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 500

créer > partager > vendre

MARS & AVRIL 2023
ELEKTORMAGAZINE.FR

basé sur ESP32

Cloc 2.0

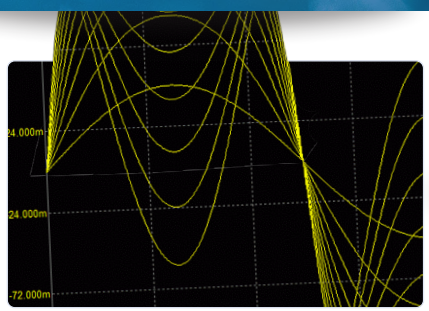
Le réveil-matin de vos rêves

FOCUS SUR
Embarqué et IA

Quel avenir pour l'IA et les systèmes embarqués ?
Outils, plateformes et remplacement des rédacteurs

PIO du RP2040 en pratique
Expérimentation des E/S programmables du RP2040

Sortie vidéo sur les microcontrôleurs
Sortie VGA et DVI
p. 92



Simulation de circuit avec Micro-Cap Premiers pas dans un monde compliqué
p. 48

Poor Man's ChipTweaker
Nous avons des moyens (abordables) de vous faire parler
p. 22

L 19624 - 500 - F: 15,50 € - RD

MICROCHIP

CURIOSITY NANO

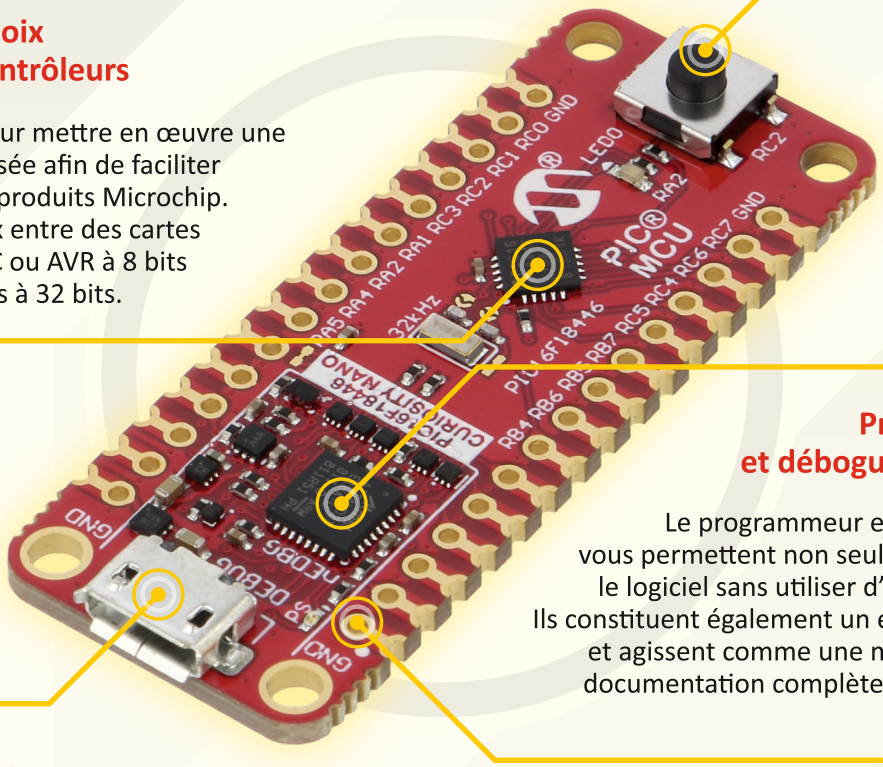
Cartes prototypes coopérant avec des modules d'extension

Composants de base

Les cartes Curiosity Nano ont un bouton, une diode LED, un résonateur à quartz et un régulateur de tension programmable. Grâce à cela, le prototypage de base et l'habitude de travailler avec la plate-forme ne nécessitent aucun accessoire supplémentaire.

Un large choix de microcontrôleurs

La série est conçue pour mettre en œuvre une plate-forme standardisée afin de faciliter la migration entre les produits Microchip. Les clients ont le choix entre des cartes équipées de puces PIC ou AVR à 8 bits et de microcontrôleurs à 32 bits.



Programmeur et débogueur intégrés

Le programmeur et le débogueur intégrés vous permettent non seulement de mettre à jour le logiciel sans utiliser d'outils supplémentaires. Ils constituent également un émulateur de port COM et agissent comme une mémoire de masse où la documentation complète de la carte est stockée.

Interface USB

Pour programmer et exécuter votre code sur la plate-forme MC Nano, connectez simplement la carte au port USB de l'ordinateur et copiez le fichier .hex sur l'appareil.

Broches standardisées

Quel que soit le modèle de microprocesseur choisi ou même son architecture, les broches de base du PCB sont toujours au même endroit. Une standardisation élevée réduit le coût et le temps du processus de prototypage.

46ème année
n° 500 – mars-avril 2023

ISSN 0181-7450
Dépôt légal : avril 2023
CPPAP 1125 T 83713
Directeur de la publication : Donatus Akkermans

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

Pour toutes vos questions :
service@elektor.fr

www.elektor.fr | www.elektormagazine.fr

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

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

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

Jens Nickel

rédacteur en chef d'Elektor Magazine

Des temps passionnants

En planifiant ce numéro il y a plusieurs mois, nous ne pensions pas que l'intelligence artificielle (IA) ferait l'objet d'un tel battage médiatique. Bien entendu, le Sésame ici c'est, ChatGPT, un chatbot qui ne se contente pas de répondre à vos questions sur toutes sortes de sujets, mais qui peut également rédiger des articles, des paroles de chansons et bien d'autres choses encore tout avec un simple clic. Tout comme l'un de nos rédacteurs (voir page 64), j'ai aussi testé ChatGPT. Je lui ai posé des questions sur les systèmes en îlots et les systèmes de stockage d'électricité correspondants. J'ai été impressionné par les résultats. Non seulement ChatBot a pu estimer la capacité d'une batterie LiFePo4 pour une petite maison de campagne, mais il m'a également fourni cinq suggestions pour une telle batterie, avec indication du fabricant et du type, à ma demande. En outre, ChatGPT était de plus « conscient » des limites de ses propres prévisions. Le robot m'a prévenu que les calculs de capacité dépendaient de la consommation d'énergie et de la taille du module, et il m'a recommandé de consulter un professionnel avant d'en acheter une.
Mais bien entendu, ChatGPT n'a aucune conscience ni aucun sens des choses qu'il/elle exprime. Le résultat est une sorte de moyenne pondérée de dizaines de millions d'énoncés possibles, qui peuvent tous être trouvés sur Internet sous une forme similaire. Son habileté linguistique est surprenante, mais presque tout a déjà été dit ou entendu de cette manière. Ce « calcul de la moyenne » à partir d'un grand nombre de données fait souvent paraître les réponses de ChatGPT un peu douteuses.
Cependant, nous sommes fascinés chez Elektor. Je peux bien imaginer que le robot pourrait nous être utile, à nous et à vous, à l'avenir – par exemple, dans la recherche d'un article Elektor approprié dans nos archives, car, enfin, 62 ans de connaissances en électronique y sont accumulées. Peut-être pourrait-il également nous aider à rédiger des codes de référence, à créer des nomenclatures et à concevoir des parties de circuits réutilisables ?
Sur le moyen terme au moins, je ne vois pas comme danger que ChatGPT rende nos emplois – ou même votre contribution, chers lecteurs, à notre magazine – inutiles. Jusqu'à présent, une IA ne ressent pas la chair de poule lorsqu'elle découvre un projet ou un appareil intéressant qui pourrait être super utile pour ses propres développements. Une IA n'a pas ses meilleures idées sous la douche, et elle est incapable de m'écrire des courriels fantaisistes et nostalgiques. Ce sont ces émotions qui nous motivent, dans un labo du sous-sol, ou lors de la rédaction d'articles.
Avez-vous une vision différente ? Quels types d'expériences avez-vous déjà menées avec l'IA ? Où cela va-t-il nous mener ? Écrivez-moi à l'adresse redaction@elektor.fr !

Elektor au salon *embedded world*
Du 14 au 16 mars, *embedded world* aura lieu à Nuremberg, un événement incontournable pour tous les fabricants des microcontrôleurs et les experts en logiciels. Comme d'habitude, Elektor sera présent sur le salon. Venez nous rencontrer sur notre stand **4A-646!**

notre équipe

DANS CE NUMÉRO

Elektor n° 500
mars - avril 2023

Rubriques

Articles de fond

Industrie

ma première radio définie par logiciel réalisée en moins de 15 minutes

56

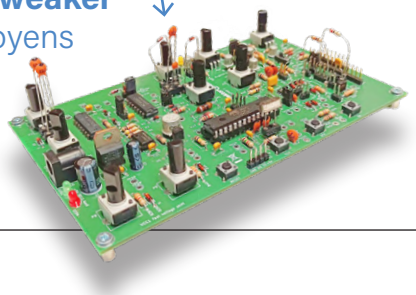
sortie vidéo sur les microcontrôleurs sortie VGA et DVI

92

Projets

Poor Man's ChipTweaker
nous avons des moyens (abordables) de vous faire parler

22



Bientôt dans ces pages

Le numéro de mai – juin 2023
Vous retrouverez dans le prochain magazine Elektor l'habituel mélange stimulant de réalisations originales, de circuits, d'articles de fond, de sujets nouveaux, de trucs et d'astuces pour les électroniciens. Le thème de ce numéro sera « Test et mesures ».

Quelques-uns des points forts :

- enregistreur d'énergie
- sonde d'oscilloscope pour RF
- télécommande universelle adaptative avec ESP32
- l'API Wi-Fi d'Android
- BL808 : RISC-V pour l'IA, la vidéo et bien plus encore
- eCO_2 Telegram Bot
- signaux analogiques et microcontrôleurs : notions de base
- Super Servo Tester
- programmation d'applications IdO contrôlées par la voix

et bien d'autres choses encore !

Le numéro de mai – juin 2023 du magazine Elektor sera publié aux alentours du 11 mai 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.

Cloc 2.0 : le réveil-matin de vos rêves

Yves Bourdon (France)

Avec le récepteur radio, l'horloge est probablement l'un des thèmes favoris des *elektorniciens* amateurs. Je m'attends à ce que tout lecteur de cet article ait construit au moins une horloge. Personnellement, j'en ai construit beaucoup. La plupart faisaient aussi réveil. Je sais donc bien ce que j'en attends. Mon réveil idéal n'étant pas commercialisé, je l'ai construit moi-même. Voici donc le *Cloc 2.0*.

Réalisée en 1971, ma première horloge utilisait des tubes *Nixie* récupérés sur un ordinateur *IBM* mis au rebut. Le circuit imprimé accueillait un nombre considérable de composants. La précision était d'environ dix secondes par jour. Par la suite, j'ai conçu et réalisé de nombreuses horloges dont les plus célèbres d'Elektor.

Mon horloge Nixie faisait aussi réveil, fonction que j'ai beaucoup développée au fil des années en simplifiant au mieux l'interface homme-machine. Animé par un PIC32, mon dernier réveil était doté de deux écrans (**figure 1**). Un module *ESP8266* exploitait un serveur de temps en ligne via *NTP* pour régler l'heure automatiquement. Le PIC32 « pressait » les boutons d'une télécommande ***authentique*** pour allumer la radio et diffuser ma station préférée ou de la musique. Un encodeur rotatif et deux poussoirs servaient à régler le réveil. Décrit ci-dessous, le projet Cloc 2.0 est basé sur celui-ci.

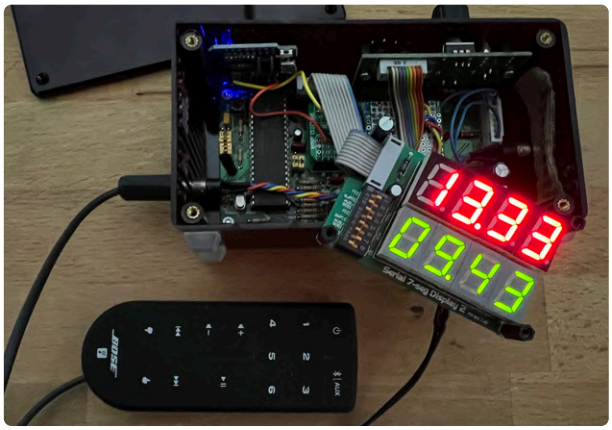


Figure 1. Cloc 1.0, la version précédant Cloc 2.0. Basé sur un PIC32, elle utilisait une authentique télécommande comme sortie IR.

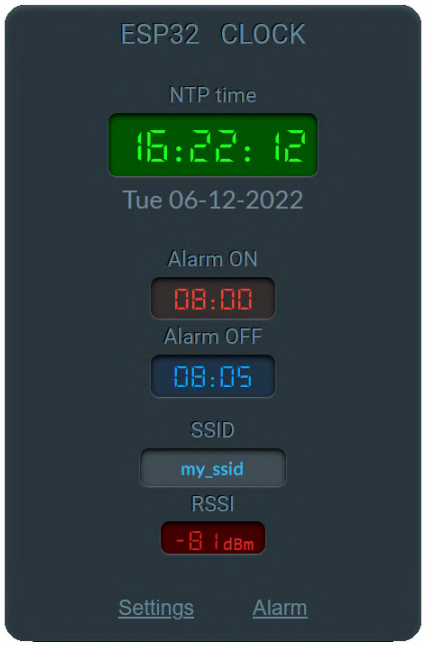


Figure 2. Outre l'heure et la date en cours, la page principale affiche, une vue d'ensemble des paramètres importants.

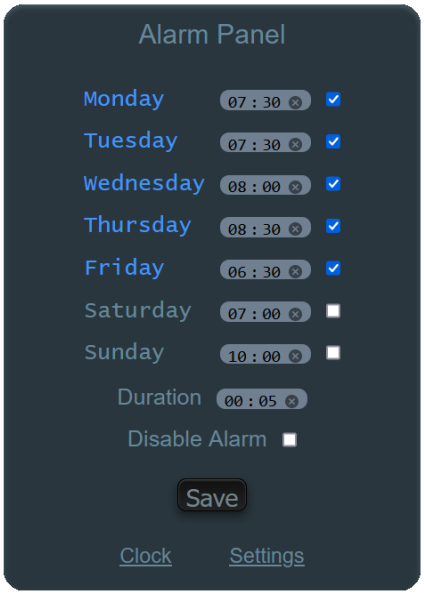


Figure 3. La page Alarme permet de définir l'heure de réveil pour chaque jour de la semaine.

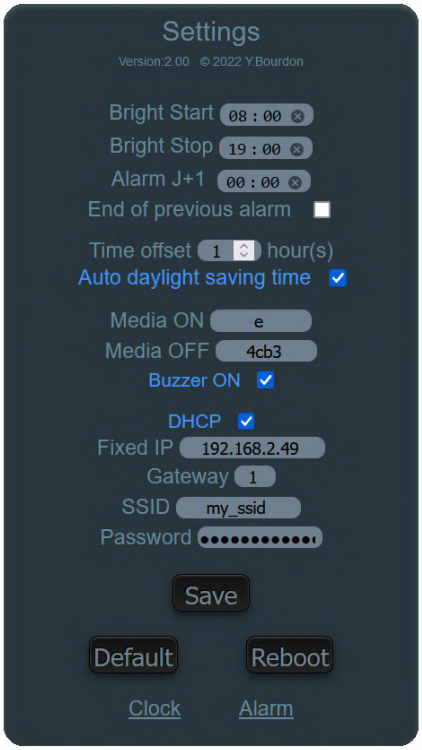


Figure 4. Tous les paramètres configurables par l'utilisateur sont sur la page Paramètres.

Tout pour séduire : beauté et simplicité

Je voulais un réveil facile à construire, sans composants CMS et s'intégrant le mieux possible dans un boîtier standard du commerce. Je voulais aussi garder la touche rétro des afficheurs 7 segments dont la luminosité variable autorise une très bonne lisibilité la nuit, sans gêner quand on aime dormir dans le noir.

L'interface Web

Cloc se connecte à un réseau wifi, il peut donc accéder à un serveur de temps existant. Il se connecte avec une adresse IP fixe ou dynamique (*DHCP*). Une fois connecté, l'accès au serveur web permet de savoir l'heure et de régler les alarmes et paramètres du réveil. L'interface graphique *HTML* optimisée est compatible avec la plupart des navigateurs et des appareils portables. L'interface comprend trois pages : Principale, Alarme et Paramètres.

Page principale

Cette page (**figure 2**) affiche l'heure et la date en cours, ainsi que l'heure de début et de fin de l'alarme à venir. Elle affiche aussi le SSID du réseau wifi auquel il est connecté et le signal *RSSI* reçu. La couleur de ce dernier (vert, orange ou rouge) symbolise sa valeur. La page principale est rafraîchie à 1 Hz environ.

Page d'alarme

La page d'alarme (**figure 3**) permet de régler les alarmes pour chaque jour de la semaine. Décocher un jour de la semaine désactive l'alarme correspondante. On peut également désactiver toutes les alarmes d'un seul clic.

La durée de l'alarme est également réglable. Un début d'alarme lance la commande `Media ON` et l'envoie à la *LED IR*. Une fin d'alarme lance la commande `Media OFF` et l'envoie à la LED IR. L'alarme peut en outre faire retentir le buzzer (v. la page Paramètres).

Sur modification d'un paramètre d'alarme, un bref signal sonore est émis. Un paramètre modifié est sauvegardé dans l'EEPROM.

Page de configuration

Les paramètres ci-après sont réglables (voir **figure 4**).

- Période d'affichage à haute luminosité (`Bright Start` et `Bright Stop`).
- Le moment auquel s'affiche l'heure de la prochaine alarme (*Alarme J+1*), par ex. le soir quand on va se coucher ou bien tout de suite après la dernière alarme (cocher Fin de l'alarme précédente).
- Heure d'été, automatique ou manuelle.
- Code IR `Media ON` et `Media OFF` pour contrôler un autre appareil (par ex. une radio).
- Activer/désactiver le buzzer.
- L'identification du réseau wifi utilisant soit le DHCP, soit une adresse IP fixe.

Boutons-poussoirs

Cloc a deux boutons-poussoirs d'interaction avec l'utilisateur : S1 (désactivation) et S2 (arrêt). En fonctionnement normal, S1 active/désactive l'alarme (allume/éteint l'affichage du bas). L'interface web dispose aussi de cette fonction. En appuyant sur S1 à la mise sous tension de Cloc, on restaure les valeurs par défaut définies dans le programme (pensez à les adapter à vos propres besoins avant de programmer l'horloge) C'est à faire à la toute première mise sous tension de l'horloge.

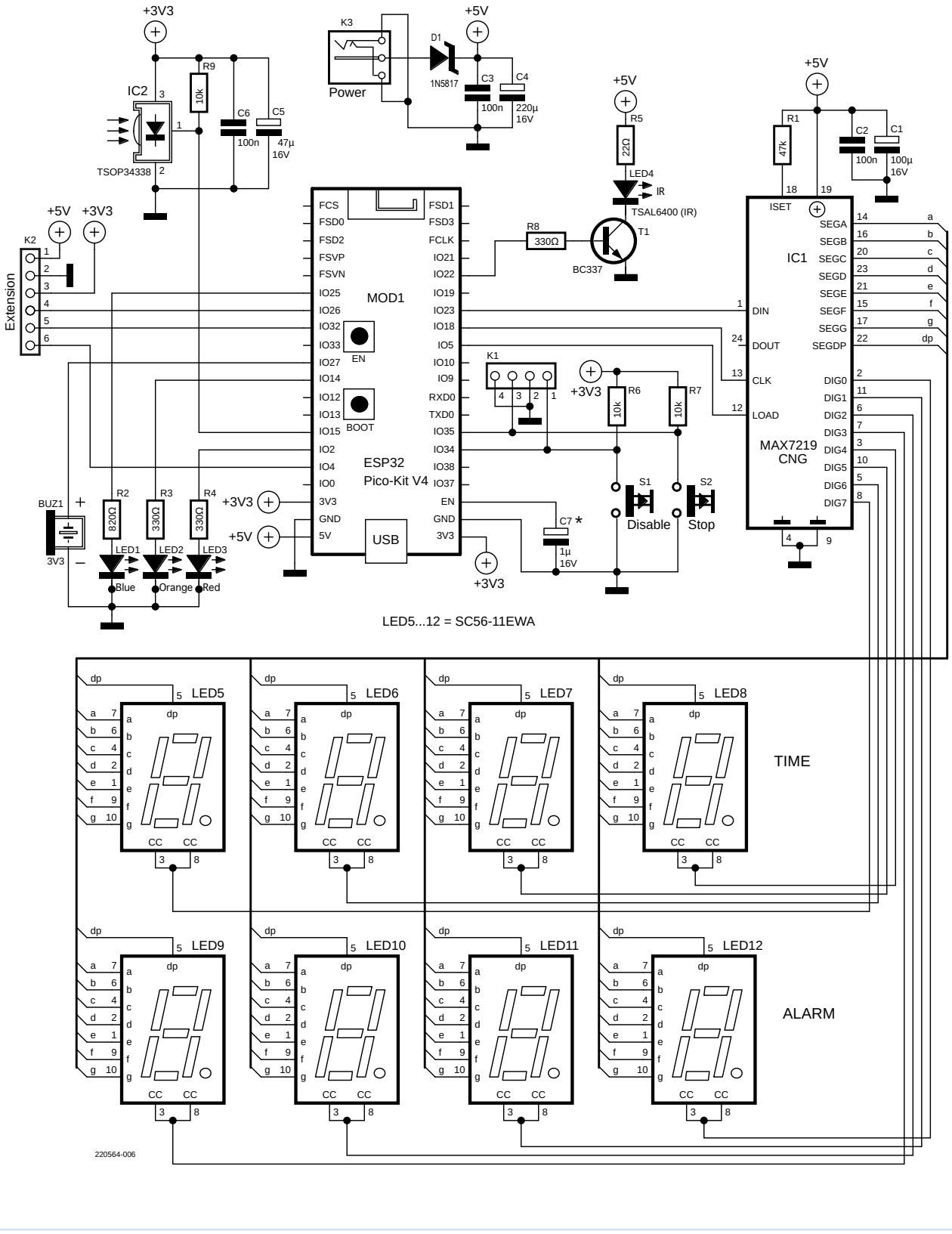


Figure 5. L'ESP32-PICO-KIT fournit la connectivité Internet et l'heure ; le MAX7219 contrôle huit afficheurs 7 segments répartis en deux rangées. Notez l'interface IR autour d'IC2 (entrée) et de LED4 (sortie).

Si une alarme a été déclenchée en mode normal, un appui sur S2 l'arrête immédiatement et l'affichage du bas cesse de clignoter. Un appui sur S2 quand aucune alarme ne retentit, envoie le code IR de mise en marche/arrêt à l'appareil de votre choix. L'horloge se comporte alors comme une télécommande.

Un appui sur S2 pendant la mise sous tension, fait passer l'horloge en mode point d'accès (AP) à l'adresse 192.168.4.1. Vous pouvez vous y connecter via tout appareil (téléphone portable, tablette ou ordinateur) pour accéder à l'interface web de configuration des paramètres de l'horloge.

Description du circuit

La **figure 5** donne le schéma du Cloc 2.0. Les deux afficheurs 7 segments et leur commande IC1 en occupent près des deux tiers. Pour piloter les huit chiffres, quatre par affichage (heure et alarme), j'ai choisi le MAX7219. Il exploite un bus SPI à 2 fils (le DIN n'est pas utilisé) et se programme facilement (bibliothèques open-source). La résistance R1 règle la luminosité max. de l'affichage (plus R1 est élevée, moins l'affichage brille). Notez que les afficheurs à haute luminosité sont vraiment lumineux, peut-être trop pour un réveil.

Pour Cloc 2.0, j'ai choisi le module *ESP32-PICO-KIT* qui offre de nombreux ports d'E/S et qui a les dimensions parfaites pour ce projet. Le condensateur C7 n'est utile que si le module ESP32 reste en mode *reset* après la mise sous tension. Des modules ESP32 (anciens ?) souffriraient de ce problème que je n'ai jamais observé. Si vous décidez de monter C7, couchez-le orienté vers l'opposé de R6 et R7.

Le buzzer d'alarme BUZ1 (sans oscillateur interne) est piloté en MLI par le port IO27 de l'ESP32.

Les ports IO34/35 de l'ESP32 sont reliés aux boutons-poussoirs S1 et S2, et au connecteur K1. Dans une réalisation type, S1 et S2 sont fixés sur le dessus de l'horloge plutôt que sur le PCB, d'où la présence de K1. Bien que l'ESP32 en fournisse, des résistances de polarisation R6 & R7 de 10 kΩ sont prévues. En effet, ne compter que sur celles de l'ESP32, qui valent plusieurs centaines de kΩ, rendrait un bouton câblé plus sensible au bruit.

Trois LED de débogage (LED1/2/3) donnent l'état des connexions : réseau wifi (bleu), interface du serveur Web (orange) et trafic IR entrant/sortant (rouge).

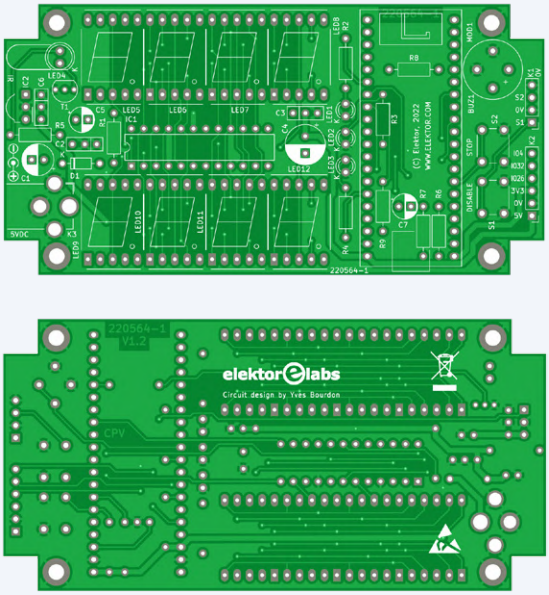
Interface IR

C5 et C6 découplent le récepteur infrarouge IC2 pour nettoyer le signal IR. La LED infrarouge TX (LED4) est pilotée par le transistor T1 et R5 limite à environ 150 mA le courant qui la traverse. La sortie IR est assez puissante pour contrôler tout équipement proche aussi bien que la télécommande d'origine de celui-ci.

Alimentation électrique

L'alimentation externe doit fournir au moins 500 mA sous 5 V_{DC}. La diode D1 protège le circuit contre une inversion de polarité.

LISTE DES COMPOSANTS



Résistances

R1 = 47 kΩ
R2 = 820 Ω
R3, R4, R8 = 330 Ω
R5 = 22 Ω
R6, R7, R9 = 10 kΩ

Condensateurs

C1 = 100 µF, 16 V
C2, C3, C6 = 100 nF
C4 = 220 µF, 16 V
C5 = 47 µF, 16 V
C7 = 1 µF, 16 V

Semi-conducteurs

D1 = 1N5817
IC1 = MAX7219
IC2 = TSOP34338
LED1 = bleu, 3 mm
LED2 = orange, 3 mm
LED3 = rouge, 3 mm
LED4 = TSAL6400, IR LED, 5 mm
LED5-12 = Chiffre à 7 segments 0.56", CC
(par exemple SC56-11EWA)
T1 = BC337

Divers

BUZ1 = Buzzer piézoélectrique sans oscillateur, 12 mm, 3,3 V
K1 = Tête de broche 1x4
K2 = Tête de broche 1x6
K3 = Jack baril
MOD1 = ESP32-PICO-KIT
S1, S2 = Interrupteur tactile 6x6 mm
PCB 220465-1
Boîtier Hammond 1591CTDR (rouge translucide)
Module d'horloge temps réel (RTC) DS3231

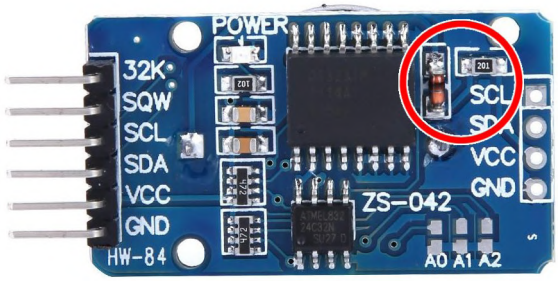


Figure 6. Si le module RTC DS3231 (option) est utilisé avec une pile bouton CR2032 non rechargeable, retirez la résistance ou la diode (ou les deux) dans le cercle rouge pour éviter d'endommager la pile.

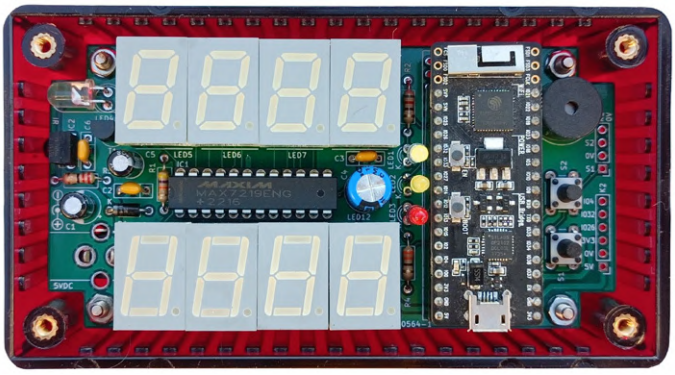


Figure 7. La carte assemblée est à l'aise dans le boîtier. Notez qu'elle peut aussi se monter sur le couvercle pour masquer les vis du boîtier. Le connecteur d'alimentation peut être monté des deux côtés de la carte.

Deux condensateurs électrolytiques (C1 & C4) amortissent les pics de courant. En parallèle des condensateurs de 100 nF (C2 & C3) atténuent le bruit HF.

Extensions

Le connecteur K2 est prévu pour une extension, par ex. à l'aide d'un module d'horloge en temps réel (v. ci-après). Trois ports d'E/S restent libres pour cela : IO4/26/32.

Option horloge en temps réel (RTC)

On peut connecter un module RTC *DS3231* au bus d'extension K2. Il prend le relais si le serveur NTP est inaccessible pour une raison quelconque (par ex. wifi en panne, problème chez le fournisseur d'accès à Internet). Le module RTC est détecté et configuré automatiquement par le logiciel et remis à l'heure une fois par jour (à 12h00). Le DS3231 est très performant et compense la dérive du quartz grâce à un capteur de température intégré.

Le module RTC DS3231 communique en I²C et n'est relié à K2 que par quatre fils : GND, 3,3 V, SDA et SCL. Si un module RTC est détecté, le port IO26 devient le signal SDA et le port IO32 prend en charge le signal SCL du bus I²C.

Danger !

Attention, le RTC DS3231 comporte un risque. En effet, ce module intègre un circuit de charge de la batterie, mais souvent, il n'est livré qu'avec une batterie CR2032 non rechargeable installée qui risque alors d'exploser. Il existe deux solutions à ce problème. La plus simple (et la plus chère) est de remplacer la pile par une pile rechargeable LIR2032. La 2e solution consiste à retirer la diode ou la résistance de 200 Ω (ou les deux) près de la broche SCL du connecteur à 4 broches (**figure 6**).

Détails mécaniques

Le 1591CTDR de Hammond fait un bon boîtier pour Cloc. Il mesure 120 x 63 x 38 mm. Il est rouge translucide (IR) et parfait pour les afficheurs 7 segments de l'heure du jour et de l'heure d'alarme (rouge et/ou orange). La transmission et la réception des signaux IR bénéficient aussi d'un tel filtre optique. Notez que grâce à ce filtrage, la lumière pauvre en contenu rouge (par ex. la LED bleue) est à peine visible.

À ce propos, l'ESP32-PICO-KIT possède une LED d'alimentation rouge très brillante que vous voudrez probablement cacher ou supprimer.

Le circuit imprimé (PCB) a été conçu en tenant compte de cette enceinte. Le projet KiCad6 peut être téléchargé depuis [1]. Le boîtier loge facilement la carte montée sur quatre goujons de 16 mm (**figure 7**). Pour fixer la carte, il suffit de percer le fond de quatre trous de 3,2 mm ; il faut aussi un trou de 8 mm sur le côté pour le connecteur d'alimentation, et deux trous de 12 mm sur le dessus pour les deux boutons poussoirs.

Le module RTC en option peut être connecté à K2 sous le PCB principal (côté composant vers le haut) en utilisant l'embase vide à 4 broches. On peut préférer retirer l'embase coudée à 6 broches pour éviter les courts-circuits. On peut aussi le coller sur le PCB principal avec du ruban adhésif double face (ce que je l'ai fait).

L'environnement de développement logiciel

Pour développer le logiciel de Cloc 2.0, j'ai utilisé l'EDI Arduino après ajout du paquet des cartes ESP32. Pour éviter les erreurs de compilation ou de gestion mémoire, il faut sélectionner la carte ESP32 PICO-D4 à une fréquence CPU de 240 MHz. Pour le *Partition Scheme* choisir *Default* (c.-à-d. 4 Mo avec SPIFFS). En sélectionnant le bon port série, compilation et téléchargement du programme se déroulent sans problème.

Avant compilation, il faut installer les bibliothèques externes suivantes dans l'IDE. J'ai indiqué dans le code source où les obtenir. La plupart peuvent aussi s'obtenir par le gestionnaire de bibliothèques de l'IDE.

- AsyncElegantOTA
- AsyncTCP
- ESPAsyncWebServer
- IRremote (v. ligne 126 de `Alarm_Clock.ino`).
- LedControl
- RTClib

Important : Il faut modifier `LedControl.h` en commentant la ligne 30 (`#include <avr/pgmspace.h>`). Voir aussi le commentaire dans le code source.

Le téléchargement [1] contient un dossier nommé *libraries* qui comprend toutes les bibliothèques utilisées, dont le `LedControl.h` pré-modifié. Copiez ces bibliothèques dans le sous-dossier *libraries* du dossier *Arduino sketchbook* (chemin indiqué dans la fenêtre *Preferences* de l'IDE).

Pour savoir comment ajouter l'option *ESP32 Sketch Data Upload* à l'IDE, voir [2]. C'est nécessaire pour charger l'interface web de l'horloge en mémoire de données du module ESP32. N'oubliez pas de charger ces données, sinon le serveur web restera muet.

Sans-fil (OTA)

Pour mettre le micrologiciel du réveil à jour sans fil via un navigateur, ciblez-y la page `IP_of_cloc/update` où `IP_of_cloc` représente l'adresse IP du réveil. Choisir alors un exécutable situé sur votre ordinateur et le télécharger dans le réveil. Une explication détaillée de l'interface OTA est disponible [3].

Au cœur du programme

Le programme est assez bien documenté (souvent en français), et après avoir lu ce qui suit, vous devriez être à l'aise. Surveillez le port série de l'ESP32 car beaucoup d'informations de débogage y sont envoyées.

Après l'inclusion des bibliothèques, définissez les valeurs par défaut utilisées soit au 1er démarrage, en maintenant l'appui sur le bouton (S1), soit depuis la page Settings en cliquant sur Defaults. L'ordre des constantes suit l'ordre des paramètres de la page Paramètres (lignes 40 à 65 d'*Alarm_Clock.ino*). Vérifiez en particulier l'adresse IP fixe et la passerelle, ainsi que les identifiants wifi. Pour la page Alarme (lignes 67 à 84), l'organisation est la même.

À la mise sous tension, on configure d'abord les fonctions. Il est facile de suivre l'ordre d'initialisation du matériel Cloc. La fonction `initRTC` vérifie si un module RTC DS3231 est connecté. Si le bouton S1 est enfoncé à ce moment-là, on charge les valeurs par défaut.

Au premier démarrage de l'horloge ou si Stop (S2) est pressé pendant la mise sous tension, on passe en mode Point d'accès (AP). Dans ce cas, comme l'horloge l'indique, un navigateur peut se connecter à son interface web : 192.168.4.1.

Après récupération de l'adresse IP (DHCP ou fixe) et s'être connecté au réseau wifi, on cherche l'heure NTP. Si elle est trouvée et qu'une RTC est présente, ils sont synchronisés. L'adresse IP de l'horloge s'affiche brièvement avant de passer à l'affichage de l'heure. L'écran commence à clignoter « HH:HH » pour l'heure et « AA:AA » pour l'alarme, si le serveur NTP n'a pas pu être trouvé et il faut redémarrer l'appareil (si aucun RTC n'est présent, sinon il affiche l'heure RTC).

L'initialisation s'achève par le démarrage du serveur web et du service `AsyncOTA` qui permet de faire une mise à jour à distance du µlogiciel et des SPIFFS.

La fonction `loop` exécute le reste du programme en une boucle sans fin. Si la RTC est présente (`flagRTC=1`), elle lit l'heure RTC. Sans RTC, le programme essaie l'heure NTP (`NTPflag=1`). Une fois l'horloge connectée, le programme lit l'heure RTC, tandis que `strTime` contient l'heure NTP valide en format « hh:mm:ss », sinon elle contient « 0 » .

Ensuite, l'état des boutons est lu (`checkButtons`) et on vérifie si l'horloge doit émettre un bip et si un code IR a été reçu. Le serveur étant asynchrone, il faut utiliser des drapeaux pour signaler au programme principal les changements de paramètres et les déclenchements d'alarme.

À chaque seconde l'écran est rafraîchi (`dispTime`) et les alarmes sont lues (`checkAlarm`). Le programme vérifie aussi s'il est en période de haute luminosité ou non (`checkBrightness`) et s'il doit synchroniser le RTC avec l'heure NTP (`checkSyncRTC`). Cela se produit une fois par jour à 12 h (`syncRTC=12* 60`). La fonction `checkTimer` vérifie si une alarme doit être activée. Pour cela, l'heure courante et l'heure de l'alarme sont converties en minutes (`computeMinutes`) avant comparaison (`testRange`).

Comme dit ci-avant, le serveur est asynchrone. Chaque fois que la page Paramètres ou Alarme est rafraîchie, les données à afficher (`ParametresProcessor` et `TimerProcessor`) sont renvoyées. Les balises `%value%` de la page HTML doivent donc être remplacées par les valeurs réelles.

Spécifications

- Basé sur un processeur ESP32.
- Deux rangées de 4 afficheurs à 7 segments : l'une indique l'heure qu'il est, l'autre l'heure d'alarme.
- Réglage automatique de l'heure par connexion à un serveur de temps en ligne.
- Heure d'alarme propre à chaque jour de la semaine.
- Sortie d'alarme : buzzer et code infrarouge, par ex. pour une radio, chaîne hi-fi ou TV.
- Deux boutons-poussoirs d'interaction avec l'appareil.
- Serveur web intégré pour la configuration à distance via Wi-Fi.
- Tous paramètres enregistrés en EEPROM
- Le mode Over-the-Air (OTA) permet la mise à jour sans fil du micrologiciel.
- Module optionnel d'horloge temps réel (RTC) basé sur DS3231 secouru par batterie.

Pour la page principale, c'est un peu différent car il faut afficher presque en temps réel l'heure du jour, la date et les heures de début et de fin de l'alarme. Une fonction écrite en JavaScript interroge environ chaque seconde la boucle principale pour récupérer les valeurs à afficher (`onDataUpdate`). À chaque fois, la LED2 clignote brièvement pour indiquer l'accès au serveur (`flashRqstBeacon`). Pour vérifier que la page affiche bien les valeurs en temps réel, appuyez sur S1 (*Alarm Disable*). L'heure de l'alarme est affichée ou désactivée simultanément sur l'afficheur 7 segments et sur la page HTML.

Bibliothèque *IRremote*

La bibliothèque `IRremote` fonctionne parfaitement, mais nécessite quelques règles d'intégration. Ne déplacez pas le code d'initialisation, sauf si vous savez ce que vous faites.

Pour le réveil en musique au lieu d'un bip, j'utilise une radio Bose. J'ai donc limité la réception et le décodage aux télécommandes Bose (voir ligne 125 de `Alarm_Clock.ino`). Vous pouvez supprimer cette ligne pour décoder d'autres télécommandes. Je recommande de limiter la réception infrarouge à la marque de votre radio ou de votre chaîne hi-fi. Pour les autres options possibles, voir [4].

De même, la transmission IR est limitée au format Bose. Il est donc fort probable que vous deviez adapter le code à la marque de votre équipement Hi-Fi. Remplacez la ligne `IrSender.sendBoseWave` des fonctions startMedia et `stopMedia` de `Alarm_Clock.ino` par une fonction correspondant à votre marque (voir [4]).

Pour découvrir les codes IR à utiliser avec votre système (audio), reliez le port USB du module ESP32 à un ordinateur utilisant un moniteur série, par exemple le Serial Monitor intégré à l'IDE Arduino. Dirigez votre télécommande vers le récepteur IR de l'horloge et pressez une touche ; le code correspondant en hexadécimal s'affiche sur l'ordinateur. Saisissez les codes en hexadécimal dans les champs *Media ON* et *Media OFF* du panneau Paramètres.

Le clignotement bref de la LED3 signale l'activité IR. Cette LED clignote donc également lorsqu'une alarme est active.

Faites-le vous-même !

Depuis plusieurs mois déjà, j'utilise cette Cloc version 2.0 et je n'ai jamais manqué un réveil à cause d'un dysfonctionnement. La possibilité de définir un heure d'alarme pour chaque jour de la semaine est très pratique.

Je recommande vivement d'ajouter l'option RTC car si le serveur NTP ne répond pas (par ex. si votre modem s'est déconnecté pour une raison quelconque), le réveil perdra l'heure au moment suivant de resynchronisation de l'heure NTP. Il faudra alors arrêter et redémarrer le réveil une fois l'heure NTP à nouveau disponible. Pour une fiabilité accrue, j'ai réalisé un petit serveur NTP qui reçoit l'heure d'un GPS [5]. Ainsi, même en l'absence de connexion Internet, mon réveil se règle automatiquement.

J'espère que vous serez nombreux à réaliser ce réveil. N'hésitez pas à me contacter sur mon courriel privé (yb.electronique@orange.fr) si vous rencontrez des difficultés ou si vous apportez des améliorations à mon réveil préféré !

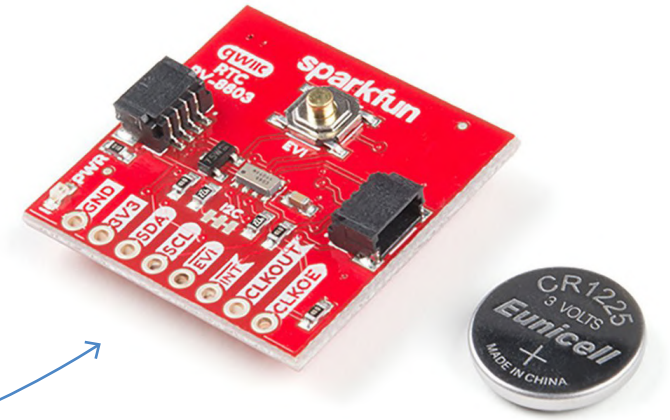
220564-04 — *VF : Yves Georges*

Des questions, des commentaires ?

Envoyez un courriel à l'auteur (yb.electronique@orange.fr) ou contactez Elektor (redaction@elektor.fr).

Produits

- **ESP32-PICO-Kit V4 (SKU 18423)**
  www.elektor.fr/18423
- **Elektor Arduino Electronics Bundle (SKU 19440)**
  www.elektor.fr/19440
- **Module RTC de SparkFun – RV-8803 (Qwiic) (SKU 19646)**
  www.elektor.fr/19646



LIENS

[1] Téléchargements pour Cloc 2.0 (Elektor Labs) : https://www.elektormagazine.fr/labs/cloc-le-reveil-20
[2] Téléchargement de données ESP32 SPIFFS : https://randomnerdtutorials.com/install-esp32-filesystem-uploader-arduino-ide/
[3] Mises à jour OTA (Over-the-Air) de l'ESP32 : https://randomnerdtutorials.com/esp32-ota-over-the-air-arduino/
[4] Bibliothèque IRremote : https://github.com/Arduino-IRremote/Arduino-IRremote
[5] Serveur NTP (Elektor Labs) : https://www.elektormagazine.fr/labs/ntp-server

PIO du RP2040 en pratique

expérimentation des E/S programmables du RP2040

Martin Ossmann (Allemagne)

La RPi Pico est une carte de développement appréciée qui offre une puissance de traitement élevée pour une faible consommation d'énergie. Et elle représente un excellent rapport qualité-prix à seulement 4 € l'unité. Le microcontrôleur RP2040 utilisé sur la carte possède une caractéristique unique (jusqu'à présent) : deux blocs PIO-I/O avec huit machines à états. À l'aide d'exemples concrets, nous examinons de plus près les avantages qu'ils peuvent apporter à une application.

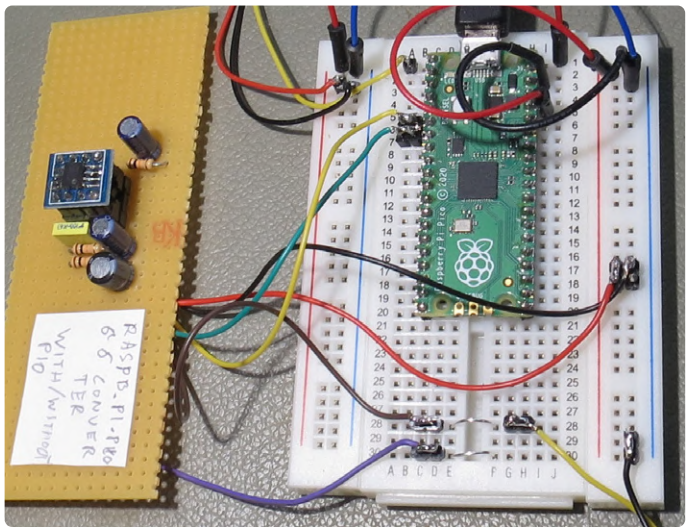


Figure 1. La carte RPi Pico montée sur une platine d'expérimentation et le circuit imprimé Delta_Sigma.

Si vous avez déjà une certaine expérience de la programmation des microcontrôleurs, vous apprécierez sans doute l'utilité d'une unité d'E/S programmable intégrée pour réduire la charge sur les cœurs de traitement principaux. La vraie question est de savoir si cette unité est inutilement compliquée à utiliser et donc n'en vaut pas la peine. Vous aurez certainement une meilleure appréciation de ce qu'elle peut faire une fois que vous aurez travaillé sur les exemples pratiques décrits ici, en utilisant les E/S programmables intégrées au processeur RP2040. La **figure 1** montre ma configuration matérielle expérimentale avec laquelle ces exemples ont été développés.

L'unité PIO

La flexibilité de l'unité d'entrées/sorties programmables vous permet d'utiliser les broches d'E/S pour mettre en œuvre d'autres types de protocoles de communication en plus des standards existants tels que SPI et I²C. Elle se compose de deux blocs PIO, chacun avec quatre processeurs d'E/S. Chaque processeur d'E/S s'intègre à l'architecture de la puce comme le montre la **figure 2**.

Au total, il y a deux PIO dans le RP2040 et chacun contient quatre machines à états (SM) et une mémoire d'instructions de 32 mots pour commander les machines à états. Chaque SM possède deux registres X et Y de 32 bits, un registre à décalage d'entrée de 32 bits (ISR) et un registre à décalage de sortie de 32 bits (OSR). Les noyaux PIO sont chacun connectés à l'unité centrale RP2040 à l'aide de deux FIFO de 32 bits de large et de quatre mots de profondeur. L'unité FIFO TX est

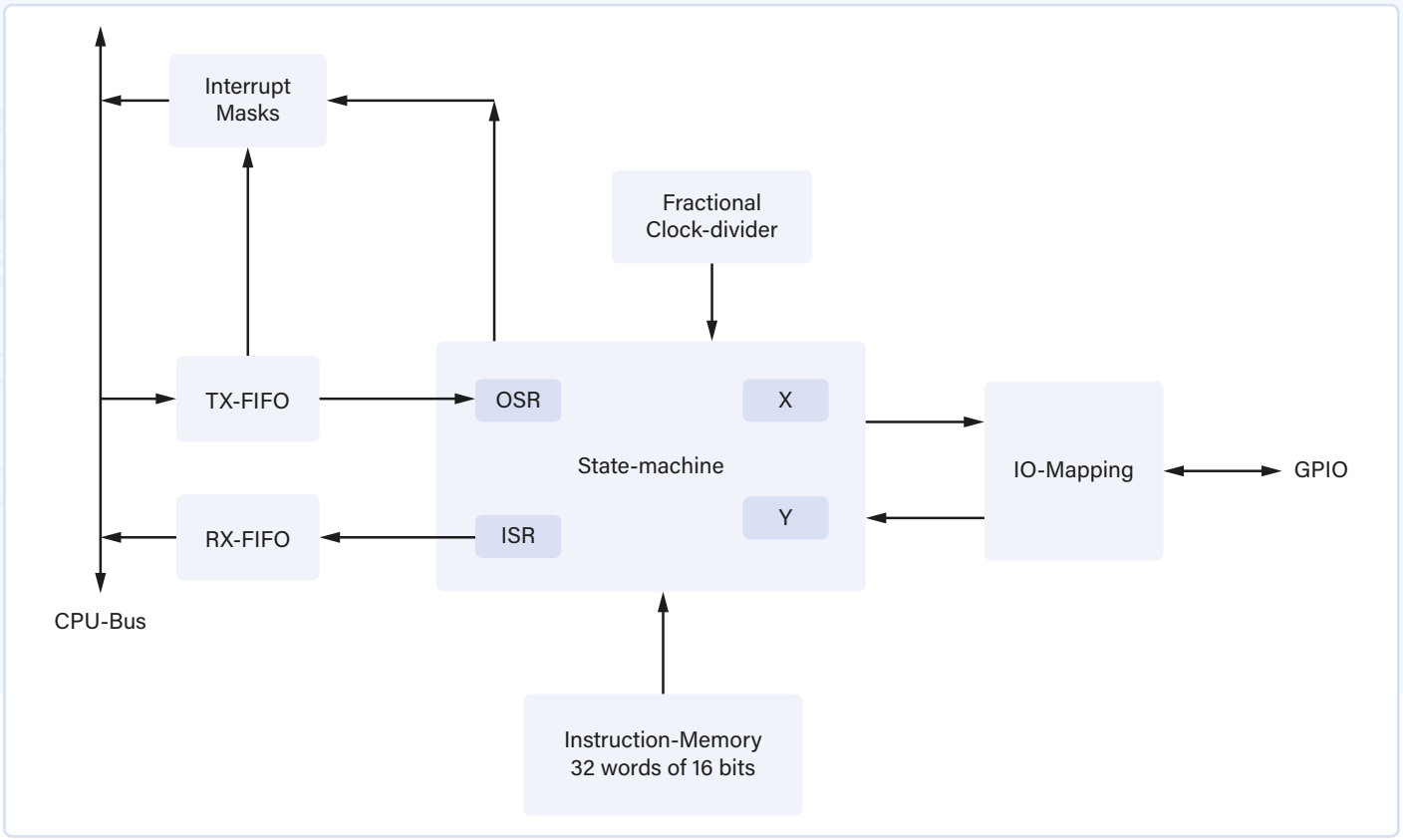


Figure 2. Schéma fonctionnel d'un des deux processeurs d'E/S [4].

utilisée pour l'émission des données et l'unité FIFO RX pour la réception. Si les données ne circulent que dans un sens, les deux FIFO peuvent être configurées pour fonctionner comme une seule FIFO de huit mots de profondeur. Chaque processeur PIO possède un diviseur sur 16 bits entiers plus huit bits fractionnaires pour la génération de l'horloge (voir ci-dessous). Cela permet de régler la fréquence d'horloge de chaque processeur entre 2 et 125 MHz.

Les processeurs PIO ne sont capables de comprendre que neuf instructions. Ces instructions ont une largeur de 16 bits et sont stockées dans une mémoire d'instructions de 32 × 16 bits qui commande les

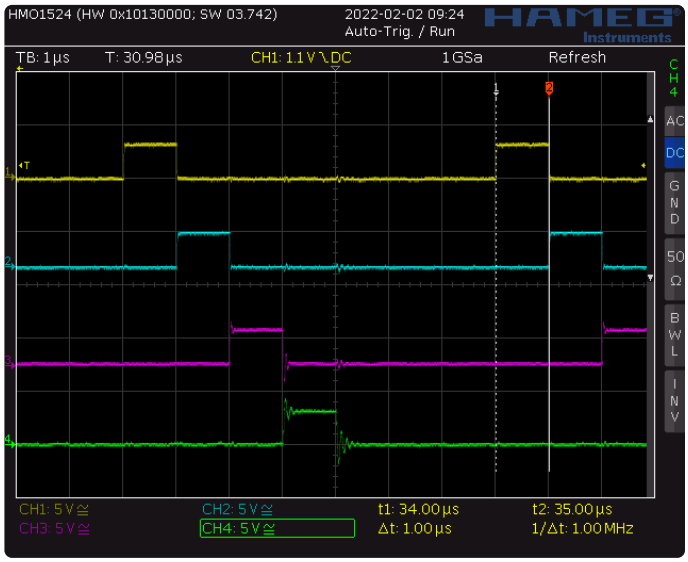


Figure 3. Formes d'onde du premier exemple.

quatre SM de chaque bloc PIO. Les programmes individuels des PIO sont suffisamment courts pour tenir dans cette petite mémoire. Les instructions sont stockées ici par l'unité centrale RP2040. Ces petits programmes sont écrits grâce à un programme assembleur PIO dédié mais assez basique. Chaque instruction est exécutée en un cycle d'horloge.

Un seul processeur PIO ne peut adresser que quelques broches GPIO, et il utilise pour cela des adresses courtes. Le mappage des E/S détermine quelles broches sont alors spécifiquement adressées. Il existe plusieurs constantes programmables qui déterminent pour la catégorie concernée (*in, out, sideset* et *set*) quelle est la première broche adressée. Cela permet de définir les broches d'E/S d'un processeur PIO de manière très souple. Plusieurs processeurs peuvent aussi accéder à la même broche.

Instruments de travail

L'environnement de développement utilisé ici est VS Code. L'installation et l'utilisation de cet outil sont bien décrites dans [1]. Vous y trouverez également des informations sur l'utilisation du générateur de projet utilisé pour créer un projet RPi avec tous les fichiers nécessaires (code source, fichiers de configuration, *Make-Setup*, etc.)

Vous devez également spécifier la sortie via l'USB ou l'UART de la fonction `printf()`. Ici, vous configurez également les bibliothèques que vous devez utiliser. Dans cet article, par exemple, nous utilisons l'outil PIO. Vous pouvez également trouver de l'aide pour l'installation de VS Code dans [2]. Il y a aussi une description détaillée des outils supplémentaires (*Make, Git*, etc.) qui doivent être installés.

Tous les exemples de logiciels peuvent être téléchargés à partir de [3].

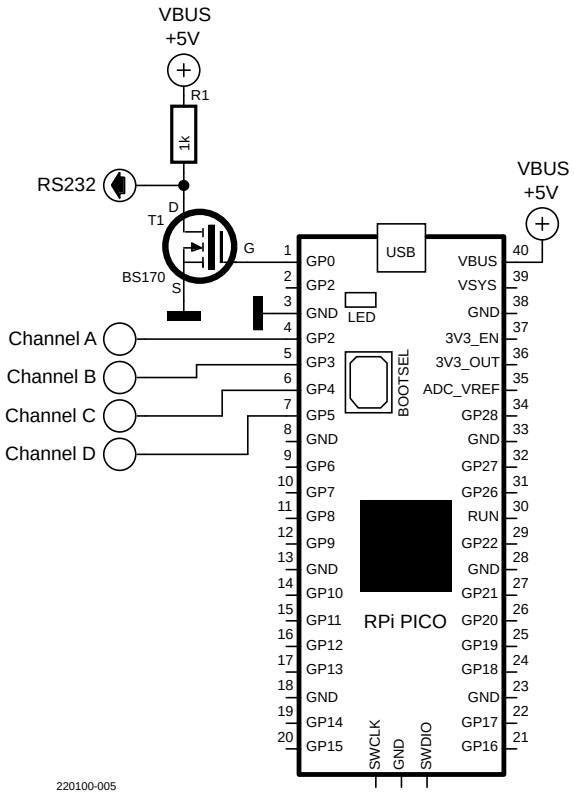


Figure 4. Connexion d'un oscilloscope à quatre canaux au RPi Pico pour visualiser les formes d'onde de sortie du premier exemple.

Commençons

Nous voulons que ce premier programme émette une impulsion de 1 µs sur quatre broches de sortie, l'une après l'autre, comme le montre la **figure 3**. Un oscilloscope à quatre canaux est connecté aux quatre broches de sortie pour visualiser les signaux de sortie résultants. La **figure 4** montre la connexion de l'oscilloscope aux broches d'E/S du RPi Pico.

Le listage suivant contient le premier exemple de programme PIO. Après trois lignes de directives assembleur, la première instruction exécutable se trouve à la ligne 4. Cette ligne et les suivantes génèrent ensuite les quatre impulsions.

Listage 1

```
001 .program pioTest
002 .side_set 2 opt
003 .wrap_target
004   nop           side 0b01 // GP2=1, GP3=0
005   nop           side 0b10 // GP2=0, GP3=1
006   set pins,0b01 side 0b00 // GP2=0, GP3=0,
        GP4=1, GP5=0
007   set pins,0b10       //    GP4=0,      GP5=1
008   set pins,0b00  [2]  //    GP4=0,      GP5=0
009 .wrap
```

Les lignes 4 et 5 contiennent les instructions `nop`, qui, comme vous vous en doutez, ne font rien, mais la ligne 4 est prolongée par le texte `side 0b01`. Cela indique que sur cette ligne, quelque chose d'autre est « fait à côté ». Dans ce cas, la broche *sideset 0* est mise à « 1 » et la broche *sideset 1* est remise à « 0 ». Chaque instruction PIO peut être étendue de cette manière. Cela vous permet d'influer facilement sur beaucoup plus de broches en même temps. Il est important d'identifier quelles sont les broches *sideset* en utilisant des commandes de configuration écrites en C. La définition est faite en C dans le **listage 2** en utilisant la fonction `sm_config_set_sideset_pins()` à la ligne 5. Cela indique que les broches *sideset* commencent à GP2.

Listage 2

```
001 #define set_base 4   // pins start at GP4
002 #define set_count 2  // number of pins is 2
003 sm_config_set_set_pins (&c,set_base,set_count);
004
005 #define side_set_base 2 // side-pins start at GP2
006 sm_config_set_sideset_pins(&c, side_set_base);
007
008 float div = 125.0;
009 sm_config_set_clkdiv(&c, div);
```

L'instruction `nop` de la ligne 4 du **listage 1** génère un front montant via `side` sur la broche *side 1* (= GP3). La ligne 6 contient une instruction `set` qui active les broches 0 et 1 des broches *set*. L'impulsion atteint alors la broche GP4. Les broches *sideset 0* (GP2) et 1 (GP3) sont réinitialisées via la commande `side`. Dans la routine de configuration du **listage 2**, ligne 3, on déclare que les broches set commencent à partir de la broche GP4. L'instruction set peut être utilisée pour charger des registres ou des broches d'E/S avec des valeurs constantes.

Le modificateur `[2]` après l'instruction `set` de la ligne 8 du **listage 1** fait que l'instruction est suivie d'un délai de deux périodes d'horloge. Avec cette spécification utilisant des crochets, chaque instruction peut être retardée de 1 à 31 périodes d'horloge.

Il ne vous reste plus qu'à utiliser une horloge appropriée pour que chaque impulsion soit exactement de 1 µs. Pour ce faire, vous devez diviser l'horloge système de 125 MHz par 125. Avec la fréquence d'horloge résultante de 1 MHz, chaque cycle dure 1 µs comme désiré. Ceci est défini dans la routine de configuration du **listage 2** aux lignes 8 et 9.

Les directives assembleur `.wrap_target` et `wrap` dans le **listage 1** font que les instructions entre elles sont répétées sans fin dans une boucle au moment de l'exécution. Le saut vers le début de la boucle (`.wrap_target`) ne prend pas de temps. Vous pouvez programmer des boucles « sans cycle ajouté ». Dans notre programme, il faut sept cycles d'horloge (cinq instructions + deux cycles de retard) pour parcourir une fois la boucle.

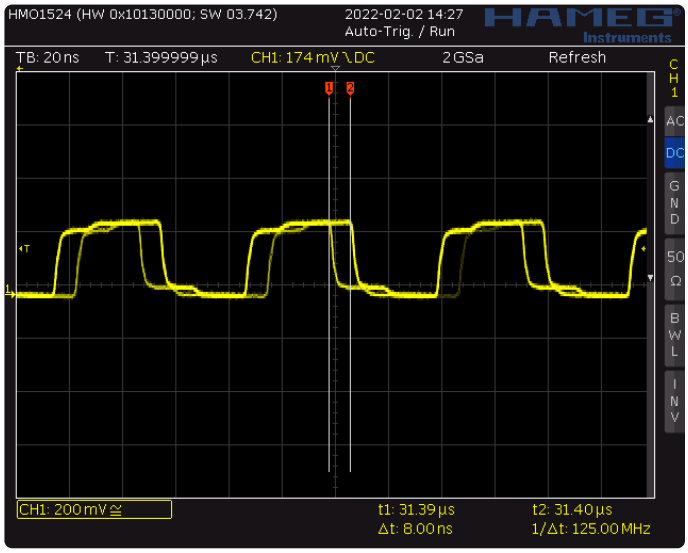


Figure 5. Le signal d'horloge généré présente une gigue.

Génération d'horloge : un diviseur fractionnaire

Comme déjà mentionné, l'horloge du processeur PIO est dérivée de l'horloge système de 125 MHz à l'aide d'un « diviseur fractionnaire ». Ce diviseur de 16 bits peut compter jusqu'à 216 = 65 536 et dispose de huit bits après la virgule. La précision est donc de 1/256. Le « diviseur fractionnaire » est mis en œuvre de la même manière que dans les générateurs de signaux DDS. La fraction derrière la virgule est incrémentée en continu et un front d'horloge est déclenché à chaque report. Le listage suivant démontre le réglage d'une fréquence d'horloge fractionnelle.

Listage 3

```
001 float clockFrequency=28e6;
002 float div=125e6/clockFrequency;
003 sm_config_set_clkdiv(&c,div);
```

L'horloge est élaborée de telle façon que sa période fluctue autour d'une période d'horloge système ce qui produit une gigue dans la forme d'onde. Dans l'exemple, nous voulons une fréquence d'horloge de 28 MHz, mais comme 28 n'est pas divisible par 125, le facteur de division de l'horloge n'est pas un nombre entier mais comporte un certain résidu. Le facteur de division de l'horloge est 125/28 = 4,4642857…

Le **listage 3** montre comment on configure ce diviseur. À titre de démonstration, le **listage 4** ci-dessous montre comment produire un simple signal carré avec le programme PIO.

Listage 4

```
001 .program theProgram
002 .side_set 1 opt
003     nop    side 0
004     nop    side 1
```

L'oscilloscope montre très clairement la gigue de l'horloge dans la forme d'onde de la **figure 5**. Avec des valeurs de diviseur d'horloge plus élevées, la gigue devient proportionnellement plus petite et beaucoup moins importante, de sorte qu'elle peut souvent être négligée. Avec le « diviseur fractionnaire », vous pouvez alors réaliser des vitesses de transmission telles que la typique 115 200 bits/s avec une précision suffisante.

Transmission de données série : un émetteur UART

L'exemple suivant est un peu plus pratique : il s'agit d'envoyer des données en série en utilisant une interface RS232. L'interface est connectée à la broche GP4 afin de laisser libre le GP0 de l'UART intégré à des fins de débogage. Comme l'interface est inversée, on doit générer un signal inversé (état de repos = haut). Lorsque le caractère ASCII est envoyé, la forme de l'impulsion sur la broche GP4 ressemble alors à celle de la **figure 6**.

En plus du signal TXD, l'UART doit émettre un 1 sur une ligne *Busy* tout en envoyant les huit bits de données. GP5 (broche *sideset 0*) peut servir de ligne *Busy*. Le programme PIO associé figure dans le **listage 5** ci-dessous.

Listage 5

```
001 .program theProgram
002 .side_set 1 opt
003     pull              [1]
004     set pins,0 side 1 [2] // startBit=0
005     set x,7               // loop for 7+1 times
006 bit:
007     out pins,1        [2] // LSB first
008     jmp x-- bit
009     set pins,1 side 0 [2] // stopBit=1
```

L'instruction `pull` de la ligne 3 prend un mot de 32 bits dans la FIFO TX et le stocke dans l'OSR. Si aucun mot n'est présent dans la FIFO TX, l'instruction attend simplement un mot et l'insère directement dans l'OSR. La broche de sortie est mise à 0 pour servir de bit de départ par l'instruction `set` de la ligne 4.

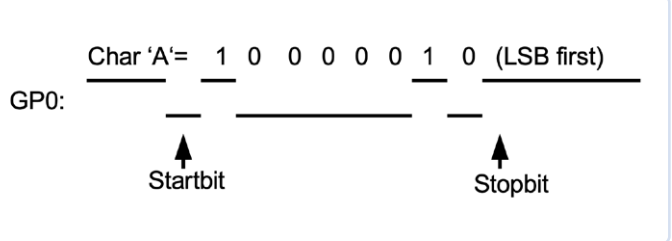


Figure 6. Séquence d'impulsions pour envoyer le caractère « A ».

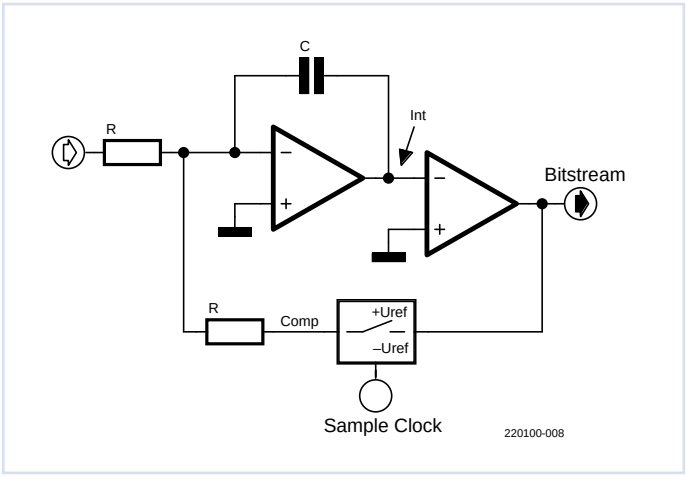


Figure 7. Diagramme fonctionnel d'un convertisseur analogique-numérique Delta-Sigma.

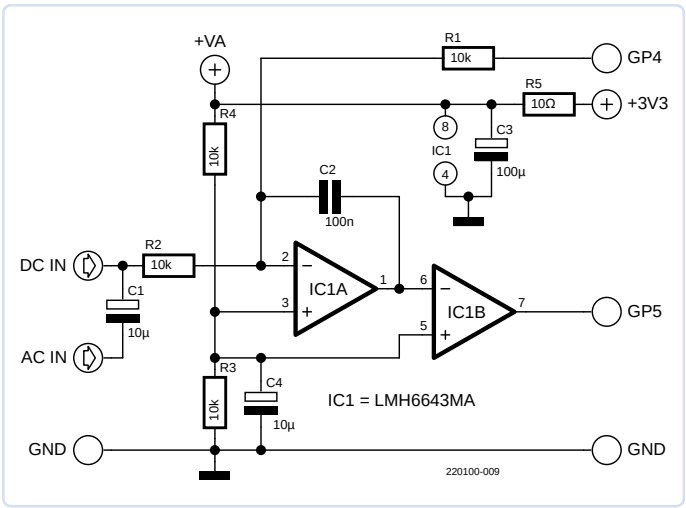


Figure 8. Circuit pratique CA/N Delta-Sigma selon la figure 7.

Les huit bits les moins significatifs du mot dans l'OSR sont maintenant émis dans une boucle. Pour cela, il est nécessaire d'initialiser le compteur de boucle. Le registre X sert ici de compteur de boucle. L'instruction `set` de la ligne 5 l'initialise avec la valeur 7.

Le label `bit` de la ligne 6 marque le début de la boucle. La première et seule instruction de la boucle est l'instruction `out` de la ligne 7. Elle émet le LSB de l'OSR sur la broche out 0 et décale les bits de l'OSR d'une position vers la droite. En plus des broches *set* et *sideset*, les broches out constituent le troisième mode d'utilisation des broches GPIO. La broche GP4 est également définie comme une broche out.

Le saut conditionnel de la ligne 8 avec l'option `x--` décrémente le registre X de 1 et saute ensuite au label `bit` (début de la boucle) lorsque `x` est supérieur à 0. De cette façon, la boucle est parcourue huit fois. La dernière instruction de la routine de transmission TX est l'instruction `set` à la ligne 9, qui crée le bit d'arrêt en mettant la broche *set* 0 à 1.

Avec les deux extensions `side`, le signal *Busy* est généré par la broche *sideset 0* (GP5).

Les extensions de délai entre crochets font que chaque bit a exactement quatre périodes d'horloge. Le débit en bauds est donc égal au débit de l'horloge divisé par quatre. En fixant `clkDiv` à 125 000 000/(4 x 115 200), on obtient bien la vitesse de transmission souhaitée de 115 200 bit/s.

On utilise la fonction `pio_sm_put_blocking(...)` pour transférer des caractères avec un programme C vers le PIO-UART, comme le montre la fonction `PIOprint()` dans le listage suivant. Ici, elle transmet un caractère à la FIFO TX. Si l'unité est pleine, elle se bloque et attend qu'il y ait de la place.

Listage 6

```
001 void PIOprint(const char *s) {
002    while (*s)
003       pio_sm_put_blocking(pio, sm, *s++);
004    }
```

La routine TX-UART montre bien qu'il est possible d'obtenir une fonctionnalité intéressante avec seulement six instructions, les extensions d'instructions `side` et `[delay]` jouant un rôle important.

Un CA/N Delta-Sigma

Dans la section suivante, on va implémenter un convertisseur A/D delta-sigma en utilisant un PIO. On peut voir le principe d'un tel convertisseur à la **figure 7**. On essaie de compenser la tension d'entrée avec une source de tension commutable. Un intégrateur à l'entrée intègre le signal d'erreur suivi d'un comparateur qui détermine le signe de la tension de compensation pour le prochain intervalle de temps. Une séquence de bits est créée à la sortie, qui suit la moyenne de la tension d'entrée. La **figure 8** montre un circuit pratique selon ce principe.

Ce circuit se connecte à GP4 et GP5 du RPi Pico. La séquence de bits est déterminée via PIO et huit bits sont stockés ensemble dans un octet dans la FIFO RX. Le microcontrôleur prend ensuite les octets de cette unité et les traite. On peut voir le programme PIO associé dans le listage suivant.

Listage 7

```
001 .program hello
002 Loop1:
003    set   x,7
004 inLoop:
005    in    pins,1
006    mov   osr,~isr
007    out   pins, 1
008    jmp   x-- go1   [20]
009    push  noblock
010    jmp   Loop1     [23]
011 go1:
012    jmp   inLoop    [26]
```

La boucle extérieure portant l'étiquette `Loop1` se répète sans fin. La boucle intérieure effectue toujours huit itérations. Chaque itération commence par l'instruction qui suit l'étiquette `inLoop`. L'instruction `in` de la ligne 5 prend un bit de la broche *in* et l'écrit dans l'ISR. A la ligne 6, l'ISR est copié inversé dans l'OSR. À la ligne 7, le bit le moins significatif est émis sur la broche out, ce qui génère l'étape de compensation suivante dans l'intégrateur.

Le bit lu est le bit de sortie suivant du convertisseur delta-sigma. Huit bits sont collectés dans l'ISR. L'instruction `jmp` de la ligne 8 commande la boucle interne. Lorsqu'elle a terminé 8 boucles, l'instruction `push` de la ligne 9 est exécutée. Cette instruction stocke l'ISR dans la FIFO RX que le MCU peut ensuite décharger. Les extensions de délai entre crochets garantissent que chaque étape delta-sigma dure 50 impulsions d'horloge. Le diviseur d'horloge a été réglé sur 25. La fréquence d'échantillonnage du convertisseur delta-sigma est donc de 125 MHz / (25 × 50) = 100 kHz.

