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

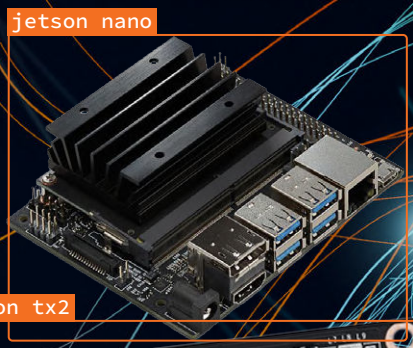
elektor 506

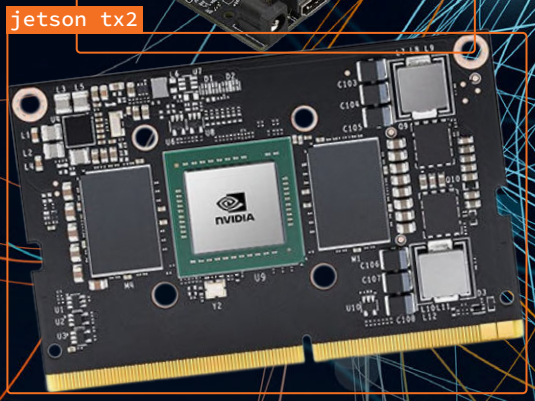
créer > partager > vendre

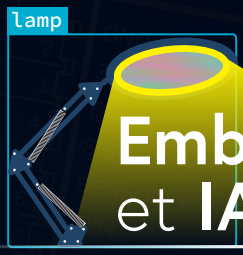
MARS & AVRIL 2024
ELEKTORMAGAZINE.FR

Carte spéciale IA

apprentissage automatique avec le Jetson Nano



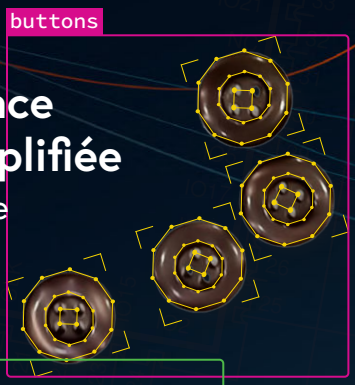




FOCUS SUR
Embarqué et IA

Reconnaissance d'images simplifiée

avec Edge Impulse

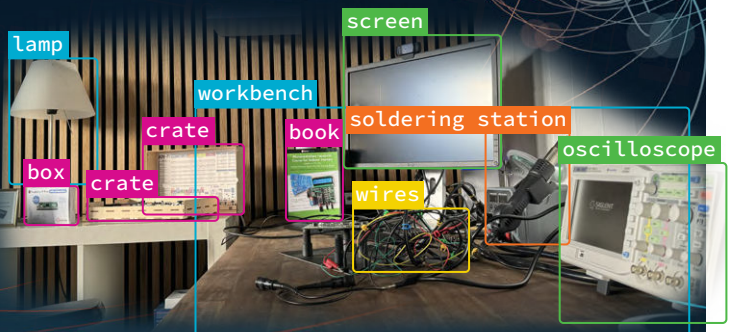


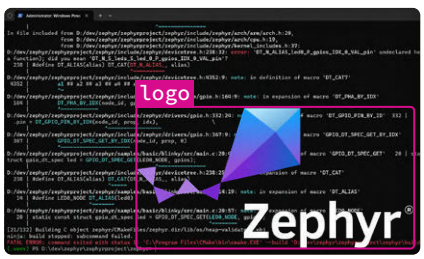
Adaptateur ESP32-RS-232

liaison sans fil pour les ports COM classiques

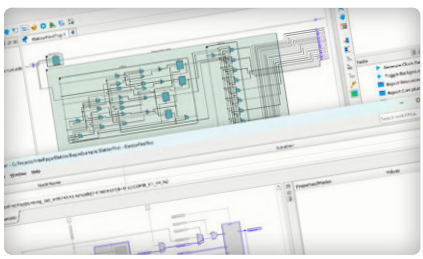
CaptureCount

détecteur et compteur d'objets basé sur le Raspberry Pi 5





Commencer avec le RTOS Zephyr
aussi puissant que difficile à maîtriser
p. 98



FPGA pour les débutants
passer de la programmation de microcontrôleurs à celle de FPGA
p. 22

Référence de tension avec Arduino Pro Mini
linéariser et calibrer les entrées analogiques
p. 14

L 19624 - 506 - F: 15,50 € - RD

NOTRE GAMME

PAR DES TECHNICIENS POUR DES TECHNICIENS

Tirer le meilleur parti de votre projet: www.reichelt.com

Uniquement le meilleur pour vous - provenant de plus de 1.500 marques

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

Robotique et apprentissage machine

La combinaison de la robotique, de l´intelligence artificielle et de l´apprentissage machine permet aux robots d´exécuter de manière autonome des tâches plus complexes et de s´adapter à de nouveaux défis. Trouvez la technologie adaptée dans notre gamme en constant développement.

Arduino Pro Nicla Vision — surveiller – analyser – détecter

Module de caméra AI avec Dual ARM CortexR M7 et M4

Nicla Vision est une caméra autonome prête à l'emploi destinée à l'analyse et au traitement des images périphériques ; elle est adaptée au suivi des actifs, à la détection d'objets et à la maintenance prédictive.

Référence:
ARD NIC VISION

99,23 (82,69)

ARM | DUAL CORE

La nouvelle génération : Go2

Désormais de série avec LiDAR ultra-grand-angle 4D

Plongez dans le monde fascinant de la robotique. Ce robot quadrupède avancé associe des technologies de pointe à une conception élégante et agile pour amener le futur de la robotique directement entre vos mains.

- Puissance moteur améliorée de 30 %
- Autonomie/capacité de batterie augmentées de 150 %
- Système de guidage latéral intelligent ISS2.0 (à partir du modèle Pro)

Référence:
QR GO2 AIR

2.640,00 (2.200,00)

DES ROBOTS PLUS INTELLIGENTS SONT CAPABLES DE PLUS !

ROBOTIQUE ET APPRENTISSAGE MACHINE LEARNING

Découvrez maintenant ▶ https://rch.lt/mlfr

NOUVEAUTÉS DU MAGASIN

LES DERNIERS ARTICLES DE TOUS LES SECTEURS EN UN COUP D'ŒIL !

Découvrez maintenant ▶ www.reichelt.com/nouveau

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

PRIX DU JOUR! Prix à la date du: 16. 2. 2024

- Excellent rapport qualité prix
- Plus de 130 000 produits sélectionnés
- Livraison fiable - depuis l'Allemagne dans le monde entier

www.reichelt.com

Assistance téléphonique: +33 9 75 18 03 04

reichelt elektronik – Tirer le meilleur parti de votre projet

COLOPHON

47ème année n° 506
mars-avril 2024
ISSN 0181-7450

N° de TVA Intracommunautaire : FR90319937454

Dépôt légal : mars 2024
CPPAP 1125 T 83713
Directeur de la publication : Donatus Akkermans

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

Pour toutes vos questions : service@elektor.fr

Devenez membre : www.elektormagazine.fr/abo

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

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

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

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

ÉDITO

Jens Nickel

rédacteur en chef d'Elektor Magazine

Fixez-vous des objectifs !

Il y a bien longtemps, dans un article, j'avais révélé qu'un but bien précis m'avait poussé à devenir un programmeur ambitieux. À cette époque, pour gérer nos articles en interne, j'ai dû apprendre à utiliser VBA pour Excel, MS Access, C#, HTML + JavaScript et bien d'autres langages. Cela m'a permis d'acquérir une foule de connaissances, précieuses aussi bien pour mon rôle d'éditeur que pour d'autres projets variés. Jamais je n'aurais atteint un tel niveau en me limitant à suivre des tutoriels ou à réaliser de simples projets pratiques. Certains professionnels, après avoir examiné mon système, ont été impressionnés. Ils avaient eux-mêmes souvent envisagé de développer leur propre système éditorial, sans jamais y parvenir.
Cette approche est tout aussi valable pour le matériel. Comme vous le savez peut-être, je me suis mis à l'électronique de façon assez détournée. Passionné par le bricolage et la programmation, je laissais jusqu'alors la conception des circuits imprimés à mes collègues. Récemment, j'ai entamé avec un ami un nouveau projet électronique bien plus ambitieux : nous avons entrepris le contrôle à distance (et la mesure à distance) d'amplificateurs audio modifiés. Cela m'a obligé à me familiariser avec KiCad, différents connecteurs, la programmation d'écrans tactiles et plusieurs fonctionnalités de GitHub. Et ce n'est certainement pas la fin de l'histoire – je prévois de publier prochainement un article à ce sujet sur elektor-labs.com.
Que veux-je dire par là ? Fixez-vous des objectifs et n'hésitez pas à vous lancer dans des projets qui vous semblent au premier abord bien trop complexes. La volonté de concrétiser une idée vous portera et vous insufflera l'enthousiasme nécessaire pour explorer de nouveaux domaines. De temps à autre, permettez-vous de mettre un point final à votre projet, même s'il paraît hors de portée et que et les progrès sont irréguliers. Le sentiment d'accomplissement qui en découle vous motivera à tout peaufiner.
Dans ce numéro, plongez sans attendre dans le monde du traitement d'images assisté par l'IA. Cela fait longtemps que vous y pensez ? Parfait ! Développer votre propre application n'est pas aussi compliqué qu'il y paraît (pages 6 et 86). Saisissez cette opportunité pour ouvrir la porte à un monde rempli de nouvelles idées !

Proposez une contribution à Elektor!

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

www.elektormagazine.com/submissions

Elektor Labs : idées et projets

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

www.elektormagazine.fr/labs

DANS CE NUMÉRO

Volume 50 – Nº 506
Mars et avril 2024

carte spéciale IA

apprentissage automatique avec le Jetson Nano 44

Rubriques

3 **Édito**

29 **STM32 Wireless Innovation Design Contest 2024**
mise à jour

52 **2024 : l'Odyssée de l'IA**
premiers essais avec TensorFlow

56 **un projet de lecteur en bref**
262 144 façons de jouer au Jeu de la Vie

62 **sur le vif**
le souffle du dragon sur mon cou

77 **démarrer en électronique...**
...en savoir plus sur les ampli-op

84 **drôle de composant**
composants piézoélectriques

112 **projet 2.0**
corrections, mises à jour et courrier des lecteurs

Articles de fond

22 **FPGA pour les débutants**
passer de la programmation de microcontrôleurs à celle de FPGA

30 **Bluetooth LE avec MAUI**
applications de contrôle pour Android et cie.

64 **faites tourner votre moteur CC**
Exemples de projets (de l'offre groupée : Elektor Motor Control Development Bundle)

80 **bibliothèques ESP recommandées**

95 **ESP32 Terminal**
un appareil portable avec un écran tactile

98 **commencer avec le RTOS Zephyr**
aussi puissant que difficile à maîtriser

Industrie

92 **des solutions à vos problèmes de développement de systèmes embarqués les plus délicats**

107 **Ethics in Electronics**
dialogue avec Alexander Gerfer, Directeur Technique chez Würth Elektronik eiSos

CaptureCount
détecteur et compteur d'objets basé sur le Raspberry Pi 5 6

Bluetooth LE avec MAUI
applications de contrôle pour Android et cie. 30

adaptateur ESP32-RS-232
liaison sans fil pour les équipements de test classiques 70

Projets

6 **CaptureCount**
détecteur et compteur d'objets basé sur le Raspberry Pi 5

14 **référence de tension avec Arduino Pro Mini**
linéariser et calibrer les entrées analogiques

38 **carte d'extension de ports**
augmentez le nombre d'E/S de votre carte de développement

44 **carte spéciale IA**
apprentissage automatique avec le Jetson Nano

70 **adaptateur ESP32-RS-232**
liaison sans fil pour les équipements de test classiques

86 **compteur d'objets intelligent**
reconnaissance d'images simplifiée avec Edge Impulse

Bientôt dans ces pages

Le numéro de mai et juin 2024
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 mesure ».

