et un module de microcontrôleur sans fil STM32WL55JC longue portée pour la communication par LPWAN (par exemple LoRa).

Le STEVAL ASTRA1B comprend un ensemble complet de capteurs capables de mesurer plusieurs paramètres environnementaux et de mouvement. Un module GNSS fournit des données de localisation en

Figure 5. Un exemple de kit de développement LR1120 proposé par SEMTECH (Source : SEMTECH).

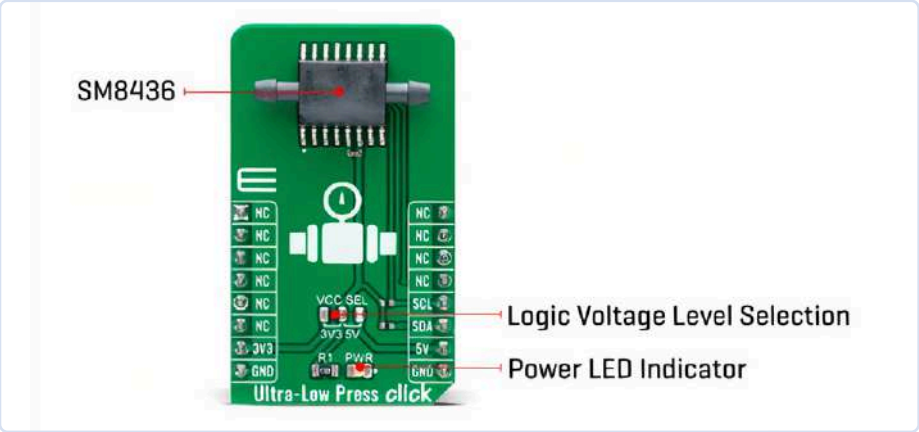


Figure 6. La carte Ultra-Low Press Click de Mikroe (Source : Mikroe).

extérieur. Enfin, la carte est livrée avec un élément sécurisé STSAFE, une batterie de 480 mAh et une démo OOB composée d'un tableau de bord pour le suivi d'actifs et d'une application pour smartphone.

Silicon Labs xG24-RB4188A

Le xG24-RB4188A de Silicon Labs [5] est un module de diversité d'antenne enfichable pour le prototypage d'applications sans fil 2,4 GHz **(figure 4)**. Il se branche sur la carte de démarrage sans fil Silicon Labs BRD4001. Le module accueille un système sur puce Gecko sans fil EFR32 de Silicon Labs, un commutateur RF, un réseau correspondant et deux connecteurs d'antenne SMA. La sortie RF de l'EFR32 est de +20 dBm.

Le kit de développement LR1120 de SEMTECH

Pour prototyper des applications LoRa LPWAN basées sur le microcontrôleur sans fil LR1120 de SEMTECH [6], le fabricant propose une gamme de kits de développement [7] LR1120 semblables au kit illustré à la **figure 5**. Des variantes régionales de ces kits sont disponibles en fonction de la bande ISM sub-GHz (bande industrielle, scientifique et médicale). Le LR1120 convient aux applications multi-régionales de géolocalisation des actifs, de gestion des stocks et de prévention des vols.

Comme nous l'avons précédemment mentionné, il est possible d'ajouter des capteurs ou des périphériques supplémentaires à une carte de développement. Nous expliquions également à propos de la carte Microchip que celle-ci est équipée d'une prise mikroBUS. Le mikroBUS, développé par Mikroe, est rapidement devenu un standard industriel adopté par de nombreux fournisseurs de semiconducteurs pour leurs cartes de développement et d'évaluation. La technologie mikroBUS rassemble les protocoles de connectivité sérielle SPI, UART et I²C ainsi que des signaux d'alimentation, analogiques et MLI dans un format de prise compact. Mikroe a développé des centaines de cartes Click [8] utilisant ce facteur de forme pratique.

La carte Mikroe Ultra-Low Press Click [9] en est un exemple. Conçue pour les mesures pneumatiques à basse pression, elle est dotée du capteur de pression TE Connectivity SM8436. Celui-ci communique à l'aide de l'interface I2C **(figure 6)**.

Aller de l'avant avec votre kit de développement IoT

Les cartes de développement simplifient grandement le prototypage d'applications IoT. Dans ce bref article, nous avons pu aborder quelques-unes des questions qui doivent s'imposer à l'esprit des ingénieurs en développement embarqué au moment de choisir une carte de développement appropriée. Outre les points mentionnés, il convient aussi de tenir compte des aspects spécifiques de l'application finale.

Alors, qu'allez-vous développer ?

230338-04

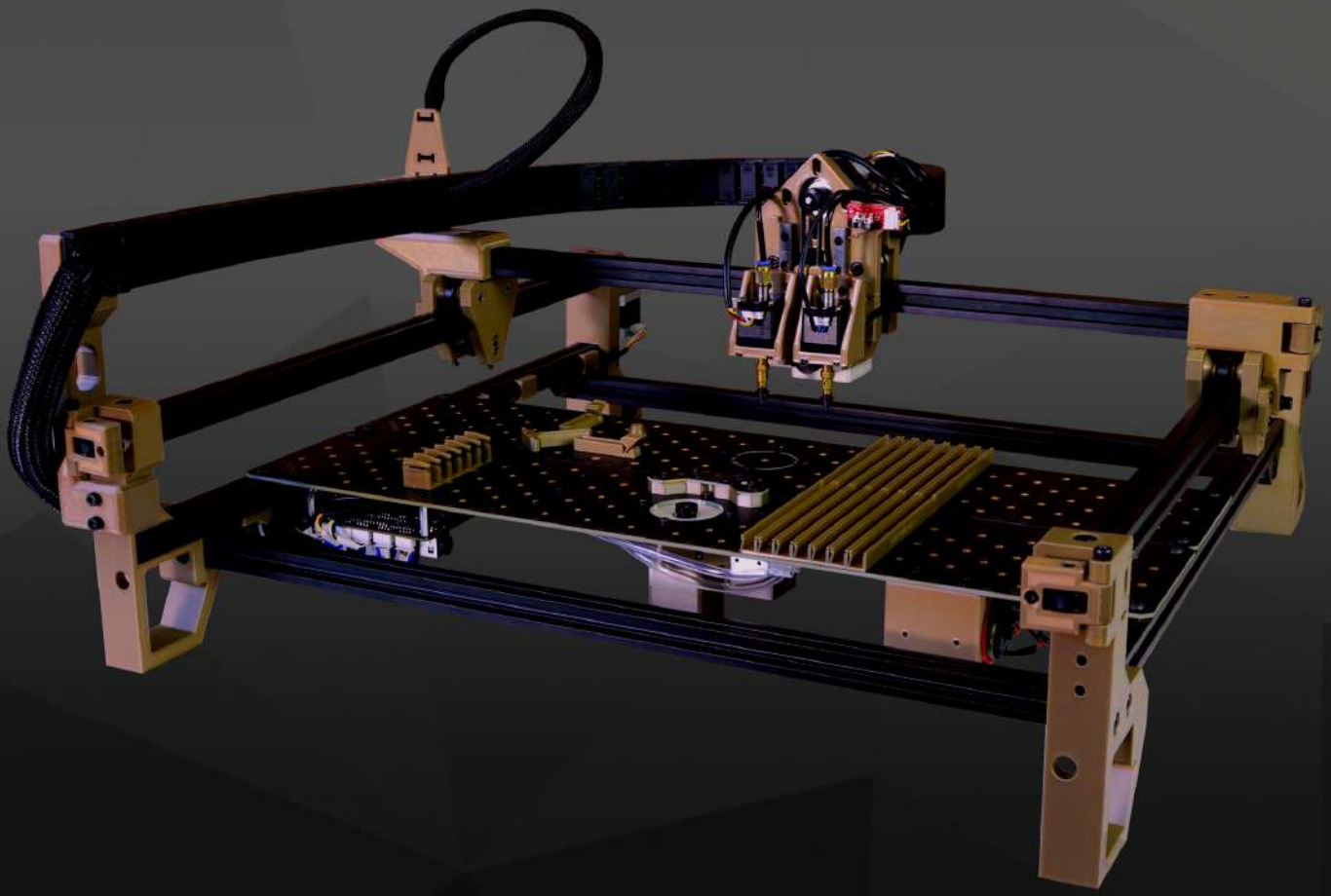
À propos de l'auteur

Mark Patrick est responsable de la création et de la diffusion du contenu technique – un contenu essentiel à la stratégie de Mouser visant à soutenir, informer et inspirer son public d'ingénieurs. Avant de diriger l'équipe de marketing technique, Patrick faisait partie de l'équipe de marketing achat de la région EMEA et jouait un rôle essentiel dans l'établissement et le développement des relations avec les principaux partenaires et fournisseurs. En plus d'avoir occupé divers postes dans les départements techniques et marketing, Patrick a travaillé pendant huit ans chez Texas Instruments, dans les services support et ventes techniques. Ingénieur expérimenté, passionné de synthétiseurs vintage et de motos, il n'hésite pas à les réparer. Patrick est titulaire d'un diplôme d'ingénieur en électronique avec mention très bien de l'université de Coventry.

LIENS

[1] EV36W50A WFI32 carte de développement IoT de Microchip Technology : https://bit.ly/3Vw4L5U
[2] WFI32-IoT sur GitHub : https://github.com/MicrochipTech/WFI32-IoT
[3] STMicroelectronics Carte d'évaluation de suivi d'actifs STEVAL-ASTRA1B : https://bit.ly/3LwN4P6
[4] STM32WB5MMG module sans fil 2,4 GHz: https://bit.ly/3NFird0
[5] xG24-RB4188A carte radio à diversité d'antennes de Silicon Labs : https://bit.ly/3LBRFQ5
[6] Semtech LR1120 Wi-Fi/GNSS Scanner + LoRa Transceiver : https://bit.ly/428oET8
[7] Kit de développement LR1120: https://bit.ly/42rdGaO
[8] Click Boards™ - Mikroe : https://bit.ly/3LAaH9j
[9] Ultra-Low Press Click - Mikroe : https://bit.ly/3LXr2GH

VS



C'est sans appel.

Le LumenPnP Desktop Pick & Place Machine assemble vos composants, vous n'aurez donc plus jamais besoin d'utiliser de pinces

-Radicalement
 Open Source
-Alimentation électrique
-Buse double
-Place les 0402s
-Prix abordable

Opulo.io

Opulo™

un condensateur **n'est pas toujours** capacitif !

René Kalbitz (Würth Elektronik eiSos)

Les condensateurs sont par définition des composants capacitifs. Ce qui semble pure lapalissade n'est pourtant vrai que sous certaines conditions et plages de fréquences. Cet article montre comment le spectre d'impédance de divers types de condensateurs permet d'anticiper leur comportement, capacitif ou non.

Les ingénieurs se servent souvent des paramètres S pour représenter les propriétés électriques d'un condensateur en function de la frequence. L'étude de ces spectres fournit des informations sur la nature électrochimique, physique et technique du composant. Pour être pertinentes, les caractéristiques cherchées doivent être dépouillées des effets parasites et des artefacts de mesure toujours présents. Puisqu'il n'est pas toujours possible d'inclure toutes les données dans une fiche technique, le concepteur doit parfois s'appuyer sur de tels spectres pour sélectionner un composant adapté à son circuit. Afin de faciliter sa tâche, Würth Elektronik eiSos a créé REDEXPERT [1], un outil en ligne qui fournit les spectres de nombreux composants ainsi que différentes mesures d'intérêt. Cet article explique comment déduire certaines propriétés électriques de ces spectres.

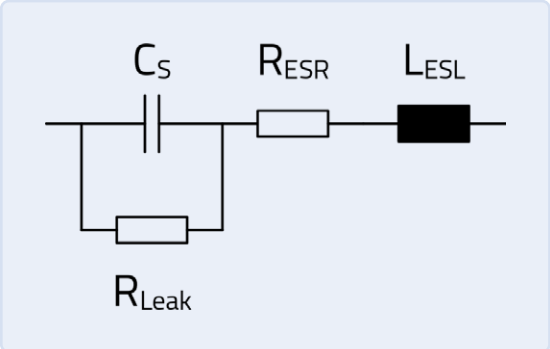


Figure 1. Modèle standard d'un condensateur : capacité C_S, résistance-série équivalente R_{ESR}, inductance-série équivalente L_{ESL}, et résistance de fuite R_{Leak}.

Circuit équivalent

Le circuit de la **figure 1** sert à modéliser le spectre d'impédance de tout type de condensateur, de la pastille céramique multicouche au supercondensateur.

C_S représente la capacité d'un condensateur idéal. Un condensateur réel subit des pertes qui « ralentissent » sa charge. Ces pertes sont représentées par la résistance-série équivalente (ESR, toutes les abréviations de cet article sont en anglais). La résistance de la charge et celle des fils de connexion contribuent également à la résistance ESR.

La capacité d'un condensateur idéal est définie par l'équation différentielle

$$C_S = \frac{dQ}{dV}$$

où dQ est la variation de charge à la surface du condensateur, et dV la variation de tension aux bornes du condensateur.

Tout courant alternatif parcourant un conducteur métallique induit un champ magnétique qui s'oppose à ce courant. Dans le modèle considéré (dit L-C-R, ou standard), cette propriété est représentée par l'inductance-série équivalente (ESL), L_{ESL} sur la **figure 1**.

C_S, R_{ESR}, et L_{ESL} permettent de décrire la majorité des spectres. Dans l'approche la plus simplifiée, suffisante

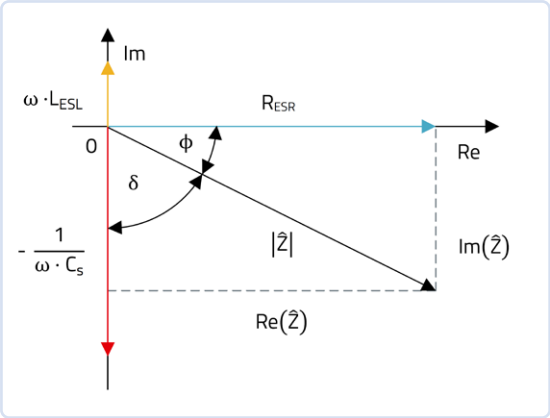


Figure 2. Représentation vectorielle de l'impédance dans le plan complexe. R_{Leak} est négligée par souci de simplicité.

en ingénierie électrique, ces trois paramètres ne varient pas avec la fréquence, autrement dit sont des constantes.

La perte de charge dans le temps, c.-à-d. le courant de fuite, peut être décrite avec une bonne approximation par la résistance ohmique idéale R_{Leak}.La valeur de R_{Leak} est d'ordinaire bien supérieure à R_{ESR}, et peut donc être ignorée ($R_{Leak} \rightarrow \infty$). Son effet n'apparaît sur le spectre qu'à des fréquences très basses, sous 1 Hz [2].

Spectres d'impédance et de capacité

Avant d'aborder les spectres, explicitons leur contexte théorique. Le circuit ci-dessus peut être décrit par diverses grandeurs à valeurs complexes qui dépendent de la fréquence : l'impédance $\hat{Z}$, la capacité $\hat{C}$, le paramètre de diffusion $\hat{S}$ (paramètre S), ou encore la permittivité $\hat{\varepsilon}$, L'impédance peut s'écrire $\hat{Z} = Re(\hat{Z}) + I \times Im(\hat{Z})$, où $Re(\hat{Z})$ est la partie réelle et $Im(\hat{Z})$ la partie imaginaire ; ou, sous forme polaire :

$$\hat{Z}=|\hat{Z}|\cdot e^{i\phi}$$

où $|\hat{Z}|$ est le module et ϕ l'argument. Dans le plan complexe (**fig. 2**), ϕ mesure l'angle entre $Re(\hat{Z})$ (abscisse) et le vecteur complexe $\hat{Z}$. Physiquement, $|\hat{Z}|$ est le rapport entre la tension et l'intensité, tandis que ϕ représente le déphasage entre la tension et le courant à une fréquence donnée. L'angle de déphasage ϕ et l'angle de perte sont reliés par :

$$\arctan\left(\frac{Re(\hat{Z})}{|Im(\hat{Z})|}\right)=\delta=\frac{\pi}{2}-\phi$$

En électricité, on utilise aussi couramment l'amplitude $|\hat{Z}|$ et sa résistance-série équivalente $R_{ESR} = Re(\hat{Z})$. Sur la **figure 2**, la résistance-série équivalente du modèle standard (celui de la fig. 1) correspond à la partie réelle de l'impédance. La **figure 2** permet également de visualiser la relation entre l'amplitude complexe $|\hat{Z}|$ et les autres paramètres du modèle (à l'exception de R_{Leak}). L'expression mathématique de ces paramètres est donnée en annexe du lien [2].

L'impédance peut aussi servir à exprimer la capacité complexe :

$$\hat{C}=\frac{1}{i\cdot 2\cdot\pi\cdot f\cdot\hat{Z}}=Re(\hat{C})+i\cdot Im(\hat{C})$$

L'ensemble des grandeurs précédentes – comme $Re(\hat{Z})$, $Im(\hat{Z})$, $|\hat{Z}|$ ou l'angle de perte δ – peut être mesuré à l'aide d'analyseurs de réseau ou d'impédance. Tout composant électronique (pas juste les condensateurs) peut être caractérisé par un jeu de variables dépendantes de la fréquence, telles que $Re(\hat{Z})$ et $Im(\hat{Z})$ ou $Re(\hat{C})$ et $Im(\hat{C})$. Ce n'est toutefois qu'au travers de circuits équivalents comme celui de la **figure 1** qu'il est possible d'interpréter les

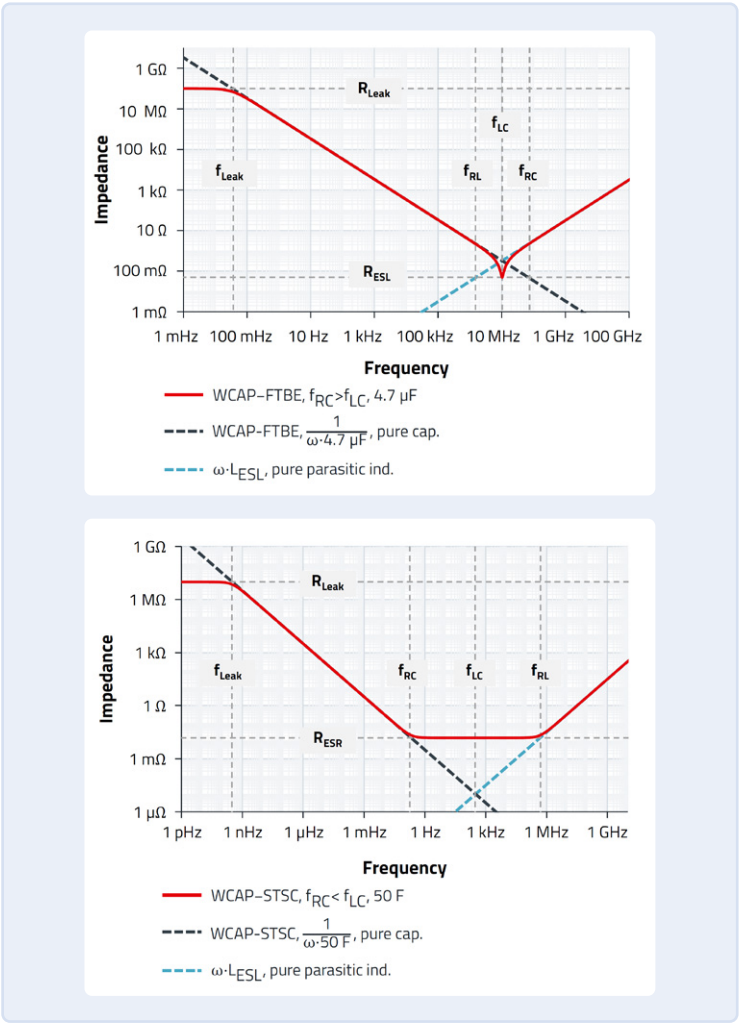
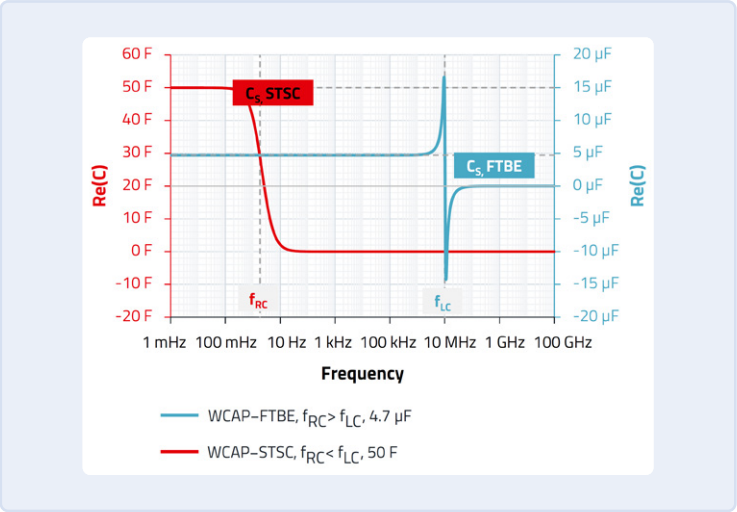


Figure 3. Spectres d'impédance $|\hat{Z}|$ pour WCAP-FTBE (haut) et WCAP-STSC (bas) calculés d'après le modèle standard.



▲

Figure 4. Spectre de capacité Re(Ĉ) calculé d'après le modèle standard. L'ordonnée rouge de gauche est pour WCAP-STSC (courbe rouge), l'ordonnée bleue de droite est pour WCAP-FTBE (courbe bleue).

valeurs mesurées. L'ajustement de C_S, R_{ESR}, L_{ESL} et R_{Leak} permet de calculer la réponse fréquentielle basique de n'importe quel condensateur. Les **figures 3** et **4** montrent les spectres d'impédance et de capacité de condensateurs de 4,7 µF et 50 F obtenus avec cette technique. Les angles de perte et de déphasage correspondants sont indiqués à l'annexe du lien [2]. Les paramètres de ces deux exemples sont les suivants :

- Supercondensateur (*WCAP-STSC*) : C_S = 50 F, R_{ESR} = 15 mΩ, L_{ESL} = 5 nH et R_{Leak} = 10 MΩ,
- Condensateur à film (*WCAP-FTBE*) : C_S = 4.7 µF, R_{ESR} = 5 mΩ, L_{ESL} = 5 nH et R_{Leak} = 10 MΩ.

Ces deux condensateurs, *WCAP-FTBE* (4,7 µF) et WCAP-STSC (50 F), appartiennent à la famille de condensateurs de Würth Elektronik eiSos. Sur les graphiques, C_S, R_{ESR}, L_{ESL} et R_{Leak} sont supposés constants et indépendants de la fréquence (**tableau 1**).

Les parties les plus significatives d'un spectre sont décrites par quatre fréquences propres :

Fréquence propre f_{RC} de l'élément R_{ESR}-C :

$$f_{RC} = \frac{1}{2 \cdot \pi \cdot R_{ESR} \cdot C_S}$$

Paramètres électriques	WCAP-FTBE	WCAP-STSC
C_S	4,7 µF	50 F
R_{ESR}	5 mΩ	15 mΩ
L_{ESL}	5 nH	5 nH
R_{Leak}	10 MΩ	10 MΩ

Tableau 1. Paramètres utilisés pour le calcul des spectres

Fréquence propre f_{LC} de l'élément L-C :

$$f_{LC} = \frac{1}{2 \cdot \pi \cdot \sqrt{L_{ESL} \cdot C_S}}$$

Fréquence propre f_{Leak} de l'élément R_{Leak}-C :

$$f_{Leak} = \frac{1}{2 \cdot \pi \cdot R_{Leak} \cdot C_S}$$

Fréquence propre f_{RL} de l'élément R_{ESR}-L :

$$f_{RL} = \frac{R_{ESR}}{2 \cdot \pi \cdot L_{ESL}}$$

Les **figures 3** et **4** montrent deux cas de figure majeurs :

Oscillation de Lorentz : $f_{RC} > f_{LC}$ pour C_S = 4.7 µF (courbe bleue) ;

Relaxation de Debye : $f_{RC} < f_{LC}$ pour C_S = 50 F (courbe rouge).

Sur la **figure 3**, les pointillés noirs et bleus indiquent les parties purement capacitives et inductives. f_{RC}, la fréquence propre de l'élément RC, est la fréquence à laquelle le condensateur peut être chargé et déchargé. L'inverse de cette fréquence correspond en gros à la durée d'une charge effectuée sous une tension idéale constante. Au-dessus de f_{RC}, le condensateur n'est plus complètement chargé (relativement à la tension maximale du signal).

Sur le spectre de capacité du supercondensateur, f_{RC} correspond à un point d'inflexion et le haut de la courbe forme un épaulement (**fig. 4**). Pour des fréquences inférieures à f_{RC}, la capacité peut être déduite de la courbe. Après le point f_{RC}, le spectre d'impédance de la **figure 3** inférieure montre un plateau au point R_{ESR}.

La fréquence propre f_{LC} de l'élément LC est la fréquence pour laquelle le couplage de l'inductance parasite et de la capacité déclenche un phénomène de résonance lorsque $f_{RC} > f_{LC}$ (**fig. 3** supérieure). Sous f_{RC} (pour des fréquences inférieures), le condensateur a un comportement capacitif, c.-à-d. peut stocker une charge électrique ; sa réponse est inductive au-dessus de f_{RC}. La résonance au point f_{LC} se traduit par un minimum étroit sur le spectre d'impédance de WCAP-FTBE (**fig. 3** supérieure). La valeur R_{ESR} correspond à ce minimum. En pratique, un condensateur ne doit pas être utilisé à f_{LC} ou au-dessus de cette valeur.

Le spectre de capacité du condensateur *FTBE* de 4,7 nF montre un pôle (**fig. 4**). Cette singularité reflète

une réponse physique, pas seulement un artefact de mesure : le système de mesure, composé du condensateur et de l'inductance parasite, se comporte comme un circuit résonnant, c.-à-d. comme un oscillateur (cf. [2] pour les détails).

f_{Leak} est la fréquence propre de l'élément R_{Leak}-C. En deçà de cette fréquence, le condensateur se comporte comme une résistance valant R_{Leak}. Cet effet est à peine visible sur le spectre, hormis pour des fréquences bien inférieures à 1 Hz ou avec une petite valeur de R_{Leak}.

La fréquence propre f_{RL} de l'élément R_{ESR}-L est la fréquence au-dessus de laquelle le condensateur se comporte comme une inductance de valeur L_{ESL} (**fig. 3** inférieure). Les cas où $f_{RC} < f_{LC}$ marquent le début de l'augmentation de l'impédance aux hautes fréquences.

Les deux exemples présentés ici montrent qu'un modèle relativement simple permet de décrire le comportement de condensateurs de grandes et petites capacités. Les spectres calculés fournissent toutes les caractéristiques que fourniraient des spectres mesurés. Les mesures apportent bien sûr plus d'informations, mais le modèle L-C-R permet de déterminer des paramètres utiles aussi bien à un travail d'ingénierie qu'à une interprétation basique des spectres. Les fréquences propres mentionnées ici forment à cet égard un bon outil d'analyse puisqu'elles « pointent du doigt » les parties utiles d'un spectre mesuré. La note d'application *ANP109* [2] examine plus en profondeur les spectres mesurés de quatre types de condensateurs :

- supercondensateur *WCAP-STSC*
- électrolytique à l'aluminum *WCAP-AIGB*
- condensateur à film *WCAP-FTBE*
- pastille céramique multicouche *WCAP-CSGP*

VF : Hervé Moreau — 230318-04

À propos de l'auteur

René Kalbitz a étudié la physique aux universités de Potsdam (Allemagne) et de Southampton (Royaume-Uni). Sa thèse de doctorat porte sur les semi-conducteurs et les isolants organiques. Il a poursuivi ses travaux de recherche à l'Institut Fraunhofer de Recherche Appliquée sur les Polymères. Il a rejoint Würth Elektronik en 2018 comme chef de produit pour les supercondensateurs, et supervise les projets de recherche et de développement dans le domaine des condensateurs.

LIENS

[1] Simulateur en ligne REDEXPERT : https://redexpert.we-online.com/redexpert/
[2] René Kalbitz, « Impedance spectra of different capacitor technologies ». Würth Elektronik AppNote ANP109 : https://www.we-online.com/en/support/knowledge/application-notes?d=anp109-impedance-spectra-of-different-capacitor-technologies

horloge NTP en CircuitPython

pourquoi utiliser ce langage de programmation ?

Michael Bottin (France)

Nous découvrons l'une des alternatives utiles à Python, conçue pour être suffisamment simple pour fonctionner sur des microcontrôleurs : CircuitPython. Ici, nous réalisons un projet avec un Raspberry Pi Pico W, qui récupère l'heure d'un serveur NTP et l'affiche sur un écran LCD.

Python est un des langages de programmation parmi les plus populaires. C'est un langage interprété ce qui rend la phase de développement plus rapide contrairement à un langage comme le C/C++ qui nécessite d'être compilé avant son exécution. En revanche, cette absence de compilation oblige à ce qu'un environnement d'exécution soit installé sur la cible au préalable.
Depuis plus de 30 ans, ce langage continue son avancée dans le monde des PC et du cloud. Il couvre des domaines comme le calcul scientifique, le développement web (*back-end*), le développement logiciel, l'écriture de scripts système, le Machine Learning, le Big Data (analyse et visualisation de données). Il est utilisé par de nombreuses compagnies comme Google (« Python où nous pouvons, C ++ où nous devons »), Youtube, Instagram, Spotify, Intel, Facebook, Dropbox, Netflix, Pixar, Reddit… Il était donc prévisible que ce langage fasse son entrée dans le monde de l'électronique embarquée. Mais pour cela, il a fallu créer une version allégée pour qu'elle puisse tourner sur un microcontrôleur mais également une version tournée vers le hardware pour qu'elle puisse piloter les différents périphériques disponibles dans un microcontrôleur (GPIO, PWM, I²C, SPI, UART…).
Deux versions ont vu le jour :

- MicroPython développé par Damien George dès 2013 [1]
- CircuitPython développé par Limor Fried (PDG d'Adafruit), Scott Shawcroft et beaucoup d'autres contributeurs dès 2017 [2].

Ces deux langages partagent tous deux la même implémentation du langage Python (CPython). CircuitPython est un dérivé open-source de MicroPython. Toute évolution de MicroPython est rapidement répercutée sur CircuitPython. Il y a déjà eu plusieurs articles sur l'utilisation de MicroPython dans cette revue, mais à ma connaissance, aucun sur CircuitPython.

Quel est l'intérêt de CircuitPython vis-à-vis de MicroPython ?

Soyons clair, pour un informaticien habitué au Python souhaitant utiliser ce langage dans le monde de l'embarqué, aucun. Il est même préférable d'utiliser dans ce cas MicroPython. Mais pour un maker, un étudiant ou un enseignant n'ayant pas de connaissances particulières en Python, le choix de CircuitPython peut être intéressant. La chaîne de développement est très conviviale, le langage apporte une couche d'abstraction supplémentaire et la majorité des informations liées à ce langage (documentation, bibliothèques, tutoriels, forum…) sont centralisées sur le web.
C'est pour cette raison que j'ai choisi d'utiliser CircuitPython pour cet article, afin d'offrir aux lecteurs désirant démarrer en Python sur microcontrôleur une porte d'entrée facile. La communauté autour de CircuitPython est également très dynamique. De nouvelles versions sortent régulièrement (version actuelle 8.x), de nouvelles bibliothèques également. Un grand nombre de cartes populaires à base de microcontrôleur supportent ce langage. CircuitPython peut également être employé sur des SBC comme Raspberry Pi, BeagleBone, Odroid ou encore Jetson via l'API Blinka.

Ambition de cet article

Cet article constitue une découverte de l'utilisation du langage CircuitPython à travers une simple application.
Je vais donc vous présenter comment vous pouvez très rapidement mettre en œuvre une horloge synchronisée sur le web grâce à CircuitPython. Le projet propose également de choisir l'heure locale d'une vingtaine de pays dans le monde. L'horloge se synchronisera grâce à une connexion wifi

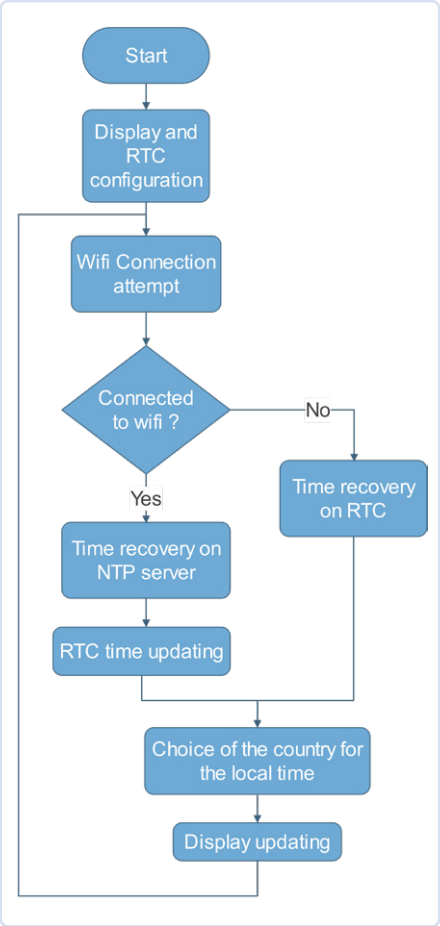


Figure 1. Algorigramme général du fonctionnement de l'horloge NTP.

(préconfigurée au niveau SSID et mot de passe) et récupérera son horodatage via un serveur NTP (*Network Time Protocol*) dédié. Cette horloge sera aussi en mesure de maintenir l'heure à jour en absence de réseau grâce à l'horloge temps réel (RTC) embarquée qui sauvegardera l'heure réseau lorsque celui-ci est disponible et qui fournira elle-même l'heure lorsque le réseau est indisponible. Une pile garantira également le maintien de l'heure de la RTC à jour en absence d'alimentation.

L'algorigramme général du fonctionnement de l'horloge est présenté à la **figure 1.**

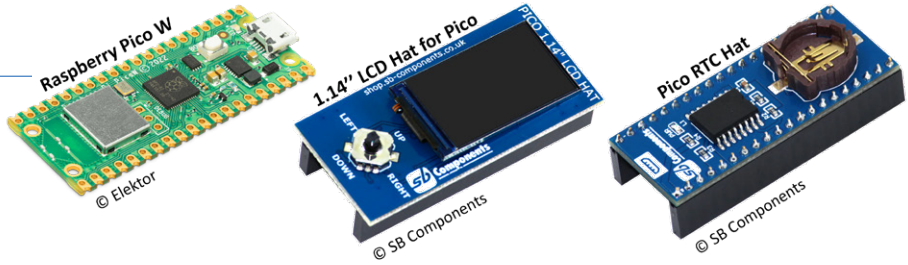


Figure 2. Modules utilisés.