Pour la démonstration, une onde sinusoïdale de 50 Hz a été convertie par le convertisseur delta-sigma. La **figure 9** montre le spectre résultant. La distance entre le signal utile et le bruit est d'environ 60 dB. Ce n'est pas mal du tout pour un circuit aussi simple !

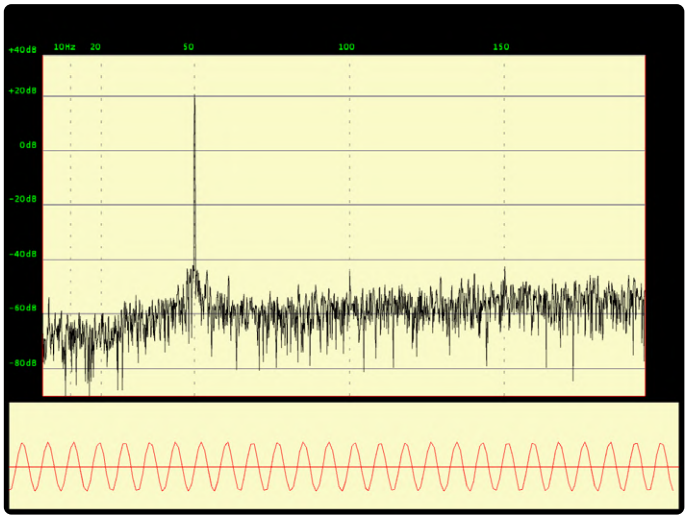


Figure 9. Contenu spectral d'une onde sinusoïdale de 50 Hz après conversion Delta-Sigma.

Construire un générateur de signal

Dans cet exemple, nous construisons un CD/A-R2R simple en utilisant des résistances arrangées en échelle, conformément à la **figure 10**. Nous le connectons aux broches d'E/S du RPi Pico pour produire des signaux de sortie analogiques. Le générateur devra produire un signal de sortie périodique continu. Pour ce faire, les valeurs de données doivent être transférées de manière répétitive d'une table en mémoire vers le convertisseur N/A. Cela fonctionne particulièrement bien en utilisant le DMA (**D**irect **M**emory **A**ccess). Le contrôleur DMA du RPi Pico transfère les données à la FIFO TX. Le **listage 8** ci-dessous montre le programme PIO, composé d'une seule instruction exécutable.

Listage 8

```
001 .program pio_serialiser
002
003 .wrap_target
004     out pins,8 // use autopull
005 .wrap
```

L'instruction `out` amène les huit bits les moins significatifs de l'OSR aux huit broches de sortie GP0 à GP7. L'OSR est alors automatiquement rechargé à partir de la FIFO TX car la fonction *auto-pull* PIO a été activée dans la routine de configuration. Les huit bits suivants sont alors émis avec l'instruction suivante. La boucle de sortie dure donc exactement une impulsion d'horloge PIO. Les données doivent alors passer de la mémoire principale à la FIFO TX le plus rapidement possible. Pour ce faire, deux contrôleurs DMA du RP2040 sont connectés en série en utilisant un DMA de données et un DMA de commande.

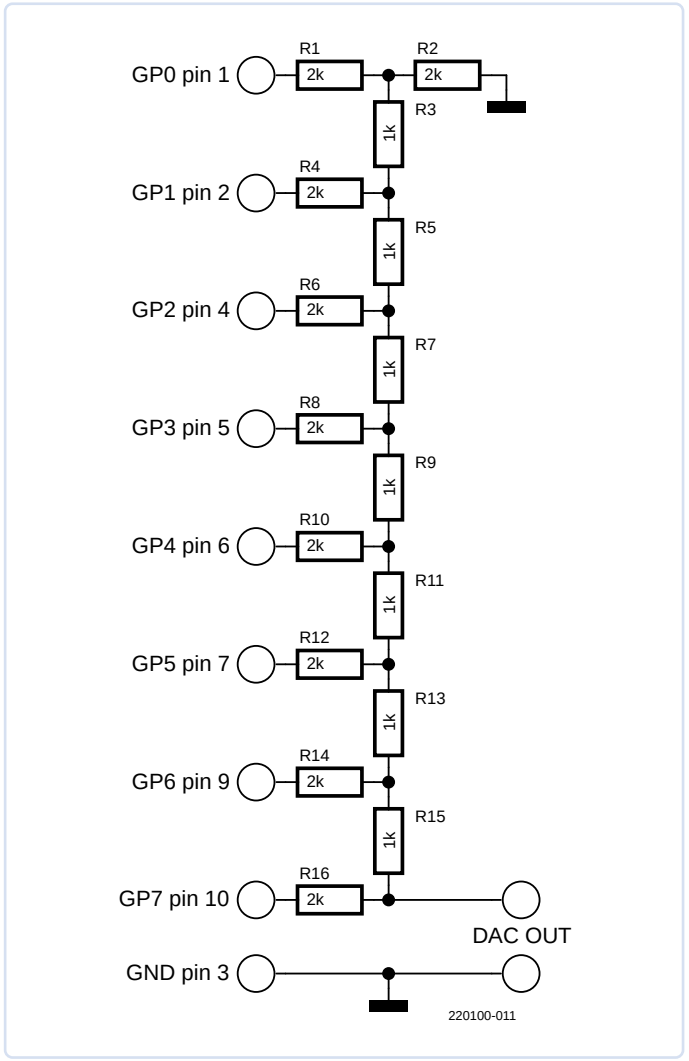


Figure 10. Circuit CD/A-R2R.

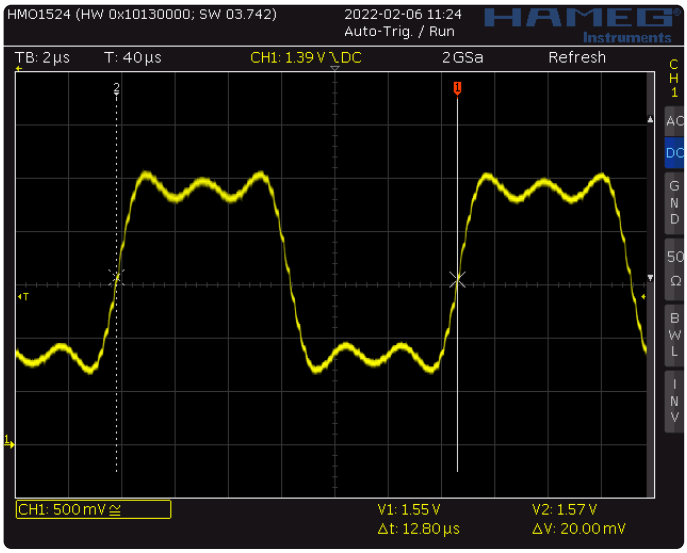


Figure 11. La fréquence de base plus deux harmoniques donnent une onde carrée.

Le contrôleur DMA de commande veille à ce que le contrôleur DMA de données soit redémarré rapidement. Avec cette technique, un nouvel octet peut être émis tous les quatre cycles d'horloge système (125 MHz / 4). Toutefois, le simple convertisseur N/A illustré à la **figure 10** n'est pas adapté à une vitesse de sortie aussi élevée.

Dans la **figure 11**, *clkDiv* a été réglé sur 25. L'horloge du DAC qui en résulte est de 125 MHz / 25 = 5 MHz. Comme une période du signal de sortie dure 64 échantillons, la fréquence générée est calculée comme suit : 5 MHz / 64 = 78,125 kHz. On a utilisé l'approximation de Fourier d'une onde carrée composée d'une onde fondamentale et de deux harmoniques pour produire la forme d'onde de sortie.

Initialisation du registre

L'instruction set peut être utilisée pour initialiser les registres X et Y avec des valeurs fixes. Comme les commandes sont codées sur 16 bits, la commande set ne peut traiter que des constantes comprises entre 0 et 31. Pour initialiser X ou Y avec des constantes plus grandes, vous pouvez utiliser une astuce.

En appelant la routine `pio_sm_exec(pio, sm, instruction)` depuis un programme C, vous interrompez la séquence d'instructions du bloc PIO `pio` et de la machine à état `sm` et insérez la commande `instruction`. On peut utiliser la séquence de fonctions suivante pour initialiser le registre X avec la valeur 500 000.

Listage 9

```
001 pio_sm_put_blocking(pio, sm, 500000);
002 pio_sm_exec(pio, sm,
      pio_encode_pull(false, false));
003 pio_sm_exec(pio, sm,
      pio_encode_mov(pio_x, pio_osr));
```

Tout d'abord, vous placez 500 000 dans la FIFO TX, puis vous exécutez l'instruction `pull` sur le contrôleur PIO, qui amène cette valeur de la FIFO dans l'OSR. Enfin, l'instruction `mov X,OSR` place la valeur dans le registre X comme souhaité. Vous pouvez maintenant lancer le PIO proprement dit. Le programme du **listage 10**, qui fait clignoter une LED, sert d'exemple.

Listage 10

```
001 .program blink
002 .side_set 1 opt
003    mov y,x   side 1
004 yLoop1:
005    jmp y--   yLoop1
006    mov y,x   side 0
007 yLoop2:
008    jmp y--   yLoop2
```

MicroPython

Le RPi Pico est une plateforme bien adaptée au développement de programmes écrits en MicroPython. Il existe une interface Python correspondante pour la programmation PIO. Celle-ci n'est pas programmée à l'aide de l'assembleur PIO, mais en utilisant la syntaxe Python appropriée, qui rappelle fortement un langage de programmation assembleur. Un avantage de l'utilisation de MicroPython est que vous pouvez sauvegarder des projets complets dans un seul fichier. L'exemple de programme associé se trouve dans le **listage MicroPython**.

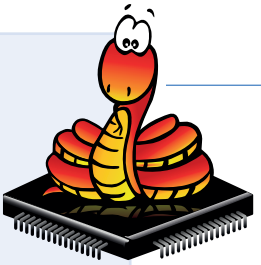
Le programme PIO proprement dit est codé comme la fonction Python `pio_prog()`. Il est constitué du code contenu dans la boucle entre `wrap_target()` et `wrap()`. Le programme bascule la broche *sideset 0* – qui est ici la broche de la LED GP25. Le temps d'activation/désactivation est régi par une boucle d'attente Y. La valeur initiale du registre Y est dans le registre X. La valeur du X est affectée de manière externe via la FIFO TX. Ceci est exécuté par la fonction `pull(noblock)`. Si un mot se trouve dans la FIFO TX, il est transféré dans le registre OSR. Le paramètre `noblock` fait en sorte que le contenu du registre X soit placé dans le registre OSR lorsque l'unité FIFO est vide pour que la séquence de commande continue.

Conclusion

Ici se termine l'aperçu de la programmation PIO du MCU RP2040. Il a été démontré que des interfaces puissantes peuvent être programmées à l'aide de quelques instructions simples. La facilité de programmation PIO de ces MCU les rend plus polyvalents, de sorte que vous n'aurez peut-être même pas besoin d'envisager l'utilisation de FPGA pour étendre les capacités du processeur.

Cet article ne peut donner qu'une vue d'ensemble de la programmation PIO. Il y a tellement plus à explorer. Si vous voulez un bon guide pratique, consultez la littérature [1][2][4][5][6][7], qui contient également de nombreux autres exemples.

220100-04 — VF : Denis Lafourcade



Listage MicroPython

```
001 from machine import Pin
002 from rp2 import PIO, StateMachine, asm_pio
003 from time import sleep
004
005 @asm_pio( sideset_init=(PIO.OUT_LOW))
006 def pio_prog():
007      wrap_target()
008      pull(noblock)
009      out(x, 32)
010
011      mov(y, x) .side(1)
012      label("Loop1")
013      jmp(y_dec, "Loop1")
014
015      mov(y, x) .side(0)
016      label("Loop2")
017      jmp(y_dec, "Loop2")
018      wrap()
019
020 class PIOAPP:
021      def __init__(self, sm_id, pinA, periodLength, count_freq):
023         self._sm = StateMachine(sm_id, pio_prog, freq=count_freq, sideset_base=Pin(pinA)
025         self._sm.active(1) # start PIO execution
026
027      def set(self, value): self._sm.put(value) # put val into TX-FIFO
029
030 pio1 = PIOAPP(0, pinA=25, periodLength=1, count_freq=1_000_000)
032 print("ready1")
033 while True:
034      pio1.set(200000) # 200 ms On/Off time
035      sleep(1) # for one second
036      pio1.set(100000) # 100 ms On/Off time
037      sleep(1) # for one second
038      print("idle")
038      print("idle")
```

Des questions, des commentaires ?

Pour toute question technique relative à cet article, veuillez contacter l'auteur (ossmann@fh-aachen.de) ou contacter Elektor (redaction@elektor.fr).

Produits

- **Raspberry Pi Pico RP2040 (SKU 19562)**
  https://elektor.com/19562
- **Dogan Ibrahim, *« Dogan Ibrahim, Raspberry Pi Pico Essentials »* (SKU 19673)**
  https://elektor.fr/19673

LIENS

[1] Démarrer avec Raspberry Pi Pico : https://datasheets.raspberrypi.com/pico/getting-started-with-pico.pdf
[2] Manuel d'installation de la chaîne d'outils par Shawn Hymel : https://tinyurl.com/yde5dd8a
[3] Téléchargement du logiciel : http://www.elektormagazine.fr/220100-04
[4] Fiche technique du RP2040 : https://datasheets.raspberrypi.com/rp2040/rp2040-datasheet.pdf
[5] Raspberry Pi Pico SDK : https://datasheets.raspberrypi.com/pico/raspberry-pi-pico-c-sdk.pdf
[6] Documentation du SDK : https://raspberrypi.github.io/pico-sdk-doxygen/index.html
[7] RP2040 MicroPython : https://docs.micropython.org/en/latest/library/rp2.html

Poor Man's ChipTweaker

Nous avons des moyens (abordables) de vous faire parler

Luka Matic (Croatie)

Les HSM (*Hardware Security Modules*, en français, boîtes noires transactionnelles) sont des dispositifs électroniques dotés de différents systèmes de protection matérielle contre la lecture non autorisée des données secrètes qu'ils contiennent. Ils servent généralement au chiffrement ou aux signatures numériques, mais ils peuvent aussi effectuer d'autres tâches. À titre d'exemple, les cartes bancaires ou les cartes d'accès sont des HSM. En cas de vol ou de perte, ces cartes doivent garder secrets le plus longtemps possible les codes confidentiels et les clés privées. En outre, de nombreux microcontrôleurs à usage général possèdent également des fonctions de protection de la mémoire, destinées à empêcher la copie illégale de microprogrammes exclusifs. L'outil Poor Man's ChipTweaker (PMCT) présenté ici sert à découvrir les éléments de base des attaques non invasives contre les HSM. Grâce à cet outil, vous allez pouvoir tenter de déverrouiller un microcontrôleur à mémoire protégée.

Comme nous l'avons expliqué dans un article précédent [1], nos espions à petit budget Alice et Bob ne peuvent pas compter sur la construction de leurs propres HSM, car cela nécessite un équipement spécialisé coûteux, et des connaissances qu'ils ne possèdent probablement pas. Pour leurs opérations critiques, ils utilisent leur propre matériel construit avec des composants bon marché et polyvalents auxquels ils peuvent se fier. Cela fonctionne bien pour eux, tant que les procédures de sécurité des opérations (OpSec) sont respectées. D'un autre côté, ils ne peuvent pas non plus éviter d'utiliser des HSM, les cartes bancaires par exemple, et ils peuvent tomber sur un modèle dont ils pourraient vouloir percer les secrets. L'outil PMCT leur sera donc très utile car il leur permet d'apprendre les éléments de base des attaques non invasives contre les HSM.

Attaques matérielles

Les attaques visant les HSM appartiennent pour l'essentiel à trois types :

1. **Non invasive :** Le HSM n'est ni ouvert ni décapsulé. L'attaque est réalisée à l'aide de différents signaux électriques appliqués à un port de communication ordinaire et aux broches d'alimentation du HSM. C'est ce pour quoi l'outil PMCT a été conçu. Une attaque peu coûteuse.
2. **Semi-invasive :** La couche supérieure de la puce de silicium du HSM est retirée. Les contacts électriques de la puce en silicium ne sont pas exposés. L'attaque est réalisée en illuminant certaines parties des microcircuits du HSM à l'aide d'impulsions lumineuses. Une attaque de coût modéré.
3. **Invasive :** La puce en silicium est entièrement exposée, ce qui permet de connecter les microsondes aux contacts de la puce. L'attaque est réalisée en injectant des signaux électriques dans les microcircuits du HSM. Il s'agit d'une attaque coûteuse qui nécessite un équipement hautement spécialisé.

De puissants systèmes « ChipWhisperer » numériques sont bien sûr disponibles [2]. Mais ce que je voulais faire, c'était créer un dispositif basé sur une technologie plus ancienne, similaire à ce que le Dr Skorobogatov a fait [3], avec tous les signaux analogiques et numériques entièrement accessibles, ce qui pourrait être une plateforme d'apprentissage plus performante pour comprendre les principes des attaques non invasives.

Spécifications

Les attaques non invasives se basent sur différentes méthodes d'injection de défauts, comme l'application de données mal formatées ou de signaux d'amplitude et de fréquence incorrectes à l'une des broches du HSM (dont les broches d'alimentation). Ces méthodes peuvent faire en sorte que le DUT (*Device Under Test*, en français Dispositif testé, ou plutôt attaqué) effectue de nombreuses actions incontrôlées et, si possible, révèle ses secrets.

Les attaques non invasives nécessitent beaucoup de traitements de données, d'ajustements et de recherches complexes pour chaque dispositif testé particulier, mais s'il est possible de le déverrouiller de manière non invasive, l'outil PMCT, simple et bon marché, peut vous aider à le faire. Il dispose des fonctionnalités suivantes :

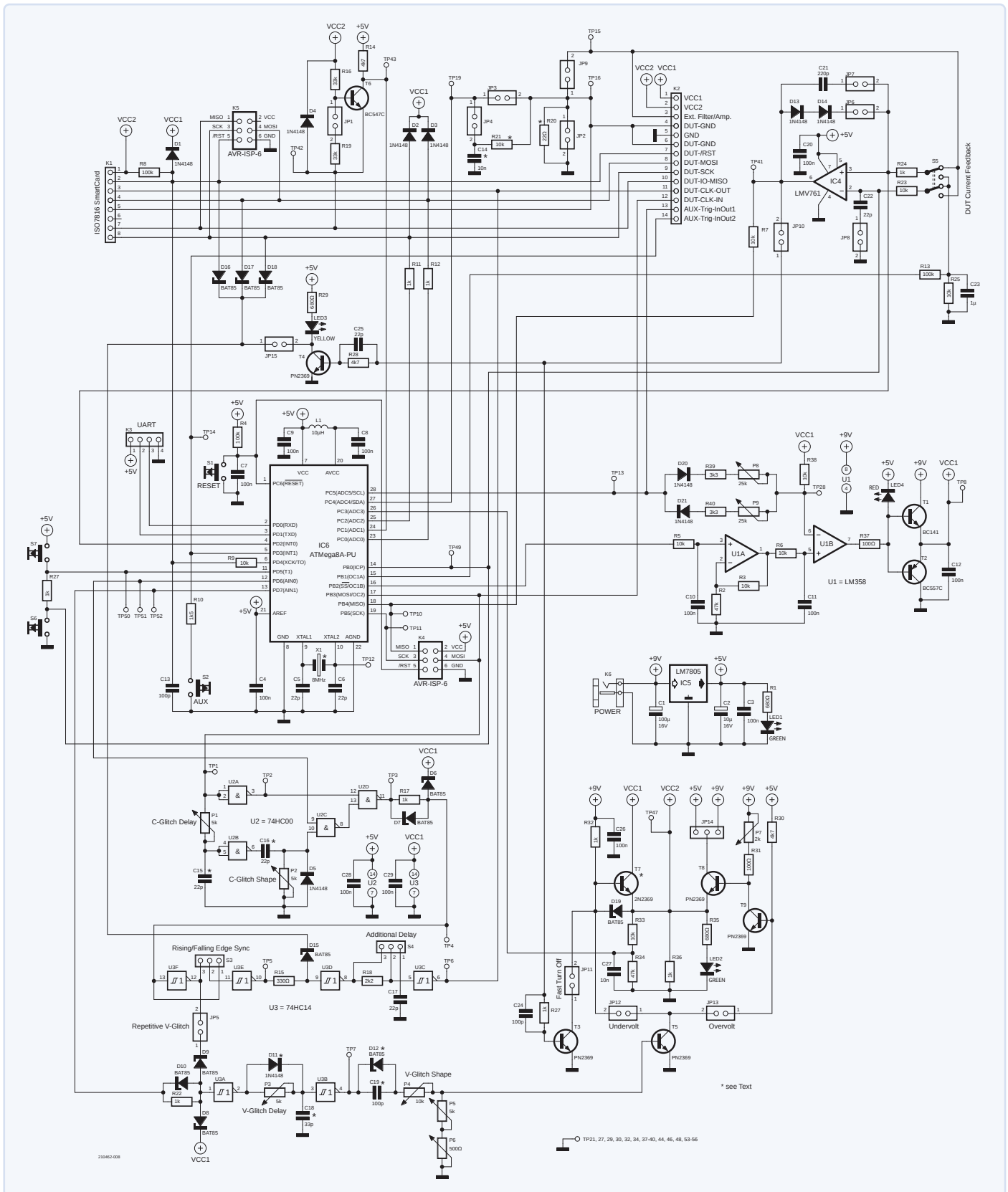


*Figure 1. Schéma de l'outil Poor Man's ChipTweaker (PMCT). À noter l'utilisation de transistors rapides de type 2369. Pour des questions de dissipation thermique, le transistor T7 doit être de type TO-18 (boîtier métallique, 2N2369). Les types PN2396 en boîtier plastique utilisés pour les autres peuvent être moins chers et/ou plus faciles à trouver. Les autres composants portant la marque « * » devraient être montés sur support pour permettre d'expérimenter facilement différentes valeurs.*

1. **Générateur de défaut de signal d'horloge :** L'insertion de signaux trop rapides sur l'entrée d'horloge (CLK) du HSM peut faire sauter une instruction machine ou l'exécuter de manière incorrecte, et donc, par exemple, ignorer une vérification du code confidentiel.
2. **Générateur de défaut de tension :** La modification de la tension d'alimentation du HSM hors de sa plage de fonctionnement nominale peut également entraîner des opérations incontrôlées. L'outil PMCT peut provoquer des défauts de tension lents, moyens et rapides, sous forme de surtensions ou de tensions insuffisantes.
3. **Alimentation variable entre 1,0 V et 6,0 V du dispositif testé :** Commandée par le microcontrôleur du PMCT.
4. **Comparateur pour désactiver rapidement un dispositif testé :** Au-dessus ou au-dessous d'un certain seuil de courant d'alimentation (par exemple, pour une attaque de sabotage d'écriture dans une mémoire EEPROM), armé-désarmé ou directement déclenché par le microcontrôleur principal.
5. **Interface SPI avec « bit-banging » (génération de signaux par logiciel) et UART bidirectionnelle mono-broche** pour la communication avec le dispositif testé (y compris tous les types de cartes à puce), comprenant des dispositifs de décalage de niveau de tension bidirectionnels couvrant la plage comprise entre 1,5 V et 6,0 V.

Au-delà des attaques citées, le PMCT peut effectuer toutes sortes d'attaques de synchronisation et d'analyse de puissance en combinaison avec un oscilloscope numérique (par exemple, un SmartScope de LabNation), un oscilloscope analogique à mémoire de 100 à 200 MHz (par exemple, le Tektronix 466) et un PC avec un logiciel comme MATLAB pour l'analyse des données hors connexion. Contrairement aux dispositifs « ChipWhisperer » intégralement numériques, tous les signaux analogiques et numériques sont accessibles pour être étudiés et analysés, et il est possible d'ajuster plus précisément la synchronisation du défaut grâce à un réglage analogique continu avec des potentiomètres dépourvus de pas de quantification. Un ATmega8 lent peut ainsi coordonner toute la procédure d'attaque, et aucun circuit FPGA n'est donc nécessaire. Il est possible de générer des défauts d'une durée inférieure à 10 ns, ce qui induit des erreurs pour les unités centrales conçues pour fonctionner jusqu'à 50 MHz, et plus. De nombreux microcontrôleurs et cartes à puce ne fonctionnent que jusqu'à 20 MHz.

Description du matériel

Au centre du schéma (**figure 1**), nous trouvons le microcontrôleur (MCU, IC6, ATmega8) qui coordonne les procédures d'attaque. Le circuit autour d'IC1 est une source de tension variable pour le dispositif testé. Sa tension de sortie VCC1 est commandée par modulation de largeur d'impulsions (PWM) par la sortie de temporisation OC1B d'IC6. Il est ainsi possible de produire des défauts de tension lents. La broche PC5 du microcontrôleur peut être dans l'un des trois états suivants : L (bas), H (haut) et high-Z (haute impédance), ce qui lui permet de modifier rapidement l'amplification d'IC1B, et donc de produire trois valeurs différentes de VCC1, définies par P8 et P9. C'est ainsi que l'on produit un défaut de tension à vitesse moyenne (mesurée en microsecondes). La LED rouge LED4 va s'allumer à titre d'avertissement et va limiter VCC1 à environ 6 V, ce qui n'est pas trop élevé pour un dispositif alimenté en 5 V.

Production de défauts

Le dispositif testé est normalement alimenté par VCC2 au travers du transistor T7 (un 2N2369 en boîtier TO-18). Le transistor T5 peut rapidement ramener sa base à l'état bas (en insérant le cavalier JP12), ce qui entraîne un défaut de sous-tension rapide (de l'ordre de 10 ns). Si le cavalier JP13 est inséré, le transistor T8 amènera rapidement la tension VCC2 au niveau haut, créant ainsi un défaut rapide de surtension. Les défauts de tension rapides sont formés par C19, P4, P5 et P6. Le retard du défaut de tension par rapport à l'impulsion CLK est ajusté avec P3. Lorsque la sortie PD7 du microcontrôleur est mise à l'état haut, un seul défaut de tension est déclenché. Si la sortie reste haute, et si le cavalier JP5 est court-circuité, les défauts de tension seront déclenchés à chaque impulsion CLK.

Le circuit IC2 sert à générer un défaut d'horloge sur le dispositif testé. La sortie en PWM OC2 du microcontrôleur produit l'impulsion CLK pour le dispositif testé. Si le port PD6 est à l'état haut, IC2C transmettra un défaut de courte durée à IC2D à chaque impulsion CLK. Il est possible de sélectionner la polarité du défaut à l'aide d'un cavalier ou d'un commutateur sur S3.

IC4 est un comparateur rapide avec retour positif (amélioré par D13, D14 et C21) destiné à éteindre le dispositif testé en quelques nanosecondes. Il est également possible de le faire à l'aide d'une commande du microcontrôleur (PD5 ou PD2 pour allumer, PB0 pour éteindre le dispositif testé) ou en appuyant sur les boutons S6 et S7.

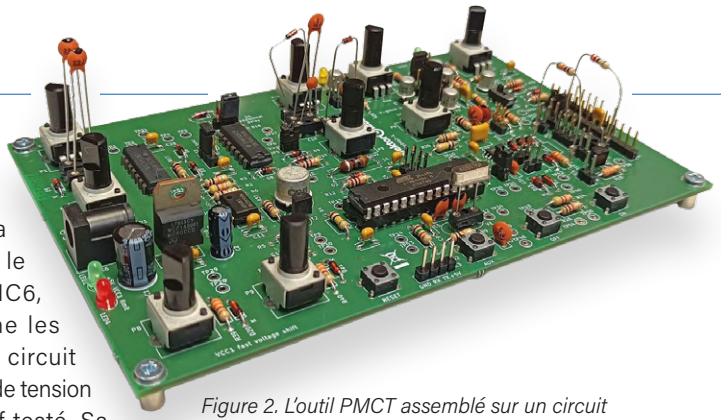


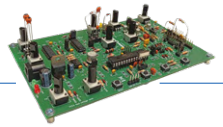
*Figure 2. L'outil PMCT assemblé sur un circuit imprimé de bonne taille. Les composants montés sur support portent la marque « * ».*

Current Consumption Attacks

IC4 peut également être configuré pour éteindre le dispositif testé si la consommation de courant de ce dernier (mesurée avec la résistance R20) est supérieure ou inférieure (selon une sélection effectuée par S5) à une tension de seuil (sur C23, contrôlée par PWM sur la broche OC1A). Cette approche est généralement utilisée pour empêcher le dispositif testé dans sa mémoire EEPROM interne (qui consomme du courant supplémentaire). La sortie d'IC4 active le transistor T3, qui, à son tour, désactive T7. Elle active également le transistor T4 (désactivation auxiliaire) pour amener toutes les lignes de la carte à puce vers le bas et éviter toute alimentation fantôme.

L'entrée analogique ADC4 surveille le courant du dispositif testé. Comme il s'agit d'un CA/N lent, le signal peut être filtré par R21/C14. Il peut aussi être traité par un filtre/amplificateur externe plus rapide (sur le connecteur K2), si nécessaire. Pour une attaque de synchronisation ou d'analyse d'alimentation, ce signal est généralement enregistré par un oscilloscope pour être traité et analysé hors ligne avec un outil comme MATLAB.

Comme le microcontrôleur fonctionne à partir de 5 V alors que le dispositif testé est alimenté par la tension VCC2 (généralement) plus faible, le dispositif de décalage de niveau bidirectionnel (construit autour du transistor T6) est nécessaire pour la broche 7 d'E/S du dispositif testé sur le connecteur K1 (un port UART bidirectionnel, qu'utilisent aujourd'hui la plupart des cartes à puce). Les broches 4 et 8 servent pour les cartes à puce avec interface SPI (par exemple, la FunCard). Comme elles sont unidirectionnelles, les dispositifs de décalage de niveau sont plus simples (diodes D2 et D3).



Circuit imprimé

Pour faciliter la construction de votre propre outil PMCT, une carte à circuit imprimé (PCB) de la taille du format Eurocard a été conçue pour lui (**figure 2**). (Voir [4] pour accéder au projet KiCad et aux fichiers gerber.) Tous les éléments sont des composants de type à trou traversant, à l'exception d'IC4, conditionné en boîtier SOIC-8. L'assemblage de la carte ne devrait pas poser de problèmes, mais il faut faire attention à certains détails :

- Les cavaliers à 3 broches « Sxx » peuvent être plus pratiques comme interrupteurs unipolaires et bidirectionnels (SPDT) à glissière ou à bascule.
- Vous préférerez peut-être utiliser des trimmers plutôt que des potentiomètres. Les trimmers sont plus faciles à trouver et offrent une plus large gamme de valeurs.
- Le transistor T7 doit être un 2N2369 dans un boîtier métallique TO-18 pour des raisons de dissipation thermique. Les transistors PN2396 TO-92 peuvent être remplacés par des types 2N2369, selon ce qui est le plus facile à trouver.
- Les composants marqués d'un symbole « * » (par exemple, D11 et C18, etc., à l'exception de T7) devraient être montés sur support pour expérimenter facilement différents types et valeurs.

La carte comporte de nombreux points de test avec une connexion de masse à proximité. N'essayez pas de comprendre leur numérotation ; elle ne répond à aucune logique.

Préparer l'attaque

Le dispositif testé pour les applications de démonstration est une FunCard standard (qui contient un microcontrôleur Microchip AVR AT90S8515 et une mémoire EEPROM AT24C). Sans aucune protection sérieuse, ce dispositif est relativement facile à attaquer. Ce sera donc la première étape si vous voulez avancer dans ce domaine. Chargez le micrologiciel dans la FunCard accessible en [4] par le biais du connecteur ISP K5. Ensuite, utilisez une des deux variantes du micrologiciel ATmega8 (concernant l'attaque d'une FunCard, également en [4]) pour programmer le microcontrôleur IC6 du PMCT à l'aide du connecteur ISP K4 (et non K5). Vous pouvez maintenant essayer quelques actions élémentaires.

Connectez un terminal série au port UART K3 et configurez-le pour 9600-8-N-1 (soit 9600n81). Après avoir actionné le bouton de réinitialisation S1, vous devriez obtenir un écran comme celui représenté sur la **figure 3**. Les options 0 et 1 permettent d'envoyer des commandes simples au microcontrôleur et à la FunCard. Toutes les commandes ont une longueur de 4 octets, le dernier étant toujours égal à 0x0d (CR). Référez-vous au **tableau 1** et au **tableau 2**.

Démonstration d'attaque par défaut d'horloge

Il s'agit d'une démonstration d'une attaque élémentaire de type défaut d'horloge. La FunCard effectue plusieurs opérations en 16 bits sur deux opérandes d'entrée, situés dans la mémoire EEPROM interne de son microcontrôleur aux adresses 0x00, 0x01, 0x02 et 0x03 au format « big-endian » (mots de poids fort en tête). Le fait d'émettre une impulsion d'horloge (impulsion trop rapide pour

```
> 0 : Send a single command to Smartcard
> 1 : Send a single command to MCU
> 2 : Find minimum Vcc2
> 3 : Adjust minimum Vt
> 5 : Funcard no glitch demo
> 6 : Funcard clock glitch demo
> 7 : Funcard volt glitch demo
> 8 : Volt glitch loop test
> 9 : Volt & Clock glitch loop test
> A : Fast Vcc2 test
> B : Brute-force PIN demo, Funcard MCU internal counter
> b : Brute-force PIN demo, Funcard AT24C external counter
> R : Reset MCU & Smartcard

Vcc2 RAW set to:0xDD    Vcc2 feedback: 5,092V
Vt RAW set to:0xFF    Vt scaled set to: 0,452V    Vi feedback: 0,166V
Smartcard fclk division factor set to RAW:0x00

> Select Option (0-X): 5

No glitch demo.

Answer To Reset: 0x45 0x67 0x78 0x9A 0xBC 0xDE 0xF0 0x12 0x34 0x56 0x78 0xDE 0xAD 0x01 0x23 0x00 0x00 0x00 0x00 0x00
```

Figure 3. Vous pouvez choisir dans un menu l'attaque que vous souhaitez.

Commande	Description
V2x	« x » est un nombre sur 8 bits définissant le ratio de PWM sur OC1B pour fixer la tension VCC2.
Vtx	« x » est un nombre sur 8 bits définissant le ratio de PWM sur OC1A pour fixer la tension Vt, un seuil sur C23 pour activer IC4.
Fxy	« x » est un facteur de division de fréquence brute, destiné à produire une impulsion CLK pour le dispositif testé sur OC2. Si le facteur est fixé à 0, OC2 fonctionnera à la moitié de la fréquence du cristal du microcontrôleur ; « y » n'est pas pris en compte.
GLx	Les 3 bits de poids le plus faible de « x » définissent les broches PD7, PD6 et PC5 du microcontrôleur ; pour le test.
onx	Mise en marche du dispositif testé ; « x » n'est pas pris en compte.
ofx	Arrêt du dispositif testé ; « x » n'est pas pris en compte.
Sxy	Enregistrement des paramètres actuels dans l'EEPROM du microcontrôleur ; « x » et « y » ne sont pas pris en compte.

Tableau 1. Commandes adressées au microcontrôleur Terminer chaque commande par 0x0d (CR).

Commande	Description
Rxy	Lecture d'un octet à l'adresse « x » de la mémoire EEPROM interne de l'AT90S ; « y » n'est pas pris en compte.
Wxy	Écriture d'un octet « y » à l'adresse « x » de la mémoire EEPROM interne de l'AT90S.
Pxy	Test d'un code confidentiel « xy » (au format BCD). Ce code est stocké dans la mémoire EEPROM interne de l'AT90S.
rxy	Lecture d'un octet à l'adresse « x » de la mémoire EEPROM de l'AT24Cv; « y » n'est pas pris en compte.
wxy	Écriture d'un octet « y » à l'adresse « x » de la mémoire EEPROM de l'AT24C.
pxy	Test d'un code confidentiel « xy » (au format BCD). Ce code est stocké dans la mémoire EEPROM de l'AT24C.

Tableau 2. Commandes adressées à la carte à puce Terminer chaque commande par 0x0d (CR).

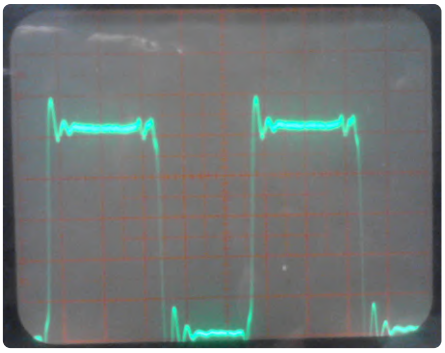


Figure 4. Un signal d'horloge propre sans défauts. L'oscilloscope a été réglé sur 50 ns/div sur l'axe horizontal et 1 V/div sur l'axe vertical.

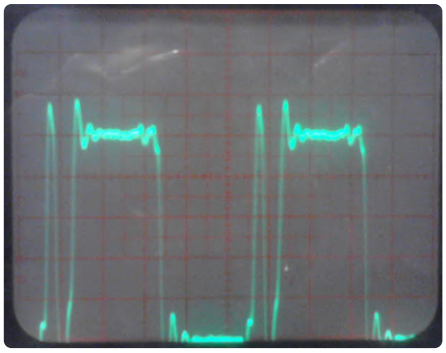


Figure 5. Un signal d'horloge assorti d'un défaut (50 ns/div axe horizontal, 1 V/div axe vertical). Réglage des potentiomètres P1 et P2 pour créer un défaut comme indiqué ici.

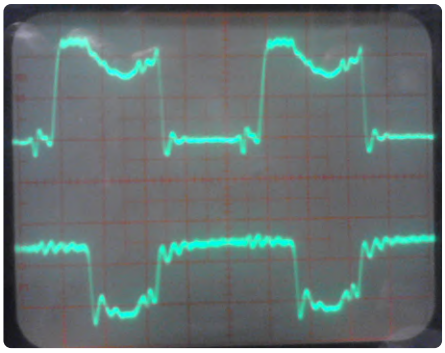


Figure 7. Le défaut s'est déclenché 30 à 40 ns suite au front montant de l'impulsion d'horloge. La trace supérieure est le signal d'horloge, la trace inférieure est la tension d'alimentation VCC2 (50 ns/div horizontal ; 1 V/div vertical).

le dispositif testé) à différentes étapes des calculs produit un certain nombre d'erreurs. Il est possible que le résultat d'une opération mathématique soit erroné, ou simplement qu'une instruction machine soit ignorée.
Se référer aux **figures 4**, **5** et **6** concernant les signaux et les résultats. Un défaut efficace pour la FunCard est une impulsion basse active, d'une durée comprise entre 10 ns et 20 ns, déclenchée environ 10 à 20 ns après le front montant de l'horloge de l'unité centrale (voir **figure 5**).

Démonstration d'attaque par défaut de tension

Il s'agit d'une démonstration d'une attaque élémentaire de type défaut de tension. La FunCard procède aux mêmes opérations que dans l'exemple précédent. Un défaut de sous-tension efficace a une durée d'au moins 50 ns et donne de bons résultats avec le timing illustré sur la **figure 7**. Ici, le défaut s'est déclenché 30 à 40 ns suite au front montant de l'impulsion d'horloge. La tension d'alimentation de la FunCard a été fixée à 2,24 V. Le défaut de tension présentait une profondeur d'environ 1,5 V.
Ajustez la forme du défaut de tension avec P4, P5 et P6. Utilisez P3, S3 et S4 pour régler le retard et la synchronisation du défaut. Les résultats sont affichés de la même manière que dans l'exemple précédent (**figure 8**).

Déverrouillage de la protection de la mémoire pour extraire le micrologiciel

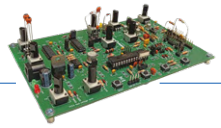
Chaque carte à puce ou microcontrôleur protégé nécessite une procédure d'attaque différente. De nombreux tâtonnements et recherches complexes sont nécessaires pour trouver les points faibles possibles. Adaptée à de nombreux microcontrôleurs AVR Microchip, la procédure s'appuie sur un défaut de sous-tension lent.
Elle fonctionne comme suit. Lorsque le microcontrôleur FunCard (AT90S8515) est protégé, les deux bits de verrouillage de la mémoire sont programmés à 0. À partir de là, la mémoire flash ne peut plus être lue, mais seulement effacée. En mode de programmation série (son entrée de réinitialisation est ramenée au niveau bas 0 V), la commande *Chip Erase* va d'abord effacer la mémoire flash (en positionnant tous les octets qu'elle contient à la valeur 0xFF), puis effacer les bits de verrouillage (en les désactivant pour les ramener à 1).
Le défaut de conception présent sur de nombreux contrôleurs AVR (mais pas tous) permet de déverrouiller la protection de la manière suivante. Si la tension d'alimentation est abaissée au-dessous du minimum nominal (2,7 V), jusqu'à une valeur comprise entre 1,6 V et 1,7 V, il n'y aura pas assez de puissance pour effacer la mémoire flash, même si l'effacement des bits de verrouillage sera toujours possible. L'attaque est lancée à 1,1 V et la tension est progressivement augmentée. Les bits de verrouillage ont été supprimés avec succès à 1,62 V sans effacer la mémoire flash !
Lorsque vous essayez de lire le micrologiciel ou la signature du dispositif à partir d'un microcontrôleur AVR protégé, la réponse sera « 0x00, 0x01, 0x02, 0x03 » et vous savez donc que la mémoire est protégée (**figure 9**). Une fois que vous obtenez la signature correcte (« 0x1E, 0x93, 0x01 » pour le microcontrôleur AT90S8515), les bits de verrouillage de la mémoire ont été supprimés et la mémoire du programme peut ainsi être lue en utilisant n'importe quel programmateur ISP.

```
> 0 : Send a single command to Smartcard
> 1 : Send a single command to MCU
> 2 : Find minimum Vcc2
> 3 : Adjust minimum Vt
> 5 : Funcard no glitch demo
> 6 : Funcard clock glitch demo
> 7 : Funcard volt glitch demo
> 8 : Volt glitch loop test
> 9 : Volt & Clock glitch loop test
> A : Fast Vcc2 test
> B : Brute-force PIN demo, Funcard MCU internal counter
> b : Brute-force PIN demo, Funcard AT24C external counter
> R : Reset MCU & Smartcard

Vcc2 RAW set to:0xDD    Vcc2 feedback: 5,092V
Vt RAW set to:0xFF    Vt scaled set to: 0,452V    Vi feedback: 0,166V
Smartcard fclk division factor set to RAW:0x00

> Select Option (0-X): 6

Clock glitch demo.

Answer To Reset: 0x45 0x67 0x78 0x9A 0xBC 0xDE 0xF0 0x12 0x34 0x56 0x78 0xDE 0xAD 0x01 0x23 0x00 0x00 0x00 0x00 0x00
0xF62F 0x01 0x25 0x25  OK!
0xF34B 0x02 0x3E 0x3E  OK!
0xF62F 0x03 0x25 0x25  OK!
0xF62F 0x04 0x25 0x25  OK!
0xF5E4 0x05 0xD9 0xD9  OK!
0xF62F 0x06 0x25 0x25  OK!
0xF62F 0x07 0x25 0x25  OK!
0xF576 0x08 0x6B 0x6B  OK!
0x48E8 0x09 0x30 0x30  OK!
0x47E8 0x0A 0x2F 0x2F  OK!
0xF676 0x0B 0x6C 0x6C  OK!
0x47E8 0x0C 0x2F 0x2F  OK!
0xF62F 0x0D 0x25 0x25  OK!
0x0000 0x0E 0x00 0x00  OK!
0x5555 0x0F 0x55 0xAA  Error!
0xF62F 0x10 0x25 0x25  OK!
0xF62F 0x11 0x25 0x25  OK!
0xF62F 0x12 0x25 0x25  OK!
0xF62F 0x13 0x25 0x25  OK!
```

Figure 6. Fenêtre de terminal montrant les résultats de l'attaque par défaut d'horloge. La **colonne 1** *contient le résultat sur 16 bits de l'opération mathématique exécutée par le dispositif testé. Il sera erroné si le défaut a réussi.* **Colonne 2 :** *délai de l'attaque. L'application du défaut à des moments différents donne des résultats différents.* **Colonne 3** *: somme de contrôle basée sur la somme des deux octets du résultat sur 16 bits calculé par la FunCard.* **Colonne 4 :** *somme de contrôle calculée par le microcontrôleur du PMCT.* **Colonne 5 :** *« Error! » si les colonnes 3 et 4 ne correspondent pas.*

Attaque par analyse de la consommation d'énergie d'une carte bancaire

Les cartes bancaires sont de plus en plus sophistiquées et utilisent de multiples



```
> 0 : Send a single command to Smartcard
> 1 : Send a single command to MCU
> 2 : Find minimum Vcc2
> 3 : Adjust minimum Vt
> 5 : Funcard no glitch demo
> 6 : Funcard clock glitch demo
> 7 : Funcard volt glitch demo
> 8 : Volt glitch loop test
> 9 : Volt & Clock glitch loop test
> A : Fast Vcc2 test
> B : Brute-force PIN demo, Funcard MCU internal counter
> b : Brute-force PIN demo, Funcard AT24C external counter
> R : Reset MCU & Smartcard

Vcc2 RAW set to:0x60    Vcc2 feedback: 2,238V
Vt RAW set to:0xFF    Vt scaled set to: 0,452V    Vi feedback: 0,053V
Smartcard fclk division factor set to RAW:0x00

> Select Option (0-X): 7

Volt glitch demo.

Answer To Reset: 0x45 0x67 0x78 0x9A 0xBC 0xDE 0xF0 0x12 0x34 0x56 0x78 0xDE 0xAD 0x01 0x23 0x00 0x00 0x00 0x00 0x00
0xF62F 0x01 0x25 0x25  OK!
0xF62F 0x02 0x25 0x25  OK!
0xF5E4 0x03 0xD9 0xD9  OK!
0xF62F 0x04 0x25 0x25  OK!
0x3F4B 0x05 0x8A 0x8A  OK!
0xF62F 0x06 0x25 0x25  OK!
0xF62F 0x07 0x25 0x25  OK!
0x482F 0x08 0x77 0x77  OK!
0x492F 0x09 0x78 0x78  OK!
0x482F 0x0A 0x77 0x77  OK!
0xF676 0x0B 0x6C 0x6C  OK!
0x482F 0x0C 0x77 0x77  OK!
0xF62F 0x0D 0x25 0x25  OK!
0x0000 0x0E 0x00 0x00  OK!
0xF642 0x0F 0x38 0x38  OK!
0x0000 0x10 0x00 0x00  OK!
0xF62F 0x11 0x2F 0x25  Error!
0xF62F 0x12 0x25 0x25  OK!
0xF62F 0x13 0x25 0x25  OK!
```

Figure 8. Fenêtre de terminal montrant les résultats de l'attaque par défaut de tension. Les colonnes sont formatées de la même façon que pour l'attaque par défaut d'horloge (voir la figure 6).

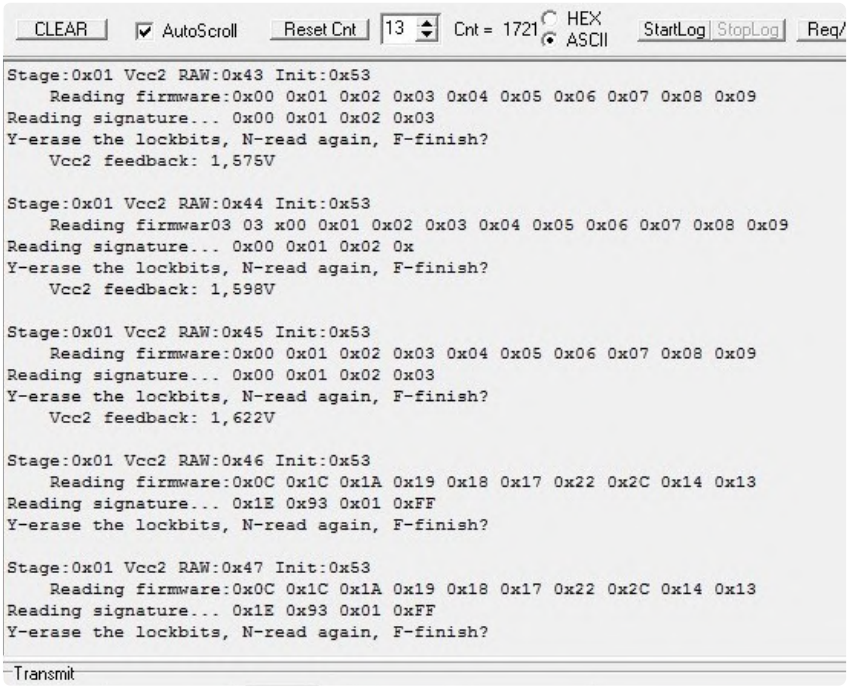


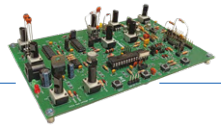
Figure 9. Déverrouillage de la protection de la mémoire flash.

Figure 10. Courant d'alimentation (trace jaune) d'une carte bancaire lors de la vérification d'un code confidentiel incorrect.

méthodes pour protéger physiquement les zones de mémoire critiques. Elles essaient également de masquer le courant d'alimentation afin que les opérations critiques soient plus difficiles à détecter. Si certaines procédures, comme la vérification du code confidentiel saisi, peuvent être localisées avec précision dans le temps, nous savons alors à quel moment des anomalies peuvent être déclenchées pour tenter de contourner les protections.

Voici quelques-unes des méthodes connues de protection des cartes bancaires :

1. Utilisation d'une horloge RC interne pour les opérations de sécurité critiques, passage à une CLK externe uniquement pour une synchronisation précise, par exemple pour la communication via l'UART. Il est ainsi possible d'éviter les attaques de type « défaut d'horloge ».
2. Utilisation de pompes de charge très rapides sur de petits condensateurs d'alimentation internes (de l'ordre du pF) pour maintenir une tension d'alimentation stable. Ceci permet d'éviter les attaques à l'aide de défauts de tension.
3. Modification aléatoire des demi-périodes de l'horloge RC interne (à l'aise d'un générateur de nombres aléatoires interne - TRNG). Ainsi, les opérations critiques (notamment la vérification du code confidentiel) ne se produiront pas toujours au même moment, ce qui rendra les attaques plus difficiles.
4. Ajout d'un bruit aléatoire au courant de l'alimentation électrique. Il est ainsi possible d'éviter les attaques par analyse de l'alimentation (**figure 10**).
5. Lors de la vérification d'un code confidentiel, il convient d'abord de diminuer le compteur de tentatives de code dans l'EEPROM de la carte à puce, puis de vérifier le code et d'augmenter ce compteur de tentatives si ledit code est correct. Les anciennes cartes n'écrivaient que dans l'EEPROM pour diminuer le compteur en cas de saisie erronée du code confidentiel. Ainsi, les attaques par force brute pouvaient extraire le code confidentiel en coupant rapidement l'alimentation électrique après chaque tentative erronée.
6. Utilisation de fonctions très soigneusement conçues pour les opérations critiques relatives à la sécurité (programmation en assembleur requise ici !), qui nécessitent toujours le même nombre de cycles d'horloge de l'unité centrale et (éventuellement) la même quantité d'énergie, quelles que soient les variables d'entrée. L'approche permet d'éviter les attaques de synchronisation et par analyse de la consommation d'énergie.



7. Prolongation d'une opération critique pour la sécurité à 200 ms, par exemple, contre 1 ms à l'origine. Un signal utile de 1 ms est donc caché dans un délai de 199 ms servant au traitement de données inutiles.

La mise hors d'état de fonctionner d'une carte bancaire moderne de manière non invasive (si tant est que cela soit possible) nécessite une analyse approfondie des données enregistrées et un travail long et minutieux. Pour des informations plus complètes et détaillées sur les attaques par analyse de la consommation d'énergie, lisez l'ouvrage spécialisé *Power Analysis Attacks* [5].

Tester et découvrir

L'outil PMCT n'est pas un appareil high-tech, mais il peut servir à tester et découvrir des attaques non invasives sur les HSM. J'espère que vous avez trouvé cet article intéressant, au moins comme base pour vous lancer. De nos jours, la conception et l'attaque des HSM monopuce représentent un travail très difficile pour les personnes concernées. C'est ce qui explique que ce domaine reste ouvert à de nouvelles recherches.

Bon espionnage !

VF : Pascal Godart — 210462-04

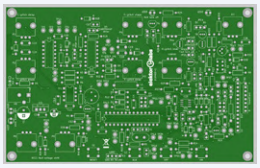
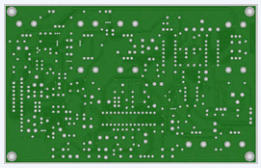
Des questions, des commentaires ?

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

Produits

- **LabNation SmartScope oscilloscope USB (SKU 17169)**
  www.elektor.fr/17169
- **Luka Matic, *Livre en anglais « Electronic Security and Espionage »*, Elektor 2021 (SKU 19903)**
  www.elektor.fr/19903

LISTE DES COMPOSANTS

Résistances

R1,R29,R35 = 680 Ω
R2,R34 = 47 kΩ
R3,R5,R6,R7,R9,R23,R25,R33,R38 = 10 kΩ
R4,R8,R13 = 100 kΩ
R10 = 1,5 kΩ
R11,R12,R17,R22,R24,R26,R27,R32,R36 = 1 kΩ
R14,R28,R30 = 4,7 kΩ
R15 = 330 Ω
R16,R19 = 33 kΩ
R18 = 2,2 kΩ
R20* = 22 Ω*
R21* = 10 kΩ*
R31,R37 = 100 Ω
R39,R40 = 3,3 kΩ
P1,P2,P3,P5 = potentiomètre, 5 kΩ, vertical
P4 = potentiomètre, 10 kΩ,vertical
P6 = potentiomètre, 500 Ω, vertical
P7 = potentiomètre, 2 kΩ, vertical
P8,P9 = potentiomètre, 25 kΩ,vertical

Condensateurs

C1 = 100 µF 16 V, pas de 3,5 mm
C2 = 10 µF 16 V, pas de 2,5 mm
C3,C4,C7,C8,C9,C10,C11,C12,C20,C26,C28,C29 = 100 nF, pas de 5 mm
C5,C6,C17,C22,C25 = 22 pF, pas de 2,5 mm
C13,C24 = 100 pF, pas de 2,5 mm
C14* = 10 nF, pas de 5 mm*
C27 = 10 nF, pas de 5 mm
C15*,C16* = 22 pF, pas de 5 mm*
C18* = 33 pF, pas de 5 mm*
C19* = 100 pF, pas de 5 mm*
C21 = 220 pF, pas de 2,5 mm
C23 = 1 µF, pas de 5 mm

Inductances

L1 = 10 µH

Semi-conducteurs

D1,D2,D3,D4,D5,D13,D14,D20,D21 = 1N4148
D11* = 1N4148*
D6,D7,D8,D9,D10,D15,D16,D17,D18,D19 = BAT85
D12* = BAT85*
IC1 = LM358
IC2 = 74HC00
IC3 = 74HC14
IC4 = LMV761, SOIC8
IC5 = 7805, TO220
C6* = ATmega8A-PU*
LED1,LED2 = LED, 3 mm, verte
LED3 = LED, 3 mm, jaune
LED4 = LED, 3 mm, rouge
T1 = BC141, TO39
T2 = BC557C
T3,T4,T5,T8,T9 = PN2369, TO92
T6 = BC547C
T7 = 2N2369, TO18

Divers

JP1-JP13,JP15 = barrette 2 broches, pas de 2,54 mm
JP14,S3,S4 = barrette 3 broches, pas de 2,54 mm
K1 = barrette 8 broches, pas de 2,54 mm
K2 = barrette 14 broches, pas de 2,54 mm
K3 = barrette 4 broches, pas de 2,54 mm
K4,K5 = barrette 2 rangées, 6 broches, pas de 2,54 mm
K6 = embase jack
S5 = commutateur à glissière (C&K JS202011CQN)
X1* = quartz 8 MHz*

* montage avec support pour faciliter l'expérimentation

Funcard. (Source : www.cellularcenter.it)

LIENS

[1] Luka Matic, « Boîte inviolable protégée par un témoin d'effraction », Elektor 5/ 2020 : https://www.elektormagazine.fr/magazine/elektor-148/58644
[2] ChipWhisperers from NewAE Technology Inc. : https://www.newae.com/chipwhisperer
[3] Sergei P. Skorobogatov, « Copy Protection in Modern Microcontrollers » : https://www.cl.cam.ac.uk/~sps32/mcu_lock.html
[4] Téléchargements pour cet article depuis Elektor Labs : https://www.elektormagazine.fr/labs/poor-mans-chipwhisperer-or-a-smartcard-tweaker
[5] S. Mangard, E. Oswald, T. Popp, Power Analysis Attacks: Revealing the Secrets of Smart Cards, Springer, 2007 : https://amzn.to/3RWBtuy

MagPi, le magazine **officiel** du Raspberry Pi

12 mois
Plus de
100 projets
Le prix
54,95 €

- **6 X MAGPI : ÉDITION IMPRIMÉE**
- **ACCÈS AUX ARCHIVES EN LIGNE DU MAGPI**

COMMANDEZ DÈS MAINTENANT AU
WWW.MAGPI.FR/ABO

MagPi
www.magpi.fr Magazine

générateur **de nombres réellement aléatoires** avec interface USB

deux PIC pour le prix d'un AVR

Matthias Wolf (Allemagne)

Il est utile de disposer d'un bon générateur de nombres réellement aléatoires (TRNG). Sans ce dernier, le cryptage des données de manière sécurisée est presque impossible. Les applications de jeux et de paris nécessitent également des TRNG de top-niveau. Elektor a publié un projet d'un TRNG analogique en 2017 ; dans cette mise-à-jour, nous avons rajouté une interface USB.

Le générateur de nombres réellement aléatoires (TRNG) basé sur des composants abordables par Luka Matic [1], utilise une carte SD pour enregistrer la séquence aléatoire produite. Dans ces pages, nous présentons une adaptation de son travail, où l'emplacement de la carte mémoire est remplacé par une interface USB.

Le nouveau circuit (**figure 1**) possède deux microcontrôleurs PIC de Microchip Technology au lieu d'un seul ATtiny2313A : un pour la partie traitement de signal analogique (PIC16F19156) et un pour l'interface USB (PIC18F25K50). Les deux microcontrôleurs communiquent entre eux via un bus SPI, isolé galvaniquement avec des optocoupleurs à haute vitesse pour empêcher au bruit numérique du circuit USB de perturber le signal analogique.

Étalonnage du filtre

Un petit programme sur PC a été écrit pour étalonner les filtres analogiques en temps réel de manière simple (**figure 2**).

Mes réglages pour les filtres analogiques sont les suivants :

- S5 : C37,C38,C39,C40,C41 ON avec C39 à 109 pF
- S6 : C45,C46,C47 ON avec C47 à 25 pF
- S7 : Switch1 ON et P6 à 409 Ω

Ces paramètres dépendent des composants utilisés pour construire le TRNG, vous pouvez en utiliser d'autres.

Le type de diode zener utilisée est également important. J'ai d'abord utilisé la BZX384C12-E3-08 de Vishay, mais le bruit généré était mauvais.

Finalement, le PIC16 utilise un taux d'échantillonnage légèrement plus élevé (800 kHz, 1,25 µs par échantillon) par rapport au TRNG original de Luka.

Allez chercher les fichiers !

Tous les fichiers du projet peuvent être téléchargés [2]. Ils comprennent les schémas (trois pages), le fichier CAD Target 3001, le micrologiciel pour le PIC16 (échantillonnage analogique, EDI MPLAB X avec le compilateur PCM C de CCS), le micrologiciel pour le PIC18 (traitement USB, EDI MPLAB X avec XC8), et l'application de bureau (C#, Visual Studio 2017 version *commmunity*).

190235-04

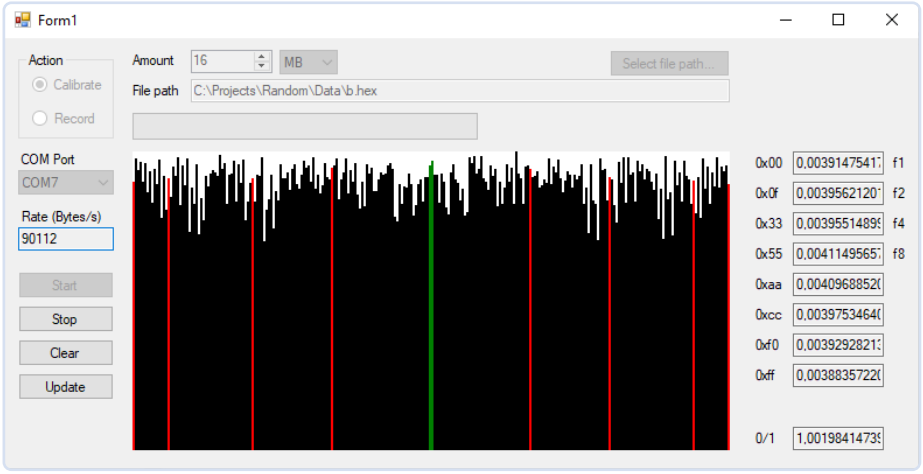


Figure 2. Copie d'écran du programme pour PC permettant d'étalonner les filtres analogiques et d'enregistrer des données aléatoires.

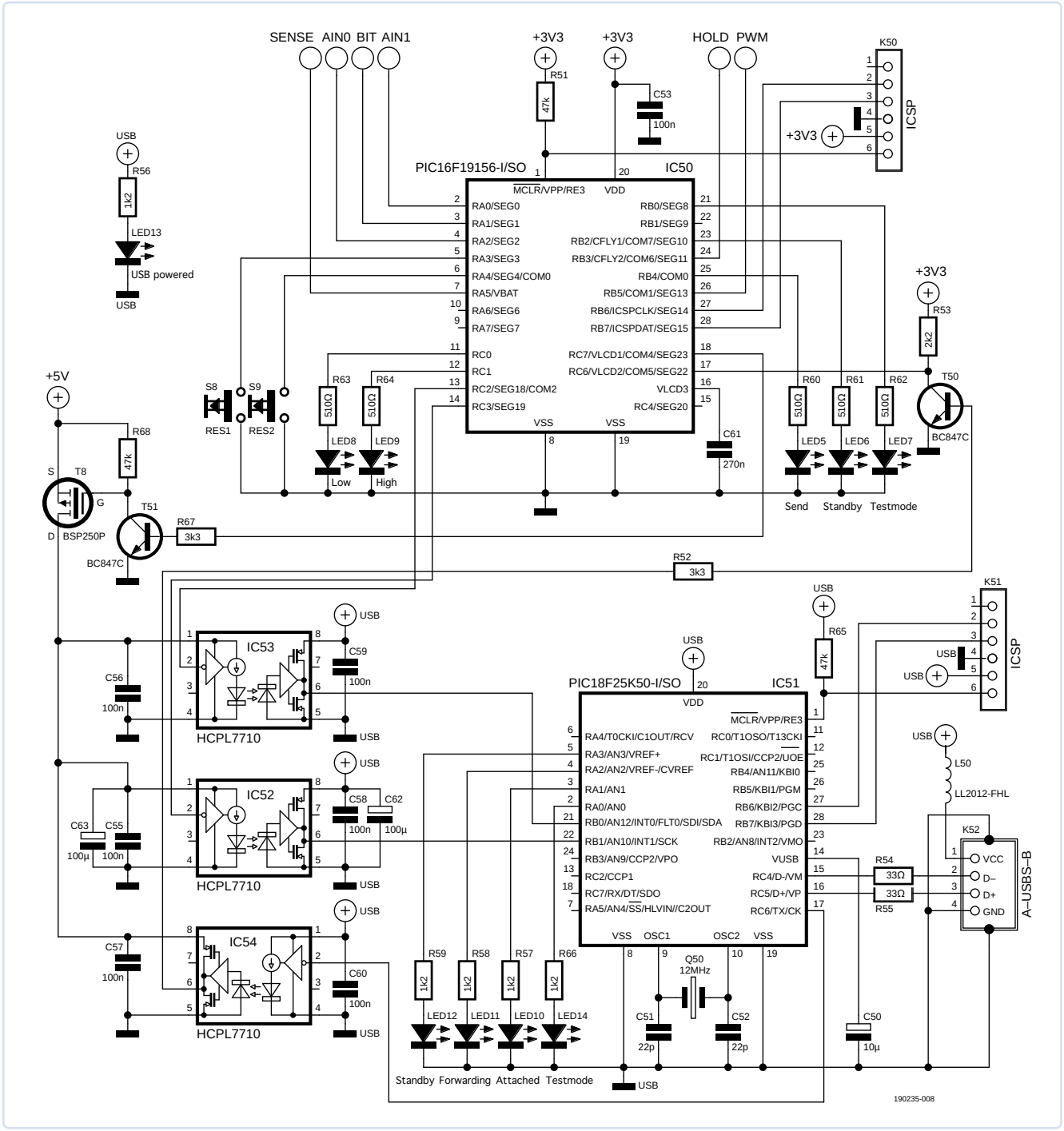


Figure 1. La partie numérique du TRNG original de 2017 avec son processeur ATtiny2313A a été remplacée par deux microcontrôleurs PIC. Le port USB remplace l'emplacement pour carte SD.

Des question, des commentaires ?

Contactez Elektor redaction@elektor.fr.

Produits

- **Livre en anglais « *Electronic Security and Espionage* », L. Matic (Elektor 2021, SKU 19903)**
  www.elektor.fr/electronic-security-and-espionage

LIENS

[1] L. Matic, générateur de nombres réellement aléatoires, Elektor 3/2017 : https://www.elektormagazine.fr/magazine/elektor-201703/40251
[2] Ce project sur Elektor Labs : https://www.elektormagazine.fr/labs/usb-random-number-generator

vitamines audio pour **micro ramollo**

Comment gonfler soi-même le niveau de sortie d'un microphone

Thomas Scherer

L'année 2020 restera dans les mémoires comme l'année « Corona », mais aussi comme celle de la généralisation soudaine des réunions en ligne, avec souvent une vidéo de piètre qualité. Passe encore que l'image soit en dessous de tout, problèmes de bande passante et tout, mais pourquoi devrait-il en être de même pour la qualité du son ?

Les mots *Skype*, *Zoom, Teams* et autres sont entrés dans le langage courant et beaucoup d'entre nous déplorent les faiblesses de ces services de vidéoconférence. Il y a bien sûr les maladresses de certains participants, mais aussi les limites de la bande passante des connexions Internet. Celles-ci se traduisent par une détérioration des images vidéo déjà médiocres à cause de mauvaises caméras ou d'un éclairage catastrophique. Que faire pour éviter que la qualité de la transmission audio s'ajoute à la liste de ces désagréments ? Notamment pour ceux qui voyagent avec un ordinateur portable. Sur un PC de bureau, vous pouvez choisir votre matériel. Sur un portable, webcam et micros sont intégrés.

Figure 1. Le microphone proprement dit est fourni avec une housse en mousse, une pince (y compris le trépied) et un câble USB.

Frustration des ordinateurs portables

J'utilise fréquemment la vidéotéléphonie depuis longtemps, et souvent un ordinateur portable quand je suis en déplacement. J'ai donc beaucoup maudit leur mauvaise vidéo.

Après avoir échangé l'année dernière mon vieux MacBook Pro contre un MacBook Air, j'ai eu du mal à en croire mes yeux devant l'image incroyablement bruitée et sous-exposée. Comment Apple ose-t-il ? J'aurais pu acheter une webcam USB extérieure, mais si je venais d'acheter un appareil ***très*** portable, c'était précisément pour ne ***pas*** avoir à transporter un sac à dos plein d'accessoires, non ? En plus, l'interface d'un MacBook Air, c'est USB-C, ce qui est très bien à ceci près que les webcams avec USB-C sont rares. Je suis maso d'accord, mais l'adaptateur USB-C/USB-A, ça dépasse ma capacité à souffrir. J'ai revendu le nouvel ordinateur et suis revenu à l'ancien modèle.

Ne ricanez pas, le problème des caméras et des micros de piètre qualité concerne aussi d'autres fabricants de portables. Ne vous croyez pas à l'abri derrière Windows 10. Je connais au moins un portable Samsung (cher) dont l'image est horriblement bruitée et le son affreux. Oui, il y a toutes sortes d'ordinateurs portables dont la prestation vidéo est médiocre. Dans les bancs d'essai, ces performances sont soigneusement négligées. La pertinence de la qualité des caméras dont les fabricants équipent leurs appareils devrait désormais être évidente pour tous. Comment se fait-il que le moindre téléphone tactile bon marché soit doté d'une bien meilleure caméra et d'un microphone utilisables, ce qui n'est toujours pas le cas de bien des ordinateurs portables ?

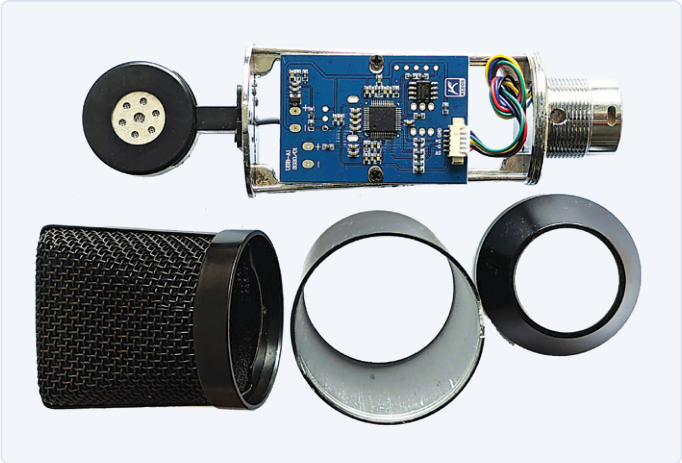


Figure 2. Le nouveau microphone démonté en ses différentes parties.

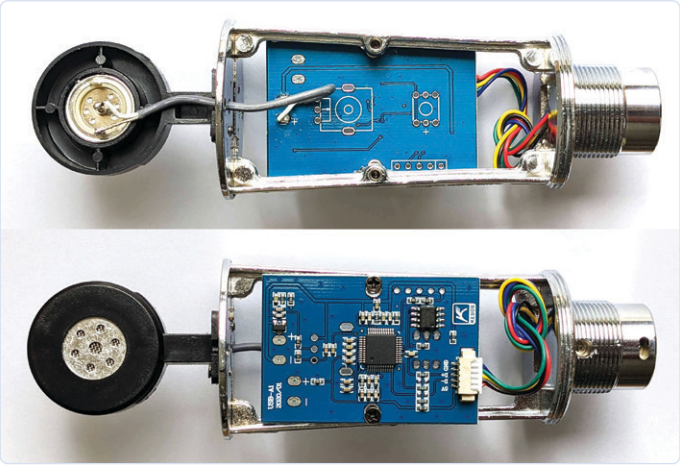


Figure 3. L'intérieur du microphone vu de devant et derrière.

Remède

Le remède le plus simple est d'acheter une webcam. Sur un PC, il n'y a pas le choix. Réfléchissez-y bien avant l'achat d'un ordinateur portable. Toutes les variantes chinoises très bon marché ne sont pas mauvaises, mais Logitech, Microsoft et d'autres fabricants connus proposent des webcams potables à partir de 35 €. Sans être parfait, le micro intégré suffit généralement pour la voix.

Si la qualité de la caméra est suffisante, mais pas la qualité du son, vous pouvez envisager d'acheter un micro externe. On trouve désormais plusieurs modèles avec USB au lieu de la prise jack. Grâce au convertisseur A/N intégré et à l'incorporation des pilotes nécessaires dans tous les systèmes d'exploitation courants, la connexion de tels appareils à un PC sous Windows, MacOS ou Linux ne pose aucun problème. Vous pouvez également utiliser un micro classique, à sortie analogique, du moins sur la plupart des PC de bureau. Sur les ordinateurs portables récents, l'entrée pour micro tend à disparaître. Le convertisseur audio USB bon marché est un autre choix possible. Il offre une telle entrée et, généralement, une sortie audio. Mais pour moi ce n'est pas la solution.

Microphone USB

Peu satisfait par divers microphones bon marché essayés ces dernières années, j'ai récemment commandé un grand micro à diaphragme avec prise USB. Ce genre de micro à condensateur est courant dans les studios d'enregistrement. On trouve sur l'internet des informations sur les caractéristiques des microphones, leur choix et l'art de les disposer pour l'enregistrement de diverses sources sonores (voir [1], article intéressant, en allemand seulement).