- Détecteur de présence humaine
- Surveillance du graphique ECG
- Analyseur de gain et de phase avec interface sonore
- Réparation d'équipements électronique
- Générateur de signaux DDS
- LC-Mètre en circuit
- Interface pour wattmètre (5 A / 60 V)
- Disque stroboscopique actif pour platines

Le numéro de mai – juin 2024 du magazine Elektor sera publié aux alentours du 15 mai 2024. La date d'arrivée du magazine papier chez les abonnés dépend des aléas d'acheminement.

référence de tension avec Arduino Pro Mini
linéariser et calibrer les entrées analogiques 14

FOCUS SUR

Embarqué et IA

CaptureCount

détecteur et compteur d'objets basé sur le Raspberry Pi 5

Saad Imtiaz (Elektor)

La vision artificielle est une technologie fascinante qui permet la reconnaissance et la classification d'une large gamme d'objets de notre environnement. Cet article présente un étrange projet à base de Raspberry Pi qui utilise le modèle de détection d'objet YOLO. Le projet est capable d'identifier de nombreux objets, depuis des choses courantes comme des voitures et des vélos jusqu'aux différents animaux tels que des chats, chiens ou oiseaux.

Figure 1. CaptureCount tournant sur un Raspberry Pi 5 avec le module Caméra du Raspberry Pi (Wide).

Le projet a débuté avec un but précis : créer une détection d'objets omnidirectionnelle avec un système de comptage qui reconnaît le type d'objet et qui compte les objets de chaque catégorie détectée. Le Raspberry Pi était un choix évident par ses fonctionnalités et support dans les projets d'IA. Comme c'était mon premier projet de vision artificielle, il m'a fallu un peu de temps pour apprendre le langage Python et utiliser les bons outils et bibliothèques pour développer ce projet. Comme il y a énormément de projets, de ressources [1][2] et grâce à notre cher ChatGPT, il n'a pas été difficile d'apprendre et de concevoir un tel projet.

Détection d'objet

Dans la recherche de l'élaboration de ce projet, le voyage a commencé par la recherche du bon modèle de détection d'objets, de préférence un modèle qui peut détecter et reconnaître une grande variété d'objets avec un temps d'apprentissage minimal. Mais avant d'aborder cette recherche de modèle de détection d'objets, arrêtons-nous un peu et examinons comment fonctionne la détection d'objets.

La détection d'objets est une technologie qui permet aux ordinateurs d'identifier et localiser des objets dans une image ou une vidéo. Contrairement à la reconnaissance d'image qui vous dit ce qu'il y a dans une image, la détection d'objets va plus loin en pointant exactement l'endroit où ces objets se trouvent et en les surlignant. Cette opération est souvent accomplie au travers de deux processus clés :

- Localisation de l'objet : détermine la localisation d'un objet unique dans une image. Le modèle fournit les coordonnées de la boîte entourant l'objet.
- Classification de l'objet : identifie la nature de l'objet dans la boîte à partir d'une série de catégories connues.

Les modèles avancés de détection d'objets intègrent ces deux étapes en traitant efficacement une image pour fournir à la fois la localisation et la classification de multiples objets dans cette image.

Examen des modèles de détection d'objet

La recherche du modèle idéal de détection d'objets a impliqué l'évaluation de plusieurs options, chacune avec ses avantages et ses limitations. Les modèles tels que R-CNN et ses dérivés offrent une grande précision mais avec une vitesse de traitement lente, incompatible avec des applications en temps réel. Le modèle « Single Shot Multibox Detector « (SSD) présente une alternative plus rapide mais avec, en contre-partie, une perte au niveau précision. Le modèle « Mask R-CNN « possède une détection précise, mais est trop complexe pour notre utilisation. Finalement, le modèle YOLO (*You Only Look Once*, vous ne regardez qu'une seule fois) a été choisi pour son équilibre optimal entre la vitesse, l'efficacité et une précision raisonnable. Sa capacité à traiter les images en temps réel et sa simplicité, alliées à

un bon soutien de la communauté IA, a fait de YOLO le meilleur choix pour l'objectif notre projet.
La sélection du modèle YOLO a été un point crucial dans ce projet. YOLO par Joseph Redmon [3], est reconnu par sa capacité à effectuer une détection en temps réel — une caractéristique cruciale pour tout système de détection. Ce qui m'a séduit dans YOLO était son approche particulière dans la détection d'objets.
Contrairement aux méthodes traditionnelles qui traitent une image en plusieurs étapes, YOLO le fait en une seule passe. Ceci accélère le processus et améliore la capacité du modèle à généraliser à partir de son apprentissage, le rendant plus efficace dans la détection d'objets dans les conditions diverses et imprévisibles du monde réel. Cette fonctionnalité a été fondamentale pour ce projet qui visait à la reconnaissance d'un grand nombre d'objets.

Intégration du logiciel matériel et mise en œuvre

Le projet nécessitait le Raspberry Pi 5, connu pour sa puissance de traitement améliorée et un module caméra compatible. Le procédé d'intégration comprenait la mise en place de l'OS du Raspberry Pi, le branchement de la caméra et l'installation de YOLO. Pour démarrer avec le Raspberry Pi, Il faut d'abord installer l'OS sur sa carte microSD. Ceci se fait simplement en téléchargeant l'imageur Raspberry Pi depuis le site officiel du Raspberry Pi [4]. Puis, il faut lancer l'imageur, sélectionner le bon périphérique, sélectionner le dernier OS Raspberry Pi (64 bit), sélectionner la carte microSD et enfin cliquer simplement sur le bouton *Next*.
En quelques minutes, l'OS Raspberry Pi sera installé sur la carte microSD. Ensuite, insérez la carte SD dans la fente pour carte microSD du Raspberry Pi et mettez-le en marche. Vous aurez aussi besoin d'un moniteur, d'un câble Mini HDMI vers HDMI, d'un clavier et d'une souris pour interagir avec le Raspberry Pi la première fois, mais une fois que le VNC Server ou SSH sera activé, le Raspberry Pi pourra être contrôlé à distance par votre ordinateur par wifi ou par la connexion Ethernet.
Il est temps maintenant d'installer les bibliothèques requises pour ce projet. L'installation des bibliothèques est basique ; cela se fait via le Terminal. Avant d'installer de nouvelles bibliothèques, il est recommandé de mettre à jour les paquets du système en tapant :

```
sudo apt-get update && sudo apt-get upgrade
```

Ceci peut prendre un certain temps et ensuite seulement, nous pouvons installer les bibliothèques nécessaires au projet, en tapant les commandes suivantes :

› Installation des paquets Image I/O

```
sudo apt-get install libjpeg-dev libtiff5-dev libjasper-dev libpng12-dev
```

› Installation des paquets Video I/O et les bibliothèques de développement GTK

```
sudo apt-get install libavcodec-dev libavformat-dev libswscale-dev libv4l-dev
sudo apt-get install libxvidcore-dev libx264-dev
sudo apt-get install libgtk2.0-dev
```

› Dépendances supplémentaires pour OpenCV

```
sudo apt-get install libatlas-base-dev gfortran
```

› Installation de *pip* et *pipx* (outil de gestion des paquets)

```
sudo apt-get install python3-pip
sudo apt install pipx -y
```

› Installation de Numpy, Pandas et Open CV

```
pip install numpy
pip install pandas
sudo apt install python3-opencv
```

Après l'installation de toutes les bibliothèques, il est important de s'assurer d'une communication sans faille entre la partie matérielle et la partie logicielle pour permettre au Raspberry Pi de traiter efficacement les entrées en direct de la caméra. Pour conserver un encombrement minimal, la caméra du Raspberry Pi a été directement branchée sur le port MIPI du Raspberry Pi 5. Maintenant que tout est installé, parlons du code du projet.

Plongeons-nous dans le code

Le cœur de ce projet réside dans son code en Python, qui est disponible sur GitHub [4]. Le script commence par l'importation des bibliothèques essentielles comme OpenCV pour le traitement de l'image, Pandas pour la gestion des données et Numpy pour les opérations numériques.

```
import cv2
import pandas as pd
import numpy as np
import subprocess
import os
from datetime import datetime
```

Au lancement, le script importe les bibliothèques essentielles. *cv2* (OpenCV) est crucial pour les tâches de traitement d'image. *pandas* et *numpy* se chargent respectivement de la manipulation des données et des calculs numériques. *subprocess* et *os* sont des bibliothèques Python standard pour interagir avec le système, comme faire tourner des commandes externes et gérer les chemins d'accès aux fichiers. *datetime* est utilisé pour la détection de l'horodatage.

```
model = './yolo/yolov3.weights'
config = './yolo/yolov3.cfg'
net = cv2.dnn.readNetFromDarknet(config, model)
```

Le script spécifie les chemins vers le fichier de configuration et des valeurs de pré-configuration de YOLO, puis les charge en utilisant le module du réseau neuronal profond OpenCV (DNN). Cette étape initialise le modèle YOLO pour la détection d'objets.

```
classes = []
with open("./yolo/coco.names", "r") as f:
    classes = [line.strip() for line in f.readlines()]
```

Le fichier *coco.names*, contenant le nom des objets que YOLO peut détecter, est lu ligne par ligne pour créer une liste de noms de classe.

Plusieurs fonctions sont définies pour optimiser le processus de détection. Jetons un coup d'œil dans le **listage 1** pour voir les fonctions et les paramètres de départ de la boucle principale.
La fonction `get_output_layers(net)` extrait les noms des couches de sortie du réseau YOLO. L'architecture YOLO a plusieurs couches de sortie et cette fonction aide à les identifier pour traiter les détections.
`draw_bounding_box(…)` dessine les rectangles (*bounding boxes*) autour des objets détectés et les annote avec le nom de l'objet et l'indice de fiabilité.
`calculate_centroid(x, y, w, h)` calcule le point central du rectangle entourant l'objet détecté, point critique pour suivre les objets d'une image à l'autre.

```
def capture_image(image_path):
    subprocess.run(["libcamera-still", "-o", image_path,
"--width", "1920", "--height", "1080", "-n", "-t", "1"],
stdout=subprocess.DEVNULL, stderr=subprocess.STDOUT)
```

Cette fonction utilise le module `subprocess` pour faire tourner *libcamera-still*, un outil de commande en ligne de l'OS du Raspberry Pi, qui capture une image de la caméra et la sauvegarde dans le chemin spécifié. La résolution est 1,920×1,080. Une résolution plus basse peut aussi être utilisée pour minimiser la charge du CPU et également du GPU. Pour y arriver, le fragment du code ci-dessus peut être ajusté comme suit :

```
def capture_image(image_path):
    subprocess.run(["libcamera-still", "-o", image_path,
"--width", "1280", "--height", "720", "-n", "-t", "1"],
stdout=subprocess.DEVNULL, stderr=subprocess.STDOUT)
```

Avant d'entrer dans la boucle principale de détection, le script initialise un `DataFrame` Pandas pour ajouter aux détections, un horodatage et un dictionnaire pour suivre la trace du comptage de chaque type d'objet. `centroid_tracking` est utilisé pour suivre le mouvement des objets.

```
data_frame = pd.DataFrame(columns=['Timestamp', 'Type',
'Count'])
object_counts = {cls: 0 for cls in classes}
centroid_tracking = {}
```

La portion suivante du code (voir **listage 2**) vérifie si l'objet détecté est nouveau ou s'il a déjà été détecté auparavant. Elle utilise `centroid_tracking` pour le déterminer. Si un objet est identifié comme nouveau, elle incrémente le comptage de ce type d'objet, ajoute l'heure de détection et met à jour le DataFrame Pandas. Ce `DataFrame` peut être exporté plus tard au format *.csv* pour une analyse ultérieure.