Présentation du matériel

Un grand nombre de cartes de développement (et donc de microcontrôleurs) peuvent être facilement programmées avec le langage CircuitPython. J'ai choisi ici la carte Raspberry Pico W à la fois car elle est très populaire, peu coûteuse, disponible chez un grand nombre de revendeurs dont Elektor [3] et qu'elle dispose d'une interface Wifi. Toutefois, elle ne dispose ni d'un écran d'affichage, ni d'une horloge temps-réel. Il est donc nécessaire d'ajouter quelques composants supplémentaires pour notre projet.

Plutôt que de réaliser une carte dédiée, j'ai choisi ici d'assembler des modules du commerce pour réaliser ce projet. Trois modules empilés suffisent au projet (cf. **figure 2**) :

- Un module Raspberry Pico W (attention à bien prendre la version W qui dispose d'une interface Wifi)
- Un module d'affichage avec un écran LCD 240x135 pixels et un joystick pour la navigation (SB Components [4])
- Un module avec une horloge temps réel DS3231 (SB Components [5])

Remarque : le microcontrôleur RP2040 dispose également en interne d'une horloge temps réel RTC. Vous pourriez l'utiliser à la place du module *Pico RTC Hat*, cependant la RTC du RP2040 n'est pas aussi précise dans le temps et ne dispose pas d'une connexion vers une batterie de sauvegarde.

Schéma de câblage

Même si l'on utilise des modules du commerce qui facilitent la mise en œuvre de la partie hardware, on a besoin de connaître les connexions entre ces modules afin de pouvoir écrire les lignes de code du programme de notre application. La **figure 3** présente les interconnexions réalisées entre les différents modules ainsi que les noms donnés à ces connexions, noms qui seront réutilisés dans le programme. (Notez que le brochage indiqué dans la documentation GitHub de SB Components est incorrect au moment de la rédaction de cet article, mais leur code source indique l'affectation correcte des broches illustrée dans la **figure 3**).

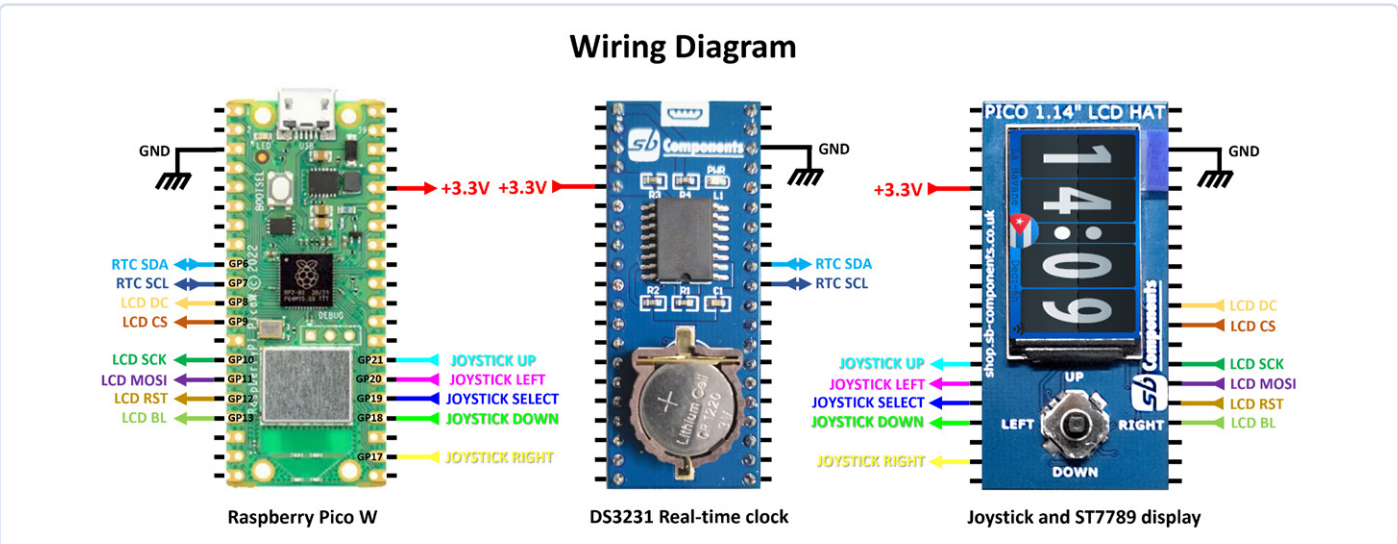


Figure 3. Interconnexions réalisées entre les différents modules.

Figure 4. Site web de CircuitPython.

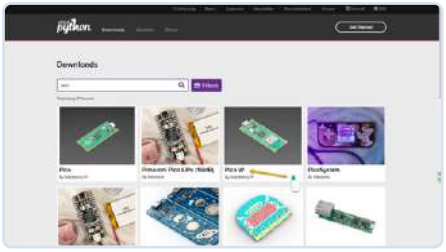


Figure 5. Sélection du Pico W parmi les cartes compatibles avec CircuitPython.

Figure 6. Téléchargement du fichier .UF2.

Installation CircuitPython

Le microcontrôleur RP2040 de la Raspberry Pico W n'intègre pas CircuitPython lorsque vous achetez le module. Avant toute chose, vous allez donc devoir le « flasher » sur la carte.

Tout comme il y a des versions différentes de compilateur C selon la cible utilisée, il y a autant de versions de CircuitPython qu'il y a de cartes qui le supportent. Au moment où ces lignes sont écrites, plus de 380 cartes du commerce sont compatibles avec CircuitPython. Il y a bien évidemment la plupart des cartes récentes d'Adafruit, mais également de Pimoroni, d'Expressif, de SeeedStudio, de Waveshare, de LilyGo, de Cytron, de DFRobot, de Wiznet…

La majorité d'entre elles utilisent des microcontrôleurs de chez Microchip (SAMD21/SAMD51), de chez Nordic (nRF52840), d'Expressif (ESP32) et de la Fondation Raspberry (RP2040). Vous devez donc télécharger la version de CircuitPython qui correspond à la carte Raspberry Pico W :

- Dans un navigateur, rendez-vous sur le site web de CircuitPython [6] (**figure 4**).
- Cliquez sur le lien *Downloads* puis recherchez la carte Pico W (**figure 5**).
- Téléchargez alors le fichier UF2 disponible sur la page affichée (**figure 6**).

Le microcontrôleur RP2040 de la carte Raspberry Pico W contient déjà un bootloader qui facilite l'installation de CircuitPython dans ce dernier. Voici la démarche pour installer la version de CircuitPython que vous venez de télécharger :

1. Déconnectez la carte Raspberry Pico W de votre ordinateur de travail
2. Maintenez le bouton *BOOTSEL* appuyé de la carte Raspberry Pico W pendant que vous la connectez à votre ordinateur via un câble micro-USB (compatible avec un transfert de données).
3. Un lecteur *RPI-RP2* doit apparaître dans votre explorateur de fichier.
4. Copiez alors le fichier UF2 que vous avez téléchargé sur le lecteur *RPI-RP2*.

Vous n'avez rien d'autre à faire. Une fois CircuitPython flashé sur la carte Raspberry Pico W, un lecteur *CIRCUITPY* doit apparaître dans votre navigateur.

Utilisation du lecteur *CIRCUITPY*

Voici quelques points importants à respecter lors de l'utilisation de ce lecteur *CIRCUITPY* :

- TIl s'utilise comme une clef USB. Vous pouvez ajouter, supprimer des fichiers/dossiers depuis votre explorateur de fichiers. Sa capacité représente la mémoire Flash disponible sur le microcontrôleur RP2040 pour y stocker votre code et vos ressources (images, audio…).
- Il n'y a pas de précaution à prendre lorsque vous branchez le Raspberry Pico W à votre ordinateur. **En revanche, il est plus que souhaitable de l'éjecter à la manière d'une clef USB lorsque vous voulez la débrancher ; faute de quoi, vous risquez de corrompre le système de fichiers.**
- En ouvrant le fichier *boot.txt*, vous pouvez contrôler quelle version de CircuitPython est installée sur la carte Raspberry Pico W.
- Dès que la carte Raspberry Pico W est alimentée via son port micro-USB (grâce à votre ordinateur ou un chargeur secteur), le code CircuitPython s'exécute. Mais seuls les fichiers nommés *code.py* ou *main.py* vont s'exécuter. Prenez bien soin de nommer votre fichier de travail *code.py*.
- Il est préférable (mais pas obligatoire) de copier les fichiers images dans un dossier *./images*, les fichiers de bibliothèques dans un dossier *./lib*,… Cela permet d'avoir une arborescence propre pour vos projets.

Utilisation de l'EDI

CircuitPython étant un langage interprété, il n'y a donc pas de phase de compilation. Si un fichier *code.py* contenant du code CircuitPython existe sur le lecteur CIRCUITPY, il s'exécute automatiquement. De ce fait, on pourrait tout simplement éditer ce fichier dans un éditeur de texte et l'enregistrer pour qu'il s'exécute. Toutefois, l'écriture d'un programme s'accompagne généralement d'une phase de débogage. L'utilisation d'un EDI permet alors d'accéder à des outils comme une console pour obtenir un retour sur ses erreurs.

L'EDI que l'on va utiliser ici est Mu Editor [7]. C'est une solution logicielle simple disponible gratuitement sous Windows, Mac OSX et Linux. Il est également possible d'utiliser Thonny, VS Code, Atom ou encore PyCharm en installant l'extension CircuitPython

La **figure 7** présente l'interface de l'EDI Mu Editor. La barre d'outils est réduite à l'essentiel :

- *Mode* : choix du langage de programmation. Veillez à bien être dans le mode *CircuitPython*.
- *New* : création d'un nouveau fichier vide
- *Load* : chargement d'un fichier existant. En mode *CircuitPython* et si votre carte Raspberry Pico W est bien branchée, le logiciel vous ouvre automatiquement le dossier du lecteur *CIRCUITPY*.
- *Save* : sauvegarde du fichier en cours (onglet actif). Un petit point rouge à côté du nom du fichier dans l'onglet vous indique qu'il y a eu des modifications depuis la dernière sauvegarde.
- *Serial* : affiche/masque la console. Il est préférable de toujours afficher la console (REPL) car c'est dans celle-ci que s'afficheront les messages d'erreurs ou les informations de votre programme.
- *Plotter* : affiche/masque un graphe déroulant permettant de visualiser des données numériques envoyées depuis le code.
- *Zoom-in* et *Zoom-out* : augmente ou réduit la taille de la police d'affichage.
- *Theme* : bascule entre un thème clair (*Day*), sombre (*night*) ou à fort contraste.
- *Check* : recherche les erreurs de votre code, même celles qui n'empêchent pas son exécution.
- *Tidy* : nettoie votre code en supprimant,

Figure 7. L'interface de l'EDI Mu Editor.

par exemple, les espaces inutiles ou les lignes vides inutiles.
- *Help* : aide en ligne.
- *Quit* : quitte l'éditeur.

Voici donc le déroulement classique du développement d'un programme dans l'EDI :
- Vous ouvrez le fichier *Code.py* du lecteur *CIRCUITPY*
- Vous modifiez le code dans la zone d'édition.
- Vous enregistrez les modifications du fichier.
- Vous observez le résultat de l'exécution dans la console. Vous repartez à l'étape n°2 si des erreurs sont survenues durant l'exécution.

Vous trouverez davantage d'informations à l'adresse suivante :
- *Start Here!* (concernant l'interface) [8]
- *What is a REPL?* (concernant la console REPL) [9]
- *Plotting Data with Mu* (concernant le graphe déroulant) [10]
- Raccourcis clavier [11]

Maintenant que ces informations sont données, vous allez pouvoir vous intéresser sur le projet, à savoir l'horloge NTP.

Test de la connexion wifi

CircuitPython gère le wifi en natif sur la carte Raspberry Pico W. Aucune bibliothèque supplémentaire n'est donc à installer. La connexion à un réseau wifi nécessite de fournir le SSID du réseau et son mot de passe. Pour éviter que ces informations apparaissent directement dans le code, CircuitPython utilise des variables d'environnement.
Un fichier *settings.toml* [12] situé à la racine du lecteur *CIRCUITPY* va donc contenir ces informations « secrètes ». On pourrait y stocker toute information confidentielle comme des clefs d'API. Dans notre cas, il ne contient que le *SSID* et le *PW* du réseau auquel vous souhaitez connecter votre Raspberry Pico W (cf. **figure 8**). (Remarque : ce fichier n'est pas éditable dans l'EDI, vous devez utiliser un éditeur de texte pour le créer et le remplir)
Le code pour se connecter au réseau est présenté dans le **listage 1.** Vous pouvez rapidement constater que le code est très compact :
- **Ligne 2** : importation de la bibliothèque *os* qui permet d'accéder aux

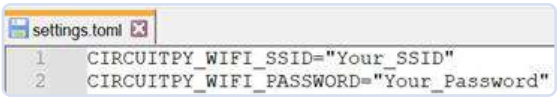


Figure 8. SSID et password dans le fichier settings.toml.

Listage 1. Code pour se connecter au réseau wifi

```
 1 # CircuitPython's own libraries
 2 import os
 3 import ipaddress
 4 import wifi
 5
 6 print()
 7 print("Connecting to Wi-Fi")
 8
 9 # connect to your Wi-Fi network with SSID/PASSWORD in 'settings.toml'
10 wifi.radio.connect(os.getenv('CIRCUITPY_WIFI_SSID'), os.getenv('CIRCUITPY_WIFI_PASSWORD'))
11
12 print("Connected to Wi-Fi")
13
14 # pings Google DNS server
15 ipv4 = ipaddress.ip_address("8.8.8.8")
16 print("Ping google.com: %f ms" % (wifi.radio.ping(ipv4)*1000))
```

Figure 9. Téléchargement des bibliothèques (la date figurant dans le nom du fichier sera évidemment différente.

variables d'environnement du fichier *settings.toml*.

- **Ligne 4** : importation de la bibliothèque *wifi* qui permet la connexion au réseau.
- **Lignes 6,7 et 12** : ces lignes ne sont pas obligatoires, elles permettent juste d'informer l'utilisateur dans la console.
- **Line 10** : une seule ligne suffit ici à la connexion physique au réseau.
- **Lines 3, 14–16** : ces lignes ne sont pas obligatoires, elles permettent de faire un petit test (*ping*) en interrogeant un serveur sur le web et ainsi vérifier que la connexion wifi fonctionne correctement puis afficher le résultat sur la console série.

Ajout des bibliothèques utiles

CircuitPython contient un grand nombre de bibliothèques fondamentales, mais il ne peut pas contenir toutes les librairies disponibles car cela saturerait inutilement la mémoire Flash du microcontrôleur. Il est donc nécessaire, selon les exigences de vos projets, d'installer des bibliothèques supplémentaires.

L'installation de bibliothèques supplémentaires se résume en CircuitPython à copier les fichiers correspondants sur le lecteur *CIRCUITPY*. Pour n'avoir à le faire qu'une seule fois, vous allez installer ici toutes les bibliothèques utiles pour la version finale du projet.

Voyons la démarche :

1. Créez un dossier *lib* sur votre lecteur *CIRCUITPY* s'il n'existe pas déjà.
2. Toutes les bibliothèques disponibles sont téléchargeables sous forme d'une archive unique sur le site de CircuitPython (remarque : elles sont aussi disponibles individuellement sur Github)
3. Sur la page du site, cliquez sur le lien *Libraries*.
4. Téléchargez alors l'archive correspondant à votre version de CircuitPython, 8.x dans notre cas (cf. **figure 9**).
5. Dézippez la où vous le souhaitez sur votre ordinateur.
6. Dans le dossier de l'archive dézippée, ouvrez le dossier *lib*.
7. Sélectionnez :
   a. Les dossiers :
      i. *adafruit_register*
      ii. *adafruit_imageload*
      iii. *adafruit_display_text*
   b. Les fichiers :
      i. *adafruit_debouncer.mpy*
      ii. *adafruit_ds3231.mpy*
      iii. *adafruit_ntp.mpy*
      iv. *adafruit_st7789.mpy*
      v. *adafruit_ticks.mpy*
8. Copiez l'ensemble de votre sélection dans le dossier *lib* du lecteur *CIRCUITPY*.

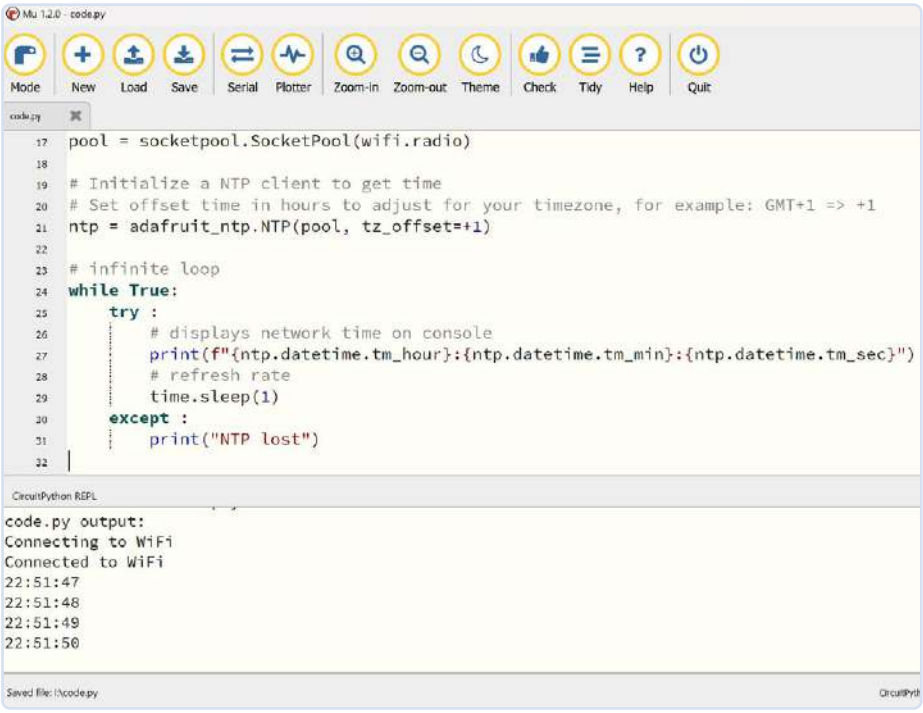


Figure 10. L'horloge NTP affichée dans la console série.

Remarque : CircUp est un outil écrit en Python qui permet de vérifier la version de vos bibliothèques, de les mettre à jour et d'installer leurs dépendances. [13].

Affichage de l'heure dans la console

Dans le précédent programme, vous avez connecté votre Raspberry Pico W au réseau wifi. Vous allez maintenant interroger un serveur NTP dédié pour qu'il vous fournisse la date et l'heure. Le code permettant la récupération de l'heure et son affichage dans la console (**figure 10**) est disponible dans le **listage 2**.

Voyons ce qui a été ajouté :

- **Lignes 5 et 18** : importation de la bibliothèque native *socketpool* et création d'un socket qui permettra le maintien de la connexion client-serveur.
- **Ligne 22** : instanciation d'un client NTP via le socket. L'attribut `tz_offset` permet de régler le décalage horaire entre le fuseau horaire GMT (*Greenwich Mean Time*) et votre fuseau horaire. Étant en France, mon fuseau horaire CET correspond au fuseau horaire GMT +1, d'où la valeur +1. Vous devez l'ajuster à votre fuseau. Le serveur par défaut est celui d'Adafruit (mais on peut le modifier) et le *timeout* est de 10 secondes par défaut.
- **Lignes 24-32** : dans une boucle infinie, on affiche l'heure dans la console au rythme du rafraîchissement souhaité (ici 1 seconde – ligne 29). Pour cela, on utilise l'attribut `ntp.datetime`. Celui-ci renvoie un tuple nommé de la classe `time` qui contient les informations de date et d'heure du serveur. On accède à chaque élément du tuple par son champ nommé [15]. On utilise l'instruction `try…except` pour intercepter une exception qui correspondrait à la non réception d'une donnée une fois que la durée de *timeout* est écoulée.

Listage 2. Code permettant la récupération de l'heure.

```
# CircuitPython own's libraries
import time
import os
import wifi
import socketpool

# CircuitPython external libraries
import adafruit_ntp

print("Connecting to WiFi")

# connect to your WiFi network with SSID/PASSWORD in 'settings.toml'
wifi.radio.connect(os.getenv('CIRCUITPY_WIFI_SSID'), os.getenv('CIRCUITPY_WIFI_PASSWORD'))

print("Connected to WiFi")

# Setting up a socket
pool = socketpool.SocketPool(wifi.radio)

# Initialize a NTP client to get time
# Set offset time in hours to adjust for your timezone, for example: GMT+1 => +1
ntp = adafruit_ntp.NTP(pool, tz_offset=+1)

# infinite loop
while True:
    try :
        # displays network time on console
        print(f"::")
        # refresh rate
        time.sleep(1)
    except :
        print("NTP lost")
```

Figure 11. L'horloge NTP affichée sur l'écran LCD.

Affichage de l'heure sur l'ecran LCD

Pour cette étape, nous allons utiliser l'écran plutôt que la console pour l'affichage de l'heure. CircuitPython dispose d'une bibliothèque native générique appelée *displayio* qui permet de fournir des méthodes et des attributs communs pour tous les écrans supportés par CircuitPython.
Ici, l'écran embarqué sur le module LCD couleurs a les caractéristiques suivantes :

- Il utilise le protocole SPI
- Sa définition est de 240x135 pixels
- Son *driver* est le ST7789

Vous pouvez télécharger le code via le lien à la fin de l'article. Il serait long de le détailler ici, mais voici les principales étapes pour aboutir au résultat de la **figure 11** :

1. Bibliothèques natives à CircuitPython :
   board : pour accéder aux noms des broches du Raspberry Pico W
   displayio : pour la gestion des graphiques
   busio : pour le bus de communication SPI
   terminalio : pour disposer d'une police de caractères pour l'affichage
2. Bibliothèques externes à CircuitPython :
   adafruit_st7789 : pour la gestion de l'écran LCD
   adafruit_display_text : pour l'affichage de zone de texte sur l'écran

On configure ensuite tout l'affichage :

1. On crée le bus de communication SPI `SPI_bus`.
2. On crée le bus `display_bus` qui relie le Raspberry Pico W à l'écran LCD. Il contient le bus SPI mais également deux signaux supplémentaires (*Command* et *chip_select*).
3. On crée enfin l'écran `display` en lui fournissant le bus précédent et en précisant sa définition (`width=135`, `height=240`) et les offsets en x (`rowstart=40`) et y (`colstart=53`). Les valeurs de ces deux derniers paramètres s'expliquent par le fait que la définition maximale du driver ST7789 est 320x240 et que celle de notre écran n'est que de 240x135 (Cf. **figure 12**).
4. En CircuitPython, tout élément graphique doit appartenir à un groupe. On commence donc par créer un groupe `display_group` puis une zone de texte `time_label`. On ajoute la zone de texte au groupe. Et enfin on affiche le groupe sur l'écran.
5. Pour mettre à jour l'affichage de l'heure sur le LCD, il suffit, dans la boucle infinie, de modifier l'attribut `text` de notre zone de texte `time_label` en y copiant la chaîne de caractères que l'on affichait précédemment dans la console.

Listage 3. Code pour la RTC.

```
# SPDX-FileCopyrightText: 2021 ladyada for Adafruit Industries
# SPDX-License-Identifier: MIT

# Simple demo of reading and writing the time for the DS3231 real-time clock.
# Change the if False to if True below to set the time, otherwise it will just
# print the current date and time every second.  Notice also comments to adjust
# for working with hardware vs. software I2C.

import time
import board
import busio
import adafruit_ds3231

i2c = busio.I2C(scl=board.GP7, sda=board.GP6)
rtc = adafruit_ds3231.DS3231(i2c)

# Lookup table for names of days (nicer printing).
days = ("Monday", "Tuesday", "Wednesday", "Thursday", "Friday", "Saturday", "Sunday")

# pylint: disable-msg=using-constant-test
if True:  # change to True if you want to set the time!
    #                     year, mon, date, hour, min, sec, wday, yday, isdst
    t = time.struct_time((2023, 03, 09, 14, 58, 15, 3, -1, -1))
    # you must set year, mon, date, hour, min, sec and weekday
    # yearday is not supported, isdst can be set but we don't do anything with it at this time
    print("Setting time to:", t)  # uncomment for debugging
    rtc.datetime = t
    print()
# pylint: enable-msg=using-constant-test

# Main loop:
while True:
    t = rtc.datetime
    # print(t)     # uncomment for debugging
    print(
        "The date is {} {}/{}/{}".format(
            days[int(t.tm_wday)], t.tm_mday, t.tm_mon, t.tm_year
        )
    )
    print("The time is {}:{:02}:{:02}".format(t.tm_hour, t.tm_min, t.tm_sec))
    time.sleep(1)  # wait a second
```

Test de l'horloge temps réel RTC

On pourrait s'arrêter là car notre horloge synchronisée sur le web fonctionne bien, mais que se passe-t-il si la connexion au réseau est interrompue ou si l'on coupe l'alimentation du montage pendant un certain temps ?

Pour éviter de perdre l'heure actuelle, une horloge temps réel va être ajoutée. Couplée à une batterie de sauvegarde (pile CR1220), elle maintiendra l'heure à jour pendant des années en absence d'alimentation via le port micro-USB. Cette RTC utilise le circuit populaire DS3231 qui emploie le bus I²C pour sa communication avec le module Raspberry Pico W.

Avant de combiner la RTC avec le serveur NTP, vous allez pouvoir la tester seule. C'est la bibliothèque *adafruit_ds3231* qui est nécessaire. Comme la majorité des bibliothèques CircuitPython, elle dispose d'exemples et de tutoriels en ligne [16].

Dans le code proposé dans ce tutoriel, vous devez juste modifier les premières lignes conformément au **listage 3**. En effet, de nombreuses cartes Raspberry Pico W ne supportent pas le protocole I²C [17]. Il faut donc préciser quelles sont celles qui sont physiquement reliées au composant RTC DS3231 (cf. **figure 3**). Lors de l'exécution, vous devriez obtenir dans la console un résultat similaire à celui de la **figure 13**.

Code final

La fonction première du projet est maintenant réalisée. Le code final est disponible en téléchargement sur [24]. la **figure 14** présente l'aspect général de l'affichage sur l'écran. On peut constater que l'aspect

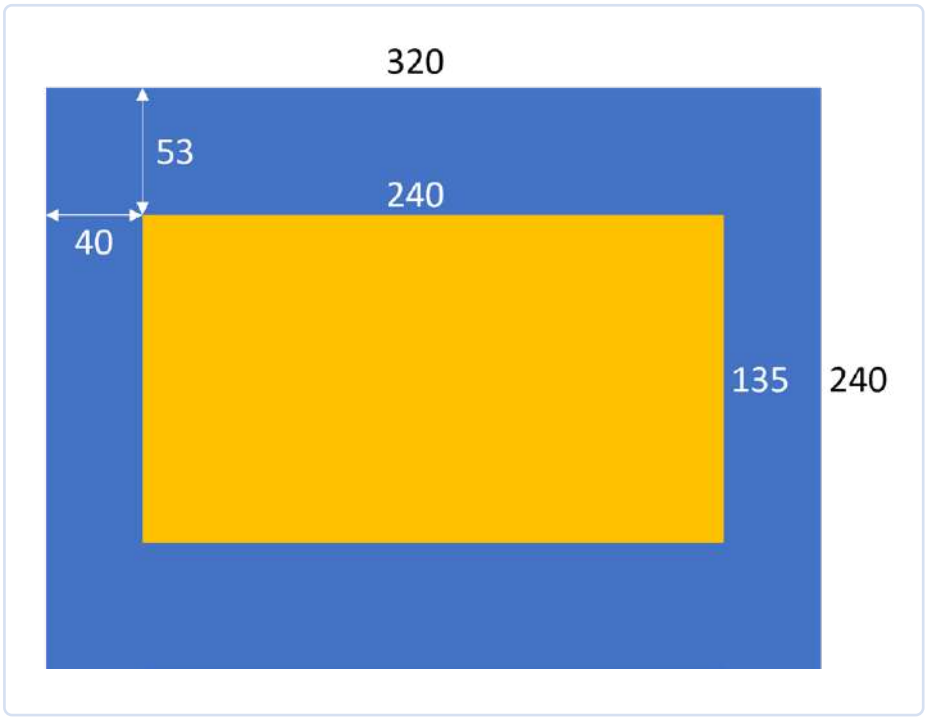


Figure 12. Offset de l'écran ST7789.

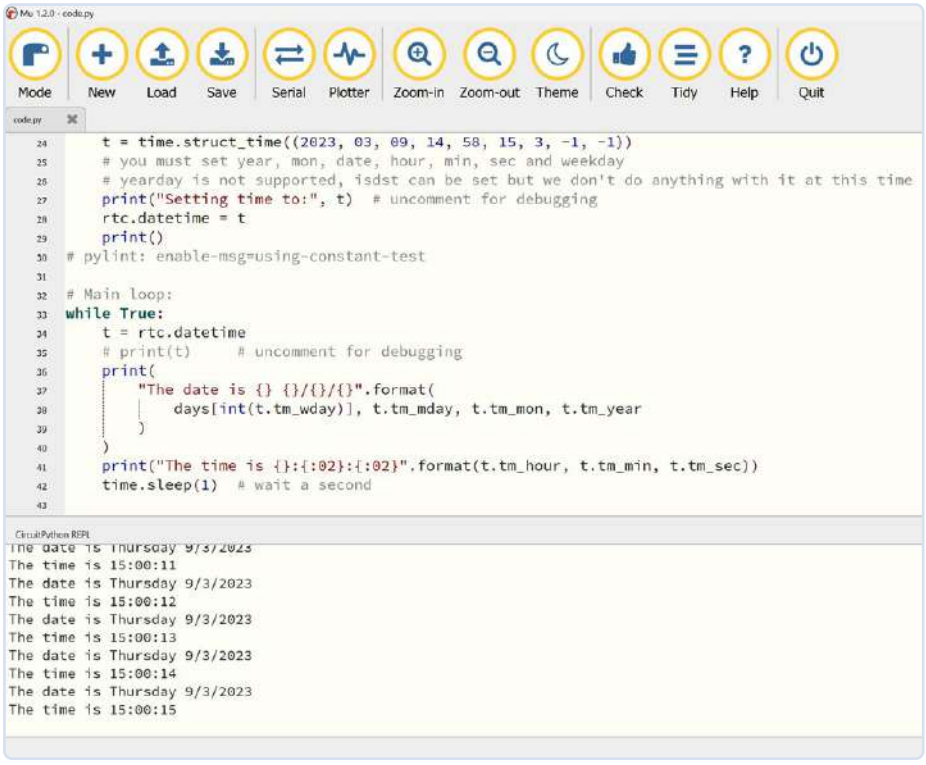


Figure 13. Résultats de l'horloge RTC.

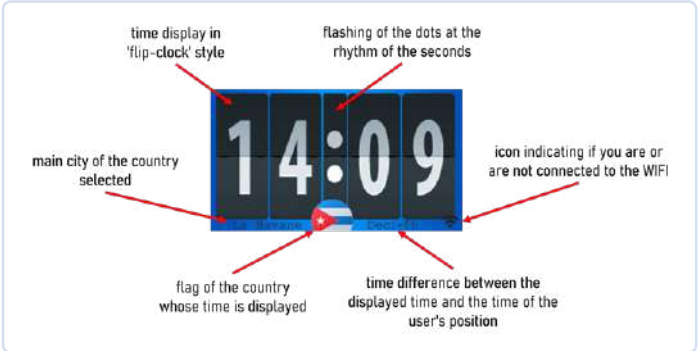


Figure 14. Interface finale.

Figure 15. Spritesheet des digits.

graphique a été amélioré et que des éléments nouveaux figurent sur l'écran. Encore une fois, une explication complète du code serait fastidieuse. Un tutoriel, disponible sur [18], fournit toutes les bases nécessaires.

Qu'est-ce qui change vis-à-vis des précédents codes ?

- Le code de gestion de l'horloge temps réel RTC a été mixé avec celui de la récupération de l'heure via le serveur NTP.
- L'affichage de l'heure se fait maintenant à l'aide d'images bitmap préparées dans un logiciel dédié. On utilise notamment la bibliothèque *adafruit_imageload* pour les charger en mémoire. Pour éviter de charger une image à chaque changement de digit (risque de ralentissement), on utilise ici une *spritesheet* comme en animation (**figure 15**). On définit ensuite quelle portion de l'image on désire afficher. On procède de même pour les points clignotants.
- Avec le joystick disponible sur le module *1.14" LCD hat* for Pico, on peut sélectionner un pays et voir ainsi l'heure actuelle dans ce pays et le décalage horaire engendré. Pour éviter les rebonds du joystick, on utilise ici une bibliothèque *adafruit_debouncer*. Cette dernière a besoin de la bibliothèque *adafruit_ticks* pour fonctionner.

Figure 16. Spritesheet des drapeaux.

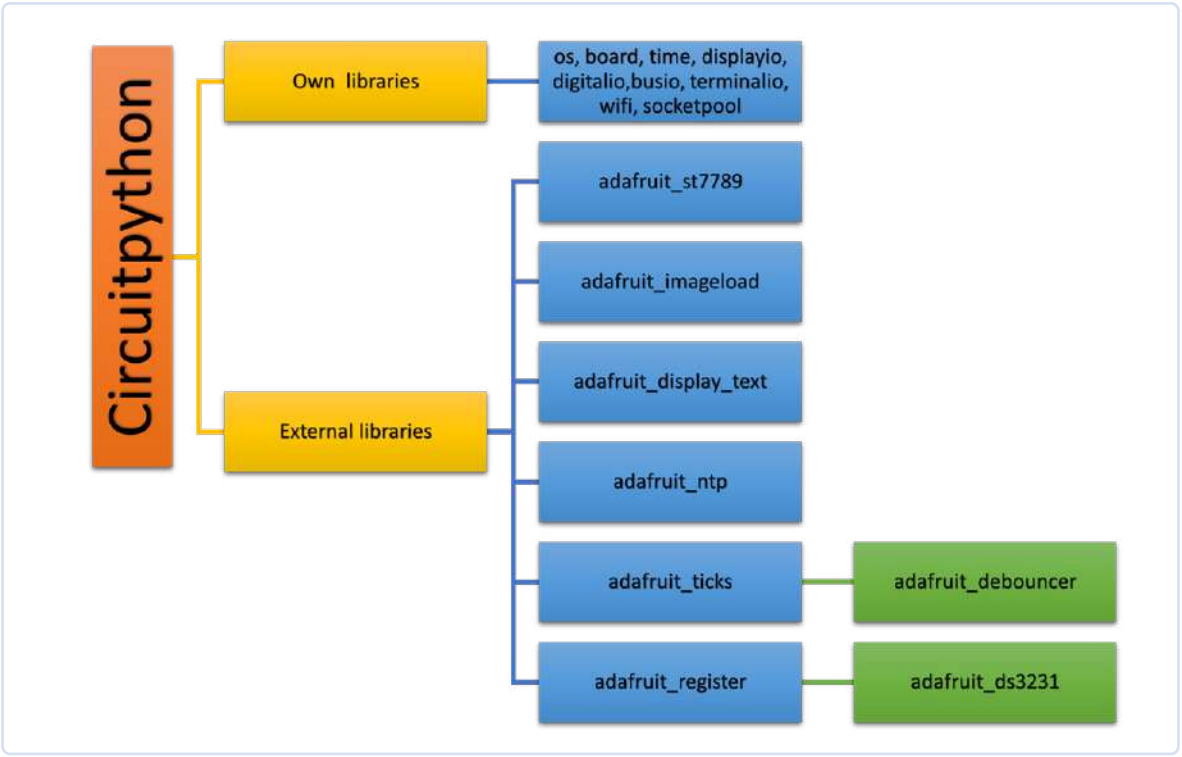


Figure 17. Arborescence des bibliothèques.