Du fait de l'énorme croissance des réseaux sociaux, de nombreuses jeunes personnes produisent des podcasts et gèrent activement leur propre chaîne vidéo. Il faut pour cela non seulement une bonne caméra vidéo mais aussi un bon micro. C'est précisément dans ce but que sont proposés des « kits de podcasteur », composés d'un grand microphone à diaphragme (fourni avec parfois une araignée), d'un support, d'une perche et d'un filtre anti-pop.

Je voulais tenter ma chance. Dans le matériel audio, tous les arguments avancés ne sont pas rationnels et la qualité devient rapidement très chère. Malheureusement, le rapport prix/performance n'est pas une ligne droite à pente douce, mais plutôt une courbe exponentielle. Et, comme mon budget son est une somme modeste et limitée, j'ai cherché des micros USB à grande membrane bon marché… et j'ai trouvé ce que je cherchais.

CAD Audio U29

Ce micro est livré avec un trépied et un filtre anti-pop, et ne coûte que 34 €. Je ne pouvais pas me fourvoyer en l'achetant, me suis-je dit. Je l'ai donc commandé et deux jours plus tard j'avais sous les yeux ce que montre la **fig. 1**. Le micro fait bonne impression : pas de plastique - tout en métal.

Je l'ai immédiatement connecté à mon PC, j'ai lancé le programme Audacity [2] pour faire un enregistrement et le comparer au micro de ma webcam (Logitech C525). Je n'ai pas été déçu. L'amélioration du son était sensible, tellement évidente même que je n'ai pas éprouvé le besoin de faire d'autres mesures. Bien !

Mes interlocuteurs sur Skype remarqueront-ils la différence ? Oui, la différence n'est pas passée inaperçue chez deux de mes correspondants. Le registre grave du nouveau micro CAD Audio est plus aéré, les basses sont moins grasses. Dans le registre aigu aussi, les timbres sont beaucoup plus clairs.

Niveau riquiqui

L'histoire aurait pu s'arrêter là quand, pendant les enregistrements, j'ai remarqué que le niveau du micro de CAD Audio était à 5 dB en dessous de celui de la webcam. Pour retrouver un volume équivalent, il aurait fallu pousser à fond le curseur de sensibilité du programme de vidéoconférence.

Ce n'était pas la première fois que je constatais ce phénomène. J'ai souvent installé des amplificateurs dans des microphones ou augmenté le gain de l'électronique existante en changeant des composants. Cela devrait être possible avec ce microphone aussi, n'est-ce pas ?

Aussitôt dit, aussitôt fait, je démonte le micro **(fig. 2).** Voyez la capsule du micro dont le diaphragme, d'un diamètre d'environ 2,5 cm, me paraît bien petit si je repense au qualificatif de « grand diaphragme ». La capsule (**fig. 3**) est riquiqui aussi.

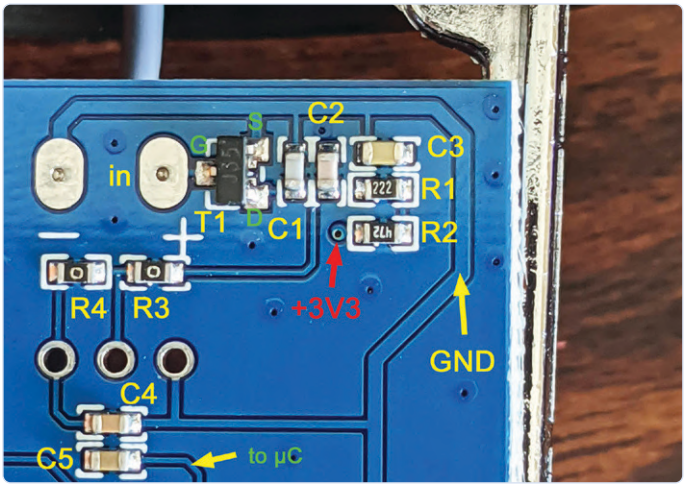


Figure 4. Détail du circuit du convertisseur d'impédance avec annotations.

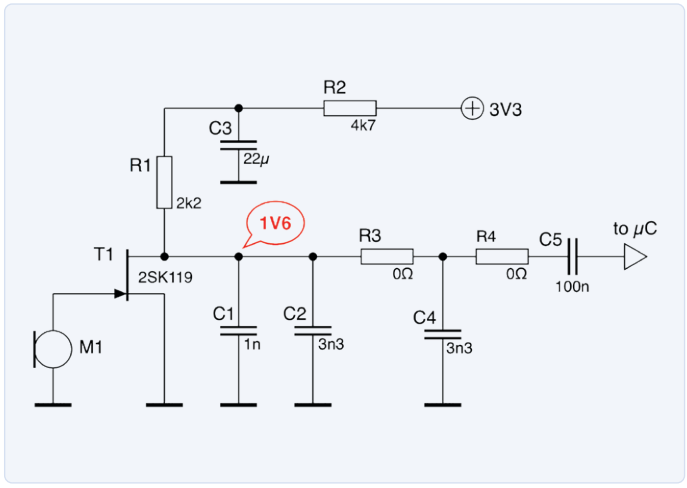


Figure 5. Le circuit du convertisseur d'impédance.

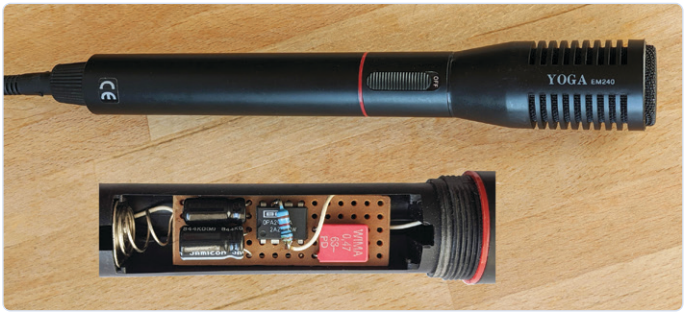


Figure 6. Le microphone bon marché Yoga EM240 avec préamplificateur intégré.

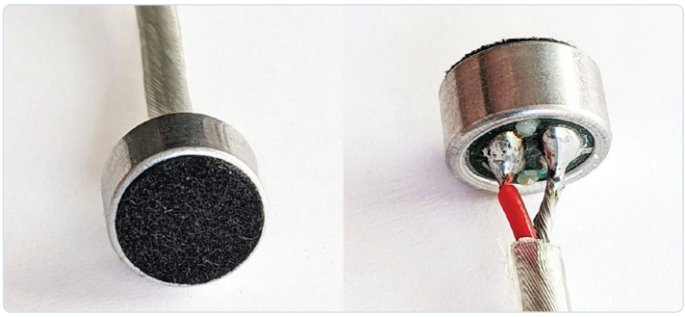


Figure 7. Petite capsule d'électret avec membrane de 5 mm.

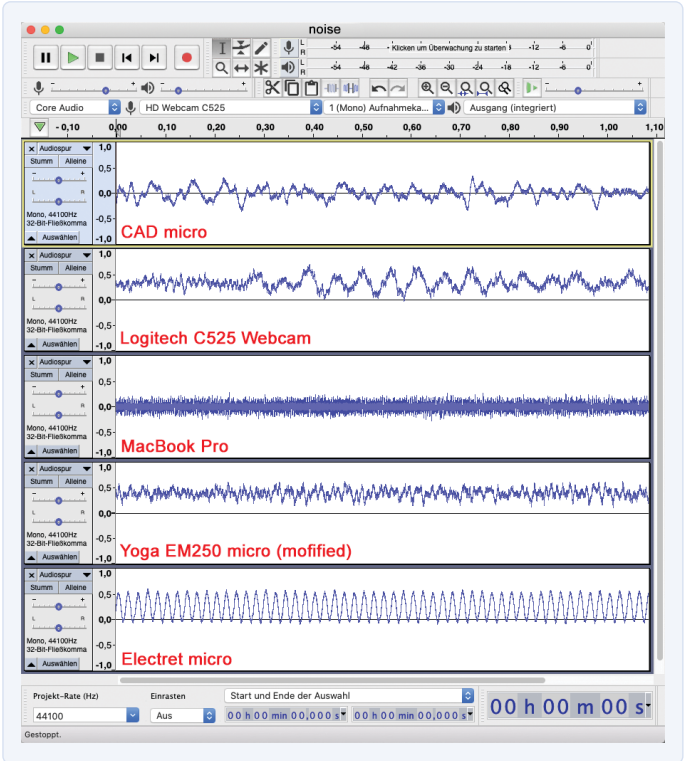


Figure 8. Niveaux de bruit des différents microphones testés.

Le circuit imprimé est intéressant. La puce au milieu est un SoC avec entrée audio et interface USB. Contrairement à la mémoire EEPROM en bas à droite, le SoC n'est pas étiqueté, mais le véritable préampli, c'est le petit CMS noir à trois pattes, en haut.

Une première mesure sur les deux surfaces « – » et « + » au milieu en haut révèle qu'avec exactement 0,0 V, aucune tension continue n'est appliquée à la membrane du micro. Donc, pas de polarisation. Il devrait donc s'agir d'un micro à électret. Me voilà un peu déçu, en dépit du bon son. Le CMS porte la référence « J35 », et c'est un JFET à canal N de type 2SK1109, avec environ 200 µA de courant de drain et une grille court-circuitée.

Convertisseur d'impédance

La figure 4 montre une partie du circuit imprimé autour de ce JFET. Une conversion d'impédance est nécessaire, car la capacité de la membrane du microphone est très faible et son impédance pour les signaux audio est très élevée. Le gain introduit par T1 permet d'obtenir un niveau de sortie plus élevé. La **figure 5** donne le schéma du circuit de la **figure 4**. R2 est utilisée avec C3 pour lisser la tension d'alimentation. R1 et R2 ensemble chutent d'environ 1,7 V ; le courant de drain est donc d'environ 245 µA.

Ce qui est remarquable, c'est que nous pouvons permuter R1 et R2 sans mettre en danger le réglage du courant continu, mais obtenir en même temps que le gain double (≈ +6 dB). Cela ne pose aucun

problème, puisque le filtre passe-bas formé par la résistance 2,2 kΩ et le condensateur 22 µF se situe à une fréquence respectable de 3,2 Hz. Cependant, le filtre passe-bas formé par R1 = 4,7 kΩ et C1 + C2 + C4 se situe alors en dessous de 5 kHz. Ce qui peut être modifié en supprimant C2 ou C4.

Résultats

Après modification, le niveau est légèrement supérieur à celui de ma webcam sans que la qualité sonore soit affectée. Une modification qui s'est donc bien passée.

Pour comparer le son du microphone CAO modifié à celui d'autres microphones, j'ai fait des enregistrements : le nouveau micro, ma webcam, mon MacBook Pro, deux vieux micros bon marché (**fig. 6)** et une capsule nue de microphone à électret bon marché (**fig. 7**). Sans surprise, c'est le MacBook pro qui (subjectivement) sonnait le moins bien. Le déficit dans le grave donne de ma voix plutôt sonore une impression de voix fluette. Il y a quelques années, j'avais construit un préampli à faible bruit pour le micro « Yoga » que je trouvais trop bruyant malgré son faible niveau de sortie.

Pour vous donner une idée des résultats, j'ai préparé des échantillons MP3 à télécharger sur le site d'Elektor [3]. Chaque fichier contient une courte phrase suivie d'une seconde de bruit blanc, tous réglés au même niveau, de sorte que seul le timbre diffère. La **figure 8** montre les cinq niveaux de bruit différents, chacun étant amplifié de 30 à 40 dB. Pour les micros de CAD Audio, Logitech et Yoga on peut parler simplement de bruit, mais avec le Logitech on entend aussi un bourdonnement superposé, sur le MacBook Pro un résidu de bruit de haute fréquence, et avec la capsule à électret on distingue un ronflement (secteur).

Conclusion

Suis-je satisfait du son du microphone de CAD ? Ma réponse pourrait venir de Radio Erevan [4] : En principe oui, mais…

On m'a vendu un petit micro à membrane comme un grand. Il est bon marché et en fait meilleur que le micro de la webcam, et c'est toujours ça.

Tout est une question d'angle, comme ce micro d'ailleurs : vous remarquerez qu'il n'est pas utilisé de face mais de côté, comme le montre la **fig. 9**. C'est sa caractéristique directionnelle qui veut ça.

200434-04 — VF : Marguerite Gerné

Figure 9. Voici comment le microphone CAD Audio doit être positionné : à droite (ou à gauche) du moniteur.

Des questions, des commentaires ?

Adressez vos questions ou vos commentaires sur cet article à redaction@elektor.fr

Produits

- **Les oscilloscopes anciens & modernes pour les débutants**
  www.elektor.fr/19124
- **Préamplificateur simple pour micro professionnel - circuit imprimé nu**
  www.elektor.fr/18096

LIENS

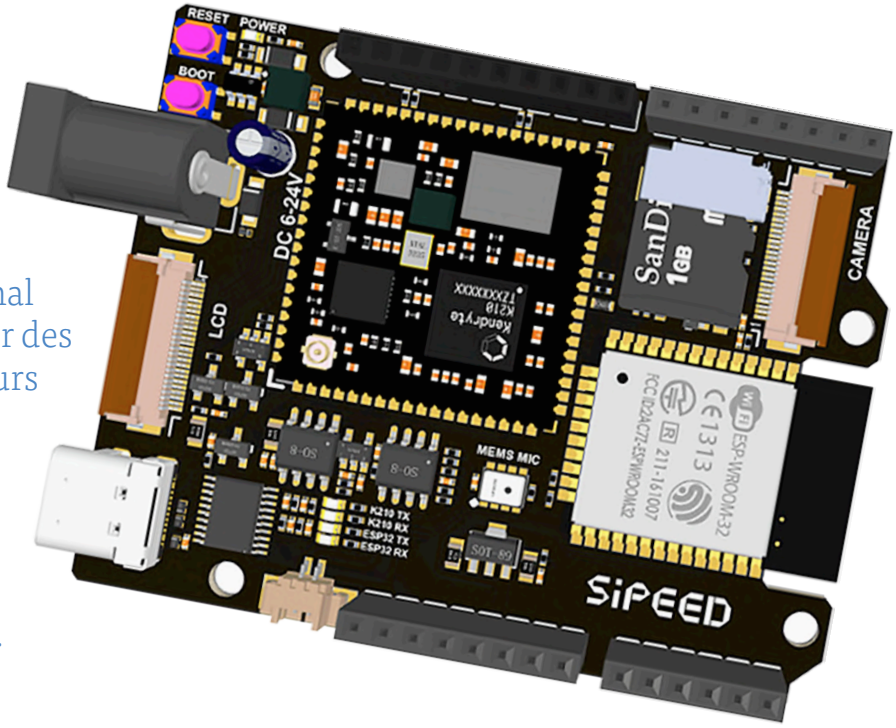
[1] Wikipedia: Mikrofonierung: https://de.wikipedia.org/wiki/Mikrofonierung
[2] Audacity: http://www.audacity.de
[3] Échantillons audio : http://www.elektormagazine.fr/200434-03
[4] Blagues sur Radio Erevan : https://fr.wikipedia.org/wiki/Radio_Erevan
Mike Costa, « préamplificateur simple pour micro pro », Elektor 5/2017 : https://www.elektormagazine.fr/magazine/elektor-201705/40352

FFT avec Maixduino

Acquisition de spectres de fréquences

Tam Hanna (Slovénie)

La transformée rapide de Fourier est intéressante, mais pas seulement dans le domaine audio : pour l'analyse du bruit d'une turbine ou des niveaux de tension de véhicules mal amortis, la FFT permet souvent de tirer des conclusions sur les régimes des moteurs et autres. La carte Maixduino avec le K210-SoC disponible chez Elektor se prête non seulement à accompagner vos premiers pas dans l'univers des réseaux neuronaux, mais aussi pour un accès à la FFT rapide en temps réel.



Les progrès des logiciels et du matériel mettent à la disposition de monsieur Tout-le-Monde de techniques auparavant réservées aux milices privées et aux gouvernements.

Sipeed, spécialiste en semi-conducteurs connu jusqu'à présent surtout pour ses expériences avec le processeur RISC-V, propose une nouvelle carte avec le Maixduino. Celui-ci réunit un ESP32 pour la communication en réseau, et un SoC appelé *K210 Kendryte*, que le fabricant a l'intention d'utiliser dans les applications d'IA. Avec Google vous récolterez de nombreuses informations si vous recherchez «kendryte + problème».

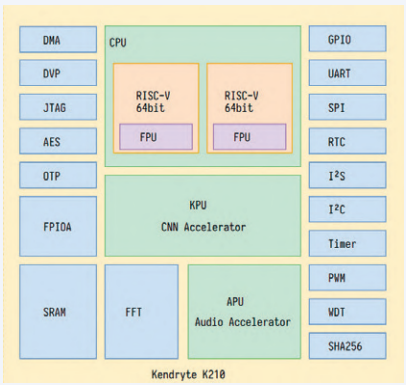


Figure 1. Le K210 est livré avec quelques extensions (source : Canaan Inc.).

Pour en savoir plus sur ce SoC, visitez le site [1] où il faut parfois faire preuve de patience. La fiche technique du K210 contient principalement une description des broches. Examinez le schéma (**fig. 1**), il est également intéressant.

Le processeur principal (CPU) adopte l'architecture RISC-V (avec les options I, M, A et C) à source ouverte, avec deux auxiliaires : d'abord la KPU, derrière laquelle se trouve un processeur de réseau neuronal (sans autre information dans la fiche technique), qui met en œuvre un réseau neuronal conservé dans le matériel, semblable au processeur *Tristar* de DPO utilisé autrefois par *Danaher* dans les TDS 5xxD et TDS 7xxD.

L'autre, un processeur audio appelé APU, traite les informations audio jusqu'à 192 kHz, stocke ces informations par DMA directement dans la mémoire principale et peut également appliquer automatiquement des traitements FFT. Nous utiliserons cet APU dans les étapes suivantes pour réaliser une petite FFT.

Mentionnons encore divers périphériques sur lesquels nous ne nous attarderons pas ici.

Une question d'outils

Ce qui importe pour nous, ce sont les outils de développement. Au niveau le plus bas se trouve le SDK autonome téléchargeable [2] – une bibliothèque (heureusement) en C, qui nous permet d'interagir directement avec les primitives du processeur.

Un niveau au-dessus, c'est ArduinoCore [3], une infrastructure pour l'EDI Arduino, également utilisable avec Platform.IO. Enfin, il y a MaixPy, un environnement de programmation dérivé de MicroPython, réservé

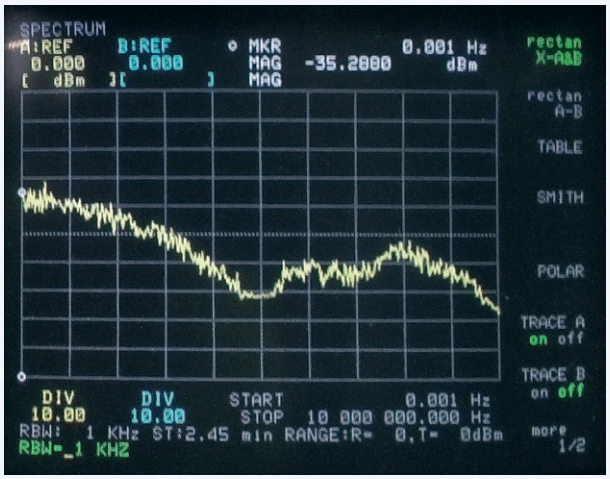


Figure 2. Les bandes étroites offrent une haute résolution mais imposent un temps de balayage plus long (Sweep Time)...

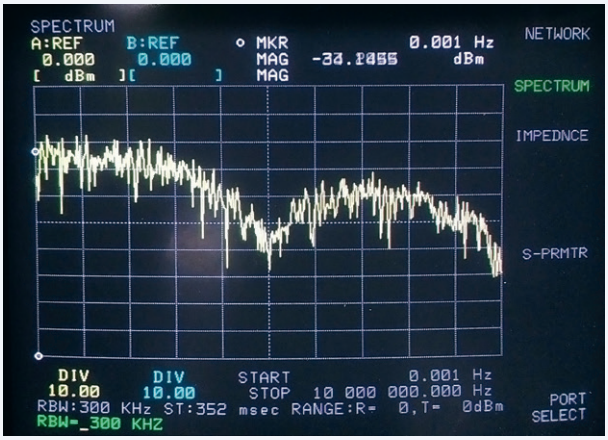


Figure 3. ...alors que des bandes plus larges et à balayage plus rapide fournissent des informations moins précises.

à ceux qui ont beaucoup d'expérience avec Python (lisez l'encadré !). L'étape suivante consiste à réfléchir à la manière de contruire les tampons à traiter. Dans la fiche technique déjà mentionnée, nous découvrons que le Kendryte ne fonctionne pas avec des tailles arbitraires. Pour utiliser l'accélération matérielle, il faut choisir l'une des tailles de tampon suivantes : 64, 128, 256 ou 512.
Lorsqu'on fait de la FFT, il est raisonnable d'avoir à portée de main un analyseur de spectre classique. Dans notre cas, la taille du *godet* est pertinente, car elle décrit la finesse des bandes de fréquences résultantes.

Les **figures 2** et **3** viennent du célèbre HP4195A et illustrent le principe. Alors que notre analyseur de spectre analogique affiche, pour cause de bruit, des bandes instables dans les deux cas, la courbe réelle est plus régulière avec la FFT. La résolution en fréquence découle du rapport *fréquence d'échantillonnage / taille de la FFT* ; si 1024 échantillons sont prélevés à 8192 Hz, la largeur de bande de résolution est RBW = 8 Hz. Comme c'est si souvent le cas, avec la FFT non plus, la maximisation n'est pas tout. La corrélation entre nombre de bandes de fréquences et nombre d'échantillons entrants est en principe le facteur 2 ; cependant, c'est plus difficile en réalité, car la complexité des algorithmes FFT augmente rapidement avec le nombre d'échantillons. La **figure 4** montre cette croissance problématique qui, dans le monde réel, implique une limitation de la précision de notre FFT.
À l'étape suivante, nous définirons un groupe de constantes. En pratique, spécifiez la longueur de la FFT au moyen d'un `#define` pour faciliter les ajustements ultérieurs entre précision et vitesse :

```
uint32_t rx_buf[1024];
#define FFT_N                512U
#define FFT_FORWARD_SHIFT    0x0U
#define SAMPLE_RATE          38640
#define FRAME_LEN            512
```

Le stockage effectif des informations fournies par le DMA nécessite un groupe de champs de mémoire supplémentaires :

```
 nt16_t  real[FFT_N];
int16_t  imag[FFT_N];
int    hard_power[FFT_N];
uint64_t fft_out_data[FFT_N / 2];
uint64_t buffer_input[FFT_N];
uint64_t buffer_output[FFT_N];
fft_data_t *output_data;
fft_data_t *input_data;
complex_hard_t data_hard[FFT_N] = {0};
```

Initialisation du matériel

Le fabricant du Maixduino nous gratifie d'un microphone MEMS MSM261S4030H0. Ces petits microphones sont intéressants (**fig. 5**). L'affectation des broches révèle que le composant s'occupe de l'adaptation du signal et fournit les données sur un bus série.
Le premier obstacle est que le logiciel n'initialise pas automatiquement les lignes de données nécessaires à la communication avec le microphone. Il est facile d'y remédier au début de la configuration :

```
void setup()
{
    lcd.begin(15000000, COLOR_RED);
    fpioa_set_function(20, FUNC_I2S0_IN_D0);
    fpioa_set_function(18, FUNC_I2S0_SCLK);
    fpioa_set_function(19, FUNC_I2S0_WS);
```

L'étape suivante consiste à configurer le matériel I2S. Le matériel de la carte détermine généralement le code ; un réglage crucial est le taux d'échantillonnage à définir avec la méthode `i2s_set_sample_rate`. Comme les options sont réduites du fait de la taille de la FFT, le taux

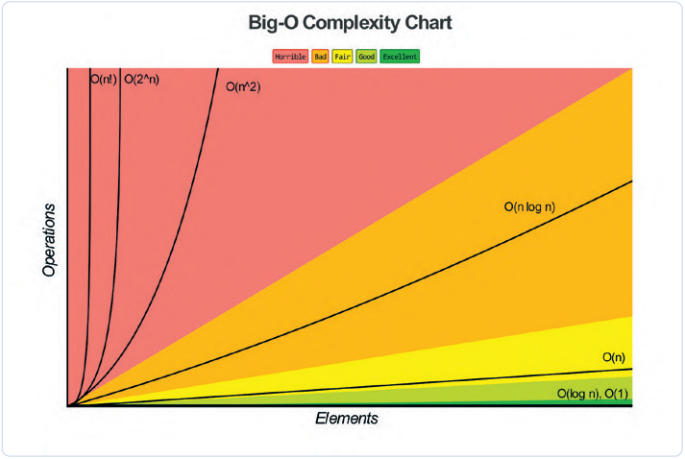
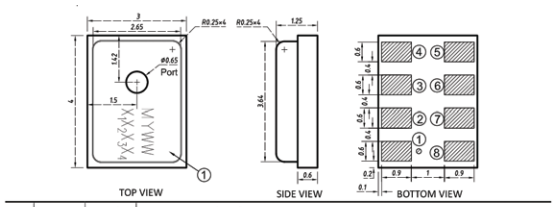


Figure 4. Pour obtenir des bandes de fréquences infiniment petites, vous avez besoin de beaucoup de puissance de calcul. (Photo : https://www.bigocheatsheet.com/).



1	GND	Ground	Connect to ground on the PCB.
2	N/C	—	Do not connect
3	WS	Input	Serial Data-Word Select for I²S Interface.
4	CHIPEN	Input	Microphone Enable. When set low (ground), the microphone is disabled and put in power-down mode. When set high (VDD), the microphone is enabled.
5	L/R	Input	Left/Right Channel Select. When set low, the microphone outputs its signal in the left channel of the I²S frame; when set high, the microphone outputs its signal in the right channel.
6	SCK	Input	Serial Data Clock for I²S Interface.
7	SD	Output	Serial Data Output for I²S Interface. This pin tristates when not actively driving the appropriate output channel. The SD trace should have a 100 KΩ pull-down resistor to discharge the line during the time that all microphones on the bus have tristated their outputs.
8	VDD	Power	1.8 to 3.3 V.

Figure 5. Avec MEMSensing, l'utilisateur du capteur a beaucoup moins de travail (source : MEMSensing Microsystems).

d'échantillonnage sera l'un des rares moyens de «traiter» la largeur de la bande de fréquences :

```
    i2s_init(I2S_DEVICE_0, I2S_RECEIVER, 0x3);
    i2s_set_sample_rate(I2S_DEVICE_0, SAMPLE_RATE );
    i2s_rx_channel_config(I2S_DEVICE_0, I2S_CHANNEL_0,
                          RESOLUTION_16_BIT, SCLK_CYCLES_32,
                          TRIGGER_LEVEL_4, STANDARD_MODE);
```

Il ne reste que quelques initialisations :

```
    dmac_init();
    sysctl_enable_irq();
}
```

La prochaine tâche est l'opération FFT proprement dite. Il faut s'assurer que les informations fournies par I2S sont stockées en mémoire :

```
void loop()
{
  int i, sampleID;

  i2s_receive_data_dma(I2S_DEVICE_0, &rx_buf[0],
   FRAME_LEN * 2, DMAC_CHANNEL3);
  dmac_wait_idle(DMAC_CHANNEL3);
```

En appelant `dmac_wait_idle`, nous imposons une pause au processeur jusqu'à ce que le canal DMA se libère. Cela garantit que chaque `boucle` fonctionne avec une base de données cohérente et nouvelle. Si (pour voir) vous supprimez (provisoirement) l'appel à `dmac_wait_idle`, vous constaterez d'étranges artefacts (*aliasing*) à l'écran. C'est dû probablement au fait que, selon la fiche technique, les modules FFT ne peuvent être utilisés que via le moteur DMA. En cas de collision, la catastrophe est imminente !
Lorsque vous travaillez avec les moteurs FFT ou des unités fonctionnelles fixes similaires, vous devez toujours tenir compte du fait que le matériel attend des informations dans un format de stockage prescrit. Dans le cas de `tag_fft_data`, il s'agit de la structure suivante, déclarée à plusieurs endroits dans le SDK :

```
typedef struct tag_fft_data
{
    int16_t  I1;
    int16_t  R1;
    int16_t  I2;
    int16_t  R2;
} fft_data_t
```

À ce stade, le lecteur attentif se demandera pourquoi on attend deux composantes appelées R et I. Rappelons que R signifie réel et I imaginaire.
La transformation de Fourier est définie en mathématiques classiques, c'est-à-dire symboliques, comme une opération basée sur des nombres complexes. Les algorithmes FFT attendent donc qu'on leur fournisse des composantes réelles et imaginaires, du moins dans la version générique, que les développeurs de matériel aiment intégrer dans le matériel pour obtenir une flexibilité maximale. Or, comme nos informations sonores comportent des éléments exclusivement réels, à ce stade nous mettons toujours à zéro les valeurs de I :

```
    for ( i = 0; i < FFT_N / 2; ++i)
    {
        input_data = (fft_data_t *)&buffer_input[i];
        input_data->R1 = rx_buf[2*i];
        input_data->I1 = 0;
        input_data->R2 = rx_buf[2*i+1];
        input_data->I2 = 0;
    }
```

C'est là que le «recyclage» du processus FFT devient pratique. Le classique *Numerical Recipies in C* [5], PDF disponible en ligne gratuitement ou à un prix modique, propose dans le douzième chapitre (de la deuxième édition) quelques formules intéressantes, mais qui requièrent de solides connaissances en maths. La récompense serait un processus FFT qui traiterait deux fonctions réelles en même temps. Il ne reste plus au développeur qu'à séparer les informations de sortie, ce qui n'est pas difficile.
La FFT est donc un de ces domaines où l'électronicien peut s'en donner à cœur joie – en fait, cette approche pratique est souvent plus efficace que l'épuisant effort d'apprentissage des maths.

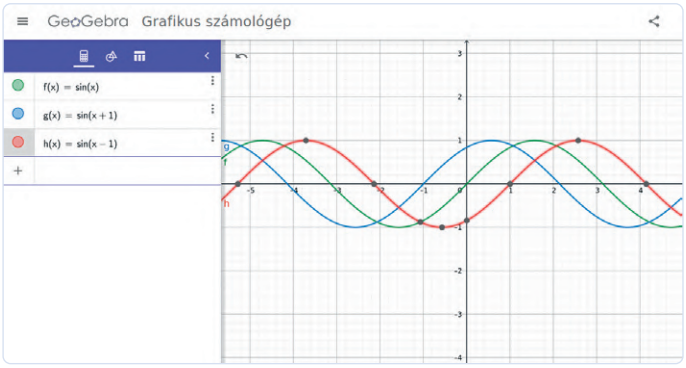


Figure 6. Malgré une fréquence et une amplitude identiques, ces oscillations ne sont pas identiques.

La prochaine étape consiste à exécuter les commandes FFT pour obtenir ensuite les résultats :

Notez ici le passage de l'opération `FFT_DIR_FORWARD`, qui informe le moteur matériel que nous passons du domaine temporel au domaine fréquentiel. En pratique, il y aura toujours des situations dans lesquelles vous aurez à transférer un signal du domaine des fréquences dans le domaine temporel. Dans ce cas, c'est la constante `FFT_DIR_BACKWARD` qui serait passée ici. Dans tous les cas, les données sont récoltées ensuite :

```
for ( i = 0; i < FFT_N / 2; i++)
{
    output_data = (fft_data_t*)&buffer_output[i];
    data_hard[2 * i].imag = output_data->I1 ;
    data_hard[2 * i].real = output_data->R1 ;
    data_hard[2 * i + 1].imag = output_data->I2 ;
    data_hard[2 * i + 1].real = output_data->R2 ;
}
```

Les informations renvoyées ont une partie réelle et une partie imaginaire. Pour comprendre ce problème, revenons encore une fois (brièvement) aux maths. La formule

$$S_f(t) = a_0/2 \, \Sigma \, [a_n \cos(n\omega t) + b_n \sin(n\omega t)]$$

montre l'une des nombreuses représentations de la série fondamentale de Fourier. L'utilisation de I ou J dans la formule nous montre que nous travaillons avec des unités complexes – puisque la FFT n'est en fait qu'une représentation numérique de la transformée de Fourier classique, elle doit suivre les règles.

Pourquoi complexe ?

L'échantillonnage effectué est certes réel, mais il ne fait pas le bonheur de la mathématicienne. L'une des raisons en est qu'il n'existe pas de déclencheur «défini» – du point de vue de la mathématicienne, les signaux sinusoïdaux de la **fig. 6** sont déphasés. L'opération FFT exprime cela par le fait que les valeurs renvoyées représentent des nombres complexes. Cela permet d'exprimer à la fois la puissance

et le déphasage de la composante respective – les deux sinusoïdes seraient également différentes dans le résultat.
L'électronicien préfère généralement ignorer les composantes complexes, que nous détruisons donc en calculant la valeur absolue :

```
float pmax=10;
for (i = 0; i < FFT_N; i++)
{
   hard_power[i] = sqrt(data_hard[i].real * data_
   hard[i].real + data_hard[i].imag * data_hard[i].
   imag);
```

Simplifions logarithmes et racines carrées, de façon à optimiser les calculs. Simplifier n'implique pas pour autant des procédures plus faciles à comprendre, au contraire. Par souci de clarté, nous renonçons à la simplification ici.
Si nous effectuons la combinaison déjà mentionnée en une opération FFT de deux fonctions différentes, il faudrait à ce stade effectuer une extraction et séparer les deux résultats (par un algorithme simple comparé à la FFT). La prochaine étape est une logarithmisation pour obtenir des valeurs de l'ordre du dB :

```
    hard_power[i] = 20*log10(2*hard_power[i]/FFT_N);
    if( hard_power[i]>pmax && i>5)
        pmax = hard_power[i];
}
```

Puis il faut afficher les valeurs. Le principal inconvénient est ici l'API d'affichage de l'Arduino, et les lourdes pertes de performance lors de la transmission de valeurs de coordonnées *invalides* :

```
    lcd.setRotation(0);
    lcd.fillScreen(COLOR_WHITE);
    for (int i=0;i<256;i++)
    {
      int dval = 240 - 240*(hard_power[i]/pmax);
      if (dval<2)dval=1;
      if (dval>238)dval=238;
      lcd.drawLine( i,  240, i, dval  , COLOR_RED);
    }
    delay(50);
}
```

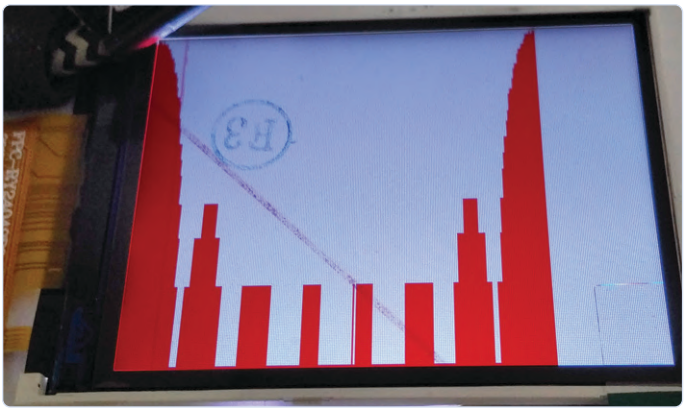


Figure 7. Cette «symétrie» est déroutante.

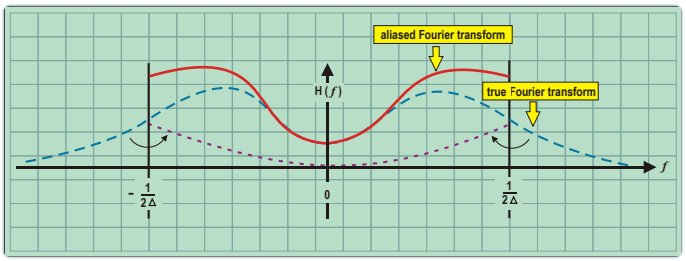


Figure 8. Les éléments spectraux «éliminés» de la fenêtre de Nyquist apparaissent comme des artefacts.

Nous voici prêts pour un premier essai (**fig. 7**). Ceux qui le souhaitent peuvent à ce stade ajouter au système des tonalités de test. La récompense de cet effort est la possibilité de déplacer sur l'écran la tonalité de test ou la ligne qui la représente.

De la réflexion

Le FFT est un sujet parmi les plus chronophages pour l'électronicien. Les observateurs attentifs remarqueront que le signal de test parcourt l'écran en partant aussi bien de gauche que de droite.
Plusieurs obstacles nous attendent. Notre FFT se répète donc dès le 128e champ, alors que, selon les théorèmes qui viennent d'être établis, cela devrait ne se produire qu'au 256e champ. La raison en

IL Y A PYTHON ET PYTHON

Dans le domaine de Python pour microcontrôleurs, deux concepts sont en compétition. L'un est MicroPython, qui met en œuvre un *runtime* complet. Le système cible contient un interpréteur Python complet, qui charge même des bibliothèques depuis l'internet, comme sur une station de travail. Cette approche a l'avantage d'une plus grande flexibilité, l'inconvénient de problèmes causés par des ressources limitées (RAM).

L'autre n'est actuellement représentée que par Zerynth, indisponible sur Maixduino. Le poste de travail crée un monolithe qui est transféré sur le système cible. À la flexibilité moindre s'oppose une meilleure stabilité – si les bibliothèques sont sur le poste de travail, les processus gourmands seront traités en dehors du microcontrôleur.

est un problème de livraison des échantillons via le bus I2S, un sujet que nous laisserons de côté maintenant. Vous trouverez sous [6] une version modifiée qui *atténue* ce problème.
Insistons sur le choix du verbe *atténuer*, car il y aura toujours une *réflexion* dans le cas d'une FFT avec des valeurs d'entrée entièrement réelles. Les valeurs de zéro à N/2 correspondent à celles de N/2+1 à N. La raison en est que les parties complexes de la FFT doivent s'annuler mutuellement.
Pour d'autres recours à la FFT, les domaines sont nombreux, par exemple l'introduction d'une fonction de fenêtre pour «découper» l'entrée. L'idée sous-jacente est que les parties situées à l'extérieur de la fenêtre de Nyquist peuvent être repliées ou réfléchies vers l'intérieur de part et d'autre, alors qu'une fonction de fenêtre supprimerait ce phénomène (**fig. 8**). Je recommande à ce sujet la lecture de *Some Windows with Very Good Sidelobe Behavior* [7], publié par Albert H. Nuttall en 1981.

Conclusion

Le moteur FFT matériel permet des expériences assez accessibles avec des algorithmes et des applications FFT. Ce ne sont pas les applications qui manquent. Vous ne devriez regretter ni le temps que vous aurez consacré ni l'argent que vous aurez investi.

200297-04 — VF : Denis Meyer

Produits

- **Sipeed MAix BiT Kit for RISC-V AI+IoT**
  www.elektor.fr/sipeed-maix-bit-kit-for-risc-v-ai-iot

LIENS

[1] Kendryte K210 : https://bit.ly/3fptbHu
[2] Standalone-SDK : https://github.com/kendryte/kendryte-standalone-sdk
[3] ArduinoCore-k210 : https://github.com/Seeed-Studio/ArduinoCore-k210
[4] fft_lcd : https://github.com/MrJBSwe/fft_lcd/blob/master/main.c
[5] Teukolsky, Press, Vetterling & Flannery: Numerical Recipes in C : http://www.numerical.recipes/
[6] stft_lcd : https://github.com/jpiat/stft_lcd
[7] Albert H. Nuttall : Some Windows with Very Good Sidelobe Behavior : https://bit.ly/3fTkCWb

LISTAGE 1. LISTAGE COMPLET DE MAIXDUINO.

```
#include <Sipeed_ST7789.h>
#include "i2s.h"
#include "fft.h"
SPIClass spi_(SPI0); // MUST be SPI0 for Maix series on board LCD
Sipeed_ST7789 lcd(320, 240, spi_);
uint32_t rx_buf[1024];
#define FFT_N               512U
#define FFT_FORWARD_SHIFT   0x0U
#define SAMPLE_RATE         38640
int16_t  real[FFT_N];
int16_t  imag[FFT_N];
int    hard_power[FFT_N];
uint64_t fft_out_data[FFT_N / 2];
uint64_t buffer_input[FFT_N];
uint64_t buffer_output[FFT_N];
fft_data_t *output_data;
fft_data_t *input_data;
complex_hard_t data_hard[FFT_N] = ;
#define FRAME_LEN           512

void setup()
 {
     lcd.begin(15000000, COLOR_RED);
```

suite au verso

```
    fpioa_set_function(20, FUNC_I2S0_IN_D0);
    fpioa_set_function(18, FUNC_I2S0_SCLK);
    fpioa_set_function(19, FUNC_I2S0_WS);

   i2s_init(I2S_DEVICE_0, I2S_RECEIVER, 0x3);
   i2s_set_sample_rate(I2S_DEVICE_0, SAMPLE_RATE );
   i2s_rx_channel_config(I2S_DEVICE_0, I2S_CHANNEL_0,
                         RESOLUTION_16_BIT, SCLK_CYCLES_32,
                         TRIGGER_LEVEL_4, STANDARD_MODE);

    dmac_init();
    sysctl_enable_irq();
}
void loop()
 {
   int i, sampleID;

   i2s_receive_data_dma(I2S_DEVICE_0, &rx_buf[0], FRAME_LEN * 2, DMAC_CHANNEL3);
    dmac_wait_idle(DMAC_CHANNEL3);//wait to finish recv

    for ( i = 0; i < FFT_N / 2; ++i)
    {
        input_data = (fft_data_t *)&buffer_input[i];
        input_data->R1 = rx_buf[2*i];   // data_hard[2 * i].real;
        input_data->I1 = 0;             // data_hard[2 * i].imag;
        input_data->R2 = rx_buf[2*i+1]; // data_hard[2 * i + 1].real;
        input_data->I2 = 0;             // data_hard[2 * i + 1].imag;
    }

   fft_complex_uint16_dma(DMAC_CHANNEL0, DMAC_CHANNEL1, FFT_FORWARD_SHIFT, FFT_DIR_FORWARD,
                                                        buffer_input, FFT_N, buffer_output);
   for ( i = 0; i < FFT_N / 2; i++)
    {
        output_data = (fft_data_t*)&buffer_output[i];
        data_hard[2 * i].imag = output_data->I1 ;
        data_hard[2 * i].real = output_data->R1 ;
        data_hard[2 * i + 1].imag = output_data->I2 ;
        data_hard[2 * i + 1].real = output_data->R2 ;
    }
   float pmax=10;
    for (i = 0; i < FFT_N; i++)
    {
        hard_power[i] = sqrt(data_hard[i].real * data_hard[i].real +
                                          data_hard[i].imag * data_hard[i].imag);
       //Convert to dBFS
        hard_power[i] = 20*log10(2*hard_power[i]/FFT_N);
       if( hard_power[i]>pmax && i>5)
            pmax = hard_power[i];
    }

    lcd.setRotation(0);
    lcd.fillScreen(COLOR_WHITE);
    for (int i=0;i<256;i++)
    {
      int dval = 240 - 240*(hard_power[i]/pmax);
      if (dval<2)dval=1;
      if (dval>238)dval=238;
      lcd.drawLine( i,  240, i, dval  , COLOR_RED);
    }
    delay(50);
}
```

embeddedworld2023

Exhibition&Conference

... it´s a smarter world

UNITED

JOIN THE EMBEDDED COMMUNITY

14–16.3.2023

Get your free ticket now!

embedded-world.com/voucher

Use the voucher code **GG4ew23**

Media partners

Markt&Technik

Elektronik

SmarterWorld

DESIGN& ELEKTRONIK

Elektronik automotive

medical-design

computer & automation

elektroniknet.de

NÜRNBERG MESSE

sur le vif

sans raison apparente

Connecteur Amphenol à 36 voies, adopté en 1970 pour les imprimantes 101. Source : Shutterstock/KPixMining.

Ilse Joostens (Belgique)

Lorsque j'étais adolescente, j'assemblais des circuits de façon aussi frénétique qu'irréfléchie. La façon ils fonctionnaient ne m'intéressait guère. Tout ce que je voulais, c'était obtenir un résultat, et vite. J'étais pleinement satisfaite si ce que je venais de construire fonctionnait plus ou moins. « Fini » et « plus ou moins fini » étaient chez moi synonymes.

Avec le temps, la copiste que j'étais est devenue la créatrice de ses propres circuits. Je m'étais fixé comme objectif de toujours réaliser le meilleur circuit possible, mais j'ai vite dû renoncer à ce beau principe lorsque je suis devenue professionnelle. Bien des facteurs entrent en jeu lors de la réalisation d'un projet commercial : disponibilité des composants en quantité requise, considérations financières, matériel à disposition, etc. Ces contraintes forcent parfois à prendre des décisions qui peuvent sembler illogiques à première vue.

Le fiasco de Centronics

Un de mes projets passés nécessitait un grand nombre de LED RVB. Je les avais commandées auprès d'un fournisseur allemand dont les prix étaient intéressants. Leur qualité était irréprochable, et le fournisseur en question avait même joint à la commande un lot de résistances qui, avait-il précisé, pourraient me servir de résistances-talons. C'était fort aimable de sa part, mais je n'en avais nul besoin. La plupart valaient 510 Ω, une valeur peu courante. J'ai toutefois pu les caser dans mon *shield* à VFD pour Arduino Uno (R1 sur le schéma) [1]. Les kits de ce projet se sont vendus comme des petits pains et, avant même que je le réalise, mon stock de 510 Ω était épuisé. J'étais victime de mon propre succès. Il me fallait en trouver de nouvelles, qui plus est auprès d'une source d'approvisionnement continue. Autre contrariété, à l'époque ces résistances étaient bien plus chères que des 470 Ω ou des 560 Ω. Je m'étais vraiment tiré une balle dans le pied ! La popularité du kit était telle que des versions modifiées du circuit commençaient à fleurir en ligne, dont certaines avec six tubes. Mais tout le monde reprenait plus ou moins aveuglément la résistance de 510 Ω.

Le fabricant d'imprimantes Centronics [2] s'est un jour retrouvé dans une situation semblable. C'était en 1970, à l'époque où l'entreprise venait de mettre au point une imprimante matricielle bon marché, le modèle 101 [3]. Wang Laboratories, la société mère de Centronics, disposait

Guglielmo Marconi. Source : Shutterstock/Morphart Creation.

alors d'un stock excédentaire de 20 000 connecteurs Amphenol à micro-ruban de 36 contacts. Ils étaient destinés à l'une de leurs premières calculatrices, mais décision fut prise de les utiliser pour l'interface parallèle de la nouvelle imprimante. Ce connecteur – à mes yeux plutôt encombrant – est rapidement devenu un standard. Qui sait combien de milliers d'autres Centronics a dû en commander par la suite…

Bas coût, coup bas et « Muntzing »

Je me souviens aussi de critiques visant mon horloge à tubes VFD et ESP32 [4]. Tout est parti d'un message posté sur un forum allemand. L'utilisateur avait construit le projet de façon erronée et tentait d'en rejeter la faute sur la conceptrice – mon innocente personne. Un autre utilisateur du forum avait enchaîné en faisant remarquer que l'utilisation d'une inductance de 47 µH dans le convertisseur élévateur entraînait des surintensités, et que l'ensemble n'était pas très efficace puisque la grille du MOSFET était attaquée directement par la sortie du 7555. Cette critique n'était pas totalement infondée, mais il se trouve que je croulais pour ainsi dire sous les inductances de 47 µH – en raison d'un autre projet – et que j'étais donc encline à les utiliser partout où c'était possible. L'horloge consomme environ 2,25 W, dont un peu plus de 550 mW pour le convertisseur élévateur. Une telle consommation ne me poussait pas à courir après le rendement, et je ne tenais pas à me lancer dans la conception d'un coûteux convertisseur dernier cri. Si le rendement importe, j'utilise un afficheur LCD, pas des tubes VFD – même si un LCD n'a évidemment pas le glamour des tubes. Je glisse parfois vers un peu de sur-ingénierie, mais sans oublier qu'un projet doit être rentable. Ceci dit je suis loin de suivre les traces d'Earl Madman Muntz [5] [6], cet homme d'affaires haut en couleurs dont la légende dit qu'il portait toujours sur lui une paire de pinces coupantes afin de supprimer les composants superflus des circuits conçus par ses ingénieurs, et par là même en réduire le coût. Être accusée de cupidité sur un forum parce que mes kits étaient « un peu chers » aux yeux de certains, qui plus est avec une référence narquoise aux « banquiers d'investissement rapaces », c'était plutôt bas. Peut-être ces personnes n'avaient-elles jamais entendu parler de taxes et de coûts de la main-d'œuvre.

Intuition et raison

Ce qui comptait le plus durant mes études était le calcul et l'art de raisonner. La théorie primait sur la pratique, les méthodes empiriques étant même considérées avec un certain dédain. Tout cela était bien beau, mais je doute fort qu'un patron apprécierait de voir son employé passer des heures à appliquer des formules sur un circuit simple. Et pour l'indépendante que je suis, le dicton « Le temps, c'est de l'argent » relève de la lapalissade. Il m'arrive bien sûr de faire des calculs, mais je ne crains pas de faire confiance à mon intuition, surtout pour des choses relativement simples. Une approche empirique suffit à certains circuits, surtout s'ils reposent sur d'obscurs composants d'un autre âge. Et nous devrions aussi avoir le courage de réutiliser des conceptions existantes, au lieu de toujours vouloir réinventer la roue.

Guglielmo Marconi avait lui aussi adopté une approche empirique pour mettre au point son détecteur magnétique « Maggiev» [7], s'appuyant pour cela sur les travaux d'Ernest Rutherford et de Harry Shoemaker. Maggie rendit de nombreux services jusqu'à l'avènement des détecteurs à tubes à vide, y compris à bord du Titanic. Il aura fallu vingt ans pour que des scientifiques en élucident le fonctionnement. Maggie illustre bien les talents d'ingénieur « pragmatique » de Marconi et les bénéfices d'une telle approche.

220671-04 — VF : Hervé Moreau

Note : les passages susceptibles d'affecter le rendement de votre lecture ont été supprimés à l'aide du bouton Supprimer que j'emporte toujours avec moi dans mon sac à main.

LIENS

[1] Ilse Joostens, shield à VFD pour Arduino Uno, Elektor 9/2015 : http://elektormagazine.fr/magazine/elektor-201509/28018
[2] Port parallèle : http://fr.wikipedia.org/wiki/Port_parall%C3%A8le
[3] Manuals Library – Manuel de la Centronics 101 : https://manualslib.com/manual/1438231/Centronics-101.html
[4] Horloge à tubes VFD et ESP32, Elektor 5/2018 : http://elektormagazine.fr/magazine/elektor-201805/41576
[5] Bob Pease – What's All This Muntzing Stuff, Anyhow? – Electronic Design (1992) : https://elektor.link/MuntzingStuff
[6] Documentaire – Madman Muntz : https://youtu.be/deFlB2G0mH8
[7] Détecteur magnétique : http://fr.wikipedia.org/wiki/D%C3%A9tecteur_magn%C3%A9tique

pilote de moteurs pas à pas UCN5804

drôle de composant, la série

David Ashton (Australie)

Un pilote de moteur pas à pas ? Qu'a-t-il de particulier ? Il y en a probablement des centaines ! Eh bien, la particularité de celui-ci réside dans sa simplicité. Il s'agit d'un seul circuit intégré avec des entrées *Step* et *Direction* pour piloter directement un moteur pas à pas. Aucune autre logique ou circuit intégré ne sont requis.

Le UCN5804 est un circuit intégré de commande de moteur pas à pas fabriqué par Allegro Microsystems. Il se présente sous la forme d'un boîtier à 16 broches (DIP ou SOIC), avec les broches centrales de chaque côté (4/5 et 12/13) liées ensemble et à la masse. Cela permet la dissipation thermique en plus de la mise à la terre. C'est un excellent petit circuit intégré, capable de piloter quatre enroulements d'un moteur pas à pas unipolaire à 4 phases, avec seulement les entrées *Step* et *Direction*. Ses capacités de sortie (1,5 A et jusqu'à 35 V) sont également impressionnantes, il peut donc commander directement la plupart des petits moteurs pas à pas. Il existe également des entrées monophasées et demi-pas pour un peu plus de polyvalence. Agréable, simple, et facile à utiliser.

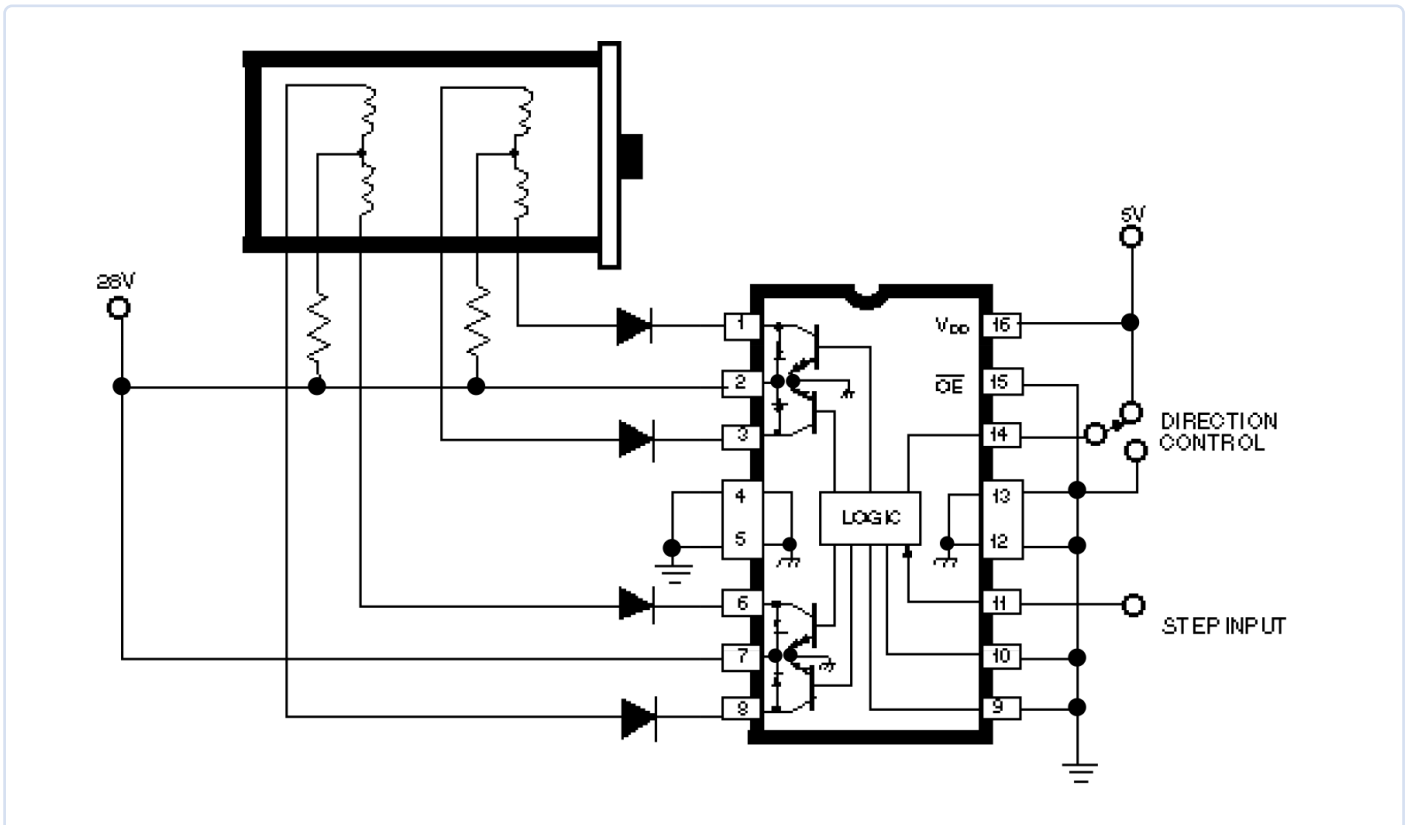


Figure 1. Circuit UCN5804. (Source : Fiche technique d'Allegro Microsystems.)

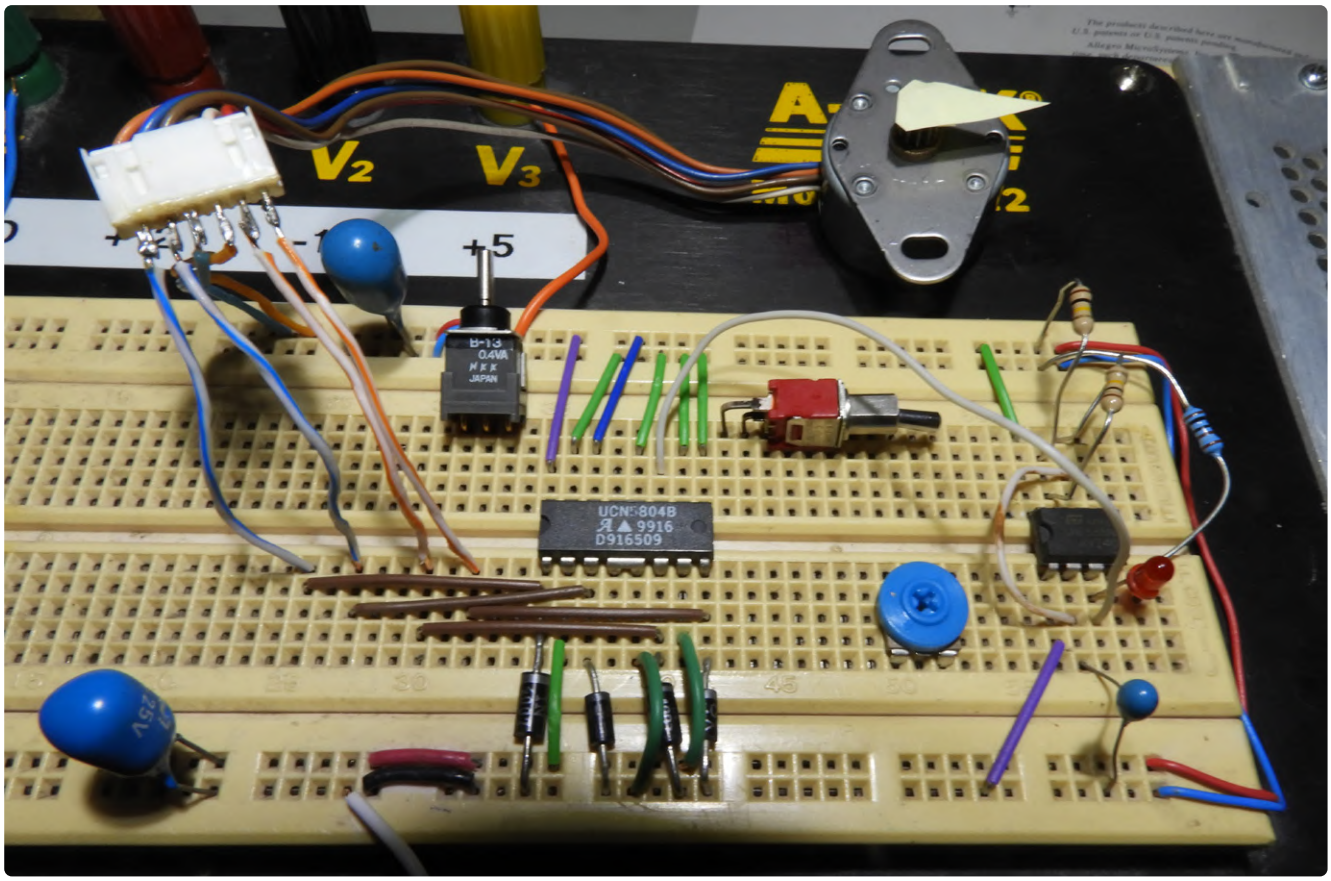


Figure 2. Une plaque d'essais utilisant un UCN5804 récupéré, un 555 pour la synchronisation des pas, et quelques interrupteurs pour le contrôle.

Malheureusement, il a été abandonné, bien qu'Allegro fournisse toujours la fiche technique sur son site [1], ce qui est louable - beaucoup de sociétés retirent les fiches techniques des composants dont la production a cessé.

Un circuit typique est illustré à la **figure 1**. Comme vous pouvez voir, tout ce dont vous avez besoin pour commander un moteur pas à pas est une poignée de diodes. Le sens de rotation est contrôlé par un interrupteur, mais il pourrait tout aussi bien être déterminé par un signal provenant d'un microcontrôleur ou d'un autre circuit logique.

Alors, pourquoi Allegro a t'il arrêté un si bon circuit de commande de moteur pas à pas ? Probablement parce qu'il s'agit d'un ancien circuit bipolaire, et non MOS, et qu'il ne possède aucune des fonctions de réduction de la consommation d'énergie exigée aujourd'hui par les fabricants pour rendre leur système plus efficace électriquement. L'avis de fin de série sur la fiche technique recommande les A3967 et A3977 d'Allegro, ce qui n'est pas du tout la même chose, car ils sont faits pour commander des moteurs pas à pas bipolaires (plus courants de nos jours, il est vrai) avec des pilotes à pont en H. Ils disposent également de beaucoup plus de fonctionnalités, comme la détection de courant, la commande PWM et les modes demi, quart et huitième de pas. L'oubli principal, surtout pour les amateurs comme moi qui bricolent encore sur des plaques d'essais, est que ces deux circuits intégrés ne sont disponibles que dans des boîtiers SOIC de 24 et 28 broches, respectivement.

Il existe plusieurs autres solutions de commande de moteurs pas à pas, mais elles sont toutes inférieures à l'UCN5804 sur un point ou un autre. La combinaison bien connue L297/298 est assez polyvalente, mais il s'agit d'une solution à deux circuits. D'autres dispositifs prétendant être des pilotes de moteurs pas à pas ne sont que des demi-ponts en H, sans les entrées *Step* et *Direction* simples et fonctionnelles. C'est donc à vous, le développeur courageux, de générer les formes d'onde de commande avec un autre circuit intégré ou votre microcontrôleur. C'est vraiment dommage qu'Allegro ait abandonné ce circuit intégré, car il est idéal pour les amateurs (oui, je sais, les amateurs sont un marché modeste). Dans la **figure 2**, vous trouverez une plaque d'essais utilisant l'UCN5804 avec un timer 555 et quelques interrupteurs. Heureusement, j'en ai encore quelques-uns récupérés sur des lecteurs de sauvegarde à bande mis au rebut. Cependant, si vous regardez autour de vous, vous pouvez encore les trouver sur Internet pour environ 5 à 10 € la pièce.

(220530-04) — VF : Laurent Rauber

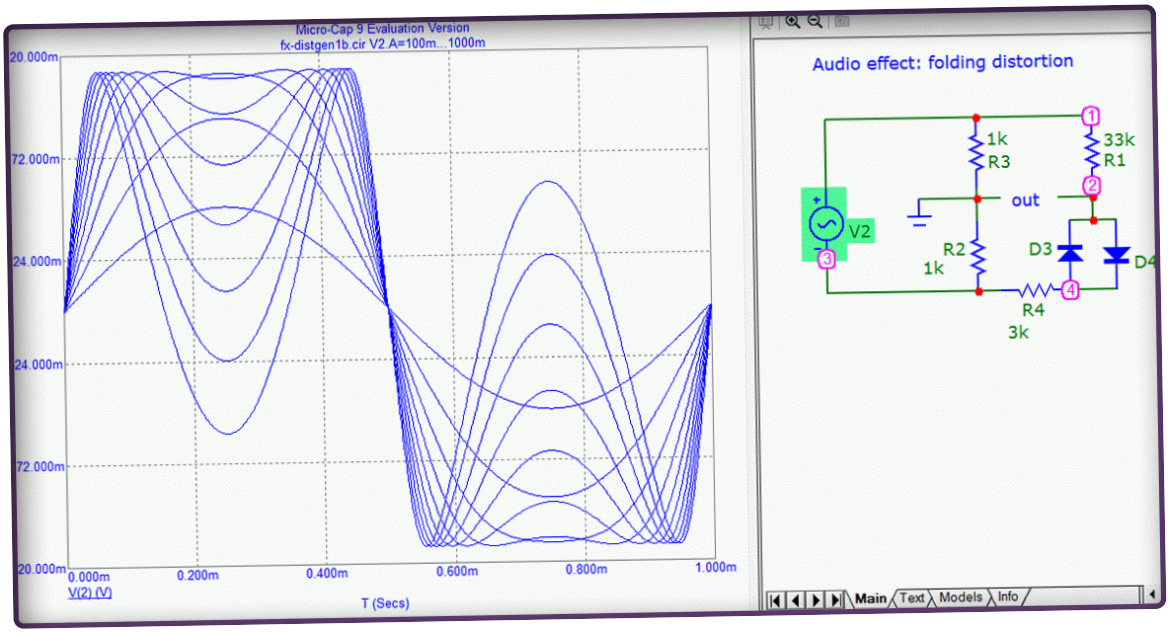
Des questions, des commentaires ?
Contactez Elektor (redaction@elektor.fr).

LIENS

[1] Fiche technique de l'UCN5804 - Allegro Microsystems : https://elektor.link/ucn5804

simulation de circuit avec Micro-Cap

premiers pas dans un monde compliqué



Raymond Schouten (Pays-Bas)

Un simulateur de circuits devrait faire partie de la boîte à outils de tout passionné d'électronique, au même titre qu'une alimentation électrique, un multimètre et un oscilloscope. Et comme pour tout autre outil, pour en tirer le meilleur parti, vous devez consacrer un peu de temps à apprendre à l'utiliser. Dans cet article, nous vous montrons comment commencer à utiliser le simulateur gratuit Micro-Cap.

Pourquoi simuler ?

L'utilisation d'un simulateur de circuit peut vous faire gagner du temps pour tester des idées ou spécifier des composants. C'est également un bon outil d'apprentissage et de débogage, qui vous montre ce qui se passe à l'intérieur d'un circuit.

Les logiciels de simulation de circuits comme LTspice (Analog Devices), Tina (Texas Instruments) et Micro-Cap utilisent le code SPICE basé sur la saisie de texte et développé à l'Université de Californie, Berkeley (1973). L'apport des logiciels modernes consiste essentiellement en une interface graphique. Une partie importante de cet article est donc valable pour les trois, surtout lorsqu'il s'agit de conseils pour éviter les erreurs de simulation.

Le simulateur présenté dans cet article est Micro-Cap [1]. Il s'agit d'un simulateur de circuit de qualité professionnelle (4 500 $) qui a été publié sous forme de freeware après l'arrêt de son développement. Il bénéficie ainsi de fonctions plus avancées que d'autres simulateurs gratuits mais risque l'obsolescence à terme. Pour l'instant, il est disponible gratuitement sur [1].

Micro-Cap vous permet de dessiner des circuits et de visualiser les signaux qui s'y propagent tout en « tournant des boutons » (en faisant varier les paramètres). Il inclut une grande bibliothèque de modèles de composants, avec même des tubes à vide. Il comprend des outils de conception de filtres actifs et passifs. Il vous permet d'écouter la forme d'onde produite par votre circuit simulé : un plus pour les applications audio.

Comment simuler ?

Simuler un circuit, c'est comme construire quelque chose sur une plaque d'expérimentation. Vous placez les composants, les reliez

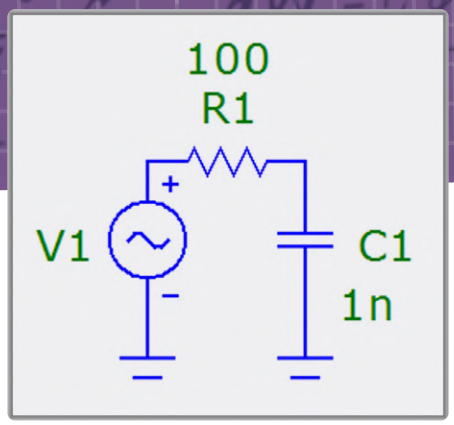


Figure 1. Mon premier circuit, un simple filtre RC passe-bas.

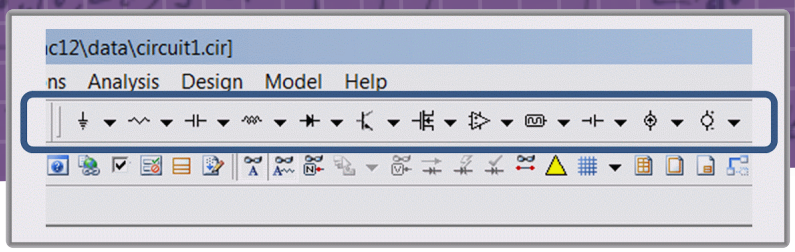


Figure 2. Micro-Cap possède une barre de sélection de composants.

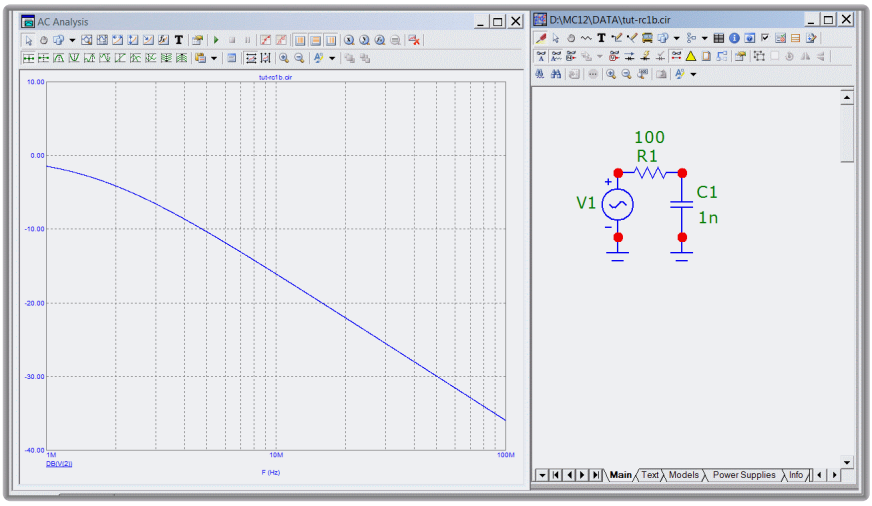


Figure 3. En cliquant sur le nœud RC, la courbe de transfert du filtre s'affiche.

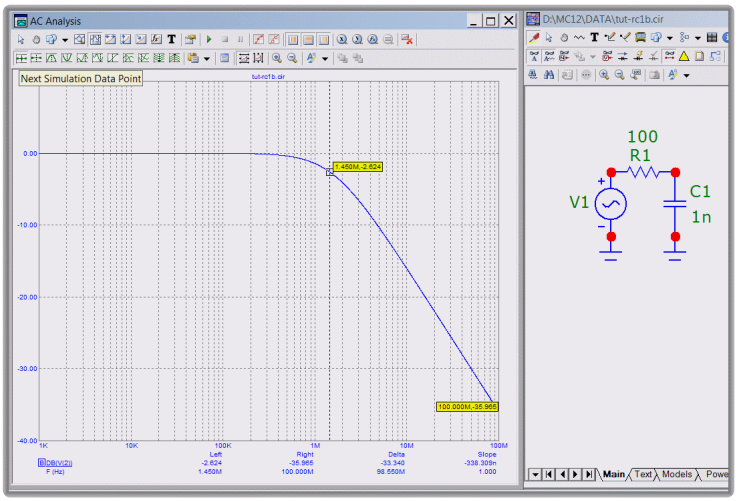


Figure 4. Fixez les limites du graphique en fonction de ce que vous voulez voir.

pour former un circuit, ajoutez les tensions d'alimentation et (éventuellement) connectez une source de signaux. Ensuite, vous choisissez de mesurer les signaux avec un oscilloscope (dans le domaine temporel) ou avec un analyseur de spectre (dans le domaine fréquentiel). Vous trouverez plus de détails à ce sujet dans l'encadré **« Types d'analyse »**.

Dessiner notre premier circuit

Après avoir lancé Micro-Cap, nous commençons par créer un simple filtre RC passe-bas et simulons sa réponse en fréquence comme le montre la **figure 1**. Il existe plusieurs façons et raccourcis pour ajouter des composants au circuit (voir encadré). Pour l'instant, nous utilisons la barre de composants située en haut de l'écran (**figure 2**).