Configuration de la boucle principale de détection

La boucle principale de détection du Raspberry Pi 5 et du système YOLO est la pièce maîtresse par sa fonctionnalité. Elle débute par la capture d'une image de la caméra du Raspberry Pi, puis par la conversion dans un format conforme au traitement par YOLO. Le système traite ensuite cette image au travers de YOLO, en extrayant les informations vitales telles que l'identification de classe, l'indice de confiance et les coordonnées du rectangle de sélection.
Pour affiner les détections, un filtre « Non-Maximum Suppression « (NMS) est appliqué, éliminant les détections redondantes et moins pertinentes. Le système dessine ensuite des rectangles de sélection autour des objets détectés et, si de nouveaux objets sont détectés, sauvegarde ces images. Tout au long de ce traitement, le système met à jour les structures de données de suivi et d'enregistrement pour maintenir un enregistrement continu des détections. Cette boucle est primordiale dans les capacités de détection en temps réel du Raspberry Pi 5 et du système YOLO. Voir **listage 3** pour la boucle principale du code complet.
Après le traitement des détections, le script compile les données dans un DataFrame et le sauvegarde dans un fichier *.csv*. Ceci permet un examen détaillé de chaque détection.

CaptureCount en action

Lorsqu'il est en action, le programme CaptureCount fait preuve d'une bonne détection des objets, bien qu'accompagné de quelques travers intrigants. Le Raspberry Pi était monté sur un tripode de caméra et la caméra visait au travers d'une fenêtre comme on peut le voir sur la **figure 2**. Comme vous pouvez le constater, je vis dans un quartier urbain de la ville et les seules choses que CaptureCount détecte sont les voitures, les camions, les passants et les motos. Il aurait été préférable d'être à la campagne pour pouvoir capturer un peu de vie sauvage.
L'étonnante performance était caractérisée par un admirable taux de précision de 80 % dans l'identification et la catégorisation des différents objets. Ce niveau de précision était particulièrement remarquable en

Figure 2. Point de vue de la caméra.

Figure 3. Deux voitures détectées par CaptureCount.

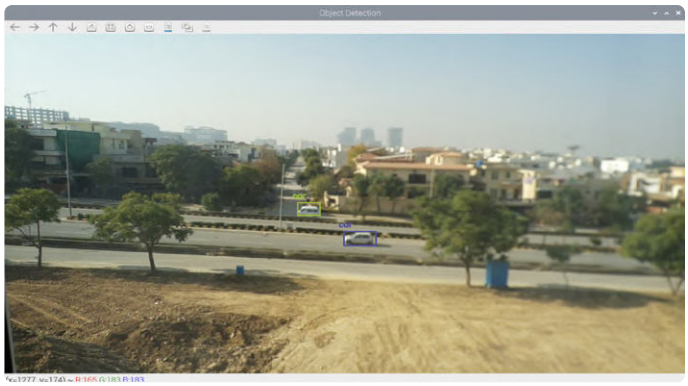


Figure 4. Image des voitures détectées par le système.

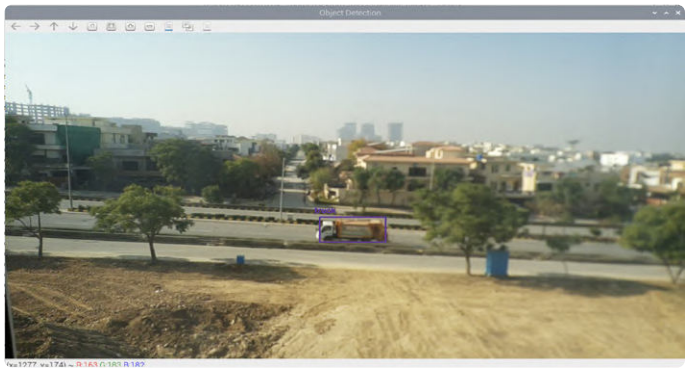


Figure 5. Camion détecté par le système.

considérant l'emplacement de la caméra qui était relativement éloignée des objets, ajouté à l'utilisation d'un objectif grand-angle. Dans les **figure 3**, **4** et **5,** on peut voir les images des objets détectés. Dans les **figure 6** et **figure 7**, le rapport complet est montré dans un fichier *.csv* qui indique quand l'objet a été détecté et combien d'objets de la même catégorie ont été détectés pendant un intervalle de temps donné.

Problèmes rencontrés et observations particulières

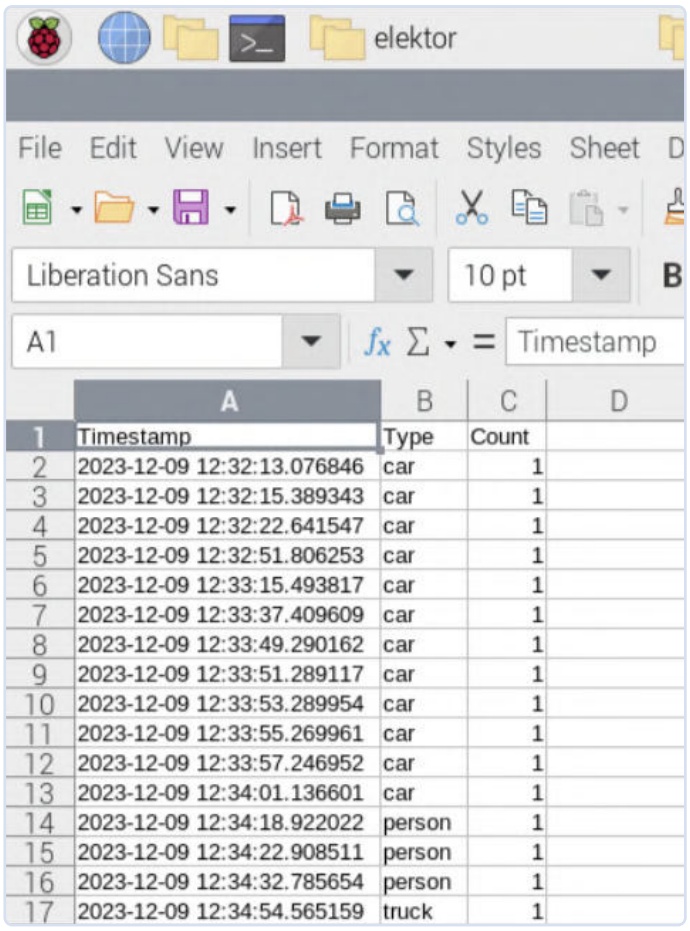
Malgré son succès, le système a rencontré quelques problèmes intéressants :

- Détection sélective dans les réglages dynamiques : certains cas ont été notés où le système a identifié une personne sur une moto mais a omis de détecter la moto elle-même. Cette détection sélective a mis en avant le problème de pouvoir distinguer avec précision des objets étroitement liés, en particulier lorsqu'ils sont en mouvement.
- Mauvaise classification sporadique : il y a eu quelques rares occasions où des piétons ont été pris pour des vélos. Cette mauvaise classification met l'accent sur la complexité impliquée dans la classification des objets, en particulier lorsque les objets partagent des caractéristiques spatiales similaires.

Et le futur de CaptureCount ?

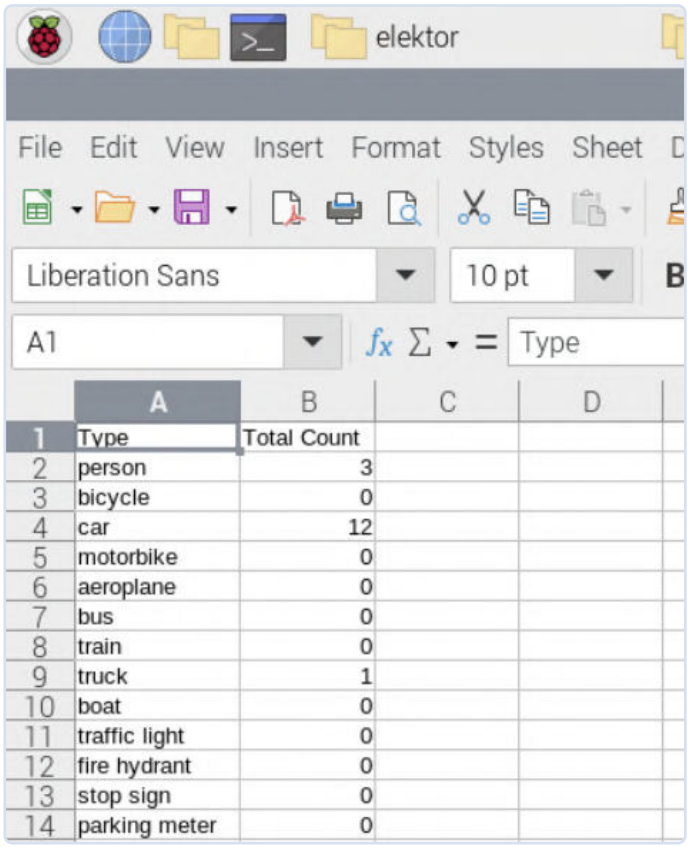
En résumé, ce projet, exploitant le Raspberry Pi 5 et YOLO, a démontré une bonne performance et a ouvert des portes vers diverses applications et améliorations. Les anomalies de détection du système et les mauvaises classifications ponctuelles, non seulement soulignent les domaines pour de futures améliorations, mais aussi ouvrent des opportunités vers plus de recherches.

Des améliorations futures peuvent inclure l'intégration de servomoteurs pour le suivi d'objets déterminés et une connectivité IdO pour des réponses automatiques. Juste un exemple : faire clignoter une



	A	B	C	D
1	Timestamp	Type	Count	
2	2023-12-09 12:32:13.076846	car	1	
3	2023-12-09 12:32:15.389343	car	1	
4	2023-12-09 12:32:22.641547	car	1	
5	2023-12-09 12:32:51.806253	car	1	
6	2023-12-09 12:33:15.493817	car	1	
7	2023-12-09 12:33:37.409609	car	1	
8	2023-12-09 12:33:49.290162	car	1	
9	2023-12-09 12:33:51.289117	car	1	
10	2023-12-09 12:33:53.289954	car	1	
11	2023-12-09 12:33:55.269961	car	1	
12	2023-12-09 12:33:57.246952	car	1	
13	2023-12-09 12:34:01.136601	car	1	
14	2023-12-09 12:34:18.922022	person	1	
15	2023-12-09 12:34:22.908511	person	1	
16	2023-12-09 12:34:32.785654	person	1	
17	2023-12-09 12:34:54.565159	truck	1	

Figure 6. Capture d'écran du fichier .csv montrant les objets détectés avec respectivement leurs horodatages.



	A	B	C	D
1	Type	Total Count		
2	person	3		
3	bicycle	0		
4	car	12		
5	motorbike	0		
6	aeroplane	0		
7	bus	0		
8	train	0		
9	truck	1		
10	boat	0		
11	traffic light	0		
12	fire hydrant	0		
13	stop sign	0		
14	parking meter	0		

Figure 7. Capture d'écran du fichier .csv montrant la somme totale des objets détectés dans une catégorie donnée.