- L'affichage du drapeau utilise aussi un bitmap sous forme de *spritesheet* avec 25 pays disponibles (**figure 16**).
- Les textes correspondant à la ville et au décalage horaire utilisent des labels comme on l'a fait lorsque l'on a affiché l'heure sur le LCD précédemment.

L'arborescence des bibliothèques internes et externes utilisées dans le projet est représentée sur la **figure 17**.

Conclusion

On pourrait bien sûr imaginer des évolutions à ce projet comme utiliser un écran LCD avec une définition plus importante (en se limitant toutefois à 320x240 vis-à-vis de la puissance du Raspberry Pico W) ou ajouter des alarmes sonores utilisateur ou encore afficher les données météos actuelles pour notre position… Il est assez aisé de programmer ces améliorations en CircuitPython à l'aide des documentations des bibliothèques en ligne, de très nombreux tutoriels existants sur le site d'Adafruit…

N'hésitez pas à vous lancer dans un projet personnel en utilisant ce langage CircuitPython, vous découvrirez par vous-même sa simplicité d'utilisation.

- Téléchargement : CircuitPython pour le Raspberry Pi Pico W [18]
- Téléchargement : bibliothèques externes [19]
- Documentation : bibliothèques natives de CircuitPython [20]
- Documentation : bibliothèques externes [21]
- Tutoriels de démarrage en CircuitPython [22], [23]

Il y a également un livre publié chez Elektor que j'ai écrit en 2020 avec la version 5.3 de CircuitPython (**figure 18**). Certes, il y a eu beaucoup de nouveautés depuis cette édition, mais les bases du langage demeurent inchangées et le code se transpose facilement à d'autres modules comme la Raspberry Pico (W) (c'est toute la force de CircuitPython avec son *unified hardware API*). Et si vous êtes bloqué, vous pouvez toujours m'envoyer vos questions à mon adresse mail.

220633-04

Figure 18. Livre « Initiation au language circuitpython et à la puce nRF52840 «.

Des questions, des commentaires ?

Si vous avez des questions ou des propositions/suggestions d'articles, envoyez un courriel à l'auteur (michael.bottin@univ-rennes.fr), ou contactez Elektor (redaction@elektor.fr).

Produits

- **Raspberry Pi Pico RP2040 W**
  https://elektor.fr/20224
- **Michael Bottin, *Initiation au langage CircuitPython et à la puce nRF52840* (livre en français)**
  https://elektor.fr/19523

LIENS

[1] MicroPython : https://fr.wikipedia.org/wiki/MicroPython
[2] CircuitPython : https://en.wikipedia.org/wiki/CircuitPython
[3] Raspberry Pico W chez Elektor : https://elektor.fr/raspberry-pi-pico-rp2040-w
[4] Module d'affichage avec un écran LCD pour le Pico : https://shop.sb-components.co.uk/products/1-14-lcd-hat-for-pico
[5] Module avec une horloge temps réel pour le Pico : https://shop.sb-components.co.uk/products/pico-rtc-hat
[6] Site web de CircuitPython : https://circuitpython.org/
[7] Mu Editor : https://codewith.mu/en/download
[8] Interface de Mu Editor : https://codewith.mu/en/tutorials/1.2/start
[9] Console REPL : https://codewith.mu/en/tutorials/1.2/repl
[10] Graphe déroulant : https://codewith.mu/en/tutorials/1.2/plotter
[11] Raccourcis clavier : https://codewith.mu/en/tutorials/1.2/shortcuts
[12] Fichier TOML : https://fr.wikipedia.org/wiki/TOML
[13] L'outil Circup : https://learn.adafruit.com/keep-your-circuitpython-libraries-on-devices-up-to-date-with-circup/
[14] UTC (Coordinated Universal Time): https://timeanddate.com/worldclock/timezone/utc
[15] Classe time : https://docs.python.org/3/library/time.html#time.struct_time
[16] Tutoriels et exmples avec la bibliothèque adafruit_ds3231 :
https://learn.adafruit.com/adafruit-ds3231-precision-rtc-breakout/circuitpython
[17] Brochage du Pico W : https://datasheets.raspberrypi.com/picow/PicoW-A4-Pinout.pdf
[18] Version de CircuitPython pour le Raspberry Pico W : https://circuitpython.org/board/raspberry_pi_pico_w/
[19] Bibliothèques externes de CircuitPython : https://circuitpython.org/libraries
[20] Bibliothèques natives de CircuitPython : https://docs.circuitpython.org/en/latest/shared-bindings/index.html#modules
[21] Tutoriels de démarrage en CircuitPython : https://docs.circuitpython.org/projects/bundle/en/latest/drivers.html
[22] Tutoriels de démarrage en CircuitPython 1 : https://learn.adafruit.com/welcome-to-circuitpython/circuitpython-essentials
[23] Tutoriels de démarrage en CircuitPython 2 : https://learn.adafruit.com/welcome-to-circuitpython/
[24] Téléchargement du logiciel : https://elektormagazine.fr/220633-04

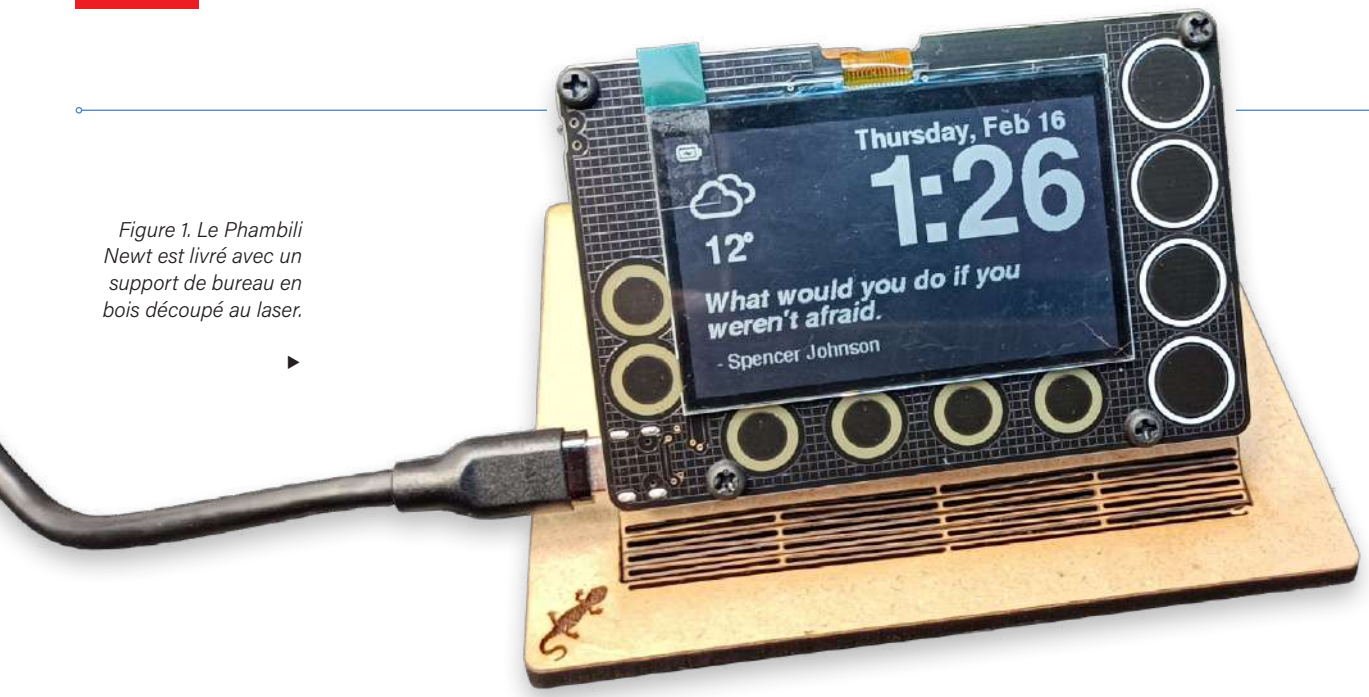


Figure 1. Le Phambili Newt est livré avec un support de bureau en bois découpé au laser.

▶

construisez un écran IdO sympa

avec le Phambili Newt

Clemens Valens (Elektor)

Découvrez le Phambili Newt, un module d'affichage compact et personnalisable qui offre bien plus qu'on ne le pense. Examinons ses caractéristiques uniques, des fonctionnalités de base aux possibilités passionnantes d'applications programmables par l'utilisateur, ce qui en fait un outil intéressant pour les passionnés d'appareils IdO.

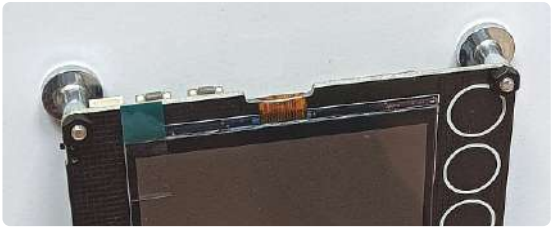
Le Phambili Newt est un écran qui peut être fixé au mur, alimenté par batterie, toujours allumé, qui va chercher des informations sur Internet et les afficher. Le module est légèrement plus grand qu'une carte de crédit et comporte 10 pavés tactiles ainsi qu'un afficheur E-Ink de 2,7 pouces (240 x 400 pixels). Derrière l'écran se trouve un microcontrôleur ESP32-S2 que vous pouvez programmer avec Arduino, CircuitPython, MicroPython [1] ou ESP-IDF. Bien qu'il soit destiné à des applications où il est constamment allumé, il est doté d'un minuscule interrupteur à glissière qui permet de l'éteindre.

Une batterie est censée alimenter le Newt, mais elle n'est pas incluse dans le kit. Heureusement, vous pouvez aussi le faire fonctionner à partir d'un chargeur de téléphone de type USB-C. Selon la documentation, une batterie Li-Po avec une capacité minimale de 500 mAh permettrait à l'appareil de fonctionner jusqu'à deux mois entre deux charges. La batterie doit être équipée d'un connecteur JST à deux contacts.
Outre le module Newt, le kit comprend également un support en bois découpé au laser et du matériel de montage. Le support vous permet de placer l'appareil sur votre bureau, par exemple, mais il est un peu bancal lorsque vous appuyez sur les touches. Une option plus stable consiste à monter les quatre pieds magnétiques et à fixer le Newt sur un réfrigérateur ou sur toute autre surface métallique.

Que fait le Phambili Newt ?

Après le déballage, le Newt ne fait pas grand-chose puisqu'il attend qu'une connexion wifi soit configurée premièrement. Dès la mise sous tension, il donne des instructions sur la façon de le connecter à votre réseau. C'est assez facile, mais un peu lent, et j'ai observé plusieurs redémarrages de l'appareil avant qu'il ne se connecte à mon réseau. Lorsqu'il s'est finalement connecté, il a affiché la date et l'heure, puis s'est bloqué.

FigureFigure 2. Avec ses pieds magnétiques, vous pouvez fixer le Newt sur un réfrigérateur ou sur tout autre objet métallique. ▶



J'avais remarqué que la version du micrlogiciel chargé sur mon Newt était la v0.0.11, et j'ai donc cherché une version plus récente. Après avoir chargé la dernière version (v1.1.15) [2], le Newt fonctionne comme prévu. La mise à jour du micrologiciel est facile : connectez le Newt à un ordinateur et copiez le nouveau fichier du micrologiciel sur le disque externe qui est créé. Avec le micrologiciel approprié, le Newt se connecte rapidement à mon réseau et affiche l'heure, la date et les informations météorologiques de l'endroit où je me trouve. L'affichage des données météorologiques alterne avec une citation toutes les trois minutes.
Une pression sur la touche supérieure droite ouvre un menu en bas de l'écran. Il y a trois « pages » avec alarme et minuterie, des informations sur la météo et la qualité de l'air, un calendrier et d'autres choses que vous pourriez trouver utiles ou amusantes. Le menu permet également d'accéder aux réglages et à la mise à jour du micrologiciel. Malheureusement, la vis de fixation gêne un peu la touche du menu (tout comme la vis proche de la touche inférieure gauche).

Créer vos propres applications pour le Phambili Newt

Même si la fonctionnalité de base du Newt est intéressante, ce n'est probablement pas la raison pour laquelle vous en voudriez un. Sa véritable force réside dans le fait qu'il est possible de le pirater, pour ainsi dire. Le code source du micrologiciel est disponible sur GitHub, ainsi que des instructions sur la façon de configurer l'EDI Arduino [3] pour écrire vos propres applications Newt.
Un connecteur I²C au format Qwiic (SparkFun) vous permet de connecter des capteurs et d'autres extensions au Newt, le transformant en un véritable appareil IdO au lieu d'une simple horloge connectée.

Conclusion

L'ESP32-S2 est un microcontrôleur puissant, même s'il ne dispose que du wifi seulement et pas du Bluetooth. L'écran noir et blanc est très agréable et rapide, contrairement aux afficheurs E-Ink normaux. L'affichage est instantané. La combinaison de ces deux éléments en fait un module très sympa avec de nombreuses possibilités d'application, en particulier dans le domaine de l'IdO à faible consommation d'énergie. Le connecteur d'extension I²C offre encore plus d'options. ⏮

VF : Laurent Rauber — 230345-04

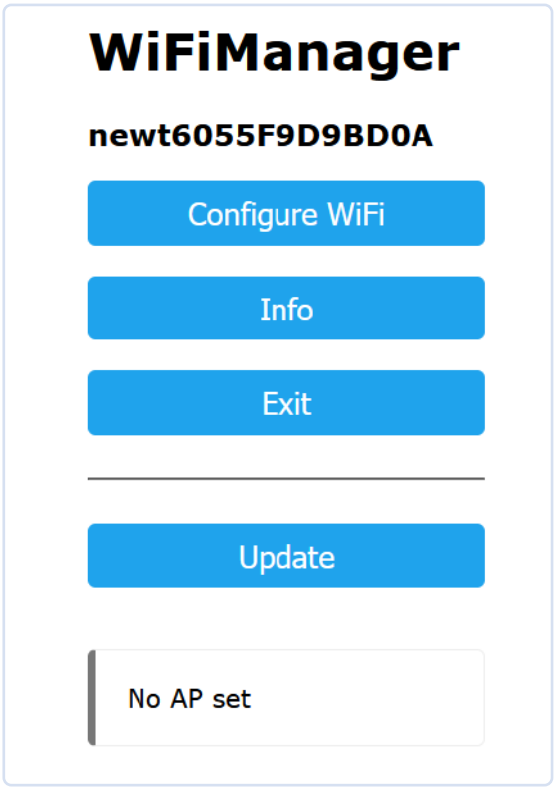


Figure 3. Voici l'affichage après avoir mis le Newt sous tension pour la première fois. ◀

Des questions, des commentaires ?

Envoyez un courriel à (Clemens.valens@elektor.com).

Produit

- **Phambili Newt Afficheur IdO 2,7 pouces (alimenté par ESP32-S2)**
  https://elektor.fr/20230
- **SparkFun Environmental Combo Breakout - CCS811/BME280 (Qwiic)**
  https://elektor.fr/19580

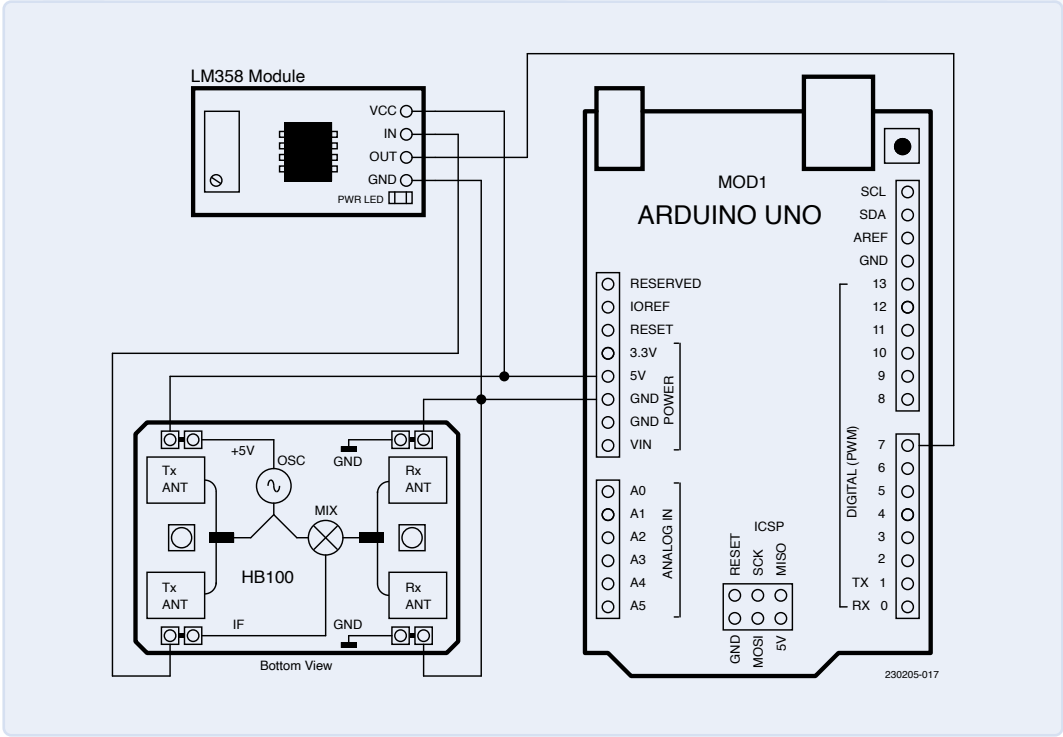
LIENS

[1] Günter Spanner, « MicroPython pour l'ESP32 et ses copains (partie 1) » : https://www.elektormagazine.fr/magazine/elektor-180/59780
[2] Dernière version du micrologiciel sur Newt : https://phambili-pub.s3.amazonaws.com/Newt.ino_latest.bin
[3] Comment configurer l'EDI Arduino : https://github.com/Phambili-Tech/Newt_Display/wiki/Arduino-Setup

détecteur de mouvement

Doppler HB100

théorie et pratique



Stefano Lovati (Italie)

La détection des mouvements d'êtres humains, d'animaux ou d'objets utilise des capteurs appropriés dits « de mouvement ». Nous étudions ici un type de capteur spécifique, apprécié pour ses capacités uniques et sa valeur pédagogique : le capteur à micro-ondes. Ce dispositif de détection de mouvement repose sur l'effet Doppler, déjà utilisé dans les systèmes radar modernes.

Le détecteur à infrarouge fonctionne par détection des infrarouges émis par l'objet tandis que le capteur à µ-ondes est émetteur et analyse les réflexions sur l'objet. Le capteur à µ-ondes a l'avantage de pouvoir détecter d'autres objets que des corps chauds. En outre, il n'est pas affecté par la température ambiante, possède une large plage de mesure et une grande sensibilité. Grâce à ses caractéristiques, il est très utilisé dans l'industrie, le transport, le contrôle automatique des portes, les détecteurs de stationnement et comme compteur de vitesse. Comme il peut détecter différents types d'objets, dans de nombreuses applications réelles, combiné à un autre type de capteur il permet une détection ciblée et infaillible. Par ex., associé à un capteur de présence à infrarouge (PIR), le capteur à µ-ondes permet de déterminer à coup sûr le passage ou la présence d'une personne, en éliminant les possibles sources de perturbation.

Remarques sur l'effet Doppler

Considérons une source sonore : l'effet Doppler désigne la variation de fréquence (hauteur) du son perçu par un auditeur quand, soit elle se rapproche ou s'éloigne de lui, soit l'auditeur se rapproche ou s'éloigne d'elle. En pratique, chacun de nous a pu expérimenter cet effet, en remarquant comment le son produit par une sirène (que pour simplifier, nous supposons monotone et de fréquence fixe) est modifié lorsque la source se rapproche ou s'éloigne de nous. Cet effet, formalisé par le physicien autrichien Christian Doppler, s'applique à toutes les ondes qu'elles soient de nature électromagnétique (radio, lumière, rayons X), matérielle (son, vagues), etc.). Ce changement apparent de fréquence entre source d'onde et récepteur est déterminé par le mouvement relatif entre eux.

Pour mieux comprendre l'effet Doppler, supposons qu'une source émette un son de longueur d'onde et de fréquence constantes. Si la source et le récepteur sont immobiles, le récepteur « entend » une fréquence sonore identique à celle que la source émet. Le récepteur perçoit chaque seconde le nombre exact de périodes que la source produit. En revanche, en cas de déplacement relatif de l'un vers l'autre, le récepteur en perçoit plus car il va à la rencontre des ondes émises en se rapprochant de la source. Il interprétera cette variation comme un son de fréquence plus élevée. Inversement, si la source et le récepteur s'éloignent l'un de l'autre, ce dernier percevra moins de périodes sonores chaque seconde et donc un son de fréquence plus basse.

Voyez la représentation graphique de ce concept (**figure 1)**. Si le véhicule s'approche de la source (l'émetteur), l'onde réfléchie a une fréquence plus élevée ; cependant, si le véhicule s'éloigne, l'onde réfléchie a une fréquence moins élevée que l'onde d'origine.

Quelques équations

Pour un système radar, la formule ci-après donne la valeur en Hertz du décalage de fréquence f_D appelé fréquence Doppler.

$$f_D = \frac{2 \cdot v}{\lambda} \quad \textbf{(Équation 1)}$$

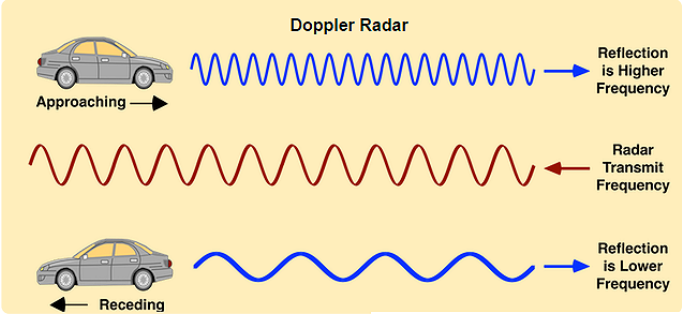


Figure 1. Exemple d'application de l'effet Doppler. (Source : physicsopenlab.org, CC BY 4.0)

où *v* est la vitesse de l'objet détecté en m/s et **λ** la longueur d'onde en mètres. Cette formule n'est valable que si la vitesse v est radiale, c.-à-d. sans composante transversale, comme c'est le cas si l'objet à détecter (celui qui réfléchit le signal) et la source (l'émetteur) se déplacent latéralement. Dans le cas le plus général, l'équation ci-dessus prend la forme ci-après.

$$f_D = \frac{2 \cdot v}{\lambda} \cdot \cos \propto \quad \textbf{(Équation 2)}$$

où **α** est l'angle que forme la direction du signal transmis/réfléchi avec la direction du mouvement de l'objet à détecter. On remarquera que si l'objet à détecter ne se déplace que perpendiculairement à la direction du signal émis, l'effet Doppler est nul ($f_D = 0$). C'est pourquoi, dans les combats aériens de type *Top Gun*, où l'on utilise des radars sophistiqués exploitant précisément l'effet Doppler, les pilotes menacés par un radar tentent toujours d'adopter une trajectoire perpendiculaire à celle de l'ennemi, ce qui complique la détection radar et le suivi de leurs position et vitesse.

Dans le cas spécifique d'un radar transmettant des ondes électromagnétiques (comme celui du capteur utilisé pour cet article), la formule ci-après exprime la valeur absolue (le module) de la fréquence Doppler f_D :

$$|f_D| = \frac{2 \cdot v_r}{\lambda} = \frac{2 \cdot v_r \cdot f_{TX}}{c_0} \quad \textbf{(Équation 3)}$$

où v_r st la vitesse radiale, f_{TX} la fréquence du signal transmis et c_0 la vitesse de la lumière.

Le module HB100

Cet article étudie les caractéristiques de l'un des capteurs de détection de présence à effet Doppler les moins chers : le *HB100*. En bref, c'est un module intégré monté sur un circuit imprimé de taille extrêmement compacte comprenant un émetteur de µ-ondes (bande X), un circuit récepteur, un mélangeur et toute la section RF. Pour rassurer nos lecteurs, je précise que même si les fréquences concernées sont assez élevées (environ 10 GHz), il n'y a pas de risque pour la santé en raison de la très faible puissance d'émission. Le principe de fonctionnement de ce module, vendu quelques euros par les grands distributeurs de composants, est très intéressant sur le plan technique et pédagogique. En effet, le module utilise un circuit superhétérodyne sur lequel reposent les systèmes radar modernes et son schéma classique se retrouve dans de nombreux récepteurs RF. La **figure 2** montre l'aspect extérieur du module, qui est très compact, avec le boîtier métallique qui sert d'écran de protection pour la partie RF. La **figure 3** montre l'arrière du module qui quant à lui, sert d'antenne émettrice-réceptrice

The HB100 miniature motion sensor is a e détecteur de mouvement miniature HB100 est un module émetteur-récepteur Doppler opérant à 10,525 GHz, donc en bande X. Il abrite un oscillateur à résonateur

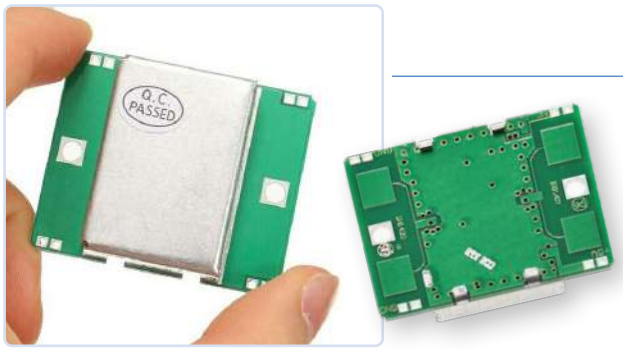


Figure 2. Vue de dessus du module HB100. (Source : [6])

Figure 3. Vue inférieure du module HB100. (Source : [7])

diélectrique (DRO) et une paire d'antennes, gravées directement sur la carte imprimée. Ce module est idéal pour réduire l'incidence des fausses alarmes dans les systèmes de détection d'intrusion, en particulier s'il est associé à un capteur infrarouge passif (PIR) classique. Les systèmes d'ouverture automatique de porte et compteurs de vitesse de véhicule peuvent aussi utiliser ce capteur. Les avantages principaux du module HB100 sont les suivants :

- détection de mouvement sans contact ;
- mesure non affectée par la température, l'humidité, le bruit, l'air, la poussière ; adaptation à des conditions environnementales particulièrement difficiles ;
- excellente immunité aux interférences radio ;
- faible puissance rayonnée = absence de risque pour l'homme et conformité aux réglementations de la Federal Communication Commission (FCC) ;
- distance de détection élevée, jusqu'à 20 m ;
- détection non limitée aux humains, mais étendue aux objets froids ;
- haute directivité des ondes radio ;
- faible consommation d'énergie ;
- capacité à fonctionner en mode CW (*Continuous Wave*, c.-à-d. transmission continue du signal radio) et en mode Pulse (transmission de courtes impulsions périodiques) ;
- compacité, faible épaisseur.

Nous passons à l'analyse détaillée du module en lien avec son schéma-bloc (**figure 4**). Commençons par l'oscillateur qui, comme déjà indiqué, est un DRO accordé à la fréquence de 10,525 GHz. Un disque céramique, généralement en titanate de baryum ($Ba_2Ti_9O_{20}$) sert de chambre de résonance pour l'énergie RF. La fréquence de résonance dépend de sa taille et de sa forme. C'est justement la fréquence du signal que le module envoie aux deux antennes rectangulaires gravées sur le circuit imprimé et visibles du côté gauche de la **figure 4**. Tout obstacle se trouvant sur le chemin du faisceau émis le réfléchira avec une différence de fréquence fonction du mouvement de l'objet par rapport au module d'émission. Aux vitesses ordinaires, cette différence reste faible. La manipulation d'un signal RF en bande X est très ardue, sur le plan matériel, mais surtout pour le logiciel. La solution à cet écueil est justement apportée par le circuit superhétérodyne et, plus précisément, par le mélangeur RF (v. **fig. 4**). Sa fonction est de « combiner » le signal transmis avec celui reçu par les deux antennes rectangulaires, visibles à droite (v. **fig. 4**). Si un mélangeur reçoit deux signaux d'entrée de fréquences f_{e1} et f_{e2}, il produit deux signaux de sortie de fréquences f_{s1} et f_{s2} respectivement égales à $f_{e1} + f_{e2}$ et $f_{e1} - f_{e2}$. Ainsi, dans notre cas, deux signaux l'un à 20 GHz environ (somme des fréquences émise et reçue) et l'autre en BF (différence des dites fréquences). Ce mécanisme résout élégamment notre problème, car il transforme le problème de la mesure d'un signal en bande X en la mesure d'un signal BF facilement gérable par un µcontrôleur courant à faible coût, par ex. Arduino. En mouvement radial, selon la vitesse de l'objet on aurait par ex. 1 kHz à 185 km/h (vitesse très élevée) et 30 Hz à 5,5 km/h (homme qui marche).

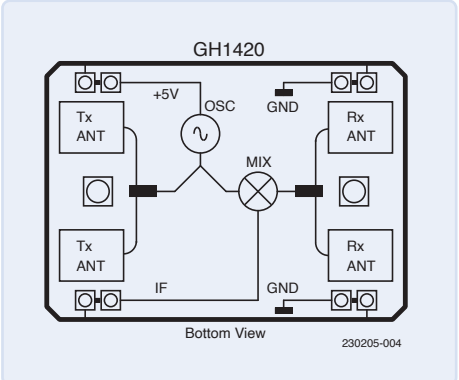


Figure 4. Schéma-bloc du capteur. (Source : [5])

Le signal appelé FI (Fréquence Intermédiaire) en sortie du mélangeur est justement le signal de différence. Ce processus courant en RF, est aussi dit abaisseur de fréquence (down-conversion en anglais) : son but est de réduire une fréquence élevée pour simplifier sa manipulation. La fréquence ainsi obtenue est appelée « fréquence intermédiaire » pour la distinguer de la fréquence d'origine, dite de « bande de base ».

L'examen de la **figure 4** révèle aussi la présence de 4 broches seulement. Elles sont nécessaires pour connecter le module à un µcontrôleur ou à un circuit de mesure : 2 broches de masse, 1 broche pour l'alimentation positive en 5 V et 1 broche pour le signal de sortie FI en BF. Le but est de traiter un signal dont la fréquence n'est que de quelques Hz ou kHz, en laissant la tâche ardue du traitement matériel du signal µ-ondes au module HB100. Il reste cependant un problème : le signal de sortie du capteur a une faible amplitude. Nous verrons plus loin comment le traiter.

Pour une utilisation pratique, le module doit être monté avec les antennes (v. **figure 3**) dirigées vers la zone à couvrir par la sortie rayonnée, en les orientant de manière à obtenir la meilleure couverture. Les diagrammes de rayonnement des antennes, en azimut et en élévation sont donnés (v. **figure 5**).

Signal de sortie

La broche IF représente le signal de sortie du module, correspondant au décalage de fréquence déterminé par la vitesse et la direction de tout objet éclairé par l'antenne. L'amplitude du signal de décalage de fréquence Doppler, est proportionnelle à l'énergie reçue et réfléchie en partie par l'objet. Elle vaut quelques µvolts (µV). Pour cette raison un amplificateur BF à gain élevé est normalement connecté à la broche

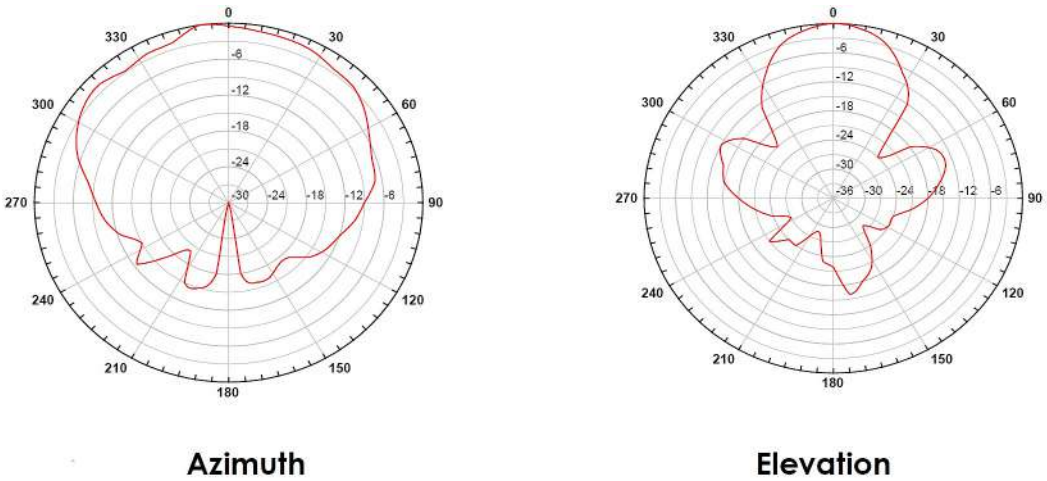


Figure 5. Diagrammes de rayonnement de l'antenne. (Source : [1])

FI afin d'amplifier le signal de sortie. Par ailleurs, la fréquence du décalage Doppler est proportionnelle à la vitesse du mouvement. Rappelons l'ordre de grandeur : une marche humaine type (5 à 6 km/h en direction du capteur) produit un décalage de fréquence de l'ordre de 30 Hz. La tension BF sur la broche de sortie IF est proportionnelle à l'intensité du signal reçu (RSS). Les équations données ci-dessus permettent de calculer la fréquence Doppler. Si le sujet éclairé par le capteur approche ou recule **le long** de la ligne radiale, la formule précédente est simplifiée et prend la forme suivante, où v_r est la vitesse radiale, exprimée en km/h **:**

$$f_D = 19.49 \cdot v_r \quad \textbf{(Équation 4)}$$

Les caractéristiques techniques du module principale sont :

- tension de fonctionnement : 5 V ± 0,25 V ;
- courant type absorbé en mode transmission continue : 60 mA maximum, 37 mA ;
- taille : 61,2 x 61,2 mm ;
- portée de détection : entre 2 m et 16 m ;
- fréquence d'émission : 10,525 GHz ;
- précision de fréquence : 3 MHz ;
- puissance de sortie min. : 13 dBm EIRP ;
- niveau des harmoniques : < -10 dBm ;
- courant moyen type : 2 mA ;
- largeur minimale de l'impulsion d'émission : 5 µs ;
- cycle de fonctionnement minimal : 1 %

Circuit de traitement du signal

La sortie du capteur HB100 (broche IF), est un signal BF reflétant le décalage Doppler de fréquence du signal reçu. La difficulté est qu'un signal non amplifié de quelques µV d'amplitude apparaît sur cette sortie. Dans les applications pratiques, il est obligatoire de l'amplifier suffisamment, de préférence en insérant un filtre passe-bas éliminant toutes les fréquences parasites au-delà de quelques centaines de Hertz. La fiche technique [1] et la note d'application [2] du capteur sont très utiles à cet égard. Cette dernière, présente deux schémas possibles de circuit de conditionnement du signal. Le 1er (v. **figure 6**), s'applique au mode d'émission en continu (CW) et est basé sur l'AOP à gain élevé *LM324* de Texas Instruments.