Cliquez brièvement avec le bouton gauche de la souris sur le symbole de la résistance et déplacez le pointeur de la souris dans votre feuille de travail vide (vous pouvez aussi cliquer sur la petite flèche ou le triangle à droite du bouton du symbole pour en changer l'orientation). Un symbole de résistance apparaît. Cliquez à nouveau avec le bouton gauche de la souris pour placer la résistance à l'endroit souhaité dans la feuille de travail. Dans la fenêtre qui s'ouvre maintenant, entrez une valeur de 100 dans la case *Value* en haut. Fermez la fenêtre à l'aide du bouton *OK* ou en appuyant sur la touche Entrée.

Répétez cette procédure pour un condensateur ; fixez sa valeur à *1n*. Si vous placez le condensateur de manière à ce que l'un de ses fils touche l'un des fils de la résistance, ils seront connectés automatiquement.

Ensuite, nous ajoutons un générateur ou une source d'ondes sinusoïdales en cliquant sur *Component* dans le menu principal, puis en sélectionnant *Analog Primitives Waveform Sources Sine Source*. Sélectionnez *GENERAL* dans la liste sur la droite de la fenêtre des propriétés de la source qui s'est ouverte, puis cliquez sur *OK*.

Terminez le circuit en ajoutant les symboles de masse de la barre de composants au condensateur et à la source du signal.

Si vous avez placé les composants loin les uns des autres, il faut les relier. Un clic sur le bouton *Wire Mode* ou l'appui sur Ctrl-W active l'éditeur de câblage. Pour dessiner un fil, cliquez sur le point de départ du fil et, tout en maintenant le bouton gauche de la souris enfoncé, faites glisser le pointeur vers la destination du fil.

La simulation

Nous sommes maintenant prêts à exécuter une simulation. Dans le menu *Analysis*, sélectionnez *Probe AC...* pour passer en mode analyse, prêt à « mesurer ». Dans ce mode, vous ne pouvez plus modifier le dessin du circuit, seulement les valeurs des composants. Les points rouges dans le circuit sont les nœuds. En cliquant sur un point, vous obtiendrez un graphique dans la fenêtre de *AC Analysis* similaire à celui de la **figure 3**.

Il s'agit d'un graphique avec les paramètres de fréquence par défaut, mais nous voulons un graphique commençant à une fréquence plus basse. Pour ce faire, sélectionnez *Limits...* dans le menu *Probe*. Changez la *Frequency Range* en « *100Meg*, *1k* » (le format est « *high*, *low* ») et cliquez sur *Close* pour quitter. Le graphique devient celui de la **figure 4**. Nous pouvons maintenant voir à la fois la bande passante du filtre (du continu jusqu'à 1 MHz) et sa bande d'arrêt (au-delà de 1 MHz).

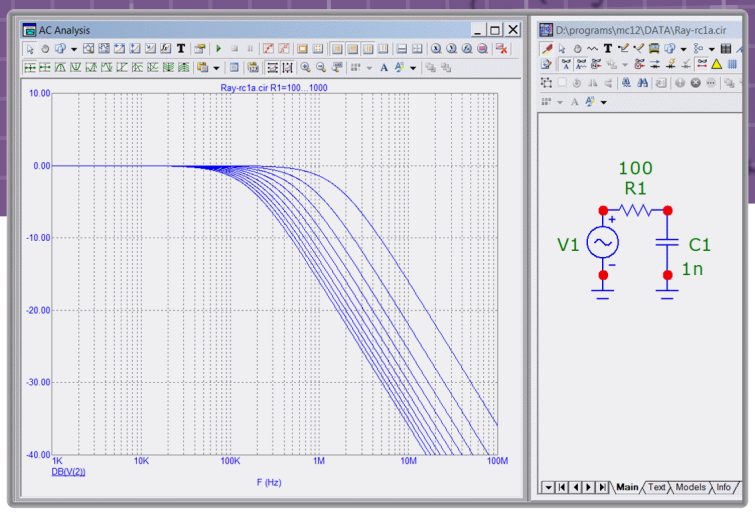


Figure 5. Le balayage d'un paramètre (ici la valeur de R1) permet de visualiser son influence sur la fonction de transfert du circuit.

Vous pouvez lire les points de données dans le graphique en utilisant un curseur. Cliquez sur le bouton supérieur gauche *Next Simulation Data Point* au-dessus du graphique et un curseur apparaît que vous pouvez déplacer.

Simulation et réalité

Selon les graphiques des **figures 3** et **4**, notre filtre bloquerait tout ce qui dépasse 1 MHz ? Donc, toutes les fréquences jusqu'à l'infini ? Nous savons qu'il n'en est rien en réalité. Alors pourquoi est-ce le cas dans la simulation ?

La raison en est qu'un simulateur utilise des composants idéaux. Si vous voulez vous rapprocher du comportement réel d'un circuit, vous devez tenir compte des imperfections de vos composants et du monde réel en ajoutant des éléments parasites. Reportez-vous à l'encadré **« Limites du modèle de composants »**, pour plus de détails sur la façon de rendre votre simulation plus proche d'un circuit réel.

Modification de la valeur d'un composant en mode analyse

En mode analyse, vous pouvez encore modifier les valeurs des composants. Pour ce faire, vous devez passer en *Select Mode* en cliquant sur le bouton portant l'icône de la flèche de la souris en haut à gauche de la fenêtre du circuit (ou en appuyant sur Ctrl-E). Vous pouvez maintenant double cliquer sur la valeur d'un composant et la modifier. Les résultats de la simulation sont automatiquement mis à jour. Si vous voulez sonder un autre endroit du circuit, vous devez revenir au mode *Probe* en cliquant sur le bouton *Probe*.

Paramètre en pas-à-pas

Une caractéristique puissante d'un simulateur est qu'on peut faire varier la valeur d'un ou plusieurs paramètres et afficher les résultats sous forme de graphiques multiples. Par exemple, augmentons la valeur de la résistance R1 comme s'il s'agissait d'un potentiomètre pour évaluer la plage de réglage pour une certaine variation du potentiomètre.

Faisons varier la fréquence de coupure du filtre en faisant passer R1 de 100 Ω à 1000 Ω par pas de 100 Ω. Pour cela, sélectionnez *Stepping*... dans le menu *Probe*. Sélectionnez R1 dans la case *Step What*. Saisissez des valeurs pour les cases *From*, *To* et *Step Value* et n'oubliez pas de positionner *Step It* sur *Yes*. Cliquez sur *OK*, puis sur le bouton *Run* (F2) (avec la flèche ou le triangle vert) dans la fenêtre *AC Analysis* pour afficher le graphique en fonction de la valeur de la résistance (**figure 5**).

Dans cet exemple, les pas étaient de taille constante (linéaire) mais ils pourraient être exponentiels. Il est possible de faire varier plusieurs valeurs de composants simultanément ou de manière emboîtée. Vous pouvez également agir sur la valeur d'un paramètre de composant, par exemple, le gain d'un transistor.

Analyse transitoire

Vous devez choisir le bon mode d'analyse pour voir le comportement non linéaire (voir l'encadré **« Types d'analyse »**). Pour obtenir un comportement non linéaire, réalisons un effet de distorsion audio simple, un écrêteur à diode comme illustré à la **figure 6**. Nous pouvons ensuite utiliser l'analyse des transitoires pour observer la distorsion comme si nous utilisions un oscilloscope.

Réglez la fréquence de la source sinusoïdale V1 sur 1 kHz (dans la case « F » dans le

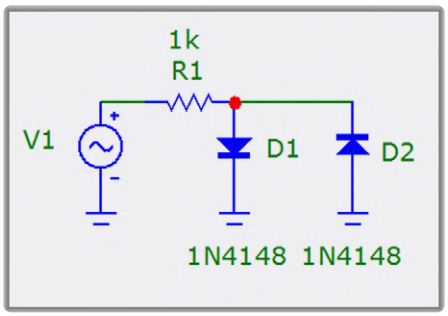


Figure 6. Ce circuit simple présente un comportement non linéaire.

Combien vaut « M » ?

Dans le langage SPICE M signifie milli (10^{-3}) donc si vous définissez la valeur d'une résistance à 1 Mohm, elle sera de 0,001 Ω.... Vous pouvez utiliser MEG pour méga ou 10^6.

Types d'analyse

Après avoir dessiné le circuit, vous souhaitez effectuer des mesures. Pour cela, vous devez sélectionner un mode d'analyse. Dans le langage SPICE, la mesure à l'aide d'un oscilloscope est appelée « analyse transitoire », tandis que la mesure à l'aide d'un analyseur de spectre est appelée « analyse fréquentielle ». Les points d'interconnexion des composants sont appelés nœuds. En mode analyse, vous pouvez sonder les tensions sur ces nœuds et les courants qui les traversent.

Pour visualiser un comportement non linéaire (par exemple une distorsion ou un écrêtage), vous devez utiliser l'analyse transitoire. L'analyse fréquentielle suppose toujours un comportement linéaire et vous donne la réponse en fréquence des petits signaux de circuits tels que les filtres et les amplificateurs.

Dans les deux types d'analyse, vous pouvez également voir les tensions et les courants continus. D'autres types sont possibles, comme l'analyse du bruit et de l'impédance.

Lorsque le circuit a été dessiné correctement (pas de nœuds flottants, par exemple), un type d'analyse est sélectionné. Un calcul est alors effectué sur une période de temps ou une plage de fréquence définie par l'utilisateur avec une résolution donnée. Le réglage de cette résolution est un compromis entre la régularité de la courbe et le temps de calcul. Lorsque le calcul est prêt, vous pouvez commencer à sonder le circuit.

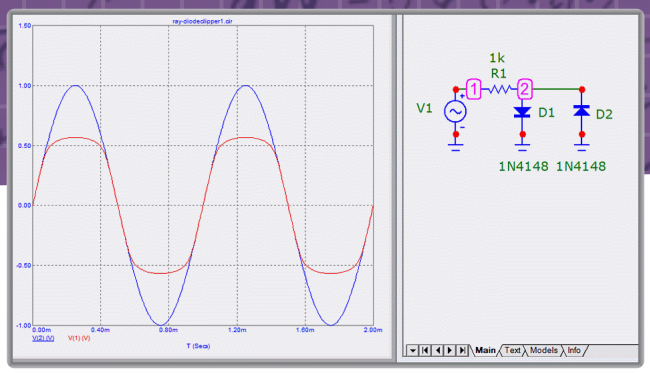


Figure 7. Une analyse transitoire du circuit de la figure 6 produit ces signaux.

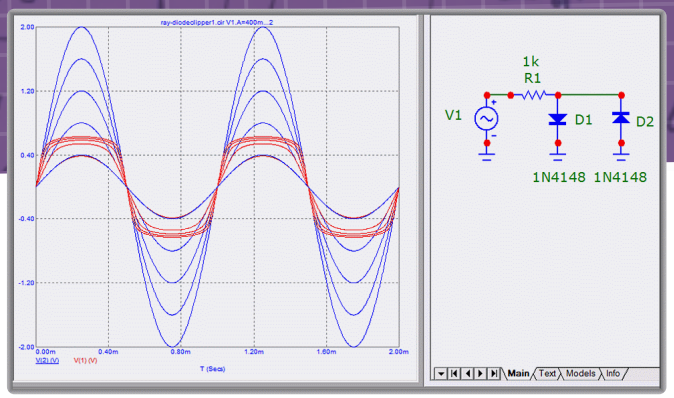


Figure 8. La saturation de la sortie augmente progressivement lorsque l'on augmente l'amplitude du signal d'entrée.

coin inférieur droit de la fenêtre des propriétés) et choisissez *Probe Transient*... dans le menu *Analysis*. Dans le menu *Probe*, choisissez *Limits*... et fixez *Maximum Run Time* à *2m* et *Maximum Time Step* à *2u*. Une bonne pratique consiste à prendre pour le pas de temps le millième de l'intervalle de temps d'exécution. Sondez maintenant le circuit, à la fois à l'entrée (montrant l'onde sinusoïdale) et à la sortie (montrant la forme d'onde écrêtée).

Balayage de l'amplitude

Comme expliqué précédemment, vous pouvez faire varier une valeur ou un paramètre de composant. Faisons-le pour l'amplitude de la source sinusoïdale. Sélectionnez *Stepping*... dans le menu *Probe*. Choisissez l'amplitude « A » de V1 et passez de *400m* à *2V* par pas de 400 mV (n'oubliez pas de positionner *Step It* sur *Yes*). La **figure 8** montre les résultats.

Comme le circuit de la **figure 6** est censé réaliser un effet audio, il serait utile d'entendre le son qu'il produit. Micro-Cap offre la possibilité de lire la forme d'onde sur votre carte son ou de la stocker dans un fichier WAV ou CSV.

La durée du son est fixée par les limites de la sonde. Dans *Probe Limits* modifiez *Maximum Run Time* à *500m* (*et Maximum Time Step* à *50u*) pour obtenir une demi-seconde de son. Ensuite, double-cliquez sur la fenêtre de la trace pour ouvrir ses propriétés et sélectionnez l'onglet *Save Curves*. Sélectionnez l'onde

Où trouver les composants ?

Tous les composants sont accessibles via l'onglet Components et dans la liste déroulante de la bibliothèque, mais il existe également des raccourcis. Il y a la barre des composants en haut de la fenêtre, mais encore plus rapides sont les touches comme « R » pour une résistance et « C' » pour un condensateur. En outre, le programme enregistre les composants récemment utilisés dans la fenêtre d'ouverture de l'onglet Components.

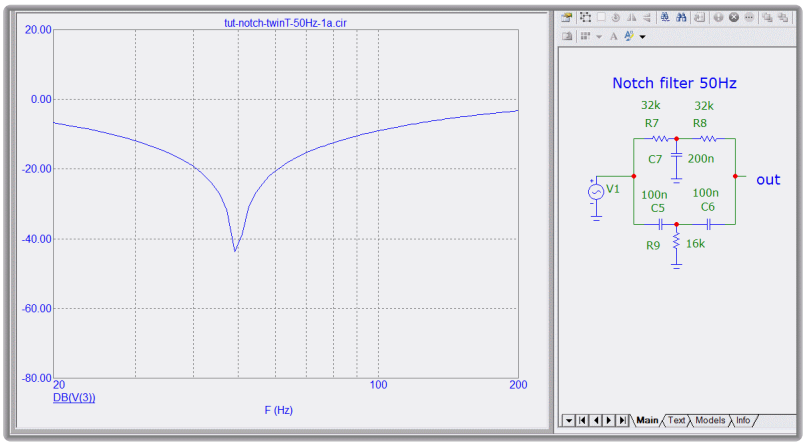


*Figure 9: La résolution de la simulation est importante. Ici, **Probe Limits** a été fixé à 100 points.*

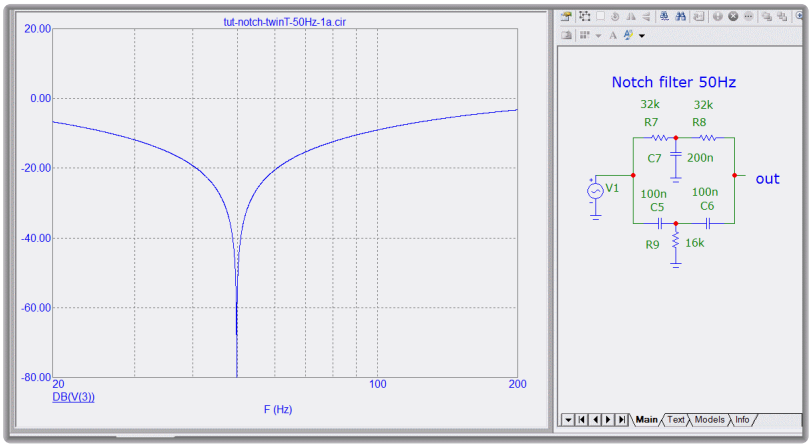


*Figure 10. La même simulation que dans la Figure 9, mais maintenant avec **Probe Limits** fixé à 10 000 points.*

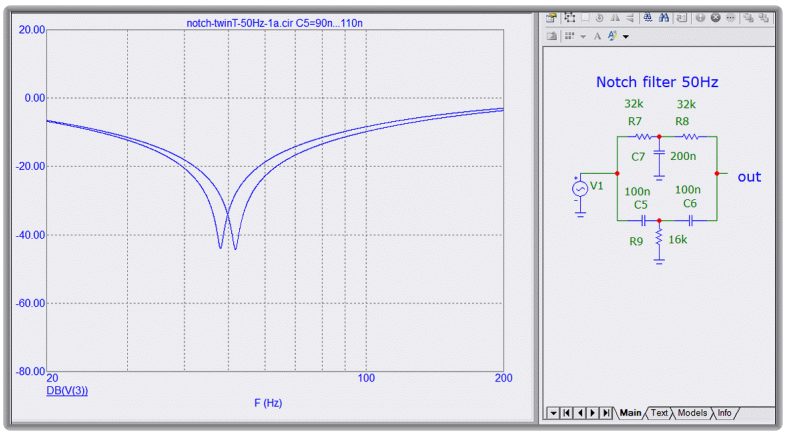


Figure 11. Le même circuit que celui de la figure 9, mais avec C5 non identique à C6.

que vous voulez entendre dans la liste *Curves*. Si vous avez activé le pas-à-pas (par exemple, celui de l'amplitude du signal d'entrée), vous pouvez écouter chaque pas en le sélectionnant dans la case *What To Save*, puis en appuyant sur *Play*.

Résolution des simulations

Des réglages de résolution incorrects peuvent affecter ou même supprimer des détails des résultats de la simulation. Par exemple, un filtre coupe-bande présente un creux prononcé à la fréquence à laquelle il fonctionne. Si la résolution en fréquence est insuffisante, le creux semble moins profond qu'il ne l'est. La **figure 9** et la **figure 10** montrent les résultats du même filtre coupe-bande Twin-T simulé deux fois, mais avec des réglages de résolution différents.

Composants idéaux

Ce filtre présente un creux infiniment profond à 50 Hz avec les valeurs de composants données. Cependant, avant de construire ce filtre « idéal », réfléchissez un peu aux valeurs des composants. Dans un simulateur, les deux condensateurs de 100 nF seront identiques et exactement 100 nF, et vous obtiendrez un filtre coupe-bande 50 Hz infiniment profond, comme illustré ici.

La modification de la valeur de C5, par exemple, de plus ou moins 10 %, a une grande influence (**figure 11**). Le creux n'est plus que de 40 dB et le filtre est aussi désaccordé, ce qui entraîne une atténuation de seulement 35 dB à 50 Hz. Cela montre comment un simulateur peut vous aider à prédire les effets des tolérances des composants du monde réel sur les performances de votre circuit avant même de le construire.

Pour des simulations encore plus réalistes, vous devez également prendre en compte d'autres paramètres parasites, tels que la longueur des pistes conductrices et des fils (voir encadré).

Ce n'est pas tout, les amis

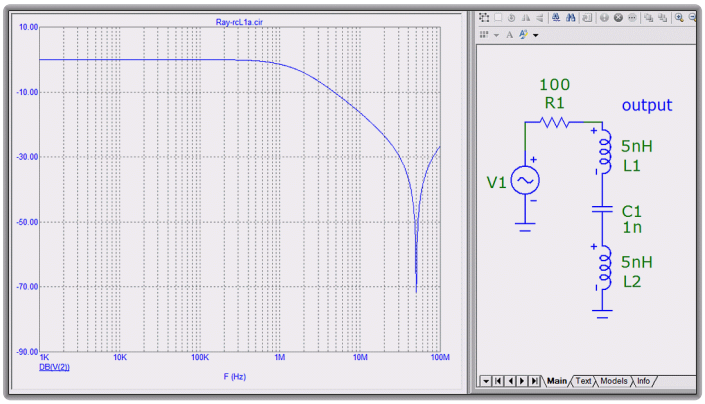
Cet article n'a fait qu'effleurer la surface d'un sujet complexe. Comme indiqué au début de cet article, un simulateur de circuit doit faire partie de votre boîte à outils, à côté d'un multimètre et d'un oscilloscope. Cependant, gardez toujours à l'esprit qu'un simulateur de circuit simplifie la réalité. Obtenir des résultats corrects avec un simulateur, Micro-Cap ou autre, nécessite une certaine connaissance des aspects pratiques de la construction d'un circuit. Savoir comment le configurer et quelles sont ses limites est une condition préalable. Combiner la simulation avec la construction d'un prototype réel est toujours une bonne idée. Ne vous débarrassez donc pas trop vite de votre plaque d'expérimentation avec ses mauvais contacts (comment les modéliser ?).

220336-04 — VF : Helmut Müller

Limites du modèle de composants

Une simulation n'est aussi bonne que les modèles qu'elle utilise. Tous les composants de notre filtre sont idéaux. En pratique, il n'en est rien. Pour notre filtre passe-bas simple, la principale limitation est l'inductance série des fils du condensateur. Chaque millimètre de fil ajoute environ 1 nH d'inductance qui augmente l'impédance du condensateur au-delà d'une certaine fréquence, réduisant ainsi l'effet de filtrage.

Supposons que votre condensateur ait des fils de 5 mm, et simulons cela en ajoutant deux inductances de 5 nH en série avec les connexions du condensateur (ce qui revient à ajouter une inductance en série de 10 nH).



Nous constatons que le filtre présente une forte atténuation autour de 50 MHz, puis commence à « fuir » à partir de 50 MHz. Ce phénomène est causé par le circuit LC résonnant. Si vous voulez filtrer des interférences RF jusqu'à 1 GHz, c'est un problème, mais dans des cas particuliers, vous pouvez utiliser cet effet, par exemple pour filtrer une fréquence d'horloge d'échantillonnage de 50 MHz, mais il vous faudra retoucher finement la valeur de l'inductance.

L'exemple ci-dessus illustre l'aide apportée par un simulateur dans la conception correcte d'un circuit.

Question : si les fils ajoutent une inductance, l'utilisation de condensateurs CMS permet-elle d'obtenir un filtre sans inductance ? Non, mais cela s'en rapproche. Le condensateur a toujours une longueur finie (2 mm correspondent à 2 nH) et sa connexion à la masse sur le circuit imprimé ajoute également une inductance. Les vias peuvent ajouter 0,3 nH à 1 nH selon la façon dont vous les définissez et les placez.

Des questions, des commentaires ?

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

Produits

- **Joy-IT ScopeMega50 oscilloscope USB (SKU 18277)**
  www.elektor.fr/18277
- **Gilles Brocard, livre en anglais *« The LTspice XVII Simulator »* (SKU 19741)**
  www.elektor.fr/19741

LIENS

[1] Micro-Cap website : https://www.spectrum-soft.com/download/download.shtm

PAUL Award 2022

les jeunes talents de la technologie et leurs solutions créatives

Par Florian Schäffer, pour le compte d'Elektor

La promotion des jeunes talents de l'électronique est une question majeure, notamment pour l'association professionnelle allemande de conception et de fabrication électroniques (FED). Cette année à Potsdam, dans le cadre de la conférence FED, le prix PAUL doté de 6 000 euros a également été décerné en 2022. Toute personne âgée de 15 à 25 ans pouvait soumettre un projet électronique innovant, les trois équipes finalistes gagnantes vous sont présentées ici : Fisego (première), Manuel Wilke (deuxième) et BMK (troisième).

Nominés et gagnants du PAUL Award 2022 lors de la photo de groupe à Potsdam (copyright FED e.V.).

Bloc multiprises avec protection contre les incendies

Un bloc multiprise avec protection thermique intégrée, basée sur un microcontrôleur, a été développée par l'équipe gagnante Fisego, afin d'éviter les incendies dus à une surcharge. L'idée, la mise en œuvre et la documentation ont été récompensés par le jury avec le premier prix et une prime de 3 000 euros. Lors de la cérémonie de remise des prix à Potsdam, Sophia Reiter (22 ans, étudiante en électrotechnique) et Fabian Goedert (25 ans, étudiant en génie civil) ont également eu le plaisir de recevoir le prix média, décerné indépendamment par Elektor.

Jürgen Deutschmann, membre du jury, remet le premier prix pour la multiprise polyvalente anti-incendie à Sophia Reiter et Johannes Steube. (Copyright FED e.V.).

Dans le prototype imprimé en 3D, un circuit composé d'ATmega328, d'un capteur à effet Hall ACS712 et d'un capteur de température DS18B20 surveille la puissance absorbée et la température dans le bloc multiprise. Dans une phase d'alerte à trois niveaux, les paramètres sont affichés à l'utilisateur sur un écran OLED. En cas de dépassement d'une valeur limite, il y a d'abord un avertissement avec des instructions sur des actions immédiates, avant que la prise ne soit automatiquement coupée lorsque la valeur limite suivante est atteinte. Contrairement aux prises avec fusible thermique ou disjoncteur, l'utilisateur a ici la possibilité d'intervenir lui-même en coupant par exemple une charge individuelle avant la coupure totale.

D'après Sophia Reiter, l'objectif que se sont fixé les deux étudiants est une prise multiple pour l'Internet des objets (IdO), avec un système de protection contre l'incendie intégré, qui ne sera pas seulement utilisée dans le domaine privé, mais qui convaincra également les utilisateurs industriels et pourra surveiller n'importe quel type de machine ou d'installation. L'entreprise récemment créée a déjà franchi un pas de plus vers cet objectif avec la fabrication de prototypes de la multiprise par moulage par extrusion.

La FED reconnaît les problèmes de recrutement des jeunes et commence à promouvoir les idées

Le PAUL Award a été décerné pour la première fois en 2020, mais il a ensuite dû être reporté pour cause de Covid-19, et ce n'est que cette année, le 28 septembre, que la deuxième édition a pu avoir lieu. Comme l'a souligné Jürgen Braunsteiner, membre nouvellement élu du conseil d'administration de la FED, dans son discours d'introduction devant une centaine d'invités, le secteur est confronté à un changement générationnel, dans lequel les petites et moyennes entreprises sont particulièrement confrontées à des problèmes de relève.

C'est pourquoi la promotion de jeunes gens aux idées innovantes est un objectif important : le PAUL Award doit leur permettre de les motiver à développer leurs projets, d'avoir la possibilité de les présenter et de trouver des contacts dans l'industrie.

Braunsteiner a cité comme exemple positif la jeune Australienne Cynthia Sin Nga Lam, avec son projet *H2prO*, qui veut purifier les eaux usées par la photocatalyse et produire de l'électricité en même temps. Son dispositif, qui a le potentiel de changer le monde, a fait beaucoup parler de lui dans les médias en 2014. Ce projet a montré que les nouvelles idées germent partout, à condition d'oser travailler à leur réalisation, même avec des moyens rudimentaires.

L'équipe Fisego avec le prix spécial d'Elektor (copyright FED e.V.).

La deuxième place est revenue à Manuel Wilke pour son système de gestion de batterie BMS (copyright FED e.V.).

Récupération d'énergie avec un élément Peltier

Le petit groupe de cinq candidats devait en outre relever le défi de la récolte d'énergie : les projets devaient fonctionner sans source d'énergie externe. Un défi que l'équipe Drink Safe a également relevé. Ils ont développé une sous-tasse qui rappelle beaucoup les gadgets qui gardent les tasses au chaud par USB, mais qui a une toute autre fonction : la sous-tasse mesure la température du mug, et indique par une LED lorsque la boisson est suffisamment refroidie pour être consommée sans se brûler. Une idée qui n'a rien d'inutile au vu des avertissements tels que les « *hot drink* » présents sur les gobelets jetables.

Le fonctionnement repose sur un élément Peltier, qui a pour effet de générer un flux électrique en cas de différence de température entre ses deux côtés. Comme la tension ainsi générée est très faible, une commande d'élévation de tension intégrée est nécessaire pour piloter la LED. C'est pour cela qu'aucune source d'énergie supplémentaire comme une pile ou une batterie n'est nécessaire. L'équipe a reçu la troisième place avec l'idée et sa mise en oeuvre.

Station météo et gestion de batteries BMS

Avec son idée d'installer une roue à aubes dans une descente de gouttière, et de l'utiliser pour faire fonctionner une station météo IdO autonome, sans utiliser d'alimentation électrique supplémentaire, une autre équipe n'a malheureusement pas réussi à convaincre le jury, ce qui est peut-être dû au fait que le projet était encore confronté à des erreurs initiales. À la fin de la manifestation, les deux étudiants n'ont pas su répondre eux-mêmes à la question de savoir si cela les empêcherait de continuer à bricoler ou de développer une nouvelle idée.

Le deuxième prix (2000 euros) a été décerné à Manuel Wilke, étudiant en cinquième semestre à l'université des sciences et technologies appliquées de Berlin (BHT), qui a développé le système *nandomBMS* de gestion de batteries (BMS). Le projet est basé sur un microcontrôleur Cortex-M4 XC4000 et un équilibreur de charge LTC6813 avec BMS d'Analog Devices, et peut surveiller jusqu'à 18 cellules connectées en série. Le microcontrôleur peut communiquer avec le monde extérieur via CAN. Le circuit imprimé doit surtout être utilisé dans un E-Buggy interne et présenter une isolation galvanique entre la haute tension et la basse tension.

La prochaine édition des PAUL Awards de la FED aura lieu en 2023

Pour l'année prochaine, les organisateurs souhaitent avant tout un plus grand nombre de participants, ce qui se traduit notamment par un travail de relations publiques plus important. En outre, il n'y a plus de contraintes techniques supplémentaires. Seuls les coûts doivent se situer dans le cadre de l'argent de poche, ce qui fait que ce sont plutôt les groupes scolaires et le groupe des *makers* qui sont visés. Ceux qui souhaitent participer individuellement ou en équipe peuvent déposer leur candidature dès maintenant et travailler sur leur projet jusqu'au 30 septembre 2023. Toutes les informations sont disponibles sur le site de PAUL Award [1].

220599-04 — VF : Laurent Rauber

Les projets gagnants du PAUL Award ont été exposés dans le cadre de la conférence FED : un nouveau prototype de multiprise polyvalente, le circuit imprimé nu du système de gestion de batterie et l'appareil Drink Safe.

L'équipe BMK, récompensée pour son projet Drink Safe, est arrivée à la troisième place (copyright FED e.V.).

LIENS

[1] PAUL Award : https://www.paul-award.de

ma première **radio définie par logiciel**

réalisée en moins de 15 minutes

Clemens Valens (Elektor Labs)

Saviez-vous que vous pouvez créer un récepteur radio FM simplement en dessinant quelques blocs et lignes sur une toile virtuelle ? Si le dessin n'est pas encore votre passion, elle le deviendra après avoir lu cet article.

Figure 1. L'adaptateur SDR utilisé dans cet article est HackRF One de Great Scott Gadgets.

La radio logicielle, également connue sous le nom de SDR (*Software-Defined Radio*), consiste à moduler et démoduler les signaux radio par logiciel. Au lieu de disposer de circuits électroniques spéciaux pour ce faire, comme un mélangeur ou un démodulateur, la SDR utilise un logiciel. Bien sûr, la transformation des signaux radio analogiques en signaux numériques et vice-versa implique un certain matériel, mais c'est tout ce dont vous avez besoin.
Veuillez noter que pour garder les choses simples, nous nous limiterons dans cet article à la réception de signaux radio, mais la plupart des éléments présentés sont également valables pour l'émission. Elle peut fonctionner dans les deux sens.

La SDR peut tout faire

La radio logicielle offre de nombreux avantages par rapport à la radio analogique. Tout d'abord, elle comprend toutes les techniques de modulation auxquelles vous pouvez penser, et plus encore. Cela signifie que vous pouvez non seulement écouter la radio AM et FM, mais aussi faire du wifi ou du Bluetooth, décoder la télévision numérique ou le DAB+, faire de la radio amateur, etc.
C'est possible parce qu'elle met en œuvre la technique de modulation ou de démodulation dans le logiciel qui est facile à modifier. Comme tout est fait par logiciel, elle peut également faire des choses très difficiles, voire impossibles, à réaliser avec des composants électroniques uniquement.

Une taille unique

Un deuxième avantage de la radio logicielle est que vous n'avez besoin que d'un seul matériel pour faire tout cela. Pas besoin de disposer d'un récepteur FM et d'un récepteur wifi et d'un récepteur de télévision numérique, etc. Elle utilise le même adaptateur pour tout.
Cet adaptateur est un convertisseur analogique-numérique (CA/N) avec une antenne qui transforme les signaux radio analogiques en signaux numériques pouvant être traités par un ordinateur. Pour la transmission, un convertisseur numérique-analogique (CN/A) est nécessaire. En réalité, cet appareil est un peu plus qu'un simple CA/N et/ou CN/A. Il comporte également un mélangeur pour amener le signal RF d'intérêt dans la gamme qu'il peut échantillonner (et l'inverse lors de la transmission), mais sa fonction de base reste la même.
Les convertisseurs SDR existent en plusieurs versions, et vous devez en choisir un qui corresponde à votre budget et à vos souhaits. Plus le convertisseur est cher, plus vous pouvez faire de choses avec. Ici, nous utilisons le HackRF One de Great Scott Gadgets [1] (**figure 1**). Ce n'est pas l'adaptateur le moins cher, mais il supporte la réception et l'émission, et il fonctionne jusqu'à 6 GHz. De plus, et ce n'est pas sans importance, il est supporté par de nombreuses plateformes logicielles SDR comme GnuRadio [2], SDR Sharp (SDR#) [3] et SDR++ [4]. Pour le projet décrit ci-dessous, d'autres frontaux SDR (moins chers) fonctionneront probablement aussi bien tant qu'ils sont capables de fonctionner dans la bande radio FM (87.5...108 MHz).

Open Source et matériel

Un autre avantage de la radio logicielle est qu'elle est en grande partie open-source et open matériel, créée par une grande communauté de passionnés. Ils conçoivent et publient des schémas pour les adaptateurs et développent des outils logiciels et des algorithmes pour les applications SDR. HackRF One est également open-source et les fichiers de conception sont disponibles sur GitHub [5].

Conditions préalables

Comme mentionné précédemment, pour ce projet SDR, j'ai utilisé HackRF One comme adaptateur avec une antenne télescopique de 88 cm. Nous utiliserons GNU Radio pour la partie logicielle. En fait, nous utiliserons GNU Radio Companion, alias GRC, l'interface graphique de GNU Radio qui est disponible pour Linux, Windows et macOS. Il y a donc de fortes chances que votre ordinateur puisse également l'exécuter. Il fonctionne même sur un Raspberry Pi.

Programmation par glisser-déposer

GRC est un outil de programmation graphique qui vous permet de créer une application radio sans faire de réelle programmation. Vous faites simplement glisser et déposer des blocs sur un canevas que vous interconnectez avec des fils virtuels. La programmation reste limitée à la configuration des blocs. GRC a d'emblée une large bibliothèque de blocs permettant de créer toutes sortes de systèmes de modulation et de démodulation.

Nous commençons par un canevas neuf. Il n'est pas complètement vide, car il contient déjà deux blocs : un bloc *Options* et un bloc *Variable*. Un double clic gauche de la souris sur un bloc l'ouvre, ce qui vous permet de modifier ses paramètres. Le bloc *Options* permet de définir quelques options de bas niveau, et vous pouvez saisir un titre pour le canevas, voir **figure 2**. Nous allons tout laisser aux valeurs par défaut.

Taux d'échantillonnage global

Le bloc *Variable* définit une variable globale de fréquence d'échantillonnage nommée `samp_rate`. Elle est utilisée dans d'autres blocs. Strictement parlant, vous n'en avez pas besoin, mais c'est pratique, et donc nous la gardons. Cependant, sa valeur est trop faible pour l'échantillonnage de signaux FM, et nous la changeons donc en 10 millions. La valeur maximale pour HackRF One est de 20 millions. Si vous la fixez trop haut, votre ordinateur risque d'avoir du mal à suivre.

Ajouter une source RF

Maintenant, nous avons besoin d'un dispositif d'entrée RF. En GRC, cela s'appelle une source. Il n'y a pas de bloc source spécial HackRF One. La section *Soapy* a une *Soapy HackRF Source*, mais cela n'a pas fonctionné pour moi. La source *osmocom*, elle, a fonctionné.

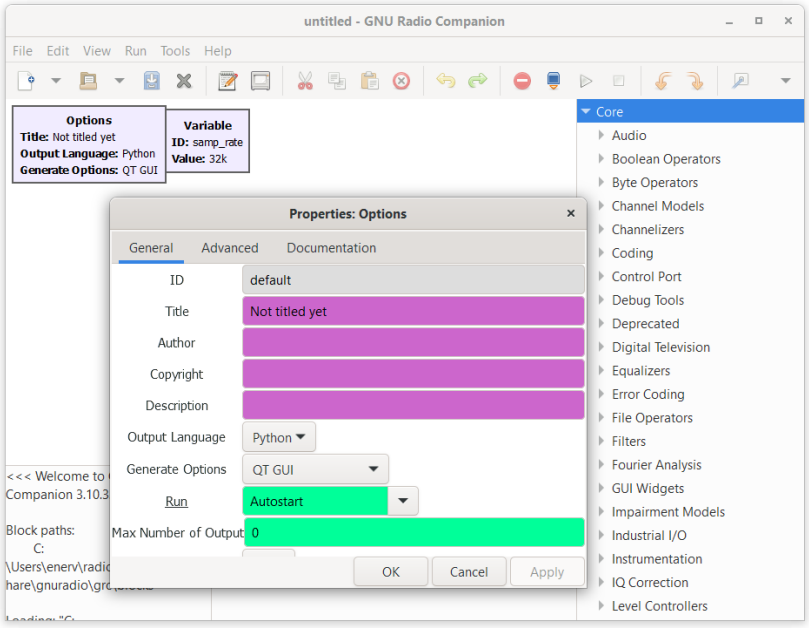
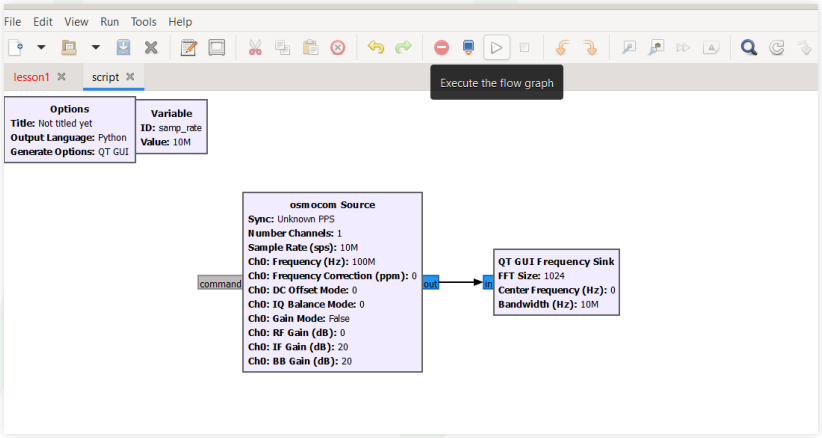


Figure 2. Un nouveau projet dans GNU Radio Companion (GRC) a d'emblée deux blocs. Un double-clic sur un bloc ouvre ses propriétés.



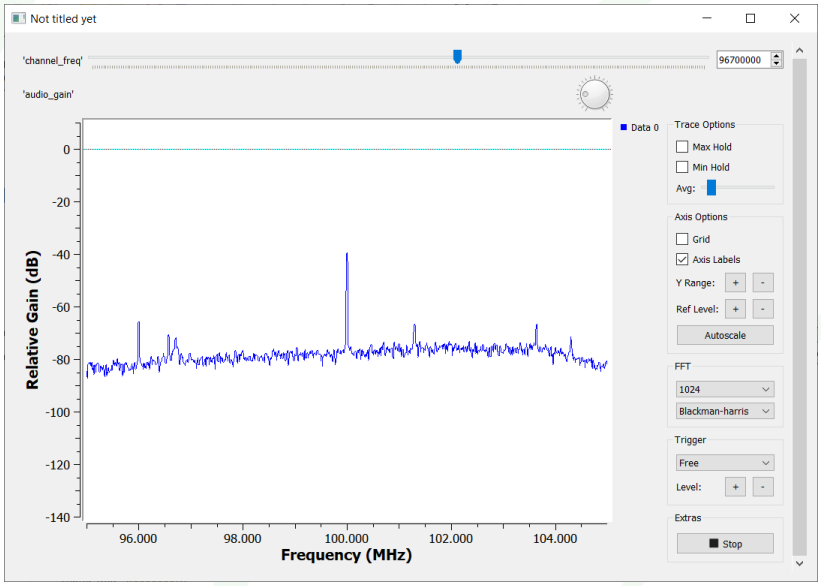
▲ *Figure 3. Cliquez sur le bouton Play pour exécuter votre premier graphique de flux.*

L'adaptateur OsmoSDR n'existe plus, mais le bloc supporte aussi d'autres matériels SDR, y compris HackRF One. Double-cliquer dessus ouvre ses propriétés, et nous voyons que le taux d'échantillonnage est défini par `samp_rate`, le nom du bloc variable que GRC nous a donné. Nous n'avons qu'un seul paramètre à modifier ici, et c'est le gain RF du canal 0, qui est trop élevé pour les signaux radio FM qui ont tendance à être plutôt forts. Nous pouvons le mettre à zéro.

Analyseur de spectre

Pour voir rapidement si les choses fonctionnent, nous pouvons ajouter un bloc analyseur de spectre au canevas et le connecter à la sortie du bloc source RF. Ce bloc s'appelle le *QT GUI Frequency Sink* (« puits de fréquence ») et se trouve dans la section *Instrumentation* de GRC. La façon dont ces blocs sont nommés peut être un peu déroutante, ce qui les rend parfois difficiles à trouver.

Pour connecter l'analyseur de spectre à la source RF, il suffit de cliquer sur son entrée, puis sur la sortie de la source. L'inverse fonctionne également, de même que le dessin avec le bouton gauche de la souris enfoncé. Notez que vous ne pouvez connecter que des entrées et des sorties de la même couleur, bleue dans ce cas.



▲

Figure 4. L'activation du panneau de contrôle sur le QT GUI Frequency Sink donne accès à plusieurs contrôles.

Premier essai

Vous pouvez maintenant cliquer sur le bouton *Play* (avec le petit triangle) pour exécuter le diagramme de flux (**figure 3**). Si vous ne l'avez pas déjà fait, vous devez enregistrer votre canevas avant de pouvoir continuer. Si tout va bien, un graphique de fréquence (le spectre) s'ouvrira. En jouant avec l'antenne, vous devriez obtenir des résultats légèrement différents, mais ils seront peut-être difficiles à percevoir. Pour améliorer un peu les choses, nous devons configurer le graphique de fréquence.

Cliquez sur le bouton *Stop* (avec le petit carré) pour arrêter le graphique de flux. Double-cliquez sur le bloc *Frequency Sink* pour ouvrir ses propriétés et définir le moyennage sur moyen. Réglez la fréquence centrale sur la même valeur que la fréquence du canal 0 de la source *osmocom*, soit 100 MHz dans notre cas. Ensuite, cliquez sur l'onglet *Config* et réglez le Panneau de configuration sur *Yes*. Fermez la fenêtre des propriétés et cliquez à nouveau sur *Play*. Maintenant, le tracé de fréquence a un panneau de contrôle, et vous pouvez cocher la case *Max Hold* (**figure 4**). Le fait de tripoter l'antenne devrait devenir plus visible et si c'est le cas, vous savez que vous pouvez recevoir des signaux RF.

Plug 'n' Play !

Une chose agréable à noter est que nous n'avons rien eu à faire pour que le diagramme de flux parle à HackRF One. Il n'y a pas de pilotes à installer, pas de ports à sélectionner, c'est tout, *plug and play* !

Les variables sont pratiques

Nous venons de définir la fréquence centrale du *Frequency Sink* à la même valeur que celle de la source RF, ce qui signifie qu'elle est nécessaire à au moins deux endroits. Par conséquent, si nous la remplaçons par une variable globale, comme GRC l'a fait pour la fréquence d'échantillonnage, nous pouvons la définir à un seul endroit.

Copiez le bloc variable de *sample rate*, ouvrez ses propriétés et changez l'ID en `center_freq`. Définissez la valeur à 100 MHz, saisissez `center_freq` pour la fréquence du canal 0 de la source *osmocom*, et saisissez `center_freq` pour la fréquence centrale du *Frequency Sink*. Les deux paramètres sont maintenant connectés. Si nous modifions la valeur de la variable `center_freq`, elle sera également modifiée dans tous les autres blocs qui l'utilisent.

Mélanger

Pour réaliser quelque chose avec la radio logicielle, vous devez bien sûr avoir quelques connaissances sur le fonctionnement des radios. Ses algorithmes font les mêmes choses que les circuits électroniques d'une radio analogique. Ainsi, pour recevoir une station de radio FM, nous devons la syntoniser, la démoduler, la filtrer et la rendre audible.

L'accord peut être effectué à l'aide d'un mélangeur ou d'un multiplicateur de fréquence. Lorsqu'il est effectué avec des signaux en quadrature ou complexes, il permet de décaler les fréquences vers le haut et vers le bas. Dans GRC, les entrées et sorties bleues correspondent à des signaux complexes à virgule flottante qui conviennent au mélangeur.

Le but est de décaler la fréquence d'intérêt vers le centre du diagramme de fréquence, d'où elle est plus facile à extraire. Pour se syntoniser sur une station de radio, nous avons donc besoin d'un multiplicateur et d'un autre signal pour contrôler le décalage de fréquence. Un bloc *Multiplier* est disponible dans la section *Math Operators*. Pour le signal d'accord, nous pouvons utiliser le bloc *Signal* Source de la section *Waveform Generators*.

Syntoniser une station

Les stations de radio FM ont une fréquence connue, que nous appellerons la fréquence du canal. Comme nous voulons pouvoir la changer facilement, nous créons pour elle une variable globale que nous appelons `channel_freq`.

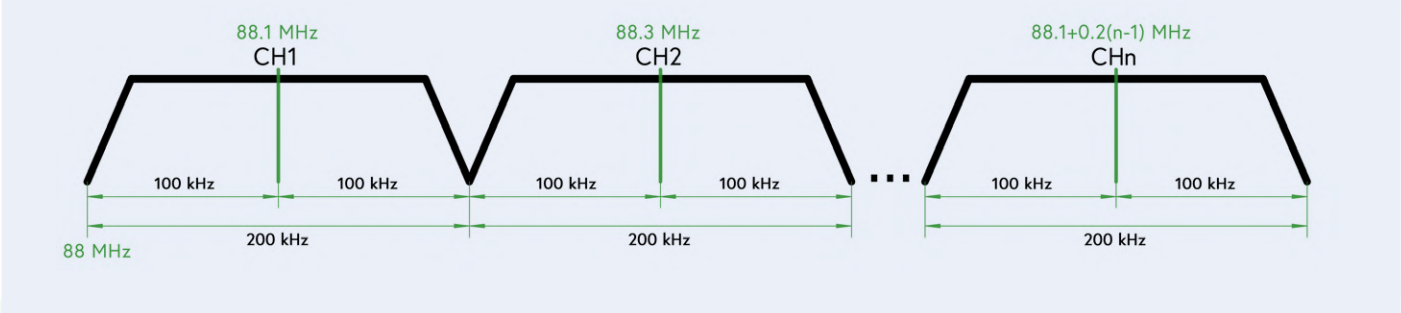
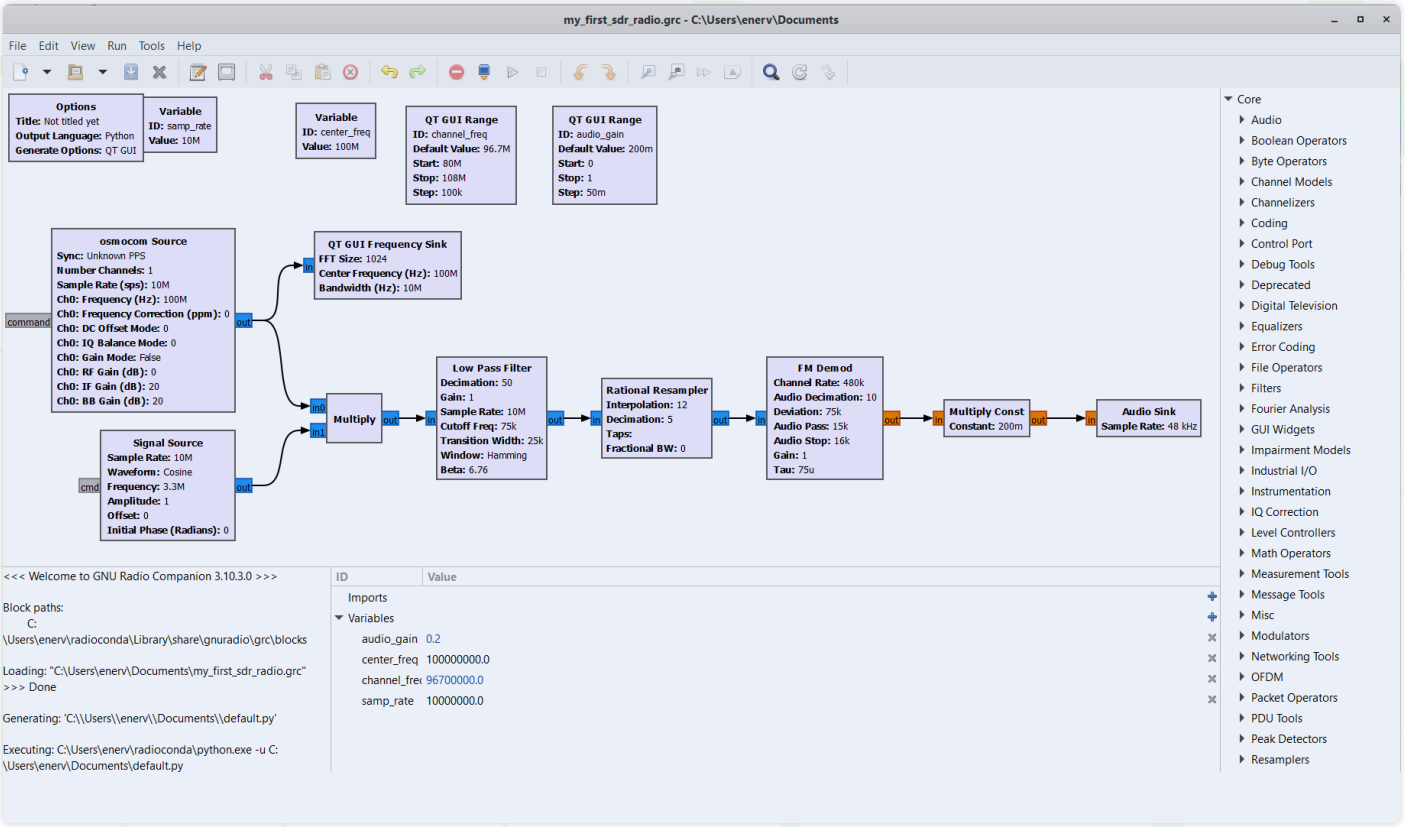


Figure 5. La bande de fréquences radio FM est divisée en canaux de 200 kHz de largeur.



▲

Figure 6. Le diagramme de flux du récepteur radio FM terminé avec les commandes de syntonisation et de volume. Lorsque vous l'exécuterez, il ressemblera à la figure 4.

Là où je vis, il y a une station de radio à 96,7 MHz, donc je mets cette valeur dans `channel_freq`.
Le mélangeur doit décaler la fréquence du canal vers la fréquence centrale, donc la fréquence de la source du signal doit être la différence des deux. La différence entre 100 MHz et 96,7 MHz est de 3,3 MHz et c'est bien la fréquence indiquée dans le bloc *Signal Source*.

Conversion vers le bas

Avec la station à la fréquence centrale, nous devons extraire son signal et l'abaisser pour qu'il puisse être démodulé. Extraire signifie filtrer et abaisser en SDR correspond à réduire ou décimer la fréquence d'échantillonnage. Le *Low Pass Filter* (filtre passe-bas) de la section *Filters* combine les deux opérations en un seul bloc.
La bande FM est divisée en canaux. La largeur varie selon les pays, mais se situe généralement autour de 200 kHz. Une station de radio se trouve au milieu d'un canal, ce qui signifie qu'elle dispose d'une bande passante de 100 kHz de chaque côté (**figure 5**). La fréquence d'échantillonnage minimale requise pour un signal de 100 kHz est de 200 kHz (théorème de Nyquist-Shannon). Notre fréquence d'échantillonnage est de 10 MHz, et nous fixons donc la valeur de décimation du bloc filtre à 10 MHz / 200 kHz, soit 50. Pour extraire le signal d'intérêt, nous n'avons pas besoin d'un filtre super-puissant. Une fréquence de coupure de 75 kHz avec une largeur de transition de 25 kHz convient pour une bande passante totale de 100 kHz.

Démodulation

Nous avons maintenant un signal en quadrature modulé en FM avec une fréquence d'échantillonnage de 200 kHz que nous voulons démoduler. Pour cela, nous pouvons utiliser le bloc *FM Demod* de la section *Modulators*.
La sortie du démodulateur est un signal audio que nous pouvons envoyer vers un récepteur audio, la carte son de l'ordinateur dans notre cas.

Questions relatives à la fréquence d'échantillonnage

Le problème que nous rencontrons maintenant est que les fréquences d'échantillonnage sont toutes différentes. La carte son (*Audio Sink, un « puits de son »*) supporte plusieurs taux d'échantillonnage qui dépendent de votre carte son. Sur mon ordinateur, le rapport entre les taux d'échantillonnage audio possibles et le taux d'échantillonnage d'entrée de 200 kHz est une valeur non entière dans tous les cas. Or, le démodulateur FM ne peut décimer que par une valeur entière. Alors, comment faire pour que cela soit correct ?
La solution consiste à insérer un convertisseur de fréquence d'échantillonnage capable d'effectuer une conversion fractionnelle. GRC propose deux options pour cela : un rééchantillonneur fractionnel et un rééchantillonneur rationnel. Comme nos taux d'entrée et de sortie sont des valeurs entières, nous pouvons utiliser le rééchantillonneur rationnel de la section *Resamplers*. Un rééchantillonneur fractionnel peut également être utilisé, mais il consomme beaucoup plus de ressources de calcul précieuses.
Pour obtenir la meilleure qualité audio, nous fixons la fréquence d'échantillonnage audio à 48 kHz. Ensuite, si nous faisons travailler le démodulateur à, disons, dix fois cette fréquence, à 480 kHz, nous avons besoin d'une conversion de fréquence d'échantillonnage de 200 : 480. Ce rapport peut être simplifié en trouvant le plus grand diviseur commun des deux valeurs, qui est 40. 200 : 40 = 5, et 480 : 40 = 12, donc si nous commençons par sur-échantillonner ou interpoler par 12 et ensuite

par sous-échantillonner ou décimer par cinq, un taux de 200 kHz est transformé en un taux de 480 kHz. Nous devons définir ce taux dans le bloc *FM Demod*, ainsi qu'un facteur de décimation de dix, pour nous assurer que le taux de sortie correspond au taux d'entrée du récepteur audio.

Ajouter les contrôles de l'utilisateur

Si vous exécutez ce diagramme de flux avec la fréquence du canal réglée sur une station de radio FM existante à proximité, vous devriez maintenant entendre le programme radio. C'est cool, non ?

Nous pouvons améliorer un peu notre radio en ajoutant une commande de syntonisation, afin que vous puissiez facilement syntoniser une autre station de radio. Pour cela, nous pouvons utiliser un bloc *QT GUI Range* de la section GUI Widgets. Définissez son ID sur `channel_freq`. Remplissez les paramètres en vous assurant qu'ils sont tous dans la plage voulue. Notez qu'il est inutile de définir une (très) petite valeur de pas, car la plupart des stations FM sont sur des fréquences MHz avec une seule décimale, 100 kHz conviendra probablement.

Vous pouvez choisir une forme pour le contrôle, mais *Counter + Slider* est le plus pratique parce qu'il vous permet de voir la fréquence comme une valeur numérique. De plus, vous pouvez taper une fréquence directement dans la boîte. Renommez ou supprimez l'ancien bloc variable `channel_freq`, car il n'est plus nécessaire et entre en conflit avec le *slider*. Vous pouvez maintenant régler la radio comme n'importe quelle autre radio.

Couleurs des signaux

Une autre amélioration consiste à ajouter un contrôle du volume. Il s'agit d'un contrôle similaire à celui de l'accord, et vous pouvez le copier et ajuster ses paramètres. Renommez l'ID en `audio_gain`. Choisissez une *Const Multiply* dans la section *Math Operators* et insérez-la entre le bloc *FM Demod* et l'*Audio Sink*. Notez que les fils sont devenus rouges. C'est parce que l'entrée et la sortie du multiplicateur sont bleues au lieu d'être orange. Ils ne sont pas du même type.

Ouvrez les propriétés du multiplicateur et définissez le *Type IO* sur *float*. Saisissez également `audio_gain` dans le champ intitulé *Constant* pour le connecter au curseur. Fermez le bloc et remarquez que les fils sont redevenus noirs, car toutes les entrées et sorties audio sont maintenant orange.

Options de sortie

Par défaut, la sortie de GRC est un script Python avec une interface graphique telle que définie par les options de génération dans le bloc Options. Mais il existe d'autres possibilités, comme C++. Si vous supprimez tous les blocs GUI du diagramme de flux et que vous définissez *Generate Options* sur *No GUI*, le script peut être exécuté en dehors de GRC. Cela vous permet de construire des applications SDR autonomes.

Voilà, c'est tout. Exécutez le diagramme de flux et profitez de votre récepteur radio FM défini par logiciel ! Pour approfondir le monde fascinant de la radio logicielle, je vous recommande vivement de regarder l'excellent cours vidéo (en anglais) de Michael Ossmann [6] qui a inspiré cet article.

220659-04 — VF : Maxime Valens

Questions ou commentaires ?

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

Produits

- **Great Scott Gadgets HackRF One Software Defined Radio (1 MHz à 6 GHz) (SKU 18306)**
  www.elektor.fr/18306
- **Great Scott Gadgets ANT500 antenne télescopique (75 MHz à 1 GHz) (SKU 18481)**
  www.elektor.fr/18481
- **Elektor SDR kit de démarrage (SKU 19041)**
  www.elektor.fr/19041

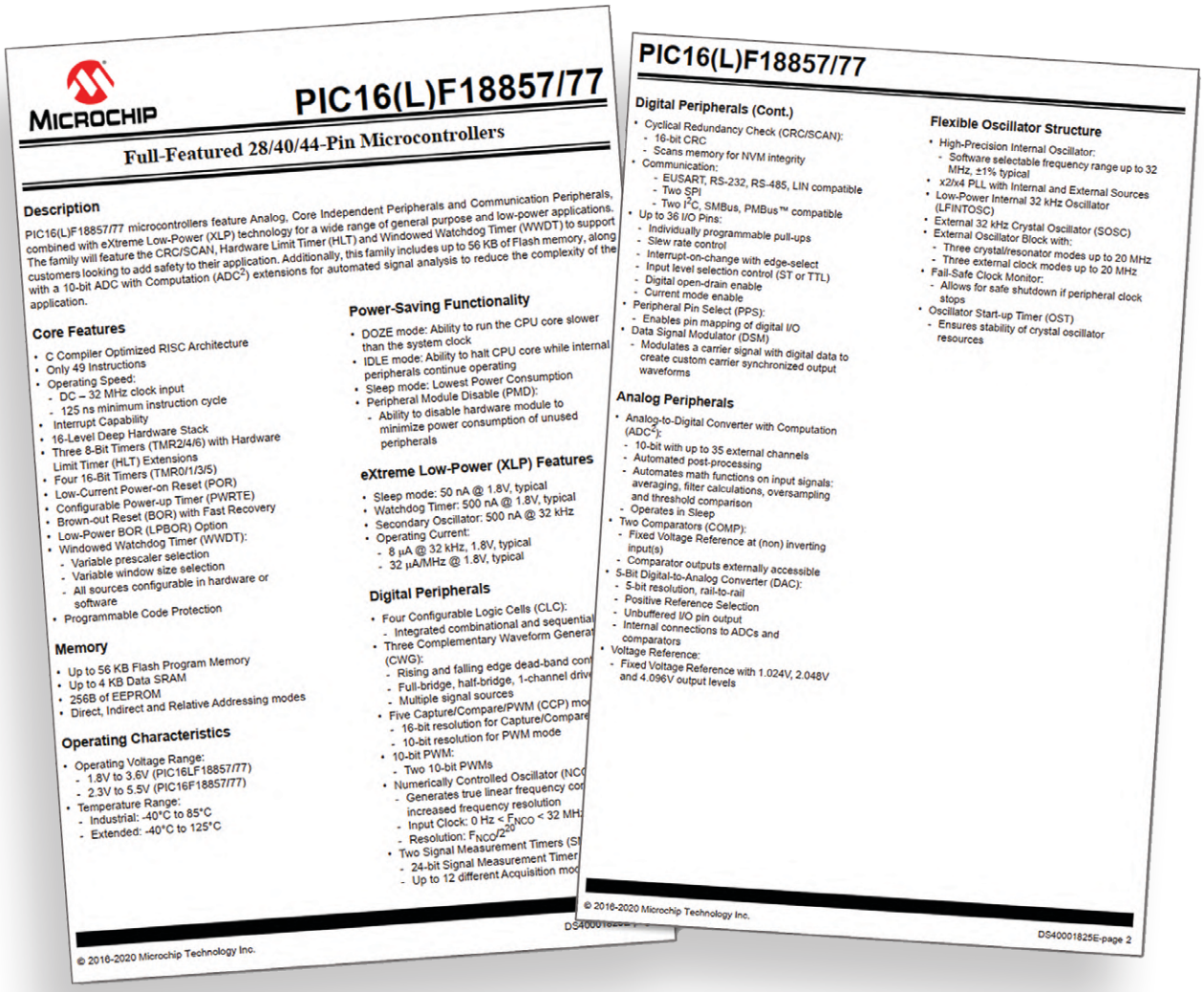
elektor TV
Récepteur radio FM basé sur la SDR chez Elektor.TV

LIENS

[1] HackRF One de Great Scott Gadgets : https://greatscottgadgets.com/hackrf/
[2] GNU Radio et GRC : https://www.gnuradio.org/
[3] SDR# alias SDR Sharp : https://airspy.com/download/
[4] SDR++ : https://www.sdrpp.org/
[5] HackRF One sur GitHub : https://github.com/greatscottgadgets/hackrf
[6] Cours vidéo sur la SDR par Michael Ossmann : https://greatscottgadgets.com/sdr/

la documentation des microcontrôleurs sans peine (1)

la structure d'une fiche technique



MICROCHIP

PIC16(L)F18857/77

Full-Featured 28/40/44-Pin Microcontrollers

Description

PIC16(L)F18857/77 microcontrollers feature Analog, Core Independent Peripherals and Communication Peripherals, combined with eXtreme Low-Power (XLP) technology for a wide range of general purpose and low-power applications. The family will feature the CRC/SCAN, Hardware Limit Timer (HLT) and Windowed Watchdog Timer (WWDT) to support customers looking to add safety to their application. Additionally, this family includes up to 56 KB of Flash memory, along with a 10-bit ADC with Computation (ADC^2) extensions for automated signal analysis to reduce the complexity of the application.

Core Features

- C Compiler Optimized RISC Architecture
- Only 49 Instructions
- Operating Speed:
  - DC – 32 MHz clock input
  - 125 ns minimum instruction cycle
- Interrupt Capability
- 16-Level Deep Hardware Stack
- Three 8-Bit Timers (TMR2/4/6) with Hardware Limit Timer (HLT) Extensions
- Four 16-Bit Timers (TMR0/1/3/5)
- Low-Current Power-on Reset (POR)
- Configurable Power-up Timer (PWRTE)
- Brown-out Reset (BOR) with Fast Recovery
- Low-Power BOR (LPBOR) Option
- Windowed Watchdog Timer (WWDT):
  - Variable prescaler selection
  - Variable window size selection
  - All sources configurable in hardware or software
- Programmable Code Protection

Memory

- Up to 56 KB Flash Program Memory
- Up to 4 KB Data SRAM
- 256B of EEPROM
- Direct, Indirect and Relative Addressing modes

Operating Characteristics

- Operating Voltage Range:
  - 1.8V to 3.6V (PIC16LF18857/77)
  - 2.3V to 5.5V (PIC16F18857/77)
- Temperature Range:
  - Industrial: -40°C to 85°C
  - Extended: -40°C to 125°C

Power-Saving Functionality

- DOZE mode: Ability to run the CPU core slower than the system clock
- IDLE mode: Ability to halt CPU core while internal peripherals continue operating
- Sleep mode: Lowest Power Consumption
- Peripheral Module Disable (PMD):
  - Ability to disable hardware module to minimize power consumption of unused peripherals

eXtreme Low-Power (XLP) Features

- Sleep mode: 50 nA @ 1.8V, typical
- Watchdog Timer: 500 nA @ 1.8V, typical
- Secondary Oscillator: 500 nA @ 32 kHz
- Operating Current:
  - 8 µA @ 32 kHz, 1.8V, typical
  - 32 µA/MHz @ 1.8V, typical

Digital Peripherals

- Four Configurable Logic Cells (CLC):
  - Integrated combinational and sequential
- Three Complementary Waveform Generat (CWG):
  - Rising and falling edge dead-band con
  - Full-bridge, half-bridge, 1-channel driv
  - Multiple signal sources
- Five Capture/Compare/PWM (CCP) mo
  - 16-bit resolution for Capture/Compare
  - 10-bit resolution for PWM mode
- 10-bit PWM:
  - Two 10-bit PWMs
- Numerically Controlled Oscillator (NC
  - Generates true linear frequency co increased frequency resolution
  - Input Clock: 0 Hz < F_{NCO} < 32 MH
  - Resolution: $F_{NCO}/2^{20}$
- Two Signal Measurement Timers (S
  - 24-bit Signal Measurement Timer
  - Up to 12 different Acquisition mo

© 2016-2020 Microchip Technology Inc.

DS40001825E-page 2

PIC16(L)F18857/77

Digital Peripherals (Cont.)

- Cyclical Redundancy Check (CRC/SCAN):
  - 16-bit CRC
  - Scans memory for NVM integrity
- Communication:
  - EUSART, RS-232, RS-485, LIN compatible
  - Two SPI
  - Two I^2C, SMBus, PMBus™ compatible
- Up to 36 I/O Pins:
  - Individually programmable pull-ups
  - Slew rate control
  - Interrupt-on-change with edge-select
  - Input level selection control (ST or TTL)
  - Digital open-drain enable
  - Current mode enable
- Peripheral Pin Select (PPS):
  - Enables pin mapping of digital I/O
- Data Signal Modulator (DSM)
  - Modulates a carrier signal with digital data to create custom carrier synchronized output waveforms

Analog Peripherals

- Analog-to-Digital Converter with Computation (ADC^2):
  - 10-bit with up to 35 external channels
  - Automated post-processing
  - Automates math functions on input signals: averaging, filter calculations, oversampling and threshold comparison
  - Operates in Sleep
- Two Comparators (COMP):
  - Fixed Voltage Reference at (non) inverting input(s)
  - Comparator outputs externally accessible
- 5-Bit Digital-to-Analog Converter (DAC):
  - 5-bit resolution, rail-to-rail
  - Positive Reference Selection
  - Unbuffered I/O pin output
  - Internal connections to ADCs and comparators
- Voltage Reference:
  - Fixed Voltage Reference with 1.024V, 2.048V and 4.096V output levels

Flexible Oscillator Structure

- High-Precision Internal Oscillator:
  - Software selectable frequency range up to 32 MHz, ±1% typical
- x2/x4 PLL with Internal and External Sources
- Low-Power Internal 32 kHz Oscillator (LFINTOSC)
- External 32 kHz Crystal Oscillator (SOSC)
- External Oscillator Block with:
  - Three crystal/resonator modes up to 20 MHz
  - Three external clock modes up to 20 MHz
- Fail-Safe Clock Monitor:
  - Allows for safe shutdown if peripheral clock stops
- Oscillator Start-up Timer (OST)
  - Ensures stability of crystal oscillator resources

Figure 1. Les premières pages énumèrent les fonctionnalités du microcontrôleur. (Source : Microchip Technology)

Stuart Cording (Elektor)

Que vous aimiez ou non la documentation, vous aurez à négocier pour l'utiliser. Les microcontrôleurs ont une documentation volumineuse, car, par rapport à d'autres dispositifs à semi-conducteurs plus simples, ils sont vraiment complexes. Dans cette série d'articles, nous allons examiner la documentation des microcontrôleurs, ce qui figure dans la fiche technique, ce qui n'y figure pas et où trouver les éléments détaillés manquants.

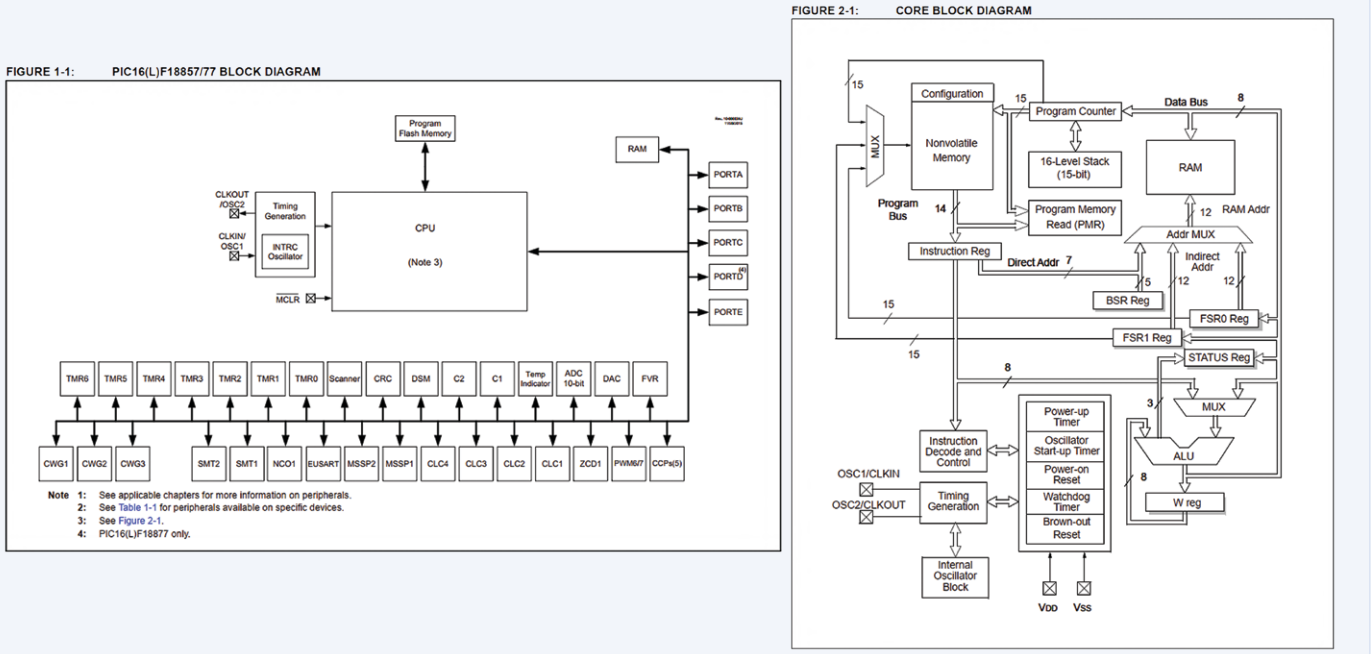


Figure 2. Le schéma de principe est un bon point de départ pour déterminer les capacités globales du microcontrôleur (à gauche) et parfois du cœur de traitement (à droite). (Source : Microchip Technology)

Les fiches techniques des microcontrôleurs peuvent facilement atteindre aujourd'hui plus de 600 pages. Heureusement, Elektor propose de nombreuses ressources, notamment des articles pour débutants et des livres pour vous mettre le pied à l'étrier. Cependant, à un moment donné, vous allez devoir vous familiariser avec la véritable documentation du microcontrôleur. Bien qu'une fiche technique de microcontrôleur compte de nombreuses pages, il est peu probable que vous y trouviez toutes les informations que vous recherchez.

Pour la compléter, vous devrez également trouver la documentation des outils chargés de transformer le code source en micrologiciel, mais aussi des outils de développement de code, de débogage et de programmation en série. Si l'univers des microcontrôleurs est une nouveauté pour vous, ce guide vous aidera à comprendre comment accéder aux documents nécessaires, de quelle manière décoder les arcanes de leur contenu et où trouver des informations s'il y avait des erreurs dans ce que vous avez lu.