LED RGB pour un type spécifique d'objet. Pour vous donner quelques astuces pour enrichir votre projet avec cette idée ou tout autre contrôle de matériel, voir **listage 4.** Cette fonction allume la LED rouge pour une voiture, verte pour une personne et rouge et vert (donnant du jaune) pour un camion. Après un délai, toutes les LED sont éteintes. De plus, des modifications peuvent être effectuées pour des applications plus vastes dans la surveillance avancée, la gestion des foules et du trafic, la surveillance de la faune, les statistiques de vente et la santé. La capacité en temps réel du système et la détection d'objets précis posent les fondations de solutions innovantes pour diverses industries en montrant les possibilités étendues dans les domaines de l'IA et de la vision par ordinateur.

VF : Chris Elsass — 230749-04

Questions ou commentaires ?

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

À propos de l'Auteur

Saad Imtiaz (Ingénieur Expérimenté, Elektor) est un ingénieur en mécatronique avec une expérience dans les systèmes embarqués, les systèmes mécatroniques et le développement de produits. Il a collaboré avec de nombreuses entreprises, allant des startup jusqu'aux entreprises mondiales, dans le développement et le prototypage. Saad a aussi passé du temps dans l'industrie aéronautique et a dirigé une startup technologique. Chez Elektor, il dirige des développement de projets dans les domaines matériel et logiciel.

Produits

- **Raspberry Pi 5 Ultimate Starter Kit (8 GB)**
  www.elektor.fr/20721
- **Raspberry Pi High Quality Camera Module**
  www.elektor.fr/19279
- **Raspberry Pi Camera Module 3 Wide**
  www.elektor.fr/20364

LIENS

[1] Implementation of YOLOv3: Simplified: https://analyticsvidhya.com/blog/2021/06/implementation-of-yolov3-simplified
[2] YOLOv3 code explained: https://pylessons.com/YOLOv3-code-explanation
[3] YOLO: Real-Time Object Detection: https://pjreddie.com/darknet/yolo
[4] Dépôt Github de CaptureCount : https://github.com/ElektorLabs/CaptureCount

Listage 1. Fonctions de détection

```
import cv2
import pandas as pd
import numpy as np
import subprocess
import os
from datetime import datetime

# Load pre-trained model and configuration
model = './yolo/yolov3.weights'  # Update this path to your model's path
config = './yolo/yolov3.cfg'     # Update this path to your config file's path
net = cv2.dnn.readNetFromDarknet(config, model)

# Load class names
classes = []
with open("./yolo/coco.names", "r") as f:  # Update to the path of your coco.names file
    classes = [line.strip() for line in f.readlines()]

# Function to get output layers
def get_output_layers(net):
    layer_names = net.getLayerNames()
    return [layer_names[i - 1] for i in net.getUnconnectedOutLayers().flatten()]
```

```
# Function to draw bounding box
def draw_bounding_box(img, class_id, confidence, x, y, x_plus_w, y_plus_h):
    label = str(classes[class_id])
    color = np.random.uniform(0, 255, size=(3,))
    cv2.rectangle(img, (x, y), (x_plus_w, y_plus_h), color, 2)
    cv2.putText(img, label, (x-10, y-10), cv2.FONT_HERSHEY_SIMPLEX, 0.5, color, 2)

# Function to calculate centroid of a bounding box
def calculate_centroid(x, y, w, h):
    return (int(x + w/2), int(y + h/2))

# Function to capture image using libcamera-still
def capture_image(image_path):
    subprocess.run(["libcamera-still", "-o", image_path, "--width", "1920", "--height", "1080", "-n", "-t",
                                                "1"], stdout=subprocess.DEVNULL, stderr=subprocess.STDOUT)

# Initialize a DataFrame to store counts
data_frame = pd.DataFrame(columns=['Timestamp', 'Type', 'Count'])
object_counts =  # Object count per category
centroid_tracking = {}  # Tracks centroids of detected objects

# Ensure output directory exists
output_dir = "output"
if not os.path.exists(output_dir):
    os.makedirs(output_dir)
```

Listage 2. Suivi du point central — détecte si un objet est nouveau ou s'il a déjà été détecté

```
# Check if this object was already detected
if class_id not in centroid_tracking or not any(np.linalg.norm(np.array(centroid) - np.array(old_centroid))
                                                < 50 for old_centroid in centroid_tracking[class_id]):
    object_counts[classes[class_id]] += 1
    centroid_tracking.setdefault(class_id, []).append(centroid)
    new_row = pd.DataFrame([{'Timestamp': datetime.now(), 'Type': classes[class_id], 'Count': 1}])
    data_frame = pd.concat([data_frame, new_row], ignore_index=True)
```

Listage 3. Boucle main

```
# Main loop
image_path = "temp.jpg"
object_id = 0  # Unique identifier for each object

try:
    while True:
        capture_image(image_path)
        frame = cv2.imread(image_path)
        if frame is None:
            continue

        Width = frame.shape[1]
        Height = frame.shape[0]
        scale = 0.00392
```

(Suite à la page suivante)

```
        # Create a blob and pass it through the model
        blob = cv2.dnn.blobFromImage(frame, scale, (416, 416), (0, 0, 0), True, crop=False)
        net.setInput(blob)
        outs = net.forward(get_output_layers(net))

        # Process the outputs
        class_ids = []
        confidences = []
        boxes = []
        centroids = []
        conf_threshold = 0.5
        nms_threshold = 0.4

        for out in outs:
            for detection in out:
                scores = detection[5:]
                class_id = np.argmax(scores)
                confidence = scores[class_id]
                if confidence > conf_threshold:
                    center_x = int(detection[0] * Width)
                    center_y = int(detection[1] * Height)
                    w = int(detection[2] * Width)
                    h = int(detection[3] * Height)
                    x = center_x - w / 2
                    y = center_y - h / 2
                    class_ids.append(class_id)
                    confidences.append(float(confidence))
                    boxes.append([x, y, w, h])
                    centroids.append(calculate_centroid(x, y, w, h))

        indices = cv2.dnn.NMSBoxes(boxes, confidences, conf_threshold, nms_threshold)

        for i in indices:
            box = boxes[i]
            x, y, w, h = box
            x, y, w, h = int(x), int(y), int(w), int(h)  # Ensure the values are integers

            centroid = centroids[i]
            class_id = class_ids[i]

            # Check if this object was already detected
            if class_id not in centroid_tracking or not any(np.linalg.norm(np.array(centroid) -
                            np.array(old_centroid)) < 50 for old_centroid in centroid_tracking[class_id]):
                object_counts[classes[class_id]] += 1
                centroid_tracking.setdefault(class_id, []).append(centroid)


                # Update data_frame using pandas.concat
                new_row = pd.DataFrame([{'Timestamp': datetime.now(),'Type': classes[class_id],'Count': 1}])
                data_frame = pd.concat([data_frame, new_row], ignore_index=True)

                # Save the entire frame when an object is detected
                frame_filename = os.path.join(output_dir, f"frame__.jpg")
                cv2.imwrite(frame_filename, frame)
                object_id += 1

            draw_bounding_box(frame, class_id, confidences[i], round(x), round(y), round(x+w), round(y+h))

        # Display the frame
        cv2.imshow("Object Detection", frame)

        # Break loop with 'q' key
```

```
        if cv2.waitKey(1) & 0xFF == ord('q'):
            break
finally:
    cv2.destroyAllWindows()
    if os.path.exists(image_path):
        os.remove(image_path)

# Print and save the total count of objects detected per category
total_counts = pd.DataFrame(object_counts.items(), columns=['Type', 'Total Count'])
print(total_counts)
total_counts.to_csv("total_object_counts.csv", index=False)

# Save detailed counts to CSV
data_frame.to_csv("object_counts.csv", index=False)

# Write the total count to a text log
with open("total_counts_log.txt", "w") as log_file:
    log_file.write(str(total_counts))
```

Listage4. Clignotement d'une LED RGB pour les objets détectés

```
#include these libraries in the shared code
import RPi.GPIO as GPIO
import time

# GPIO pin setup
RED_PIN, GREEN_PIN, BLUE_PIN = 17, 27, 22

# Update with your GPIO pin numbers

# add this part in the initial part of the code
GPIO.setmode(GPIO.BCM)
GPIO.setup([RED_PIN, GREEN_PIN, BLUE_PIN], GPIO.OUT)

def blink_rgb_led(object_type):
    GPIO.output(RED_PIN, object_type == 'car' or object_type == 'truck')
    GPIO.output(GREEN_PIN, object_type == 'person' or object_type == 'truck')
    GPIO.output(BLUE_PIN, False)
    time.sleep(1)
    GPIO.output([RED_PIN, GREEN_PIN, BLUE_PIN], False)

# Usage of this function:

# ... [Previous code] ...
for out in outs:
    for detection in out:
# ... [add the following lines to the code to this 'for' loop]
        scores = detection[5:]
        class_id = np.argmax(scores)
        confidence = scores[class_id]
        if confidence > conf_threshold:
            detected_object = classes[class_id]
             blink_rgb_led(detected_object) # Blink the LED based on the detected object

# ... [Rest of your detection and saving logic] ...
# ... [Remaining code] ...
```

référence de tension avec Arduino Pro Mini

linéariser et calibrer les entrées analogiques

Giovanni Carrera (Italie)

Si vous utilisez les entrées analogiques de vos microcontrôleurs préférés, vous aurez tôt ou tard besoin de vérifier leur linéarité et leur précision sur toute la plage des valeurs d'entrée. Ce calibrateur de tension précis et compact basé sur Arduino, capable de générer des tensions de référence stables et précises dans la plage de 0 à 5 V, résoudra ce problème.

Vous vous demandez peut-être ce qu'est un calibrateur de tension et à quoi sert-il ? Un calibrateur de tension est un outil qui délivre des tensions connues avec une grande précision et qui est utilisé pour calibrer ou vérifier la classe des voltmètres, des CAN et CNA, des systèmes d'acquisition, etc.

Lors de mesures de tension effectuées avec des microcontrôleurs, il est souvent nécessaire de connaître la constante de conversion ou de vérifier la linéarité du convertisseur analogique-numérique alimenté. Si on se contente d'une précision autour de 5 % il suffit de diviser la tension V_{ref} par le nombre maximum que l'on obtient en sortie, par exemple pour l'Arduino UNO c'est 2^{10} - 1 = 1023. Si nous utilisons la tension d'alimentation (5 V) comme V_{ref}, celle-ci peut varier selon que nous alimentons la carte par USB ou par une autre source d'alimentation et est également bruyante. Si nous utilisons la tension V_{ref} interne (cas préférable), par exemple 1.1 V, nous ne connaissons pas exactement sa valeur. D'autres microcontrôleurs tels que l'ESP32 n'ont même pas d'entrées analogiques fournissant une tension de 0 V, mais de 80 à 150 mV, et la pleine échelle est entre 950 et 1100 mV.

Pour pallier ces inconvénients, il faut procéder à un calibrage, c'est-à-dire utiliser une source de tension stable et peu bruyante ainsi qu'un voltmètre numérique d'une précision supérieure à celle du convertisseur du microcontrôleur. Il faut ensuite effectuer plusieurs mesures de différentes valeurs dans la plage d'entrée du CAN, noter les valeurs sur une feuille de calcul de Excel et procéder à une régression linéaire pour obtenir la pente et l'ordonnée à l'origine. Nous aurons besoin de ces valeurs pour obtenir une sortie en V ou en mV. Ce projet combine : un générateur de tension très stable et peu bruyant et un voltmètre numérique très précis avec une plage de mesure de 0 à 5 V et une résolution d'environ 0,2 mV.