Le 2è circuit, (v. **figure 7**), s'applique au mode Impulsion, dans lequel l'émetteur émet des impulsions successives selon une fréquence de répétition donnée et avec un rapport cyclique donné (en anglais PRF,

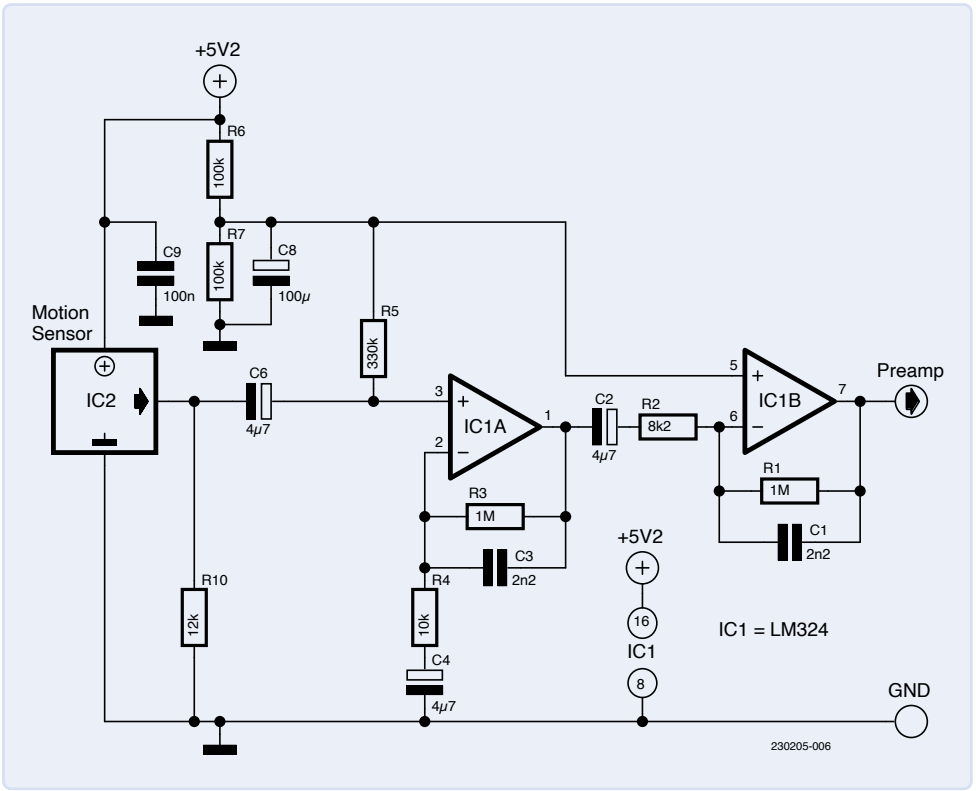


Figure 6. Circuit de conditionnement pour le mode CW. (Source : [5])

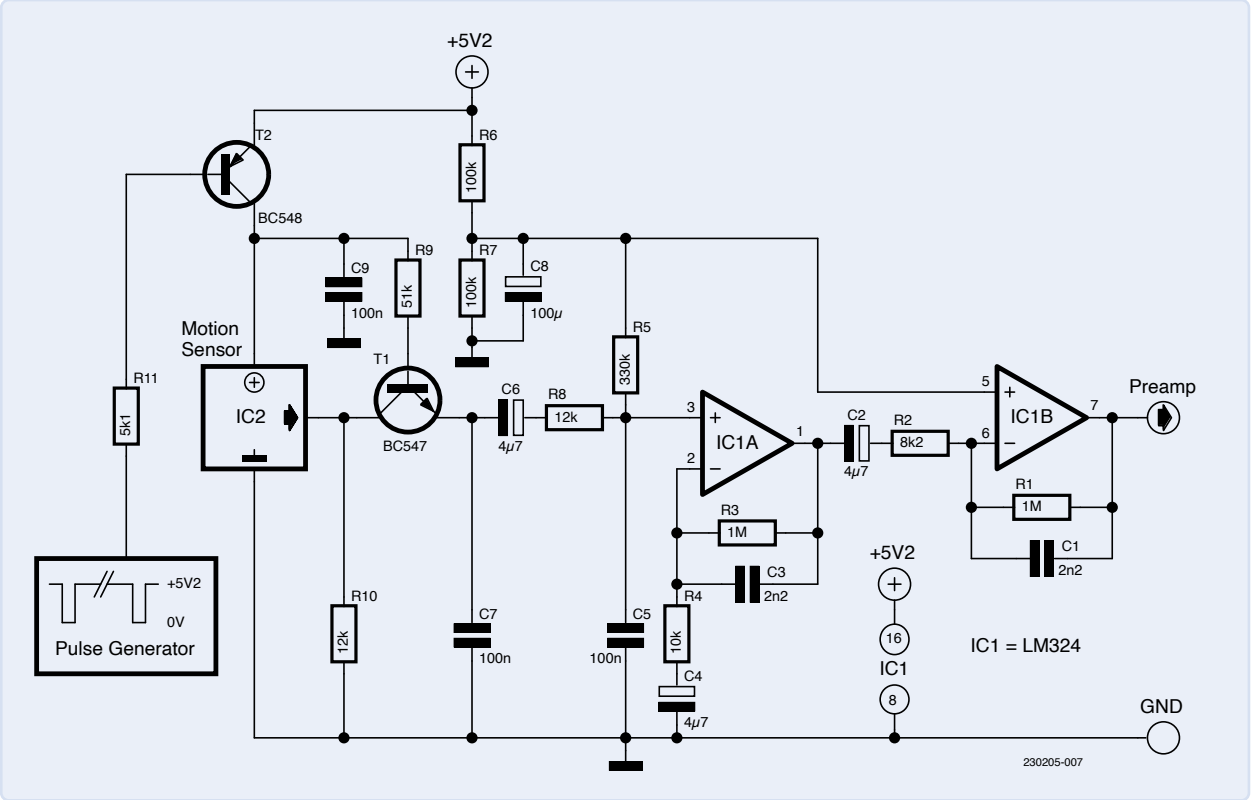


Figure 7. Circuit de conditionnement pour le mode pulsé. (Source : [5])

abréviation de *Pulse Repetition Frequency*). À ce sujet, le fabricant du HB100 suggère d'utiliser une PRF de 2 kHz et un rapport cyclique de 4 %.

Outre l'AOP à gain élevé LM324, ce circuit utilise deux transistors. Comme on le voit en bas à gauche (**figure 7**), le mode impulsion peut être produit en agissant directement sur l'alimentation du module, de manière à respecter la PRF et le rapport cyclique suggérés ci-dessus par le fabricant. C.-à-d. que dans ces conditions, le transistor de commutation ajouté en série est passant toutes les 0,5 ms (2 kHz) et le reste seulement 4 % de ce temps soit 0,02 ms.

Toutefois, pour simplifier l'utilisation du capteur, un AOP à usage général, a été choisi. Il est livré monté sur un circuit imprimé avec un connecteur et un potentiomètre de réglage du gain. Le module, très bon marché et facilement disponible, est basé sur le LM358, (**figure 8**). Le LM358 fournit une amplification monovoie avec un gain variable entre 1× et 100×, réglable à l'aide du potentiomètre : tourner la vis dans le sens horaire diminue le gain ; tourner la vis dans le sens antihoraire augmente le gain. En outre, le module est équipé d'un témoin LED de présence de l'alimentation du module. L'AOP LM358 peut traiter des signaux jusque 700 kHz (donc bien au-dessus de la bande qui nous intéresse) et peut être alimenté avec une tension comprise entre 3 V et 32 V. Le module comporte 4 broches d'interface : *VCC* , *GND*, *Signal In* et *Signal Out* (**figure 8**).

Essais avec Arduino

Pour l'essai du capteur Doppler HB100, une classique carte Arduino UNO et le module amplificateur LM358 (ou bien, tout autre amplificateur à grand gain et faible bruit, à voie et alimentation uniques) conviennent. En exécutant un croquis spécifique, une carte Arduino UNO peut détecter la fréquence du signal fourni par le module, et le cas échéant en déduire la vitesse à laquelle la « cible » potentielle se déplace. Le croquis en question utilise une bibliothèque spécifique développée pour la mesure précise des fréquences. En effet, pour mesurer une fréquence en bande audio ou inférieure, il faut déterminer la période du signal d'entrée avec une grande précision. Pour ce faire, la bibliothèque utilise un module *Counter and Capture* (Compteur et capture) très précis, fourni par l'architecture matérielle du µcontrôleur ATmega. La bibliothèque renvoie la période mesurée sous forme de nombre entier avec 1/16 µs de résolution. Déduire la fréquence, se résume à diviser la fréquence de l'horloge par la valeur renvoyée par la bibliothèque. Dans notre cas, nous fixons la fréquence d'horloge à 16 000 400 pour compenser les possibles imprécisions de la carte Arduino.

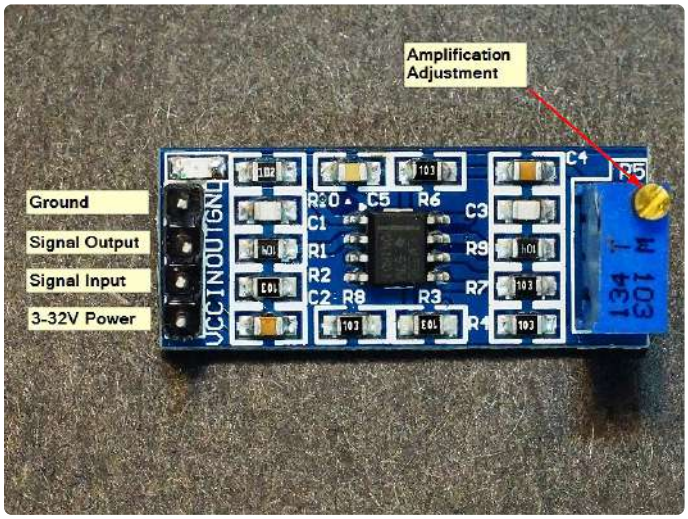


Figure 8. Le module d'amplification LM358.

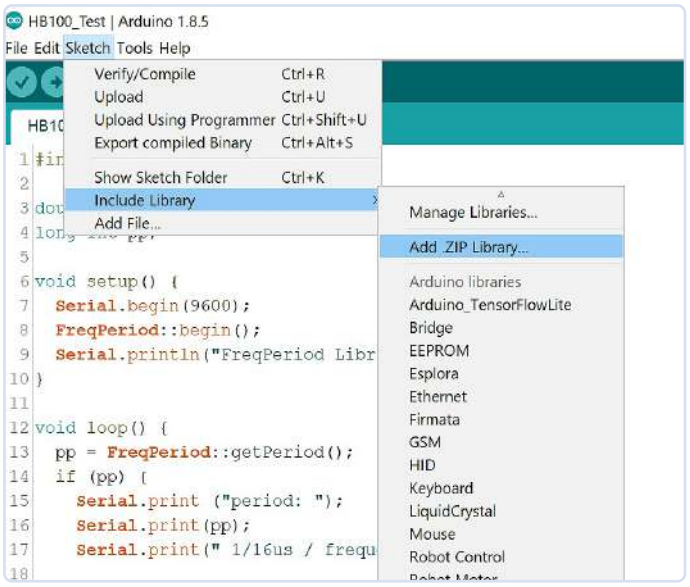


Figure 9. Installation de la bibliothèque FreqPeriod.

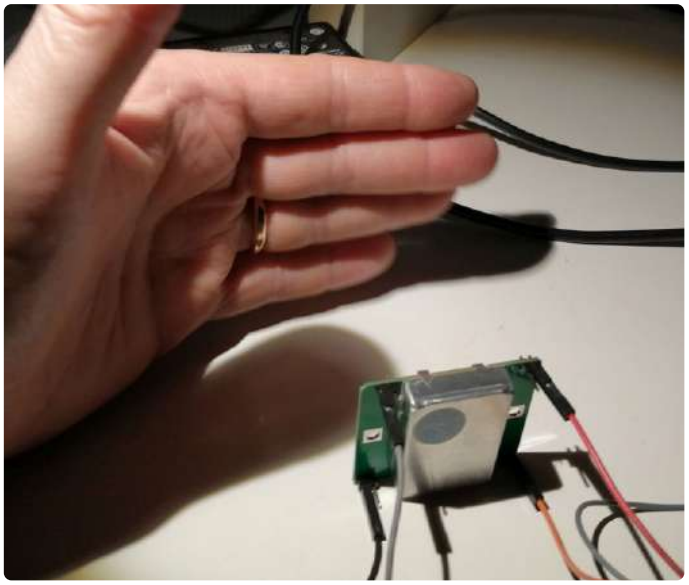


Figure 10. Mouvements rapides de la main devant l'antenne du capteur.

Cette bibliothèque n'est pas incluse dans la bibliothèque officielle Arduino, mais elle s'installe facilement dans l'EDI une fois le fichier .zip disponible à l'adresse [3] téléchargé. À cet effet, ouvrez l'EDI Arduino, sélectionnez l'élément de menu *Sketch Include Library Add .ZIP Library*..., (v. **figure 9**). Puis sélectionnez le fichier *.zip* de la bibliothèque téléchargée ci-avant. Fermez et redémarrez l'EDI : la nouvelle bibliothèque y est alors disponible.

Le croquis est présenté ici (**listage 1**). Il est très compact et relativement simple. La fonction `setup()`, initialise la ligne série (à 9600 bps) et la bibliothèque *FreqPeriod* par la méthode `begin()`. La boucle `loop()`, acquiert en 1er la période du signal d'entrée en appelant la méthode `getPeriod()`. Elle calcule et affiche ensuite les valeurs correspondantes de la fréquence (en Hertz) et de la vitesse radiale (en km/h).

Pour vérifier le fonctionnement du croquis, il faut connecter le capteur HB100 et le module amplificateur LM358 à la carte Arduino. Les connexions sont visibles sur l'**illustration de tête** de cet article.

Mettons ensuite le circuit sous tension et activons le moniteur série de l'EDI Arduino et observons les résultats. En déplaçant par ex. la main (v. **figure 10**) devant l'antenne du capteur HB100 (dont on se souvient qu'elle est située à l'arrière de la carte imprimée, côté opposé au conteneur métallique), nous devrions voir une variation de la vitesse calculée par le croquis.

La sortie envoyée via l'interface série est illustrée (v. **figure 11**).

Le calcul de cette vitesse, permet de concevoir des algorithmes complexes pour des applications relatives au fonctionnement des portes (battantes ou coulissantes), à l'allumage automatique de l'éclairage des escaliers, à l'ouverture des portes piétonnes, à la détection

Listage 1. Croquis Arduino.

```
#include <FreqPeriod.h>
double lfrq;
long int pp;

void setup() {

   Serial.begin(9600);
   FreqPeriod::begin();
   Serial.println("FreqPeriod Library Test");

}

void loop() {

   pp = FreqPeriod::getPeriod();

   if (pp) {
     Serial.print ("period: ");
     Serial.print(pp);
     Serial.print(" 1/16us / frequency: ");
     lfrq = 16000400.0 /pp;
     Serial.print(lfrq);
     Serial.print(" Hz ");
     Serial.print(lfrq/19.49);
     Serial.println( " km/h ");
   }

}
```

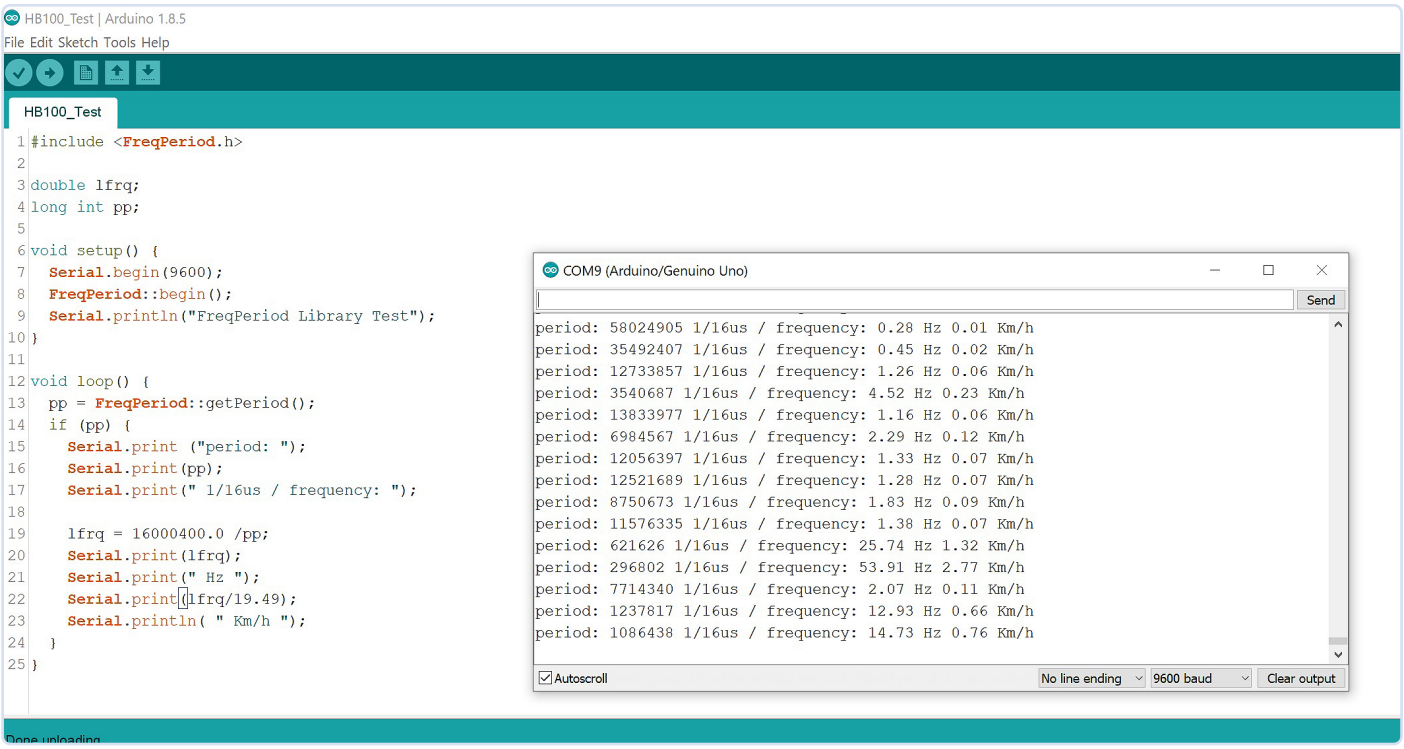


Figure 11. Sortie sur le moniteur série.

d'intrusion, à la vidéo-surveillance, etc. Le module fournit une assez bonne sensibilité s'il fonctionne dans sa limite de portée, soit environ 20 mètres. On peut améliorer les performances en utilisant un circuit amplificateur de gain plus élevé que celui offert par le LM358 (100×).

Conclusion

Cet article avait pour objet de présenter un capteur très intéressant et peu connu, le capteur Doppler de mouvement HB100. Particulièrement bon marché et facilement interfacé avec un µcontrôleur, le HB100 permet d'apprendre et d'expérimenter les techniques et concepts principaux de détection de signaux sur lesquels reposent les radars commerciaux sophistiqués et coûteux. Les applications de ce capteur ne manquent pas : il ne fait aucun doute que de nombreux électroniciens réfléchissent déjà à d'autres développements futurs.

VF : Yves Georges — 230205-04

Des questions, des commentaires ?

Si vous avez des questions techniques, n'hésitez pas à envoyer un courriel à l'équipe éditoriale d'Elektor (redaction@elektor.fr).

À propos de l'auteur

Fort de son diplôme d'ingénierie électronique obtenu au Politecnico di Milano, Stefano entama une carrière de développeur de micrologiciels et de logiciels. Au fil des ans, il vécut de l'intérieur la transformation progressive du monde embarqué, depuis les premiers µprocesseurs 8 bits - programmables uniquement en assembleur - jusqu'aux plus récents Soc, FPGA, DSP et logiques programmables aux performances et caractéristiques exceptionnelles. Il s'intéresse à tout ce qui touche technologie et électronique et consacre ses loisirs en partie à l'étude de nouveaux composants et à la réalisation de petits projets.

Produits

- **YDLIDAR X2 Lidar - Télémètre laser 360 degrés (8 m)**
  https://elektor.fr/18941
- **Arduino Uno Rev3**
  https://elektor.fr/15877
- **Arduino Uno Mini (édition limitée)**
  https://elektor.fr/20098

LIENS

[1] Fiche technique du module HB100 : https://limpkin.fr/public/HB100/HB100_Microwave_Sensor_Module_Datasheet.pdf
[2] Note d'application HB100 : https://limpkin.fr/public/HB100/HB100_Microwave_Sensor_Application_Note.pdf
[3] Bibliothèque de mesures de fréquence : https://github.com/Jorge-Mendes/Agro-Shield/tree/master/OtherRequiredLibraries/FreqPeriod
[4] Logiciel sur la page web de ce projet : https://elektormagazine.fr/230205-04
[5] Schémas des circuits : https://mantech.co.za/Datasheets/Products/MSAN-001_AGILSENSE.pdf
[6] Source de la photo de la vue de dessus : https://mantech.co.za/Datasheets/Products/HB100_RADAR.pdf
[7] Source de la photo de vue de dessous : https://kuongshun-ks.com/uno/uno-sensor/hb100-microwave-doppler-radar-wireless-module-moti.html

guide de programmation bare-metal (1)

pour STM32 et autres microcontrôleurs

Sergey Lyubka (Irlande)

Vous souhaitez programmer des microcontrôleurs et interagir avec leurs niveaux matériels pour mieux comprendre leur fonctionnement ? Ce guide destiné aux développeurs vous aidera à démarrer en utilisant simplement le compilateur GCC et un manuel de référence. Les notions apprises ici vous aideront à mieux comprendre le fonctionnement des cadres tels que Cube, Keil et Arduino. Dans ce guide en deux parties, nous utiliserons le contrôleur STM32F429 de la carte Nucleo-F429ZI, mais, vous pouvez facilement appliquer les connaissances acquises à d'autres microcontrôleurs.

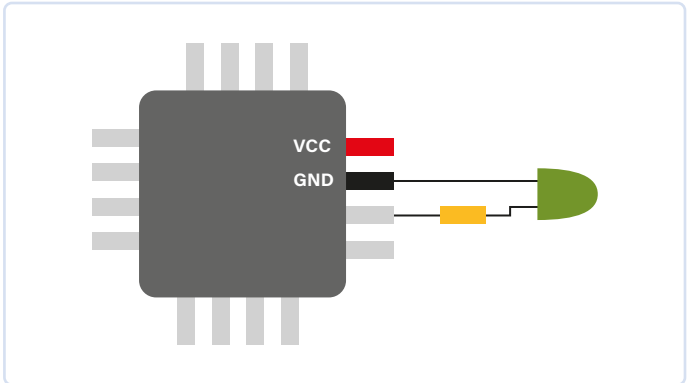


Figure 1. Le code du micrologiciel peut appliquer une tension de niveau haut ou bas sur une broche de signal, ce qui fait clignoter une LED.

Un microcontrôleur (µC, ou MCU) est un micro-ordinateur. Il est généralement doté d'un processeur, d'une mémoire vive, d'une mémoire flash pour stocker le code et d'un ensemble de broches. Certaines broches sont utilisées pour l'alimentation du microcontrôleur, généralement désignées par GND (masse) et VCC. D'autres broches sont utilisées pour communiquer avec le microcontrôleur en appliquant une tension de niveau haut ou bas à ces broches. L'un des exemples de communication les plus simples est de relier une LED à une broche : l'une des bornes de la LED est reliée à la masse (GND) et l'autre est reliée à une broche de signal avec une résistance de limitation de courant en série. Le micrologiciel peut appliquer une tension de haut ou bas niveau sur une broche de signal, ce qui fait clignoter la LED (**figure 1**).

Matériel et outils nécessaires

Tout au long de ce guide, nous utiliserons une carte de développement Nucleo-F429ZI (disponible chez Mouser et d'autres distributeurs). Pour suivre ce tutoriel, téléchargez le manuel de référence du MCU STM32F429 [1] puis le manuel d'utilisation de la carte de développement [2].

Pour démarrer, les outils suivants sont nécessaires :

ARM GCC, *https://launchpad.net/gcc-arm-embedded* - pour la compilation et l'édition des liens

GNU make, *https://gnu.org/software/make* - pour la construction automatisée

ST link, *https://github.com/stlink-org/stlink* - pour le flashage

Nous présentons ici les instructions d'installation pour Linux (Ubuntu).

Lancez un terminal, puis exécutez :

```
$ sudo apt -y update
$ sudo apt -y install gcc-arm-none-eabi make
stlink-tools
```

Pour configurer les outils sur un Mac ou un PC Windows, voir [3].

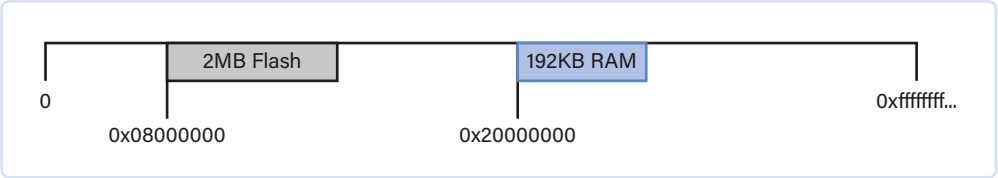


Figure 2. Emplacements des zones de la mémoire flash et de la mémoire vive du STM32F429.

Mémoire et registres

L'espace d'adressage d'un microcontrôleur à 32 bits, par exemple le STM32F429 de STMicroelectronics, est divisé en zones. Par exemple, une zone mémoire (à une adresse spécifique) est attribuée à la mémoire flash interne du microcontrôleur. Les instructions du micrologiciel sont lues et exécutées en lisant dans cette zone de mémoire. Une autre zone est la RAM, qui est également attribuée à une adresse spécifique. Nous pouvons lire et écrire n'importe quelle valeur dans la région RAM.

La section 2.3.1 du manuel de référence du STM32F429 [1] explique que la zone RAM commence à l'adresse 0x20000000 et a une taille de 192 KB. La section 2.4 nous indique que la mémoire flash commence à l'adresse 0x08000000. Notre microcontrôleur dispose de 2 Mo de mémoire flash, donc les zones flash et RAM sont réparties comme montré dans la **figure 2**.

Le manuel nous indique également qu'il existe de nombreuses autres zones mémoires. Leurs plages d'adresses sont indiquées dans la section 2.3 « Memory map ». Par exemple, il existe une zone « GPIOA » qui commence à 0x40020000 et qui a une taille de 1 Ko. Ces zones mémoires correspondent à différents *périphériques* dans le microcontrôleur – un circuit en silicium qui permet à certaines broches de se comporter d'une manière particulière. Une zone mémoire périphérique est un ensemble de registres de 32 bits. Chaque registre représente une plage de mémoire de 4 octets située à une certaine adresse, qui correspond à une certaine fonction du périphérique en question. En écrivant des valeurs dans un registre – en d'autres termes, en écrivant une valeur de 32 bits à une certaine adresse mémoire – nous pouvons contrôler le comportement d'un périphérique donné. En consultant ces registres, nous pouvons relire leurs contenus ou leurs configurations.

Il existe de nombreux périphériques différents. Parmi les plus simples, on trouve les ports GPIO (*General Purpose Input Output*), qui permettent à l'utilisateur de configurer les broches du microcontrôleur en mode « sortie » et de leur appliquer un niveau haut ou bas. Il est également possible de les configurer en mode « entrée » et de lire leurs tensions. Il existe un périphérique UART qui peut transmettre et recevoir des données série sur deux broches grâce à un protocole série. Il existe de nombreux autres périphériques. Un périphérique existe souvent sous plusieurs « versions », par exemple *GPIOA*, *GPIOB*, etc., qui contrôlent différents ensembles de broches du microcontrôleur. De même, on trouve *UART1*, *UART2*, etc., qui permettent l'implémentation de plusieurs canaux UART. Sur le STM32F429, il y a plusieurs périphériques GPIO et UART.

Par exemple, le périphérique *GPIOA* commence à l'adresse 0x40020000. Les registres GPIO sont décrits dans la section 8.4 [1]. Le manuel de référence indique que le registre `GPIOA_MODER` a un *offset* de 0, ce qui signifie que son adresse est `0x40020000 + 0`. Le format du registre est illustré dans la **figure 3**.

8.4.1 GPIO port mode register (GPIOx_MODER) (x = A..I/J/K)

Address offset: 0x00

Reset values:

- 0xA800 0000 for port A
- 0x0000 0280 for port B
- 0x0000 0000 for other ports

pin 15 … pin 8

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
MODER15[1:0]		MODER14[1:0]		MODER13[1:0]		MODER12[1:0]		MODER11[1:0]		MODER10[1:0]		MODER9[1:0]		MODER8[1:0]	
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
MODER7[1:0]		MODER6[1:0]		MODER5[1:0]		MODER4[1:0]		MODER3[1:0]		MODER2[1:0]		MODER1[1:0]		MODER0[1:0]	
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

pin 7 … pin 3 … pin 2 … pin 1 … pin 0 (A0, B0, …)

Bits 2y:2y+1 **MODERy[1:0]:** Port x configuration bits (y = 0..15)
These bits are written by software to configure the I/O direction mode.
00: Input (reset state)
01: General purpose output mode
10: Alternate function mode
11: Analog mode

Figure 3. La description des registres GPIO est disponible dans le manuel de référence. Un registre MODER contrôle 16 broches physiques. (Source : [1])

Le manuel indique que le registre *MODER* de 32 bits est une série de valeurs de 2 bits, 16 au total. Par conséquent, un registre *MODER* contrôle 16 broches matérielles, les bits 0…1 contrôlent la broche 0, les bits 2…3 contrôlent la broche 1, et ainsi de suite. La valeur de 2 bits permet de coder le mode de fonctionnement de la broche : 0 signifie entrée, 1 signifie sortie, 2 signifie « fonction alternative » (un comportement spécifique décrit ailleurs), et 3 signifie analogique. Le nom du périphérique étant *GPIOA*, les broches sont nommées « A0 », « A1 », etc. Pour le périphérique *GPIOB*, le nom des broches serait « B0 », « B1 »…
Si nous écrivons la valeur 0 dans le registre *MODER* de 32 bits, nous mettrons les 16 broches, de A0 à A15, en mode entrée :

```
* (volatile uint32_t *) (0x40020000 + 0) = 0;
  // Set A0...A15 to input mode
```

Nous aborderons la signification du qualificateur `volatile` ultérieurement. En définissant des bits spécifiques, nous pouvons configurer des broches spécifiques dans le mode souhaité. Par exemple, cet extrait configure la broche A3 en sortie :

```
* (volatile uint32_t *) (0x40020000 + 0) &= ~(3 << 6);
 // Clear bit range 6...7
* (volatile uint32_t *) (0x40020000 + 0) |= 1 << 6;
 // Set bit range 6...7 to 1
```

Expliquons ces opérations. Notre objectif est de mettre les bits 6…7, qui sont responsables de la broche 3 du périphérique *GPIOA*, à une valeur spécifique (1, dans notre cas). Cela se fait en deux étapes. Tout d'abord, nous devons effacer le contenu actuel des bits 6…7, car ils peuvent déjà contenir une certaine valeur. Ensuite, nous devons définir les bits appropriés pour obtenir la valeur que nous voulons. Nous devons donc commencer par mettre à 0 la plage de bits 6…7 (deux bits en position 6). Comment mettre à 0 un certain nombre de bits ? Les quatre étapes sont décrites dans le **tableau 1**.
Notez que la dernière opération fait passer N bits à la position X à 0 (combinés par ET logique avec 0), mais conserve les valeurs de tous les autres bits (combinés par ET avec 1). Il est important de conserver la valeur existante, car nous ne voulons pas modifier les paramètres dans d'autres plages de bits. En général, si nous voulons effacer N bits à la position X, nous pouvons écrire ce qui suit :

```
REGISTER &= ~((2^N - 1) << X);
```

Enfin, nous voulons attribuer une valeur spécifique à une plage de bits donnée. Nous décalons cette valeur de X positions vers la gauche, et nous la combinons par OU logique avec la valeur actuelle du registre (afin de conserver les valeurs des autres bits) :

```
REGISTER |= VALUE << X;
```

Tableau 1. Mise à zéro de certains bits.