Exemple de documentation sur les microcontrôleurs : PIC16F18877

Pour entamer la première partie de cette série de trois articles sur la documentation des microcontrôleurs, commençons par un microcontrôleur 8 bits simple tel que le PIC16F18877 de Microchip Technology, et sa documentation. Ouvrez ce lien [1] et cliquez ensuite sur « Documents ». Vous verrez que l'on nous propose la fiche technique, la section « errata », des documentations techniques (*Supporting Collateral*) à propos d'articles spécifiques pour certaines applications et de cartes d'évaluation, les spécifications de programmation et une longue liste de notes d'application. Vous y trouverez aussi un code source accompagnant certaines des notes d'application, des brochures commerciales et un livre blanc sur les convertisseurs analogiques-numériques (CA/N).

Que contient la fiche technique du microcontrôleur ?

Nous allons commencer par télécharger la fiche technique du PIC16F18877. Les fiches techniques des microcontrôleurs peuvent être assez décourageantes. Surtout, elles ne contiennent pas nécessairement tout ce que vous avez à savoir. La **figure 1** montre ce que vous devriez au minimum y trouver.

- Informations détaillées sur un ou plusieurs microcontrôleurs — Souvent, plusieurs microcontrôleurs différents avec un nombre variable de broches et de boîtiers seront créés à partir d'une seule puce de silicium. Plutôt que de publier et de maintenir une documentation pour chaque variante, il est plus probable de trouver plusieurs dispositifs couverts par une seule fiche technique. C'est le cas ici, comme expliqués page 1, avec deux variantes abordées : le PIC16F18857 et le PIC16F18877.
- Schéma de principe du microcontrôleur — Il comprend généralement le cœur de traitement (le niveau de détail varie ; plus le cœur est complexe, plus son implémentation schématique est simple), les mémoires, les bus et les périphériques. À partir de là, vous pouvez rapidement déterminer les capacités de base du microcontrôleur. Le schéma de principe du microcontrôleur (**figure 2**) se trouve page 18, tandis que celui du cœur de traitement figure page 33.
- Options de boîtiers — Elles vont des types pour trous traversants (s'ils sont encore disponibles) à une gamme d'options de montage en surface. Dans cet exemple, elles commencent à la page 4.
- Options de taille de mémoire — Le microcontrôleur peut être proposé avec différentes tailles de mémoires RAM, flash, EEPROM et tout autre type de mémoire, comme les caches. Dans notre exemple, nous avons un aperçu rapide à la page 3. Le tableau énumère les tailles de mémoire ainsi que le nombre de chaque périphérique implémenté (**figure 3**).

PIC16(L)F188XX Family Types

Device	Data Sheet Index	Program Flash Memory (Words)	Program Flash Memory (KB)	EEPROM (bytes)	Data SRAM (bytes)	I/O Pins(1)	10-Bit ADC2 (ch)	5-Bit DAC	Comparator	8-Bit (with HLT)/ 16-Bit Timers	SMT	Windowed Watchdog Timer	CRC and Memory Scan	CCP/10-Bit PWM	Zero-Cross Detect	CWG	NCO	CLC	DSM	EUSART/I2C/SPI	Peripheral Pin Select	Peripheral Module Disable
PIC16(L)F18854	(1)	4096	7	256	512	25	24	1	2	3/4	2	Y	Y	5/2	Y	3	1	4	1	1/2	Y	Y
PIC16(L)F18855	(2)	8192	14	256	1024	25	24	1	2	3/4	2	Y	Y	5/2	Y	3	1	4	1	1/2	Y	Y
PIC16(L)F18856	(3)	16384	28	256	2048	25	24	1	2	3/4	2	Y	Y	5/2	Y	3	1	4	1	1/2	Y	Y
PIC16(L)F18857	(4)	32768	56	256	4096	25	24	1	2	3/4	2	Y	Y	5/2	Y	3	1	4	1	1/2	Y	Y
PIC16(L)F18875	(2)	8192	14	256	1024	36	35	1	2	3/4	2	Y	Y	5/2	Y	3	1	4	1	1/2	Y	Y
PIC16(L)F18876	(3)	16384	28	256	2048	36	35	1	2	3/4	2	Y	Y	5/2	Y	3	1	4	1	1/2	Y	Y
PIC16(L)F18877	(4)	32768	56	256	4096	36	35	1	2	3/4	2	Y	Y	5/2	Y	3	1	4	1	1/2	Y	Y

Figure 3. Jetez un coup d'œil aux options de mémoire pour les variantes du PIC16F18x7, ainsi qu'aux périphériques implémentés. (Source : Microchip Technology)

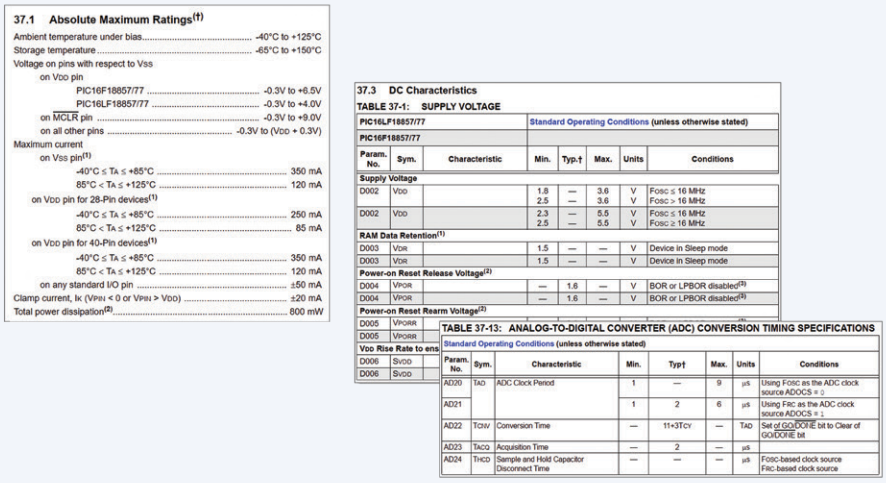


Figure 4. Les spécifications électriques sont fournies sous forme de limites absolues, mais aussi de valeurs minimales, typiques et maximales pour les valeurs en courant continu et alternatif, et de limites de temps. (Source : Microchip Technology)

- Schéma de principe des périphériques sur puce — Les fonctionnalités des périphériques sur puce s'expliquent plus facilement par des schémas que par des mots. Les schémas de principe forment une source essentielle de compréhension et de clarté concernant le raccordement des broches et les sources d'horloge. Le microcontrôleur peut disposer de plusieurs sorties d'horloge issues de son oscillateur et, le cas échéant, d'une PLL (boucle à verrouillage de phase). Nous examinerons quelques exemples dans la deuxième partie de cette série.
- Description des registres — Chaque périphérique peut être configuré de manière spécifique. Par exemple, une UART (interface série universelle asynchrone d'émission/réception) peut souvent être configurée pour différents débits en bauds, nombre de bits, etc. La description du registre explique comment configurer le périphérique et vérifier son état suite à un événement, comme la réception d'un octet de données. Le décryptage de ces registres est abordé plus loin dans la deuxième partie de la série.
- Spécifications électriques — Elles informent l'utilisateur des limites de tension et de courant qui peuvent être appliquées aux broches du microcontrôleur ou tirées de celles-ci. Elles sont généralement définies deux fois : une fois en tant que valeurs maximales absolues, et une autre fois en tant que valeurs minimales, typiques et maximales en fonctionnement normal. Il existe également des caractéristiques de synchronisation, comme le montre la **figure 4**, et vous pouvez les trouver à partir de la page 592 de cette fiche technique.
- Recommandations d'outils — Vous aurez besoin d'outils pour convertir le code source en assembleur et d'outils de débogage des résultats. La plupart des fiches techniques contiennent sous une forme ou sous une autre des recommandations sur les outils disponibles. Elles sont abordées ici page 638.

Au-delà des schémas de principe

Maintenant que la structure de base d'une fiche technique de microcontrôleur est comprise, nous allons examiner comment les registres sont décrits et de quelle manière déchiffrer les schémas de principe. Nous allons également aborder en détail les deux blocs les plus importants de tout microcontrôleur - l'horloge et l'oscillateur - et la mise en œuvre du circuit de réinitialisation.

200721-04 — VF : Pascal Godart

Des questions, des commentaires ?

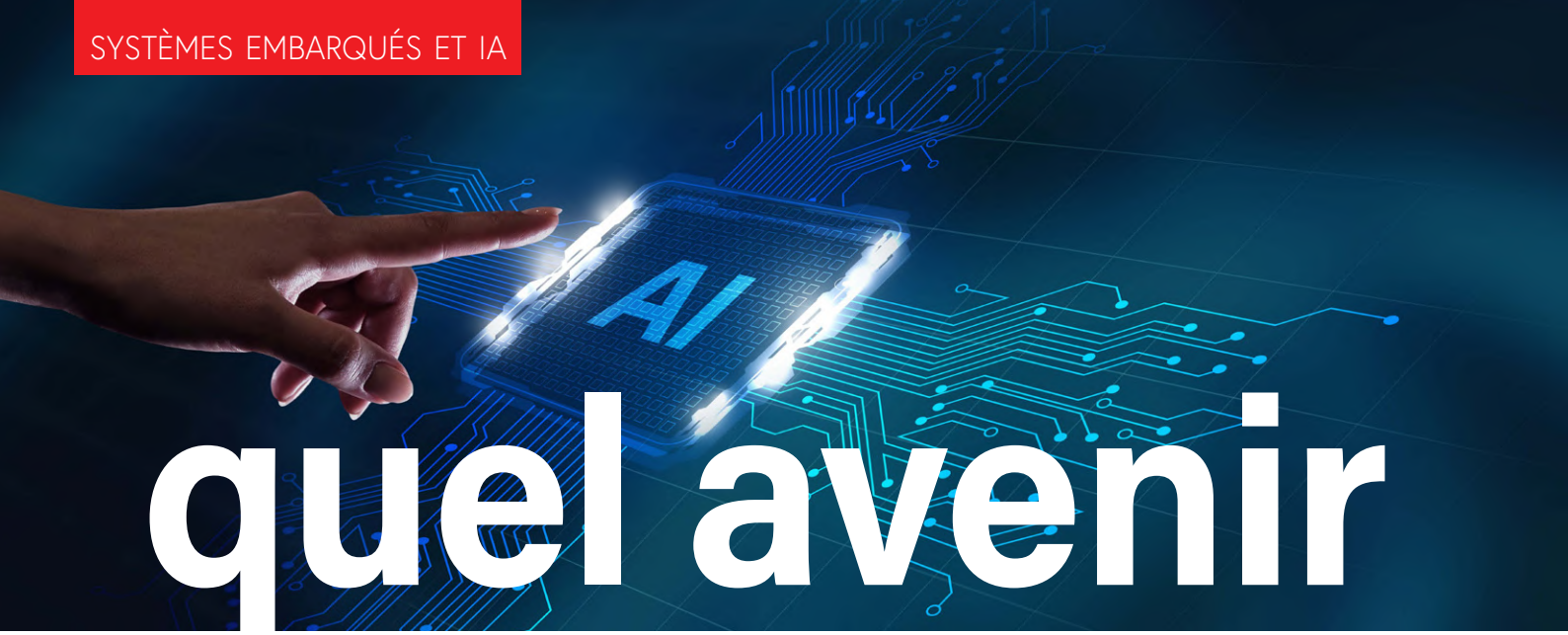
Envoyez un courriel à l'auteur (stuart.cording@elektor.com).

LIENS

[1] Page du produit PIC16F18877 : http://bit.ly/2KS1s8C
[2] Fiche technique du PIC16F18877 : https://bit.ly/3nNwPjn

Produits

- **Livre en anglais « Microcontroller Basics with PIC », T. Hanna, Elektor 2020**
  www.elektor.fr/19188
- **Livre en anglais « Programming the Finite State Machine », A. Pratt, Elektor 2020**
  www.elektor.fr/19327



quel avenir pour l'IA et les systèmes embarqués ?

outils, plateformes et remplacement des rédacteurs

Stuart Cording (Elektor)

L'intelligence artificielle (IA) est souvent présentée comme une arme universelle pour résoudre différents problèmes, quel que soit le défi à relever. Parfois aussi, elle est présentée comme annonçant la fin de la civilisation elle-même. En tant qu'ingénieurs, nous savons qu'aucune de ces deux affirmations n'est vraie. Pour autant, comment pouvons-nous mieux utiliser l'IA, ou plus exactement l'apprentissage automatique, dans les applications que nous développons ? Et les progrès les plus récents de l'IA feront-ils vraiment de nous des intervenants superflus ?

Mentionner l'IA est le plus sûr moyen d'attirer l'attention de la presse grand public. Grâce à l'internet et aux services dans le nuage, mais aussi du fait de sources de données parfois douteuses, de nouveaux services basés sur l'IA semblent surgir partout. En quelques clics de souris, vos propres mots peuvent être placés dans la bouche de Morgan Freeman ou votre image intégrée dans une nouvelle œuvre d'art. Et même si cela peut être amusant et donner à réfléchir, il n'est pas facile de trouver un lien avec le monde des systèmes embarqués. Voyons précisément ce qui se passe dans ce secteur.

Pour fonctionner, la plupart des systèmes embarqués utilisent des approches de programmation à base de règles. Pour un ensemble de données d'entrée défini, une série d'instructions *if/else* ou *switch* détermine la réponse à donner. Le principe fonctionne bien pour un nombre limité d'entrées. Pourtant, à un moment, le nombre d'entrées ou la subtilité de leurs relations rend difficile la définition de règles programmatiques claires.

À titre d'exemple, imaginez un moteur installé dans une usine, et fonctionnant 24 heures sur 24, sept jours sur sept. L'expérience montre qu'avec le temps, il se produit une usure et un épaississement du lubrifiant des roulements. Au fil du temps, cela modifie la durée de rotation, le bruit produit, les modes de vibration, la température de fonctionnement du moteur et le courant consommé. Assurer une maintenance périodique permet de résoudre cette problématique, même si elle entraîne des temps d'arrêt fixes préjudiciables à la production. Il est souvent difficile de dire si la fréquence est trop importante ou trop faible.

En outre, il est peu probable de détecter une défaillance imminente résultant d'une fracture capillaire de l'arbre, des roulements, du boîtier ou des fixations. Une fois correctement entraînées, les approches par IA peuvent déterminer les défaillances en cours en utilisant un mélange complexe de sources de données. S'il est possible de déployer cette intelligence dans un système à base de microcontrôleur, vous obtenez un dispositif de surveillance abordable qui permet de gagner du temps et de diminuer les dépenses, mais aussi de réduire le gaspillage dû aux opérations de lubrification inutiles et au remplacement superflu des pièces.

Introduire de l'intelligence dans les microcontrôleurs

Concernant les microcontrôleurs, il s'agit davantage d'apprentissage automatique (ou *machine learning* - ML) plutôt que d'IA. L'objectif est ici de prendre des décisions en programmant une machine pour qu'elle utilise des règles élaborées grâce à l'analyse des données disponibles. Si les microcontrôleurs disposent d'une puissance plus que suffisante pour exécuter de tels algorithmes de ML, l'apprentissage à partir de données d'entraînement reste hors de portée pour eux

et nécessite au moins un ordinateur de bureau, voire un serveur cloud. Fondée en 2019, l'entreprise Edge Impulse a développé une plateforme dédiée au ML dans les systèmes embarqués et s'est associée avec succès aux fournisseurs de semi-conducteurs du monde entier [1] pour assurer une prise en charge très large. Les données constituent le point de départ de toute application de ML. Alors que certaines applications, la conduite autonome par exemple, nécessitent des téraoctets de données d'entraînement, les systèmes simples basés sur des microcontrôleurs peuvent apprendre à partir de quelques kilo-octets de données seulement. Le premier défi consiste donc à transférer les données depuis le microcontrôleur vers l'environnement Edge Impulse. L'idée initiale serait d'utiliser l'interface série d'un Arduino pour les envoyer à votre PC, puis à partir de là, les télécharger dans un fichier texte. La plateforme est cependant configurée pour ingérer les données directement.

Les données, carburant de l'IA

L'un de ces outils est le Data Forwarder [2], une application à ligne de commande (CLI) qui envoie les données d'une carte de développement directement à l'environnement Edge Impulse. Grâce à votre nom d'utilisateur et votre mot de passe, une liaison est établie entre le port série de votre PC et le serveur. Du côté du microcontrôleur, il suffit d'envoyer les données sur l'interface série dans un format délimité par des virgules ou des tabulations. Tant que le taux d'échantillonnage est relativement faible, il s'agit d'un moyen idéal pour recueillir des données représentatives directement à partir de vos capteurs (**figure 1**). Les systèmes embarqués plus puissants, comme le Raspberry Pi ou le NVIDIA Jetson Nano, peuvent utiliser le kit de développement logiciel (SDK) [3] fourni. Cela permet également de prendre en charge des capteurs comme les microphones et les caméras qui produisent des quantités plus importantes de données.

Une fois les données téléchargées, l'étape suivante consiste à définir une « impulsion », avec une répartition en deux blocs. Le premier découpe les données en petits morceaux et utilise des techniques de traitement du signal pour extraire des caractéristiques. Cela permet de s'assurer que les données des capteurs disponibles sont transformées en informations cohérentes pour la deuxième étape de traitement. Le deuxième bloc est celui où sont effectués l'apprentissage et la classification (**figure 2**). Dans un exemple de projet, « Reconnaissance d'un mouvement continu », on trouve une bonne explication de la façon dont ces blocs sont configurés pour analyser les données d'un accéléromètre et classer une entrée si elle correspond à l'un des quatre gestes [4].

Il s'agit de l'étape la plus critique d'un développement de ML. Elle nécessite souvent une démarche de réflexion latérale (sous plusieurs angles ou hors du champ habituel d'études), mais aussi différentes itérations pour déterminer la meilleure approche. Parfois, ignorer certaines entrées de capteurs est la meilleure solution, alors que dans d'autres cas, il est nécessaire de disposer de davantage de données. Vous constaterez éventuellement que vous êtes en situation de sur-apprentissage, ou que le modèle de réseau neuronal choisi est mal adapté à la classification que vous tentez d'effectuer. Une autre étape cruciale est la classification des anomalies. Dans l'exemple de projet, il y a quatre gestes définis. Cependant, il est nécessaire d'exclure d'autres mouvements similaires aux gestes

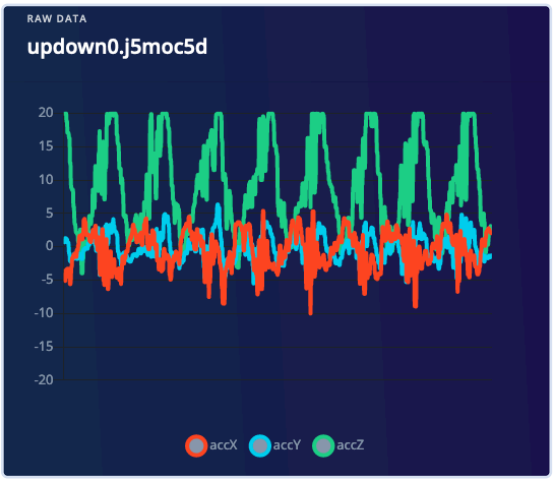


Figure 1. Un accéléromètre à trois axes produit des données relatives aux mouvements pour développer une application de reconnaissance des gestes basée sur le ML. (Source : Edge Impulse)

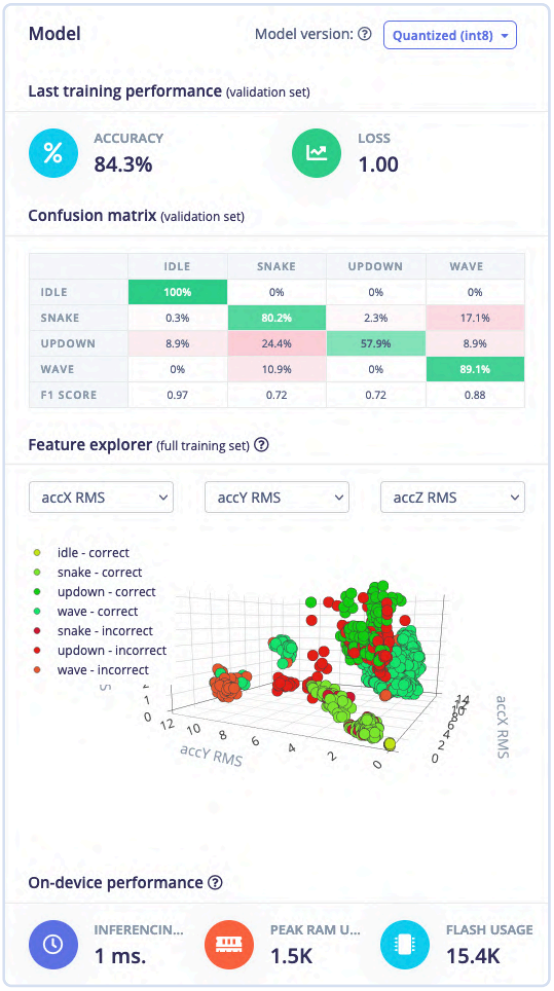


Figure 2. L'environnement Edge Impulse donne des informations sur la précision, la vitesse et l'utilisation de la mémoire après un cycle d'entraînement initial. (Source : Edge Impulse)

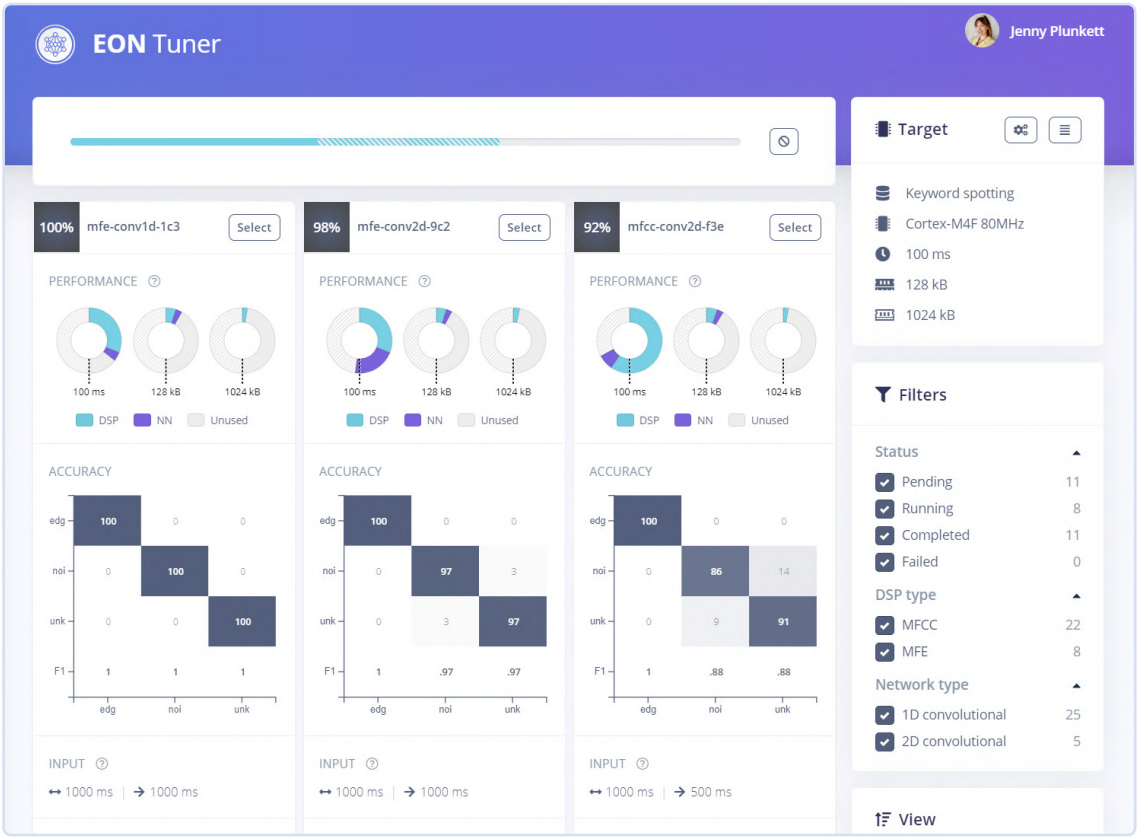


Figure 3. Les algorithmes de ML sont optimisés pour le microcontrôleur cible à l'aide d'EON Tuner, ce qui permet d'améliorer les temps de réponse des inférences et de réduire l'utilisation de la mémoire. (Source : Edge Impulse)

appris. Une bonne détection des anomalies permet d'obtenir un résultat de ML plus robuste.

L'étape finale est le déploiement. Par le biais de l'interface web, le micrologiciel de l'appareil choisi est téléchargé pour être intégré dans votre application. Pour Arduino, une bibliothèque est générée, tandis que pour les autres microcontrôleurs, vous pouvez générer un fichier C++. Bien entendu, les performances des microcontrôleurs varient de manière considérable. Pour garantir le meilleur résultat possible, Edge Impulse propose son EON Tuner [5]. Cet outil peut améliorer la précision de la détection, accélérer les inférences et réduire les besoins en mémoire en utilisant des informations comme le dispositif cible, la taille de la mémoire et la latence (**figure 3**).

Figure 4. SlateSafety a utilisé Edge Impulse pour assurer une surveillance physiologique ML de pointe et améliorer la surveillance de la sécurité du BAND V2. (Source : SafetySlate)

Le ML dans les applications réelles

Des applications réelles utilisent cette approche pour intégrer des capacités de ML. Le BAND [6] de SlateSafety intègre une série de capteurs biométriques pour surveiller les personnes qui travaillent dans des situations difficiles (**figure 4**). Cela concerne aussi bien les premiers secours que les personnes travaillant en milieu industriel, et portant des équipements de protection individuelle (EPI) lourds, comme les pompiers. Le produit télécharge normalement les données dans le cloud afin que leurs collègues puissent surveiller les signaux vitaux d'un utilisateur. Cependant, notamment dans les situations de catastrophe, la connectivité peut être inégale ou inexistante. L'équipe de développement a utilisé Edge Impulse pour intégrer le ML en périphérie dans le produit existant, entraîné sur des données biométriques historiques. Grâce à l'EON Tuner, l'algorithme a été optimisé pour le matériel, puis déployé à l'aide d'une mise à jour par ondes radio. Le BAND peut ainsi avertir le porteur en cas de risque d'épuisement dû à la chaleur, même en l'absence de connexion sans fil.

Améliorer le développement des produits grâce à l'IA

Bien entendu, l'IA ne doit pas nécessairement être intégrée au produit. Elle peut également servir à son développement. Aujourd'hui, de nombreuses applications complexes sont développées grâce à une approche fondée sur un modèle, qui consiste essentiellement à décrire le fonctionnement d'un produit à l'aide d'un logiciel et d'équations mathématiques issues de la physique. Cependant, même cette approche a ses limites. C'est là que Monolith et sa plateforme d'IA avec auto-apprentissage [6] entrent en jeu.

La plateforme est capable d'apprendre les propriétés physiques de systèmes complexes à partir de données déjà collectées. À titre d'exemple, les véhicules subissent une série de tests sur une piste d'essai, où de multiples capteurs surveillent le lacet et le roulis ainsi que la vitesse et l'accélération des roues. La collecte de données pour différentes rigidités de suspension donne un bon aperçu de la façon dont le véhicule réagit à une série de situations de conduite. En général, les données sont analysées, ce qui permet d'obtenir de nouveaux réglages à appliquer lors d'un autre essai. Monolith peut évaluer les données de la première série de tests et prédire le résultat des modifications de la suspension avec un haut degré de précision. Les résultats servent à affiner plus rapidement les meilleurs réglages de la suspension, réduisant ainsi le nombre d'autres essais physiques nécessaires.

Cette approche peut également s'appliquer à la métrologie. Les compteurs à gaz doivent être exceptionnellement précis pour garantir une facturation correcte, mais c'est difficile à réaliser lorsque le compteur doit effectuer des mesures pour différents gaz. Pour un client, la simulation de compteurs à ultrasons a poussé l'analyse purement mathématique à ses limites, laissant les processus d'essais consécutifs comme seule solution d'étalonnage pour obtenir la certification nécessaire. Fort heureusement, tous les essais ont donné lieu à une vaste collection de données à analyser. Grâce à l'utilisation de modèles d'IA avec apprentissage automatique, la quantité de tests nécessaires a diminué jusqu'à 70 %, ce qui a considérablement accéléré le développement.

Miniaturisation des capacités de calcul nécessaires à l'IA

Des concours comme le *DARPA Grand Challenge* [8], au cours duquel il s'agissait de construire des véhicules autonomes capables de suivre un parcours sinueux, ont déclenché une vague d'intérêt pour les voitures de ce type. Aujourd'hui, près de vingt ans plus tard, des budgets considérables ont été dépensés, avec des résultats mitigés. Tesla fait régulièrement les gros titres à ce sujet, souvent lorsque des propriétaires trop enthousiastes font preuve d'une confiance excessive dans les capacités de leur véhicule [9]. Par ailleurs, seul Waymo semble proposer de véritables véhicules à conduite autonome sous la forme de services de covoiturage [10], même s'ils ne sont proposés qu'à Phoenix et San Francisco aux États-Unis.

L'un des problèmes est l'extrême difficulté du pilotage d'une voiture par un ordinateur. Non seulement le véhicule doit évaluer en permanence la situation environnante, mais il doit également prévoir les actions des autres conducteurs et usagers, comme les piétons et les cyclistes, qui, le cas échéant, ne respectent pas le code de la route.

L'architecture électrique et électronique (E/E) des véhicules est en train d'évoluer pour répondre aux besoins futurs des véhicules autonomes. Grâce aux innombrables capteurs qui produisent de vastes quantités de données, l'industrie se tourne vers l'Ethernet automobile. Actuellement, cette approche est à l'origine des systèmes avancés d'aide à la conduite (ADAS) qui, en contrôlant le freinage, l'accélération et la direction, peuvent intervenir en cas d'erreur du conducteur. Selon les niveaux d'autonomie des véhicules définis par la *Society of Automotive Engineers* (SAE), les véhicules haut de gamme atteignent actuellement le niveau 2+, voire le niveau 3 pour certains. Toutefois, l'autonomie complète permettant au conducteur de « s'asseoir et se détendre » est de niveau 5, ce qui signifie que nous avons encore du chemin à parcourir.

Figure 5. Le développement d'algorithmes d'IA est accéléré par l'utilisation d'ordinateurs embarqués hautement performants, comme ce DYNACOR 40-36 à refroidissement liquide. (Source : Eurotech)

Des entreprises comme Eurotech soutiennent l'industrie pour accélérer le développement des algorithmes nécessaires. Actuellement, un essai routier de huit heures permet de recueillir 120 To de données qui doivent être renvoyées au laboratoire pour être traitées et analysées. Il est possible de tester en laboratoire les améliorations apportées aux algorithmes d'IA à l'aide des données recueillies, mais il existe peu de ressources pour faciliter les essais et le développement d'algorithmes sur le terrain.

Tirant parti de son expérience en matière de refroidissement liquide, Eurotech propose une série de matériels d'IA de pointe, essentiellement sous la forme de superordinateurs compacts qui tiennent dans le coffre d'un véhicule. Un ensemble durci comme le DYNACOR 40-36 peut s'installer dans des véhicules routiers ou non routiers [11]. Doté d'un CPU Intel Xeon à 16 cœurs avec 64 Go de RAM et jusqu'à deux GPU NVIDIA10 GV100 avec 32 Go de mémoire RAM, cet ordinateur sans ventilateur dispose de 237 TFLOPS pour traiter les applications d'apprentissage profond (**figure 5**). Plusieurs interfaces gigabit Ethernet permettent d'ingérer des quantités massives de données de capteurs, issues de différents dispositifs (radar, lidar, caméras, notamment), et de les stocker dans un système de stockage électronique offrant un volume de 32 To de stockage. En effectuant davantage de tests d'inférence et de renforcement au cours des essais routiers, la possibilité d'une autonomie de niveau 5 pourrait être considérablement accélérée.

Figure 6. Image produite par IA avec le système DALL-E 2 d'OpenAI d'un véhicule autonome doté de l'incontournable Commodore 64. Chantier en cours.

L'IA met-elle mon emploi en danger ?

Une discussion permanente se développe sur les médias sociaux autour de la question des risques qui pèseraient sur les emplois dans les industries créatives, du fait des progrès de l'IA. Le lancement de DALL-E 2 [12] par OpenAI transforme en images les demandes en langage naturel (**figure 6**). Mais, ce qui est peut-être plus impressionnant, c'est sa capacité à modifier des images existantes de manière réaliste. Le système peut, par exemple, supprimer des objets au premier plan ou en arrière-plan. Ainsi, à partir d'une œuvre du peintre néerlandais Vermeer, l'IA peut étendre le tableau « La Jeune Fille à la perle » pour y inclure une restitution crédible de la pièce où le modèle se trouvait lorsqu'il a été peint.

Cependant, les rédacteurs, notamment ceux de l'équipe de la rédaction d'Elektor et d'autres organes de presse renommés, ont été désarçonnés par le lancement de ChatGPT [13]. Cette IA peut interagir avec les utilisateurs de manière conversationnelle et dans une multitude de langues. Jusqu'à présent, les discussions sur les mérites des MOSFET au carbure de silicium (SiC) et des transistors au nitrure de gallium (GaN), et leurs avantages par rapport aux MOSFET en silicium, ont donné lieu à des réponses très précises. Les sujets les plus spécialisés semblent donc bien couverts.

Bien qu'il soit exceptionnellement intelligent, l'outil ne connaît cependant que les réponses aux sujets connus avant son entraînement. Comme il n'est pas en apprentissage continu, il ne sera pas à la pointe de l'information concernant les affaires courantes ou les derniers drames des groupes de K-pop (dommage). Autre problème : au bout d'un certain temps, les réponses semblent quelque peu figées et stéréotypées. Cependant, ceux qui cherchent l'inspiration ou un poème d'anniversaire comme le composerait une célébrité ne seront pas déçus.

Pour savoir si l'avenir d'Elektor passe par des êtres en chair et en os, ou plutôt par des ordinateurs, je vous propose un résumé du sujet de cet article écrit par ChatGPT. À la prochaine… ou pas !

En résumé, les systèmes embarqués et l'IA sont deux technologies de plus en plus utilisées conjointement pour créer des dispositifs et des systèmes intelligents et autonomes. Les systèmes embarqués fournissent la plateforme matérielle et logicielle nécessaire aux algorithmes d'IA, tandis que ces algorithmes permettent aux systèmes de détecter, d'analyser et de réagir à leur environnement de manière plus intelligente et plus humaine. Comme les capacités des systèmes embarqués et de l'IA continuent de s'améliorer, nous pouvons nous attendre à voir un large éventail de nouvelles applications passionnantes dans des domaines comme la robotique, les soins de santé, les transports, etc.

220673-04 — VF : Pascal Godart

Des questions, des commentaires ?

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

LIENS

[1] Edge Impulse Partners : http://bit.ly/3vbqRhG
[2] Edge Impulse CLI Installation : http://bit.ly/3BQQt71
[3] Edge Impulse Ingestion SDK : http://bit.ly/3jplhp3
[4] Projet d'exemple de reconnaissance d'un mouvement continu : http://bit.ly/3WAV9G5
[5] EON Tuner : http://bit.ly/3PQhFsr
[6] SlateSafety BAND : http://bit.ly/3YVpe5x
[7] Monolith : http://bit.ly/3BWZImh
[8] DARPA Grand Challenge, Wikipedia : https://fr.wikipedia.org/wiki/DARPA_Grand_Challenge
[9] J. Stilgoe, « Tesla crash report blames human error - this is a missed opportunity », Guardian, janvier 2017 : http://bit.ly/3Vjp7gI
[10] DYNACOR 40-36 : http://bit.ly/3GdmgBS
[11] Waymo One : http://bit.ly/3FNG8Ku
[12] DALL·E 2 : http://bit.ly/3PNPWsD
[13] ChatGPT : http://bit.ly/3PLjZRo

numériser une ferme verticale

Contribution de Würth Elektronik eiSos

La population mondiale croît, et avec elle les besoins en nourriture. L'agriculture verticale est une réponse prometteuse à ce défi. L'optimisation des facteurs de croissance des plantes permet en effet d'obtenir des produits de qualité et un rendement élevé. Cet article présente un prototype de mini-serre relié à l'IdO.

Selon l'Organisation des Nations unies pour l'alimentation et l'agriculture (FAO), il faudrait augmenter de 50 % la production agricole mondiale d'ici 2050 pour être en mesure de nourrir les 10 milliards d'habitants que comptera alors notre planète. Les terres cultivables n'ont cependant pas un potentiel infini. Leur surface a même diminué au cours des dernières décennies, passant de près de 40 % de la surface terrestre en 1991 à seulement 37 % en 2018 [1].

La surface des terres agricoles était estimée à 4,7 milliards d'hectares en 2020, soit environ un tiers des terres émergées. Un tiers de cette surface comprend des terres cultivées (environ 1,6 milliards d'ha), les deux autres tiers étant des prairies et pâturages permanents (environ 3,2 milliards d'ha) [2]. 1,6 milliards d'ha représentent 12 % des terres émergées. Ces chiffres n'ont guère évolué depuis 2000. La population mondiale est quant à elle passée de 6,15 milliards d'habitants en 2000 à 7,91 milliards en 2021, soit une augmentation de près de 30 %. Cette croissance se poursuit.

Des techniques novatrices comme les fermes verticales devraient permettre d'accroître la production agricole et ainsi répondre aux besoins en nourriture, être sources d'aliments frais et nutritifs pour des milliards de personnes, et aussi réduire l'impact environnemental de l'agriculture conventionnelle. Ces exploitations numériques ne pourront prospérer que si elles sont reconnues et soutenues par une réglementation appropriée, et si elles recourent à des systèmes d'éclairage et de surveillance à la pointe de la technologie.

Le but de cet article est de montrer le rôle clé de l'éclairage artificiel et des systèmes de contrôle à distance pour l'agriculture verticale, et comment vous pourriez, de façon rapide et économique, mettre en œuvre une ferme verticale à l'aide d'outils de prototypage adaptés.

Qu'est-ce que l'agriculture numérique ?

« L'agriculture numérique est l'intégration des technologies numériques dans les processus tels que la gestion du bétail et des cultures. Elle offre aux exploitants la possibilité d'accroître leur production, de réduire leurs coûts d'exploitation à long terme, et d'éliminer les aléas. » [3]

Cette approche peut s'appliquer aux terres agricoles actuellement disponibles, mais aussi aux zones urbaines denses n'offrant que peu de surfaces exploitables. Les cultures en intérieur présentent plusieurs avantages. Vous pouvez choisir votre propre semis, maîtriser les facteurs environnementaux, obtenir des produits de qualité, et minimiser les émissions de carbone. Quant aux fermes verticales, leur bénéfice premier est bien sûr celui de l'espace, puisque par définition elles permettent de démultiplier la production d'une surface de sol donnée.

L'agriculture verticale numérique pose toutefois quelques problèmes. Le principal est le coût de l'électricité nécessaire aux nombreuses LED qu'exigent de telles installations. Il est possible d'utiliser des LED ne produisant que la partie du spectre lumineux utile à la croissance des plantes, mais optimiser des conditions d'éclairage à grande échelle tout en cherchant à minimiser leur coût reste difficile.

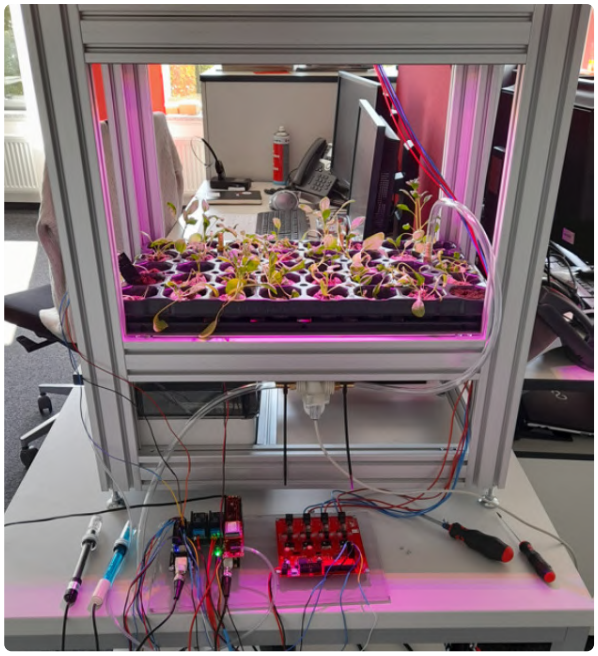


Figure 1. Prototype de culture d'intérieur. (Source : Würth Elektronik eiSos)

De plus, les conditions environnementales à maîtriser dépendent à la fois de la nature de la plante et de sa phase de croissance. Or il est très difficile de surveiller, modifier et optimiser dynamiquement et manuellement des facteurs tels que la température, l'hygrométrie et l'humidité du sol.

L'IdO comme solution d'automatisation

Cet article propose une solution à base d'IdO pour relever ces défis et automatiser une ferme verticale. Le terme *Internet des Objets* (IdO) décrit l'ensemble des technologies qui permettent de connecter plusieurs dispositifs entre eux et de les faire interagir. L'interconnexion de dispositifs produisant des données a conduit à un grand nombre d'applications et progrès dans les domaines de l'automatisation industrielle, de la santé, de la domotique, ou encore des villes, exploitations agricoles et réseaux électriques dits « intelligents ».

Considéré comme la « quatrième révolution industrielle », aussi appelé l'industrie 4.0, l'*Internet Industriel des Objets* (IIdO) représente quant à lui la numérisation des ressources et procédés reliant les machines, services et sites de production à leurs travailleurs, gestionnaires et partenaires. Une interconnexion plus dense du monde numérique et du monde physique peut améliorer productivité, sécurité, rentabilité et durabilité.

L'IIoT crée un univers de capteurs qui accélère l'apprentissage profond des opérations existantes, d'où une contextualisation et une détection rapides et automatiques des modèles et tendances. Cette saisie quantitative d'opérations qualitatives conduit notamment à une meilleure qualité, à plus d'efficacité, à une sécurité accrue et à une réduction des coûts.

La solution présentée ici permet d'automatiser et commander l'éclairage, l'irrigation et l'environnement d'une ferme verticale. Elle repose sur des outils de prototypage rapide et d'analyse des données en nuage. La **figure 1** montre un prototype de mini-serre réalisé avec ces outils.

Implantation des cartes « Feather »

Toute solution à IdO récupère des données, puis les envoie vers le cloud pour les analyser et en tirer la valeur ajoutée souhaitée.

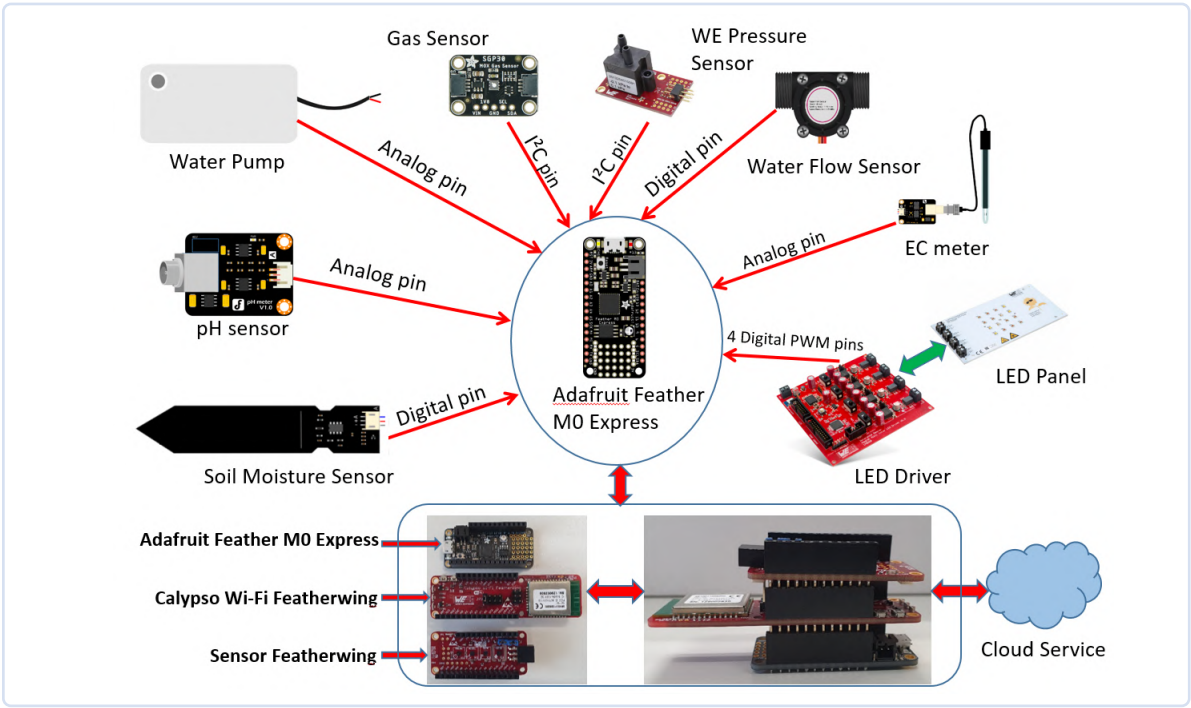


Figure 2. Les cartes au format Feather utilisées pour notre prototype à IdO. (Source : Würth Elektronik eiSos)

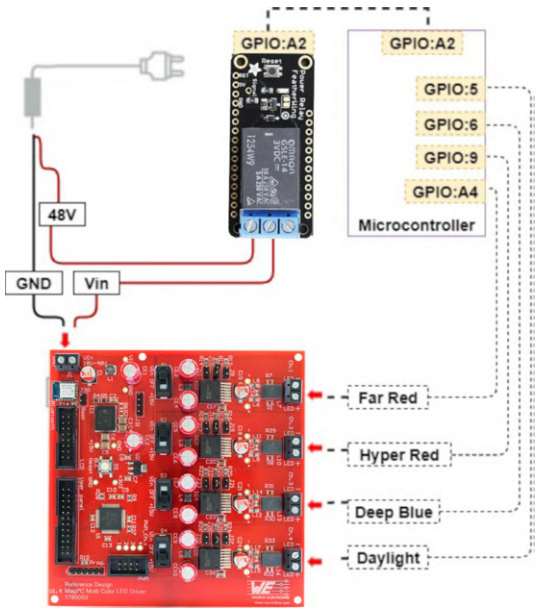


Figure 3. Le système d'éclairage horticole comprend une matrice de LED monocolores et un pilote de LED. (Source : Würth Elektronik eiSos)

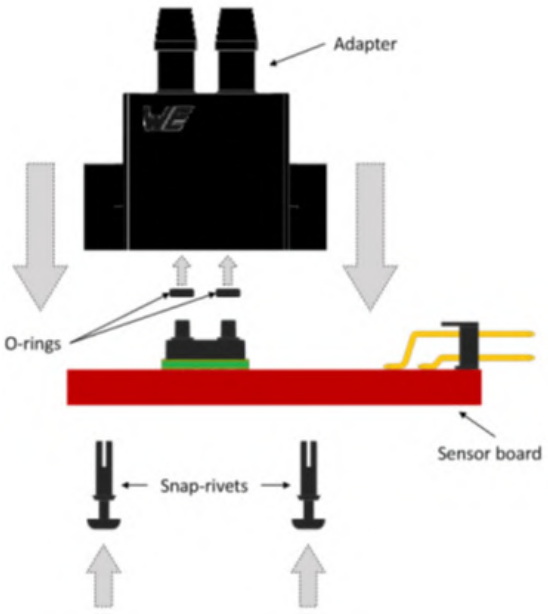


Figure 4. Le capteur de pression différentielle WSEN-PDUS est très précis, à système MEMS et piézorésistif. (Source : Würth Elektronik eiSos)

La solution présentée ici repose sur du matériel open source et des écosystèmes logiciels, plus précisément sur des cartes *FeatherWing* de Würth Elektronik, la carte *M0 Express Feather* d'Adafruit, nos LED horticoles, leurs pilotes, et divers capteurs et composants (**fig. 2**).

L'idée était de créer un système en optimisant d'un côté la croissance, de l'autre les consommations d'eau et d'électricité.

Irrigation et éclairage

Notre système est constitué d'un support pour le sol et d'un éclairage assuré par un kit comprenant un panneau de LED RVBB (le dernier B pour Blanc) et sa carte de commande à 4 canaux. Ce kit d'éclairage permet de combiner des couleurs RVBB en fonction de l'éclairage souhaité, de stimuler la croissance de plantes, ou encore de créer un éclairage dit « axé sur l'humain ». Le système d'irrigation et le réservoir d'eau utilisent de l'eau recyclée dont le pH et la conductivité électrique sont mesurés et contrôlés. Les valeurs mesurées sont envoyées dans le cloud au moyen de modules de connexion de Würth Elektronik.

Un capteur surveille l'humidité du sol, de l'eau étant apportée au besoin à l'aide d'une petite pompe. Le reste de l'eau est collecté et filtré avant de retourner à un réservoir. Au centre du système se trouvent les cartes *Feather M0 Express* et *Power Relay* d'Adafruit. Les modules *FeatherWing* servent de commutateurs. Les données sont envoyées dans le cloud puis traitées, analysées et utilisées pour le contrôle du système d'irrigation.

Système d'éclairage adaptatif

Le système d'éclairage (**fig. 3**) repose sur deux cartes : un panneau à LED horticoles monochromatiques doté de 4 canaux séparés, et le pilote de LED multicolores *MagI³C* ; ces deux cartes font partie du *Lighting Development Kit* de WE [4], un kit conçu pour favoriser la croissance des plantes. Un module *MagI³C LED Step Down High Current* permet de définir individuellement l'intensité et la couleur des 4 bandes de LED en fonction de la plante. Le panneau à LED horticoles comprend 16 LED à boîtier en céramique, dont 6 LED *Hyper Red* (660 nm), 4 *Far Red* (730 nm), 2 *Deep Blue* (450 nm), et 4 LED blanches. Le système se commande par Bluetooth, WiFi ou connexion cellulaire. La solution cloud décrite ici se veut générale.

Système d'irrigation

Le niveau d'eau du réservoir est mesuré par un module *WSEN-PDUS*, un capteur de pression différentielle très précis, à microsystème électromécanique et piézorésistif (**fig. 4**). L'eau enrichie en nutriments est envoyée dans l'égouttoir par une pompe de 12 V couplée à un débitmètre.

La qualité de l'eau est surveillée à l'aide des modules *DFR-05874* (pH) et DFR0300 (conductivité) de DFRobotGravity.

La plupart des eaux naturelles ont un pH compris entre 5 et 8. Pour les eaux d'irrigation, un pH compris entre 5,5 et 7,5 est généralement recommandé, mais nous avons obtenu nos meilleurs résultats avec un pH compris entre 5,5 et 7. Une eau dont le pH est dans cet intervalle maintient l'équilibre des nutriments, assure une désinfection chimique efficace, et empêche la formation de tartre [5].

La « fertigation », mot-valise formé à partir des mots fertilisation et irrigation, consiste à ajouter des fertilisants à l'eau d'irrigation. Elle est nécessaire ici en raison de la faible surface du substrat de culture. Un conductimètre détecte tout excès ou manque d'engrais dans l'eau – la conductivité de l'eau dépend de sa concentration en ions ; plus une eau est pure, plus faible est sa conductivité.

L'humidité du sol est mesurée par le capteur capacitif *STEMMA Soil Sensor* d'Adafruit. Il n'a qu'une seule sonde, n'expose aucune partie métallique – donc ne s'oxyde pas – et n'introduit pas de courant continu dans les plantes. La pompe est commandée depuis le nuage. Sa durée d'activation dépend du nombre de plantes, du niveau d'humidité du sol requis, et du débit de la pompe (**fig. 5**).

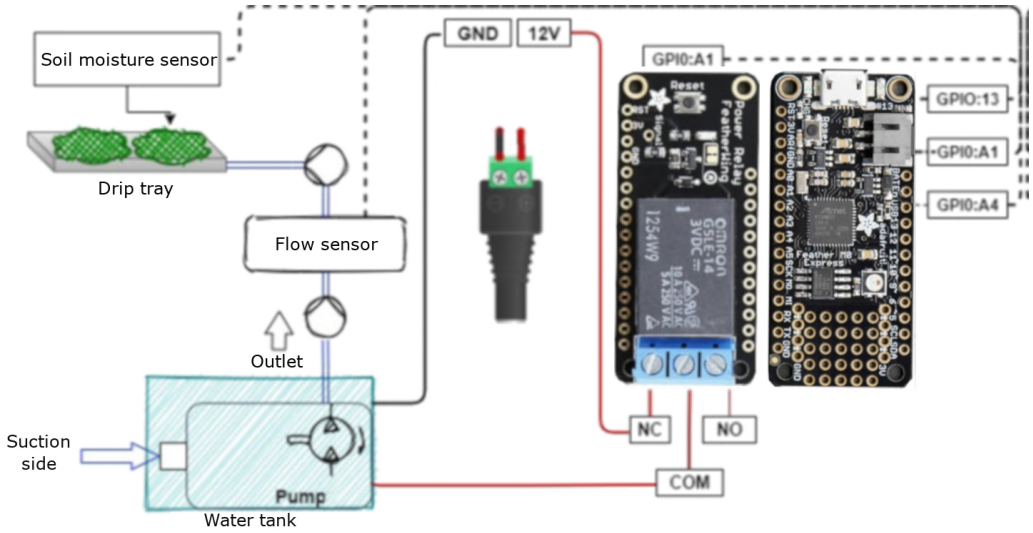


Figure 5. Le réservoir d'eau, les capteurs, la pompe et les cartes de commande du système d'irrigation. (Source: Würth Elektronik eiSos)

Facteurs environnementaux

Un capteur mesure une condition environnementale sous forme de valeur analogique ou numérique, tandis qu'un actionneur la modifie au besoin. Les progrès réalisés dans le domaine de l'électronique, en particulier dans celui des semi-conducteurs, ont permis de créer un vaste éventail de tels capteurs et actionneurs, à la fois efficaces et compacts.

Bon nombre de procédés d'enrichissement et d'extraction reposent sur le dioxyde de carbone. Le CO_2 accélère la croissance des plantes, mais peut aussi servir de fumigant. On utilise généralement un niveau de concentration de CO_2 de 800 à 1500 ppm durant la phase d'enrichissement pour accroître la photosynthèse de 20 à 30 %.

L'augmentation du métabolisme global aide les plantes à résister aux effets de la chaleur. Des plantes plus grandes, plus saines et plus robustes supportent mieux les conditions environnementales extrêmes. Une plante dont le métabolisme est accru nécessite cependant plus d'eau et de nutriments, et aussi une ventilation supplémentaire.

Les niveaux de CO_2 sont mesurés et surveillés à distance avec le capteur de qualité de l'air *SPG30* d'Adafruit, module lui-même relié au capteur *FeatherWing* de Würth Elektronik. Notez que le capteur mesure « l'équivalent CO_2 » et que cette valeur est calculée d'après la concentration en H_2 – ce n'est donc pas un « vrai » capteur de CO_2 de laboratoire. L'étape suivante sera l'enrichissement.

Les plantes passent par plusieurs stades de croissance, chacun nécessitant des conditions thermiques et d'éclairage différentes. La majorité des plantes tolèrent des fluctuations de température non-extrêmes. Les conditions thermiques optimales de croissance peuvent varier selon le stade de croissance, mais aussi au cours de la journée. L'optimisation peut se faire par adaptation du cycle diurne, et en baissant la température de quelques degrés lorsque l'éclairage est coupé.

L'humidité atmosphérique est exprimée en pourcentage d'humidité de l'air. Là encore le niveau d'humidité optimal dépend du stade de croissance de la plante. Un environnement trop humide peut favoriser la propagation de maladies. Humidité et température sont surveillées par le capteur *FeatherWing* de WE.

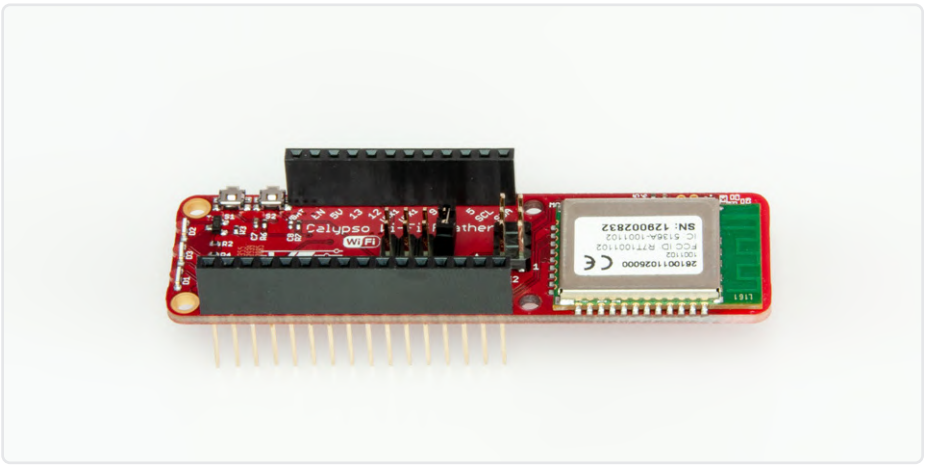


Figure 6: La carte Calypso FeatherWing assure la connexion des environnements disposant d'un réseau wifi. (Source : Würth Elektronik eiSos)

Connexion à l'IdO

Capteurs et actionneurs logent en général dans des dispositifs n'ayant qu'un accès limité au monde numérique, d'où l'intérêt des modules à connexion sans fil. Nous disposons à cet effet d'un grand nombre de normes et protocoles, ainsi que d'une grande variété de solutions propriétaires. Le choix d'une solution sans fil dépend de nombreux facteurs : portée, débit, réglementation (notamment pour le spectre utilisé), budget…

Notre système repose sur deux cartes : la *Calypso Wi-Fi FeatherWing* (**fig. 6**) pour les environnements disposant d'un réseau wifi,

la *Adrastea-I FeatherWing* (**fig. 7**) pour les lieux dépourvus d'un tel réseau. Ces deux cartes sont reliées au reste du système via la carte *M0 Express Feather* d'Adafruit.

Calypso est un module Wi-Fi compact, basé sur IEEE 802.11 b/g/n (2,4 GHz), à pile TCP/IP intégrée et à protocole MQTT natif. *Adrastea-I* est un module cellulaire compact à protocoles LTE-M/NB-IoT, à GNSS intégré et soutenu par un processeur ARM Cortex-M4 capable de faire tourner toute application de l'IdO.

Ces deux modules peuvent être connectés facilement et de façon sécurisée à n'importe quel *cloud*. Nous avons opté pour un processeur M0 externe et la plateforme *IoT Central* de Microsoft, une PaaS (plateforme en tant que service) simple et facile à utiliser. Elle fournit une interface utilisateur et une API pour la connexion et l'exploitation des dispositifs de l'IdO. Sa télémétrie et ses fonctions nous ont servi à surveiller et contrôler tous les aspects de notre min-serre.

Recours à l'IA

Il est possible de recourir à l'IA pour détecter des motifs et créer des profils de plantes dans le nuage, puis de les utiliser pour définir automatiquement des paramètres tels que la valeur des LED, la température et l'humidité afin d'optimiser les conditions de croissance.

L'agriculture verticale ne nourrira pas seule la planète, mais fournira des produits frais à plus de personnes. L'approche décrite ici nous a permis de construire une ferme verticale miniature adaptée à la cuisine d'un logement. Fut une époque où ce genre de système était appelé une jardinière.

230077-04 — VF : Hervé Moreau

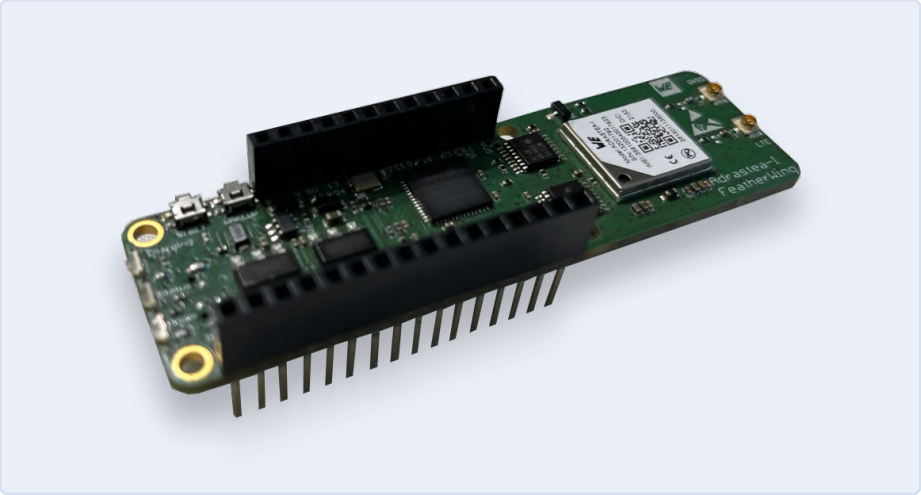


Figure 7: La carte Adrastea-I est un module cellulaire LTE-M/NB-IoT compact doté d'un processeur ARM Cortex-M4 et d'un système GNSS. (Source : Würth Elektronik eiSos)

À propos des auteurs

Miroslav Adamov a étudié la physique et l'informatique à l'université serbe de Belgrade. Il a ensuite été chercheur à l'université technique et à l'institut WIAS de Berlin, à l'université Friedrich-Alexander d'Erlangen-Nuremberg, et au Center of Private Equity Research de Munich. Après quelques années passées dans la finance quantitative, il rejoint Würth Elektronik en 2015 en tant qu'analyste d'affaires. Depuis 2007 il est architecte principal des solutions IdO et IIdO.

Adithya Madanahalli a obtenu un master en ingénierie de l'information à l'université technique de Munich. Il a travaillé plusieurs années comme ingénieur logiciel dans le domaine des communications et capteurs sans fil. Adithya est depuis 2022 ingénieur en IdO à l'unité Wireless Connectivity and Sensors de Würth Elektronik eiSos. Il conçoit des solutions IdO dont il a en charge le matériel, les logiciels embarqués et la sécurité de bout en bout.

LIENS

[1] Le futur de l'alimentation et de l'agriculture, tendances et défis, rapport de la FAO, 2017 [PDF en anglais] : https://fao.org/3/a-i6583e.pdf

[2] Statistiques de l'utilisation des sols, indicateurs et tendances 2000-2020, rapport de la FAO [PDF en anglais] : https://fao.org/3/cc0963en/cc0963en.pdf

[3] E. Ambrose, "Digital agriculture: Why the future is now." : https://ag.purdue.edu/stories/digital-agriculture-why-the-future-is-now/

[4] Kit de développement d'éclairage de Würth Elektronik : https://we-online.com/en/components/products/LIGHTING_DEVELOPMENT_KIT

[5] V. Brunton, "Irrigation water quality" [PDF]: https://dpi.nsw.gov.au/__data/assets/pdf_file/0005/433643/Irrigation-water-quality.pdf

Elektor infographies

l'embarqué et l'IA d'aujourd'hui et demain

Malgré les innombrables défis présentés par la pandémie de COVID-19 et les récents événements géopolitiques, il existe un large éventail d'opportunités commerciales passionnantes pour les cadres, les entrepreneurs, les professionnels et les étudiants qui se concentrent à la fois sur les technologies embarquées et les solutions liées à l'intelligence artificielle. Examinons quelques faits et chiffres.

Systèmes embarqués : l'avenir

Vous vous interrogez sur l'avenir du secteur des systèmes embarqués ? La croissance semble être au rendez-vous. Selon un récent rapport d'Allied Market Research, le secteur est en passe de dépasser les 163 milliards de dollars d'ici 2031, avec la région Asie-Pacifique en tête. [1]

Entreprises à suivre

- Advantech Corp.
- Analog Devices
- Infineon Technologies
- Intel Corp.
- Microchip Technology
- NXP Semiconductors
- Qualcomm
- Renesas Electronics
- STMicroelectronics
- Texas Instruments

(Source : Allied Market Research)

L'intelligence artificielle dans le viseur

20 %

Environ un doctorant en informatique sur cinq diplômés en 2020 s'est spécialisé dans l'IA/ML.

93,5 milliards de dollars

Les investissements privés dans l'IA ont atteint environ 93,5 milliards de dollars en 2021, soit plus du double du montant de 2020.

Si le cabinet d'études de marché IDC a raison, le marché mondial de l'intelligence artificielle (IA), y compris les logiciels, le matériel et les services, affichera un taux de croissance annuel composé de 18,6 % durant la période 2022-2026 pour atteindre 900 milliards de dollars en 2026. [2] Les opportunités pour les entreprises, les professionnels et même les makers sont nombreuses. Regardez ces données clés tirées d'un rapport de l'université de Stanford pour 2022. [3] La croissance du secteur de l'IA est assez claire.

Amérique du Nord

De 2010 à 2021, 56,96 % des brevets d'IA accordés ont été déposés en Amérique du Nord. 31,09 % provenaient d'Asie et du Pacifique. 11,27 % provenaient d'Europe et d'Asie centrale.

2x

Le nombre de publications en anglais sur l'IA (revues, livres, thèses, conférences) a doublé (dans le monde), passant de 162 444 en 2010 à 334 497 en 2021.

30x

Le nombre de brevets d'IA déposés en 2021 était plus de 30 fois supérieur à celui de 2015. Cela représente un taux de croissance annuel de 76,9 %.

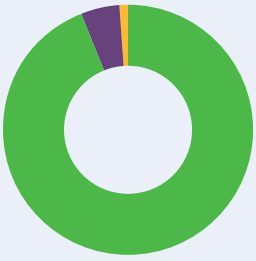
Où sont les emplois dans l'IA ?

(Source : www.mckinsey.com)

Les ingénieurs logiciels ont un bel avenir devant eux ! Selon le récent rapport de McKinsey sur l'état de l'IA, ils sont en tête des demandeurs d'emploi spécialisés dans l'IA qui ont été embauchés au cours des derniers mois. Les ingénieurs et les scientifiques en données d'IA arrivent en deuxième et troisième position. [4] Il s'agit d'un autre signe clair que de nombreuses organisations sont passées de l'expérimentation de l'IA à son intégration active dans les applications d'entreprise.

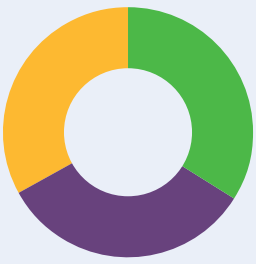
Que pensent les chefs d'entreprise de l'IA ?

Nous savons tous maintenant que l'IA est synonyme de grosses affaires. Mais comment les chefs d'entreprise l'envisagent ? Si l'intérêt varie selon les secteurs d'activité, les solutions d'IA sont assurément un sujet de leurs discussions dans les salles de conférence, lors des réunions du conseil d'administration et des sorties d'entreprise. Les données suivantes de Deloitte offrent un aperçu de la situation. [5]



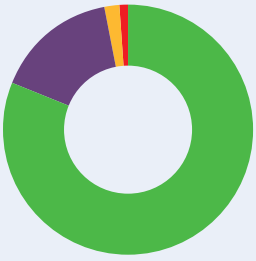
Importance de l'IA pour le succès d'une entreprise :

- 94 % Très important ou important
- 5 % Assez important
- 1 % Pas important



Quels sont les trois principaux défis associés au lancement et à la réalisation de projets ?

- 30 % Manque de financement pour les technologies d'IA
- 29 % Sélection des technologies d'IA appropriées
- 29 % Compétences techniques insuffisantes



Travailler avec de l'IA améliorera-t-il les performances et la satisfaction au travail ?

- 82 % D'accord
- 16 % Ni d'accord ni en désaccord
- 2 % Pas d'accord
- 1 % Pas sûr

LIENS

[1] CISION, "Embedded Systems Market to Reach $163.2 Billion by 2031, Globally, by 2031 at 6.5% CAGR", Allied Market Research, 2022: https://bit.ly/embedded-outlook-AMR

[2] IDC, "IDC Forecasts 18.6% Compound Annual Growth for the Artificial Intelligence Market in 2022-2026", 2022: https://bit.ly/IDC-AI-2022-2026

[3] D. Zhang, et al, "The AI Index 2022 Annual Report", HAI, Stanford University, March 2022: https://bit.ly/AI-index-rep-22

[4] McKinsey, "The State of AI in 2022 — And a Half Decade in Review », QuantumBlack, Dec 6, 2022: https://bit.ly/mckinsey-AI-22

[5] Deloitte, "Fueling the AI transformation", October 2022: https://bit.ly/deloitte-ai-state

Présentation du TinyML

Mark Patrick (Mouser Electronics)

Dans cet article, nous allons nous pencher sur l'apprentissage automatique (ML en anglais pour « machine learning ») propulsé par des microcontrôleurs de faible puissance et à faible consommation d'énergie. Il s'agit d'une nouvelle approche de l'apprentissage automatique baptisée TinyML. L'apprentissage automatique a pénétré de nombreux aspects de notre vie quotidienne, tant chez soi qu'au travail ou entre les deux. De nombreuses applications d'apprentissage automatique nécessitent une puissance de calcul importante pour traiter des données scientifiques ou financières complexes. Or, les applications ML conçues pour l'Internet des objets (IoT) et d'autres applications en périphérie (edge) n'ont qu'une faible capacité de calcul et une faible connectivité.

L'IA et le ML font déjà partie de nos vies

Il n'y a pas si longtemps encore que parler à sa montre-bracelet ou à un gadget dans le style du Communicator de Star Trek relevait de l'esprit fécond des auteurs de science-fiction. C'est devenu aujourd'hui un geste presque banal. Nous parlons à nos applications pour smartphones, au système d'infodivertissement de notre voiture et à l'enceinte intelligente dans notre salon.

L'intelligence artificielle (IA) – la science qui vise à créer des ordinateurs capables de penser, de détecter, de reconnaître et de résoudre des problèmes – est devenue la pierre angulaire de l'informatique et de la science des données d'aujourd'hui. L'apprentissage automatique est un domaine d'application de l'IA et concerne l'utilisation par des ordinateurs d'algorithmes leur permettant d'apprendre et d'améliorer des méthodes pour réaliser une tâche sans qu'ils aient été explicitement programmés pour l'accomplir.

Le monde qui nous entoure porte déjà l'empreinte du ML : prévisions météorologiques, recherche d'itinéraires, encarts publicitaires dans vos applications de médias sociaux… De nombreux domaines de la recherche scientifique dépendent désormais du ML pour parcourir des pétaoctets de données afin d'en dégager des tendances.

Fonctionnement de l'apprentissage automatique

Des exemples d'apprentissage automatique que nous venons de citer, nous ne percevons en général que le résultat. Peu d'entre nous ont eu l'occasion d'admirer de près la complexité d'opérations telles que la recherche d'un nouveau trou noir ou le calcul du nombre de permutations d'événements météorologiques passés en vue pour déterminer les prévisions météorologiques du jour. De telles tâches complexes s'appuient sur de grands ensembles de données, plusieurs algorithmes candidats et une puissance de calcul importante. En creusant un peu le sujet, on découvre qu'il existe différentes catégories d'apprentissage automatique, chacune convenant plus particulièrement à des tâches spécifiques. Sans entrer dans les détails, il s'agit des méthodes d'apprentissage supervisé, non supervisé, semi-supervisé et par renforcement.

Le réseau de neurones est un élément essentiel de toute application de ML. Pour résumer grossièrement, un réseau neuronal dans un modèle mathématique imite la fonction des neurones dans le cerveau humain. Le modèle utilise des algorithmes pour déduire les probabilités d'un résultat. Par exemple, « la probabilité que cette image représente un chien est de 95% » en résultat d'une opération de reconnaissance d'image. Il existe différents types de réseaux de neurones. Des termes comme réseaux de neurones convolutifs (CNN) ou réseaux de neurones récurrents (RNN) ne vous sont peut-être pas inconnus. Chaque type de réseau neuronal possède un ensemble différent de couches interconnectées, ce qui le rend plus adapté à des tâches spécifiques.