La source de tension

Pour avoir une tension de référence stable, c'est-à-dire à faible dérive en température, il ne suffit pas d'utiliser une diode Zener ou un régulateur de tension ordinaire, mais un composant conçu pour fournir une tension de sortie qui reste aussi constante que possible au cours du temps, lorsque la tension d'alimentation, et la température varient. Pour ce projet, nous avons utilisé un LM236H-2.5, dont la dérive en température n'est que de 3.5 mV entre -25 et +85°C lorsqu'il est alimenté par un courant inverse de 1 mA. Si son comportement était linéaire, il y aurait une dérive d'environ 32 µV / °C (environ 12,8 ppm / °C). En maintenant la température et le courant constants, la variation de tension est d'environ 20 ppm sur une période de 1 000 heures.

Si vous souhaitez obtenir des valeurs légèrement plus élevées, vous pouvez utiliser un AD680, et apporter quelques modifications au circuit. Si, au contraire, vous n'avez besoin que de puces aux caractéristiques inférieures, vous pouvez remplacer ce circuit intégré par un TL431, plus répandu, sans avoir à modifier le circuit. La **figure 1** montre le schéma du circuit. Les deux diodes D2 et D3 sont utilisées pour réduire encore la dérive thermique, et le trimmer Rp1 doit être calibré pour avoir une tension de sortie de 2,49 V, la valeur avec laquelle il atteint les meilleures caractéristiques. L'amplificateur opérationnel (nous reviendrons plus loin sur sa fonction) utilisé est un MCP6002,

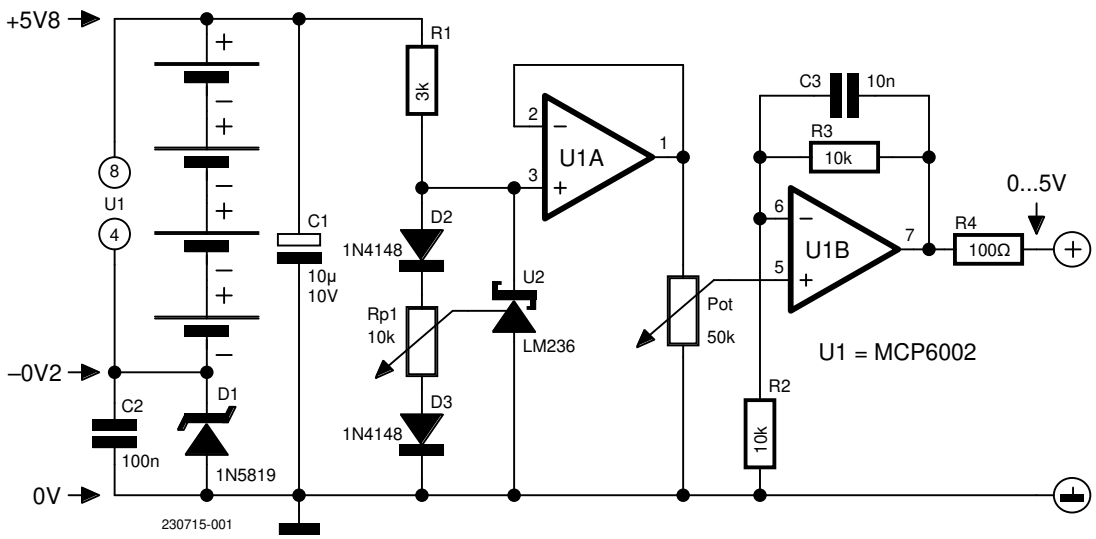


Figure 1. Schéma de la source de tension variable.

un ampli-op double rail à rail, capable de fonctionner à des niveaux de tension très proches de ceux des rails, contrairement aux ampli-op ordinaires qui saturent à quelques volts de différence contre quelques dizaines de mV de rail à rail.

La dérive en température de cette puce est très faible : ±2 µV/C° et l'offset est de quelques mV. Pour se rapprocher encore plus de zéro, nous avons utilisé la diode schottky D1. Avec ces astuces, la tension négative des ampli-op est inférieure à zéro d'environ 100...200 mV (tension directe de ces diodes), et la tension minimale de sortie n'est influencée que par l'offset des ampli-op. Comme le montre la figure, le potentiel de masse ou GND ne correspond pas au pôle négatif des piles mais à l'anode de la diode D1, ce qui explique la nécessité d'une entrée différentielle du DVM.

Il est possible d'utiliser un LM358 ordinaire à la place du MCP6002. L'ampli-op U1A joue le rôle de séparateur de haute impédance d'entrée, et à sa sortie le potentiomètre qui fait varier le potentiel entre zéro et 2,5 V est connecté. Le potentiomètre utilisé est un 10-tours ; la valeur n'est pas très importante ; elle peut varier entre 5 kΩ et 100 kΩ. Le second ampli-op U1B fonctionne comme un amplificateur non inverseur avec un gain égal à

$G = (1+R3/R2) = 2$

La sortie sera donc comprise entre environ 0 V et 5 V. Le condensateur C3 sert de filtre passe-bas pour réduire le bruit thermique des semi-conducteurs. La résistance R4 sert à limiter le courant de sortie en cas de court-circuit accidentel. Une résistance de sortie de 100 Ω ne pose aucun problème car elle est négligeable par rapport aux résistances d'entrée des systèmes à calibrer. L'alimentation est sur batterie pour éviter le bruit du secteur, et il faut éviter à tout prix une alimentation à découpage qui présente un bruit de sortie de quelques centaines de kHz avec des pics allant jusqu'à 100 mV.

Le voltmètre numérique haute précision

Pour construire un calibrateur, il faut avoir un voltmètre numérique dont la précision est supérieure de plusieurs ordres de grandeurs à celle du système que nous souhaitons calibrer, et dont l'exactitude est suffisante. Le DVM (Digital Volt Meter) proposé est doté d'un convertisseur A/N de 16 bits. Pour les valeurs positives seulement, il y a 215 = 32,768 niveaux à partir de zéro, avec un nombre maximum de pleine échelle de 32,767. Dans ce système, le CAN a été programmé à une pleine échelle de 6144 mV, la résolution est donc de 6 144 / 32 767 = 0,1875 mV. Le convertisseur n'accepte pas d'entrées avec des tensions supérieures de 0,3 V à la tension d'alimentation Vcc, soit 5,3 V, et la source de tension est conçue pour fournir une valeur maximale d'environ 5 V. Le nombre maximal correspondant à 5 V est donc 5 000 / 0,1875 = 26 666.

Il s'agit d'une valeur très élevée par rapport à celle d'un DVM normal à 3 1/2 chiffres, qui est de 1 999. Avec l'ajout d'un interrupteur, on

Liste des composants de la source de tension

Résistances

R1 = 3 kΩ, 1%, ¼ W
R2, R3 = 10 kΩ, 1%, ¼ W
R4 = 100 Ω, 5% 3 W, bobiné
Rp1 = 10 kΩ, trimmer multi-tour
Pot = 50 kΩ, potentiomètre, linéaire, 10 tours

Condensateurs

C1 = 10 µF, 25V, tantale
C2 = 100 nF, céramique
C3 = 10 nF, céramique

Semi-conducteurs

U1 = MCP6002, ampli-op
U2 = LM236H-2.5, référence de tension 2,49 V
D1 = 1N5819, diode schottky
D2, D3 = 1N4148, diode

Divers

SW1 = interrupteur, SP
B1...B4 = 4 x piles alcalines AA, 1,5 V avec porte-piles

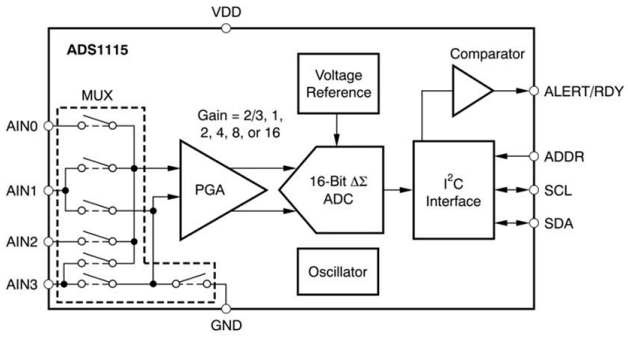


Figure 2. Principaux composants du convertisseur ADS1115. (Source : https://ti.com/lit/ds/symlink/ads1115.pdf)

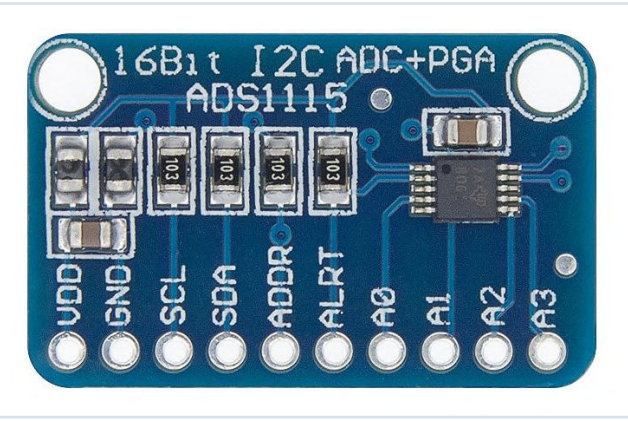


Figure 3. Module ADS1115.

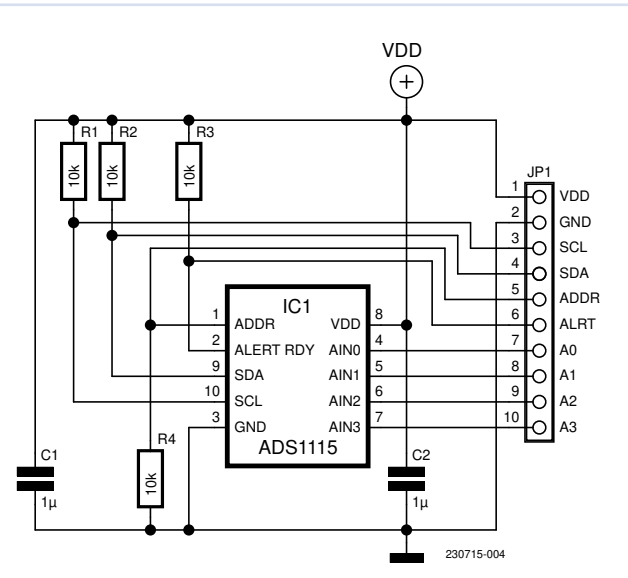


Figure 4. Schéma du module ADS1115.

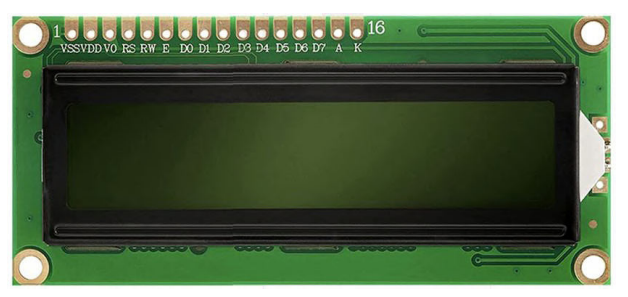


Figure 5. LCD 16×2.

peut débrancher le voltmètre pour l'utiliser séparément. Ce qui ne fonctionne pas, c'est de mettre un diviseur d'entrée à haute impédance pour mesurer des tensions plus élevées, quelque chose qui avait été essayé pour d'autres applications et qui créait des problèmes de bruit qui faisaient que même deux chiffres variaient avec une tension d'entrée stable.