Action	Expression	Bits (12 premiers bits sur 32)
Choisir un nombre avec N ensembles de bits contigus : 2^N-1, ici N = 2	`3`	`000000000011`
Décaler ce nombre de X positions vers la gauche	`(3<<6)`	`000011000000`
Inverser le nombre : transformer les zéros en uns et les uns en zéros	`~(3<<6)`	`111100111111`
ET logique avec la valeur existante	`VAL &= ~(3<<6)`	`xxxx00xxxxxx`

Programmation de périphériques aisée

Dans le paragraphe précédent, nous avons appris qu'il est possible d'écrire et lire un registre périphérique en accédant directement à certaines adresses de la mémoire. Examinons l'extrait qui permet de configurer la broche A3 en sortie :

```
* (volatile uint32_t *) (0x40020000 + 0) &= ~(3 << 6);
 // Clear bit range 6...7
* (volatile uint32_t *) (0x40020000 + 0) |= 1 << 6;
 // Set bit range 6...7 to 1
```

Sans commentaires détaillés, un tel code serait assez difficile à comprendre. Nous pouvons réécrire ce code et le rendre beaucoup plus lisible. L'idée est de représenter le périphérique sous la forme d'une structure contenant des champs de 32 bits. Les registres disponibles pour le périphérique GPIO sont décrits dans la section 8.4 du manuel de référence. Il s'agit de *MODER, OTYPER, OSPEEDR, PUPDR, IDR, ODR, BSRR, LCKR, AFR*. Leurs décalages (offsets) sont 0, 4, 8, etc. Cela signifie que nous pouvons les représenter sous la forme d'une structure avec des champs de 32 bits, et créer un `#define` pour *GPIOA* :

```
struct gpio {
 volatile uint32_t MODER, OTYPER, OSPEEDR,
   PUPDR, IDR, ODR, BSRR, LCKR, AFR[2];
};
#define GPIOA ((struct gpio *) 0x40020000)
```

Ensuite, pour définir le mode de la broche GPIO, nous pouvons définir une fonction :

```
// Enum values are per reference manual: 0, 1, 2, 3
enum {GPIO_MODE_INPUT, GPIO_MODE_OUTPUT,
   GPIO_MODE_AF, GPIO_MODE_ANALOG};

static inline void gpio_set_mode
 (struct gpio *gpio, uint8_t pin, uint8_t mode) {
 gpio->MODER &= ~(3U << (pin * 2));
 // Clear existing setting
 gpio->MODER |= (mode & 3) << (pin * 2);
 // Set new mode
}
```

Nous pouvons réécrire le code pour A3 comme suit :

```
gpio_set_mode(GPIOA, 3 /* pin */, GPIO_MODE_OUTPUT);
 // Set A3 to output
```

Le microcontrôleur possède plusieurs périphériques GPIO (aussi appelés *banques*) : A, B, C, … K. D'après la section 2.3, ils sont espacés de 1 Ko : *GPIOA* est à l'adresse 0x40020000, *GPIOB* à 0x40020400, et ainsi de suite :

```
#define GPIO(bank) ((struct gpio *)
 (0x40020000 + 0x400 * (bank)))
```

Nous pouvons définir une numérotation qui inclut la banque et le numéro de la broche. Pour cela, nous utilisons une valeur `uint16_t` de 2 octets, où l'octet de poids fort désigne la banque GPIO, et l'octet de poids faible désigne le numéro de la broche :

```
#define PIN(bank, num) ((((bank) - 'A') << 8) | (num))
#define PINNO(pin) (pin & 255)
#define PINBANK(pin) (pin >> 8)
```

Ainsi, nous pouvons spécifier des broches pour n'importe quelle banque GPIO :

```
uint16_t pin1 = PIN('A', 3); // A3 - GPIOA pin 3
uint16_t pin2 = PIN('G', 11); // G11 - GPIOG pin 11
```

Réécrivons la fonction `gpio_set_mode()` pour qu'elle reçoive la configuration de la broche :

```
static inline void gpio_set_mode(uint16_t pin, uint8_t
mode) {
 struct gpio *gpio = GPIO(PINBANK(pin));
   // GPIO bank
 uint8_t n = PINNO(pin); // Pin number
 gpio->MODER &= ~(3U << (n * 2));
   // Clear existing setting
 gpio->MODER |= (mode & 3) << (n * 2);
   // Set new mode
}
```

Le code pour A3 est simple :

```
uint16_t pin = PIN('A', 3); // Pin A3
gpio_set_mode(pin, GPIO_MODE_OUTPUT); // Set to output
```

Notez que nous avons créé une API initiale utile pour le périphérique GPIO. D'autres périphériques, tels que UART (communication série) et autres, peuvent être configurés de la même manière. Il s'agit d'une bonne pratique de programmation qui rend le code clair et lisible par l'utilisateur.

Démarrage du microcontrôleur et table vectorielle

Lorsqu'un microcontrôleur ARM démarre, il lit une « table vectorielle » située au début de la mémoire flash. Une table vectorielle est un élément commun à tous les microcontrôleurs ARM : Il s'agit d'un tableau d'adresses 32 bits de gestionnaires d'interruptions. Les 16 premières entrées sont réservées par ARM et sont communes à tous les microcontrôleurs ARM. Le reste des gestionnaires d'interruptions est spécifique au microcontrôleur en question – il s'agit de gestionnaires d'interruptions pour les périphériques. Les microcontrôleurs plus simples avec peu de périphériques ont peu de gestionnaires d'interruption, et ceux qui sont plus complexes en ont plusieurs.

La table vectorielle du STM32F429 est documentée dans le tableau 62 du manuel de référence [1]. On y apprend qu'il existe 91 gestionnaires de périphériques, en plus des 16 standards.

Chaque entrée de la table vectorielle est une adresse d'une fonction que le microcontrôleur exécute lorsqu'une interruption matérielle (IRQ) est déclenchée. Les deux premières entrées, qui jouent un rôle clé dans le processus de démarrage du microcontrôleur, font exception. Ces deux premières valeurs sont un pointeur de pile initial et l'adresse de la fonction de démarrage (un point d'entrée du micrologiciel à exécuter).

Dans le code, nous devons donc veiller à ce que la deuxième valeur de 32 bits dans la mémoire flash contienne l'adresse de notre fonction de démarrage. Au démarrage, le microcontrôleur lira cette adresse dans la mémoire flash et passera à la fonction de démarrage.

Micrologiciel minimal

Créons un fichier, *main.c*, et spécifions notre fonction de démarrage, qui ne fait rien au départ (commence par une boucle infinie), et spécifions également une table vectorielle qui contient 16 entrées standard et 91 entrées STM32. Dans l'éditeur de votre choix, créez et ouvrez le fichier *main.c* et entrez ce qui suit :

```
// Startup code
__attribute__((naked, noreturn)) void _reset(void) {
  for (;;) (void) 0; // Infinite loop
}

extern void _estack(void); // Defined in link.ld

// 16 standard and 91 STM32-specific handlers
__attribute__((section(".vectors")))
      void (*tab[16 + 91])(void) = {_estack, _reset};
```

Pour la fonction `_reset()`, nous avons utilisé les attributs `naked` et `noreturn` spécifiques au compilateur GCC – ils signifient que le prologue et l'épilogue de la fonction standard ne doivent pas être créés par le compilateur et que la fonction ne retourne rien.

L'expression `void (*tab[16 + 91])(void)` signifie : définir un tableau de 16 + 91 pointeurs vers des fonctions qui ne renvoient rien `(void)`, et reçoivent deux arguments. Chacune de ces fonctions est un *IRQ handler* (*Interrupt ReQuest handler*). Un tableau de ces gestionnaires est appelé une table vectorielle.

La table vectorielle `tab` est placée dans une section séparée appelée `.vectors` – nous en aurons besoin plus tard pour indiquer à l'éditeur de liens de placer cette section au début du micrologiciel généré, consécutivement, au début de la mémoire flash. Les deux premières entrées sont la valeur du registre du pointeur de pile et le point

d'entrée du micrologiciel. Nous laissons le reste du tableau vectoriel rempli de zéros.

Compilation

Compilons notre code. Démarrez un terminal (ou l'invite de commande sous Windows) et exécutez :

```
$ arm-none-eabi-gcc -mcpu=cortex-m4 main.c -c
```

Cela fonctionne ! La compilation a produit un fichier, *main.o*, qui contient notre micrologiciel minimal qui ne fait rien. Le fichier *main.o* est au format binaire *ELF*, qui contient plusieurs sections (voir **listage 1**).
Notez que les adresses VMA/LMA des sections sont fixées à 0, ce qui signifie que le fichier *main.o* est incomplet, car il ne contient pas d'informations sur l'emplacement de ces sections dans l'espace d'adressage. Nous devons utiliser un éditeur de liens pour produire le fichier du micrologiciel complet, *firmware.elf*, à partir de *main.o*.
La section `.text` contient le code du micrologiciel, qui dans notre cas consiste simplement en une fonction `_reset()`, de deux octets – une instruction de saut à sa propre adresse. Il y a deux sections `.data` et `.bss` vides [8] (pour les variables non initialisées, mais déclarées, cette section est généralement remplie avec des 0).
Notre micrologiciel sera copié dans la mémoire flash à la position 0x8000000, mais notre section de données doit être dans la RAM – par conséquent, notre fonction `_reset()` doit copier le contenu de la section `.data` vers la RAM. Elle doit également écrire des zéros dans toute la section `.bss`. Dans notre cas, les sections `.data` et `.bss` sont vides, mais modifions tout de même la fonction `_reset()` pour les gérer correctement.
Pour ce faire, nous devons savoir où commence la pile et où commencent les sections data et bss. Nous pouvons le spécifier dans le script *linker*, qui est un fichier contenant les instructions de l'éditeur de liens concernant l'emplacement des différentes sections dans l'espace d'adressage et sur les symboles à créer.

Script *Linker*

Créez un fichier appelé `link.ld`, et copiez-collez le contenu de [4]. L'explication est donnée ci-dessous :

```
ENTRY(_reset);
```

Cette ligne indique à l'éditeur de liens la valeur de l'attribut « entry point » dans l'en-tête ELF généré – il s'agit donc d'une copie de ce que contient une table vectorielle. Il s'agit d'une aide pour débogueur (tel qu'Ozone, décrite dans la deuxième partie de ce guide) qui nous permet de placer un point d'arrêt au début du micrologiciel. Un débogueur ne connaît pas la table vectorielle, il se fie donc à l'en-tête ELF.

```
MEMORY {
flash(rx) : ORIGIN = 0x08000000, LENGTH = 2048k
sram(rwx) : ORIGIN = 0x20000000, LENGTH = 192k /*
remaining 64k in a separate address space */
}
```

Cela indique à l'éditeur de liens que nous avons deux zones mémoires dans l'espace d'adressage, ainsi que leurs adresses et leurs tailles.

```
_estack = ORIGIN(sram) + LENGTH(sram); /* stack points
to end of SRAM */
```

Cela indique à l'éditeur de liens de créer un symbole, `_estack`, contenant une valeur à la fin de la zone RAM. Ce sera la valeur initiale de notre pile !

```
.vectors : { KEEP(*(.vectors)) } > flash
.text : { *(.text*) } > flash
.rodata : { *(.rodata*) } > flash
```

Listage 1. Compilation du fichier main.o

```
$ arm-none-eabi-objdump -h main.o
 ...
 Idx Name          Size      VMA       LMA       File off  Algn
   0 .text         00000002  00000000  00000000  00000034  2**1
                   CONTENTS, ALLOC, LOAD, READONLY, CODE
   1 .data         00000000  00000000  00000000  00000036  2**0
                   CONTENTS, ALLOC, LOAD, DATA
   2 .bss          00000000  00000000  00000000  00000036  2**0
                   ALLOC
   3 .vectors      000001ac  00000000  00000000  00000038  2**2
                   CONTENTS, ALLOC, LOAD, RELOC, DATA
 ...
```

Listage 2. Code de démarrage.

```
int main(void) {
   return 0; // Do nothing so far
}

// Startup code
__attribute__((naked, noreturn)) void _reset(void) {
   // memset .bss to zero, and copy .data section to RAM region
   extern long _sbss, _ebss, _sdata, _edata, _sidata;
   for (long *src = &_sbss; src < &_ebss; src++) *src = 0;
   for (long *src = &_sdata, *dst = &_sidata; src < &_edata;) *src++ = *dst++;

   main();             // Call main()
   for (;;) (void) 0;  // Infinite loop in the case if main() returns
 }
```

Ces lignes indiquent à l'éditeur de liens de placer la table vectorielle dans la mémoire flash en premier, suivie de la section `.text` (code du micrologiciel), puis des données en lecture seule `.rodata`. Vient ensuite la section `.data` :

```
.data : {
 _sdata = .; /* .data section start */
 *(.first_data)
 *(.data SORT(.data.*))
 _edata = .; /* .data section end */
} > sram AT > flash
_sidata = LOADADDR(.data);
```

Notez que nous demandons à l'éditeur de liens de créer les symboles `_sdata` et `_edata`. Nous les utiliserons pour copier la section de données en RAM dans la fonction `_reset()`.
Idem pour la section `.bss` :

```
.bss : {
 _sbss = .; /* .bss section start */
 *(.bss SORT(.bss.*) COMMON)
 _ebss = .; /* .bss section end */
} > sram
```

Code de démarrage

Nous pouvons maintenant mettre à jour la fonction `_reset()`. Nous copions la section `.data` dans la RAM, et initialisons la section `.bss` avec des zéros. Ensuite, nous appelons la fonction `main()` - et entrons dans une boucle infinie lorsque `main()` s'arrête (voir **listage 2**).
Le diagramme de la **figure 4** illustre comment `_reset()` initialise `.data` et `.bss`.
Le fichier *firmware.bin* n'est qu'une concaténation des trois sections : `.vectors` (table vectorielles IRQ), `.text` (code) et `.data` (données). Ces sections ont été créées conformément au script de l'éditeur de liens : `.vectors` se situe au début de la mémoire flash suivi par `.text` et `.data` se situe bien au-dessus. Les adresses dans `.text` se trouvent dans la mémoire flash, et les adresses dans `.data` se trouvent dans la mémoire vive. Si une fonction a une adresse, par exemple `0x8000100`, elle se trouve exactement à cette adresse dans la mémoire flash. Mais si le code accède à une variable de la section `.data` par son adresse, par exemple `0x20000200`, il n'y a rien à cette adresse, car, au démarrage, la section `.data` du fichier *firmware.bin* réside dans la mémoire flash ! C'est pourquoi le code de démarrage doit déplacer la section `.data` de la mémoire flash vers la RAM.
Nous sommes maintenant prêts à produire le fichier du micrologiciel complet, *firmware.elf* :

```
$ arm-none-eabi-gcc -T link.ld -nostdlib main.o -o
firmware.elf
```

Examinons les sections du fichier *firmware.elf* – voir **listage 3**.
La section `.vectors` se trouve au tout début de la mémoire flash, à l'adresse 0x8000000, suivie par la section `.text` à l'adresse 0x80001ac. Notre code ne crée pas de variables, il n'y a donc pas de section de données.

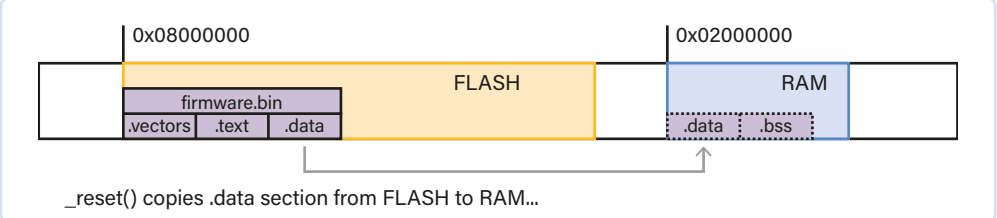


Figure 4. Le diagramme montre comment `_reset()` initialise `.data` et `.bss`.

Listage 3. Extrait du fichier firmware.elf

```
$ arm-none-eabi-objdump -h firmware.elf
 ...
 Idx Name          Size      VMA       LMA       File off  Algn
   0 .vectors      000001ac  08000000  08000000  00010000  2**2
                   CONTENTS, ALLOC, LOAD, DATA
   1 .text         00000058  080001ac  080001ac  000101ac  2**2
                   CONTENTS, ALLOC, LOAD, READONLY, CODE
 ...
```

Flasher le micrologiciel

Nous pouvons maintenant flasher le micrologiciel ! Tout d'abord, extrayez les sections du fichier *firmware.elf* en un seul bloc binaire contigu :

```
$ arm-none-eabi-objcopy -O binary firmware.elf firmware.bin
```

Utilisons ensuite l'utilitaire *st-link* pour flasher le *firmware.bin*. Branchez votre carte sur le port USB et exécutez :

```
$ st-flash --reset write firmware.bin 0x8000000
```

C'est fait ! Nous avons flashé un micrologiciel qui ne fait rien.

Makefile : automatisation de la construction

Au lieu de taper ces commandes de compilation, d'édition de liens et de flashage, nous pouvons utiliser l'outil de ligne de commande `make` pour automatiser l'ensemble du processus. L'utilitaire `make` utilise un fichier de configuration appelé *Makefile*, dans lequel il lit les instructions d'exécution. Cette automatisation est très utile, car elle permet également de documenter le processus de création du micrologiciel, les drapeaux de compilation utilisés, etc.
Il existe un excellent tutoriel sur *Makefile* [5]. Je le recommande à ceux qui veulent s'initier à `make`. Ci-dessous, j'énumère les concepts les plus essentiels nécessaires à la compréhension de notre *Makefile bare-metal* simple. Ceux qui sont déjà familiers avec `make` peuvent sauter cette section.
Le format du fichier *Makefile* est simple :

```
action1:
  command ... # Comments can go after hash symbol
  command .... # IMPORTANT: command must be preceded with
the TAB character
action2:
  command ... # Don't forget about TAB. Spaces won't work!
```

Nous pouvons utiliser `make` avec le nom de l'action (également appelé *target*) pour exécuter l'action correspondante :

```
$ make action1
```

Il est possible de définir des variables et de les utiliser dans les commandes. Les actions peuvent également correspondre aux noms des fichiers qui doivent être créés :

```
firmware.elf:
  COMPILATION COMMAND .....
```

De plus, toute action peut avoir une liste de dépendances. Par exemple, *firmware.elf* dépend de notre fichier source, *main.c*. Chaque fois que le fichier *main.c* est modifié, la commande `make build` recrée *firmware.elf* :

```
build: firmware.elf
firmware.elf: main.c
  COMPILATION COMMAND
```

Nous pouvons maintenant créer un Makefile pour notre micrologiciel. Nous définissons une action `build`/*target* :

```
CFLAGS ?= -W -Wall -Wextra -Werror -Wundef -Wshadow
-Wdouble-promotion \ -Wformat-truncation -fno-common
-Wconversion \ -g3 -Os -ffunction-sections -fdata-
sections -I. \ -mcpu=cortex-m4 -mthumb -mfloat-abi=hard
-mfpu=fpv4-sp-d16 $(EXTRA_CFLAGS)
LDFLAGS ?= -Tlink.ld -nostartfiles -nostdlib --specs nano.
specs -lc -lgcc -Wl,--gc-sections -Wl,-Map=$@.map
SOURCES = main.c
build: firmware.elf
firmware.elf: $(SOURCES)
  arm-none-eabi-gcc $(SOURCES) $(CFLAGS)
  $(LDFLAGS) -o $@
```

Nous y définissons les drapeaux de compilation. `?=` représente une valeur par défaut ; nous pouvons les remplacer à partir de la ligne de commande :

```
$ make build CFLAGS="-O2 ...."
```

Nous définissons les variables `CFLAGS`, `LDFLAGS`, et `SOURCES`. Puis nous indiquons à `make` que si une instruction *build* est reçue, un fichier *firmware.elf* doit être créé. Ce dernier dépend du fichier *main.c*, et pour le créer, `make` doit lancer le compilateur *arm-none-eabi-gcc* avec les drapeaux donnés. La variable spéciale `$@` se développe en un nom de cible – dans notre cas, *firmware.elf*.
Appelons `make` :

```
$ make build arm-none-eabi-gcc main.c -W -Wall -Wextra
-Werror -Wundef -Wshadow -Wdouble-promotion -Wformat-
truncation -fno-common -Wconversion -g3 -Os -ffunction-
sections -fdata-sections -I. -mcpu=cortex-m4 -mthumb
-mfloat-abi=hard -mfpu=fpv4-sp-d16 -Tlink.ld -nostartfiles
-nostdlib --specs nano.specs -lc -lgcc -Wl,--gc-sections
-Wl,-Map=firmware.elf.map -o firmware.elf
```

Listage 4. Extrait du fichier main.c du projet Blinky LED.

```
#include <inttypes.h>
#include <stdbool.h>
#define BIT(x) (1UL << (x))
#define PIN(bank, num) ((((bank) - 'A') << 8) | (num))
#define PINNO(pin) (pin & 255)
#define PINBANK(pin) (pin >> 8)

struct gpio {
   volatile uint32_t MODER, OTYPER, OSPEEDR, PUPDR, IDR, ODR, BSRR, LCKR, AFR[2];
};
#define GPIO(bank) ((struct gpio *) (0x40020000 + 0x400 * (bank)))

// Enum values are per datasheet: 0, 1, 2, 3
enum { GPIO_MODE_INPUT, GPIO_MODE_OUTPUT, GPIO_MODE_AF, GPIO_MODE_ANALOG };

static inline void gpio_set_mode(uint16_t pin, uint8_t mode) {
   struct gpio *gpio = GPIO(PINBANK(pin));  // GPIO bank
   int n = PINNO(pin);                      // Pin number
   gpio->MODER &= ~(3U << (n * 2));         // Clear existing setting
   gpio->MODER |= (mode & 3) << (n * 2);    // Set new mode
}
```

Si nous l'exécutons à nouveau :

```
$ make build
make: Nothing to be done for 'build'.
```

L'utilitaire make examine les temps de modification de la dépendance *main.c* et *firmware.elf* – et ne fait rien si *firmware.elf* est à jour. Mais si nous modifions *main.c*, le prochain `make build` recompilera :

```
$ touch main.c # Simulate changes in main.c
$ make build
```

Il ne reste plus que la cible `flash` :

```
firmware.bin: firmware.elf
  $(DOCKER) $(CROSS)-objcopy -O binary $< $@ flash:
firmware.bin
  st-flash --reset write $(TARGET).bin 0x8000000
```

Voilà, c'est fait ! Maintenant, la dernière commande `make flash` crée un fichier *firmware.bin*, et le flashe sur la carte. Elle recompilera le micrologiciel si le fichier *main.c* change, parce que *firmware.bin* dépend de *firmware.elf*, et celui-ci, à son tour, dépend de *main.c*. Donc, le cycle de développement consisterait en ces deux actions en boucle :

```
# Develop code in main.c
$ make flash
```

Il est conseillé d'ajouter une cible `clean` pour supprimer les artefacts de `build` :

```
clean:
  rm -rf firmware.*
```

Le code source complet du projet se trouve dans le dossier minimal de *Step 0* [6].

LED clignotante

Maintenant que nous avons configuré l'infrastructure build / flash, il est temps que notre firmware apprenne à effectuer une tâche utile, par exemple faire clignoter une LED. La carte Nucleo-F429ZI possède trois LED intégrées. Dans la section 6.5 du manuel d'utilisation de la carte Nucleo [2], nous pouvons savoir à quelles broches sont connectées les LED intégrées :

- PB0 : LED verte
- PB7 : LED bleue
- PB14 : LED rouge

Modifions le fichier *main.c* et ajoutons nos définitions pour PIN, `gpio_set_mode()`. Dans la fonction `main()`, nous configurons la LED bleue en sortie, et nous lançons une boucle infinie. Tout d'abord, copions les définitions des broches et des GPIO dont nous avons parlé précédemment. Notez que nous ajoutons également une macro de commodité, `BIT(x)` (voir **listage 4**).
Lorsque certains microcontrôleurs sont mis sous tension, tous leurs périphériques sont alimentés et activés automatiquement. Les périphériques des microcontrôleurs STM32 restent désactivés par défaut afin d'économiser de l'énergie. L'activation d'un périphérique GPIO doit se faire via l'unité RCC (*Reset and Clock Control*). Dans la section 7.3.10 du manuel de référence, nous constatons que le registre *AHB1ENR* (*AHB1 peripheral clock enable register*) est responsable de l'activation ou de la désactivation des banques GPIO. Tout d'abord, nous ajoutons une définition pour toute l'unité RCC :

```
struct rcc {
  volatile uint32_t CR, PLLCFGR, CFGR, CIR, AHB1RSTR,
AHB2RSTR, AHB3RSTR, RESERVED0, APB1RSTR, APB2RSTR,
RESERVED1[2], AHB1ENR, AHB2ENR, AHB3ENR, RESERVED2,
APB1ENR, APB2ENR, RESERVED3[2], AHB1LPENR, AHB2LPENR,
AHB3LPENR, RESERVED4, APB1LPENR, APB2LPENR, RESERVED5[2],
BDCR, CSR, RESERVED6[2], SSCGR, PLLI2SCFGR;
};
#define RCC ((struct rcc *) 0x40023800)
```

Selon la documentation du registre *AHB1ENR*, les bits 0 à 8 inclus configurent l'horloge pour les banques GPIO GPIOA-GPIOE :

```
int main(void) {
  uint16_t led = PIN('B', 7); // Blue LED
  RCC->AHB1ENR |= BIT(PINBANK(led));
  // Enable GPIO clock for LED
  gpio_set_mode(led, GPIO_MODE_OUTPUT);
  // Set blue LED to output mode
  for (;;) asm volatile("nop"); // Infinite loop
  return 0;
}
```

Il ne reste plus qu'à découvrir comment activer ou désactiver une broche GPIO, puis à modifier la boucle principale pour activer une broche de LED, ajouter un délai, désactiver puis ajouter un délai. Selon la section 8.4.7 du manuel de référence, le registre *BSRR* est responsable de la mise de la tension au nivau haut ou bas. Les 16 bits de poids faible sont utilisés pour mettre le registre *ODR* (i.e. mettre la broche à l'état haut), et les 16 bits de poids fort sont utilisés pour réinitialiser le registre *ODR* (i.e. mettre la broche à l'état bas). Définissons une fonction API pour cet effet :

```
static inline void gpio_write(uint16_t pin, bool val) {
  struct gpio *gpio = GPIO(PINBANK(pin));
  gpio->BSRR |= (1U << PINNO(pin)) << (val ? 0 : 16);
}
```

Ensuite, nous devons implémenter une fonction de délai. Nous n'avons pas besoin d'un délai précis pour l'instant, alors définissons une fonction, `spin()`, qui exécute simplement une instruction NOP un certain nombre de fois :

```
static inline void spin(volatile uint32_t count) {
  while (count--) asm("nop");
}
```

Enfin, nous pouvons modifier notre boucle *main* pour faire clignoter une LED :

```
for (;;) {
  gpio_write(pin, true);
  spin(999999);
  gpio_write(pin, false);
  spin(999999);
}
```

Le code source complet du projet est disponible dans le dossier blinky de l'étape 1 [7]. Lancez `make flash` et admirez la LED bleue qui clignote !
Dans la deuxième partie de cet article, nous aborderons la sortie UART, le débogage, l'implémentation d'un serveur web, les tests automatiques, et bien plus encore. Restez à l'écoute !

220665-04

À propos de l'auteur

Sergey Lyubka est ingénieur et entrepreneur. Il est titulaire d'un Master en physique de l'université d'État de Kiev, en Ukraine. Sergey est directeur et cofondateur de Cesanta, une entreprise technologique basée à Dublin, en Irlande (*Embedded Web Server for electronic devices : https://mongoose.ws*). Il est passionné par la programmation embarquée « bare-metal » de réseaux.

Des questions, des commentaires ?

Envoyez un courriel à l'auteur (sergey.lyubka@cesanta.com) ou contactez Elektor (redaction@elektor.fr).

LIENS

[1] Manuel de référence RM0090 pour STM32F429 : https://bit.ly/3neE7S7
[2] Manuel d'utilisation de la carte Nucleo-144 (UM1974) : https://bit.ly/3oIBXKZ
[3] Cet article sur GitHub : https://github.com/cpq/bare-metal-programming-guide
[4] Contenu du fichier link.ld : https://github.com/cpq/bare-metal-programming-guide/blob/main/step-0-minimal/link.ld
[5] Tutoriel pour Makefile : https://makefiletutorial.com/
[6] Programme de démonstration minimal (Step 0) : https://github.com/cpq/bare-metal-programming-guide/blob/main/step-0-minimal
[7] Programme de démonstration blinky (Step 1) : https://github.com/cpq/bare-metal-programming-guide/blob/main/step-1-blinky
[8] .bss [Wikipedia] : https://fr.wikipedia.org/wiki/Segment_BSS

multimètre
Siglent SDM3045X

Philippe Demerliac (France)

Il est toujours utile d'avoir un multimètre portatif classique. Un multimètre de table, cependant, offre beaucoup plus de fonctionnalités et un confort d'utilisation. Le multimètre de table SDM3045X de Siglent est l'un de ces appareils.

Je possède déjà de nombreux appareils Siglent, appareils que j'apprécie et sont d'excellente facture. Dans les nombreuses familles d'instruments qu'ils proposent, il me manquait le multimètre de table. Cette lacune vient d'être comblée et je peux maintenant partager avec vous mes impressions sur le SDM3045X.
Siglent propose 3 familles de multimètres : SDM3045X, SDM3055 et SDM3065X
Ces modèles offrent globalement les mêmes fonctionnalités mais diffèrent par leurs résolutions qui est de 4 ½ digits pour le 3045 (60 000 points), 5 ½ (240 000 points) pour le 3055 et 6 ½ (2 200 000 points) pour de 3065. Les modèles 55 et 65 offrent aussi une meilleure sensibilité basse, 200 mV / 200 µA contre 600 mV / 600 µA pour le 45.
De plus les modèles 3055 et 3065 offrent l'option SC qui permet d'avoir plusieurs entrées de mesures programmables.
Ces modèles partagent le même boitier et la même ergonomie. Les prix varient bien sûr avec la résolution.
Alors justement, parlons u peu de la résolution, elle est exprimée en « Digit », nombre de chiffres significatifs, ou en points. Le nombre de points indique le nombre de valeurs distinctes affichables pour une gamme donnée. Par exemple, le SDM3045X sur sa gamme la plus sensible de 600 mV pourra afficher des variations de 10 µV (0,6 / 60000), sur sa gamme 600 V, 10 mV, sa limite maximum étant de 1 000 V avec dans ce cas une résolution de 100 mV. (La gamme ne monte pas à 6 000 V, pour des raisons d'isolation, pas de résolution).
Sur les modèles offrant de meilleures résolutions, on pourra avoir plus de chiffres significatifs.
Attention de ne pas confondre résolution et justesse. Si on lit une mesure de 1,0000 V sur l'écran, est-on sûr que la tension est de 1 V à 100 µV prés ? Non, car même correctement étalonné les spécifications indiquent une erreur maximale de 0.06 % +- 8 digits. Donc une tension comprise entre 0,9932 V et 0.1.0068 V, ces limites étant des maximums, généralement, les appareils sont meilleurs.
Faut-il en déduire que les chiffres les plus bas sont inutiles ? Non, car ils permettent de comparer des mesures et de voir leurs évolutions dans un sens ou dans l'autre.
Pour revenir à notre SDM3045X, est-ce que 60 000 points suffisent ou faut-il éventuellement prendre des modèles au-dessus ? La réponse à cette question dépend bien évidemment de l'usage. Ce que je peux dire, c'est que pour un usage « Amateur » et même pro, cette résolution est plus que suffisante, voir même souvent superflue. J'ai utilisé et utilise encore des multimètres ayant de moins bonne résolution sans que cela ne pose le moindre souci dans 90 % des cas.
Vous pouvez facilement télécharger la fiche technique de ce modèle sur le site de Siglent [1] et voir pour chaque type de mesure et chaque gamme, la résolution et l'erreur max garantie par le constructeur.
En parlant de justesse, les multimètres sont livrés avec un certificat de calibration prouvant que l'appareil était dans les limites annoncées, voir souvent meilleur.

Cette revue en vidéo [en français]:
https://youtu.be/iTW90_JqmM4

Siglent indique à ce sujet :

> *« SIGLENT a déterminé que l'étalonnage en usine de cet instrument n'est pas affecté de manière significative par un stockage allant jusqu'à 180 jours avant la première utilisation. L'intervalle d'étalonnage doit commencer au moment où l'appareil est mis en service ou 180 jours après la "date d'étalonnage" figurant sur le certificat reçu avec l'appareil. »*

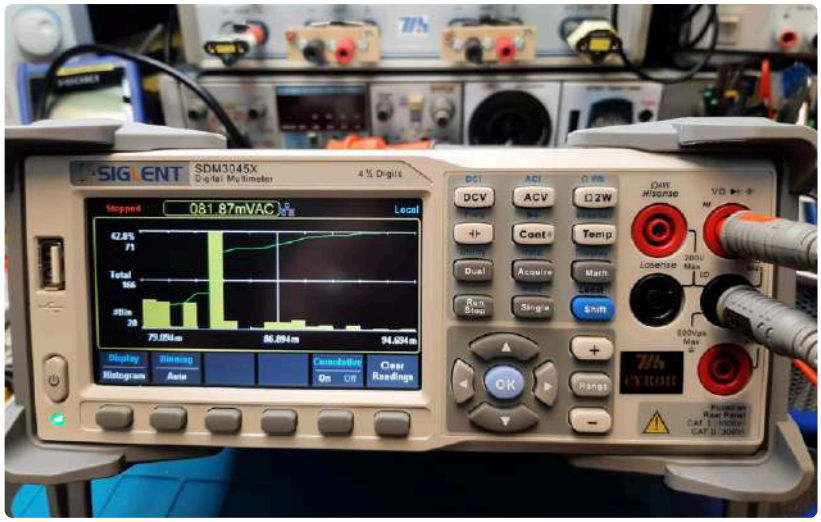
Faut-il d'ailleurs réétalonner ces multimètres périodiquement ?
Dans un cadre professionnel c'est conseillé, voire obligatoire si on veut respecter certaines normes, dans un cadre amateur, facultatif.
Par expérience, les matériels modernes dérivent assez peu avec le temps. Si on peut le faire, il est juste conseillé de vérifier périodiquement la justesse de l'appareil.
Il existe des centres de métrologie agréés qui peuvent effectuer ces étalonnages, mais c'est une opération relativement couteuse comparée au prix de l'appareil. Même si Siglent n'est pas très prolixe sur l'étalonnage, on trouve sur le net des « hacks » pour le faire soi-même, ce qui suppose quand même d'avoir les sources étalons adéquates.
En résumé, dans bien des cas, ce n'est pas vraiment un souci et votre multimètre vous servira fidèlement de longues années.
Le SDM3045X est un multimètre de table, et donc, bien qu'il soit transportable, n'est pas vraiment destiné à un usage mobile sur le terrain. Déjà, il n'est pas autonome et nécessite une alimentation secteur, ensuite il faut le poser sur une table ou au moins un support stable pour l'utiliser. Les multimètres de table offrent généralement de meilleures caractéristiques et surtout plus de fonctionnalités que les modèles autonomes comme la connectivité, la mesure 4 fils, un affichage plus riche, etc...
Il est possible de les poser devant soi de manière stable et l'usage est clairement plus confortable, sans compter que l'alimentation secteur dispense de la contrainte de changement des piles ou de recharges périodiques.
Le SDM3045X permet de mesurer des tensions et des courants en continus et alternatifs (RMS) jusqu'à 100 kHz (donc parfait pour la BF), des résistances (2 fils ou 4 fils pour les résistances très faibles), la continuité, les seuils des diodes, les condensateurs et la température avec différents capteurs (il intègre la compensation de soudure froide pour les thermocouples). Cela couvre la majorité des besoins courants. Il permet aussi de mémoriser les valeurs lues, de faire des statistiques dessus voir des alarmes en cas de dépassement.

Premières impressions

Comme pour les autres appareils Siglent, il est livré dans un emballage bien conçu qui le protège parfaitement contre les chocs. Je vous conseille si vous le pouvez de le conserver précieusement.