L'apprentissage supervisé consiste à alimenter le réseau neuronal en données d'entraînement en vue de permettre à un algorithme d'en déduire des résultats. Prenons pour exemple la reconnaissance d'images, une tâche qui convient particulièrement aux CNN. Pour identifier différents types de fruits, il faut fournir à l'algorithme du réseau neuronal des milliers d'images étiquetées représentant différents types de fruits, sous différents angles et à différents degrés de maturité. L'algorithme recherche alors des

caractéristiques discernables qui l'aident à identifier les différents types de fruits. Cette phase d'entraînement est itérative et il est alors possible qu'il faille encore affiner l'algorithme afin d'obtenir les probabilités les plus élevées lorsqu'il est mis en présence d'un ensemble de données de test (ici, des images).
Une fois que l'algorithme du réseau neuronal a atteint son meilleur niveau de performance avec l'ensemble de données de test, le modèle est prêt à être déployé. Une fois entré en phase de déploiement, aussi appelée inférence, le modèle déduit les résultats à partir d'une probabilité.
Pour entrer en interaction avec, par exemple, notre enceinte intelligente, nous utilisons généralement un mot ou une phrase de déclenchement – « OK, Google » – pour que l'appareil sorte de veille et se mette à nous écouter. Une enceinte intelligente ne dispose évidemment pas des capacités de calcul d'un centre de données. Elle se contente d'enregistrer de courts fichiers audio et de les envoyer dans le cloud, où le processus d'inférence déterminera la nature de notre demande. Ce que l'enceinte fait d'elle-même, c'est détecter le mot ou la phrase de déclenchement. C'est un excellent exemple d'apprentissage automatique simple et c'est précisément à cela que sert le TinyML !

L'apprentissage automatique et l'IoT industriel

La liste des applications susceptibles de tirer avantage de l'apprentissage automatique s'allonge de façon exponentielle à mesure que cette technologie prolifère dans notre société. Cependant, de nombreuses applications industrielles n'ont rien à voir avec le « big data », mais s'occupent de la manière de rendre les lignes de production plus efficaces. Une panne inattendue sur une ligne de production peut s'avérer très coûteuse. Imaginez par exemple qu'un moteur tombe en panne durant la transformation d'aliments réfrigérés. La production serait mise à l'arrêt et les matières premières seraient perdues. C'est pourquoi de nombreuses entreprises ont adopté un régime de maintenance prédictive afin de planifier des temps d'arrêt prévisibles et ainsi éviter que de telles pannes surviennent. La surveillance basée sur l'état des moyens de production tels que les moteurs et les actionneurs aide à prévoir quand, par exemple, un roulement de moteur commence à montrer des signes d'usure excessive. Les algorithmes d'apprentissage automatique utilisés dans les capteurs périphériques IIoT peuvent identifier quand la vibration d'un moteur s'écarte de sa fréquence habituelle, ce qui peut être un signe annonciateur suffisant.
Rien que dans le domaine de l'industrie, les possibilités d'applications de l'apprentissage automatique sont innombrables, mais plusieurs défis techniques se dressent sur leur chemin. Contrairement aux applications big data, de simples applications comme les capteurs périphériques IIoT ne disposent que d'une infime partie des ressources de calcul et de mémoire disponibles dans un centre de données. Ensuite, une seule usine peut devoir être équipée de centaines de capteurs. L'économie d'échelle est donc un facteur à prendre en compte au même titre que les dimensions physiques de l'appareil, la disponibilité d'une alimentation électrique appropriée et le type de connectivité (filaire ou sans fil). Dans une majorité des cas aussi, l'algorithme de réseau neuronal associé à un capteur de vibrations, par exemple, requiert moins de ressources qu'un algorithme utilisé dans le cadre de la recherche dans l'espace lointain. Cela autorise donc les développeurs de systèmes embarqués à chercher le moyen d'exécuter des modèles de réseau sur des microcontrôleurs à faible consommation alimentés par batterie.

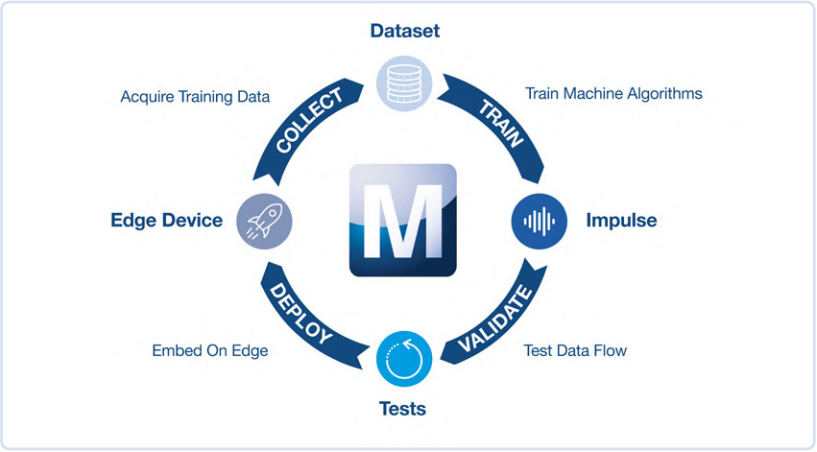


Figure 1: Étapes d'entraînement et d'inférence.

TinyML : l'apprentissage automatique en périphérie

La plateforme informatique utilisée pour entraîner un modèle de réseau neuronal ne doit pas nécessairement être la même que celle utilisée en phase de déploiement. On peut ainsi avoir recours à une plateforme moins gourmande en ressources, mais ce n'est là pas la moindre des difficultés techniques à prendre en compte. L'algorithme est-il capable de s'exécuter au bon moment ? Comment l'appareil communiquera-t-il avec le système hôte pour avertir le personnel opérationnel ? Enfin, les développeurs de systèmes embarqués ne sont généralement pas des spécialistes des systèmes de données et il leur faut, à eux aussi, un certain temps d'apprentissage pour maîtriser les concepts de l'apprentissage automatique et travailler avec des réseaux neuronaux.

230062-04

À propos de l'auteur

En tant que directeur marketing technique de Mouser Electronics pour la région EMEA, 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.

KwickPOS

KwickPOS est un système de point de vente qui aide les restaurants et commerces à traiter leurs opérations quotidiennes.

La société KwickPOS a eu son premier client en 2003 avant même de démarrer son activité : un restaurant chinois cherchant un moyen efficace de traiter les commandes de ses clients. Les propriétaires du restaurant avaient pour cela contacté leurs amis Tom Jin et Ming Ye, chacun cumulant 20 ans d'expérience dans la Silicon Valley comme développeurs de logiciels pour restaurants. Tom et Ming ont rapidement créé un système – basé sur Linux – satisfaisant la demande de leurs amis.

D'abord succès auprès des restaurants chinois (selon Ming Ye « les plus complexes pour ce qui est du menu et de l'organisation interne »), l'efficacité de ce système a vite attiré l'attention d'un cercle plus large de restaurateurs. La demande était suffisante pour songer à fonder une entreprise commerciale.

Le problème

Il fallait à Tom et Ming une grande confiance dans le potentiel de leur système pour quitter la Silicon Valley. C'est ce qu'ils firent en 2015 après en avoir conçu un grand nombre. Un système de points de vente s'appelle en anglais un POS system (*Point Of Sales system*), d'où le nom choisi pour leur entreprise basée à Houston (Texas) : KwickPOS. Tom et Ming ajoutèrent à leur système des fonctions de soutien administratif, d'achat, d'inventaire, de gestion du personnel, de paiement, etc. La plupart des systèmes existants reposaient sur Windows. Ils étaient encombrants, mais aussi coûteux, donc susceptibles d'attirer la convoitise de voleurs. « Certains systèmes pour restaurants comprennent trois ou quatre terminaux, dont l'un centralise les opérations », explique Ye. Un commerçant qui se fait voler cet ordinateur central, une mésaventure loin d'être exceptionnelle, peut ainsi se retrouver dans l'incapacité de satisfaire des commandes.

Aujourd'hui, l'entreprise compte plus de 2000 restaurants parmi ses clients.

Commerçants et restaurateurs sont confrontés à un autre problème, en particulier dans certaines régions des États-Unis où les coupures de courant ne sont pas rares : que faire en cas de panne d'internet ? L'essentiel des commandes se faisant par courriel ou en ligne, la « continuité internet » s'avère en effet vitale, de même que la capacité à pouvoir traiter un paiement à tout moment.

La solution

Lorsqu'est sorti le Raspberry Pi, Tom s'est tout de suite dit que ce nano-ordinateur pourrait faire tourner le logiciel de KwickPOS et servir de serveur local, et aussi que ses petites dimensions en feraient le cœur matériel idéal pour des terminaux de commande destinés à des restaurants dont l'espace souvent exigu doit être optimisé. Passer au RPi relevait du bon sens commercial, d'abord parce que KwickPOS reposait déjà sur Linux depuis 2013, ensuite parce qu'un module Compute ne coûtait qu'un dixième de son équivalent Windows.

Contrairement à la plupart des systèmes de points de vente, KwickPOS, repose sur un navigateur. Ce qui est affiché sur un terminal (une commande p. ex.) est répliqué sur un serveur distant, mais KwickPOS comprend aussi un serveur local – la petitesse du module Compute le permet. Le RPi de chaque terminal peut en outre traiter les données de paiement. Le point fort de cette configuration est le mode hors ligne. « Si l'internet tombe en panne, ce n'est pas grave, explique Ye. Le gérant peut mettre KwikPOS en mode hors ligne et continuer à gérer son restaurant. » Lorsque l'internet revient, toutes les transactions traitées hors ligne sont répliquées et traitées sur le serveur distant. « Le temps de disponibilité est un argument concurrentiel très fort. »

Pourquoi le Raspberry Pi ?

Tom apprécie la stabilité du module Compute. Savoir que Windows s'impose encore « aussi naturellement que la loi » parmi les entreprises le frustre particulièrement, lui qui sait que les serveurs Windows sont régulièrement le maillon faible des systèmes de points de vente. Les services KwickPOS utilisent les modules Compute 3 et 4.

Avoir placé le RPi au cœur de KwickPOS était un choix judicieux. « Nos clients en sont très satisfaits », nous a affirmé Ming. Il faut dire que l'espace réservé au service est inévitablement exigu dans un restaurant, et que KwickPOS ne prend guère de place. Il est même suffisamment discret pour ne pas attirer l'œil de voleurs opportunistes. Et à supposer qu'une unité soit volée, le restaurateur pourra continuer à travailler puisque le serveur RPi local prendra le relais. KwickPOS résout donc le problème des coupures internet et des pertes de revenus associées. Le système, quoi qu'il arrive, fonctionnera toujours comme s'il était connecté, garantissant qu'aucune donnée ne sera perdue.

KwickPOS a élargi sa clientèle aux commerces de détail, un secteur où là aussi les commandes doivent être prises et traitées rapidement et efficacement. KwickPOS est en outre indépendant de toute plateforme, en ce sens qu'il peut communiquer avec n'importe quel terminal de paiement mobile, et donc s'intégrer au flux opérationnel existant de tout magasin.

Le résultat

Depuis qu'elle est passée au Raspberry Pi en 2018, la société KwickPOS compte des clients dans 45 États américains, ainsi qu'au Canada et au Mexique. Un restaurant chinois de Londres, au Royaume-Uni, utilise également KwickPOS. Aujourd'hui, l'entreprise compte plus de 2000 restaurants parmi ses clients, aussi bien des entités individuelles que des chaînes de restauration.

230085-04 — VF : Hervé Moreau

LIENS

Les recettes du succès : magpi.cc/success

Hautes performances pour tous

Normes Computer-on-Module

Figure 1. Les différentes tailles de modules des normes relatives aux Computer-on-Modules indépendants des fabricants.

▲ **Contribué par Congatec**

Il se passe beaucoup de choses sur le marché des Computer-on-Module. COM Express 3.1, dispose d'une nouvelle version de la norme Computer-on-Module la plus réussie. De même COM-HPC, la norme informatique embarqué haute performance, soulève également de nombreuses questions. Alors, que doivent savoir les OEM et les concepteurs de systèmes ?

Selon les chiffres publiés par IHS Markit, les Computer-on-Modules sont le principe de conception embarqué le plus utilisé, devant même les cartes embarquées classiques telles que les cartes SBC de 3,5 pouces ou Mini-ITX. La grande popularité de la conception de systèmes embarqués provient du mariage réussi de la conception d'une carte porteuse flexible et spécifique au client avec des modules prêts à l'emploi et faciles à intégrer qui incluent tous les pilotes et micrologiciels nécessaires. Ces super composants intègrent tous les briques de base principals comme l'unité centrale, la mémoire vive, les interfaces à haut débit et souvent aussi l'unité graphique, le tout dans un seul boîtier dont les fonctions sont validées. Un autre avantage est le fait que les Computer-on-Modules de la même norme sont librement interchangeables, tant entre les générations de processeurs qu'entre les fabricants. Les OEM disposent ainsi d'une flexibilité totale pour adapter et mettre à niveau leurs solutions avec la dernière technologie de processeur, même après plusieurs années. Cela facilite également la mise en œuvre de stratégies multi-fournisseurs, ce qui offre des avantages en termes de prix, et surtout, garantit la disponibilité. Deux comités indépendants s'occupent de la normalisation des modules : le PICMG, qui est hébergé en Amérique, et le SGET allemand ; ensemble, ils s'occupent actuellement de quatre normes Computer-on-Module, la plupart d'entre elles fournissant de nombreux variants. Ces normes sont COM-HPC et COM

Express pour le segment haut de gamme, et SMARC et Qseven pour le segment basse consommation.

COM-HPC, le standard haute performance

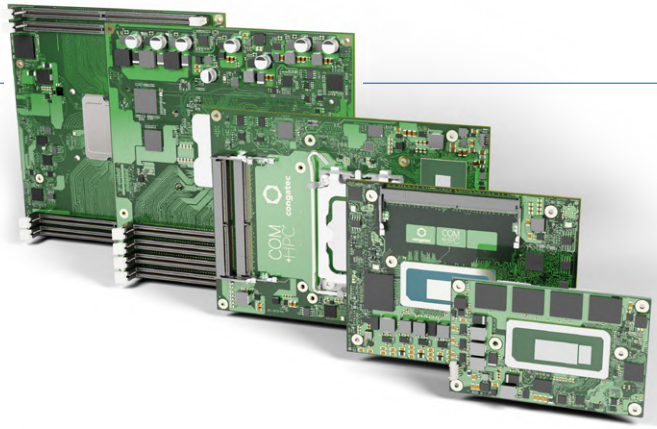
COM-HPC, la plus récente norme Computer-on-Module du PICMG, est destinée aux conceptions d'ordinateurs embarqués haute performance, qui ne pouvaient bénéficier des normes précédentes. Ses performances sont supérieures à celles de la norme COM Express, leader mondial. COM-HPC vise les nouvelles interfaces à haut débit telles que PCI Express 4.0 et 5.0, ainsi que l'Ethernet 25 Gbits. Il propose à cet effet deux versions différentes de modules, qui ont été développées par le sous-comité COM-HPC sous la direction de son président Christian Eder de congatec : COM-HPC Server et COM-HPC Client. Leurs principales différences techniques résident dans l'encombrement, le nombre et le type d'interfaces prises en charge et la capacité de mémoire. La spécification COM-HPC Mini est une nouveauté. Elle offre les mêmes avantages dans un format carte de crédit.

Modules COM-HPC Server

COM-HPC Server définit l'ultra-haut de gamme de l'informatique embarquée, en s'adressant aux nouveaux serveurs de edge et fog en environnements difficiles qui doivent gérer des charges de travail de plus en plus massives. À cette fin, COM-HPC Server spécifie deux tailles d'encombrement avec jusqu'à 64 voies PCIe et jusqu'à 256 Gigaoctets/s, ainsi que jusqu'à 8x Ethernet avec 25 Gbits/s chacun. COM-HPC Server ne se limite pas à la technologie x86 mais permet également d'utiliser des processeurs RISC, des FPGA et des GPGU - ce qui ajoute de nouvelles perspectives de modularisation. Afin de répondre aux exigences des applications serveur, les modules offrent également des modes maître-esclave et une gestion à distance. En s'appuyant sur le jeu d'instructions de la puissante norme IPMI, la technologie serveur est également disponible pour les Server-on-Modules. Avec les modules COM-HPC Server offrant un budget énergétique allant jusqu'à 300 watts, cette norme est adaptée au développement de serveurs embarqués ultra-hautes performances edge et fog. En comparaison, les Server-on-Modules COM Express Type 7 les plus puissants aujourd'hui ne prennent en charge qu'un maximum de 100 watts. Une autre différence importante réside dans le nombre de broches de signal : le connecteur COM Express compte 440 broches, tandis que le COM-HPC en offre presque le double avec 800.

Modules COM-HPC Client

Les modules COM-HPC Client sont conçus pour les systèmes embarqués haute performance avec graphiques intégrés. Ils offrent quatre sorties graphiques via trois interfaces DDI (*Digital Display*) et une interface DisplayPort embarquée (eDP).



Ils hébergent jusqu'à quatre sockets SO-DIMM avec jusqu'à 128 Go de RAM. Pour la connexion des périphériques, 48 voies PCIe et 2 x USB 4.0 sont disponibles ; les modules caméras embarquées peuvent également être connectés directement via deux interfaces MIPI-CSI. Les modules COM-HPC Client seront disponibles en trois tailles différentes : 120 mm x 160 mm (taille C), 120 mm x 120 mm (taille B) et 120 mm x 95 mm (taille A).

▲

Figure 2. Congatec propose une gamme complète de COM-HPC allant du COM-HPC Server et Client au COM-HPC Mini.

Modules COM-HPC Mini

COM-HPC Mini est une nouvelle norme pour les Computer-on-Modules en cours de développement au PICMG. Actuellement, le sous-comité technique a approuvé le brochage et l'encombrement de la nouvelle spécification Computer-on-Module haute performance de la taille d'une carte de crédit (95 x 60 mm). Avec ses 400 broches, la nouvelle norme COM-HPC Mini est conçue pour répondre aux besoins croissants d'interface des ordinateurs edge hétérogènes et multifonctionnels. Les extensions comprennent jusqu'à 4 x USB 4.0 avec une fonctionnalité complète, y compris Thunderbolt et le mode alternatif DisplayPort, PCIe Gen 4/5 avec jusqu'à 16 voies, 2 ports Ethernet 10 Gbits/s et bien plus encore. Si l'on ajoute à cela le fait que le connecteur COM-HPC Mini est qualifié pour des débits de plus de 32 Gbits/s - ce qui est suffisant pour prendre en charge PCIe Gen 5 ou même Gen 6 - il est clair que ses capacités vont bien au-delà de celles de tous les autres standards de modules au format carte de crédit.

COM Express, norme module la plus populaire au monde

Cela signifie que le plus petit emplacement COM-HPC Client est presque de la même taille que COM Express

Figure 3. La toute dernière technologie processeur comme les processeurs haut de gamme Intel Core 13e Gen en BGA se retrouve sur les modules COM-HPC et COM Express.

◄

Basic, soit 125 mm x 95 mm. Cela montre que COM-HPC Client se trouve bien au-dessus de COM Express et cible les applications qui ne peuvent être traitées avec COM Express. COM Express a été lancé en 2005 et, parmi les normes du Computer-on-Module présentées ici, c'est celle qui existe depuis le plus longtemps. La spécification définit une famille de modules de tailles et de types de broches différents. Contrairement aux spécifications de modules COM-HPC et SFF (*small form factor*) Qseven et SMARC, COM Express se concentre uniquement sur les processeurs x86. Avec la ratification de la spécification 3.1, COM Express prend désormais en charge les dernières interfaces haut débit telles que PCIe 4.0 et USB 4. Malgré ces améliorations, les modules COM Express 3.1 Type 6 sont entièrement rétrocompatibles avec les modules 3.0 et les cartes porteuses, ce qui garantit que même les conceptions plus anciennes peuvent être équipées des nouveaux processeurs.

Server-on-Modules COM Express Type 7

Comme COM-HPC, COM Express propose également des modules serveur et client, qui sont principalement disponibles dans les brochages connus Type 6 (client) et Type 7 (serveur). Comme pour COM-HPC, le brochage Type 7 server est un Server-on-Module sans tête et sans sorties graphiques. Il est également conçu pour les serveurs embarqués edge et fog. Il prend notamment en charge jusqu'à 4 x 10 GbE et jusqu'à 32 x voies PCIe Gen 3.0 à haut débit pour les interfaces et les supports de stockage. Ces Server-on-Modules sont disponibles avec des processeurs Intel® Xeon® D ou des processeurs AMD EPYC Embedded 3000. Pour eux, congatec propose également un écosystème de 100 watts avec des solutions de refroidissement prêtes à l'emploi pour simplifier la conception des Server-on-Modules COM Express les plus puissants.

Computer-on-Modules COM Express Type 6

Pour les applications embarquées classiques avec graphiques, les modules PICMG COM Express Type 6 sont le choix idéal. Ils sont équipés de processeurs embarqués allant des Intel® Core™, Pentium® et Celeron® aux AMD Embedded R-Series. Disponibles dans des surfaces de 95 mm x 125 mm (*Basic*) ou 95 mm x 95 mm (Compact), ils fournissent 440 broches à la carte porteuse pour une large gamme d'interfaces informatiques modernes. Prenant en charge jusqu'à quatre écrans indépendants, 24 voies PCIe, l'USB 2.0 et l'USB 3.0, ainsi que les interfaces Ethernet, bus CAN et série, ils offrent tout ce qui est nécessaire pour construire des API, IHM, systèmes d'atelier ou stations de travail SCADA puissants dans les salles de contrôle. Parmi les autres domaines d'application figurent les systèmes de signalisation numérique haut de gamme et les équipements médicaux performants pour l'imagerie diagnostique. Avec la mise à jour de la spécification 3.1, les modules COM Express Type 6 peuvent désormais fournir en option des connecteurs MIPI-CSI sur le module.

Modules COM Express Type 10 Mini

Le plus petit format de la spécification - COM Express Mini, mesurant 55 mm x 84 mm, est pris en compte par le brochage PICMG Type 10 et complète l'ensemble des spécifications COM Express pour les conceptions SFF. Ces modules sont conçus pour les processeurs Intel® Atom™ et Celeron® basse consommation. Ici aussi, les modules peuvent exécuter deux connecteurs MIPI CSI avec COM Express 3.1. Comme la même technologie de connecteurs et les mêmes guides de conception sont exploités dans l'ensemble de l'écosystème PICMG COM Express, les développeurs sont en mesure de réutiliser un nombre important de fonctions, ce qui constitue le principal avantage de cette mini-spécification. Toutefois, les normes SGET SMARC et Qseven sont plus largement établies et utilisées ; ces deux normes prennent en charge les processeurs d'application x86 et ARM.

SMARC pour la vision embarquée

SMARC s'adresse au haut de gamme des applications SFF. Cette norme a fait l'objet d'une mise à jour majeure avec la révision 2.1. Cette nouvelle révision présente de nombreuses nouvelles fonctionnalités, telles que la prise en charge de SerDes pour une connectivité périphérique étendue et deux interfaces supplémentaires sur le module qui peuvent être utilisées pour connecter un total de 4 caméras MIPI-CSI afin de répondre à la demande croissante de réunir l'informatique et la vision embarquées. Ces nouvelles fonctionnalités sont rétrocompatibles avec la Rev. 2.0 et toutes les extensions de la Rev. 2.0 sont optionnelles, de sorte que tous les modules SMARC 2.0 de congatec sont automatiquement compatibles avec SMARC 2.1. Outre les deux interfaces MIPI sur le connecteur, le SMARC se distingue également par la prise en charge des interfaces sans fil telles que WLAN et Bluetooth directement sur le module. Les processeurs idéals pour les modules SMARC sont les derniers processeurs Intel Atom ou toute la gamme des nouveaux processeurs d'application i.MX 8.

230071-04

Figure 4. Carte porteuse SMARC 2.1 et carte SBC évolutive de 3,5 pouces : le conga-SMC1 comble le fossé entre les conceptions basées sur des modules et les cartes standard hautement évolutives disponibles dans le commerce.

démarrer en électronique

composants actifs

Eric Bogers (Elektor)

Comme nous l'avons déjà mentionné dans un épisode précédent de cette série, tous les composants qui contiennent une puce semi-conductrice (tels que les diodes) sont appelés composants actifs. En fait, ce n'est pas tout à fait exact si l'on se réfère à la définition précise du terme « actif ». Dans ce cas, seuls les composants qui amplifient un courant ou une tension sont réellement actifs. Et maintenant – avec les transistors – nous entrons dans le monde des composants actifs.

Le mot transistor est un terme composé des mots « transfert » et « résistance ». Cela signifie que l'on pourrait ludiquement appeler un transistor une « résistance de transfert », mais heureusement, personne ne le fait. Cependant, vous pouvez réellement faire varier la résistance de ce composant en appliquant une tension de commande. Ce sont les transistors qui ont rendu possible le progrès actuel en électronique, parce qu'ils sont beaucoup plus petits et moins fragiles que les tubes, et parce qu'ils consomment beaucoup moins d'énergie. Les transistors sont également à l'origine des circuits intégrés (CI). Ces derniers contiennent un grand nombre de composants (actifs et passifs) présents sur une puce ou dans un boîtier. Dans les microprocesseurs modernes, on parle de plusieurs millions de transistors contenus dans un seul boîtier.

Commençons par les transistors comme composants individuels. Nous reviendrons sur les circuits intégrés dans un prochain épisode.

Figure 1. Divers transistors (Source : Shutterstock / Edkida).

Aujourd'hui, il existe des solutions toutes faites pour de nombreuses applications sous la forme de circuits intégrés, mais vous devez souvent fouiller dans vos tiroirs de composants pour trouver des transistors – par exemple, lorsque vous avez besoin de transistors de sortie pour un amplificateur de puissance.

Plusieurs transistors sont représentés sur la **figure 1.** En règle générale, la puissance que peut supporter un transistor dépend de sa taille et de la présence d'un boîtier métallique.

Utilisation d'un transistor bipolaire en commutation

En général, lorsqu'on parle d'un transistor, on fait référence à un transistor bipolaire. Il en existe deux types : NPN et PNP, où les lettres désignent la succession des couches de silicium dopées N ou P de la structure du transistor.

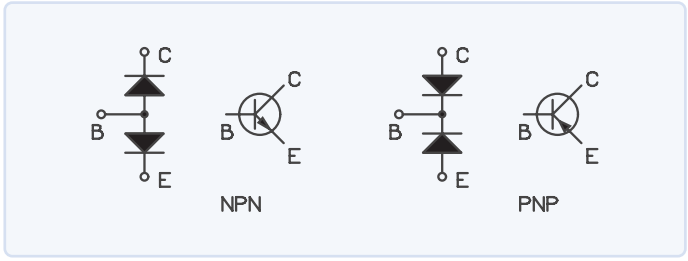


Figure 2. Circuits équivalents des transistors. C = collecteur, B = base, E = émetteur.

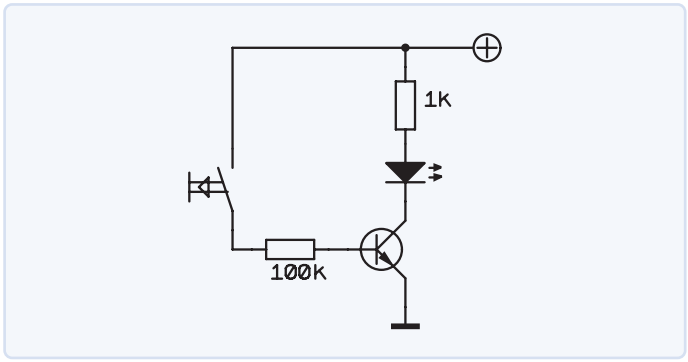


Figure 3. Transistor utilisé comme interrupteur.

Pour simplifier, il est possible d'utiliser les transistors comme interrupteurs ou comme amplificateurs. Nous allons d'abord nous intéresser à l'utilisation d'un transistor comme interrupteur, car le principe est plus facile à comprendre. Ensuite, nous aborderons les propriétés des transistors en tant qu'amplificateurs.

La **figure 2** montre un transistor NPN à gauche et un transistor PNP à droite, représentés chacun par une combinaison de deux diodes. En pratique, un transistor n'est jamais utilisé pour remplacer deux diodes, mais, il est utile de garder ces circuits équivalents en mémoire lorsque vous utilisez un testeur de continuité pour vérifier le type du transistor (ou pour tester s'il fonctionne encore ou s'il est court-circuité).

La **figure 3** montre un transistor utilisé comme interrupteur. Notez que nous utilisons toujours des transistors NPN dans les exemples de cet article, car leur fonctionnement est plus simple à comprendre lorsqu'on considère la circulation conventionnelle du courant (du positif au négatif). Cependant, il est possible de modifier facilement chacun de ces exemples en un montage à transistor PNP en changeant simplement la polarisation de tous les composants polarisés (diodes, condensateurs électrolytiques et, bien entendu, la tension d'alimentation).

Revenons à la **figure 3**. Ici, il y a deux circuits : un pour le courant de commande et un pour la charge. Le courant de commande circule à travers l'interrupteur, la résistance série et la diode base-émetteur du transistor. Cette jonction base-émetteur commande le transistor. Lorsqu'un courant circule à travers cette jonction, le transistor est conducteur, sinon le transistor est bloqué.

La jonction base-émetteur fonctionne comme une diode ordinaire. Cela signifie que le courant ne commence à circuler que lorsque la tension au niveau de la jonction atteint environ 0,7 V.

L'autre circuit est constitué de la résistance série de la LED, de la LED et de la jonction collecteur-émetteur du transistor. Vous pensez peut-être qu'aucun courant ne circule à ce niveau car la diode collecteur-base est polarisée en sens inverse. En pratique, le courant ne traverse pas la diode collecteur-base, mais circule directement du collecteur à l'émetteur, il n'y a donc pas de tension de 0,7 V au niveau de la jonction collecteur-émetteur.

Cela fonctionne grâce à la disposition particulière des différentes couches de semi-conducteurs ; après tout, il est impossible de construire un transistor à partir d'une paire de diodes séparées.

Vous vous demandez peut-être (pour une bonne raison) pourquoi nous utilisons un transistor et une résistance supplémentaire dans le circuit de la **figure 3** au lieu de commander la LED directement avec l'interrupteur. Et vous avez raison : c'est tout à fait inutile dans cet exemple spécifique.

Le courant traversant l'interrupteur dans ce circuit est 100 fois moins élevé que le courant traversant la LED, bien qu'il soit difficile de trouver un interrupteur qui ne puisse pas supporter le courant traversant la LED. Cependant, le circuit présenté ici convient également à la commutation des courants (et des tensions) beaucoup plus élevés. Il n'y a aucune raison pour que le circuit de commande et le circuit de charge soient alimentés par la même source de tension. En bref, le transistor permet de commuter une tension beaucoup plus élevée avec une tension de commande de quelques volts à condition d'utiliser un transistor approprié.

Multivibrateur bistable (Flip-Flop)

Passons maintenant à quelques exemples de multivibrateurs, c'est-à-dire de circuits qui commutent entre deux états définis. La **figure 4** montre un multivibrateur bistable, plus connu sous le nom de flip-flop.

Ce circuit est dit bistable car il possède deux états stables – soit la LED de gauche est allumée, soit la LED de droite est allumée. Le circuit ne changera jamais d'état sauf si vous appuyez sur l'un des deux boutons-poussoirs.

Supposons que le transistor de gauche soit conducteur, ce qui signifie que la LED de gauche est allumée. La tension entre le collecteur et l'émetteur de ce transistor est donc faible (environ 0,1 V), ce qui n'est pas suffisant pour activer le transistor de droite (au moins 0,7 V

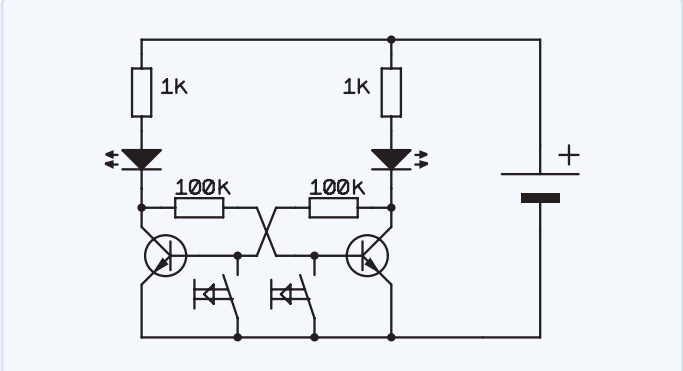


Figure 4. Multivibrateur bistable.

est nécessaire). Le transistor de droite est donc bloqué et sa tension collecteur-émetteur est quasiment égale à la tension d'alimentation U_V. Cela signifie que le transistor de gauche reste toujours passant grâce au courant traversant la résistance de 100 kΩ du côté droit. Il s'agit donc d'un état stable.

Cependant, si vous appuyez sur le bouton de gauche, la tension base-émetteur du transistor de gauche chute à zéro. Par conséquent, ce transistor se bloque et sa tension collecteur-émetteur augmente jusqu'à environ U_V. La LED de gauche s'éteint, et le transistor de droite devient conducteur grâce au courant traversant la résistance de 100 kΩ du côté droit. Sa tension émetteur-connecteur chute à presque 0 V, et la LED de droite s'allume. Maintenant, le transistor de gauche ne peut plus conduire, même si vous relâchez le bouton-poussoir. Il s'agit encore d'un état stable, mais opposé à l'état initial.

Le circuit reste dans cet état stable jusqu'à ce que l'on appuie sur le bouton de droite, ce qui fait basculer le circuit vers l'autre état stable.

Maintenant, la seule question restant à trancher est de savoir quel transistor s'allume en premier après l'application de la tension d'alimentation. Il est impossible de le prévoir à partir du schéma de principe. Si les deux parties du circuit comportent des composants identiques, ce serait purement une question de hasard. En d'autres termes, cela dépendrait du bruit imprévisible des composants. Dans la pratique, il n'existe pas de composants exactement identiques. Les résistances ont des tolérances de fabrication, et les gains en courant des transistors sont différents (en plus, le gain en courant dépend de la température, et donc du dernier transistor conducteur et qui est probablement plus chaud que l'autre).

Essayons plutôt de retourner la question : comment faire en sorte que le même transistor s'allume toujours en premier lorsque la tension d'alimentation est établie ? La réponse est simple : réduisez la résistance de la base du transistor concerné à la moitié de la valeur indiquée sur le schéma de principe.

Voilà, c'est tout pour cet épisode. Dans le prochain article, nous étudierons deux autres circuits de multivibrateurs, puis nous verrons comment utiliser les transistors comme amplificateurs.

Bref retour sur la Diac

Dans l'édition de novembre/décembre de l'année dernière [1], nous avons brièvement présenté la diac en tant que type de diode (désormais plus ou moins obsolète). Dans cet article, nous écrivions :

« Dans votre e-choppe d'électronique préférée (malheureusement, le magasin d'électronique du coin n'existe plus), vous pouvez acheter un composant portant le nom simple mais exotique de "diac", qui contient deux diodes Zener connectées de cette manière. »

Deux lecteurs d'Elektor (Pablo Medina et Maurizio Spagni), nous ont écrits individuellement que ce n'était pas correct. Et tous les deux ont tout à fait raison : un diac est un composant assez complexe avec une caractéristique de résistance négative, ce qui le rend approprié pour amorcer des thyristors et des triacs [2].

Comparer un diac à deux diodes Zener connectées en série en inverse est donc une simplification excessive. Néanmoins, nous pensons que considérer le diac comme deux diodes Zener, plutôt que comme une résistance négative, facilite la compréhension de son fonctionnement par les électroniciens débutants. Au moment où ils auront compris ce qu'est une caractéristique de résistance négative et comment un dispositif semi-conducteur peut avoir ce genre de caractéristique, ils auront probablement abandonné l'électronique.

Mais comme nous l'avons déjà mentionné, les deux lecteurs ont tout à fait raison, et à l'avenir, nous choisirons soigneusement nos mots.

220670 - VF : Asma Adhimi

Note de l'éditeur : 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).

LIENS

[1] Eric Bogers, Michael Ebner: « démarrer en électronique… avec plaisir, on continue avec les Zener » ElektorMag 11-12/2022 : https://www.elektormagazine.fr/magazine/elektor-283/61176

[2] Le Diac sur Wikipédia : https://fr.wikipedia.org/wiki/Diac

communication I²C avec Node.js et Raspberry Pi

affichez les données de vos capteurs dans un navigateur

Franck Bigrat (France)

Le bus I²C existe depuis les années 80 et est pris en charge par les plates-formes populaires actuelles, notamment le Raspberry Pi. Grâce à Node.js, il est possible de créer un serveur Web en JavaScript et de transmettre des données à un réseau local ou à Internet. Cet article vous explique comment connecter facilement un capteur I²C à un Raspberry Pi et afficher les résultats dans un navigateur Web.

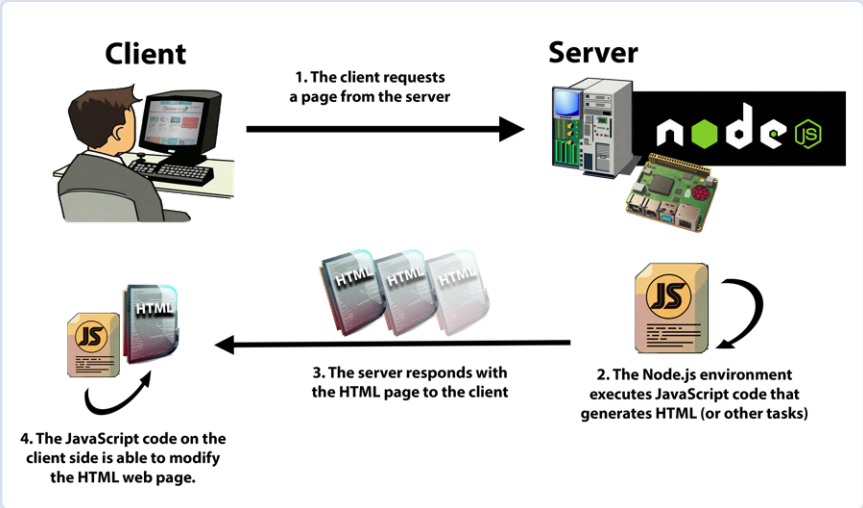


Figure 1. Flux de communication du serveur Node.js. (Source : sdz.tdct.org)

Dans ce projet, nous utilisons un simple Raspberry Pi Zero et le capteur de température I²C DS1621. Le résultat de la mesure est affiché sur une page web et est accessible sur une tablette ou un smartphone connecté au même réseau wifi.
Node.js dispose de modules tels que *http* et *express*, qui permettent au développeur de créer des serveurs web en toute simplicité. Nous n'avons donc pas besoin de recourir au serveur Apache qui est très répandu.
Bien qu'il existe un module dédié au contrôle du capteur DS1621, il est plus intéressant d'utiliser le module *i2c-bus* de Node.js, qui nous permet d'étendre le projet et d'y intégrer une grande variété d'appareils I²C.

Qu'est-ce que Node.js ?

Expliquer en quoi consiste Node.js en quelques mots n'est pas une tâche facile.

De nombreux livres et sites web sont consacrés à Node.js. Souvenez-vous simplement que Node.js est du *JavaScript exécuté par le serveur web, il n'est donc pas exécuté par un navigateur web lancé côté client* (cependant, le navigateur peut également exécuter du code JavaScript si nécessaire). Il est basé sur le moteur JavaScript de Google Chrome (V8). En termes simples, le développeur crée des applications JavaScript, mais celles-ci s'exécutent sur le serveur. La **figure 1** en donne un aperçu.

Pour apprendre (presque) tout sur Node.js, je vous recommande vivement de visiter les sites web [1] et [2].
Vous pouvez également vous référer à ces livres :

- *Node.js in Practice,* Alex Young et Marc Harter (Manning)
- *Beginning Node.js,* Basarat Ali Sayed (Apress)
- *The Node Beginner Book,* Manuel Kiessling
- *Instant Node.js Starter,* Pedro Teixeira (Packt Publishing)

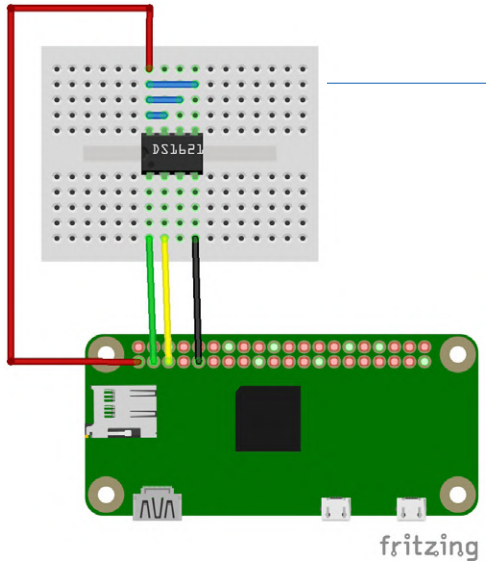


Figure 2. Schéma de connexion du DS1621 au Raspberry Pi Zero W.

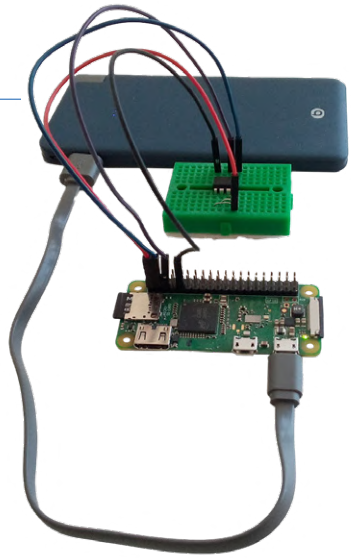


Figure 3. Le circuit est monté, avec la batterie.

I²C et SMBus

Dans ce paragraphe, je supposerai que vous êtes déjà un peu familier avec le protocole I²C (*Inter-Integrated Circuit*). Le plus important à savoir est que l'I²C a été développé par Philips dans les années 80 pour minimiser le nombre de fils entre un microprocesseur et un tas d'autres circuits intégrés. C'est un protocole de transmission de données synchrone, série, à deux fils (lignes de données et d'horloge). Il est possible de connecter plusieurs circuits, simultanément, aux mêmes lignes de données et d'horloge.

Les détails des fonctions du module *i2c-bus* de Node.js sont disponibles sur [3].

Si vous lisez attentivement, vous pouvez voir que certaines fonctions ont un préfixe *smbus* ou sont incluses dans la section SMBus. SMBus permet la communication entre le processeur d'un ordinateur et différents périphériques, notamment des appareils de gestion de l'énergie, des capteurs de température ou des ventilateurs.

Au niveau matériel, le Raspberry Pi prend en charge ce protocole : le bus de gestion du système est dérivé du bus I²C, et les deux protocoles sont tout à fait compatibles. Mais il existe quelques différences entre le SMBus et l'I²C ; elles sont expliquées dans le document de Texas Instruments [4].

Connexion et configuration des composants

Il est temps de connecter le capteur de température DS1621 au Raspberry Pi comme le montre la **figure 2**.

N'oubliez pas de relier les broches 5, 6 et 7 (A0, A1, A2) du DS1621 à V_{CC} pour définir l'adresse I²C du capteur à *4F* (hexadécimal). Le circuit réalisé est illustré dans la **figure 3**.

Configuration du Raspberry Pi

1. Téléchargez et installez *Raspberry Pi Imager* sur : *https://raspberrypi.com/software*

2. Lancez *Raspberry Pi Imager* (**figure 4a**), cliquez sur *CHOOSE OS* puis cliquez sur *Raspberry Pi OS (Other)* dans la liste qui s'affiche (**figure 4b**), puis *Raspberry Pi OS Lite* (32-bit) dans le sous-menu.

3. Cliquez sur *CHOOSE STORAGE*, puis sélectionnez votre carte microSD dans la boîte de dialogue *Storage* (**figure 4c**).

Figure 4a. Raspberry Pi Imager.

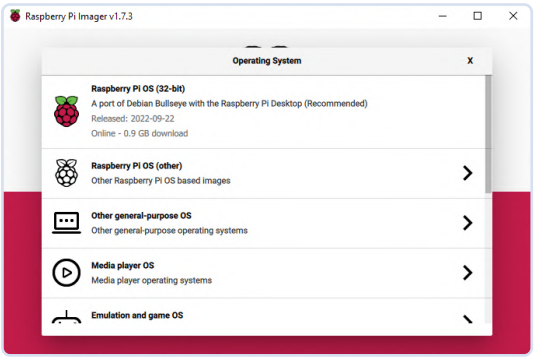


Figure 4b. Choix du système d'exploitation.

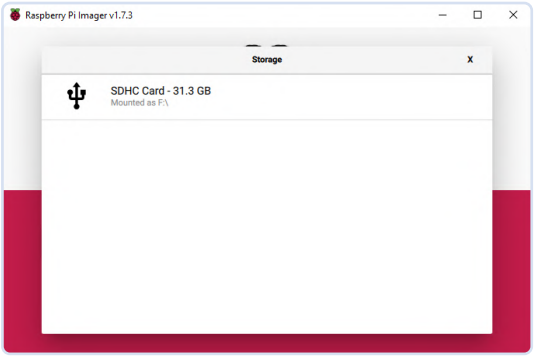


Figure 4c. Choix du support de stockage.

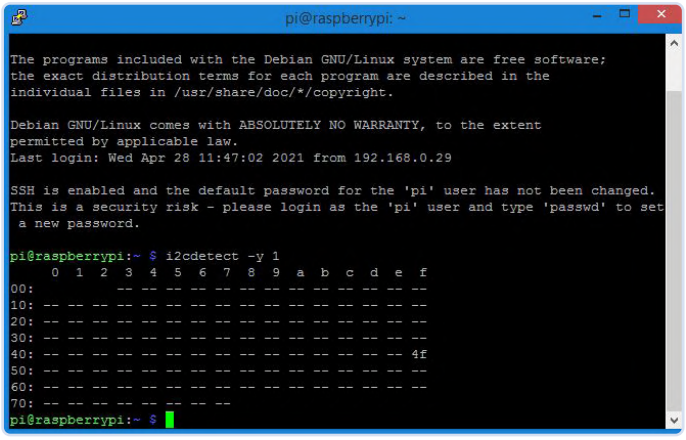


Figure 5. Sortie du programme i2cdetect.

4. Cliquez sur l'icône de paramètres (rouage) sur le côté droit dans l'application et définissez les options de configuration initiale du Raspberry Pi :
   - Cochez *Set hostname* : et entrez le nom d'hôte que vous voulez ; *raspberry* fera l'affaire.
   - Cochez la case *Enable SSH* et sélectionnez le bouton d'option *Use password authentication*.
   - Cochez la case *Set username and password*, puis saisissez le nom d'utilisateur (*pi* convient) et le mot de passe (choisissez un nom facile à retenir, par exemple *raspberry*).
   - Cochez la case *Configure wireless LAN* et saisissez le SSID et le mot de passe de votre réseau wifi.
   - Cochez la case *Set locale settings* et choisissez votre fuseau horaire (*Time zone*) et la disposition du clavier (*Keyboard layout*) si nécessaire, ou laissez les valeurs par défaut.
   - Cliquez sur le bouton *SAVE*.

5. Installez le système d'exploitation en cliquant sur *WRITE*, puis confirmez par *YES*.

6. Quand le processus d'écriture est terminé, retirez la carte microSD.

7. Insérez la carte dans la fente microSD du Raspberry Pi et mettez le Pi sous tension.

8. Trouvez l'adresse IP du Raspberry Pi sur le réseau avec un outil tel que Advanced IP Scanner ou, mieux encore, à partir des paramètres du bail DHCP de votre routeur wifi.

9. Utilisez un client SSH tel que Putty pour vous connecter au Raspberry Pi en utilisant son adresse IP et son nom d'utilisateur/mot de passe (*pi* et raspberry, respectivement, dans notre exemple).

10. Pour utiliser la totalité de l'espace de stockage de la carte microSD, étendez le système de fichiers :
    - Lancez l'outil *raspi-config* : `sudo raspi-config`
    - Choisissez *Advanced Options*, puis l'option *Expand Filesystem*.
    - Valider par la touche Entrée.
    - Passez à l'onglet <*Finish*> et confirmez par *Entrée*, puis confirmez le redémarrage en passant à l'onglet <*Yes*> et en appuyant sur Entrée.

11. Pour activer le bus I²C :
    - Ouvrez l'outil *raspi-config* : `sudo raspi-config`
    - Sélectionnez *Interfacing Options* et ensuite I²C *(Enable/disable automatic loading of I2C kernel module)*.
    - Confirmez par la touche *Entrée*.
    - Confirmez à nouveau en passant au bouton <*Yes*> et en rappuyant sur la touche Entrée.
    - Terminez par <*OK*> et ensuite <*Finish*>.

12. Connectez le Raspberry Pi au DS1621 selon notre schéma dans la **figure 2**.

13. Vérifiez la communication I²C :
    - Commencez par faire une mise à jour globale : `sudo apt update`
    - Installez les outils de diagnostic I²C : `sudo apt install i2c-tools` (vous devrez peut-être redémarrer le Raspberry Pi d'abord).
    - Exécutez la commande : `i2cdetect -y 1`
    - Dans la liste affichée, le capteur doit indiquer l'adresse que nous avons câblée, *4f* (**figure 5**).

14. Maintenant, installez Node.js et le gestionnaire de paquet NPM (*Node Package Manager*) : `sudo apt install -y nodejs npm` (cela peut prendre un peu de temps).

15. Vérifiez les versions avec `node -v` suivi de `npm -v`

16. Installez les modules *express*, *i2c-bus*, et *sleep* :
    `npm install express`
    `npm install i2c-bus`
    `npm install sleep` (peut générer des avertissements)

17. Créez un répertoire nommé *Documents* et un sous-répertoire nommé *NodeJS.*

18. Avec un client FTP tel que FileZilla, copiez les fichiers *DS1621_V3.js* et *DS1621_V4.html* dans le dossier *NodeJS*.

Et voilà, faisons un test !

Une fois que vous avez connecté tous les éléments, configuré le Raspberry Pi, installé Node.js et les différents modules, et copié les fichiers *.js* et *.html* sur le Raspberry Pi, ouvrez le répertoire *Documents/NodeJS* et exécutez le script avec `node DS1621_V3.js`.
Il suffit de connecter un PC, une tablette ou un smartphone à votre réseau wifi, d'ouvrir votre navigateur préféré et de saisir l'adresse : *[Your_Raspberry_Pi_IP_address]:8080*
Si vous avez fait les étapes précédentes correctement, le résultat devrait ressembler à la **figure 6**.

Figure 6. En attente de données.

Figure 7. Mesure et affichage de la température.

Après quelques secondes, la température sera affichée (**figure 7**). À partir de ce moment, la valeur de la température sera rafraîchie toutes les cinq secondes.

Explication des scripts Node.js et HTML

Avec Node.js, ce qui est important, mais difficile, c'est de comprendre l'exécution du script. En Node.js, l'exécution d'un script n'est pas séquentielle. Voir la **figure 8** : les instructions de la fonction `Server.get('/',...)` ne sont pas exécutées avant les instructions de la fonction `Server.get('/temp',...)`, qui ne sont pas exécutées avant les instructions suivantes.
Les instructions de la fonction `Server.get(«/»,...)` ne sont exécutées que si le serveur reçoit la requête *[Rpi_IP_Address]:8080* du client. Les instructions de la fonction `Server.get(«/temp»,...)` ne sont exécutées que si le serveur reçoit la requête *[Rpi_IP_Address]:8080/temp* du client. C'est ce qu'on appelle « la programmation événementielle ».
Dans les cinq premières lignes du script Node.js (partie gauche de la **figure 8**), nous importons les modules Node.js nécessaires et créons différents objets :

- Nous créons un objet `I2C` en utilisant le module `i2c-bus` :

```
const I2C = require('i2c-bus');
```

- Nous créons un objet `express` avec le module `express`. Ce module est un *framework* destiné à la création d'applications web avec Node.js :

```
var express = require('express');
```

- Nous créons un objet `fs` avec le module `fs` (*file system*). Cela nous permet de lire et d'écrire des fichiers sur la carte microSD :

```
var fs = require('fs');
```

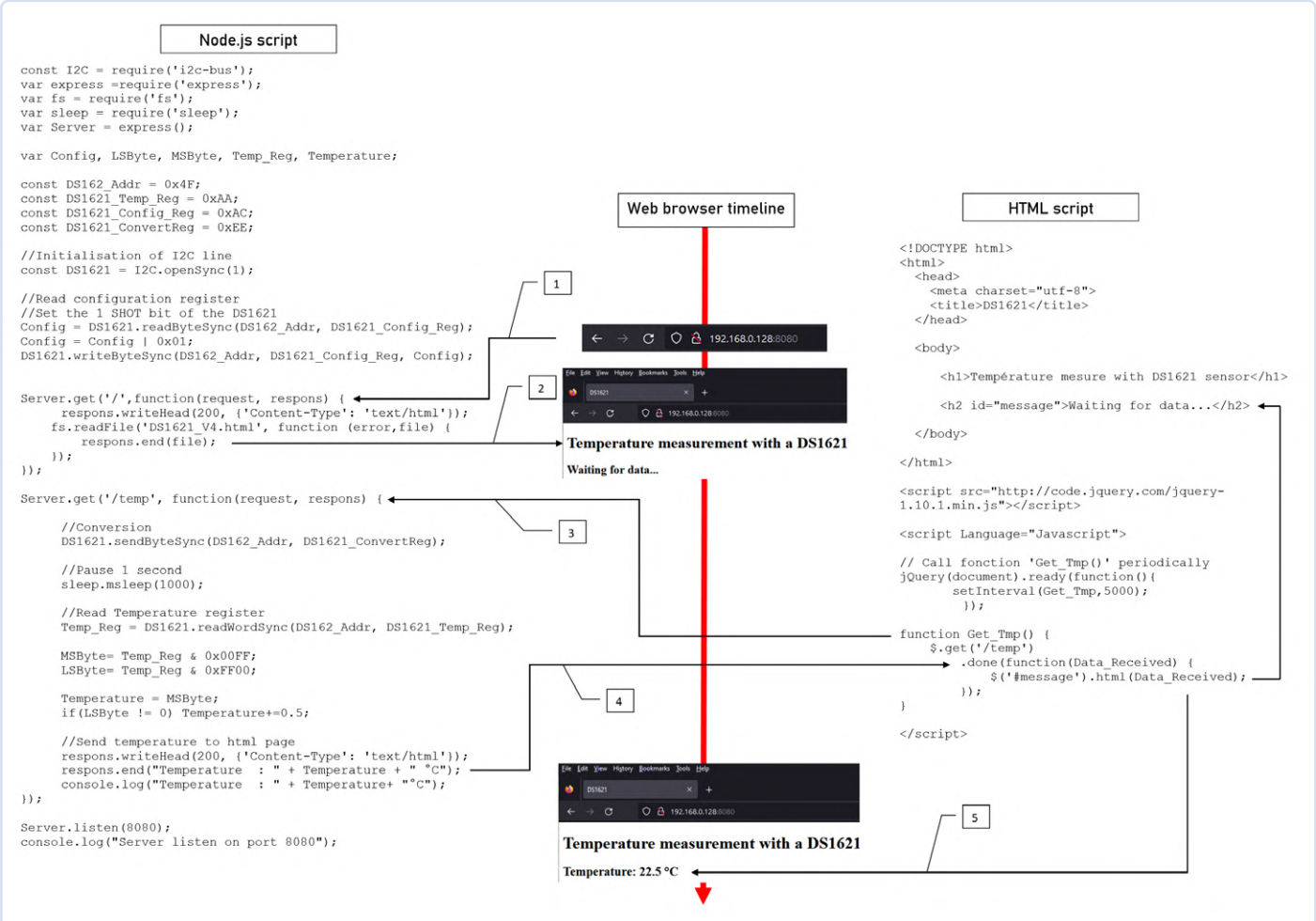


Figure 8. Ligne de temps des évènements.

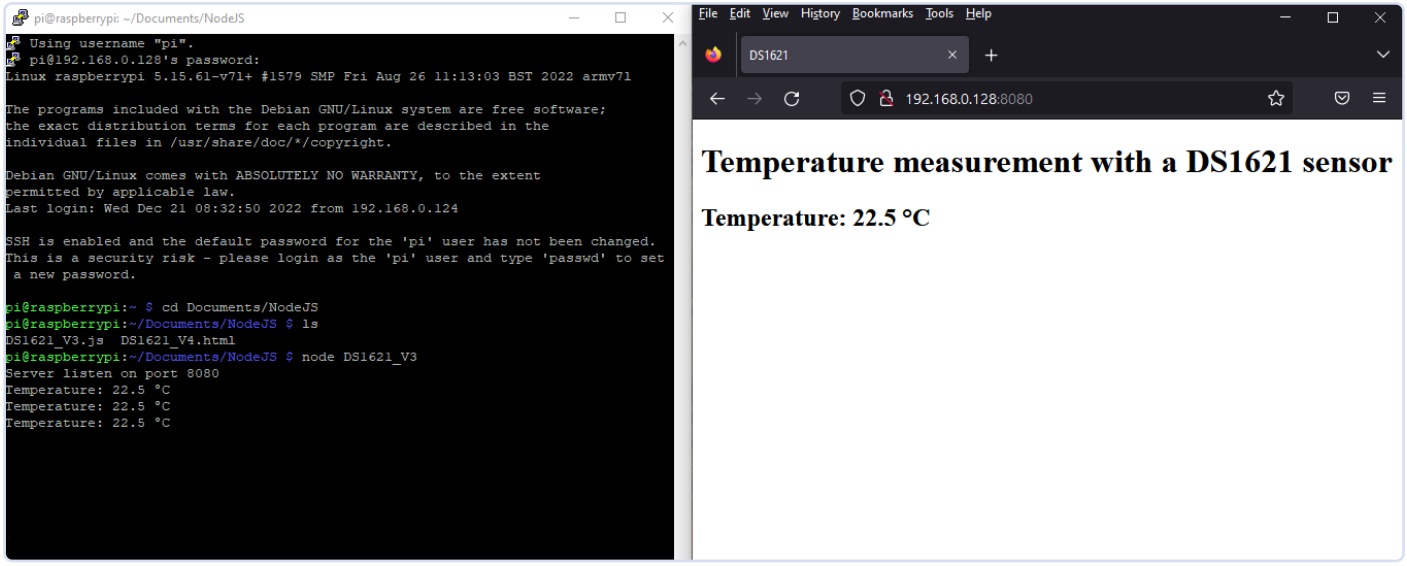


Figure 9. Terminal Linux et navigateur web.

› Nous créons un objet `sleep` avec le module `sleep`. Ce module nous permet de créer des délais :

```
var sleep = require('sleep');
```

› Enfin, nous créons un objet `Server` :

```
var Server = express();
```

Dans les cinq lignes suivantes, nous définissons 5 variables et 4 constantes. Ces constantes contiennent les adresses respectives des registres du DS1621.

```
var Config, LSByte, MSByte, Temp_Reg, Temperature;

const DS162_Addr = 0x4F;
const DS1621_Temp_Reg = 0xAA;
const DS1621_Config_Reg = 0xAC;
const DS1621_ConvertReg = 0xEE;
```

Ensuite, nous ouvrons le bus I²C et créons un objet appelé `DS1621`. Nous utiliserons cet objet pour communiquer avec le capteur via I²C.

```
const DS1621 = I2C.openSync(1);
```

Les trois lignes suivantes définissent le bit `1SHOT` du DS1621. Ainsi, la température est mesurée et convertie une fois (uniquement) lorsque le maître I²C (le Raspberry Pi) exige une conversion.

```
Config = DS1621.readByteSync(DS162_Addr,
         DS1621_Config_Reg);
Config = Config | 0x01;
DS1621.writeByteSync(DS162_Addr,
         DS1621_Config_Reg, Config);
```

Maintenant, observons ce qui se passe lorsque le navigateur envoie une requête au serveur (**figure 9**).
Lorsque le script Node.js est exécuté, le message « *Server listen on port 8080* » s'affiche dans le terminal Linux après quelques secondes. Ouvrez un navigateur web sur votre PC ou votre tablette connectée au réseau wifi. Dans la barre d'adresse, saisissez *http://* et l'adresse IP du Raspberry Pi et le port sur lequel le serveur écoute ; par exemple *http://192.168.0.128:8080*. Le port 8080 a été choisi pour être différent de celui utilisé par défaut par le serveur web Apache (Port 80).

1. Lorsque vous soumettez l'adresse IP, le navigateur envoie une requête *HTTP get* (obtenir HTTP) au serveur. Que se passe-t-il ?
   a. Le serveur détecte la requête avec `Server.get('/'...)` et exécute la fonction de rappel anonyme, `function(request, respons)`.
   b. L'objet `request` contient la requête envoyée par le navigateur.
   c. Dans l'objet `respons`, nous envoyons deux en-têtes comme paramètres :
      i. Un code http *200*, signifiant que la requête a abouti
      ii. La chaîne « *'Content-Type' : "text/html"* », signifie que le navigateur est prêt à recevoir des données texte de type HTML.
   d. L'appel de la fonction `respons.writeHead()` renvoie les données des paramètres au navigateur.

2. La fonction `fs.readFile()` lit le fichier DS1621_V4.html. La fonction de rappel anonyme, `function(error, file)`, est exécutée et, en cas de succès (nous le supposerons), le fichier est lu et placé dans l'objet *file*. L'appel de la fonction `respons.end(file)` envoie le code HTML au navigateur, qui affiche la page Web. En cas d'erreur, un code d'erreur est placé dans l'objet `error`.

La page Web est désormais au format HTML dans le navigateur. Que se passe-t-il ensuite ? Jetons un coup d'œil au script HTML (côté droit de la **figure 8**).
La première chose que nous voyons est la ligne `<h2 id="message">Waiting for data...</h2>`. Il s'agit d'un *placeholder* d'en-tête dans lequel la température sera écrite. Nous découvrirons comment plus tard.
Ensuite, nous voyons le bloc qui suit. Il s'agit d'un script écrit en JavaScript, mais il est du côté client, ce qui signifie qu'il sera exécuté par le navigateur :

```
<script src="http://code.jquery.com/jquery-1.10.1.min.
js"></script>

<script Language="Javascript">
```

```
  // Call function Get_Tmp() periodically:
  jQuery(document).ready(function() {
         setInterval(Get_Tmp, 5000);
          });

  function Get_Tmp() {
      $.get('/temp')
          .done(function(Data_Received) {
              $('#message').html(Data_Received);
          });
  }
</script>
```

D'abord, nous importons la bibliothèque *jQuery* du Web, qui nous offre des fonctions intéressantes. Grâce à jQuery, nous pouvons configurer un *timer* qui appellera la fonction `Get_Temp()` toutes les 5 secondes :

```
  jQuery(document).ready(function(){
       setInterval(Get_Tmp, 5000);
       });
```

Enfin, il y a la fonction `Get_Tmp()` :

```
function Get_Tmp() {
$.get('/temp')
     .done(function(Data_Received) {
          $('#message').html(Data_Received);
     });
}
```

Et c'est la partie intéressante du script !
Examinons comment elle interagit avec le script Node.js du côté serveur (voir la chronologie au centre de la **figure 8**) :

1. La fonction `Get_Tmp()` envoie une requête au serveur lorsqu'elle est appelée. Que se passe-t-il ensuite ?
   a. Le serveur détecte la demande avec `Server.get('/temp')` et exécute la fonction de rappel anonyme, `function(request, respons)`.
   b. Le maître I²C (Raspberry Pi) communique avec le DS1621 via le bus I²C, demande une conversion de données, les lit et les convertit en °C.
   c. L'objet de la requête contient la requête envoyée par le navigateur.
   d. Dans l'objet `respons`, nous stockons à nouveau nos deux paramètres :
      i. La valeur 200, est un code signifiant que la requête est réussie.
      ii. La chaîne « 'Content-Type' : "text/html" » signifie que le navigateur recevra du texte à interpréter comme du code HTML.
   e. La fonction `respons.writeHead()` renvoie ces données au navigateur.

2. L'appel de fonction `respons.end("Temperature : " + Temperature + " °C")` envoie le message « *Temperature : XX °C* » (qui est une chaîne ASCII) au navigateur. *XX* est la température mesurée par le DS1621.

3. Cette chaîne est placée dans l'objet `Data_Received`.

4. Vous vous rappelez la section d'en-tête `<h2 id="message">Waiting for data...</h2>` ? Nous voyons que cette balise d'en-tête comporte l'`id` du `message`, donc lorsque la chaîne est reçue, grâce à la ligne `$('#message').html(Data_Received) ;`, la chaîne envoyée par le serveur est placée dans la section d'en-tête et la page Web affiche la température.

Pour finir, expliquons un peu les deux dernières lignes du script Node.js :

```
Server.listen(8080);
console.log("Server listen on port 8080");
```

La ligne `Server.listen(8080) ;` lance le serveur, et la ligne `console.log("Server listen on port 8080") ;` écrit le message « *Server listen on port 8080* » dans la console Linux (shell).
Examinez bien la **figure 8**, je pense qu'une ligne de temps vous facilitera la compréhension.

210367-04 — VF : Asma Adhimi

Des questions, des commentaires ?

Contactez Elektor (redaction@elektor.fr).

LIENS

[1] Introduction to Node.js: A Beginner's Guide to Node.js and NPM: https://elektor.link/udaynode
[2] What is Node.js: A Comprehensive Guide: https://simplilearn.com/tutorials/nodejs-tutorial/what-is-nodejs
[3] Module i2c-bus : https://npmjs.com/package/i2c-bus
[4] Compatibilité du SMBus avec un appareil I²C : https://ti.com/lit/an/sloa132/sloa132.pdf

sortie vidéo sur les microcontrôleurs (2)

sortie VGA et DVI

Mathias Claussen (Elektor Labs)

Les microcontrôleurs sont capables d'envoyer des pixels à un moniteur via une interface VGA ou DVI. Que ce soit avec l'ESP32 ou le RP2040 du Raspberry Pi Pico, bien plus que 256 couleurs sont possibles. Grâce aux sprites, aux tuiles et à quelques astuces, les microcontrôleurs d'aujourd'hui peuvent produire des œuvres graphiques au moins aussi sophistiquées que les consoles de jeu classiques des années 1990.

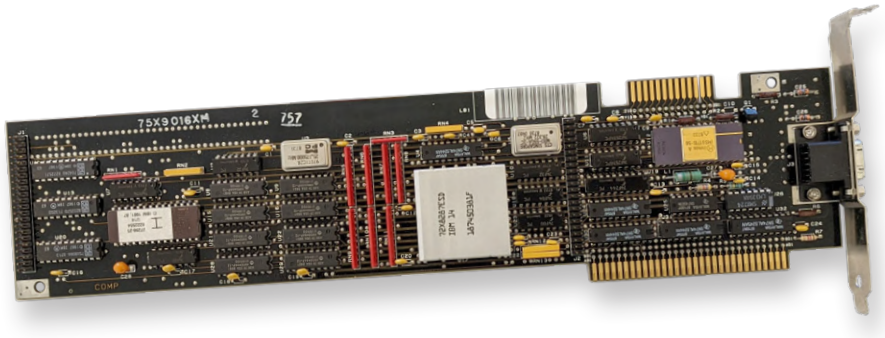


Figure 1. Carte VGA IBM (source : Wikimedia / Vlask, CCASA 4.0 https://elektor.link/cc4dot0).

La première partie de cet article traitait de la production de signaux vidéo composites aux standards PAL (**P**hase **A**lternating **L**ine) ou NTSC (**N**ational **T**elevision **S**ystems **C**ommittee). Alors que produire des images monochromes est même possible avec un Arduino UNO, les images en couleur sont plus exigeantes. Avec quelques astuces, on arrive à obtenir une sortie couleur d'un ESP32 ou même d'un ATmega, mais avec une qualité médiocre, surtout au niveau de la netteté, ce qui nuit à la lisibilité des textes. Les premiers ordinateurs grand public, tels que le Sinclair ZX81 et le Commodore 64, généraient de la vidéo composite. Cependant, pour les applications professionnelles, il fallait un affichage du texte aussi net que possible et, dans certains cas, des images en couleur. C'est ainsi qu'en 1987, IBM innova avec son interface VGA (**V**ideo **G**raphics **A**rray), dont les microcontrôleurs ont hérité. Certains microcontrôleurs peuvent même produire une image de manière entièrement numérique via DVI (**D**igital **V**isual **I**nterface). Notre deuxième partie de cette série s'intéresse donc au VGA, au DVI et aux les astuces utilisées pour l'émission et les effets graphiques.

VGA : Vidéo pour le PC IBM

VGA était la carte graphique (**figure 1**) pour les PC PS/2 introduite par IBM en 1987. Elle a rendu populaire le nouveau connecteur D-sub DE-15 à trois rangées (**figure 2**), bien que VGA ne spécifie pas seulement le connecteur physique, mais aussi les signaux nécessaires et leur *timing*.

Les signaux de couleur du VGA sont analogiques et sont transmis par trois lignes de signaux (rouge, vert et bleu). La synchronisation horizontale et verticale est effectuée numériquement à l'aide de signaux TTL distincts. La première carte VGA d'IBM pouvait émettre 256 couleurs à partir d'une palette RVB de 3 × 6 bits (= 262 144 couleurs) à une résolution de 320×200 pixels et parvenait encore à générer 640×480 pixels avec 16 couleurs. Les premiers moniteurs IBM qui convenaient ne pouvaient gérer que 640×480 ou 640×400 pixels à des taux de rafraîchissement de 60 ou 70 Hz. Les cartes VGA d'IBM ont rapidement été clonées par de nombreux autres fabricants, faisant du VGA une norme de facto pour la sortie vidéo sur ce qu'on appelait les PC compatibles IBM.

Pixels visibles et invisibles

Un signal vidéo de 640×480 pixels visibles à une fréquence de rafraîchissement de 60 Hz

Figure 2. Prise VGA sur un PC (source : Dr. Thomas Scherer).

Figure 3. Téléviseur ouvert avec tube cathodique (source : Shutterstock / Serhiy Stakhnyk).

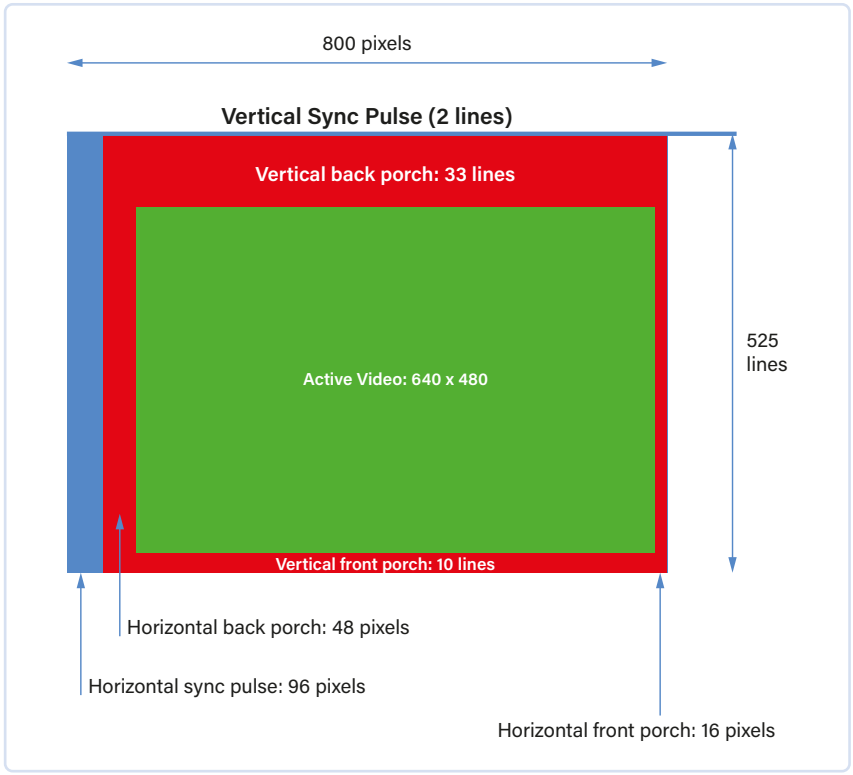


Figure 4. Timing d'un signal VGA avec 640×480 pixels visibles (source : https://elektor.link/VGAtiming).

fonctionne avec une fréquence de pixel de 25,175 MHz. Cependant, en multipliant les pixels d'une image par la fréquence d'images, on obtient seulement 18,432 MHz. D'où vient cette différence ?