Le convertisseur ADS1115

Le cœur du DVM est la puce ADS1115, un convertisseur A/N à 16 bits Delta-Sigma avec des caractéristiques de précision et de fiabilité élevées. Il peut avoir jusqu'à quatre entrées. Sur les 16 bits, le bit de poids fort est réservé au signe ; en fait, la puce peut mesurer des valeurs négatives, surtout lorsqu'elle est configurée en mode différentiel, mais avec certaines limitations liées à la tension d'alimentation. Cette puce intéressante, fabriquée par Texas Instruments, présente des caractéristiques remarquables :

- Une résolution de 16 bits, comparée aux 10 bits de l'Arduino.
- Quatre voies simples (référencées à la masse) ou deux voies différentielles.
- Taux d'échantillonnage de 8 à 860 échantillons/seconde.
- Utilisation d'un amplificateur programmable (PGA) permettant de mesurer même de faibles tensions.
- Source de référence interne à faible dérive en température.
- Possibilité d'être alimenté par des tensions de 2 à 5,5 V avec une consommation de seulement 150 µA, encore réduite en mode single-shot.
- Interface I²C avec plusieurs adresses sélectionnables.
- Fonction d'alerte (ALERT/RDY) pour surveiller efficacement les tensions. Elle peut réduire la consommation d'énergie, réveiller et activer le microcontrôleur lorsque certains événements se produisent, ou générer des interruptions.

La **figure 2** montre le schéma fonctionnel de la puce ADS1115, qui offre deux modes de conversion :

- **Conversion continue :** l'ADS1115 effectue des conversions en continu. Une fois qu'une conversion est terminée, l'ADS1115 entre le résultat dans le registre de conversion et commence immédiatement une autre conversion.
- **Conversion single-shot :** l'ADS1115 attend que le bit OS soit mis à l'état haut. Une fois affirmé, le bit est mis à 0, indiquant qu'une conversion est en cours. Une fois que les données de conversion sont prêtes, le bit OS est réaffirmé et l'appareil s'éteint. L'écriture d'un 1 sur le bit OS pendant une conversion n'a aucun effet.

Dans ce projet, nous utiliserons le mode single-shot avec une conversion toutes les demi-secondes.

Module ADS1115

Des modules similaires à celui de la **figure 3**, utilisé dans le projet, sont disponibles sur le marché. Ce module comporte quelques autres composants, comme le montre la **figure 4**. Pour utiliser cette puce avec l'EDI Arduino, nous avons utilisé la bibliothèque *ADS1115_WE* de Wolfgang Ewald, que vous pouvez télécharger à partir de l'EDI lui-même (Gestionnaire de bibliothèques) ou sur le web [1]. Elle offre

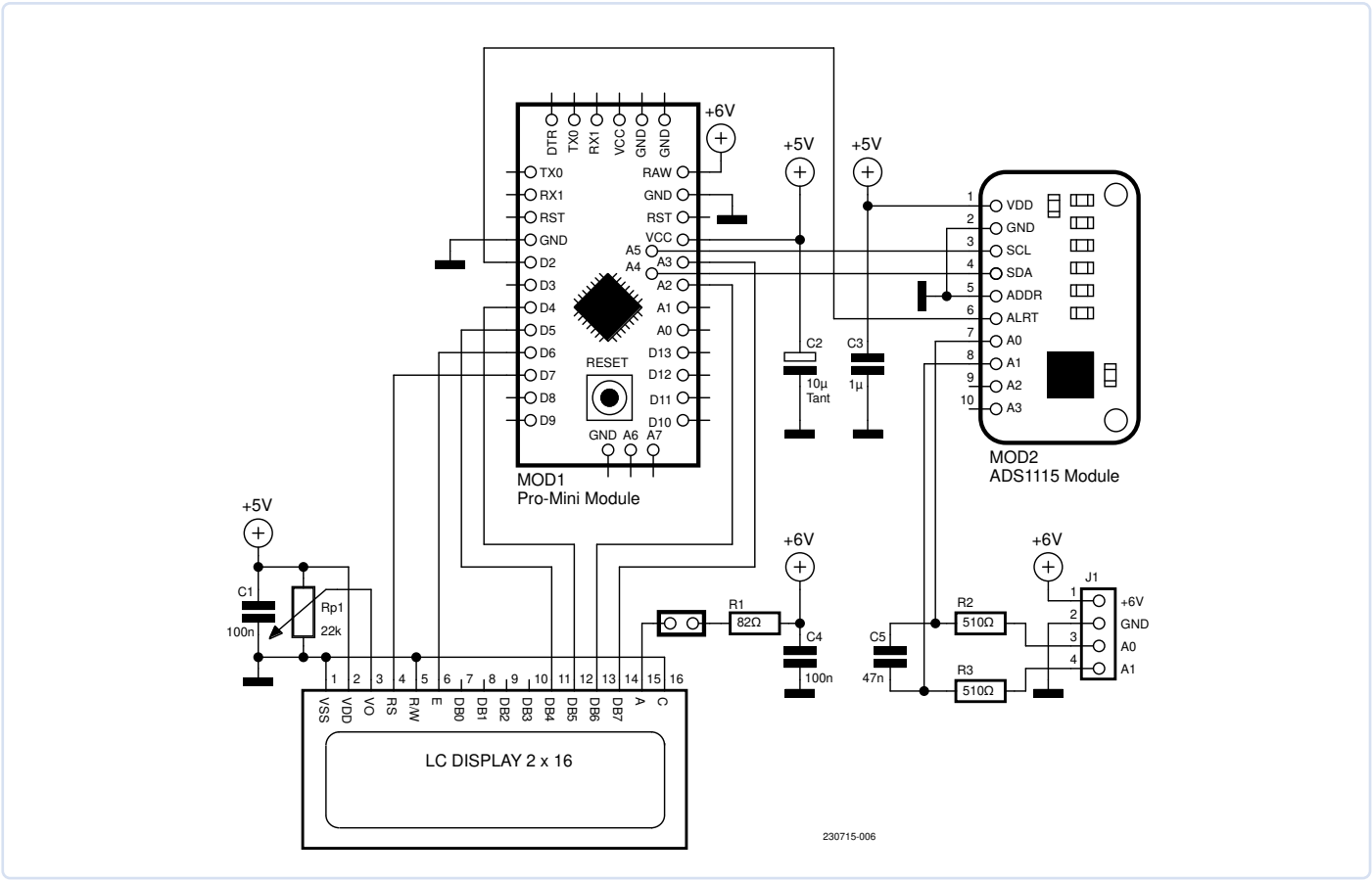


Figure 6. Schéma du DVM.

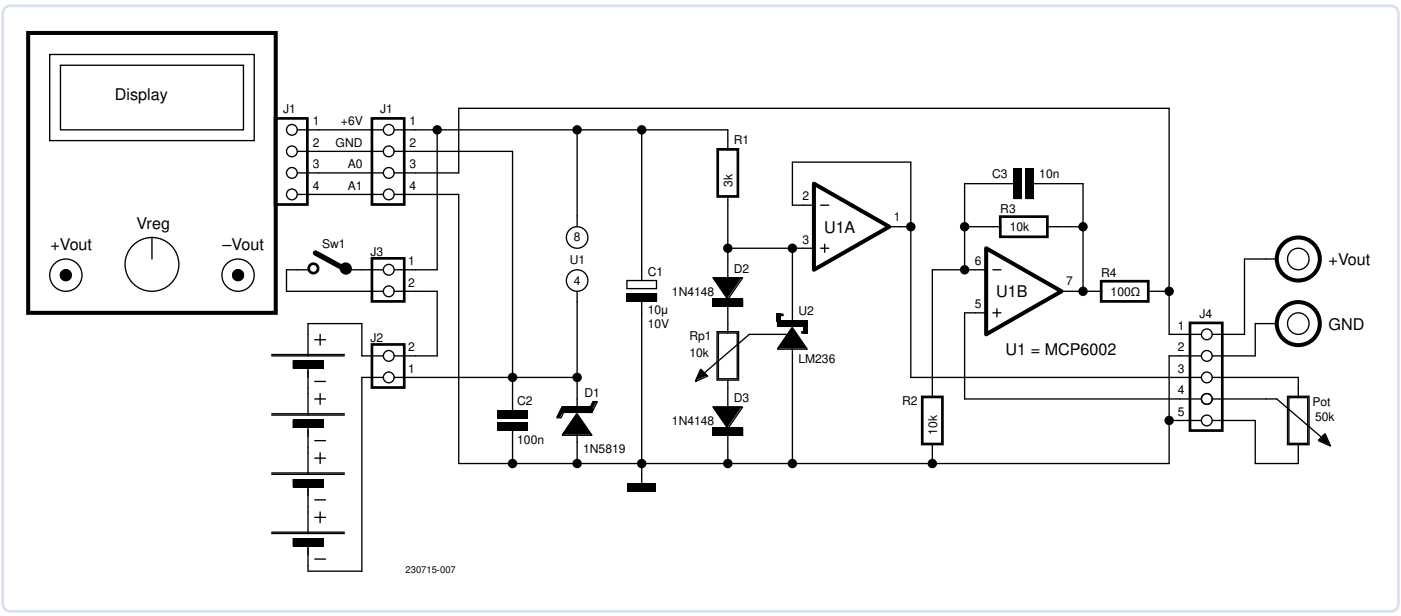


Figure 7. Câblage entre le DVM et la source de tension.

de nombreuses fonctions qui permettent d'exploiter toutes les caractéristiques de la puce.

LCD

Nous avons choisi un écran LCD compatible avec le contrôleur Hitachi HD44780. En particulier, pour afficher des caractères plus grands, nous avons utilisé l'afficheur classique à deux lignes de 16 caractères similaire à celui de la **figure 5**. Nous avons fait ce choix car ces afficheurs sont très lisibles, consomment peu d'énergie (sans rétroéclairage) et ont une alimentation et des niveaux logiques de 5 V, ce qui est parfaitement compatible avec Arduino Pro Mini. La bibliothèque Arduino nécessaire est disponible ici [2].

Schéma de câblage du DVM

La **figure 6** montre le schéma de câblage du DVM. Nous avons utilisé le microcontrôleur Arduino Pro Mini, une carte très compacte, qui fournit également les 5 V pour alimenter l'écran LCD et le convertisseur. Dans ce projet, nous avons utilisé les entrées A0 et A1 du module CAN

configurées en entrée différentielle. La consommation de l'ensemble du système est d'environ 20 à 25 mA lorsque le rétroéclairage de l'écran LCD n'est pas utilisé. La tension idéale à appliquer serait de 6 V ; le système supporte également des tensions plus élevées (12 V max) mais le petit régulateur intégré risque de surchauffer. Le courant de rétroéclairage ne doit pas être fourni par le régulateur interne, mais par le régulateur externe avec la résistance R1 en série qui doit être calculée en fonction du courant requis (voir figure 6).
Nous avons utilisé une résistance de 82 Ω dans le prototype, mais il est possible d'utiliser une valeur de 100 Ω pour économiser quelques mA. Les anciens afficheurs dont les LED étaient de technologie ancienne nécessitaient un courant allant même jusqu'à 100 mA. Les plus récents nécessitent des courants maximums autour de 20 mA car ils disposent de LED à haute luminosité. Un connecteur à 4 broches relie le DVM à la carte de source de tension variable. Le condensateur C5 et les résistances R2 et R3 de 510 Ω ont été utilisés pour réduire le bruit. Ils constituent un filtre passe-bas de premier ordre. La constante de temps est τ = 47.94 µs, à laquelle correspond une fréquence de coupure de

$f = 1/(\tau*2*\pi) = 3.320$ kHz.

Schéma de câblage

Nous avons réalisé deux cartes pour le prototype, l'une pour le DVM et l'autre pour la source de tension (figure 7), mais il est possible de monter tous les composants sur une même carte. Comme l'alimentation des circuits numériques produit du bruit causé par les courants de commutation, nous avons utilisé des condensateurs de dérivation (tels que le condensateur au tantale C2) pour réduire les pics de bruit.