L'appareil est livré avec un cordon secteur, 2 cordons de mesure souples de très bonne qualité, un câble USB A/B pour connexion à un PC, un certificat de calibration et une notice d'utilisation basique en anglais.
Il est possible de télécharger des manuels détaillés complets (dont un en français) sur le site de Siglent.
L'appareil donne une impression de qualité et la finition est impeccable. Les touches en plastiques souples offrent un toucher agréable. Le bouton de mise en route n'est pas un « Vrai » interrupteur ce qui fait que l'appareil est toujours en veille. Personnellement, je préfère une vraie déconnexion du secteur, et coupe tout quand je n'utilise pas ces appareils via une prise munie d'un interrupteur global.
L'écran est très lisible. Lors de la mise en route, il faut compter un temps d'initialisation de quelques dizaines de secondes. De toutes manières, il est conseillé de ne pas allumer/éteindre ces appareils pour de courtes périodes, la justesse maximum étant garantie après un temps de chauffe (10 à 30 mn). Mais on peut bien sûr l'utiliser avant.
Contrairement semble-t-il aux modèles supérieurs munis de ventilateurs un peu bruyants, le SDM3045X en est dépourvu et donc silencieux, exceptés parfois quelques discrets cliquetis de relais.
La face arrière, propose deux prises BNC, une entrée pour déclencher les mesures via un signal externe, et une sortie qui indique que la mesure est effectuée. Ces prises sont surtout utiles pour des bancs de mesures automatiques et rarement utilisées en usage courant.
On y trouve aussi un connecteur LAN RJ45 et une prise USB B pour relier l'appareil à un PC.
Un emplacement est aussi prévu pour fixer un câble antivol.
Le fusible 10 A de l'ampèremètre est aussi accessible pour être changé facilement en cas de surintensité.

Figure 1. Face arrière.



▲

Figure 2. Mode histogramme.

Lors de la première mise en route, je vous conseille de vérifier que le firmware est à jour. Siglent fait souvent évoluer ses appareils et les mises à jour sont facilement téléchargeables sur leur site.
La mise à jour éventuelle se fait via une clé USB branchée sur la face avant.

Utilisation courante

L'utilisation courante est simple et intuitive. On passe facilement d'un mode de mesure à l'autre via les boutons de la face avant.
Il est possible de changer la vitesse de mesure qui joue sur la résolution. Dans bien des cas, le mode rapide sera suffisant avec un temps de réponses quasi instantané.
Le changement de gamme automatique réduit aussi grandement les manipulations nécessaires. On peut cependant à tout moment choisir la gamme manuellement facilement.
En tension/courant alternatif, il offre une bande passante de 100 kHz et une mesure RMS vraie.
Comme la plupart des multimètres, il faut cependant apporter manuellement une correction du facteur de crête pour les signaux dissymétriques, la doc donne toutes les précisions sur ce point.
En mode continuité ou test diode on peut ajuster le seuil du bip et son volume ce qui est un plus.
Dans toutes les mesures, on peut fixer un mode « relatif » afin de voir les variations autour d'une valeur mémorisée.
On peut aussi stopper manuellement l'acquisition et mémoriser la dernière mesure à tout moment.

Particularités intéressantes

Le SDM3045X se comporte donc comme un multimètre de base mais il offre des possibilités additionnelles qui peuvent parfois bien faciliter la vie, voyons ici les principales :

- Différents modes d'affichage, un bargraphe qui permet de bien visualiser l'ordre de grandeur de la mesure et sa variation, un mode histogramme qui montre graphiquement la distribution statistique des mesures effectuées, ce qui permet d'un coup d'œil de voir les valeurs les plus fréquentes et pour finir un mode courbe qui montre l'évolution dans le temps de la mesure sur des périodes aussi basse qu'une seconde, très pratique pour voir l'évolution des valeurs dans le temps. Dans tous ces modes, les échelles et le limites peuvent être calculées automatiquement ou forcées manuellement.
- Un mode statistique qui montre les valeurs moyennes, min, max, l'écart type etc. des mesures.
- L'affichage « Dual » qui permet dans certains cas d'afficher 2 paramètres en même temps, comme la tension et la fréquence.
- Un mode dB / dBm pouvant faire des mesures relatives en dB, ou des mesures de puissance en dBm pour une impédance de charge qu'il faut paramétrer. (ATTENTION : ce réglage ne change en rien l'impédance d'entrée du multimètre, il faut mettre la charge souhaitée de manière externe)
- Un mode limites qui permet visuellement de voir si la valeur mesurée est comprise entre 2 bornes paramétrables, ceci est très pratique pour des réglages de circuit répétitifs sans avoir à interpréter les valeurs.
- Le mode « probe hold », un de mes préférés, qui mémorise automatiquement les mesures stables consécutives. Quel confort pour trier des composants.
- On peut aussi paramétrer finement le mode d'acquisition, la vitesse d'échantillonnage, s'il est

Figure 3. L'affichage « Dual ».

Figure 4. Mode « limit ».

Figure 5. Mode « probe hold ».

auto, manuel ou déclenché par le signal « trigger » externe dont on peut définir la valeur et la polarité.

- On peut mémoriser sur clé USB les valeurs mesurées et/ou les réglages courants afin de les rappeler facilement.
- Le mode thermomètre accepte différents types de thermocouples courants (avec compensation de la soudure froide) et aussi des capteurs résistifs. (Note : il n'y a pas de sonde de température livrée avec l'appareil)
- Une aide intégrée permet de rappeler l'utilisation et les branchements à effectuer pour les mesures. (En Anglais)

Utilisation avec PC

L'appareil est pilotable par SPI via USB ou LAN Ethernet. Les drivers LabVIEW sont disponibles sur le site Siglent.
La documentation du protocole est disponible pour piloter l'appareil.
De plus, Siglent offre gratuitement l'application Windows EasyDMM qui permet facilement de piloter tous les multimètres Siglent, de visualiser les mesures et de les exporter en CSV.
Cette application, malgré son *look* un peu vieillot fonctionne très bien.

Conclusion

Je n'ai jamais été déçu par les appareils de mesures Siglent et ce multimètre m'a globalement fait une très bonne impression aussi. Il est pratique et couvre sans doute la plupart des besoins d'un labo de dépannage ou de R & D. Le prix est certes un peu plus élevé que des modèles Asiatiques d'entrée de gamme, mais la qualité du produit le justifie. Le rapport qualité prix est excellent et si vous voulez investir dans un multimètre de table de qualité pratique et performant, je ne peux que vous le conseiller. En résumé :
J'ai particulièrement apprécié :

- La facilité d'utilisation
- La qualité globale et l'écran hyper lisible
- La richesse des fonctionnalités offertes
- Le silence (Pas de ventilateur)

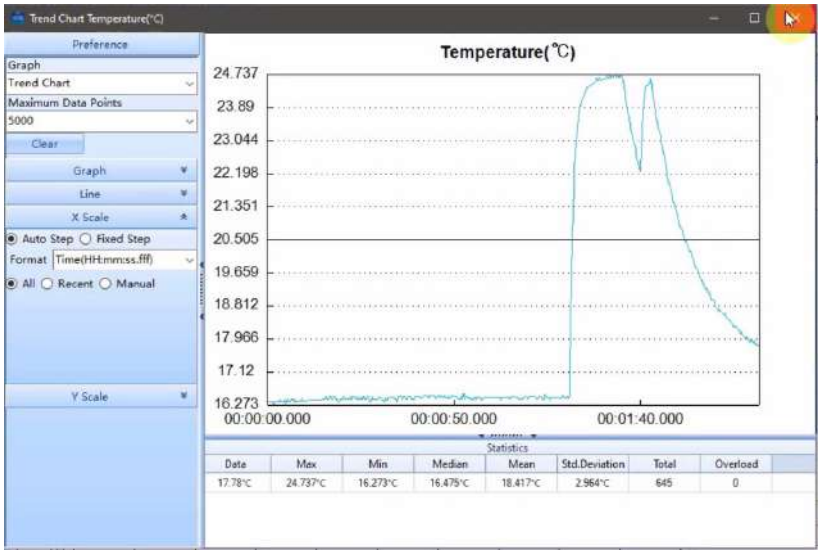


Figure 6. L'application Windows EasyDMM.

- Le rapport qualité-prix
- La documentation détaillée en français
- Les mises à jour facilement disponibles

J'ai regretté :

- L'interrupteur qui fait juste une mise en veille
- Le manuel de service dispo mais avec peu d'infos pratiques

230126-04

Des questions, des commentaires ?

Envoyez un courriel à l'auteur (info@cyrob.org) ou contactez Elektor (redaction@elektor.fr).

À propos de l'auteur

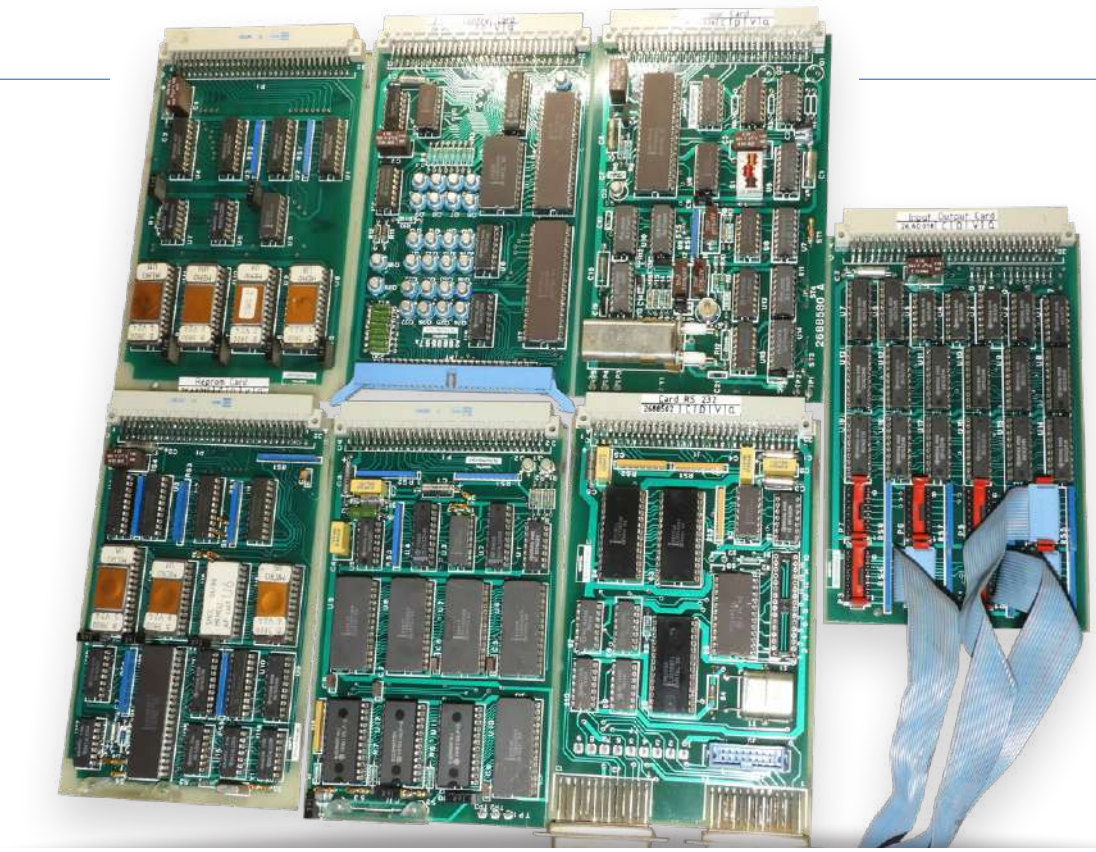
Philippe Demerliac, né en 1962, est un concepteur de circuits électroniques à vocation scientifique, passionné par la mécanique de précision, les appareils de mesure, l'usinage des métaux et la science en général depuis très longtemps. Bien que sa carrière professionnelle se soit orientée vers les logiciels, il s'intéresse au domaine de l'électronique. Dans le but contribuer à la communauté et lui rendre tout ce qu'elle lui apporte, Philippe a créé le site web cyrob.org et, et plus tard, une chaîne YouTube en français : *youtube.com/@Cyrob-org*

Produits

- **Multimètre numérique Siglent SDM3045X**
  https://elektor.fr/17892

LIENS

[1] Fiche technique du Siglent SDM3045X : https://www.siglenteu.com/wp-content/uploads/dlm_uploads/2022/12/SDM3045X_DataSheet_E03A.pdf



microprocesseurs pour systèmes embarqués

drôle de composants

David Ashton (Australie)

Les électroniciens et les amateurs de l'embarqué ont aujourd'hui l'embarras du choix en ce qui concerne la puissance de calcul, grâce à la vaste panoplie de microcontrôleurs proposée. Mais il n'en a pas toujours été ainsi, alors plongeons dans les systèmes embarqués du passé…

Aujourd'hui, même les microcontrôleurs bas de gamme tels que l'AVR ATtiny à 8 broches ont jusqu'à 8 Ko de mémoire flash et jusqu'à 512 octets d'EEPROM et de SRAM. Ensuite, il y a l'offre concernant les périphériques : deux minuteries/compteurs, une interface série, un CA/N à quatre canaux, un chien de garde et un comparateur analogique. Il exécutera votre code à 10 MHz et supportera une large plage de tension d'entrée.

En comparaison, la vie des premiers développeurs de systèmes embarqués était difficile. Ils s'appuyaient sur des microprocesseurs dépourvus de mémoire sur puce (à l'exception des registres) ou de périphériques. Au lieu de cela, toutes ces bonnes choses devaient être ajoutées pour former un système complet.

Cela signifiait qu'il fallait attacher d'autres périphériques au bus du microprocesseur, ce qui était très particulier. Généralement, il y avait 16 bits pour les adresses, ce qui permettait d'avoir 64 Ko de mémoire, et 8 bits pour les données. Il y a ensuite divers signaux de contrôle, tels que les signaux de lecture/écriture et de synchronisation. Toutes les puces périphériques que vous connectez ont principalement une seule fonction. Pour la mémoire, vous incluez une ROM (préprogrammée ou EPROM) et de la RAM, qui doit être dynamique si vous voulez une quantité de mémoire décente. Cela nécessitait des temps de rafraîchissement bigrement compliqués, nécessitant souvent l'emploi d'une puce séparée. Les périphériques étaient également ajoutés séparément. Les timers, compteurs, ports d'entrée/sortie, UART pour la communication série, CA/N, contrôleurs CRT (qui avaient des sorties pour piloter les écrans), et ainsi de suite. Chaque puce était un monstre en deux rangées de 24 à 40 broches, généralement adapté au microprocesseur du même fabri-

Figure 1. Datant du milieu des années 1980, cette carte est équipée d'un microprocesseur 8085. Combinée à six autres cartes (voir l'en-tête de l'article), elle est devenue aussi complète que les microcontrôleurs d'aujourd'hui !

cant, fonctionnant à un maximum de 4 MHz. Mais,le plaisir ne s'arrêtait pas là. Viennent ensuite les circuits intégrés logiques de base pour le décodage et la mise en mémoire tampon, connus sous le nom de logique de liaison. Les puces logiques programmables, telles que les GAL, les PAL et les UAL, étaient populaires pour cette tâche, mais parfois les fabricants intégraient cette fonctionnalité dans des puces personnalisées dédiées.

Intel a ouvert le bal en 1974 avec le vénérable 8080, dont on peut dire qu'il a annoncé le véritable début de l'ère des microprocesseurs. Il avait besoin de l'assistance de deux puces, le générateur d'horloge 8224 et le contrôleur de bus 8228. Fait inhabituel pour les ingénieurs d'aujourd'hui, il nécessitait également des alimentations de ±5 V et de +12 V. Les versions ultérieures, comme le 8085 (**figure 1**), n'avaient besoin que d'une alimentation de +5 V. Cependant, il était plus complexe en raison de son étrange schéma de multiplexage de bus. Cela nécessitait une logique de liaison supplémentaire, bien qu'une ligne de puces périphériques 8085 dédiées était disponible.

Pendant le développement, les programmes étaient stockés dans des EPROM UV, mémoires mortes programmables effaçables par exposition aux rayons ultraviolets. Grâce à une petite fenêtre au-dessus de la puce, le contenu pouvait être effacé à l'aide d'un effaceur d'EPROM, une boîte avec une ampoule UV, une minuterie et un plateau coulissant pour l'EPROM. La lumière UV effaçait le contenu en 20 à 30 minutes. Ensuite, elle pouvait être reprogrammée à l'aide d'un programmateur d'EPROM branché sur le port série ou parallèle de votre PC. Je ne sais pas comment on procédait avant l'avènement des PC ! Cependant, je me souviens d'avoir vu des programmateurs munis de claviers au format hexadécimal entrer des données dans chaque emplacement de mémoire…

La programmation se faisait en code machine, en entrant des valeurs hexadécimales dans chaque emplacement de mémoire. L'autre option était le langage assembleur qui utilisait des mnémoniques pour les opérations et des étiquettes pour les variables et les sections de code. Des instructions telles que « ADC B » signifiaient « ajouter le contenu du Registre B au Registre A, avec indication de retenue ». Les langages tels que le C n'étaient qu'une lueur dans l'œil de leur créateur, même si, avec un peu de chance, vous pouviez obtenir un interpréteur BASIC. Si votre programme ne fonctionnait pas ou devait être réécrit, il vous suffisait d'effacer l'EPROM et de recommencer la programmation. Une fois que vous aviez votre programme final, vous pouviez le faire graver sur des mémoires ROM. Ces dernières étaient moins chères, mais il était impossible de les effacer !

D'autres fabricants de microprocesseurs ont également pris le train en marche. Motorola avait son 6800, MOS Technology le 6502 et National Semiconductor le SC/MP. Certains anciens concepteurs d'Intel ont formé Zilog, qui a proposé le Z80 (**figure 2**), une sorte de 8080 de luxe. La plupart de ces puces ont été utilisées dans les premiers ordinateurs personnels, tels que les Sinclair ZX80/81 et Spectrum (Z80), le Commodore 64 (variante 6502), et bien d'autres. Le projet SC/MP d'Elektor datant de 1978 est un autre bon exemple [1].

En 1980, Intel a introduit le 8051. Celui-ci disposait d'une petite quantité de ROM (parfois EPROM) et de RAM, mais aussi de quatre ports 8 bits, d'un UART et de deux compteurs/*timers*. Avec tout cela dans un seul système, on peut dire que c'est là que l'ère des microcontrôleurs a commencé. Le reste, comme on dit, appartient à l'histoire. L'architecture 8051 est toujours utilisée aujourd'hui, mais avec beaucoup plus de fonctions périphériques que ses ancêtres, tout en se débarrassant des bus et de la logique d'interconnexion du passé.

Le 8051 et les dispositifs 68xx de Motorola ont donné naissance à la gamme riche et polyvalente de microcontrôleurs que nous connaissons et aimons aujourd'hui : AVR, PIC, et une multitude d'autres basés sur ARM, donnant naissance à des cartes comme Arduino et Raspberry Pi. Intel a transformé le 8085 en 8086 sur 16 bits, utilisé dans le premier PC XT d'IBM, puis en 80286, 80386, 80486 et les Pentiums qui sont encore utilisés dans les PC aujourd'hui. Il convient de noter que les ordinateurs monocartes à base de microprocesseurs étaient encore courants jusque dans les années 1990, plutôt que les microcontrôleurs.

Lorsque le 8080 est sorti, j'avais tout juste 18 ans et je me souviens d'avoir lu des articles à son sujet dans les magazines d'électronique de loisir de l'époque. J'ai eu la chance de travailler sur des équipements à microprocesseur au début de ma carrière, ce qui était passionnant. Mes fonctions étaient axées sur un peu de programmation et beaucoup de recherche de pannes. J'ai eu la chance de suivre cette technologie tout au long de ma carrière, et elle m'a toujours fasciné. Comparés aux microcontrôleurs d'aujourd'hui, les anciens microprocesseurs étaient certes particuliers, mais travailler dessus était toujours très amusant.

VF : Laurent Rauber — 230047-04

Figure 2. Le Z80 de Zilog était un autre cheval de bataille des premiers systèmes embarqués. Il est ici accompagné d'un contrôleur d'E/S série (SIO) et d'un circuit compteur/timer (CTC).

LIENS

[1] J. Buiting, « SC/MP Computer d'Elektor », circuits de vacances 2022, Elektor : https://www.elektormagazine.fr/magazine/elektor-264/60879

la documentation des microcontrôleurs sans peine (3)

schémas de principe, et autres documents

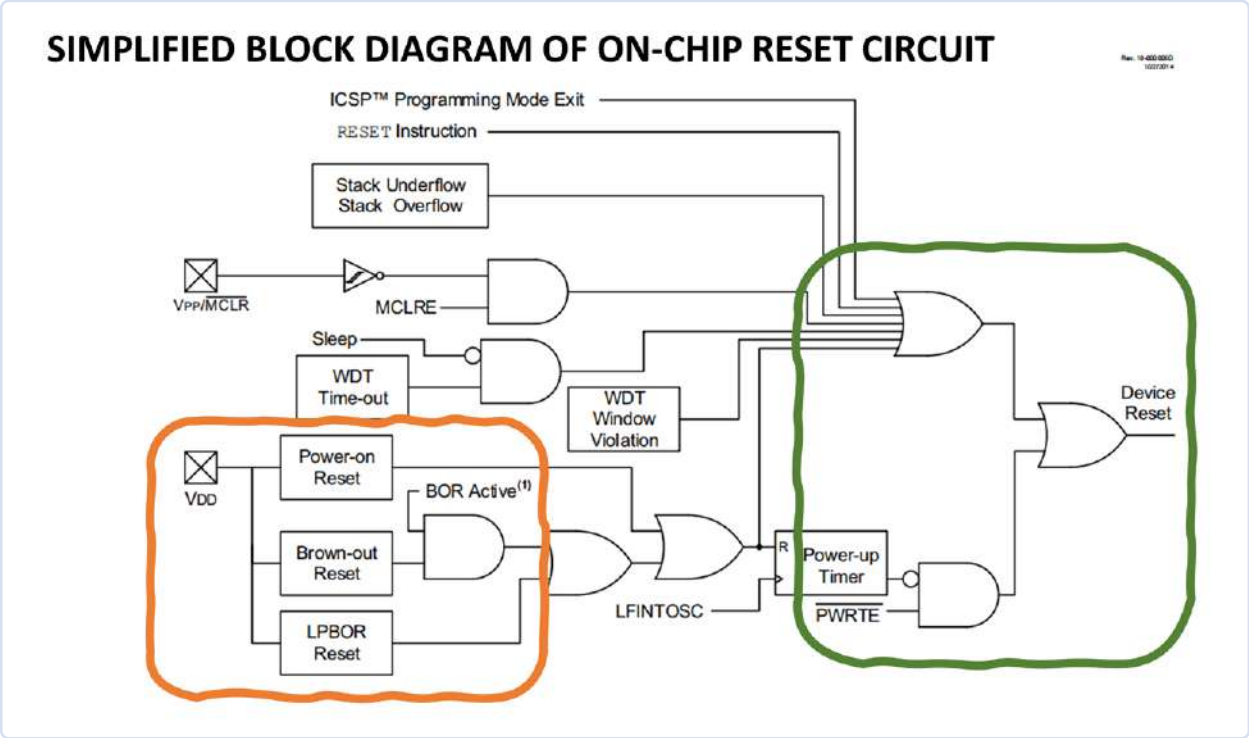


Figure 1. Le circuit de réinitialisation offre très peu d'options de configuration, certaines sources de réinitialisation étant liées à la broche d'alimentation VCC. (Source : Microchip Technology)

Stuart Cording (Elektor)

Dans les parties précédentes de cette série d'articles [1], nous avons abordé les fonctionnalités des registres et examiné le schéma de principe d'une horloge. Nous présentons ici d'autres schémas en précisant aussi où trouver le reste de la documentation nécessaire.

Cette fois-ci encore, nous allons nous focaliser spécifiquement sur le microcontrôleur 8 bits PIC16F18877 de Microchip Technology et nous vous conseillons donc de télécharger la fiche technique [2] si vous souhaitez suivre notre démarche.

Commençons par la réinitialisation

Si le périphérique d'horloge est la partie la plus importante d'un microcontrôleur, la mise en œuvre du circuit de réinitialisation est le deuxième bloc essentiel. Tout comme l'horloge, il ne doit être configuré qu'une seule fois (s'il existe des options de configuration), et vous éviterez bien des problèmes ultérieurs si vous comprenez bien ce qui peut provoquer une réinitialisation.

Dans l'exemple de la *page 100* (**figure 1**), nous pouvons voir que deux broches (représentées par des carrés contenant une croix,

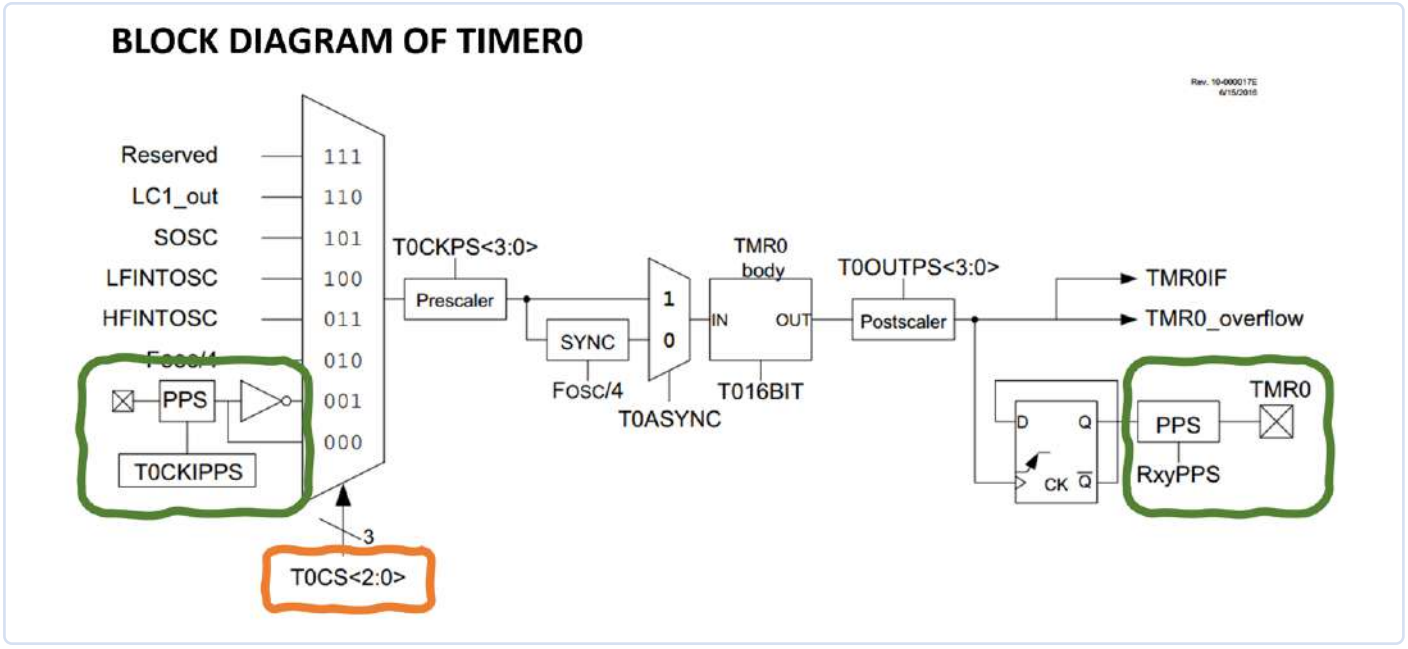


Figure 2. Le schéma de principe de `TIMER0` *montre que le dispositif possède plusieurs sources d'horloge et qu'il peut même produire un signal de sortie en plus des signaux d'interruption internes. (Source : Microchip Technology)*

du côté gauche) peuvent effectuer des réinitialisations. Elles viennent s'ajouter à différentes autres sources internes. L'une d'entre elles est l'instruction `RESET`, deux autres résultent de situations de débordement de la pile (sur- ou sous-capacité), tandis que d'autres encore sont liées au temporisateur « chien de garde ».
Représentés en orange, une série de mécanismes de réinitialisation, notamment la détection de la mise sous tension et les baisses de tension électrique (*brown-out*), sont liés à la broche d'alimentation du microcontrôleur. Dans le cas présent, le dispositif propose un contrôle limité, toutes les sources d'interruption étant pour l'essentiel des signaux de réinitialisation possibles (représentés en vert). Comme nous l'avons vu dans un précédent article avec le registre `STATUS`, il est possible dans certains cas de déterminer la cause de la réinitialisation précédente .

Schémas de principe des périphériques

Les schémas de principe des périphériques sont aussi divers que complexes. Nous avons choisi ici une partie d'un simple temporisateur (`TIMER0`) *page 396* (**figure 2**). Représenté en orange, un groupe de bits d'un registre (vraisemblablement unique) permet une sélection parmi différentes sources d'entrée pour ce périphérique `TIMER0`. En vert, nous voyons également qu'une broche du périphérique peut servir de source, et que le bloc a la possibilité d'émettre un signal vers une broche .
Bien que vous puissiez reconnaître la bascule D [3],sur le côté droit, connectée à la broche de sortie, différents blocs du schéma sont tout simplement des carrés. Les représentations de certains, comme le prédiviseur (prescaler) et le postdiviseur (postscaler), vont de soi. Il en va de même pour le bloc `SYNC`, mais un développeur devra probablement lire le texte d'accompagnement en détail pour comprendre pleinement sa fonction. Les diagrammes et le texte vont ainsi souvent de pair dans les fiches techniques. Il est parfois nécessaire d'écrire quelques lignes de code de test pour vraiment comprendre le fonctionnement de certains périphériques.

Création d'un circuit imprimé

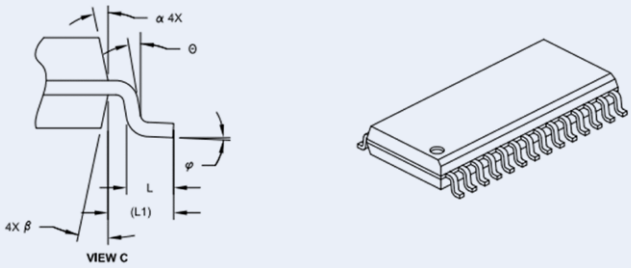
À un moment donné, le dispositif doit être mis en œuvre sur un circuit imprimé pour le produire en série, si nécessaire. La plupart des logiciels de conception assistée par ordinateur (CAO) contiennent les blocs schématiques de votre processeur, ainsi que l'« empreinte » des différents boîtiers pour la conception du circuit imprimé. Toutefois, s'ils font défaut, la fiche technique contiendra probablement des dessins techniques pour chaque boîtier et l'empreinte associée. Dans le cas présent, ils se trouvent à la *page 639*.
Le boîtier SOIC illustré à la **figure 3** est présenté *pages 645 et 646*. Pour les concepteurs de circuits imprimés, l'information la plus utile aujourd'hui est peut-être la hauteur du boîtier si la production du dispositif est en volume limité (ils connaissent déjà la largeur et la longueur). De nos jours, de nombreux boîtiers comportent également une « pastille métallique » intégrée qui doit être soudée au circuit imprimé pour une meilleure dissipation thermique. Ce n'est pas le cas ici. Sinon, des indications sur la manière de le connecter seraient données. Vous devrez peut-être aussi vérifier si cette pastille est connectée à la masse de l'appareil ou à une autre broche.

Quelles sont les informations facultatives d'une fiche technique de microcontrôleur ?

Comme vous le savez maintenant, une fiche technique contient de nombreuses informations. Toutefois, certains éléments sont facultatifs car ils rendraient la fiche trop volumineuse ou parce qu'ils sont communs à de nombreux appareils et méritent une documentation spécifique. Il s'agit notamment des éléments suivants :

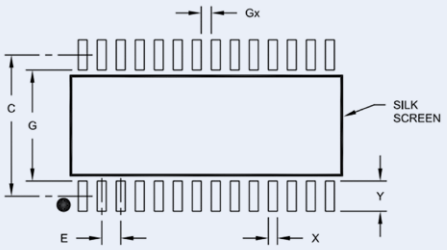
- Jeu d'instructions du processeur : si le nombre d'instructions est faible (moins de 60 à 70), elles peuvent toutes être incluses, avec une explication, dans la fiche technique. Si ce n'est pas le cas, il existe probablement un document séparé dans lequel chaque instruction sera expliquée en détail.
- Exemples de code : ils apparaissent plus souvent lorsque le microcontrôleur est encore essentiellement programmé en assembleur. Les exemples de code en langage de haut niveau, le C par exemple, n'ont guère de sens, puisque le compilateur définit les instructions précises utilisées.
- Exemples de circuits : ils sont le plus souvent associés à des fonctionnalités propres au microcontrôleur sélectionné, comme l'oscillateur ou l'alimentation, ou à des interfaces

28-Lead Plastic Small Outline (SO) - Wide, 7.50 mm Body [SOIC]



Units		MILLIMETERS		
Dimension Limits		MIN	NOM	MAX
Number of Pins	N		28	
Pitch	e		1.27 BSC	
Overall Height	A	-	-	2.65
Molded Package Thickness	A2	2.05	-	-
Standoff §	A1	0.10	-	0.30
Overall Width	E		10.30 BSC	
Molded Package Width	E1		7.50 BSC	
Overall Length	D		17.90 BSC	
Chamfer (Optional)	h	0.25	-	0.75
Foot Length	L	0.40	-	1.27
Footprint	L1		1.40 REF	
Lead Angle	θ	0°	-	-
Foot Angle	φ	0°	-	8°
Lead Thickness	c	0.18	-	0.33
Lead Width	b	0.31	-	0.51
Mold Draft Angle Top	α	5°	-	15°
Mold Draft Angle Bottom	β	5°	-	15°

28-Lead Plastic Small Outline (SO) - Wide, 7.50 mm Body [SOIC]



Units		MILLIMETERS		
Dimension Limits		MIN	NOM	MAX
Contact Pitch	E		1.27 BSC	
Contact Pad Spacing	C		9.40	
Contact Pad Width (X28)	X			0.60
Contact Pad Length (X28)	Y			2.00
Distance Between Pads	Gx	0.67		
Distance Between Pads	G	7.40		

Figure 3. Les plans du boîtier indiquent ses dimensions (à gauche) ainsi que la configuration recommandée pour le circuit imprimé (à droite). (Source : Microchip Technology)

légèrement plus complexes qui ont des exigences spécifiques en matière de charge de signaux, comme l'USB. Toutefois, il est très probable que ces circuits soient abordés plus en détail dans une note d'application relative au sujet.

Quelles sont les informations généralement absentes d'une fiche technique de microcontrôleur ?

Les éléments suivants sont normalement exclus des fiches techniques et figurent dans d'autres documents. C'est généralement parce qu'ils concernent un sujet commun à une large gamme de microcontrôleurs.