Tout comme les signaux composites PAL et NTSC, le VGA comporte également des zones de signal non visibles, car cette norme a également été conçue pour les moniteurs basés sur des tubes cathodiques (**figure 3**). En principe, les exigences en matière de timing sont les mêmes que pour la commande du tube image d'un téléviseur.

La **figure 4** montre le *timing* pour le mode 640×480 pixels avec une horloge de pixels de 25,175 MHz. Une image commence par une impulsion de synchronisation verticale de 63,5551142 µs (= deux lignes d'image de 800 pixels chacune). Les trois faisceaux d'électrons sont éteints pendant ce temps, car ils se déplacent de la fin de la dernière image complète en bas à droite au début de la partie supérieure gauche de la nouvelle image. Les lignes commencent par une impulsion de synchronisation horizontale d'une durée de 69 pixels (≈ 3,81 µs). Dans les 33 lignes suivantes (lignes 3 à 35), l'impulsion de synchronisation horizontale est suivie de ce que l'on appelle le « palier arrière vertical » de 704 pixels dans ce cas (≈ 27,96 µs). Il est destiné à donner aux faisceaux d'électrons ou à la déviation magnétique suffisamment de temps pour revenir à la position de départ de la nouvelle image. Viennent ensuite 480 lignes de données d'image proprement dites, en commençant par l'impulsion de synchronisation horizontale de 96 pixels, puis le « palier arrière horizontal » de 48 pixels (≈ 1,9 µs), plus les 640 pixels d'image d'une ligne (≈ 25,42 µs) et le « palier avant horizontal » final de 16 pixels (≈ 0,64 µs). La **figure 5** montre la structure d'une telle ligne d'image. Enfin, il y a 10 lignes, chacune avec une impulsion de synchronisation horizontale suivie du « palier avant vertical » (à nouveau constitué de 704 pixels chacun). L'image transmise de 640×480 pixels visibles est donc en réalité constituée de 800×525 pixels en ce qui concerne le *timing*.

Les faisceaux d'électrons ne sont actifs que pendant l'image proprement dite et restent éteints le reste du temps. Même si cela semble complexe, il s'agit en fait de l'un des signaux d'image les plus simples pouvant être générés par un microcontrôleur. Pour que vous ne soyez pas surpris : sur Internet, vous pouvez trouver des spécifications de *timing* légèrement différentes pour la fréquence d'images VGA. Et, si vous divisez l'horloge de pixels de 25,175 MHz par les 800×525 pixels, le résultat

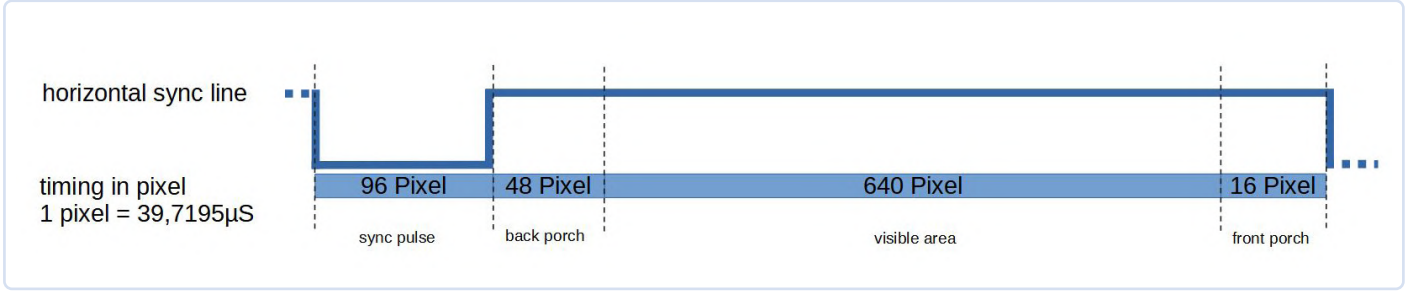


Figure 5. Timing d'une ligne en VGA.

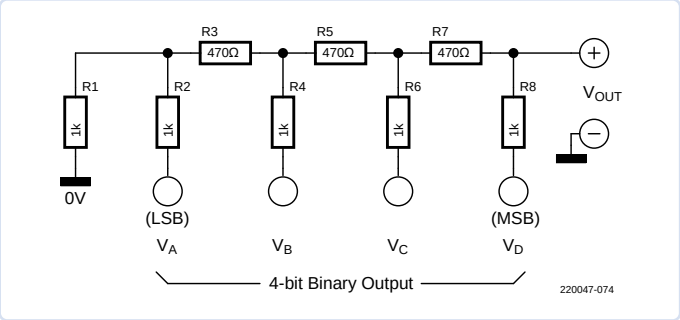


Figure 6. CNA avec échelle R2R

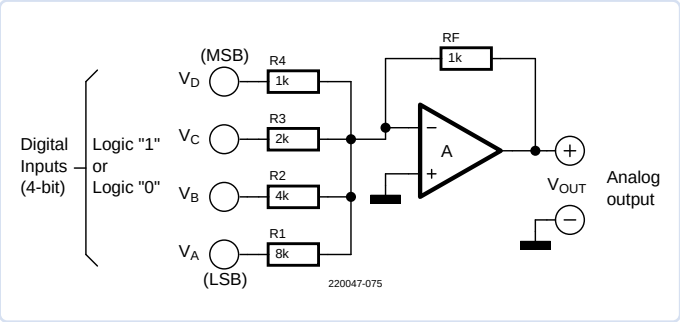


Figure 7. CNA à pondération binaire avec amplificateur opérationnel (source : https://elektor.link/BinaryWeightedDAC).

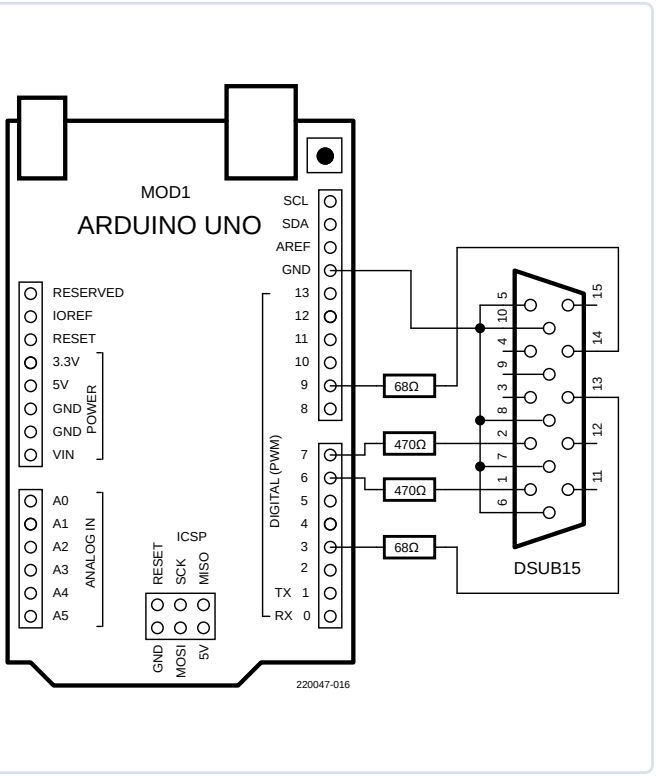


Figure 8. Câblage pour la sortie VGA sur un Arduino UNO.

n'est pas exactement 60 Hz pour la fréquence d'images, mais ≈ 59,94 Hz. Cependant, l'électronique analogique des moniteurs associés n'est pas trop exigeante à cet égard et se synchroniserait encore correctement même avec des écarts encore plus importants.

Niveaux du signal VGA

Selon la spécification VGA, la plage de tension des signaux d'image RVB analogiques est de 0 V à 0,7 V, où 0,7 V correspond à la luminosité maximale. Les signaux de synchronisation horizontale et verticale ont des niveaux TTL (0 V et 5 V).

Les microcontrôleurs dont la tension de fonctionnement est comprise entre 2,5 V et 5 V peuvent émettre les signaux de synchronisation verticale et horizontale directement via leurs broches d'E/S, puisque 2,5 V sont interprétés de manière fiable comme un niveau TTL « haut », alors que pour sortir les informations de luminosité variable pour chaque couleur, des diviseurs de tension sont nécessaires.

Les CNA internes de la plupart des microcontrôleurs n'étant pas assez rapides, les broches GPIO sont généralement utilisées comme CNA simples pour générer les trois signaux analogiques à l'aide de résistances. Lors du calcul des valeurs de résistance nécessaires, il faut tenir compte du fait que les entrées RVB analogiques du moniteur sont terminées par 75 Ω. Les CNA sont généralement mis en œuvre à l'aide de réseaux R2R (**figure 6**) ou de résistances à pondération binaire (**figure 7**).

VGA sur l'ATmega

Les signaux VGA peuvent même être générés par un microcontrôleur ATmega, de sorte qu'un Arduino UNO peut être équipé d'un périphérique VGA. Avec quatre résistances et un connecteur SUB-D à 15 broches, on peut faire en sorte qu'un Arduino UNO affiche 120×60 pixels en quatre couleurs. La bibliothèque Arduino correspondante est *VGAX* [1] et peut être téléchargée sur la page GitHub du projet.

Si un Arduino UNO est câblé comme indiqué sur la **figure 8**, il est même possible de produire des graphiques en quatre couleurs (**figure 9**). Le facteur limitant ici est la quantité de mémoire disponible. Comme les signaux de couleur sont générés par les broches GPIO, il faut être en mesure de les alimenter en nouvelles valeurs assez rapidement.

Dans ce cas, le *timing* de la génération du signal est géré par des interruptions, ce qui a pour effet secondaire que ces interruptions

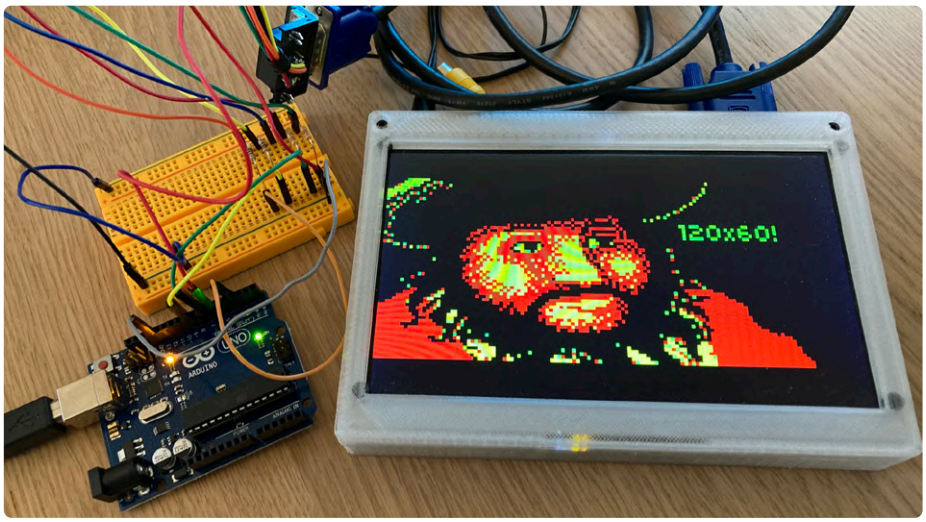


Figure 9. Sortie VGA sur un écran connecté à l'Arduino UNO.

peuvent être perturbées par celles d'autres périphériques. *Timer1* et *Timer2* génèrent les impulsions de synchronisation horizontale et verticale. *Timer0* est utilisé pour contourner la gigue dans les appels d'interruption.
La bibliothèque *VGAXUA* [2] pour l'Arduino UNO et l'Arduino Mega peut produire des images monochromes de 192×080 à 200×240 pixels. Pour gagner du temps de calcul, l'USART est utilisé ici pour générer l'image. Chaque octet dans la mémoire (d'image) correspond alors à 8 pixels. Des solutions plus anciennes pour la sortie VGA avec des puces AVR ont soit surcadencé le microcontrôleur jusqu'à 32 MHz [3], soit utilisé une mémoire externe pour la sortie vidéo [4].

VGA sur l'ESP32

Un ESP32 peut aussi faire du VGA. Les conditions idéales sont fournies par son horloge de 240 MHz et la matrice E/S flexible, qui permet de sortir des signaux numériques jusqu'à 80 MHz. La mémoire interne de l'ESP32, avec 520 Ko, est suffisante pour une sortie de 640×480 pixels avec 256 couleurs (= 307 200 octets), ce qui laisse encore assez de RAM pour d'autres logiciels.
La spécification du *timing* d'un signal VGA à 640 480 pixels et 60 Hz est disponible sur [5]. Le projet « ESP32 VGA » [6] propose une façon intéressante de produire un signal VGA en utilisant le contrôleur I²S interne (**I**nter-**IC** **S**ound) [6]. Outre l'émission de données audio, le contrôleur I²S de l'ESP32 peut servir d'interface pour une caméra ou un écran LCD (la **figure 10** détaille la partie correspondante du circuit synoptique).
L'interface LCD du contrôleur I²S peut fournir une largeur de bus allant jusqu'à 24 bits, ce qui serait même suffisant pour TrueColor (16 777 216 couleurs) si l'ensemble des 24 bits était disponible pour la sortie de la couleur. Toutefois, un signal VGA nécessite une synchronisation horizontale et verticale en plus des signaux RVB, ce qui ne laisse que 22 bits pour la couleur (**figure 11**).
Il reste deux obstacles à franchir : une image de 640×480 pixels occupe 921 600 octets de RAM en couleur 24 bits - c'est plus que ce qu'offre un ESP32. De plus, il faut atteindre une horloge de pixels de 25,175 MHz. Pour la sortie de 24 bits, le contrôleur I²S doit lire 32 bits par pixel dans la mémoire. Il semble que le contrôleur I²S de l'ESP32 puisse sortir les mots de 32 bits à un maximum de 10 MHz, limite de la vitesse de sortie série des pixels,

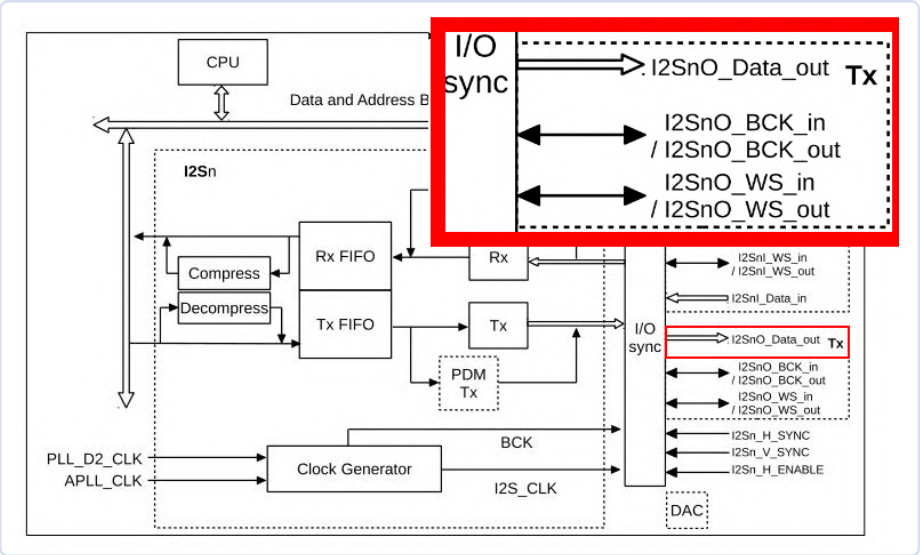


Figure 10. Le contrôleur I²S de l'ESP32 (source : Espressif – https://elektor.link/ESP32TechMan).

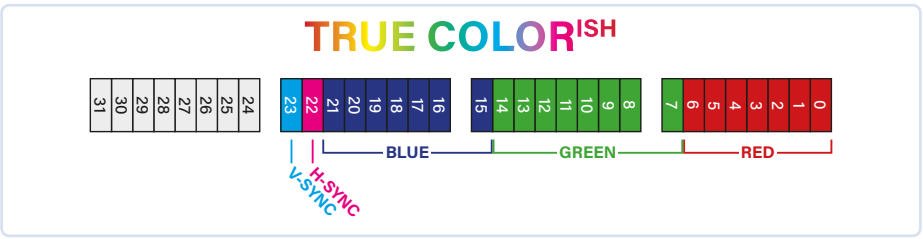


Figure 11. Sortie vidéo 22 bits avec H- et V-sync sur ESP32 (source : YouTube – https://elektor.link/YTESP32VGA).

ce qui est insuffisant. La résolution maximale, quant à elle, est limitée par la mémoire interne. Selon la liste des fréquences d'horloge de pixels en fonction de la résolution VGA sur [8], dans le mode avec 800×600 pixels à 60 Hz, l'horloge de pixels est à 40 MHz, soit exactement quatre fois le maximum d'un ESP32. Ainsi, 200×600 pixels à 60 Hz et une couleur de 22 bits seraient mathématiquement possibles en termes de *timing* et de RAM, bien que ces pixels ne seraient plus rectangulaires, mais étirés horizontalement. Pour des pixels rectangulaires, la résolution verticale doit être divisée par quatre, et chaque ligne d'image doit être sortie quatre fois. Le résultat est de 200×150 pixels en TrueColor. Une image de 800×600 pixels théoriques (**figure 12a**) est donc affichée de manière assez grossière, mais joliment colorée (**figure 12b**).
Mais il existe un compromis utilisable, car le contrôleur I²S peut également traiter des mots de 16 bits. Sur les 16 bits, 14 bits sont alors

Figure 12. 800×600 (a) devient 200×150 (b) et 400×360 pixels (c). (b) et (c) sont représentés à la même taille que (a) a pour une meilleure comparaison.

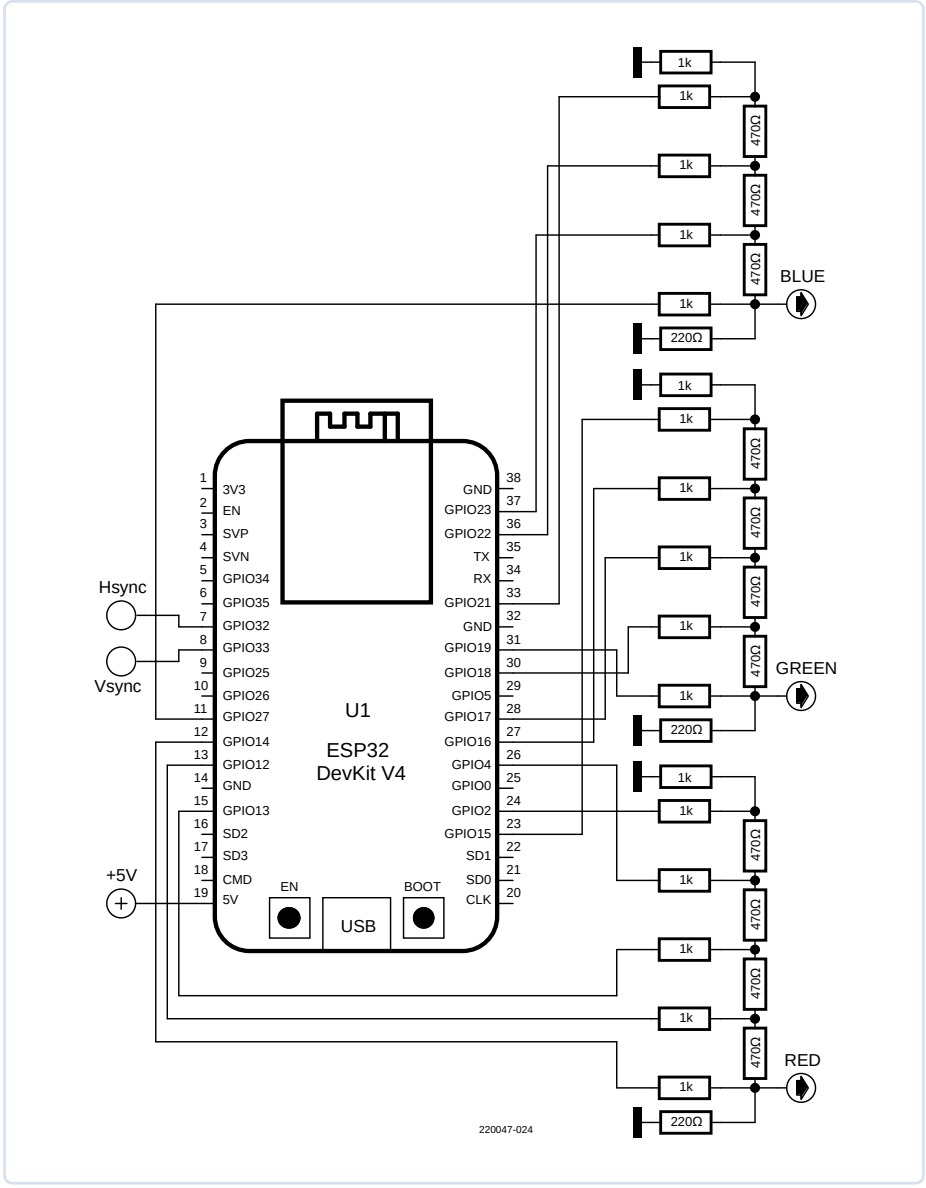


Figure 13. Génération du signal VGA via le CNA R2R sur ESP32.

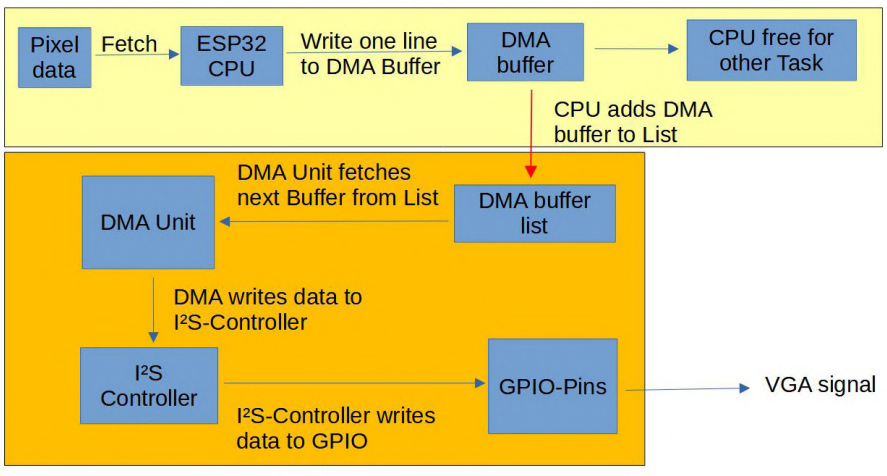


Figure 14. Sortie de pixels avec ESP32.

disponibles pour les informations de l'image et 2 bits pour les signaux de synchronisation, ce qui permet d'obtenir 16 384 couleurs affichables. En utilisant l'APLL de l'ESP32, on dispose d'une base de temps stable pour le *timing* VGA, permettant de réaliser des images en couleur 14 bits avec des résolutions de 320×240, 360×400 ou 480×400 pixels.

Pour devenir des signaux RVB, les informations de couleur sur 14 bits doivent être converties en analogique. Un CNA R2R tel que celui de la **figure 13** est une solution simple et efficace à cet effet. L'image de la **figure 12a** semble bien meilleure avec une résolution de 360×400 pixels (**figure 12c**).

Pour récupérer et afficher les données de la RAM, il y a quelques astuces : pour transférer les données vers le contrôleur I²S, le contrôleur DMA (Direct Memory Access) de l'ESP32 est utilisé pour délester la CPU, grâce à sa capacité de copier des données sans intervention de la CPU. La source, la destination, la quantité de données, le type de transfert et le signal de début de transfert sont suffisants à cet effet.

L'ESP32 prépare les données pour chaque ligne et les stocke dans un tampon de ligne qui contient les pixels visibles et les informations pour la synchronisation horizontale et verticale. Lorsqu'une ligne a été préparée, le contrôleur DMA copie les données vers le contrôleur I²S. Pendant ce temps, les cœurs de l'ESP32 peuvent s'occuper à d'autres tâches. La **figure 14** montre le déroulement du processus.

Cette procédure permet également de sortir des images sans les avoir au complet en mémoire. Pour cela, l'unité centrale de l'ESP32 doit toutefois être en mesure de calculer les pixels de la ligne suivante pendant le temps de sortie de la ligne en cours.

La bibliothèque *VGA* de bitluni pour le *framework* Arduino ESP32 est disponible sur GitHub [9]. Mais la bibliothèque *ESP32Lib* peut faire bien plus que de la sortie VGA. Elle contient également des fonctions pour les *sprites* [10] et un moteur de rendu 3D. De plus, la bibliothèque *FabGL* [11] ne gère pas seulement une sortie VGA avec l'ESP32, mais fournit les ingrédients d'un système complet d'images et de sons. Il existe même des tutoriels vidéo pour FabGL sur YouTube [12].

VGA sur le Raspberry Pi Pico

Grâce à son système sur une puce RP2040, le Raspberry Pi Pico est non seulement petit, mais aussi très polyvalent. Avec un processeur

à deux cœurs, 264 ko de RAM interne et une mémoire flash externe, cette carte à petit prix convient à de nombreux projets. La sortie du Raspberry Pi Pico fut accompagnée d'une carte de démonstration de sa capacité à produire des signaux VGA (**figure 15**), qui a depuis été mise à jour de manière à pouvoir aussi accueillir un Raspberry Pi Pico W. Les fichiers nécessaires pour la carte [13] sont disponibles sous la forme d'un projet KiCad, permettant à chacun de la réaliser.

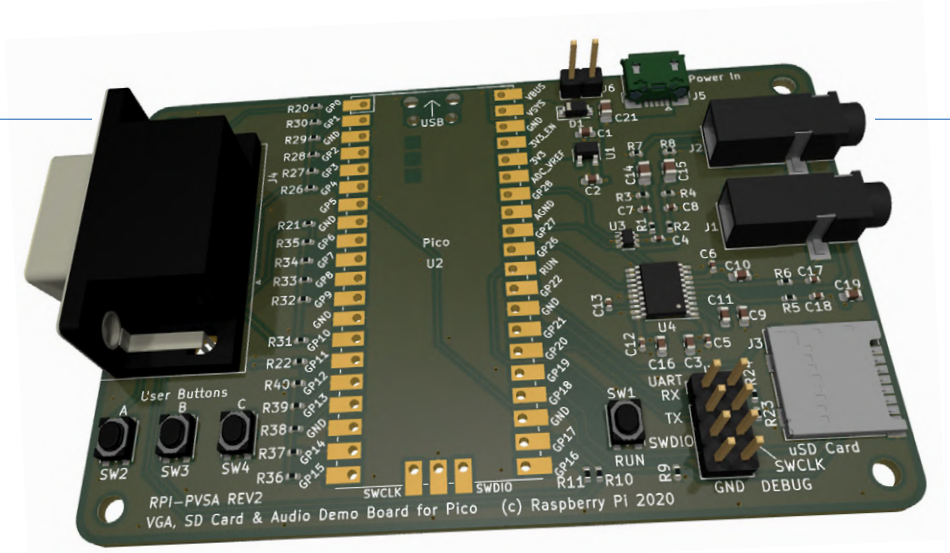


Figure 15. Carte de démonstration VGA, carte SD et audio pour Pico.

Le RP2040 offre suffisamment de mémoire pour stocker une image de 320×240 pixels à 16 bits (= 65 536 couleurs) entièrement en RAM. Les machines à états des E/S programmables (PIO) du RP2040 se prêtent idéalement à cette fin.

Pour le signal VGA, il faut des valeurs pour les niveaux RVB ainsi que pour la synchronisation horizontale et verticale. Pour la génération de ces signaux avec les PIO du RP2040, il suffit d'un logiciel. Malheureusement, le RP2040 ne possède pas de CNA, un circuit externe est donc nécessaire.

Un simple CNA R2R ou un CNA à pondération binaire convient à cet effet. Pour un CNA R2R, seules deux valeurs de résistance sont nécessaires, ce qui simplifie beaucoup les choses. Pour un CNA à pondération binaire, une valeur de résistance différente est nécessaire pour chaque bit. Le circuit de la carte de démonstration avec sortie VGA illustré à la **figure 16** est tiré d'un fichier PDF intitulé « *Hardware design with RP2040* » [14] par Raspberry Pi Ltd. Pour les couleurs spécifiées par RGB565, 5 broches GPIO chacune sont utilisées pour les couleurs rouge et bleue, tandis que le vert nécessite six broches pour les 65 536 couleurs résultantes. Les valeurs des résistances résultent de la structure du CNA, de la tension de sortie nécessaire et des résistances de terminaison du moniteur. Les valeurs des signaux RVB analogiques sont comprises entre 0 V et 0,7 V. Les entrées RVB d'un moniteur sont terminées par 75 Ω à la masse. Par conséquent, les diviseurs de tension doivent générer un maximum de 0,7 V pour chacune des trois couleurs en utilisant les 3,3 V des GPIO, ce qui se traduit par des rapports de résistance de 1:2:4:8:16. Pour obtenir 0,7 V à partir de 3,3 V, vous avez besoin d'un diviseur de tension avec un rapport de 3,7:1. Cinq ou six résistances par couleur sont connectées en parallèle. Les circuits parallèles doivent aboutir à 278 Ω pour arriver à un maximum de 0,7 V à la résistance

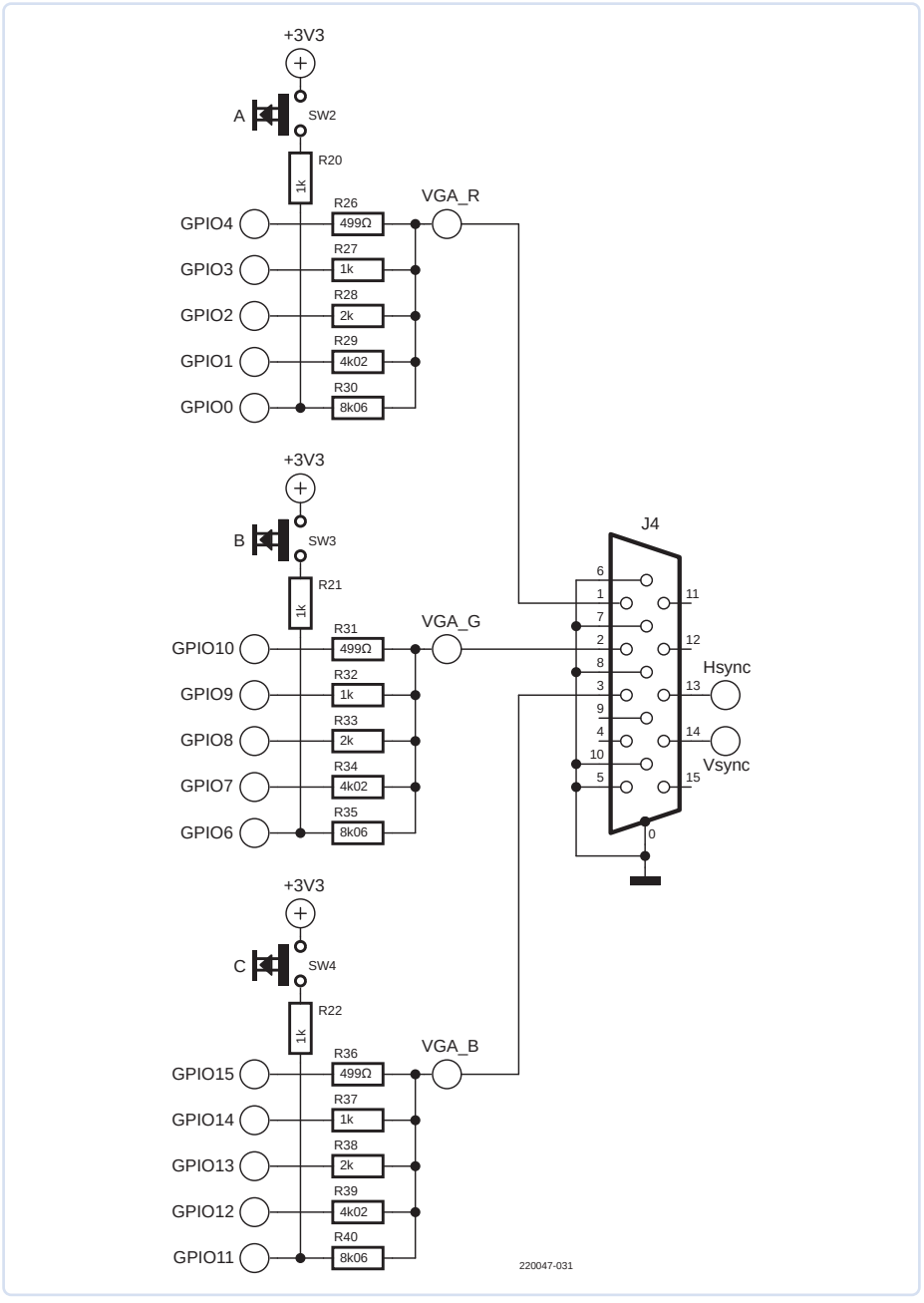


Figure 16. Une partie du circuit de la base VGA pour le Raspberry Pi Pico.

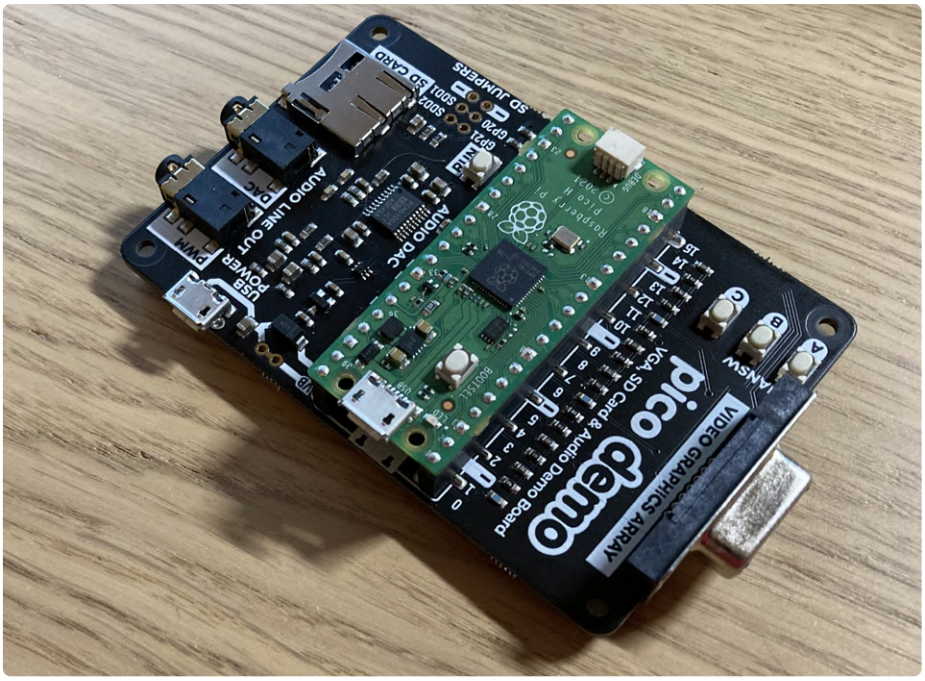


Figure 17. Base VGA Pimoroni pour le Raspberry Pi Pico.

de terminaison de 75 Ω avec une alimentation de 3,3 V.
Avec les résistances de la série E-96, des valeurs de 499, 1000, 2000, 4020 et 8060 Ω sont disponibles avec une tolérance de 1 %. Connectées en parallèle, on obtient 258 Ω, ce qui conduit à un 0,74 V suffisamment précis au niveau de la résistance de terminaison.
Si vous voulez une carte entièrement assemblée, vous pouvez acheter la carte de démonstration Pimoroni Pico VGA (**figure 17**) - voir aussi l'encadré des produits connexes. En plus de la sortie VGA, la carte comporte un codec audio et un emplacement pour carte SD.

Sortie de pixels avec le RP2040

Pour pouvoir émettre un signal VGA, il faut produire et stocker les données de l'image. La RAM du RP2040 suffit pour stocker une image de 320×240 pixels en couleur 16 bits. Le dépôt GitHub *pico-playground* [16] contient des démos adéquates.
Pour une sortie simple, les données de l'image sont stockées sous forme de valeurs RGB565 dans 150 Ko de mémoire réservée. L'image est lue de la mémoire ligne par ligne, puis le *timing* approprié pour la sortie de l'image est généré pour chaque ligne. La **figure 18** montre une séquence simplifiée.
Ce type de sortie d'image à partir de la RAM permet aux deux cœurs de rester libres pour d'autres tâches. Il suffit de fournir une nouvelle ligne d'image avec un *timing* adéquat. La bibliothèque *pico_scanvideo* qui le gère se trouve sur [17]. Elle permet la préparation de plus d'une ligne d'image à l'avance. Le nombre de lignes de l'image et son adaptation à la résolution de l'écran sont configurables. Ainsi, par exemple, vous pouvez traiter 320×200 pixels avec une résolution de 640×480 pixels. La mise en mémoire tampon des lignes d'image permet également de réaliser d'autres astuces, comme la génération d'images « à la volée ». Il s'agit plus ou moins d'une course contre la montre ou contre le faisceau d'électrons.

Le Raspberry Pi Pico en tant qu'artiste dessinateur

Le Raspberry Pi Pico et les autres cartes basées sur le RP2040 peuvent être de véritables artistes du dessin. Dans la **figure 19**, plusieurs *sprites* se déplacent sur l'écran. La sortie est de 640×480 pixels en couleur 16 bits. C'est plus que ce que peut contenir la RAM du Raspberry Pi Pico. Pour cette raison, tous les pixels sont redessinés pour chaque nouvelle image à sortir.
Alors que cette démo des fonctions de base permet de dessiner des images 2D sur l'écran, la bibliothèque *PicoVGA* de Miroslav Nemecek va un peu plus loin. La sortie graphique est limitée à 8 bits, ce qui laisse plus de broches d'E/S libres et nécessite moins de RAM pour une image complète (75 au lieu de 150 ko). Le dépôt GitHub de la bibliothèque contient plusieurs démos et petits jeux [18]. La figure 20 montre le jeu *Ants* fonctionnant sur

Figure 19. Sprites animés.

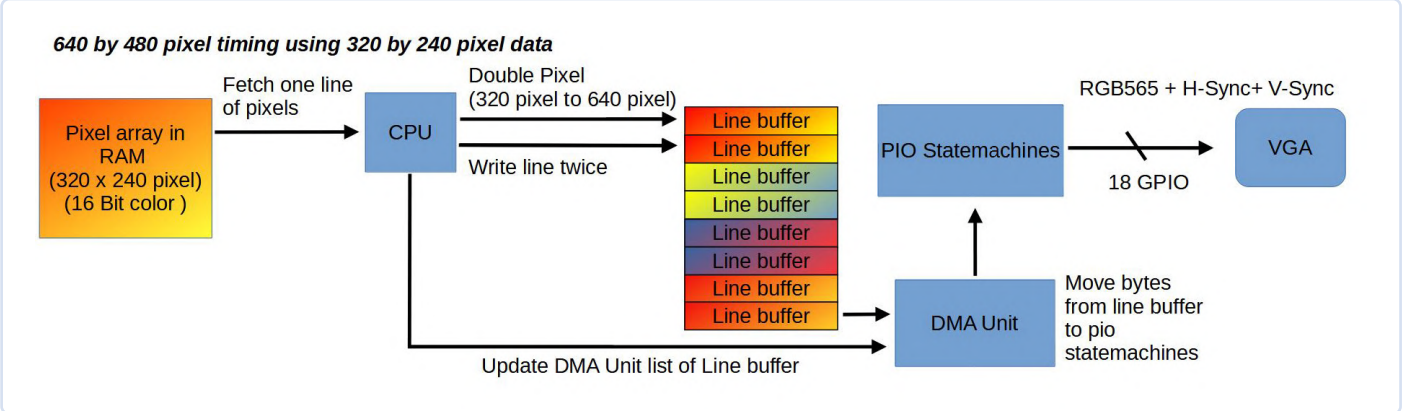


Figure 18. Séquence de sortie VGA avec Raspberry Pi RP2040.

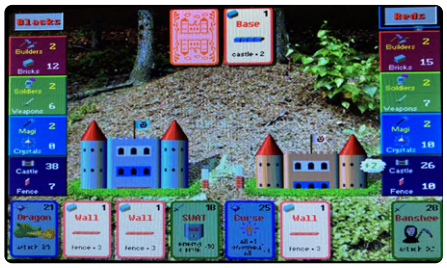


*Figure 20. **Ants** sur un Raspberry Pi Pico.*

*Figure 21. **Pac-Man** sur un Raspberry Pi Pico.*

Figure 22. Pseudo-3D avec un Raspberry Pi Pico.

le Raspberry Pi Pico et la **figure 21** le classique Pac-Man. Une brève description de la bibliothèque se trouve sur la page allemande de MagPi [19].

Le Raspberry Pi Pico contient des unités fonctionnelles qui permettent même des effets pseudo-3D (**figure 22**). Ce type d'affichage a été rendu célèbre par le SNES (Super Nintendo Entertainment System) avec des jeux tels que *Super Mario Kart* (**figure 23**) ou *F-Zero* en mode 7 [20]. La bibliothèque correspondante est disponible sur GitHub [21].

DVI

Introduit en 1999, le DVI est une interface numérique [22] d'envoi d'images à un moniteur sans perte de qualité. La **figure 24** montre une prise blanche typique pour un DVI-D à deux canaux. Les spécifications exactes du DVI se trouvent dans un fichier PDF sur [23].

Lorsque les premiers écrans à cristaux liquides (LCD) sont devenus abordables, de nombreuses cartes graphiques utilisaient encore l'interface VGA. Mais en Full HD (1920×1080 pixels), l'image n'était pas vraiment nette en raison du passage en analogique. Avec le DVI, se répandit une interface numérique qui pouvait alimenter directement un moniteur LCD, sans pertes de conversion, avec des images haute résolution en couleur 24 bits.

8b/10b, TMDS et horloge 1:10

Alors qu'avec le VGA, trois des cinq signaux étaient analogiques, avec le DVI, tout est numérique. Quatre paires de lignes différentielles (±Data 0, ±Data 1, ±Data 2 et ±Clock) représentent le minimum pour la transmission d'images à un seul canal avec DVI.

Les données de l'image RVB sont transmises via Data 0, Data 1 et Data 2. Pour une image en couleur 24 bits, chaque couleur est transmise en utilisant 8 bits. En outre, il existe un signal d'horloge qui permet de combiner les couleurs pour reconstituer les pixels du côté du récepteur.

La transmission numérique utilise la méthode TMDS (**T**ransition-**M**inimized **D**ifferential **S**ignaling) [24], qui assure un transport robuste du signal et un faible rayonnement électromagnétique grâce à son codage des données.

La méthode de codage utilisée est 8b/10b [25], où seulement 460 des 1024 combinaisons possibles sont utilisées pour coder l'information 8 bits pour les trois couleurs, rouge, vert et bleu. Plusieurs valeurs de 8 bits peuvent donc être représentées par deux combinaisons. Quatre combinaisons sont utilisées pour la signalisation de C0 et C1 pour la synchronisation horizontale et verticale. Lors de la sortie des pixels d'une ligne, le nombre de zéros et de uns dans le flux de données est équilibré pour obtenir des flux de données sans composante continue, ce qui est rendu possible par la double représentation de certaines valeurs 8 bits.

Même si les informations de l'image sont transmises de manière entièrement numérique, les signaux de synchronisation horizontale et verticale sont toujours présents. Toutefois, il ne s'agit plus de lignes de données dédiées, mais de signaux codés dans le flux de données 0 à l'aide de C0 et C1. Le flux de données DVI est conçu pour être facilement converti en un signal VGA analogique.

Le signal d'horloge en DVI ne correspond pas au débit binaire pour le rouge, le vert et le bleu, mais a un rapport de 10:1 et correspond à l'horloge de pixels en raison du codage 8b/10b. Pour la plus basse résolution DVI de 640×480 pixels à 60 Hz, une horloge de pixels de 25,175 MHz ou un débit binaire de 251,75 MHz est nécessaire pour les informations de couleur, tout comme avec le VGA. Ainsi, ses origines VGA sont encore bien visibles dans le DVI.

DVI avec le Raspberry Pi Pico

En supposant la résolution la plus basse possible spécifiée précédemment, un Raspberry Pi Pico doit émettre trois flux de données à 251,75 Mbps. Cela n'est tout simplement pas possible avec son horloge maximale de 133 MHz, car près de 2 bits devraient être émis par tic d'horloge pour chacun des trois flux de données couleur.

*Figure 23. **Super Mario Kart** avec effet pseudo-3D.*

Figure 24. Prise DVI-D sur un PC (source : Dr. Thomas Scherer).

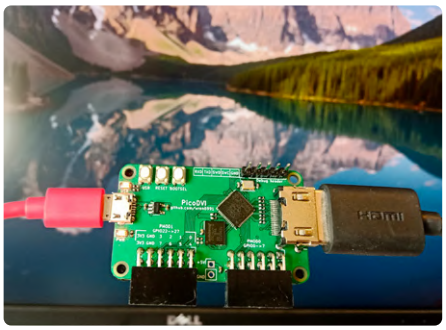


Figure 25. PicoDVI - RP2040 avec sortie vidéo DVI (source : Wren6991 / https://elektor.link/picodviimg).

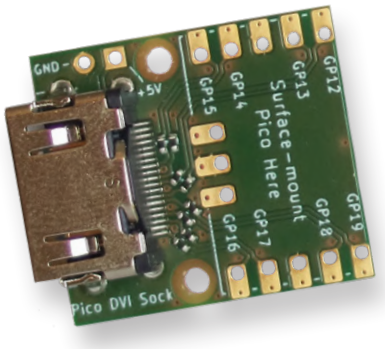


Figure 26. Pico DVI Sock pour Raspberry Pi Pico.

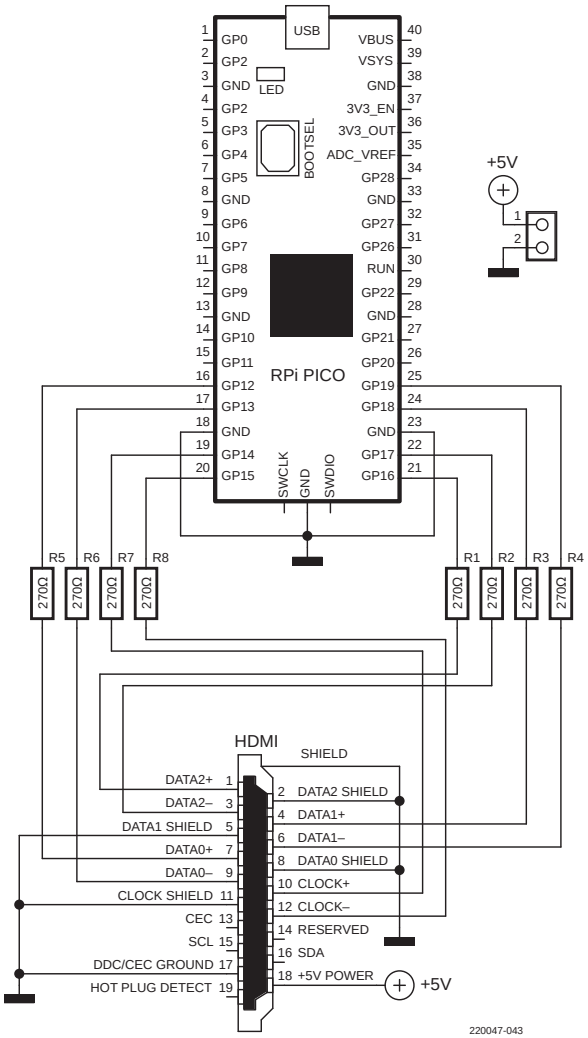


Figure 27. Schéma du Pico DVI Sock.

Ainsi, le Raspberry Pi Pico ne franchirait déjà pas cet obstacle. Cependant, le RP2040 est extrêmement surcadençable. Une horloge de 251,75 MHz pour les deux cœurs est possible sans dysfonctionnements notables à température ambiante. Cependant, les broches GPIO du RP2040 doivent également suivre cette énorme vitesse. En outre, les données d'image doivent encore être codées sous forme de flux TMDS et donc préparées en conséquence. Luke Wren a pu montrer [26] qu'un RP2040 en est capable (**figure 25**) ; voir son interview sur [27] pour plus d'informations.

L'un des deux cœurs du RP2040 est occupé à environ 60 % par l'encodage TMDS. De plus, trois des huit machines d'état PIO sont utilisées pour sortir les données avec les 251,75 MHz nécessaires. Certaines fonctions du contrôleur DMA déchargent les cœurs du transfert des données. De cette façon, un cœur reste entièrement libre et disponible pour votre propre logiciel.

Une prise DVI ou HDMI (**figure 26**) et huit résistances suffisent pour connecter électriquement le Raspberry Pi Pico à l'entrée DVI ou HDMI d'un moniteur. Le câblage nécessaire (**figure 27**) est assez simple.

Dans le dépôt GitHub de PicoDVI [26], Luke Wren décrit comment l'interpolateur du RP2040 peut être utilisé pour accélérer l'encodage TMDS, ce qui permet de gagner du temps de calcul. L'interpolateur était initialement destiné à générer des graphiques pseudo-3D, comme dans *Pilotwings* (**figure 28**). Cependant, il est suffisamment souple pour être utilisé pour soulager les cœurs du processeur lors de la génération des trois flux de données TMDS à partir des données RVB.

Alors, comment les pixels passent-ils de la mémoire vive au moniteur ? Pour une sortie avec une résolution de 640×480 pixels, une image de 320×240 pixels est mise à l'échelle. Par conséquent, il n'y a que 240 lignes à coder par TMDS 60 fois par seconde. Chaque ligne est émise deux fois, et les pixels sont donc deux fois plus hauts.

Pour ce faire, les données brutes sont fournies ligne par ligne, comme pour la sortie d'un signal VGA. Les lignes peuvent provenir d'un tampon de trame dans le Raspberry Pi Pico ou être produites « à la volée ». Comme pour le VGA, des projets intéressants sont également possibles avec la sortie DVI. Ici, un seul des deux cœurs du processeur, avec son interpolateur, est occupé à générer le signal DVI.

Figure 28. Pseudo-3D dans le jeu Pilotwings.

*Figure 29. Démonstration de **Walker** pour le Raspberry Pi Pico.*

*Figure 30. Les tuiles dans la démo de **Walker**.*

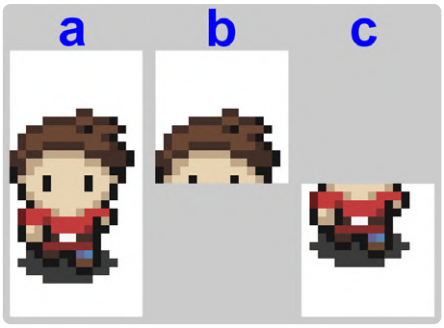


*Figure 31. Personnage de jeu avec fond transparent (a) et ses **sprites** supérieur (b) et inférieur (c).*

La démonstration du *Walker* (**figure 29**) montre de manière impressionnante qu'un Raspberry Pi Pico a encore assez de réserves pour faire de la magie à l'écran.

Sprites, tuiles et cartes de tuiles avec le RP2040

Pour les sorties VGA et DVI, il existe une bibliothèque graphique qui prend en charge les *sprites*. Dans la démo du *Walker*, l'image finie était composée de nombreux petits éléments appelés tuiles (**figure 30**). Chaque tuile a une taille de 16×16 pixels. Ces tuiles sont destinées à servir d'arrière-plan graphique et ne comportent généralement aucune information de transparence, contrairement au personnage du joueur (**figure 31a**), qui est composé de deux *sprites* (**figure 31b** et **figure 31c**). L'arrière-plan et les *sprites* peuvent être placés individuellement. Avec la sélection appropriée, de sympathiques arrière-plans peuvent être assemblés comme dans la **figure 32**.

Les cartes de tuiles constituent une bibliothèque (**figure 33**), dans laquelle des tuiles de 16×16 pixels sont puisées puis assemblées par le Raspberry Pi Pico pour former une image. Mais pourquoi fait-on ainsi ? Parce que l'avantage des *sprites*, des tuiles et des cartes de tuiles est leur faible consommation de mémoire et leur souplesse d'utilisation.
En plus de la carte des tuiles, la seule chose qui doit être stockée en mémoire est la position à laquelle les tuiles ou les *sprites* doivent être dessinés. Les animations ou les défilements sont également possibles de cette manière grâce à des mises à jour relativement simples des positions des *sprites*.
Pour le code source des démos, veuillez vous référer au dépôt GitHub correspondant [28] de Luke Wren. Les images présentées ici proviennent des applications *tiles_and_sprites*, *tiles_parallax* et *tiles*. Si vous disposez d'un Raspberry Pi Pico adéquat avec une sortie DVI, vous pouvez essayer les applications vous-même et manipuler le code source.

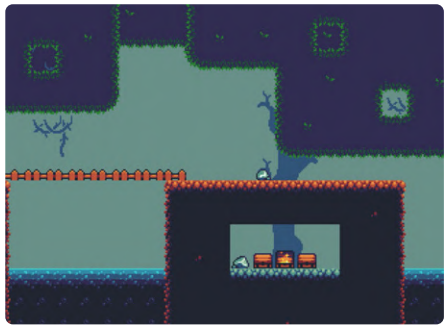


Figure 32. Démonstration de la plateforme 2D.

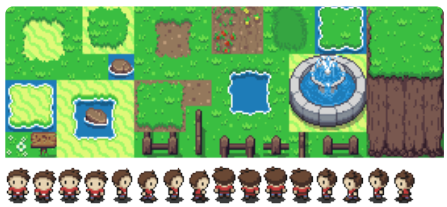


Figure 33. Cartes de tuiles (source : ArMM1998 / https://elektor.link/ZeldaTiles).

Un microcontrôleur comme artiste dessinateur ? Jusqu'à présent, nous n'avons vu que des démos – pas d'applications ou de jeux complets. Mais les démos en disent déjà assez pour que vous puissiez ajouter des images à vos projets. L'ESP32 et le Raspberry Pi Pico sont des modules peu coûteux qui permettent d'obtenir des sorties graphiques de qualité étonnante. Que ce soit *Pac-Man* ou *Tetris* via DVI, VGA ou vidéo composite, avec chaque génération, les capacités et les performances brutes des microprocesseurs augmentent. Des grands effets viennent à leur portée, comme le rendu 3D ou les images animées en couleur avec des *sprites*. Tout comme à l'époque des ordinateurs grand public des années 80, avec beaucoup de créativité et quelques astuces, il est possible de tirer des images intéressantes de ce matériel limité.

Les microcontrôleurs ont juste besoin d'une chorégraphie appropriée pour dessiner, comme tous les ingrédients décrits dans cet article. C'est à vous maintenant d'en faire usage pour créer des projets fascinants. Publiez vos créations sur Elektor Labs [29], et vous pourrez ainsi discuter de caractéristiques et de problèmes techniques avec d'autres membres d'Elektor et échanger des informations utiles.

220614-04 — VF : Helmut Müller

Des questions, des commentaires ?

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

Produits

- **Carte de démonstration Pimoroni Pico VGA (SKU 19769)**
  https://elektor.fr/19769
- **Raspberry Pi Pico (SKU 19562)**
  https://elektor.fr/19562
- **Prise DVI pour Raspberry Pi Pico (SKU 19925)**
  https://elektor.fr/19925
- **ESP32-PICO-Kit V4 (SKU 18423)**
  https://elektor.fr/18423

LIENS

[1] Bibliothèque vgax : https://github.com/smaffer/vgax
[2] Bibliothèque vgaxua : https://github.com/smaffer/vgaxua
[3] VGA-PacMan [allemand] : https://niklas-rother.de/artikel/vga-pacman
[4] Générateur vidéo Atmel ATmega avec SDRAM : http://tinyvga.com/avr-sdram-vga
[5] Tables de synchronisation VGA : http://tinyvga.com/vga-timing/640x480@60Hz
[6] ESP32 VGA : https://bitluni.net/esp32-vga
[7] Manuel de référence technique ESP32 [PDF] : https://elektor.link/ESP32TechMan
[8] Synchronisation du signal VGA : http://tinyvga.com/vga-timing
[9] ESP32Lib de bitluni : https://github.com/bitluni/ESP32Lib
[10] Sprite (infographie) - Wikipedia : https://en.wikipedia.org/wiki/Sprite_(computer_graphics)
[11] Bibliothèque FabGL : http://fabglib.org
[12] Tutoriels vidéo sur FabGL : https://elektor.link/YTFabGLTuts
[13] RPI-PVSA VGA, SD Card & Audio Demo Board for Pico - Projet KiCad [Zip] : https://elektor.link/RP2040VGAKiCad
[14] Conception de matériel avec RP2040 [PDF] : https://elektor.link/RP2040HardwareDesign
[15] Pimoroni Raspberry Pi Pico VGA Demo Base : https://elektor.fr/pimoroni-raspberry-pi-pico-vga-demo-base
[16] pico-playground : https://github.com/raspberrypi/pico-playground
[17] Bibliothèque pico_scanvideo : https://elektor.link/pico_scanvideoLib
[18] Bibliothèque PicoVGA : https://github.com/Panda381/PicoVGA
[19] MagPi : Bibliothèque graphique VGA pour le Raspberry Pi Pico [allemand] : https://elektor.link/MPPicoVGALib
[20] Mode 7 - Wikipedia : https://en.wikipedia.org/wiki/Mode_7
[21] Démonstration de l'interpolateur - GitHub : https://elektor.link/GHInterperlatorDemo
[22] DVI – Wikipedia : https://elektor.link/WPDVI
[23] Spécification DVI [PDF] : https://elektor.link/DVISpec
[24] Transmission de données à haute vitesse (TMDS) - Wikipedia : https://elektor.link/WPTMDS
[25] Codage 8b/10b - Wikipédia : https://en.wikipedia.org/wiki/8b/10b_encoding
[26] PicoDVD de Luke Wren : https://github.com/Wren6991/PicoDVI
[27] Mathias Claussen interviews Luke Wren, "DVI with the RP2040," Elektor 3-4/2023: https://elektormagazine.com/220575-01
[28] Démonstrations de PicoDVI, incluant tuiles et sprites « walker » - GitHub : https://elektor.link/GHPicoDVIDemos
[29] Elektor Labs : https://elektormagazine.fr/labs

le système d'exploitation temps réel Metronom

un RTOS pour processeurs AVR

Dieter Profos, Dr. sc. techn. ETH (Suisse)

Pour de nombreuses tâches - comme le traitement de signaux continus - les microcontrôleurs doivent effectuer des opérations dans des intervalles de temps précis. Le système d'exploitation en temps réel présenté ici est (également) adapté aux contrôleurs AVR disposant d'un espace mémoire limité. Vous devez accepter certaines limitations, dont notamment la programmation en assembleur, qui constitue néanmoins un bon compromis pour les projets où la vitesse et la capacité en temps réel sont importantes.

Mais pourquoi un autre système d'exploitation ?

Avec l'apparition de petits et micro processeurs et contrôleurs, certains processus, pour lesquels l'utilisation d'un *vrai* ordinateur n'aurait jamais été nécessaire auparavant, sont devenus automatisables. Ces microcontrôleurs n'ont pas besoin de contrôler les périphériques (clavier, souris, écran, disque, etc.), de sorte que les systèmes d'exploitation peuvent être limités à l'essentiel : organiser le traitement des programmes utilisateur.

La plupart des systèmes d'exploitation sont conçus pour exécuter le plus de programmes possible de la manière la plus efficace possible et (côté utilisateur) simultanément. Les choses sont toutefois différentes lorsqu'il s'agit de traiter des signaux continus et limités dans le temps : Cela nécessite des fonctions qui s'exécutent pendant des intervalles précis. Par exemple, la fonction `delay()` d'Arduino n'est plus suffisamment précise dans ce cas, car elle ne génère que des temps d'attente, mais ne prend pas en compte les temps d'exécution nécessaires au traitement, qui sont explicites avec des temps d'échantillonnage de 1 ms ou même plus courts.

Les deux problèmes suivants doivent donc être résolus :

- Certaines tâches doivent être exécutées exactement à des moments prédéfinis, d'autres seulement lorsqu'il leur reste du temps.
- Chaque tâche interruptible nécessite sa propre pile pour mettre en mémoire tampon le contenu des registres. Cependant, avec les microcontrôleurs, l'espace mémoire est assez limité : par exemple, 750 octets pour l'ATtiny25 ou 1 K pour l'ATmega8.

Le système d'exploitation Metronom présenté ici est disponible en téléchargement sur le site web d'Elektor [1] en tant que logiciel libre sous la licence BSD-2 ; une version via GitHub est également prévue dans les mois à venir.

Tâches cycliques

Metronom est conçu précisément pour l'exécution de ce que l'on appelle des tâches cycliques pendant des intervalles de temps prédéterminés (jusqu'à 8 durées de cycle différentes). Une seule tâche par cycle ; si différentes opérations au contenu indépendant doivent être exécutées dans le même cycle, elles doivent être combinées dans la même tâche.

Les temps de cycle sont générés comme suit :

- La période du cycle de base (par ex. 1 ms) est établie avec

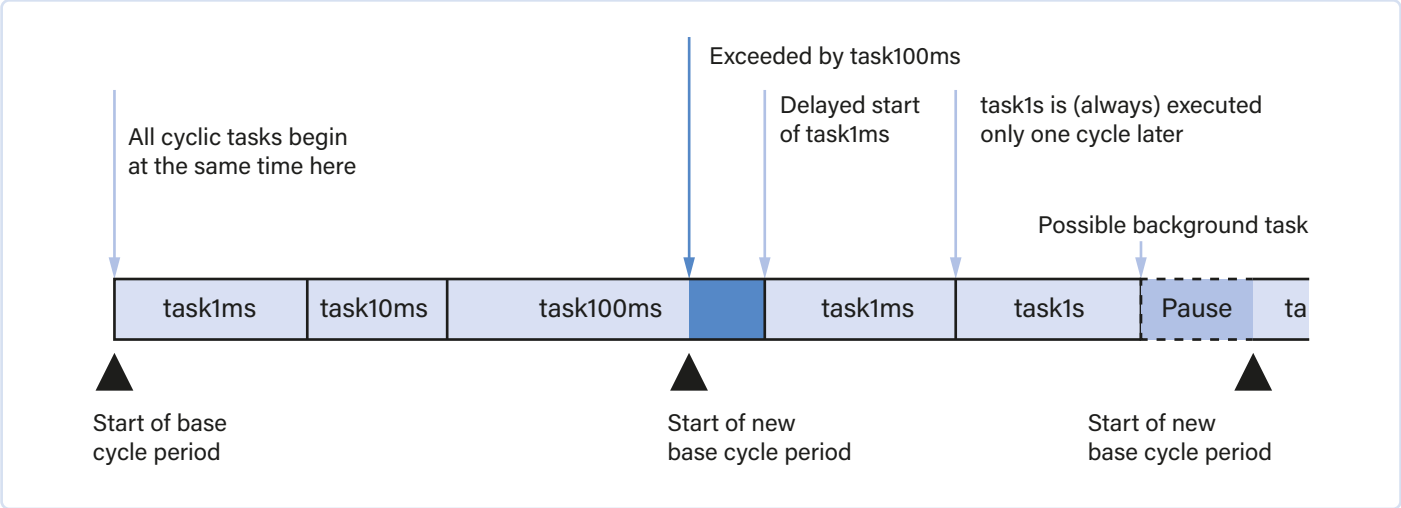


Figure 1. Séquence de plusieurs tâches cycliques (cas limite).

le matériel du processeur (quartz ou oscillateur RC interne, compteur logiciel contrôlé par le matériel et les interruptions), et
- les autres durées de cycle sont générées par une succession de compteurs, de sorte que chaque durée de cycle est un multiple de la précédente (par exemple, si le paramètre par défaut est 1 ms → 10 ms → 100 ms → 1 s).

Le système d'exploitation présenté ici a une caractéristique importante : les tâches cycliques ne peuvent pas s'interrompre (non préemptives). D'une part, cela garantit que le *timing* de ces tâches est aussi précis que possible, et d'autre part, le processeur ne gaspille pas du *temps improductif* à changer de tâche. Et comme chaque tâche cyclique – une fois lancée – s'exécute complètement sans interruption (sauf par des interruptions) avant que la tâche cyclique suivante ne soit lancée, toutes les tâches cycliques peuvent utiliser la même pile.
Mais que se passe-t-il si l'exécution d'une tâche dure plus que la période du cycle de base (ce qui doit être évité par le développeur) ou si plusieurs tâches cycliques ont été lancées dans le même cycle, où la somme des temps d'exécution dépasse la durée du cycle de base (ce qui est tout à fait justifié) ? Ici une autre caractéristique de Metronom intervient : les tâches cycliques ont différentes priorités : la tâche avec le temps de cycle le plus court a la plus haute priorité, la *deuxième* tâche la plus rapide a la deuxième priorité, et ainsi de suite. Si toutes les tâches cycliques lancées en même temps ne peuvent pas être achevées dans le temps de cycle de base, la tâche en cours d'exécution est poursuivie jusqu'à sa fin après l'écoulement de la durée du cycle de base – mais alors la tâche *la plus rapide* la plus prioritaire est exécutée en premier, et ce n'est qu'ensuite que les autres tâches cycliques lancées précédemment sont exécutées à nouveau.
Exemple : l'implémentation des temps de cycle entraîne le lancement simultané de toutes les tâches cycliques toutes les secondes. Ce qui se passe dans ce cas est illustré à la **figure 1**.
Cela signifie que chaque tâche cyclique ne doit pas dépasser le temps de cycle le plus court – c'est-à-dire 1 ms comme limite supérieure absolue dans notre exemple. Cela correspond à environ 6 000 instructions pour un ATtiny fonctionnant à 8 MHz et à environ 14 000 instructions pour un ATmega à 16 MHz (le reste des instructions est utilisé – en moyenne – par le système d'exploitation lui-même et pour la gestion des interruptions).

Tâches d'arrière-plan

Toutefois, certaines opérations prennent plus de temps à cause de leur nature :
- L'accès en écriture à l'EEPROM, par exemple, prend quelques millisecondes (typiquement environ 3,3 ms), c'est-à-dire un temps excessivement long pour un cycle de base de 1 ms.
- Transmettre du texte à 9 600 Bd est impossible dans un cycle de base de 1 ms car même la transmission d'un seul caractère prend déjà plus de 1 ms.
- Lorsque des calculs plus longs (par exemple, des opérations arithmétiques émulées) sont nécessaires ou que des chaînes de caractères doivent être traitées, cela prend souvent trop de temps au cours d'une tâche cyclique et bloque donc les processus soumis à des contraintes de temps.

Cela signifie qu'il faut encore trouver un moyen de réserver ces processus aux tâches interruptibles. Deux procédés combinés sont utilisés à cet effet :
- Utilisation d'interruptions au lieu de l'attente active : cela permet *d'affecter* l'attente de la fin d'une opération (par exemple, la transmission d'un caractère) au matériel ; cette méthode est utilisée pour les opérations contrôlées par des interruptions. Cela résout les problèmes de transmission d'un seul caractère ou d'écriture d'une seule valeur dans l'EEPROM, mais pas l'attente de la fin de l'opération globale (par exemple, la transmission d'un texte entier).
- Mise en œuvre des tâches d'arrière-plan : une tâche d'arrière-plan s'exécute uniquement pendant les phases inoccupées par les tâches cycliques. En outre, elle peut être interrompue à tout moment, de sorte qu'elle n'interfère pas avec l'exécution ponctuelle des tâches cycliques.

Cependant, une fois qu'une tâche d'arrière-plan est en cours d'exécution, elle ne peut pas être interrompue par d'autres tâches similaires. Ainsi, une seule tâche d'arrière-plan est traitée à la fois, et si elle est suspendue, le traitement des tâches d'arrière-plan l'est également. Bien que cela ralentisse le traitement des tâches d'arrière-plan, cela signifie qu'un seul emplacement de pile doit être réservé pour toutes les tâches d'arrière-plan.

Les tâches d'arrière-plan se caractérisent par :
- Une tâche d'arrière-plan peut être interrompue à tout moment en faveur de tâches cycliques, mais pas en faveur d'une autre tâche similaire.
- L'exécution d'une tâche d'arrière-plan est déclenchée par un appel au répartiteur.

- Les tâches d'arrière-plan sont exécutées l'une après l'autre dans l'ordre où elles ont été lancées.
- Les tâches d'arrière-plan peuvent attendre des événements (`WAIT_EVENT`), qui sont déclenchés, par exemple, par des processus contrôlés par des interruptions.
- Les tâches d'arrière-plan peuvent également être mises en attente pendant des durées prédéfinies (`DELAY`).
- Chaque tâche d'arrière-plan peut recevoir un message de départ de 3 mots de 16 bits, qui peut être utilisé pour spécifier son objectif (un 4e mot est réservé pour l'adresse de départ de la tâche).
- Un nombre quelconque de routines de tâches d'arrière-plan existe dans le programme utilisateur ; toutefois, un maximum de 8 peuvent être lancées simultanément.

La coordination des tâches entre elles (à *quel moment l'exécution de telle ou telle tâche est-elle autorisée* ?) est gérée par ce que l'on appelle le répartiteur. Il exécute tous les « processus administratifs », tels que le démarrage des tâches, la sauvegarde/restauration des registres du processeur, ou la désactivation/activation des interruptions.

Exceptions

Comme les microcontrôleurs ne disposent généralement pas de périphériques orientés texte, le débogage est très compliqué, notamment pour les fonctions dépendant du temps, car les points d'arrêt ou autres perturbent complètement le comportement temporel. Par conséquent, le noyau du système d'exploitation fournit un mécanisme simplifié pour le traitement des exceptions, qui est constitué de deux étapes :

- Une zone `try-catch` globale détecte toutes les exceptions (exceptions/erreurs) survenant dans le noyau et dans les émulations arithmétiques. Les données spécifiques à l'exception peuvent être stockées dans l'EEPROM et/ou sorties via USART ; ensuite, le système d'exploitation effectue un *RESET* total du système (y compris le `user-reset`). Cette zone d'exception est toujours active.
- En outre, il est possible d'utiliser une zone `try-catch` orientée application, qui ne traite que le programme utilisateur en cours. Le traitement de ces exceptions se fait initialement ainsi : les données d'exception sont stockées et/ou sorties via USART ; ensuite, une routine de *redémarrage de l'application* à spécifier par l'utilisateur est exécutée (sous-routine `user_restart`).

Gestion des interruptions

Les interruptions sont traitées de quatre manières différentes :

- L'interruption de réinitialisation est utilisée par le système d'exploitation et n'est pas directement accessible par l'utilisateur. Cependant, comme l'utilisateur a également besoin de cette interruption pour initialiser ses processus, le système d'exploitation appelle la sous-routine `user_init` après sa propre initialisation, qu'il peut utiliser avec son code d'initialisation spécifique à l'application.
- Timer/counter0 est utilisé pour la génération de l'horloge de base pour tous les processus cycliques ; il n'est donc pas accessible par l'utilisateur.
- Pour l'utilisation de l'EEPROM et de l'USART, le système d'exploitation propose des blocs pilotes prêts à l'emploi, qui peuvent être intégrés lors de la génération du système d'exploitation (voir ci-dessous). Cependant, l'utilisateur peut, de plus, associer ses propres routines de service à ces interruptions ou simplement les laisser ouvertes lorsqu'elles ne sont pas utilisées.
- Toutes les autres interruptions sont directement à disposition de l'utilisateur. Pour chaque interruption, une routine de service d'interruption ainsi qu'une routine d'initialisation d'interruption doivent être spécifiées ; si plus d'une interruption appartient à un périphérique (par exemple, des timers ou USART), une routine d'initialisation partagée est suffisante. Pour cela, l'utilisateur active les paramètres correspondants dans le fichier de génération et insère le contenu des routines d'initialisation et de service correspondantes dans son programme utilisateur.
- Les interruptions non utilisées sont automatiquement « interceptées » par le système d'exploitation.