Prototype

La **figure 8** montre le prototype. Les deux cartes sont simple-face, l'afficheur et le CAN sont connectés avec des connecteurs strip 16 et 10 broches, avec un pas de 0,1 pouce, et le module Arduino Pro est connecté avec deux connecteurs strip de 12 broches.
La **figure 9** montre la carte du DVM. Les broches A4 et A5 du bus I²C de l'Arduino Pro Mini ne sont pas disponibles sur les 12+12 broches et ne sont pas non plus sur une grille de 2,54 mm, donc nous avons soudé deux fils (blanc et marron, illustrés) et un connecteur à 2 broches pour acheminer les signaux vers le module ADS1115. Le connecteur en bande à 16 broches que l'on voit en haut à gauche est destiné à l'afficheur LCD, qui est connecté à 90 degrés. La carte est très compacte et est

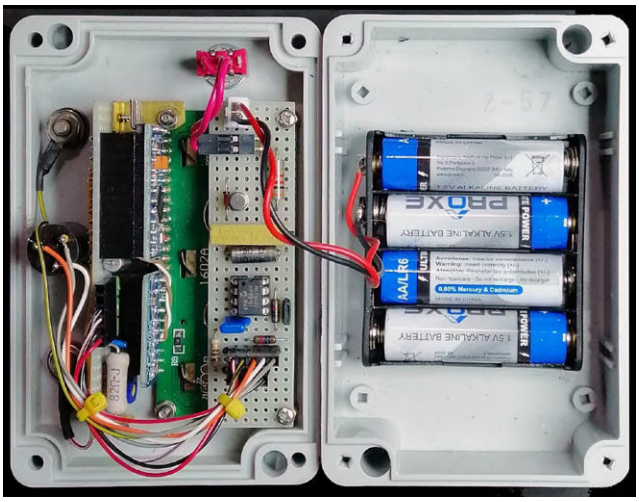


Figure 8. Vue intérieure du prototype terminé.

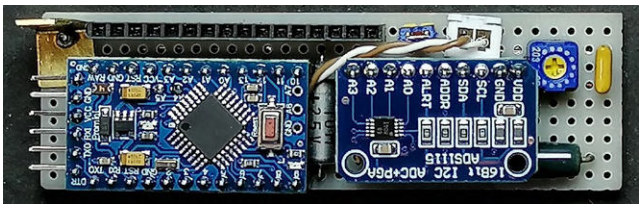


Figure 9. La carte DVM finale.

également fixée à l'écran avec un petit support en laiton, visible en haut à gauche de l'image.

Programmation

Pour la programmation, nous avons besoin d'un adaptateur série USB TTL similaire à celui de la **figure 10**. Bien sûr, nous devons également charger le pilote de l'adaptateur qui, en plus des signaux Rx et Tx, doit avoir le DTR qui est utilisé pour réinitialiser le système. Il sert également d'interface série USB, mais dans ce cas, son rôle principal est de téléverser le croquis compilé de l'EDI Arduino à notre système.

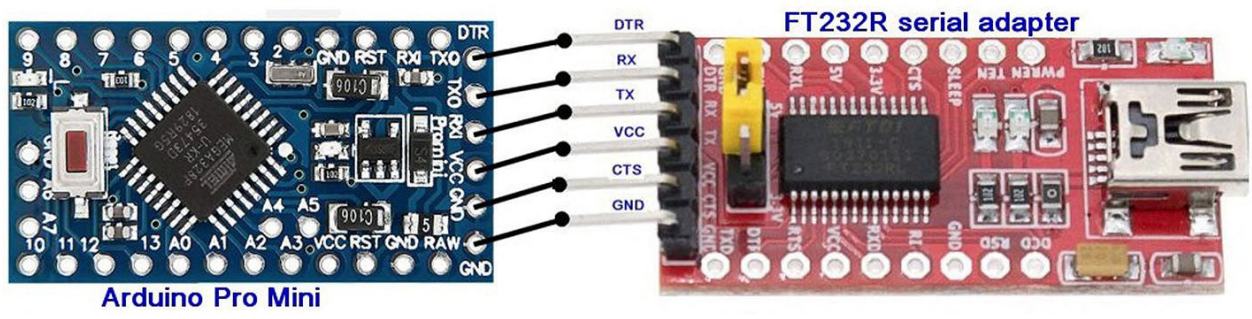


Figure 10. Connexions entre l'adaptateur série et l'Arduino Pro Mini.

Dans l'EDI Arduino, il faut configurer la carte comme Arduino Pro à 16 MHz et 5 V. Une fois programmée, si tout va bien, vous pouvez retirer le programmateur et alimenter la carte par les accus.
Nous avons utilisé un connecteur femelle à 6 broches avec un pas de 0,1 pouce, soudé directement au connecteur de l'adaptateur, qui est mâle, mais vous pouvez aussi utiliser des fils femelle-femelle compatibles avec une plaque d'essai. Les adaptateurs série possèdent des dispositions de broches et des puces différentes. Avec certains adaptateurs série, comme le Prolific PL2303, des problèmes de pilotes ont été rencontrés avec Windows 10.

Calibrage du DVM

Bien que les mesures de l'ADS1115 soient déjà très précises, le calibrage a été réalisé avec un multimètre de table HP 3478A de 5 1/2 digits avec une résolution de 0,1 µV. Pour cela, il faut remplacer la ligne, dans la fonction `printData()` :

```
// calibrated and result in millivolts
voltage = Ch0*1.000515+0.022;
```

avec celle-ci, c'est-à-dire supprimer la correction :

```
// not calibrated and result in millivolts
voltage = Ch0;
```

Environ dix valeurs ont été générées, les lectures du multimètre de référence et les lectures du calibrateur ont été insérées dans Excel, le *nuage de points* avec *la ligne de tendance* a été saisi avec les *options linéaires* et en cochant *Afficher l'équation* sur le graphique et *Afficher le R-carré* sur le graphique. Le résultat était excellent, comme le montre la **figure 11**. La valeur de R2 était de 1 avec pas moins de

Liste des composants du DVM

Résistances
R1 = 82 Ω, ±5%, 1 W , voir texte
R2, R3 = 510 Ω, ±1%, ¼ W
Rp1 = 22 kΩ, trimmer

Condensateurs
C1, C4 = 100 nF, 50 V, céramique
C2 = 10 µF, 25 V, tantale
C3 = 1 µF, 63 V, céramique
C5 = 47 nF, 63 V, polyester

Modules
1 x Arduino Pro Mini (ou compatible)
1 x Afficheur LCD 16x2, avec rétro-éclairage
1 x ADS1115 Convertisseur A/N

Divers
12+12+10+16 Pin Strip, female

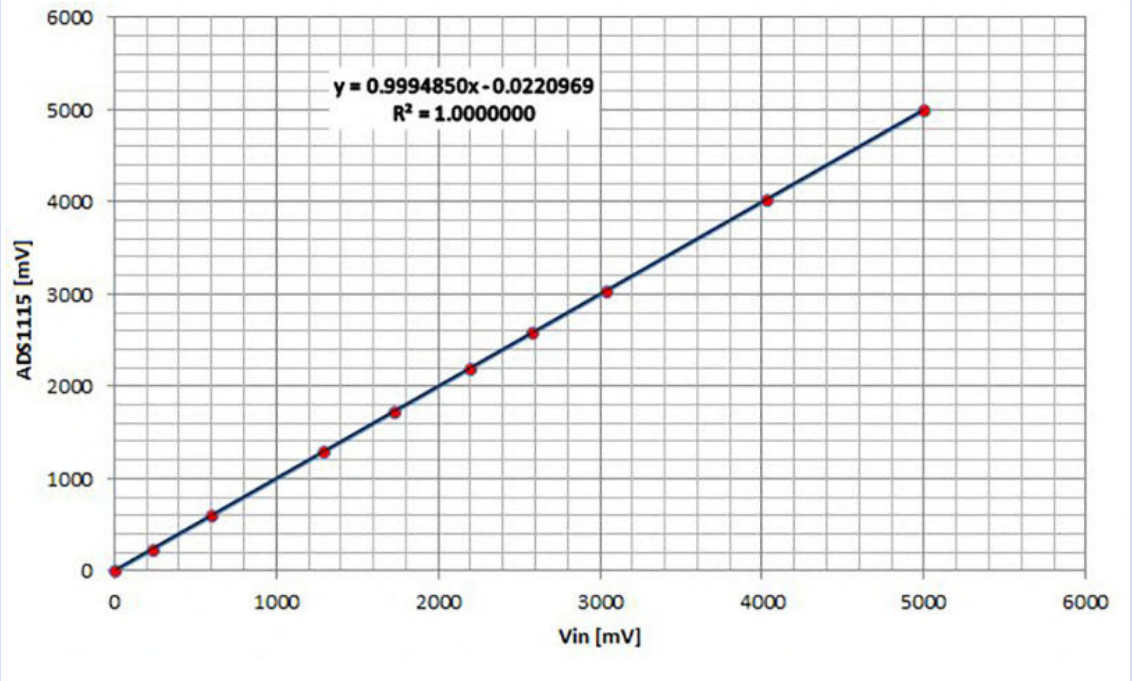


Figure 11. Graphique de calibrage (linéarisation) du DVM.

sept décimales égales à zéro : cela correspond à une valeur de linéarité extrêmement élevée. Pour corriger les données, il faut utiliser la formule inverse, c'est-à-dire inverser les axes. Nous obtenons donc :

- slope = 1.000515286
- intercept = 0.022110099
- error = 0.051941236

Comme vous pouvez le constater, l'erreur est minime. Si nous ne disposons pas d'un instrument très précis, nous pouvons nous passer de la correction, en nous contentant d'une erreur standard de 0,051941236 ; sinon, nous corrigeons la ligne avec les valeurs que nous avons obtenues lors de notre calibrage. Dans le prototype, la tension minimale générée était de 1,7 mV, la maximale de 4 997,03. Le minimum n'est pas égal à zéro à cause de l'offset des ampli-op.

Description du programme

Le programme est relativement simple ; après avoir initialisé les entrées/sorties, l'afficheur et le CAN, il effectue une mesure toutes les 500 ms. La fonction `readChannel(ADS1115_MUX channel)` est celle proposée pour le mode single-shot et effectue la mesure en millivolts. La fonction `LCDprintLine(String text, byte line)` imprime une chaîne sur la ligne 0 ou 1. La fonction `printData()` corrige la mesure avec les résultats du calibrage initial et l'imprime.

230715-04

Questions ou commentaires ?

Contactez Elektor (redaction@elektor.fr).