- Flashage de la mémoire du microcontrôleur : ce sujet est normalement traité séparément pour la programmation des microcontrôleurs dans le cadre de la production de masse.
- Utilisation d'outils de développement : le compilateur, l'environnement de développement intégré (IDE) et les outils de débogage disposent de leur propre documentation.
- Explication détaillée du cœur du processeur : ce point fait souvent l'objet d'un document séparé, en particulier pour les cœurs 16 et 32 bits.
- Explication détaillée des périphériques complexes : les périphériques USB, les interfaces graphiques et les périphériques Ethernet font généralement l'objet de documents distincts, car l'inclusion de chacun d'entre eux pourrait doubler la taille de la fiche technique du microcontrôleur.
- Errata : cette partie répertorie les erreurs contenues dans la documentation, ou les solutions de contournement pour résoudre les bogues matériels non corrigés.

Quelles sont les autres documentations mises à votre disposition ?

Outre la fiche technique, les documents suivants sont généralement proposés :

- « Manuel de référence de la gamme » : alors que la fiche technique vous indique précisément ce qu'un microcontrôleur peut faire, ces documents offrent une vue globale de ce que peuvent accomplir tous les microcontrôleurs d'une même gamme.
- Notes d'application : cette partie explique en détail comment utiliser des périphériques spécifiques, ou un groupe de périphériques, pour mettre en œuvre une application ou une interface. Si la fiche technique peut décrire, par exemple, comment utiliser un périphérique CAN (*Controller Area Network*), une note d'application indiquera plutôt comment l'exploiter dans le cadre d'un réseau CAN, avec des conseils sur les protocoles logiciels de haut niveau et la sélection d'émetteurs-récepteurs appropriés.
- Spécification de programmation : pour la programmation dans des environnements de production de masse, elle explique en détail les tensions et les temporisations requises, ainsi que le ou les protocoles éventuels utilisés pour l'interface de programmation.

Comment assimiler ces documentations volumineuses ?

Malheureusement, il n'existe pas de moyen particulier pour assimiler rapidement toutes les informations associées à un microcontrôleur et à l'ensemble de ses outils. Si vous débutez, il est probablement préférable de lire la fiche technique parallèlement à un livre ou à un article sur le microcontrôleur concerné. Elektor ne propose pas seulement des livres pour ceux qui s'intéressent aux microcontrôleurs PIC ; il existe également des livres de démarrage pour les microcontrôleurs STM32 [4] et ARM [5] en général. Et pourquoi ne pas essayer un simple projet MSP430 [6] ?

La combinaison d'exemples pratiques et de deux types de documents explicatifs écrits (un livre/article et la fiche technique elle-même) pourra vous aider. Si vous êtes plus avancé(e), la meilleure approche consiste à vous focaliser sur les sections qui concernent le cœur du processeur, l'arborescence d'horloge et le bloc de réinitialisation, puis les périphériques que vous souhaitez utiliser. Vous devrez ensuite vous familiariser avec la

documentation de la chaîne d'outils du compilateur. Faites également bon usage des bibliothèques logicielles et des exemples pour améliorer votre compréhension. Si vous avez besoin d'aide supplémentaire, posez des questions sur les forums.
Les fiches techniques peuvent être difficiles à lire et à comprendre, et elles ne sont pas particulièrement enthousiasmantes sur le plan littéraire. Pour autant, elles sont (la plupart du temps) exactes et constituent une description technique de la fonctionnalité (la section « errata » peut servir à déterminer la confiance à accorder à la fiche consultée). Gardez-les à portée de main, et, avec le temps, vous comprendrez bien comment elles sont construites et formulées.

VF : Pascal Godart — 230286-04

Des questions, des commentaires ?

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

Produits

- **T. Hanna, *Microcontroller Basics with PIC*, Elektor 2020**
  https://elektor.fr/19188
- **A. Pratt, *Programming the Finite State Machine*, Elektor 2020**
  https://elektor.fr/19327

LIENS

[1] Série relative à la documentation des microcontrôleurs : https://elektormagazine.com/tags/microcontroller-documentation
[2] Fiche technique du microcontrôleur Microchip Technologies PIC16F18877 : https://microchip.com/en-us/product/PIC16F18877
[3] Bascules D: https://fr.wikipedia.org/wiki/Bascule_(circuit_logique)
[4] Livre sur les microcontrôleurs STM32 pour débutants: https://elektor.com/programming-with-stm32-nucleo-boards-e-book
[5] Microcontrôleurs à base de processeurs ARM : https://elektor.fr/embedded-in-embedded
[6] Projet MSP430 simple : https://elektormagazine.fr/labs/elektorpost-no-1-led-earring

station météo LoRa à faible puissance

réalisez vous-même une station météo à longue portée

Edward Ringel (États-Unis)

Lorsque mon ancienne station de température et d'humidité à distance a fini par tomber en panne, je l'ai remplacée par un système de ma propre conception : un capteur à l'extérieur et un affichage à l'intérieur tous deux alimentés par batterie et connectés via LoRa. Mettre au point un dispositif fiable capable de supporter l'hiver du nord-est des États-Unis n'a pas été simple ! Mais voyez plutôt…

Figure 1. Capteur enneigé suspendu près de notre voie d'accès.

La fonctionnalité minimale du système était le développement d'un capteur extérieur distant, alimenté par piles (**figure 1**), et d'un affichage intérieur, lui aussi équipé d'un capteur et alimenté par piles. Pendant le développement du projet, j'ai ajouté un serveur web très simple.

Pour ce faire, je voulais utiliser l'environnement Arduino et ses nombreuses bibliothèques de haute qualité. Pour simplifier le projet, je prévoyais d'utiliser des composants sur étagère et qu'on puisse souder à la main. Enfin, la réduction de la consommation d'énergie était une priorité. Imprévue au départ, la climatisation efficace du capteur extérieur s'est avérée étonnamment difficile.

Des considérations générales m'ont permis de déterminer quel matériel / quelles cartes devaient être utilisés – voir plus loin pour en savoir plus et les difficultés rencontrées. Tout d'abord, il faut une station extérieure, une station intérieure et une station de base. Cette dernière collecte les données des deux stations de mesure et les transmet à un ordinateur. Les trois unités ont donc besoin d'au moins une carte microcontrôleur et d'une unité radio pour communiquer entre elles. Les deux stations de mesure pour l'extérieur et l'intérieur ont également besoin de capteurs et d'une commande de minuterie, dont il sera question plus loin. Les trois stations sont constituées des « briques de base » suivantes :

- Station extérieure : Artemis Nano, module radio LoRa, minuterie, capteurs
- Station intérieure : Artemis Thing Plus, module radio LoRa, minuterie, capteurs, affichage
- Station de base ou station domotique : Raspberry Pi Pico, Arduino Mini, module radio LoRa

Parlons de consommation

Des mesures de température, d'humidité et de pression barométrique sont effectuées périodiquement. La consommation d'énergie est minimisée en plaçant le dispositif de mesure à distance dans un état de faible consommation ou de veille, en le réveillant, en effectuant et en transmettant les relevés, puis en le ramenant dans un état de faible

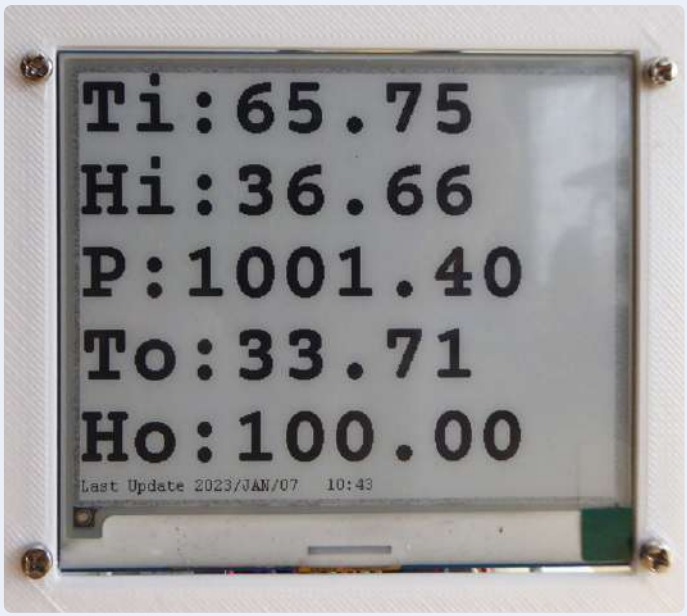


Figure 2. Mon écran ePaper affiche la température (en °F), l'humidité relative et la pression barométrique non corrigée/non normalisée. Les indices « i » et « o » désignent l'intérieur et l'extérieur. Seule la pression extérieure est affichée.

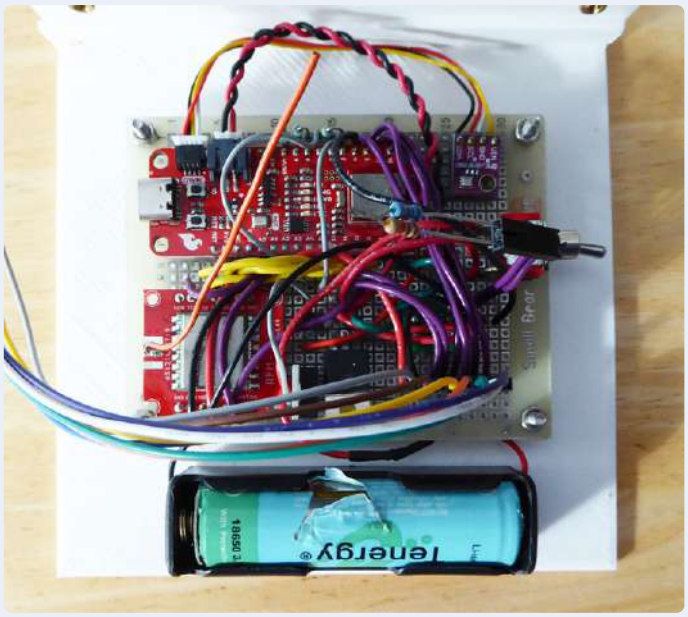


Figure 3. Première itération de la carte d'affichage. Première version du circuit d'affichage, avant la modification de la carte Artemis et l'ajout du TPL5111. Optocoupleurs, MOSFETs, TPL5110, interrupteurs pour la recharge, youpi !

consommation. Le rapport cyclique de la consommation d'énergie est représenté par T_{ON} / T_{TOT} . La consommation d'énergie effective est régie par :

- La consommation de l'appareil lorsqu'il est en marche et qu'il traite des instructions ou des données.
- La fraction du temps pendant laquelle l'appareil est en marche pour collecter et envoyer (ou afficher) des données - la minimisation du cycle d'utilisation affecte de manière significative la consommation globale.
- La consommation de l'appareil entre les mesures. Idéalement, il ne devrait pas y avoir de consommation entre les instants d'échantillonnage, mais ce n'est pas une attente réaliste.

J'ai testé la consommation d'énergie de base de plusieurs cartes microcontrôleurs en les mettant sous tension sans exécuter de programmes ni attacher de périphériques. Voir le **tableau 1** pour les résultats (ceux-ci peuvent ne pas représenter la consommation d'énergie après un réglage fin tel que la désactivation des interfaces périphériques, le ralentissement de l'horloge, etc.) À la lumière de ces résultats, j'ai décidé d'utiliser les cartes Artemis de Sparkfun.
L'utilisation de l'énergie a également été un facteur dans la conception de la station d'affichage intérieure, mais choisir l'écran a été facile : ePaper a la plus faible consommation d'énergie. Cette technologie ne consomme du courant que lors du rafraîchissement de l'affichage (c'est pourquoi le Kindle équipé d'ePaper peut afficher un graphique alors qu'il est inactif et ne consomme pas d'énergie). Le WaveShare dispose d'un bel écran ePaper de 10,7 cm. Les instructions étaient difficiles à suivre et j'ai dû générer quelques polices de caractères, mais comme le montre la **figure 2**, l'écran a bien fonctionné.
J'ai utilisé des capteurs Bosch BME280 pour les mesures. Ces unités mesurent la température, l'humidité et la pression barométrique (T/H/P) et disposent de bibliothèques I²C robustes. Leur consommation de courant est négligeable.
Un mécanisme de minuterie contrôlait le rapport cyclique. La carte microcontrôleur, la radio et le capteur devaient être allumés et éteints - cela n'aurait eu aucun sens d'utiliser une carte microcontrôleur efficace qui consommait 20 µA en veille alors que la radio consommait 20 mA. Plutôt que de coder la mise en veille du microcontrôleur et la mise en veille de la radio, j'allumais et j'éteignais l'ensemble de l'appareil.
Le régulateur 3,3 V de la carte Artemis a une puissance de sortie maximale de 600 mA, suffisante pour alimenter tous les composants de l'affichage intérieur de même que le capteur extérieur / la radio. Comme je devais couper l'alimentation des cartes, la meilleure solution était une puce de minuterie à très faible consommation. Ces dispositifs possèdent un circuit de minuterie interne, dont l'intervalle est réglé par une simple résistance externe, et s'utilisent avec un microcontrôleur. Lorsque la minuterie se déclenche, le flux de courant est activé. Lorsque le microcontrôleur a terminé sa tâche, il envoie un

Tableau 1. Cartes et consommation

Carte	Consommation électrique (mA)
Teensy 3.5	74,5
Raspberry Pi Pico	20,4
Teensy 4.1	92,6
ESP32 Adafruit Huzzah Feather	125,0
Artemis Thing Plus	8,5
Generic ESP32-WROOM	70,3

Consommation d'énergie de différents microcontrôleurs sous tension, alimentés en 5 V pour V_{in}, mais sans exécution de programme. La consommation d'énergie est régie, non seulement par le microcontrôleur, mais aussi par d'autres composants de la carte (LED, régulateurs de tension, mémoire externe, configuration sans fil, etc.)

message *DONE* à la minuterie, et le flux de courant est désactivé. J'ai travaillé avec la puce TPL5110 sur une carte maison et sur la carte Adafruit, toutes deux en combinaison avec une première version de la carte d'affichage (**figure 3**). Mes tribulations avec ce petit truc sont documentées dans [1].
J'ai finalement utilisé un TPL5111 connecté à la broche d'activation du régulateur 3,3 V sur les cartes Artemis (**figure 4**). Cette implémentation a nécessité la suppression de la résistance CMS de rappel R1 sur l'Artemis Nano (**figure 5**), ou R3 sur l'Artemis Thing Plus (**figure 6**), mais m'a permis de contrôler toute l'alimentation de la carte. Ces résistances rappellent la broche *ENABLE* du régulateur de tension au plus, sauf si la broche est mise à la masse. Pour finir, les cartes modifiées ont constitué une solution astucieuse, le circuit complet ne consommant que 20 µA en veille.
Les TPL5110 et TPL5111 sont des puces intéressantes qui méritent d'être étudiées, en particulier dans le domaine des capteurs IdO distants. Les cartes Adafruit sont des implémentations formidables et offrent une excellente flexibilité de conception.

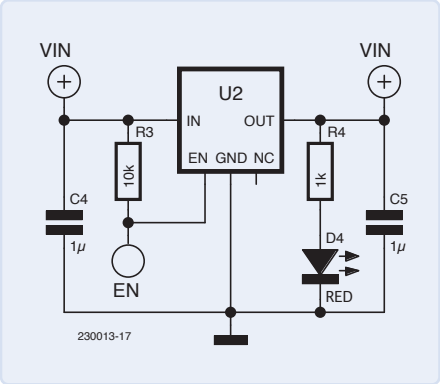


Figure 4. Régulateur de tension Artemis. La broche ENABLE est rappelée au plus par R3 (R1 sur Artemis Nano). Cette résistance doit être retirée. La broche ENABLE est destinée au régulateur de tension de la carte, pas au microcontrôleur.

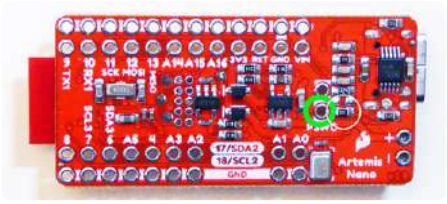


Figure 5. Carte Artemis Nano. R1 est entouré en blanc. Contrairement à la Thing Plus, il n'y a pas de connecteur EN mais un connecteur « PSWC ». La broche ENABLE du TPL5111 doit être reliée au trou entouré en vert une fois que R1 est retiré. Notez que R1 se trouve à l'arrière de la carte et doit être retirée avant le montage.

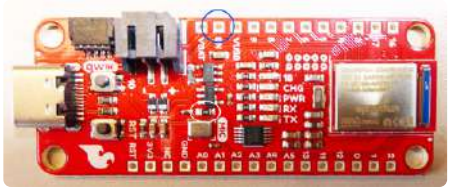


Figure 6. Carte Artemis Thing Plus. R3 est entourée en blanc. La broche ENABLE du TPL5111 peut être reliée directement à la broche EN (entourée en bleu) après que la résistance a été retirée.

Transmission des données

J'ai choisi les cartes HopeRF LoRa à 915 MHz pour la transmission des données (d'autres pays peuvent exiger une fréquence différente). De nombreuses bibliothèques sont disponibles pour l'environnement Arduino et simplifient la mise en œuvre. La fréquence porteuse proche du gigahertz permet une meilleure pénétration des murs et des bâtiments et de plus grandes distances de transmission. Les protocoles de transmission sont optimisés pour les petits paquets de données. Ils n'ont pas besoin de signaux d'échange qui prennent du temps ni de surcharge de sécurité. En revanche, la charge de garantir la livraison des données repose sur le programmeur plutôt que sur les protocoles de transmission intégrés.
À ce stade, vous pouvez vous demander comment le capteur et les stations d'affichage communiquent. Après tout, les temps T_{ON} des deux appareils se chevauchent rarement. Si le capteur s'allume et transmet des données, la station d'affichage sera probablement éteinte. La solution a consisté en une troisième station, toujours active, basée sur un Raspberry Pi Pico ne contenant pas de capteurs mais fonctionnant comme un concentrateur et un transmetteur de données, relié à un ordinateur. Je l'appellerai la station domotique. Son seul

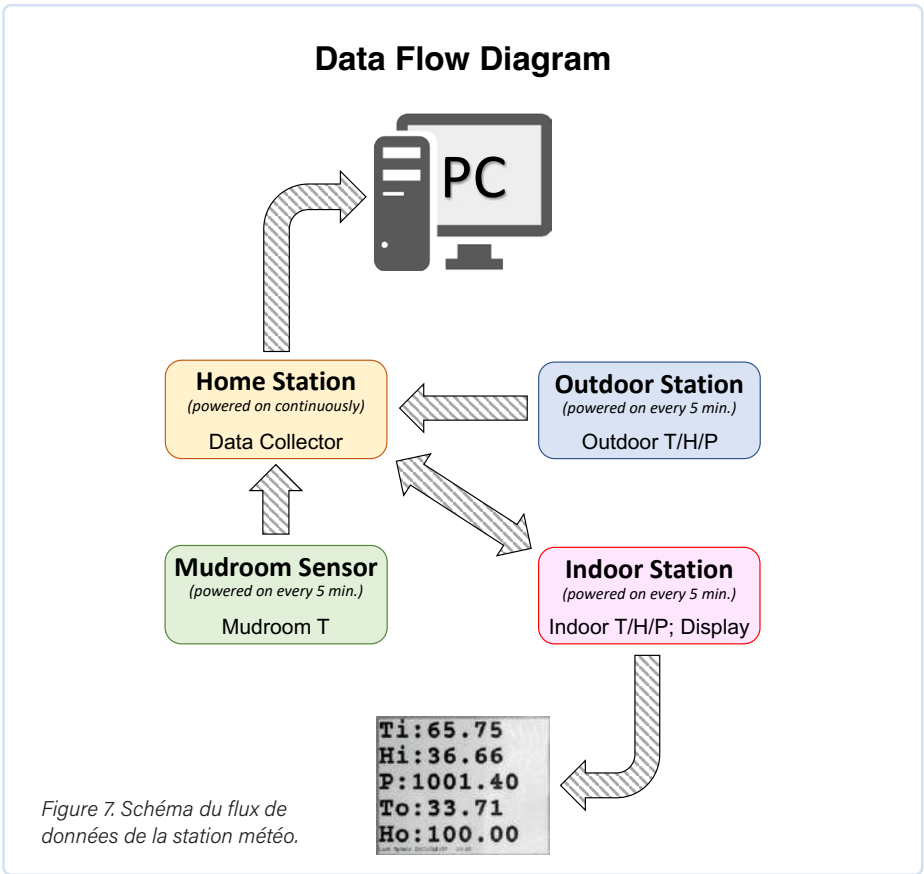


Figure 7. Schéma du flux de données de la station météo.

Figure 8. Carte du capteur noyée dans la résine. Notez le petit carré de plastique dans le coin supérieur gauche qui empêche la résine de recouvrir le capteur. Notez également l'adaptateur USB-C à angle droit (à gauche, au milieu) qui permet d'accéder à la carte pour la recharge et les mises à jour logicielles.

Figure 9. Détail du boîtier du capteur. Tous les orifices d'aération sont grillagés pour protéger des attaques biologiques plus que des cyberattaques ;-).

périphérique est la radio et quelques LED pour le débogage. Son régulateur de tension peut supporter un courant de quelques centaines de milliampères, et j'ai alimenté la radio à partir de ce régulateur. Le Pico est alimenté par le port USB d'un PC et communique en série grâce à un programme en Processing. Quelques centaines de lignes de code suffisent pour tracer la pression barométrique et afficher un tableau des relevés sur une page web. La **figure 7** illustre le flux de données comme suit :

1. Le capteur extérieur se réveille toutes les cinq minutes et envoie les relevés de température, d'humidité et de pression (T/H/P) à la station domotique. Il n'y a pas de retour entre la station domotique et le capteur extérieur. Le capteur extérieur s'éteint après la transmission sans confirmation de réception par la station domotique.
2. La station domotique stocke les relevés les plus récents du capteur extérieur en mémoire vive. Les relevés extérieurs sont également envoyés à l'ordinateur pour traitement ultérieur.
3. La station intérieure d'affichage avec son capteur se réveille toutes les 5 minutes. Elle mesure les paramètres intérieurs T/H/P, envoie ces informations à la station domotique et conserve une copie des mesures en mémoire vive.
4. Lorsqu'elle reçoit une demande de mise à jour de la station d'affichage, la station domotique transfère cette demande à l'ordinateur. Elle envoie alors les observations extérieures les plus récentes à la station intérieure, qui met alors à jour l'affichage et s'éteint.

Au fil du temps, j'ai constaté des blocages répétés de la station domotique avec pour conséquence l'impossibilité de mettre à jour l'affichage et le programme en Processing. Les messages postés sur plusieurs forums d'assistance suggèrent que la puce de l'émetteur-récepteur LoRa SX1276 peut se figer et provoquer ce comportement, bloquant à la fois la radio et le microcontrôleur lorsqu'elle est laissée en marche pendant de longues durées. Ce comportement n'a jamais été constaté avec la station d'affichage ni celle de mesure, qui ont des cycles de mise sous et hors tension fréquents.

La solution pour la Pico a été d'utiliser un second microcontrôleur, un Arduino Mini, comme minuterie intelligente. La sortie du Mini était connectée à la broche d'activation du Pico, ce qui permettait d'éteindre le régulateur de tension pendant 500 ms toutes les heures. Une sortie de la Pico était connectée à la Mini, signalant le moment où le flux de données serait le moins vulnérable aux effets d'une réinitialisation : la réinitialisation du matériel ne se produirait qu'après une mise à jour des données et alors que le Pico ne recevrait pas de nouvelles informations.

Bien que cela puisse sembler être une utilisation extravagante d'un microcontroleur, les appareils sont si bon marché que cela se justifie. J'encourage la communauté Elektor à me faire part de ses commentaires sur la façon de mieux résoudre ce problème. Deux versions du logiciel de la station domotique sont fournies [2], l'une destinée à être utilisée avec le programme en Processing, l'autre indépendante d'une connexion informatique et ne nécessitant qu'une alimentation électrique telle qu'un vieux chargeur de téléphone avec le connecteur USB approprié. Le code en Processing est encore en chantier et ne prétend pas être une solution complète ni exempte de bogues.

Climatisation

Au début, mon capteur extérieur tombait en panne par intermittence en cas de changements rapides de température et d'humidité, et je soupçonnais la condensation. Après avoir examiné toutes mes soudures et mon câblage et n'avoir trouvé aucun défaut, j'ai décidé d'enrober de résine ma carte extérieure. J'ai imprimé en 3D un boîtier pour mon circuit imprimé et une petite protection à placer autour de la carte du capteur pour qu'elle ne soit pas recouverte par la résine (**figure 8**). Je me suis servi d'un adaptateur USB-C à 90° pour conserver l'accès au connecteur USB-C de la carte Artemis.

J'ai utilisé de la résine époxy marine

Liste des composants

Résistances

R1 = voir texte
R2 à R4 = 1 M Ω ; 1/4 W
R5 = 680 Ω ; 1/4 W
R6 à R8 = 360 Ω ; 1/4 W

Semi-conducteurs

D1 = LED bleue
D2 = LED rouge
D3 = LED jaune
D4 = LED verte

Cartes

2 × BoB TPL5111 (Adafruit Industries)
1 × Raspberry Pi Pico
1 × Artemis Thing Plus (Sparkfun)
1 × Artemis Nano (Sparkfun)
2 × cartes de connexion BME280 (ASIN B08DHTGNHR *)
1 × BoB FTDI basique 3.3 V (Sparkfun)
3 × Émetteurs-récepteurs LoRa HopeRF RFM95CW 915 MHz certifiés FCC (Anarduino.com)
1 × Arduino Pro Mini, 3,3 V 8 MHz (voir texte)

Autres

2 × Connecteurs ST à 4 broches (ASIN B01DUC1M2O* ; voir Conseils de réalisation)
1 × Écran ePaper 10,7 cm (WaveShare ; ASIN B074NR1SW2*)
3 × Plaques perforées FR4 double face 7 X 9 cm (ASIN B08F7X8JHV*)
2 × Batteries rechargeables LiPo18560
2 × Supports pour les batteries 18650
Kit de résine époxy (ASIN B07TVWTG829*)
Adaptateur USB-C à 90° (ASIN B0BBVWF54L*)
Barrettes et fils selon les besoins

* Les codes ASIN font référence à des numéros d'inventaire Amazon consultables.

transparente bon marché, disponible sur Amazon. L'époxy marine est formulé pour durcir très lentement et donne à l'utilisateur plus de temps pour travailler avec le matériau. Comme je ne recherchais pas un résultat esthétique parfait et que je ne voulais pas que mes idiots de chats soient couverts d'époxy, j'ai placé le récipient sur un chauffe-plats électrique à environ 65° C. En moins d'une heure, l'époxy était à l'épreuve des pattes. J'ai ensuite laissé le durcissement se terminer pendant la nuit à température ambiante (18-20° C). Le circuit imprimé est logé dans un bocal en plastique ventilé peint en blanc (**figure 9**) et les trous de ventilation sont grillagés pour protéger le capteur contre les insectes.
Avant de résiner le circuit, j'ai confirmé, par des tests, que l'époxy ne court-circuiterait pas les fils nus et ne perturberait pas les connexions électriques. Depuis que j'ai résiné le circuit, il n'y a plus eu de pannes intermittentes.

Logiciels et données

Le logiciel pour les trois stations utilise les bibliothèques LoRa et le code WaveShare. Pour les stations intérieure et extérieure, il faut des bibliothèques pour le capteur BME280. Vous pouvez télécharger gratuitement les paquets complets à l'adresse [2].
La charge de codage pour le TPL5111 est triviale. J'ai échangé la simplicité du code et le raccourcissement du T_{ON} contre la garantie de transmission des données. Après la mesure, une courte chaîne est créée qui contient les observations et l'identification de la station d'envoi ; cette chaîne est envoyée deux fois (voir ci-dessous). Les transmissions de la station domotique à la station d'affichage comprennent également des données temporelles afin que l'affichage puisse indiquer l'heure de la dernière mise à jour.
On dirait que le code n'est pas complet. En réalité, l'exécution du code s'arrête lorsque le régulateur de tension est coupé. Pour cela, l'unité centrale envoie un signal *DONE* à la minuterie, ce qui permet de combler les lacunes du logiciel. À l'exception du code WaveShare, je n'ai pas inclus de bibliothèques dans le téléchargement du logiciel. Celles-ci sont facilement ajoutées à votre environnement par l'EDI Arduino. J'ai utilisé la bibliothèque *Sparkfun BME280* parce qu'il n'y a aucune dépendance de code autre que *Wire.h*. La bibliothèque *Sandeep Mistry LoRa* a fourni toutes les fonctionnalités nécessaires et n'avait aucune dépendance de code autre que *SPI.h*.
Comme la station domotique ne peut pas coordonner les transmissions entre les capteurs et la station d'affichage, les collisions de données et les pertes de données qui en résultent sont inévitables. La chaîne de données est envoyée deux fois avec un délai entre les transmissions pour réduire le risque de données non délivrées. Au global, la perte de quelques points de données par jour est acceptable pour une application non critique.
Au cours du développement, j'ai réalisé que la station domestique pourrait communiquer avec un ordinateur et envoyer les observations météorologiques à une base de données, une page web ou autre. Il s'agit d'un travail en cours. J'ai écrit un serveur de pages web rudimentaire en Processing, un langage dérivé de Java avec une base d'utilisateurs étendue et de nombreuses bibliothèques de qualité. Comme indiqué, le code Processing est fourni, ainsi que les croquis Arduino. Bien qu'il puisse y avoir des similitudes superficielles, il ne s'agit pas d'un projet LoRaWan, et la station d'accueil n'est pas une passerelle LoRa.
Ma configuration surveille les conditions extérieures, les conditions dans ma cuisine (emplacement de l'affichage) et un troisième capteur dans notre vestibule, qui est exposée au risque de gel en hiver. Les données du capteur du vestibule n'apparaissent que sur la page web. Le nombre de capteurs et d'affichages qui peuvent être pris en charge avec cette configuration est limité. Avec l'augmentation du trafic sur le réseau, la perte de données due aux collisions finira par devenir inacceptable. De nombreux facteurs influent sur ce phénomène, notamment la précision des résistances de synchronisation du TPL5111, les facteurs environnementaux et, surtout, la fréquence d'échantillonnage et/ou de

Conseils de réalisation

1. **Retirer R1 et R3** : Voir les images pour l'identification des résistances R1 et R3 qu'il faut retirer du Nano et de la Thing respectivement. Si vous ne le faites pas, les circuits ne fonctionneront pas.

2. **Connexion ENABLE sur Artemis Nano** : voir figure 5 pour la connexion ENABLE sur Nano. J'ai pris une photo du bas de la carte. Vous ne devez PAS utiliser la broche à la masse !

3. **Circuit imprimé principal** : n'importe quel circuit imprimé peut être utilisé, mais les articles référencés étaient pré-étamés, en FR4, de taille adéquate et résistants aux retouches.

4. **Fréquence radio** : achetez une carte radio légale pour votre pays.

5. **Connexion au système Qwiic** : le connecteur JST mentionné s'adapte parfaitement au système QWIIC, mais les couleurs NE correspondent PAS à la convention Sparkfun. Pour ces connecteurs, blanc = GND, jaune = 3.3 V, noir = SDA, et rouge = SCL.

6. **Cartes BME280** : soyez prudent lors de l'achat de cartes BME. Certains vendeurs d'Amazon remplacent (sciemment ou non) les puces BMP280 par des puces BME280. La version BMP est moins chère et semble presque identique, mais les bibliothèques pour BME ne fonctionneront pas et vous n'aurez pas de données d'humidité.

7. **Notes sur la connexion WaveShare** : l'unité WaveShare est livrée avec un câble facile à utiliser. Violet = BUSY, Blanc = RESET, Vert = DC, Orange = CS, Jaune = CLK, Bleu = DIN (MOSI), Marron = GND, Gris = Vcc. L'unité n'a pas de connexion MISO.

8. **Intervalle de temps du TPL5111** : l'intervalle de temps du TPL5111 est programmé à l'aide d'un potentiomètre intégré ou d'une résistance externe. Le site web d'Adafruit et la fiche technique du circuit intégré fournissent un tableau des valeurs de résistance pour différents intervalles de temps. Si vous utilisez une résistance externe, vous devez couper une trace au dos de la carte. Si la consommation est vraiment critique, la LED d'activité peut également être retirée du circuit en coupant une trace.

9. **Connexion de la batterie aux cartes Artemis** : la batterie se connecte à la carte à l'aide d'un câble JST à 2 fils disponible partout.

10. **Carte d'interface radio** : j'ai utilisé un BoB pour la radio de Diycon.nl (LoRa Node PCB 100 Shield Only pour RFM92/RFM95). Il est facile d'y fixer une antenne filaire ou un connecteur d'antenne.

11. **Antenne radio** : j'ai utilisé une simple antenne filaire ¼ d'onde. La longueur de fil correcte pour 915 MHz est de 78 mm, soudée au connecteur central de la carte radio.

12. **Résine** : très salissante. Gants, récipients jetables pour mesurer et mélanger, bâtons de mélange jetables et chiffon. Assurez-vous que la carte fonctionne parfaitement avant de procéder. L'adaptateur USB-C à 90° doit être installé, et la minuterie doit avoir la bonne résistance. Le récipient peu profond doit être légèrement plus grand que la carte et la batterie. Ne pas immerger le BME280 dans l'époxy. J'ai fait une protection autour du capteur (la résine remontera par les trous, protégez donc aussi le dessous). Vous pouvez également connecter le capteur par des fils au connecteur QWIIC et maintenir la carte du capteur au-dessus de la résine pendant qu'elle durcit. Chauffer doucement la résine accélère considérablement le temps de durcissement. Les trois points clés sont que le capteur ne soit pas recouvert de résine, que le connecteur USB-C reste accessible à travers l'adaptateur et qu'à l'exception de la carte du capteur, tous les composants du circuit imprimé soient recouverts.

13. **Le boîtier** : il y a plusieurs problèmes de conception. Les trous de ventilation doivent être situés et protégés afin que l'eau ne puisse s'infiltrer à l'intérieur du boîtier qu'en contrant la gravité. Deuxièmement, le boîtier doit être réfléchissant ou au moins peint en blanc pour minimiser l'effet de serre. Troisièmement, le boîtier doit être léger. L'inertie thermique d'un boîtier lourd et volumineux rendra vos mesures lentes et imprécises. Enfin, le boîtier doit être protégé d'une manière ou d'une autre contre les insectes.

14. **Batteries 18560** : les batteries vantant des densités d'énergie impossibles à des prix ridiculement bas sont omniprésentes sur Amazon et eBay. Soyez sceptique face à ces affirmations. Les piles de marque avec de 2500 à 3500 mAh sont un choix sûr !