Environnement de programmation

Pour des raisons d'efficacité, Metronom est écrit en assembleur AVR (Atmel/Microchip) et implique donc que les programmes utilisateurs soient également écrits en assembleur ; aucune interface pour le langage C n'a été implémentée. Cependant, il existe une bibliothèque avec de nombreux sous-programmes, pour l'arithmétique 8 bits ainsi que l'arithmétique 16 bits (4 opérations arithmétiques de base) entre autres ; une bibliothèque de fractionnement 16 bits est en cours de préparation. Pour faciliter la programmation, tous les appels du système d'exploitation sont disponibles sous forme de macros. Pour éviter les conflits de nommage, la convention d'affectations de noms suivante s'applique : toutes les variables et les cibles de saut dans le système d'exploitation et les bibliothèques commencent par un trait de soulignement (« _ »). Par conséquent, tous les noms dans le programme utilisateur doivent uniquement commencer par des lettres. Les caractères autres que les lettres, les chiffres et le trait de soulignement ne sont pas autorisés. La structure globale du Metronom et le programme utilisateur correspondant sont illustrés dans la **figure 2**.

Appels du système d'exploitation

Pour la liste complète des appels du système d'exploitation, veuillez vous reporter aux références en fin d'article ; nous n'en donnons ici qu'un aperçu :

Macros pour le traitement des exceptions

- `KKTHROW` lance une exception à l'échelle du système, c'est-à-dire qu'après avoir sauvegardé/sorti les informations relatives à l'exception, le système entier est redémarré.
- `KTHROW` lance une exception limitée au programme utilisateur, c'est-à-dire qu'après la sauvegarde/la sortie des informations relatives à l'exception, seule la sous-routine utilisateur `user_restart` est exécutée ; ensuite, les tâches cycliques sont relancées.

Macros pour l'utilisation des tâches d'arrière-plan

- `_KSTART_BTASK` lance une tâche d'arrière-plan.
- `_KDELAY` met en veille la tâche d'arrière-plan appelante pendant n (0 à 65 535) ms.
- `_KWAIT` met en veille la tâche d'arrière-plan appelante, qui peut être reprise avec ...
- `_KCONTINUE`.

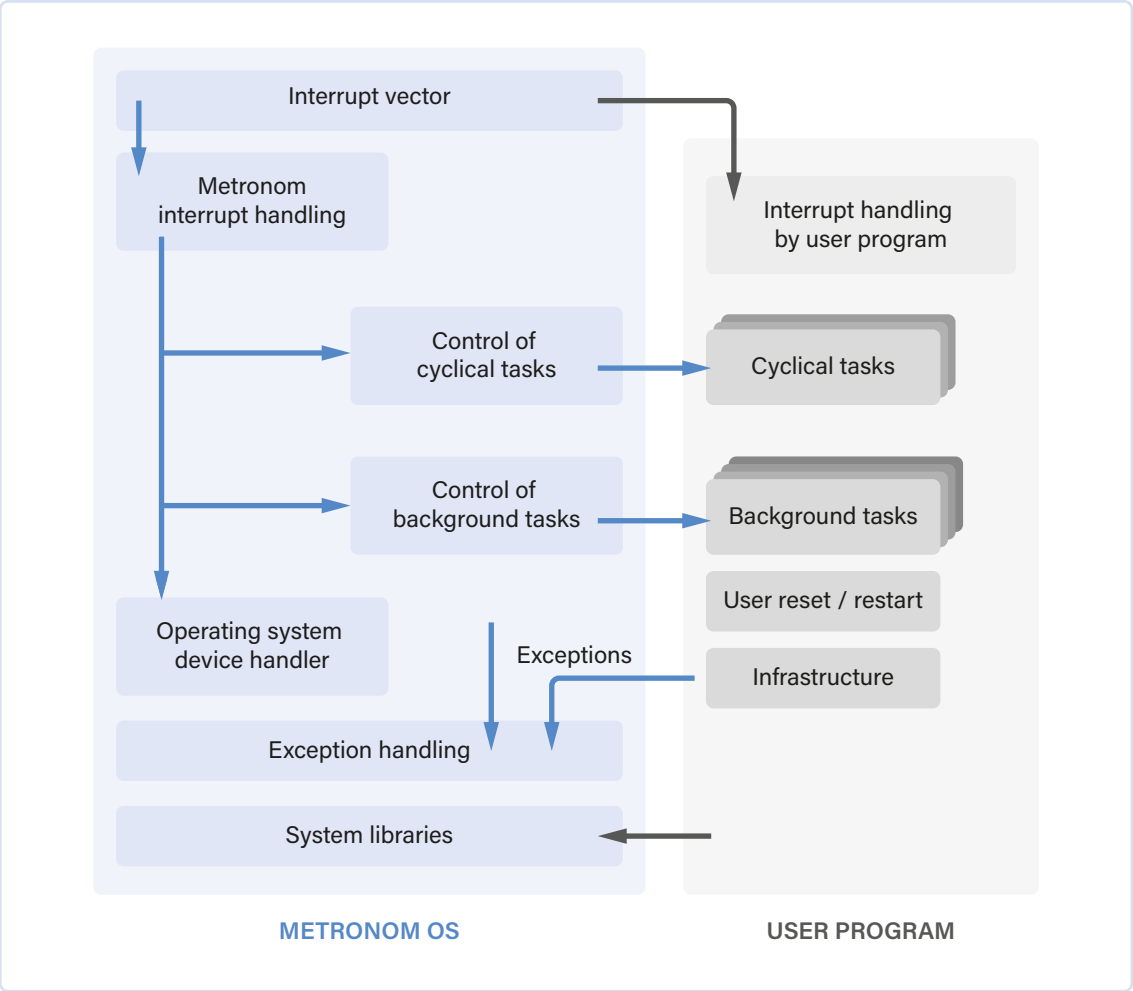


Figure 2. Structure générale du Metronom et du programme utilisateur.

Macros pour l'arithmétique 8 bits et 16 bits

En général, pour les opérations arithmétiques de toutes sortes, les registres r25:r24 sont utilisés comme accumulateur et r23:r22 comme mémoire pour le second opérande (si nécessaire). À cette fin, il existe plus de 20 fonctions différentes, telles que `_mul8u8` pour une multiplication 8×8 bits ou `_abs16` pour une valeur absolue de 16 bits. En outre, il existe de nombreux pseudo-codes de chargement et de sauvegarde, comme `_ld16` (chargement d'un nombre de 16 bits dans l'accumulateur).

Macros pour l'utilisation de l'EEPROM

- `_KWRITE_TO_EEPROM` pour écrire dans l'EEPROM
- `_KREAD_FROM_EEPROM` pour la lecture de l'EEPROM

Macros pour l'utilisation de l'USART

- `_KWRITE_TO_LCD` est un pilote USART spécifique pour un écran LCD 2×16, qui ajoute les caractères de contrôle nécessaires au texte à afficher.
- `_KREAD_FROM_USART` (pas encore implémenté).

Générateur de système SysGen

Pour générer un système (c.-à-d, le code complet), un générateur de système dédié SysGen, qui fait également partie du paquet global, est utilisé. SysGen n'est pas limité à Metronom, mais peut également être utilisé pour des tâches de génération générales.

Vous pouvez vous demander pourquoi un générateur de système distinct a été développé, étant donné qu'il existe une grande variété de préprocesseurs et de générateurs de macros. Pour la génération du système d'exploitation Metronom, les fonctionnalités des préprocesseurs d'Atmel Studio ainsi que du C standard ne sont pas suffisantes. En particulier, comme le préprocesseur ne prend pas en charge l'arithmétique des chaînes, il est impossible de spécifier un « répertoire par défaut » ou un « répertoire de bibliothèque » et d'y sélectionner les fichiers qu'il contient. En faisant une recherche sur Stack Overflow, j'ai constaté que d'autres personnes ont le même problème que moi, mais aucun des préprocesseurs existants ne peut le résoudre.

Le préprocesseur d'Atmel Studio (ainsi que le préprocesseur GNU) offre les fonctions suivantes pour rassembler les fichiers requis :

- `define / set =`
- `if … elif … else … endif`, également intégré
- `ifdef, ifndef`
- `include | exit`

Il manque les fonctionnalités suivantes :

- `<path>` ne peut être passée que sous la forme d'une chaîne fixe, mais une expression de (n'importe quel nombre) chaînes partielles, à la fois des variables et des constantes de chaîne, serait nécessaire.
- `define` et `set` ne peuvent affecter que des valeurs numériques, pas de chaînes de caractères, pas de concaténation de chaînes de caractères, et pas non plus d'expressions logiques.
- Pour la declaration (unique) de programmes de bibliothèque, les possibilités de macros offertes par AVRASM ou le préprocesseur en C ne sont pas suffisantes ; comme les macros d'AVRASM

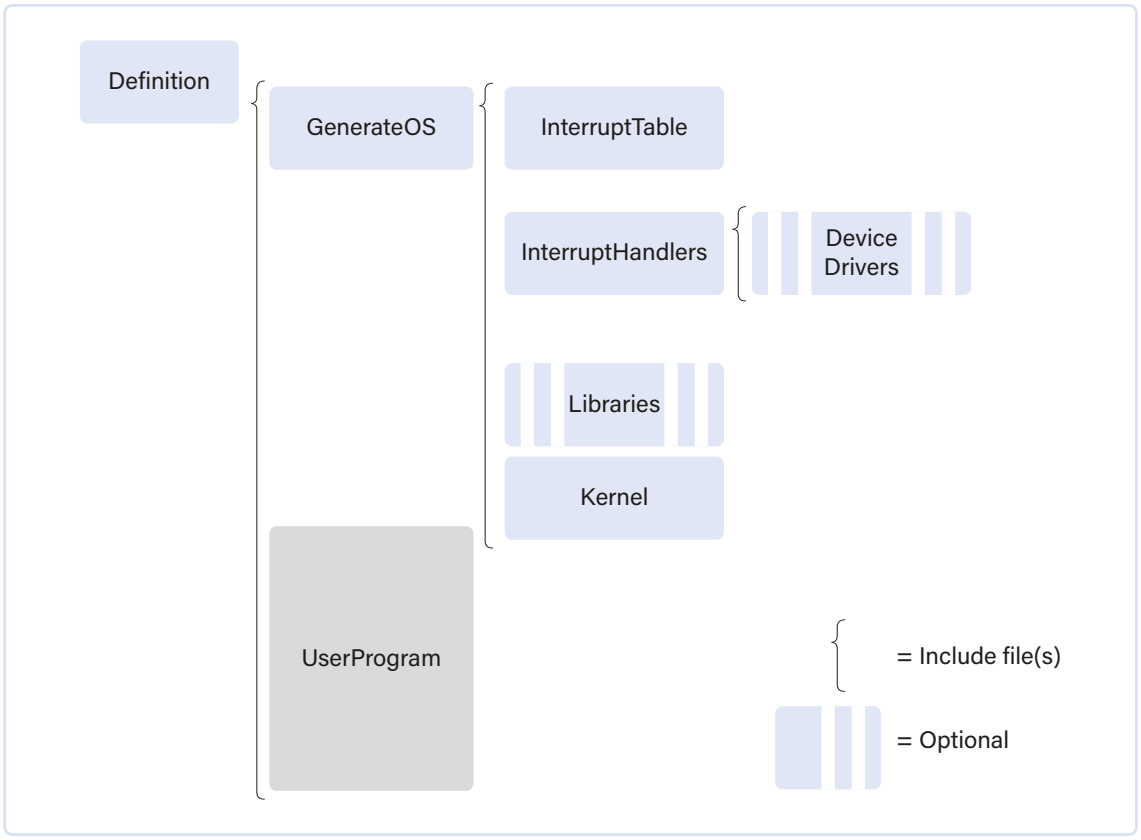


Figure 3. Structure de génération des systèmes Metronom.

ne peuvent pas contenir d'instructions *include*, la déclaration automatique de routines d'émulation, par exemple, est impossible.

Cela donne lieu aux fonctions suivantes :

- `define / set = | |`
- `if … elif … else … endif`, également intégré
- `ifdef, ifndef` est converti en `if isdef(..)` ou `! isdef(..)` et peut donc également être utilisé dans des expressions booléennes.
- `include | exit`
- `message | error`
- `code` (pour créer des lignes de code)
- `macro/endmacro` avec un étiquetage approprié des paramètres
- Il faut également que les instructions des préprocesseurs existants puissent être combinées avec celles de SysGen sans interférer les unes avec les autres.

Vous pouvez également télécharger le programme SysGen sur [1]. SysGen est écrit en Java (version 12) et nécessite une installation Java correspondante pour fonctionner.

Programmer avec Metronom

Pour vous faciliter la tâche, l'ensemble du système d'exploitation est structuré pour être automatiquement généré. Cela signifie que l'utilisateur n'a qu'à remplir le fichier de définition et – si nécessaire – les routines d'interruption programmées par lui-même ; leur emplacement et la manière dont elles sont connectées sont assurés automatiquement par le processus de génération.
Dans sa forme de base, un système utilisateur est composé des éléments illustrés à la **figure 3**.
La table d'interruption et le noyau sont toujours incorporés ensemble dans le programme global résultant. En revanche, dans le cas des gestionnaires de périphériques et des bibliothèques, seules les parties qui sont réellement nécessaires sont incorporées.
Pour illustrer le processus de génération, le script de génération d'un de mes propres projets est présenté dans le **listage 1**.

210719-04

Des questions, des commentaires ?

Envoyez un courriel à l'auteur (profos@rspd.ch) ou contactez Elektor (redaction@elektor.fr).

LIENS

[1] Generation files for Metronome, generation program SysGen, documentation : http://www.elektormagazine.fr/labs/metronom-a-real-time-operating-system-for-avr-processors

Listage 1

```
; ******************************************************
; Master Definition
; ******************************************************
; Stand: 03.05.2022
;
; This file contains all informations required to generate your user system for
; AVR processors.
; It consists of three parts:
;
; 1. Definitions
;    A bunch of variable definitions defining which functionalities to include.
;    This part must be edited by the user.
;
; 2. An $include statement for the actual generation of the operating system.
;    DO NOT MODIFY THIS STATEMENT!
;
; 3. The $include statement(s) adding the user program(s).
;    This part must be edited by the user.
;

; ******************************************************
; PART 1: DEFINITIONS
;
; This script is valid for ATmega8, ATmega328/P and ATtiny25/45/85 processors.
; If you want to use it for any other processors feel free to adapt it accordingly.

$define processor = "ATmega8"

; Remove the ; in front of the $set directive if you want to use the EEPROM
; $set _GEEPROM=1
; if you want to write your own routines to write to the EEPROM use the following
; definition:
; $set _GEEPROM=2
; Enabling this definition will insert an appropriate JMP instruction to your
; interrupt service routine e_rdy_isr in the InterruptHandlers.asm file
; Remove the ; in front of the $set directive if
; ... you want to output serial data via the USART, or
; ... you want exception messages to be sent outside via the USART
; $set _GUSART=1
; if you want to write your own routines to use the USART
; use the following definition instead
; $set _GUSART=2
; Enabling this definition will enable the interrupt service routines usart_udre_isr,
; usart_rxc_isr and usart_txc_isr in the InterruptHandlers.asm file.

; ------------------------------------------------------
; Define the division ratios of the time intervals for cyclic tasks
; The definition shown here is the standard preset for 1 : 10 : 100 : 1000 ms
; The first ratio being 0 ends the divider chain.
.equ _KRATIO1 = 10◦◦◦◦◦◦◦◦◦◦◦ ; 1 -> 10ms
.equ _KRATIO2 = 10◦◦◦◦◦◦◦◦◦◦◦ ; 10 -> 100ms
.equ _KRATIO3 = 10◦◦◦◦◦◦◦◦◦◦◦ ; 100ms -> 1s
.equ _KRATIO4 = 0◦◦◦◦◦◦◦◦◦◦◦◦ ; end of divider chain
.equ _KRATIO5 = 0
.equ _KRATIO6 = 0
.equ _KRATIO7 = 0
; NOTE: Do not remove "superfluous" .EQU statements but set them to 0 if not used!

; ------------------------------------------------------
; Define the constants used for generation of the 1ms timer interrupt
; IMPORTANT: The following definitions depend on the processor being used
; and the frequency of the master clock
```

```
$if (processor == "ATmega8")
; The definitions below are based on a system frequency of 12.288 MHz (crystal)
; This frequency has been chosen in order to use the crystal also for USART@9600 Bd
;
; set prescaler for counter0 to divide by 256, yields 48kHz counting freq for Counter0
.equ _KTCCR0B_SETUP = 4
; Counter0 should divide by 48 in order to produce interrupts every 1ms;
; since counter0 produces an interrupt only at overflow we must preset
; with (256-48) - 1 = 207.
$code ".equ _KTCNT0_SETUP = " + (256 - 48) - 1

$elif ... similar for other processors
;
$endif
;
; ------------------------------------------------------------
; Define the characteristics of USART transmission
; (if you don't use the USART just neglect these definitions):
$set fOSC = 12288000
$set baud_rate = 9600
$code ".equ _KUBRR_SETUP = " + (fOSC / (16 *baudrate) - 1)

; parity: 0 = Disabled,
; (1 = Reserved), 2 = Enable Even, 3 = Enable Odd
.equ _KPARITY = 0

; stop bits: 0 = 1 stop bit, 1 = 2 stop bits
.equ _KSTOP_BITS = 1

; data bits transferred: 0 = 5-bits, 1 = 6-bits, 2 = 7-bits, 3 = 8-bits, 7 = 9-bits
.equ _KDATA_BITS = 3
;
; ------------------------------------------------------------
; Connect a user defined interrupt handler (except RESET and Timer0)
; by removing the ; in front of the appropriate $set directive;
; don't change any names but just let the $set statement as is

; Interrupts for ATmega8
; $set _kext_int0 = 1    ; IRQ0 handler
$set _kext_int1 = 1      ; IRQ1 handler/initializer is supplied by user
; $set _ktim2_cmp = 1    ; Timer 2 Compare Handler
; $set _ktim2_ovf = 1    ; Timer 2 Overflow Handler
; $set _ktim1_capt = 1   ; Timer 1 Capture Handler
;
; etc. etc. etc.

;
; *********************************************************
; PART 2: GENERATING THE OPERATING SYSTEM
;
.LISTMAC
;
$include lib_path + "\GenerateOS.asm"
;
;
; *********************************************************
; PART 3: ADD THE USER PROGRAM
;
$include user_path + "\MyApplication.asm"
;
$exit
```

DVI sur le RP2040

Entretien avec Luke Wren, développeur de composants chez Raspberry Pi

Mathias Claußen (Labo d'Elektor)

Luke Wren, l'un des ingénieurs du Raspberry Pi RP2040, a réalisé des choses étonnantes en le poussant dans ses retranchements. Mathias Claussen, ingénieur chez Elektor, a voulu en savoir plus, notamment sur les astuces et les bidouillages nécessaires pour obtenir une sortie vidéo à partir d'un appareil aussi minuscule.

Le Raspberry Pi RP2040 est un microcontrôleur à moins d'un euro doté de capacités étonnantes. L'un de ses développeurs est Luke Wren, qui a non seulement travaillé sur le Raspberry Pi RP2040, mais a également montré que le micro peut « faire du DVI ». (Reportez-vous à l'article « Sortie vidéo sur les microcontrôleurs (2) » [1]). Lorsqu'il ne travaille pas sur des projets secrets pour Raspberry Pi, il partage certains de ses projets personnels avec la communauté sur Twitter (@wren6991) et le code sur GitHub [2].

Mathias Claussen : pouvez-vous nous parler un peu de vous ?
Luke Wren : je commence par la question la plus difficile ! Je suis ingénieur chez Raspberry Pi, et quand je ne fais pas ça, je travaille généralement sur mes projets de loisirs, je joue mal de la guitare, ou, plus récemment, je consacre du temps à l'apprentissage des langues. Je vis à Cambridge, au Royaume-Uni, non pas parce que c'est une ville particulièrement excitante, mais plutôt par inertie après avoir obtenu mon diplôme universitaire. J'ai vécu en Allemagne quand j'étais plus jeune, mais mon allemand est assez rouillé, donc je suis content que nous fassions cela en anglais.

Mathias : depuis combien de temps travaillez-vous chez Raspberry Pi ?
Luke : je suis entré en tant qu'employé en septembre 2018, mais j'ai fait un stage chez Raspberry Pi auparavant.

Mathias : quel a été votre rôle dans l'élaboration du RP2040 ?
Luke : j'ai travaillé sur une partie de la conception numérique, principalement PIO, DMA, cache XIP, conception de bus et PWM. J'ai également travaillé sur la ROM de démarrage et le SDK, et, bien sûr, j'ai dû participer à la documentation.

Mathias : obtenir un signal vidéo à partir d'un processeur est quelque chose que le Sinclair ZX81 arrivait déjà à faire, mais sortir du DVI, c'est nouveau à ce prix. Alors que la sortie VGA peut être réalisée par la plupart des microcontrôleurs, quel est le défi avec le DVI ?
Luke : il y a deux choses qui rendent le DVI-D plus difficile que le VGA. La première est la sérialisation des données : l'horloge minimale des pixels pour DVI-D est de 25 MHz, et celle des bits est dix fois plus élevée, donc, au minimum, vous pilotez trois lignes série différentielles (rouge/vert/bleu) à 250 Mbps.
La seconde est que le DVI-D encode les pixels avant de les envoyer. Le codage est simple sur le plan matériel, mais un peu compliqué sur le plan logiciel, surtout lorsqu'il doit suivre la vitesse de la sortie série. Tout le reste est similaire. Il ne s'agit en fait que de DPI dans un tuyau plus rapide.
(*Note du rédacteur : Display Pixel Interface (DPI) est une interface de pixels RVB parallèle, incluant une horloge de pixel, pour transporter les pixels à un dispositif d'affichage*).

Mathias : quelle était votre motivation pour essayer le DVI sur le RP2040 ?
Luke : une fois le stress de la mise en place du silicium passé, quelques-uns d'entre nous ont voulu voir jusqu'où ils pouvaient pousser la fréquence d'horloge du système. Il y a, en pratique, une certaine marge au-dessus de la fréquence nominale de 133 MHz. J'avais joué avec DVI-D sur FPGA, dans le cadre de mon projet RISCBoy [3], et lorsque j'ai remarqué qu'il y avait un chevauchement entre les fréquences d'horloge des

bits DVI les plus basses et les fréquences d'horloge système les plus élevées sur RP2040, une idée a germé dans ma tête. La motivation était : « je me demande si c'est possible ».

Mathias : quelle a été la partie la plus difficile pour obtenir une sortie DVI (figure 1) sur le RP2040 ?
Luke : le codage TMDS. Si vous suivez l'algorithme dans la spécification DVI, il n'y a aucun espoir de le rendre suffisamment rapide sur deux cœurs Cortex-M0+ fonctionnant à la fréquence de l'horloge des bits. Il y a donc des astuces et des raccourcis pour rendre cela possible, puis du code soigneusement écrit à la main pour le rendre suffisamment rapide. Le RP2040 a beaucoup de mémoire, mais pas assez pour stocker la valeur d'une image de pixels encodés par TMDS, donc vous devez « courir après le faisceau » pendant l'encodage.

Mathias : vous avez dû légèrement overclocker le RP2040 (de 133 MHz de base à 252 MHz). Y a-t-il un point critique pour les signaux DVI (vous devez piloter les broches d'E/S avec des vitesses qui sont également plus rapides que la vitesse de base) ?
Luke : la première contrainte que vous rencontrerez sur le RP2040 est que l'horloge système doit être 1:1 avec l'horloge de bits, donc si vous essayez de passer à des modes de résolution plus élevés, les processeurs vont tout simplement se planter. Le chemin de configuration critique pour le domaine de l'horloge système sur le RP2040 sont les signaux de phase d'adresse du processeur vers les SRAM.
Cela dit, nous sommes également assez proches des limites de ce que l'on peut faire passer par les ports 3V3 à usage général. Si vous regardez le diagramme de l'œil (**figure 2**) pour 720p30 (372 Mbps) sur mon GitHub [2], ça passe, mais c'est rare. Je doute que vous puissiez voir du 1080p30 sans matériel dédié.

Mathias : outre l'augmentation de la vitesse, quel est le rôle crucial des PIO et de l'interpolateur dans le RP2040 pour faire fonctionner le DVI ?
Luke : c'est une grande exigence le fait de devoir présenter trois bits de données série, plus leurs compléments différentiels, sur les GPIO à 250 Mbps minimum. Pouvoir effectuer la conversion du mode asymétrique au mode pseudo-différentiel dans le PIO réduit de moitié la bande passante DMA, et le fait d'avoir les voies TMDS divisées en trois FIFO est utile si vous effectuez l'encodage en logiciel, car cela vous permet de spécialiser votre code pour l'encodage des composants rouge/vert/bleu. Donc, quelque chose comme PIO est crucial si vous n'avez pas de matériel dédié.
Les interpolateurs contribuent aux performances de génération d'adresses dans le codage TMDS, ce qui est certainement la clé de certaines des démonstrations que vous avez vues, mais mon truc de codage TMDS doublé en pixels tiendrait toujours sur un seul cœur Cortex-M0+ sans les interpolateurs.

Mathias : votre Pico DVI Sock pour le RP2040 utilise une connexion HDMI physique (figure 3), clairement étiquetée comme DVI uniquement, donc nous n'aurons pas d'audio. S'agit-il « simplement » d'un problème de licence avec le consortium HDMI ?
Luke : il n'y a rien qui vous empêche d'ajouter des îles de données HDMI et de faire une sortie audio. En fait, quelqu'un l'a fait avec un port d'émulateur NES ! [4] [5]. Aucune connexion physique supplémentaire n'est requise pour les signaux audio, bien que, strictement parlant, vous ne soyez pas censé utiliser les fonctions HDMI avant d'interroger le canal de données de l'écran, qui n'est pas branché sur mon Pico DVI Sock. La situation de la licence HDMI est certainement une

Figure 1. Raspberry Pi Pico avec le Pico DVI Sock. (Source : Luke Wren).

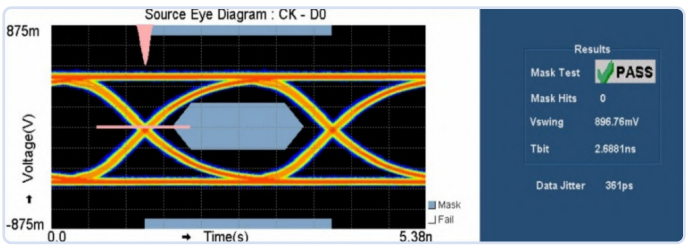


Figure 2. Diagramme de l'œil pour RP2040 DVI à 720p30. (Source : Luke Wren).

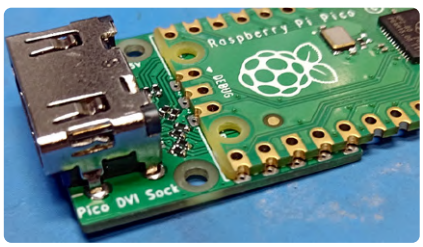


Figure 3. Pico DVI Sock pour le Raspberry Pi Pico. (Source : Luke Wren).

Figure 4. Images à haute résolution avec quelques astuces. (Source : Luke Wren).

boîte de Pandore que je ne veux pas ouvrir, et je me suis également mis en retrait en appelant le référentiel « PicoDVI », donc je vais laisser cette question à la communauté.

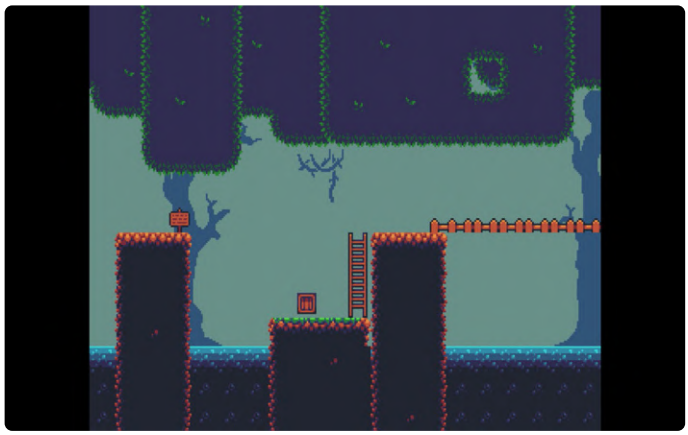
Mathias : lorsque vous utilisez la sortie DVI, quelle proportion des ressources du RP2040 est liée à cette tâche ? Y a-t-il du temps libre pour exécuter d'autres fonctions sur le microcontrôleur ?
Luke : cela dépend du mode vidéo. Disons que pour une sortie RVB565 doublée en pixels, vous finissez par dépenser environ 65 % d'un cœur pour le codage TMDS et les interruptions DMA, et l'autre cœur est alors entièrement disponible pour générer la vidéo et exécuter votre programme principale.
*Note du rédacteur : outre la génération de vidéo pure, quelques applications pour le Pico DVI Sock ont été ajoutées. L'une d'entre elles consiste à déplacer plusieurs visages d'Eben Upton sur l'écran. Si nous faisons quelques calculs, avoir une image de 640 × 480 pixels stockée comme une image complète avec une résolution de 8 bits prendrait ~308 KB de RAM (plus que ce que le RP2040 possède), donc nous le fixons à un maximum de 320 × 240 avec une couleur de 16 bits (154 KB) dans la RAM, mais la démo (***figure 4***) n'est pas pixelisée de cette façon. Il semble donc qu'il y ait un trucage du logiciel.*

Figure 5. Démo Sprite pour le Raspberry Pi RP2040.

Mathias : le logiciel permettant de générer un signal DVI est accompagné d'une bibliothèque qui gère également les sprites. Pouvez-vous en parler davantage ?
Luke : bien sûr ! Donc, lorsque vous créez une sortie vidéo, le problème suivant, c'est d'avoir besoin de données vidéo à sortir, et la bibliothèque sprite ARMv6-M est quelque chose que j'ai conçu tout en travaillant sur PicoDVI dans ce but précis.
La caractéristique essentielle de cette bibliothèque est qu'elle ne nécessite pas de tampon de trame pour le rendu, mais seulement un tampon de ligne de balayage. Votre rendu suit le faisceau, juste avant l'encodage TMDS. Cela signifie que vous pouvez prendre en charge des résolutions de sortie vidéo qui ne tiendraient pas dans la mémoire en tant que tampon de trame plat, et cela laisse la majeure partie de votre mémoire libre pour des objets graphiques.
Il y a des routines de remplissage et de *blit* raisonnablement rapides, des routines de tuilage et des routines de transformation affine de *sprites* qui vous permettent de faire tourner des *sprites* ou les mettre à l'échelle. C'est suffisant pour des jeux du niveau d'une Game Boy Advance ou autre. (*Note du rédacteur : vous pouvez voir un exemple dans la* **figure 5**).

Mathias : où avez-vous trouvé l'inspiration pour la bibliothèque, et le temps pour l'écrire ?
Luke : j'ai passé un certain temps à travailler sur du matériel graphique basé sur le *scanline* pour RISCBoy, donc, ayant fait cela matériellement, il était assez facile de le reproduire en logiciel. Tout ce qui se trouve dans le référentiel PicoDVI a été fait pendant mon temps libre sur mon ordinateur portable, à l'exception des diagrammes de l'œil, pour lesquels j'ai utilisé un oscilloscope au travail.

Mathias : il existe une démo sprite inspirée de Zelda pour le RP2040 (figure 6). Pouvez-vous nous parler de l'idée derrière cela ? Une NES ou une SNES utiliserait un matériel dédié pour composer



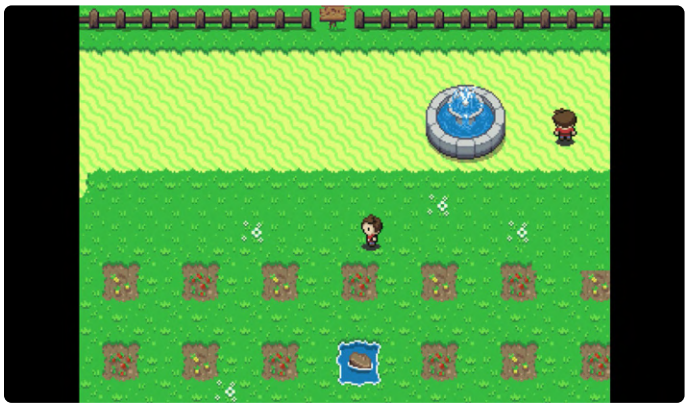


Figure 6. Démonstration du Walker avec sortie DVI.

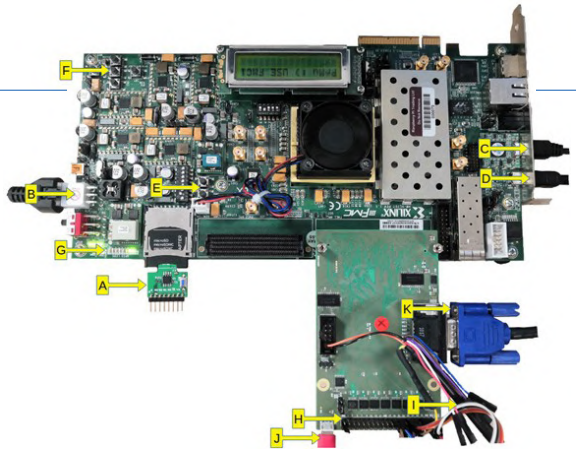
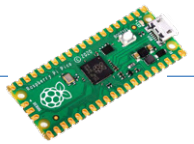


Figure 7. Prototype du Raspberry Pi RP2040 sur FPGA. (Source : Luke Wren).

des images de ce type, et ici nous avons juste deux CPU qui déplacent les pixels.
Luke : c'est en fait un portage d'une des démos RISCBoy. Comme vous le dites, il faut beaucoup de ressources pour faire tout cela en logiciel, et RISCBoy à 36 MHz peut mettre autant de *sprites* à l'écran que le RP2040 à 252 MHz. [6]

Mathias : dans les documents relatifs à la bibliothèque mentionnée, il y a l'idée d'un clone de Mario Kart pour le RP2040 ? Est-ce que c'était juste une idée, ou est-ce qu'on a commencé à travailler pour le faire fonctionner ? Il est également mentionné que l'interpolateur serait utile pour cela.
Luke : je dois donc cracher le morceau à ce stade et admettre que le fait de vouloir faire des textures et des mappages de tuiles dans le style du MODE7 de la SNES est la raison initiale pour laquelle l'interpolateur se trouvait dans la puce, bien qu'entre-temps nous ayons passé du temps à en faire un matériel plus utile et plus performant.
Nous n'avons jamais mis au point un véritable clone de MK, bien que vous puissiez voir de nombreux exemples de personnes utilisant des techniques similaires en ligne ; nous avions un cube 3D texturé avec le visage d'Eben fonctionnant sur FPGA.

Mathias : dans la documentation de votre Pico DVI Sock pour le Pico RP2040, vous mentionnez un prototype FPGA de 48 MHz. Pouvez-vous nous en dire un peu plus ?
Luke : nous utilisions une tâche automatisée pour compiler toutes les nuits une image FPGA à partir du dernier code source du RP2040 afin que, le lendemain, nous puissions l'utiliser pour le développement de logiciels.
Nous avons simplement utilisé une carte de développement Virtex 7 du commerce avec une carte fille pour décaler le niveau des E/S du FPGA jusqu'à 3,3 V (**figure 7**), et une autre petite carte qui place une mémoire flash QSPI dans le socket SD, qui est connectée à l'interface RP2040 XIP [7].
L'exécutable du FPGA est pratiquement un RP2040 [8] complet, le circuit d'horloge/de réinitialisation est simplifié et l'ADC est supprimé, mais à part cela, tout est présent et correct. Cela en fait une plateforme idéale pour le développement de logiciels, bien que la vérification réelle ait utilisé des simulations conventionnelles et un contrôle formel du modèle.



Mathias : outre la sortie DVI, vous travaillez ou avez travaillé sur d'autres projets. L'un d'eux est la PicoStation 3D. Pouvez-vous nous en parler un peu ? Si vous pouviez vous procurer toutes les pièces, serait-ce toujours le même design aujourd'hui ?
Luke : donc, PicoStation 3D (**figure 8**) est l'un de mes nombreux projets personnels que j'avais en vol avant le lancement de la RP2040. Il s'agit d'une carte contenant une RP2040, un FPGA iCE40UP5K, une carte microSD, une sortie audio, une sortie DVI-D via une prise HDMI et deux prises pour contrôleur SNES. Je lisais beaucoup sur le matériel graphique 3D à ce moment-là et je voulais une plateforme pour jouer avec, dans le contexte d'une petite console de jeux.
Cela me fait de la peine d'avoir laissé ce projet en veilleuse pendant si longtemps, mais outre les problèmes de pièces, j'ai également trop d'autres projets en cours. Tout est open-source, donc j'adorerais que quelqu'un d'autre reprenne l'idée et l'exploite.
Je pense que le choix du FPGA est tout à fait approprié, il est petit et suffisamment lent pour vous faire travailler dur pour vos démos, tout juste compatible DVI-D, et il dispose d'une mémoire embarquée généreuse et d'une poignée de tuiles DSP 16 bits, ce qui en fait une plateforme idéale pour jouer avec du matériel graphique. Il se marie également bien avec le RP2040.
Ce à quoi j'aimerais réfléchir, c'est à l'entrée-sortie. Par exemple, que se passerait-il si vous déplaciez le DVI vers le microcontrôleur et si vous déplaciez les contrôleurs SNES vers le FPGA, et si vous amélioriez un peu le circuit audio, ce genre de chose.

Figure 8. Le prototype de PicoStation3D. (Source : Luke Wren).

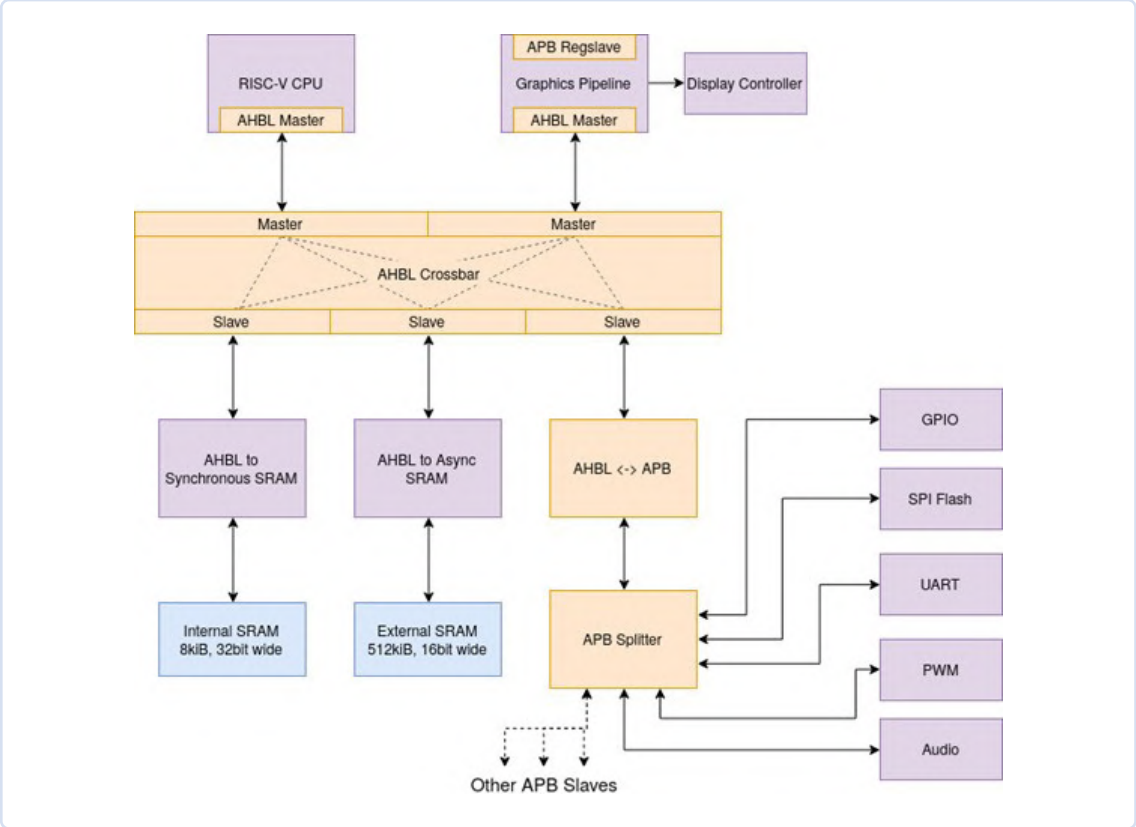


Figure 9. L'architecture du RISCBoy. (Source : Luke Wren).

Je pense que les dimensions sont à peu près correctes, puisqu'elles sont définies par les deux connecteurs de la manette SNES.

Mathias : outre les produits basés sur Raspberry Pi, vous avez également réalisé un RISCBoy, qui est alimenté par un RISC-V Core développé par vos soins et d'autres périphériques, tels qu'un moteur graphique (basé sur la 2D-sprite ?). Pouvez-vous nous dire quelques mots sur ce développement ?
Luke : RISCBoy est mon concurrent un peu tardif de la Game Boy Advance. Le matériel complet tient dans un iCE40 HX8K avec un peu de SRAM parallèle externe (**figure 9**). Eventuellement, il y aura une version physique, mais, pour l'instant, c'est toujours une carte de développement HX8K avec un peu de SRAM et des boutons, et un écran SPI suspendu à elle (**figure 10**). J'ai commencé à travailler dessus au moment où j'ai quitté l'université.

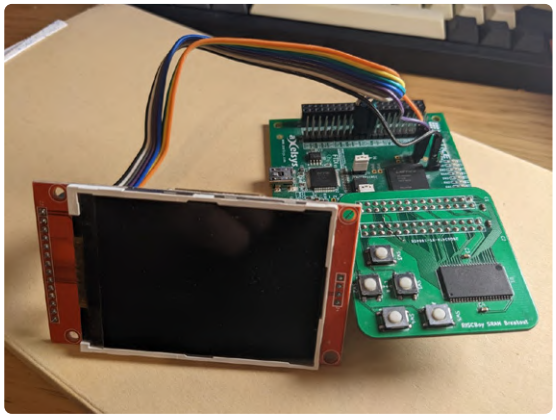
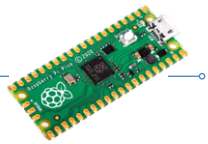


Figure 10. Le prototype du RISCBoy. (Source : Luke Wren).

Tout est écrit à partir de rien, ce n'est pas une bonne façon de faire, mais c'est une excellente façon d'apprendre, et cela rend le débogage plus amusant quand il n'y a pas une seule partie de la pile matériel/logiciel à laquelle vous pouvez faire confiance. Il y a un processeur à 32 bits (Hazard5), du matériel graphique 2D programmable, et toute l'infrastructure qui relie le tout.
Le matériel graphique réalise tous les *sprites* et tuiles et leurs transformations affine, etc. habituels, mais il le fait en exécutant, à partir de la mémoire, des listes de commandes qui fournissent un support limité pour le flux de contrôle et les branchements aux sous-routines. Pendant chaque image, le processeur écrit la liste de commandes pour l'image suivante. Il y a une paire de tampons de lignes de balayage dans le matériel, un dans lequel le rendu est effectué et un autre qui est envoyé à l'écran, de sorte que vous évitez les coûts de bande passante, de latence et d'empreinte mémoire du rendu dans un tampon de trame.
Il est en suspend pour le moment, mais j'ai bien l'intention de le terminer un jour.

Mathias : votre cœur RISC-V gagne en maturité de jour en jour. Pouvez-vous nous parler de ses origines ?
Luke : j'ai quelques processeurs à différents stades d'achèvement en ce moment, mais je suppose que vous demandez à propos de Hazard3. C'est un processeur scalaire à exécution dans l'ordre et à trois étages, originellement dévié de la source Hazard5 (le processeur RISCBoy). Vraiment, Hazard3 est un outil d'apprentissage pour moi, c'est bien beau de lire les spécifications RISC-V, mais ce n'est pas la même chose que



d'aller les implémenter, et utiliser un simple « trois étages » comme base me permet de me concentrer sur la périphérie parce que le pipeline de base fonctionne. Cela dit, les performances sont assez compétitives de nos jours (environ 3.2 CoreMark/MHz) et j'ai consacré beaucoup de temps à la vérification et à la documentation. L'introduction du débogage matériel et l'exécution de tests de bout en bout avec GDB, OpenOCD et mon propre module de transport JTAG a probablement été le sommet du projet jusqu'à présent, mais j'ai beaucoup appris tout du long.
À l'avenir, je pense que Hazard3 sera une mine d'or de composants pour tous les futurs processeurs 32 bits de classe embarquée sur lesquels je travaille. J'ai déjà vu quelqu'un sur Twitter utiliser mon module de débogage pour ajouter le support de débogage au noyau de quelqu'un d'autre. Les sources sont toutes autonomes et devraient être assez portables. Il est également sous licence Apache-2.0.

Mathias : actuellement, la conception est un RISC-V 32 bits, qui est fait entièrement en open-source. Avez-vous obtenu des ressources de Raspberry Pi pour son développement (temps, matériel, outils logiciels) à un moment donné ?
Luke : tous mes projets maison sont réalisés sur mon temps, avec mon matériel et des outils open-source. En ce moment, j'ai une machine Ryzen 7 5800X à la maison, ce qui aide à exécuter des travaux par lots. J'utilise Yosys pour la synthèse FPGA, nextpnr pour le placement et le routage, CXXRTL pour la simulation. J'utilise un iCEBreaker (iCE40UP5K) et un ULX3S (ECP5 85F) comme plateformes de référence pour Hazard3, bien que j'aie un nombre assez embêtant d'autres cartes de développement FPGA que je devrais utiliser davantage. Lorsque je dois déboguer quelque chose sur FPGA, je m'en sors bien avec mon Saleae Logic 8 que j'ai acheté lorsque j'étais étudiant, mais la plupart du temps, je peux me fier aux simulations.

Mathias : actuellement, vous passez de 32 bits à 64 bits avec votre noyau. Quel a été l'obstacle le plus difficile à surmonter jusqu'à présent au cours de son développement ?
Luke : jusqu'à présent je n'ai passé qu'un week-end à jouer avec RV64, et c'était suffisant pour passer la conformité RV64IC et les tests de conformité de débogage, donc il n'y a pas une courbe d'apprentissage énorme, les choses deviennent juste plus grandes et plus lentes.
Je hackais la base de code Hazard3 pour des raisons de commodité, mais si je voulais être sérieux dans l'implémentation d'un RV64, alors il y aurait des changements micro-architecturaux nécessaires. Il y a une raison pour laquelle vous ne voyez pas beaucoup de processeurs 64 bits à trois étages, et certainement pas ceux avec la décision de branchement à l'étage 2. Hazard3 va rester un processeur 32 bits de classe embarquée.

Mathias : est-il prévu d'essayer le cœur à un moment donné dans le futur sur du silicium réel ? Ou même de le combiner avec un moteur 2D pour obtenir un RISCBoy64 ?
Luke : j'aimerais bien essayer une bande SkyWater PDK à un moment donné, mais, pour l'instant, je me concentre sur d'autres projets. Je veux construire quelque chose de plus intéressant que « juste un autre système RISC-V » et je ne suis pas encore sûr de ce que ce sera. Pour quelque chose comme RISCBoy, il n'y a pas beaucoup d'avantages à utiliser un processeur à 64 bits.

Mathias : merci pour votre temps, Luke Wren.

220575-04 — VF : Maxime Valens

Produits

- **Raspberry Pi Pico (SKU 19562)**
  www.elektor.fr/19562
- **DVI Sock pour Raspberry Pi Pico (SKU 19925)**
  www.elektor.fr/19925

LIENS

[1] Mathias Claußen, « sortie vidéo sur les microcontrôleurs (2) », Elektor 3-4/2023 : http://www.elektormagazine.fr/220614-04
[2] Dépôt GitHub de Luke Wren : https://github.com/Wren6991
[3] RISCBoy @ GitHub : https://github.com/Wren6991/RISCBoy
[4] pico-infones @ GitHub : https://github.com/shuichitakano/pico-infones
[5] Vidéo @ Twitter : https://twitter.com/shuichi_takano/status/1477702448907419649
[6] Démonstration de Sprite fonctionnant sur le matériel RISCBoy à 36 MHz : https://twitter.com/wren6991/status/1333708886956707840
[7] Photo de la carte SD QSPI : https://twitter.com/wren6991/status/1134719550027632640
[8] Microcontrôleur RP2040 : https://raspberrypi.com/products/rp2040/

L'afficheur HAT Mini

affichez les prévisions météo avec Raspberry Pi !

Clemens Valens (Elektor)

L'afficheur HAT Mini de Pimoroni est équipé d'un écran LCD rectangulaire de 2,0 pouces de 320 x 240 pixels, de quatre boutons poussoirs et d'une LED RGB. Destiné aux Raspberry Pi Zero et Zero 2 W, il est idéal pour les applications IdO et domotiques.

Figure 1. L'afficheur HAT Mini dispose également de quatre boutons poussoirs et d'une LED RGB.

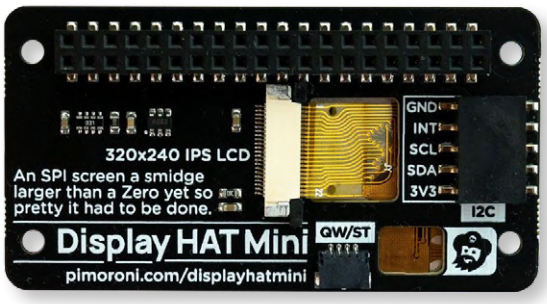


Figure 2. Des connecteurs d'extension I²C sont disponibles sur la face arrière.

L'afficheur HAT Mini de Pimoroni est équipé d'un écran LCD rectangulaire IPS (*In-Plane Switching*) d'une diagonale de 2,0 pouces avec interface SPI. Il est destiné aux Raspberry Pi Zero et Zero 2 W. Mais comme il possède un connecteur HAT standard à 40 broches, il est possible de le brancher sur n'importe quel Raspberry Pi disposant d'un tel connecteur. Vous devez faire attention, car le connecteur I²C du HAT (« *Breakout Garden* ») peut entrer en contact avec le connecteur d'affichage du Raspberry Pi.

Spécifications

La résolution de l'écran est de 320x240 pixels (3:2), qui est équivalente à environ 200 ppi (pixels par pouce). La profondeur de couleur est de 65 K. Montée sur un Raspberry Pi Zero sans entretoises, la hauteur totale est d'environ 15 millimètres. L'afficheur est livré avec deux entretoises de 10 mm de haut et quatre petites vis (cinq dans mon cas). À mon avis, la longueur de ces entretoises est très élevée. Une longueur de 8,5 mm aurait été plus adaptée (mais pas standard).

Outre l'écran, le HAT comporte également quatre boutons poussoirs et une LED RGB (**figure 1**). Les boutons poussoirs sont très proches de l'écran, ce qui les rend un peu difficiles à utiliser. Je pense qu'ils ont été placés ainsi pour fournir un support mécanique supplémentaire pour l'afficheur.

Même si le HAT bloque l'accès au connecteur HAT à 40 broches, des extensions sont toujours possibles grâce au connecteur Qw/ST (Qwiic/STEMMA QT) et à l'embase dite *Breakout Garden*. Ces deux connecteurs permettent d'accéder au bus I²C (**figure 2**).

Pris en charge par les bibliothèques Python

Pour utiliser l'afficheur HAT Mini avec un Raspberry Pi, vous devez installer une bibliothèque. Pour le faire, des instructions détaillées sont fournies par le fabricant sur GitHub [2]. Vous trouverez également des exemples d'utilisation de l'afficheur avec *Pygame* et *PIL*.

J'ai branché l'afficheur Mini HAT sur un Raspberry Pi Zero 2 W exécutant Buster et je l'ai contrôlé via SSH. Après avoir installé les bibliothèques, tous les exemples ont été réalisés avec succès.

Construisons un afficheur de prévisions météorologiques

Une fois l'afficheur installé et fonctionnel, j'ai voulu en faire quelque chose. Sur Internet, j'ai trouvé une collection d'icônes météo attrayantes. J'ai donc décidé de créer un afficheur de prévisions météo avec Python 3 qui affiche l'icône météo correspondante ainsi que la température, la pression atmosphérique, l'humidité, la direction et la vitesse du vent (**figure 3**). Il est possible d'obtenir ces données à partir l'un des serveurs de prévisions météo en ligne, il y en a beaucoup qui offrent un accès gratuit.

J'ai utilisé les boutons poussoirs « A » et « B » pour contrôler la luminosité du rétroéclairage de l'écran. Le bouton « X » vous permet de choisir entre les degrés Celsius et les degrés Fahrenheit, tandis que le bouton « Y » permet de choisir la direction du vent en degrés ou en points cardinaux (par exemple, « SW » ou « N »). La LED RGB fournit des informations sur l'état de l'appareil. En fait, le vert devrait indiquer que tout fonctionne bien. Mais, j'ai trouvé que la LED est très brillante même à des valeurs très faibles, alors je l'ai désactivée. Désormais, la LED ne s'allume qu'en cas de problème : rouge lorsque les données météo n'ont pas pu être récupérées et orange lorsque les données météo sont invalides.

Notez que par défaut, l'afficheur est à l'envers par rapport aux légendes des boutons poussoirs du HAT. Pour cette raison, le programme fait pivoter le tampon d'affichage de 180° avant de le copier sur l'écran.

La consommation en courant de mon système à l'intensité maximale du rétroéclairage était d'environ 200 mA.

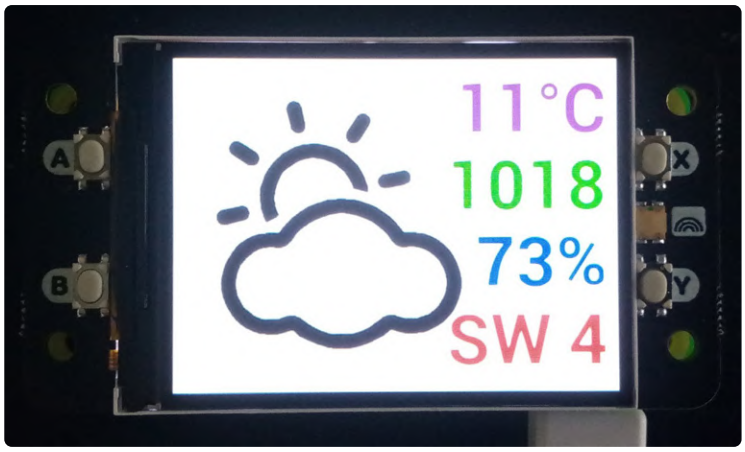


Figure 3. L'afficheur HAT Mini constitue un excellent écran pour les applications IdO ou domotiques.

Mon code (y compris la collection complète d'icônes) est disponible sur [3].

Un accessoire sympa

L'afficheur HAT Mini est un accessoire sympa pour le Raspberry Pi Zero (2W). La qualité d'image est excellente et vous pouvez facilement l'utiliser dans vos applications. La bibliothèque dédiée permet également de contrôler les boutons poussoirs (placés un peu trop proches de l'écran à mon avis), à la LED RGB (un peu trop lumineuses) et au rétroéclairage.

Notez qu'à cause de l'écran, le HAT est légèrement plus grand qu'un Raspberry Pi Zero (2W), 35 mm au lieu de 30 mm, alors choisissez bien votre boîtier.

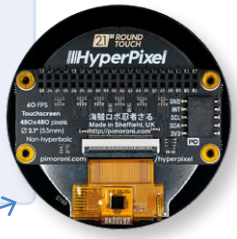
220126-04

Des questions, des commentaires ?

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

Produits

- **Afficheur HAT Mini de Pimoroni pour Raspberry Pi Zero (SKU 19990)**
  www.elektor.fr/19990
- **HyperPixel 2.1 Round – écran haute résolution for Raspberry Pi (SKU 19870)**
  www.elektor.fr/19870



LIENS

[1] L'afficheur HAT Mini de Pimoroni : https://www.elektor.fr/pimoroni-display-hat-mini-for-raspberry-pi-zero
[2] Pimoroni sur GitHub : https://github.com/pimoroni/displayhatmini-python
[3] ClemensAtElektor sur GitHub : https://github.com/ClemensAtElektor/rpi_weather_display

WEEF 2022 Awards : célébrez les bonnes choses

Priscilla Haring-Kuipers (Pays-Bas)

À la foire electronica de Munich le 15 novembre 2022, le deuxième Forum mondial de l'électronique éthique (WEEF) a présenté des discussions et des débats approfondis, sur une série de problèmes d'éthique liés à l'électronique. Rejoignez-nous pour rendre hommage aux quatre lauréats du WEEF.

Lors de notre deuxième Forum mondial de l'électronique éthique (WEEF), qui s'est tenu à Munich le 15 novembre 2022, nous avons remis quatre prix à des personnes remarquables. À partir de notre index WEEF, notre jury a sélectionné quatre lauréats en examinant leur contribution à l'électronique éthique selon trois critères :
1) leur niveau d'influence
2) leur niveau d'innovation
3) leur volonté de partager

Notre index de références du *World Ethical Electronics Forum* est rempli de personnes dont nous savons, au WEEF, qu'elles font de leur mieux pour l'éthique dans l'électronique, **et** de personnes nommées par nos membres et des industriels. Si vous connaissez quelqu'un qui devrait être inclus dans ce groupe de bonnes personnes (et qui devrait être dans la course pour les prochains prix du WEEF), vous pouvez le nommer en remplissant ce formulaire de candidature sur le site web du WEEF. [1]

Le gagnant du WEEF Sven Krumpel

Sven Krumpel [2] est le PDG de CODICO, qui attache une grande importance à l'équilibre entre la vie professionnelle et la vie privée de ses employés. L'entreprise offre une flexibilité quasi totale en matière de lieu et de temps de travail et constitue un excellent exemple d'investissement réel dans ses employés.
« Merci de m'avoir nommé. Je ne savais pas que j'étais une personne éthique. Nous sommes une entreprise familiale, nous concevons et faisons progresser l'électronique. Nous voyons combien il est important de développer des projets d'efficacité énergétique. Je pense que l'éthique est une affaire de personnes, et en tant que personne, vous êtes un exemple. En tant que propriétaire d'une entreprise, vous n'avez pas seulement des employés, mais aussi des personnes qui partagent les mêmes valeurs. Nos collaborateurs font partie de la famille. Nous venons d'ouvrir un parc de 12 000 mètres à côté de nos bureaux, non seulement pour travailler à l'extérieur, mais aussi pour les loisirs. Nos employés font du sport dans le parc, invitent leurs familles et organisent des fêtes d'anniversaire. Nous cultivons des légumes et des herbes aromatiques. Nous avons des abeilles. Je pense que nous avons vécu une vie extrêmement bénie où nous avons utilisé trop de ressources et utilisé les ressources des autres, et nous devons un peu rendre la pareille. C'est ce que nous faisons et il n'y a rien d'exceptionnel à cela. »

Le gagnant du WEEF Gopal Kumar Mohoto

Gopal Kumar Mohoto [3] est un ingénieur qui travaille à l'électrification des transports au Bangladesh. Il a été choisi comme entrepreneur numéro un mondial au *Climate-Launchpad* de 2020, et a été l'ambassadeur du climat au *Global Youth Climate Network*. Il a travaillé sur des stations d'échange de batteries pour les *tuk-tuks* électriques, et est sur le point de mettre à la disposition des clients au Bangladesh le premier groupe de voitures à essence rétrofitées en voitures électriques. Vous pouvez lire son interview ici [4].
« Je suis très honoré d'avoir été désigné, même si je ne sais pas par qui. Les options de conduite électrique disponibles en Occident sont trop coûteuses dans les pays du Sud. C'est pourquoi je travaille sur une solution durable et abordable pour électrifier les transports ici. Nous devons mieux prendre soin des choses que nous possédons déjà ainsi que leur utilisation car elles sont précieuses. Nous devons prendre soin de tous les matériaux que nous avons extraits. Nous devons prendre soin de la terre nourricière. »

Le gagnant du WEEF Frank Stührenberg

Frank Stührenberg [5] est le PDG de Phoenix Contact et membre du conseil d'administration de la fondation KlimaWirtschaft. Il est l'un des principaux contributeurs de All Electric Society, un monde visionnaire dans lequel la régénération électrique est disponible dans le monde entier, en tant que source d'énergie primaire et en quantité suffisante pour tous.
« Je ne sais pas qui nous a nommés. Les prix que notre entreprise reçoit sont généralement techniques, j'ai donc été surpris par celui-ci. Les valeurs de WEEF correspondent à celles de notre entreprise. Nous sommes une entreprise familiale qui compte aujourd'hui plus de 20 000 membres. Nous devons nous efforcer de ne pas perdre nos valeurs morales et nous y travaillons activement. Notre stratégie à long terme à été consacrée à la production d'une énergie électrique renouvelable à 100 %.

Milda Pladaitė (à gauche) reçoit un prix de Beatriz Souza d'ELEKTOR. (Photo prise à la foire de Munich)

Sven Krumpel (à droite) reçoit un prix de Johann Wiesböck d'ELEKTRONIKPRAXIS. (Photo prise à la foire de Munich)

Gopal Kumar Mohoto.

Frank Stührenberg (à droite, avec Reinhard Pfeiffer, le PDG du salon de Munich, photo prise au salon de Munich).

Nous avons pensé que nous devions le faire et nous avons donc commencé. Nous avons besoin d'énergie pour développer le monde, et quelle est la finalité d'être une entreprise valant 10 milliards d'euros, en étant dans un monde invivable pour nous ? Merci de soutenir ce projet et de soutenir cette initiative. »

La gagnante du WEEF Milda Pladaitė

Milda Pladaitė [6] est une ingénieure civile lituanienne et fait partie du comité des futurs leaders des jeunes ingénieurs de la fédération mondiale des organisations d'ingénieurs (WFEO). Elle a mis en place un groupe de travail consacré à l'objectif de développement durable 13 à la WFEO, visant à contribuer au développement durable des pays, en déployant le travail des jeunes ingénieurs de la WFEO dans l'action climatique. Elle travaille actuellement sur une initiative sociale visant à aider les ingénieurs dans leur entrepreneuriat, en les mettant en relation avec des industriels, des investisseurs sociaux et des organisations universitaires.

« J'ai désigné de nombreuses personnes pour des récompenses, mais je ne sais pas qui m'a nominée pour celle-ci. Une amie m'a dit quand elle a appris que j'étais nominée pour ce prix : "Je ne savais pas que tu étais une spécialiste de l'éthique !". Bien sûr, cela signifie que tout le monde peut se sentir concerné par l'éthique. Ce forum est une excellente initiative, et je suis sûr qu'il va se développer à l'avenir. C'est une occasion unique de discuter de l'éthique dans l'électronique. »

Nous pouvons constater par l'étonnement de nos lauréats du prix WEEF que l'éthique dans l'électronique n'est pas si évidente que cela. Nous sommes peut-être réticents à parler d'éthique en tant qu'industrie alors qu'il ne devrait pas en être ainsi. L'objectif de notre Forum mondial de l'électronique éthique est de fournir une plate-forme, pour discuter de ce que nous faisons et pourquoi nous le faisons, pour partager des témoignages et les bonnes pratiques les uns avec les autres, ainsi que pour célébrer ce qui est bien tous ensemble.

220650-04 — VF : Laurent Rauber

À propos de l'auteur

Priscilla Haring-Kuipers écrit des articles sur la technologie d'un point de vue des sciences sociales. elle s'intéresse particulièrement aux technologies pour le bien de l'humanité et croit fermement à la recherche sur les impacts. Elle est titulaire d'un master en psychologie et amène à la réalité *This Is Not Rocket Science*.

Le Forum mondial de l'électronique éthique

Le *World Ethical Electronics Forum* (WEEF), inspire les innovateurs internationaux par des discussions ouvertes et des publications sur l'éthique et les objectifs de développement durable. Visitez le site **worldethicalelectronicsforum.com** pour l'inspiration et les modalités de participation.

LIENS

[1] Formulaire de nomination à l'index WEEF : https://worldethicalelectronicsforum.com/nomination-form
[2] WEEF, Sven Krumpel : https://worldethicalelectronicsforum.com/index/184374/Sven__Krumpel
[3] WEEF, Gopal Kumar Mohoto : https://worldethicalelectronicsforum.com/index/182949/Gopal_Kumar_Mohoto
[4] P. Haring-Kuipers, "Retrofitting and Upcycling: Interview with Gopal Kumar Mohoto," ElektorMagazine.com, 2022 : https://www.elektormagazine.com/gopal-interview-for-next-eia
[5] WEEF, Frank Stührenberg : https://worldethicalelectronicsforum.com/index/184378/Frank__St%C3%BChrenberg
[6] WEEF, Milda Pladaitė : https://worldethicalelectronicsforum.com/index/182955/Milda__Pladait%C4%97

elektorstore

www.elektor.fr

e-choppe Elektor

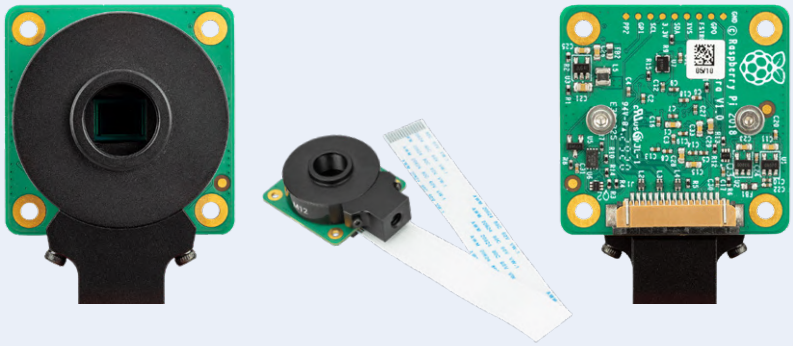
des produits et des prix surprenants

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

Raspberry Pi High Quality Camera Module (M12 Mount)

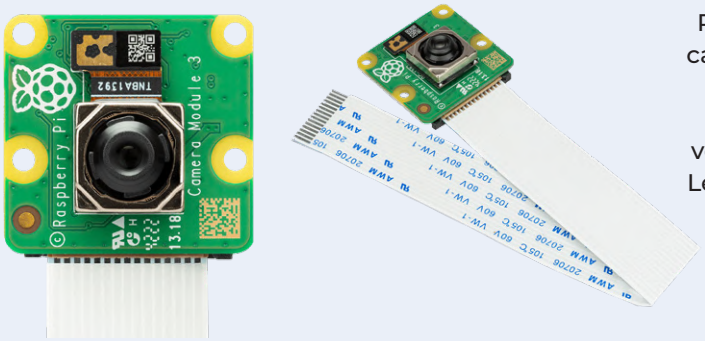
La caméra haute qualité (HQ) pour Raspberry Pi offre une résolution de 12 mégapixels et est dotée d'un capteur de 7,9 mm de diagonale pour des performances impressionnantes en basse lumière.

59,95 €



www.elektor.fr/20366

Raspberry Pi Camera Module 3



La Camera Module 3 est une caméra Raspberry Pi compacte. Elle est dotée d'un capteur IMX708 de 12 mégapixels avec HDR et de l'autofocus à détection de phase. La Camera Module 3 est disponible en version standard et en version grand angle. Les deux variantes sont disponibles avec ou sans filtre infrarouge.

32,95 €

www.elektor.fr/20362

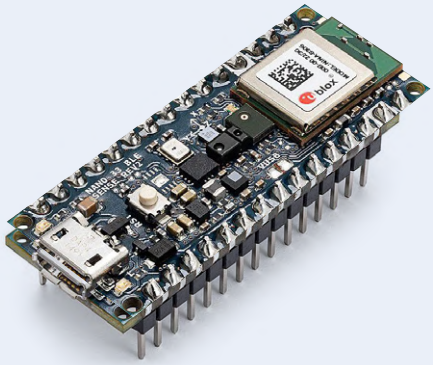
Caméra de profondeur 3D YDLIDAR OS30A

139,95 €

Prix (membres) : 125,96 €

www.elektor.fr/20350

Arduino Nano 33 BLE Sense Rev2 avec connecteurs



54,95 €

Prix (membres) : 49,46 €

www.elektor.fr/20404

Arduino Student Kit

74,95 €

Prix (membres) : 67,46 €

www.elektor.fr/20329

Home Appliance Hack-and-IoT Guidebook

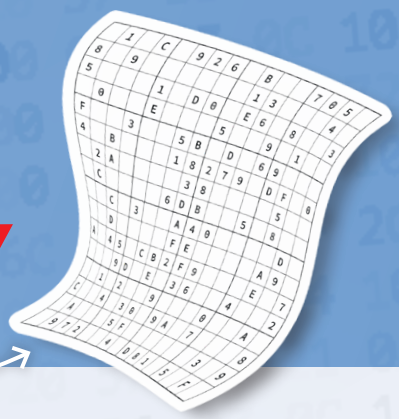
~~54,90 €~~

Prix spécial : 39,95 €

www.elektor.fr/20370

hexadoku

casse-tête pour elektorniciens



La dernière page de votre magazine propose toujours une grille de sudoku particulière. Éteignez le fer à souder, oubliez les écrans, rangez les pinces et les fils. Faites travailler votre matière grise d'une autre façon : attrapez un crayon pour remplir la grille d'hexadoku. N'oubliez pas d'envoyer votre réponse pour tenter de gagner un bon d'achat.

Une grille hexadoku est composée de chiffres du système hexadécimal, de 0 à F. Remplissez le diagramme de 16 x 16 cases de telle façon que **tous** les chiffres hexadécimaux de 0 à F (0 à 9 et A à F) n'apparaissent **qu'une seule et unique fois** dans chaque rangée, colonne et carré de 4 x 4 cases (délimités par un filet gras). Certains chiffres, déjà placés dans la grille, en définissent la situation de départ.

Pour participer, inutile de nous envoyer toute la grille, il suffit de nous donner **la série de chiffres** sur fond grisé.

Participez et gagnez !

Nous tirons au sort cinq des réponses internationales correctes reçues dans les délais ; leurs auteurs recevront chacun un bon d'achat Elektor d'une valeur de 50 €.

Où envoyer votre réponse ?

Envoyez votre réponse (les chiffres sur fond grisé) avec vos coordonnées par courriel, avant le **15 avril 2023** à l'adresse **hexadoku@elektor.fr**

LES GAGNANTS

La solution de la grille du numéro de janvier/février 2023 est **AEF1C**.

La liste des gagnants est publiée ici : www.elektormagazine.fr/hexadoku

Bravo à tous les participants et félicitations aux gagnants !

	1	3	F									E	2	7	
0	6	5											8	C	D
	8				2					E				6	
9	B	D	E	4		7			C		2	F	3	A	5
F	7	E	8			D			B			A	6	5	1
	9	0		6	1					3	F		4	B	
	D	4			F					9			7	8	
1		6	3	2			B	D			8	9	E		F
			6		8	0			D	2		5			
	E	A			D	9	2	8	5	F			0	1	
5				1			E	4			9				8
	0		9		5		4	A		1		C		F	
D						1			8						E
		9		8	7					D	A		5		
	A		1		9	3			2	7		8		D	
7	5					4	F	E	3					9	2

E	5	2	A	4	B	8	C	6	3	F	0	D	7	9	1
0	3	B	4	6	F	2	7	D	9	1	A	E	5	8	C
C	F	1	7	D	E	9	A	B	4	5	8	2	0	3	6
6	D	8	9	0	1	3	5	C	7	E	2	B	A	F	4
1	8	F	3	5	7	4	0	E	B	2	6	9	D	C	A
B	4	C	D	3	2	6	9	8	0	A	5	F	E	1	7
5	A	E	2	8	C	D	1	7	F	9	4	6	B	0	3
7	0	9	6	B	A	E	F	1	C	3	D	4	2	5	8
F	9	7	B	E	6	5	4	0	8	D	C	1	3	A	2
3	E	5	C	7	8	A	2	F	1	B	9	0	4	6	D
D	2	0	8	9	3	1	B	4	A	6	E	C	F	7	5
4	6	A	1	C	0	F	D	5	2	7	3	8	9	B	E
8	C	3	F	1	5	B	E	9	D	4	7	A	6	2	0
9	1	4	0	2	D	7	6	A	5	C	B	3	8	E	F
A	B	6	5	F	4	0	3	2	E	8	1	7	C	D	9
2	7	D	E	A	9	C	8	3	6	0	F	5	1	4	B

Tout recours est exclu, de même que le sont, de ce jeu, les personnels d'Elektor International Media et leur famille. Un seul gagnant par foyer.

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 pourles 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

Une offre encore décuplée

La plus vaste sélection de semi-conducteurs et composants électroniques en stock et prêts à être expédiés™

mouser.fr

MOUSER ELECTRONICS