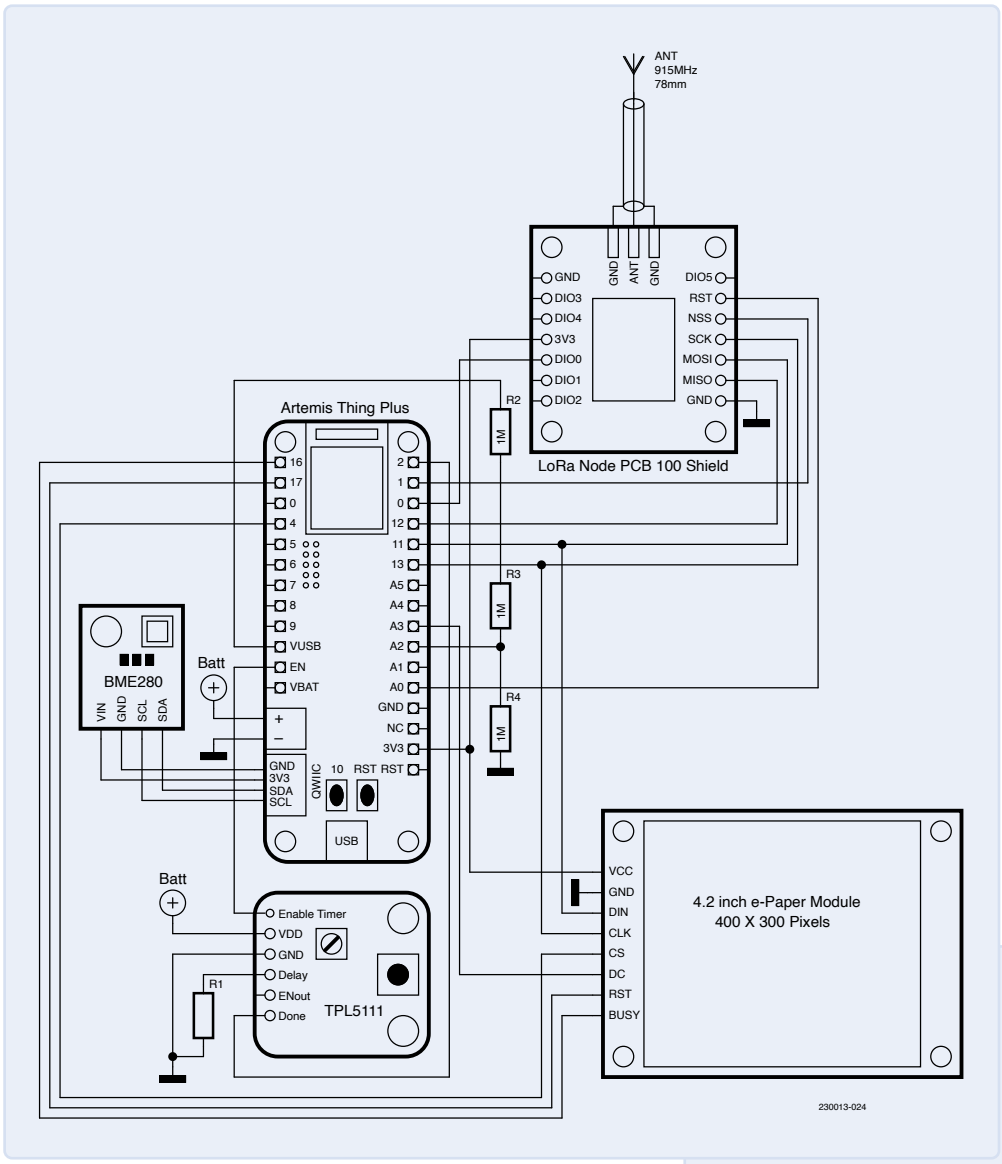


Figure 10. Circuit de la station d'affichage.

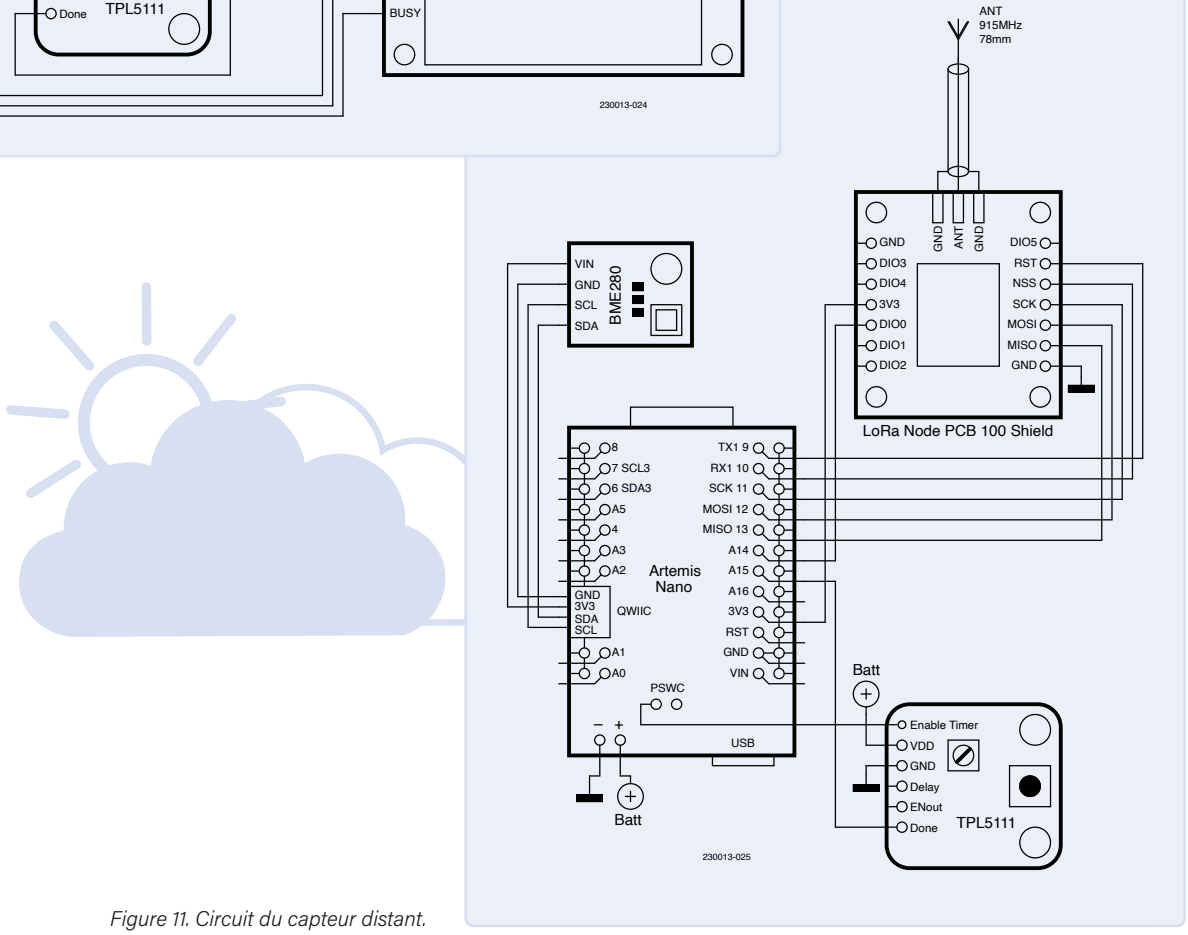


Figure 11. Circuit du capteur distant.

demande de mise à jour. Il n'y a pas de limite au nombre d'affichages n'effectuant pas de demandes de données. Les unités effectuant des demandes de données contribuent au trafic radio. C'est le trafic radio total, et non les transmissions de mesures des capteurs, qui limite le réseau. Le calcul des limites théoriques des différentes configurations est laissé à l'appréciation du lecteur.

Considérations pratiques

Le matériel est câblé à la main. Lisez attentivement l'encadré **Conseils de réalisation**. Les capteurs communiquent via I²C. J'ai utilisé le système Qwiic de Sparkfun, qui est très pratique. Il est facile de câbler un connecteur compatible Qwiic aux petites cartes BME280 disponibles sur Amazon et eBay. Veillez à acheter une BM**E** plutôt qu'une BM**P**280 si vous souhaitez obtenir des informations sur l'humidité. Le module HopeRF a un pas de 2 mm au lieu de 2,54 mm, ce qui nécessite un BoB ou une soudure soignée. J'utilise une simple antenne filaire ¼ d'onde.

J'ai utilisé FontEdit pour créer une police de 36 points pour l'affichage ePaper. Les instructions de WaveShare sont difficiles à comprendre, mais j'ai fini par développer un code fonctionnel. Si vous utilisez un MCU autre que l'Artemis, n'oubliez pas que les tables de polices et le tampon bitmap consomment beaucoup de mémoire vive.

J'ai utilisé une batterie rechargeable au lithium 18560 pour l'alimentation. Voir les **Conseils de réalisation** pour des observations sur l'approvisionnement en 18560. D'autres configurations d'alimentation sont possibles, mais les piles alcalines ne sont absolument pas recommandées pour les températures inférieures à -18° C.

Les schémas (voir **figures 10** à **12**) sont indicatifs. Les nouvelles cartes 32 bits sont flexibles, avec de nombreuses broches supportant les interruptions et avec des interfaces SPI, I²C et UART alternatives.

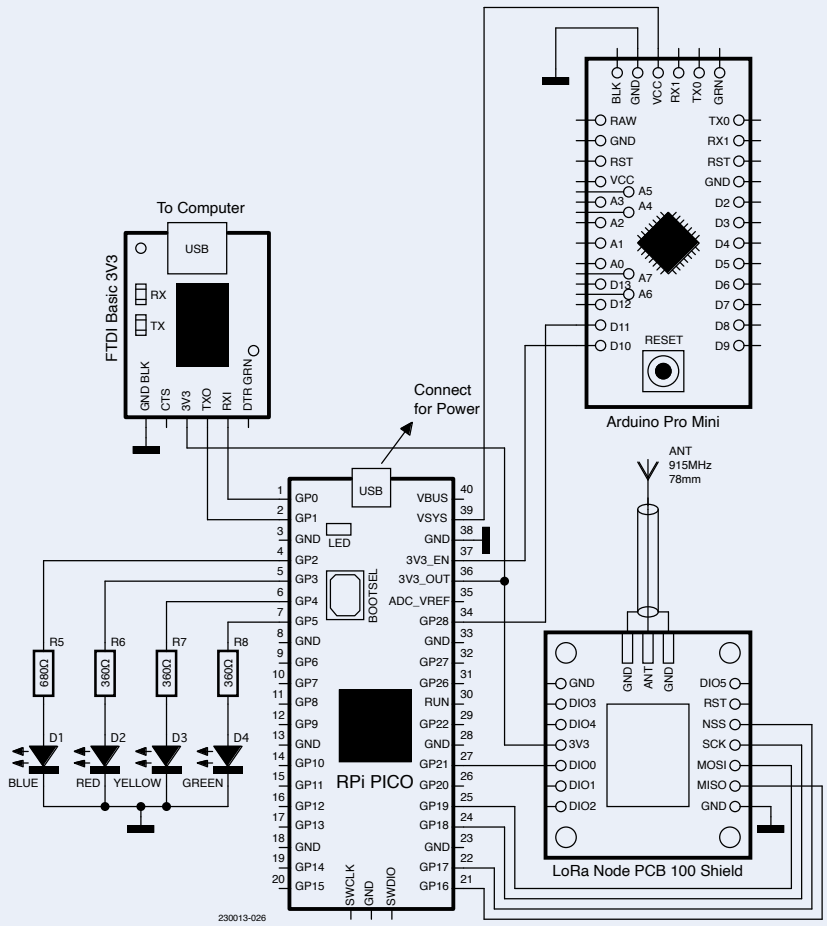


Figure 12. Circuit de la station domotique.

De ce fait, certaines connexions ont été basées sur l'optimisation de la disposition physique de l'ensemble du circuit.

Bonne lecture !

VF : Denis Lafourcade — 230013-04

Des questions, des commentaires ?

Envoyez un courriel à Elektor à l'adresse redaction@elektor.fr.

À propos de l'auteur

Ed Ringel est un médecin semi-retraité spécialiste des soins respiratoires et intensifs. Il profite de la nature du Maine avec sa femme, écrit de la science-fiction, fabrique des objets avec son imprimante 3D et développe des projets électroniques avec des microcontrôleurs.

Produits

- **Kit capteur SparkFun**
  https://elektor.fr/19620
- **Raspberry Pi Pico RP2040**
  https://elektor.fr/19562

LIENS

[1] Tribulations : https://forums.adafruit.com/viewtopic.php?p=927526
[2] Ce projet sur Elektor Labs : https://elektormagazine.fr/labs/low-power-lora-weather-station

Transverter pour la bande des 70 cm

Jan Buiting PE1CSI (Elektor)

En 1981, Elektor a impressionné la communauté des radioamateurs avec un transverter (convertisseur pour émission/réception) soigneusement conçu pour la bande des 70 cm, qui était alors une partie plutôt vide du spectre radioélectrique utilisée par de véritables expérimentateurs pour communiquer dans l'espace sans téléphone portable, vous imaginez !

Prototype du transverter 70 cm construit par Gerrit, PA0HKD et Ed, PE1CJP pour Elektor Labs en 1981. Un émetteur-récepteur Icom IC211 2-m tous modes commande le transverter.

Le transverter 70 cm (430-440 MHz) présenté ici dans cet article a fait l'objet de deux articles publiés dans Elektor en juin et octobre 1981 [1] [2]. C'est un excellent exemple de projet destiné aux radioamateurs ne souhaitant pas débourser la somme exorbitante qu'était l'achat d'un appareil du commerce à l'époque. Ces mêmes passionnés de radio souhaitaient la BLU (Bande Latérale Unique) sur le 70 cm de la même manière qu'ils avaient pu profiter de ce mode en ondes courtes ainsi que sur la bande des 2 m (144 à 146 MHz) pendant de nombreuses années. En contraste avec la FM, la BLU est un mode linéaire requérant une bonne linéarité de tous les étages de l'émetteur/récepteur jusqu'à l'embase d'antenne même.

Attention : les radioamateurs au travail

Le concepteur du transverter est J. de Winter PE0PJW. Gerrit Dam PA0HKD, l'un des concepteurs d'Elektor, et Ed Warnier (PE1CJP devenu PA1EW depuis) en stage chez nous à l'époque, « elektorisèrent » le projet original de l'auteur. Ils veillèrent à la reproductibilité par le lecteur et au respect des normes légales quand il s'agissait des niveaux des harmoniques et signaux parasites.

Ed se souvienne fort bien des tracas que représenta la mise sur circuit imprimé au standard Elektor non seulement en raison des parasites produits par la section de l'exciteur 288 MHz mais aussi de l'inexpérience des dessinateurs de platine de l'époque en ce qui concernait les spécificités des signaux à 400 MHz, habitués qu'ils étaient au continu, de l'audio, des microcontrôleurs et autres alimentations. Ils firent finalement d'une pierre deux coups en gravant des lignes microstrip à même le cuivre de la platine.

Fort et clair

Au début des années 80, la bande des 70 cm était particulièrement intéressante, n'étant pas uniquement l'endroit de rencontre d'amateurs ayant fabriqué leur propre équipement à 100 % (y compris la télévision amateur - ATV) mais aussi des épris de la communication par satellite qui permettait des QSO intercontinentaux

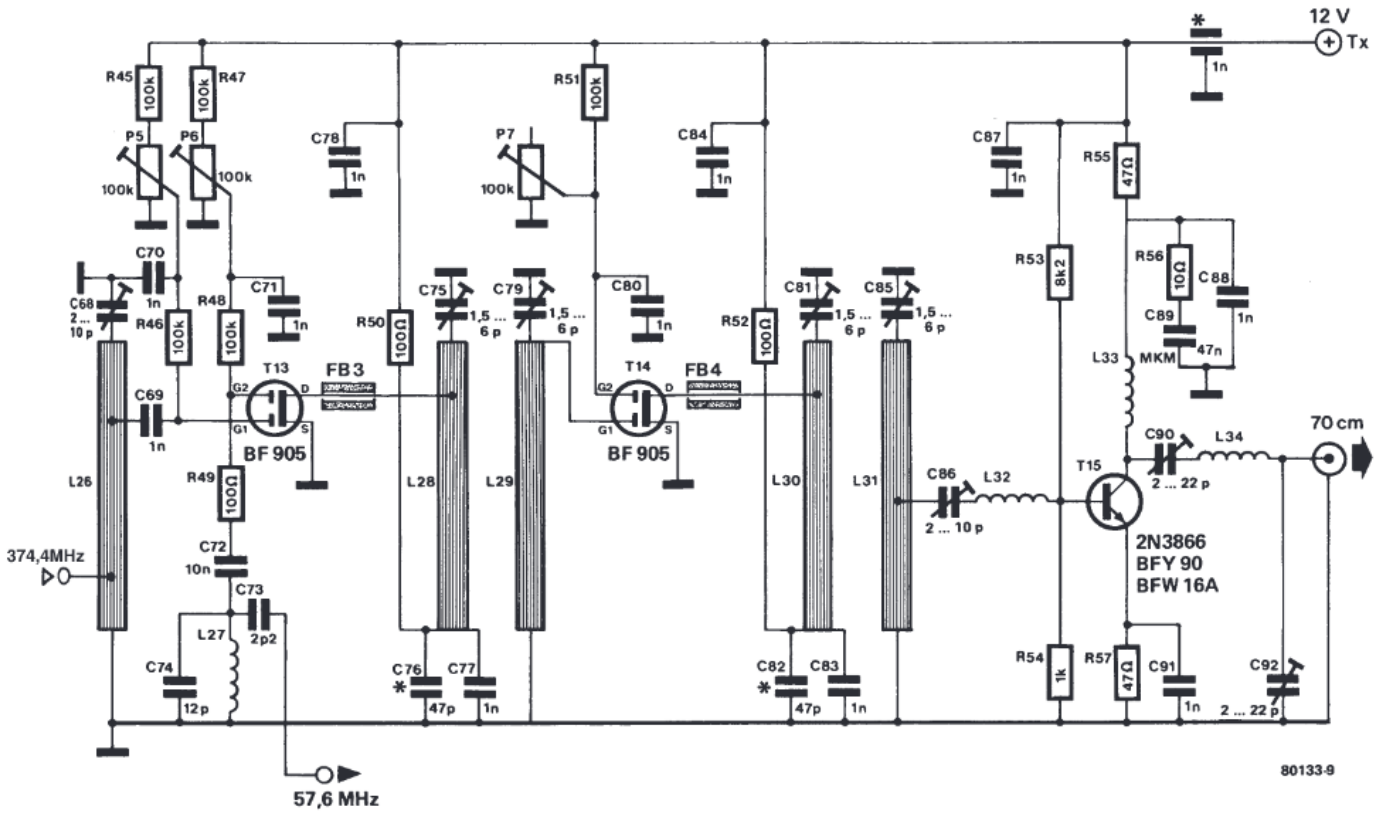
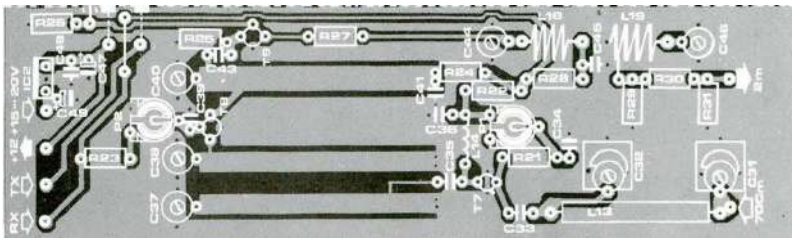


Schéma du circuit de la section UHF du transverter. En 1981, il a fallu que les dessinateurs d'Elektor créent un nouveau symbole pour ces « lignes microstrip » et qu'on ajoute un terme au dictionnaire technique des rédacteurs.

en CW (Onde Entretenue Pure) et BLU, travaillant tous à des puissances d'émission relativement faibles (mais avec des antennes hautement directionnelles).

Le transverter 70 cm d'Elektor a été très apprécié par les radioamateurs, car il est apparu au bon moment et a été adapté parfaitement à leurs besoins. Le projet a été brillamment conçu et parfaitement documenté dans deux articles du magazine Elektor, suivis d'une publication UFB (*ultra-fine business*).

220214-04



Une partie du circuit imprimé du transverter, reproduite à partir du numéro d'octobre 1981. Ces zones rectangulaires entre les lignes noires épaisses ne sont pas de courts-circuits, mais des ligness « microstrip » accordés, fonctionnant à environ 430 MHz. Il ne s'agit pas de quelque chose que l'on peut faire glisser, déposer et tracer automatiquement !

Des questions, des commentaires ?

Envoyez un courriel à l'auteur (jan.buiting@elektor.com).

LIENS

[1] J. de Winter, « transverter 70 cm (1) », Elektor 10/1981 : https://www.elektormagazine.fr/magazine/elektor-198110/51639
[2] J. de Winter, « transverter 70 cm(2) », Elektor 11/1981 : https://www.elektormagazine.fr/magazine/elektor-198111/51653

Climate Calling Engineers

Move Fast and Fix Things

By Priscilla Haring-Kuipers

The sixth synthesis report by the Intergovernmental Panel on Climate Change (IPCC) is clear: "Human activities, principally through emissions of greenhouse gases, have unequivocally caused global warming." What are we going to do about it?

The sixth synthesis report by the Intergovernmental Panel on Climate Change (IPCC) is the collective scientific wisdom on climate change and how to fix it. [1] They inform the UN. The main message of their latest report has hope: "There are multiple, feasible and effective options to reduce greenhouse gas emissions and adapt to human-caused climate change, and they are available now," [2] but currently, we are not applying the technical solutions we have with enough vigour, scale or speed.

Calling all engineers! There is a lot to do, so let me give you some highlights:

Current Climate

We are now at 1.1°C global warming and will likely reach 1.5°C in the early 2030s and shoot up to 3.5°C this century if we don't change drastically. "There is a rapidly closing window of opportunity to secure a liveable and sustainable future for all." [1] We have already caused a lot of damage across ecosystems. More than climate scientists estimated earlier. We have lost many species, nearly 50% of coastal wetlands, and we are impacting ecosystems in ways that are not reversible. Cities have become hotter and the air we breathe more polluted.

Our supply chain has already been impacted by the more frequent occurring extreme weather, making factories freeze or catch fire. Water is fast becoming a contested resource, and factories should look into either recycling or using seawater.

We have not done nothing. Agreements made at Kyoto and Paris have helped. Social movements have accelerated climate action. We can still save ourselves with climate resilient development based on science, indigenous knowledge and local context. High-tech and low-tech solutions working together.

"Individuals with high socio-economic status contribute disproportionately to emissions, and have the highest potential for emissions reductions, e.g., as citizens, investors, consumers, role models, and professionals." [1] That means us. What we do and what we demand of our governance makes a big difference. What you choose to work on as an engineer will either contribute to a liveable world or to further heating up the place. When we support developing regions with our technological development, they can leapfrog to low-emissions solutions with us.

Stop That

If we are ever to stay under 2°C of global warming, a lot of fossil fuels are going to have to stay in the ground. Today, new fossil fuel developments are still being funded, and the fossil fuel industry receives more money in private investments, public subsidies and tax breaks than developments tackling climate adaptation and mitigation. [1] Simply ending fossil fuel subsidies would lower greenhouse gas emissions with 10% by 2030, while improving public revenue that could be redirected to our necessary transition. If your work or your pension funds are connected to the fossil fuel industry, you might want to start looking for a way to decouple before the well is shut down. Our electronics are heavy on petrochemicals and will be looking to shift to bio-based alternatives. Opportunities abound.

By Priscilla Haring-Kuipers, made with DALL-E: Electronics engineer soldering a product to achieve a future save from climate change.

Carbon pricing such as carbon taxes or emission trading have led to some low-cost emission reduction measures but have not been very successful to promote the higher-cost measures that are necessary to shift an industry. We need more. Luckily, climate laws are increasing, and they are helping to fight climate change causes and effects. Climate-related litigation is growing and has already had an effect on the "outcome and ambition of climate governance." [1] It is likely that climate law will grow in the near future, both internationally and on regional levels. The WEEE regulations and the Supply Chain Act are early versions of what is coming. Your efforts and your company can be ahead of the curve, riding the green wave, or you can be dragged along by legislation, but everyone is coming eventually.

Technology for the Win

"If all technically available options were used, global emissions could be at least halved by 2030, at manageable costs." [1] We need tons of engineers to roll out, scale up, improve and adapt to local circumstances many of the already available and proven solutions.

In the last decade, the cost of solar energy has dropped by 85%, wind energy by 55% and lithium-ion batteries by 85%. Meanwhile, deployment has increased over tenfold for solar and over a hundredfold for electric vehicles. In some areas and industries, keeping the old is becoming more expensive than changing to the new. Work on whatever you can to push, develop and spread this development.

Green energy will not only curb our emission, but the economic benefit in air quality alone would offset the cost of the transition. Co-development of energy efficiency and renewable energy will create a happy feedback loop of improvement. Work on big renewables and small-scale nets, smart-grids, transmission and capacity is very much needed.

Cities are critical in this transition. We can build or retrofit to match our new low-emission lifestyles and make space for cycling and walking, teleworking and electric public transport. More plants and water in cities would help cool during heatwaves, process heavy rainfall slower and keep moist during droughts while benefiting the health and well-being of all who live there. Engineers should work on building materials and practices, sustainable urban planning and maintenance, digital communities and smart transport solutions. Many cities have already announced a net-zero emissions target. My city started a green jobs market for all the technical roles we are going to need to develop, install and maintain this bright new future. Your city might have a similar initiative.

Carbon capture is most reliably done by reforestation, improved forest management, soil carbon sequestration, peatland restoration and coastal blue carbon management. Protecting high-carbon ecosystems would have an immediate impact. Globally, we need to protect 30-50% of our land and water to maintain a resilient biodiversity. Throw your skill set behind any project that supports conservation and restoration.

Your time is now.

230265-01

No Geo-Engineering

Blocking the sun with solar shields or sulfur is a no-go. Short-term and local cooling effects are likely, but the amount of green house gases would still grow and the acidification of our oceans would continue. We don't know enough about the effects on the targeted region nor our global ecology. Once it is up and running, turning it off could cause "rapid climate change." The risks are too great, and the reward is too uncertain.

WEB LINKS

[1] IPCC, "AR6 Synthesis Report: Climate Change 2023," 2023: https://www.ipcc.ch/report/ar6/syr/

[2] IPCC, "Urgent climate action can secure a liveable future for all," March 20, 2023: https://www.ipcc.ch/2023/03/20/press-release-ar6-synthesis-report/

elektor store

www.elektor.fr

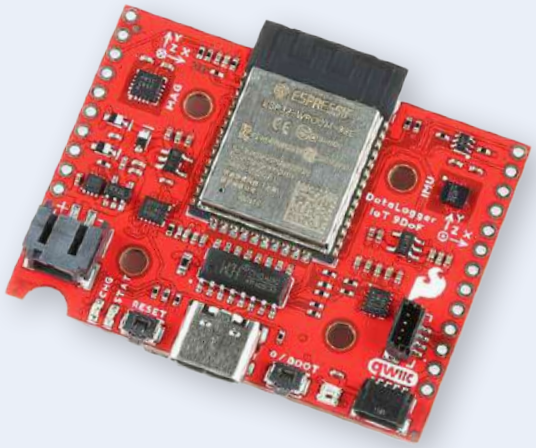
e-choppe Elektor

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.fr.

SparkFun DataLogger IoT (9DoF)

Le SparkFun DataLogger IoT (9DoF) est un enregistreur de données préprogrammé qui enregistre automatiquement les données des capteurs IMU, GPS, ainsi que divers capteurs de pression, d'humidité et de distance. Tout cela sans écrire une seule ligne de code ! Le DataLogger détecte, configure et enregistre automatiquement les capteurs Qwiic. Il a été spécialement conçu pour les utilisateurs qui ont simplement besoin de capturer une grande quantité de données dans un fichier CSV ou JSON et de revenir à leur projet principal.



Prix : 94,95 €

Prix (membres) : 85,46 €

www.elektor.fr/20487

Ensemble d'alimentation numérique Miniware MDP-XP (MDP-M01 + MDP-P906)

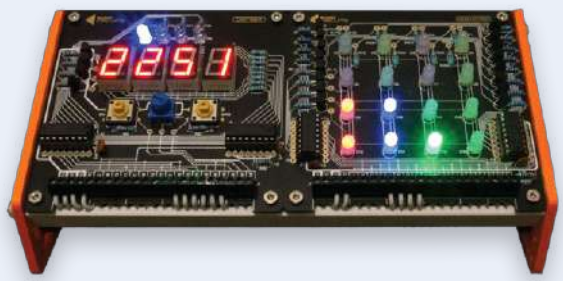
Le MDP (Mini Digital Power System) est un système d'alimentation linéaire programmable en courant continu basé sur une conception modulaire, capable de connecter différents modules pour les utiliser selon les besoins. Le MDP-XP se compose d'un module de contrôle d'affichage (MDP-M01) et d'un module d'alimentation numérique (MDP-P906).

Prix : 269,00 €

Prix (membres) : 242,10 €

www.elektor.fr/20458

Short Circuits: The 4-Pack (plateforme compatible Arduino)

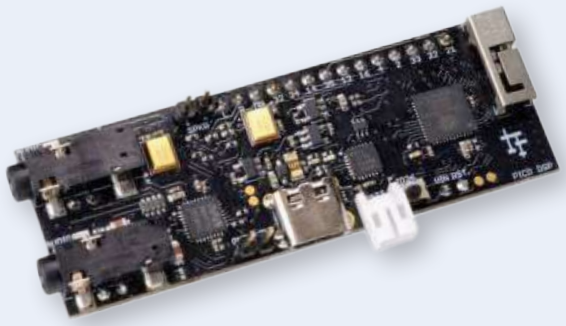


Prix : 99,95 €

Prix (membres) : 89,96 €

www.elektor.fr/20474

Carte de développement PÚCA DSP ESP32

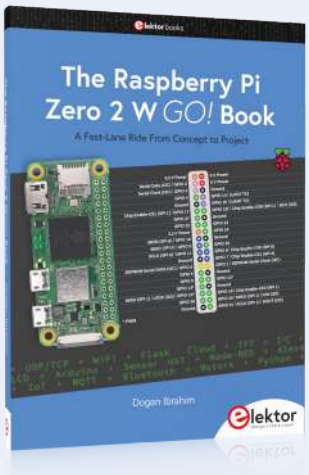


Prix : 69,95 €

Prix (membres) : 62,96 €

www.elektor.fr/20504

The Raspberry Pi Zero 2 W *GO!* Book



Prix : 34,95 €

Prix (membres) : 31,46 €

www.elektor.fr/20445

The Elektor Power Supply Collection (clé USB)

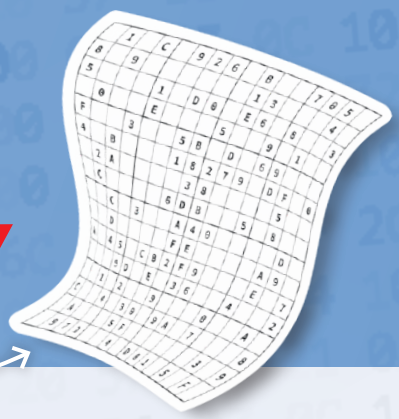
Prix : ~~49,95 €~~

Prix spécial : 34,95 €

www.elektor.fr/20451

hexadoku

casse-tête pour elektorniciens



La dernière page de votre magazine propose toujours une grille de sudoku particulière. Éteignez le fer à souder, oubliez les écrans, rangez les pinces et les fils. Faites travailler votre matière grise d'une autre façon : attrapez un crayon pour remplir la grille d'hexadoku. N'oubliez pas d'envoyer votre réponse pour tenter de gagner un bon d'achat.

Une grille hexadoku est composée de chiffres du système hexadécimal, de 0 à F. Remplissez le diagramme de 16 x 16 cases de telle façon que **tous** les chiffres hexadécimaux de 0 à F (0 à 9 et A à F) n'apparaissent **qu'une seule et unique fois** dans chaque rangée, colonne et carré de 4 x 4 cases (délimités par un filet gras). Certains chiffres, déjà placés dans la grille, en définissent la situation de départ.

Pour participer, inutile de nous envoyer toute la grille, il suffit de nous donner **la série de chiffres** sur fond grisé.

Participez et gagnez !

Nous tirons au sort cinq des réponses internationales correctes reçues dans les délais ; leurs auteurs recevront chacun un bon d'achat Elektor d'une valeur de 50 €.

Où envoyer votre réponse ?

Envoyez votre réponse (les chiffres sur fond grisé) avec vos coordonnées par courriel, avant le **15 août 2023** à l'adresse **hexadoku@elektor.fr**

LES GAGNANTS

La solution de la grille du numéro de mai/juin 2023 est **EBoC8.**

La liste des gagnants est publiée ici : www.elektormagazine.fr/hexadoku

Bravo à tous les participants et félicitations aux gagnants !

D		1	6	0		2	7		A						B
	3							0	1	8	2	7			
5		F					4	3	B		D			0	A
A					3	5					F			4	
3				4			1	9	C			E			
			1		B			6					C	8	
6			5			9	0	A	8				D	F	
0		A		C		D				1		4	5	6	7
	D	6		8	4	F			E			2			
2	E	8		5		0		7	4			6	1		
	F						6			2	B			5	
	5	0	4							9		C		E	8
	6			9			3	C	7		0				
					5	4	2		9						
		5	2		0	E	A			D	8			B	
8		9					F				4				6

E	2	5	A	0	3	7	B	1	F	8	C	6	4	9	D
6	C	D	F	1	5	8	9	3	2	4	0	E	B	A	7
B	0	1	7	6	E	4	F	D	9	5	A	C	8	3	2
8	3	4	9	2	A	C	D	6	7	B	E	F	0	1	5
D	5	F	8	4	B	A	7	C	1	6	3	0	E	2	9
7	4	2	3	E	0	1	C	F	5	9	B	D	6	8	A
9	6	B	0	3	D	F	8	2	A	E	7	5	1	4	C
A	1	C	E	5	2	9	6	4	8	0	D	7	3	B	F
C	7	8	2	A	9	D	3	0	E	1	6	B	F	5	4
F	D	A	1	7	6	0	2	5	B	3	4	8	9	C	E
3	E	9	6	B	8	5	4	A	C	7	F	1	2	D	0
4	B	0	5	C	F	E	1	8	D	2	9	3	A	7	6
0	9	E	B	D	1	2	A	7	3	F	5	4	C	6	8
1	A	7	D	9	4	6	E	B	0	C	8	2	5	F	3
2	8	6	C	F	7	3	5	E	4	A	1	9	D	0	B
5	F	3	4	8	C	B	0	9	6	D	2	A	7	E	1

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.

Rejoignez la communauté Elektor

Devenez membre **maintenant !**

elektor GOLD membership

- accès à l'archive numérique depuis 1978 !
- 8x magazine imprimé Elektor
- 8x magazine numérique (PDF)
- 10 % de remise dans l'e-choppe et des offres exclusives pour les membres
- accès à plus de 5000 fichiers Gerber

Également disponible

abonnement elektor GREEN membership sans papier !

- accès à l'archive numérique d'Elektor
- 10 % de remise dans l'e-choppe
- 8x magazine Elektor (PDF)
- accès à plus de 5000 fichiers Gerber

www.elektormagazine.fr/membres

elektor design > share > earn