Listage 1.

```
/*  program ArduCalibrator.ino for the ADS1115, 16 bit 4 ch ADC
a single differential input (A0,A1), Ch0 6144 mV (5000 generated)
read the  precision voltage source
uses an Arduino Pro Mini, LCD 16x2 display
Giovanni Carrera 23/11/2022 with calibration
*/
#include <Wire.h>
#include<ADS1115_WE.h>
#include <LiquidCrystal.h>
ADS1115_WE adc = ADS1115_WE();// uses Wire / ADC Address = 0x48
// LCD pins
#define rs 7
#define en 6
#define d4 5
#define d5 4
#define d6 A2
#define d7 A3
// initialize the library by associating any needed LCD interface pin
LiquidCrystal lcd(rs, en, d4, d5, d6, d7);
#define Alert 2  // AD Alert pin
const unsigned long deltat_ms = 500;
unsigned long cms, pms;
float Ch0,Ch1;
void setup(){
   Wire.begin();
   //Serial.begin(115200);
   pinMode(Alert,INPUT);
   // set up the LCD's number of columns and rows:
   lcd.begin(16, 2);
   lcd.print(F("ArduCalibrator"));
   lcd.setCursor(0, 1);// print on the second row
   lcd.print(F("GCar V231122"));
   delay(2000);
   if(!adc.init()){
```

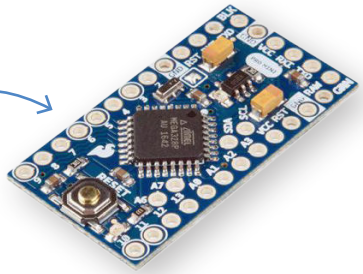
```
    lcd.setCursor(0, 1);
    lcd.print(F("No ADS1115 found"));
    while( true );// ends here
  }
  adc.setVoltageRange_mV(ADS1115_RANGE_6144); // 6144 mV input range
  LCDprintLine("PRECISION SOURCE", 0);
}
void loop() {
  cms = millis();
  // checks if passed deltat milliseconds
  if(cms - pms > deltat_ms) {   // period timebase
      pms = cms;// update pms
    Ch0 = readChannel(ADS1115_COMP_0_1);// Ch0 differential A0(+) with A1(-)
    Ch0 = readChannel(ADS1115_COMP_0_1);// repeating helps
    printData();
  }
}
```

À propos de l'auteur

Giovanni Carrera est titulaire d'un diplôme en ingénierie électronique. En tant que professeur d'université à la faculté d'ingénierie navale de Gênes, en Italie, il a enseigné de nombreux cours, tels que l'automatisation navale et la simulation des systèmes de propulsion des navires. Carrera a commencé à travailler à la fin des années 1970 avec le CPU 6502, avant de passer à d'autres processeurs. Aujourd'hui, il s'adonne à la conception et au développement de circuits électroniques analogiques et numériques, dont il a présenté un grand nombre sur ses blogs (ArduPicLab et GnssRtkLab) et dans divers magazines.

Produits

- **SparkFun Arduino Pro Mini 328 (5 V, 16 MHz)**
  www.elektor.fr/20091
- **Écran LCD rétroéclairé standard de 2 x 16 caractères**
  www.elektor.fr/16414



LIENS

[1] Wolfgang Ewald's ADS1115 library: https://github.com/wollewald/ADS1115_WE
[2] Bibliothèque LiquidCrystal : https://github.com/arduino-libraries/LiquidCrystal
[3] Téléchargements pour ce projet: https://elektormagazine.fr/Arducalibrator
[4] G. Carrera, « Calibrateur de CAN à faible bruit pour les microcontrôleurs modernes «, circuits de vacances 2022 : https://www.elektormagazine.fr/magazine/elektor-264/60886
[5] Ultra-Small, Low-Power, 16-Bit Analog-to-Digital Converter with Internal Reference: https://ti.com/lit/ds/symlink/ads1114.pdf

FPGA pour les débutants

de la programmation des microcontrôleurs à celle des FPGA

Theo Mulder (Pays-Bas)

Il existe de nombreuses cartes de développement à microcontrôleur peu coûteuses, mais les FPGA restent chers. Pour les applications qui nécessitent des performances de calcul plus élevées (par exemple, l'intelligence artificielle) ou des E/S plus rapides (par exemple, une caméra à grande vitesse, un écran à grande vitesse, un CAN ou un CNA rapide), il est préférable de trouver des cartes FPGA de petite taille et d'un prix abordable. Plongeons dans le monde passionnant des FPGA pour les débutants !

En cherchant des FPGA peu coûteux à utiliser, il est utile de comprendre les tendances actuelles du marché. Le marché des FPGA était évalué à 8,0 milliards de dollars en 2022, et devrait atteindre 15,5 milliards de dollars d'ici 2027. Le secteur est concurrentiel et a connu une consolidation importante des fournisseurs. Aujourd'hui, les marques restantes sont AMD/Xilinx, Intel/Altera, Lattice, Microchip, QuickLogic, Efinix, Flex Logix, GOWIN, Achronix, S2C et Renesas.

Grâce aux acquisitions, le paysage ne cesse de changer. Dès 1999, Lattice a acquis Vantis, puis Acree en 2001, et Microsemi a acquis Actel en 2010. En 2013, les principaux acteurs étaient Xilinx, Altera, Actel, Vantis, Lattice, Lucent, QuickLogic et Cypress. Cependant, le mouvement de fusions et d'acquisitions n'a jamais cessé ; même Xilinx et Altera, ont été rachetés par AMD et Intel, respectivement.

En 2019, Xilinx dominait le marché avec une part de 52 %, suivi par Altera avec 35 %, et des parts plus modestes étaient détenues par Microchip, Lattice et d'autres. Malgré la domination de ces grandes entreprises, les petits fournisseurs de FPGA restent actifs, une excellente situation pour soutenir différents types d'entreprises et promouvoir l'innovation.

L'utilisation accrue des FPGA dans l'industrie automobile offre des opportunités à des fournisseurs tels que Lattice, car les concepteurs industriels recherchent des alternatives rentables aux offres plus onéreuses des plus grandes entreprises. Renesas apparaît comme un concurrent important grâce à sa présence bien établie dans le secteur automobile.

La dynamique du marché des FPGA, avec ses changements concurrentiels, ses acquisitions et ses nouveaux acteurs, pourrait profiter à ceux qui recherchent des FPGA à bas prix pour leurs projets.

Cartes FPGA abordables pour les débutants

Étant donné que les chaînes de compilation des différents fabricants de FPGA sont très différentes les unes des autres et que l'apprentissage de l'utilisation d'un nouvel outil logiciel prend du temps, il est judicieux de différencier les cartes en fonction des fabricants de FPGA. Ainsi, le temps d'apprentissage aura été bien investi si, à l'avenir, vous pouvez décider de passer à un modèle plus haut de gamme du même fabricant de FPGA.

Les trois principaux fabricants à prendre en considération sont Altera/Intel, Xilinx/AMD et Lattice. Ce dernier a tendance à proposer des modèles moins puissants et plus abordables que les deux premiers.

En commençant par les cartes dotées de FPGA de Lattice Semiconductor, la carte d'évaluation Olimex iCE40HX1K-EVB, dont le prix est de 15 €, constitue un point d'entrée très abordable, bien qu'elle nécessite un programmateur externe. Cette carte dispose d'un iCE40HX1K avec 1280 éléments logiques (LE). Fait intéressant, ce FPGA possède une mémoire de configuration interne, non volatile, programmable via l'interface SPI d'une carte Arduino que vous possédez peut-être, grâce au croquis fourni par Olimex. Le programmateur officiel de Lattice utilise un FT232H de FTDI, et il est également possible d'utiliser un breakout FT232H. L'une des caractéristiques intéressantes de cette famille de FPGA est qu'elle est compatible avec l'outil graphique open source Icestudio, qui permet aux utilisateurs d'expérimenter la programmation graphique par blocs.

La famille TinyFPGA propose des produits intéressants pour ceux qui recherchent d'autres modèles très compacts, abordables. et, compatibles avec des plaques d'essai. Le TinyFPGA BX est proposé à 40 €, avec un iCE40LP8K de Lattice, également compatible avec Icestudio. Les versions AX1 et AX2 sont disponibles respectivement à 13 et 17 €, offrant des alternatives plus

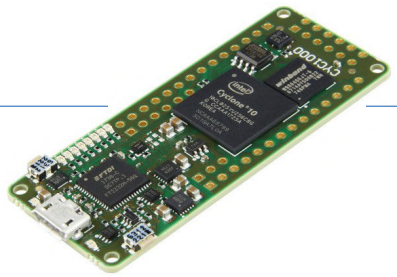


Figure 1. La carte CYC1000.

petites et moins chères, mais avec moins de fonctions. Malheureusement, ces trois modèles sont souvent en rupture de stock. L'Upduino v3.1 d'un autre fabriquant, au prix de 30 €, offre un FPGA iCE40UP5K (5.3 kLE).

Le iCEstick de Lattice est une option compacte au format d'une clé USB, intégrant un FPGA iCE40HX1K et un programmateur embarqué, au prix de 48 €. C'est l'outil officiel de Lattice, il fonctionne donc bien avec le logiciel de Lattice et il existe de nombreux tutoriels sur son utilisation.

Pour compléter les options de Lattice, mentionnons également une plateforme open-source dans un format Arduino UNO, l'Alhambra II (iCE40HX4K, 3,52 kLE, 60 €) et aussi la carte Go Board (iCE40HX1K, 70 €) qui permet de soutenir le créateur d'excellents tutoriels de Nandland.com.

Passant à Altera (maintenant propriété d'Intel), la carte MAX1000 de Trenz Electronic fournit un FPGA MAX10, disponible en 8 ou 16 kLE (34 € à 49 €), offrant une conception compatible avec les plaques d'essai. La CYC1000 [1] (**figure 1**), également de Trenz, au prix de 40 €, surenchérit avec un FPGA Cyclone 10 (25 kLE).

Pour des cartes plus classiques et riches en fonctionnalités, il y a les solutions de Terasic. Certains de leurs produits sont chers, mais d'autres sont très intéressants pour les débutants, comme le DE0-nano qui offre un FPGA Cyclone IV avec 22 kLE et une multitude de fonctionnalités embarquées pour 110 €. Le DE10-Lite (140 €) est une carte riche en fonctionnalités avec un FPGA MAX10 (50 kLE), diverses options d'E/S, des afficheurs à sept segments, des LED et des interrupteurs, ainsi qu'un support pour les shields Arduino.

Enfin, sous l'égide d'AMD (anciennement Xilinx), les Cmod S7 et Cmod A7-35T de Digilent proposent tous deux des designs compatibles avec les plaques d'essai, avec des FPGA Spartan 7 et Artix 7, respectivement de 23,4 et 33,3 kLE, au prix de 90 € chacun.

Pour une expérience d'apprentissage riche avec une carte plus grande, le Digilent Basys 3 apporte à la table un FPGA Artix 7 avec 33,3 kLE, des options d'E/S étendues, y compris une sortie VGA et un hôte USB, au prix de 155 €.

Dans cet article, je me concentrerai sur les produits Intel/Altera. Je recommande le CYC1000. Dans les sections suivantes, nous allons expérimenter avec Quartus Prime Lite d'Intel, qui est la version gratuite de l'outil de développement officiel pour les FPGA Altera/Intel.

Installation

Sur la page Intel Quartus Prime Design Software [2], vous trouverez les liens de téléchargement de la dernière version, sous Windows ou Linux. Il est nécessaire d'avoir une bonne connexion Internet, car le fichier fait environ 5,5 Go. L'installateur nécessite 15 GB d'espace libre. Certaines parties de Quartus Prime Lite utilisent du code provenant des premières versions de Quartus. Par conséquent, vous devez éviter les espaces dans les noms de fichiers, même si cela peut paraître surprenant par rapport aux normes d'aujourd'hui.

Seul l'outil de conception de bas niveau est nécessaire. Il existe des packages supplémentaires qui dépassent le cadre de cette introduction, tels que DSP Builder (pour les applications de traitement du signal), High Level Synthesis Compiler (utilisé avec C++ en entrée, pour les applications avancées) ainsi que Nios II Embedded Design Suite (utilisé pour implémenter un processeur logiciel Nios II sur un FPGA).

Créer un nouveau projet

Commencez par créer un répertoire - par exemple, *C:\NProjects\NIntelFpga\NElektor\NBeginExample*. Ensuite, lancez Quartus Prime Lite. L'écran principal est représenté sur la **figure 2**. Allez dans *File*, puis sélectionnez *New Project Wizard*.

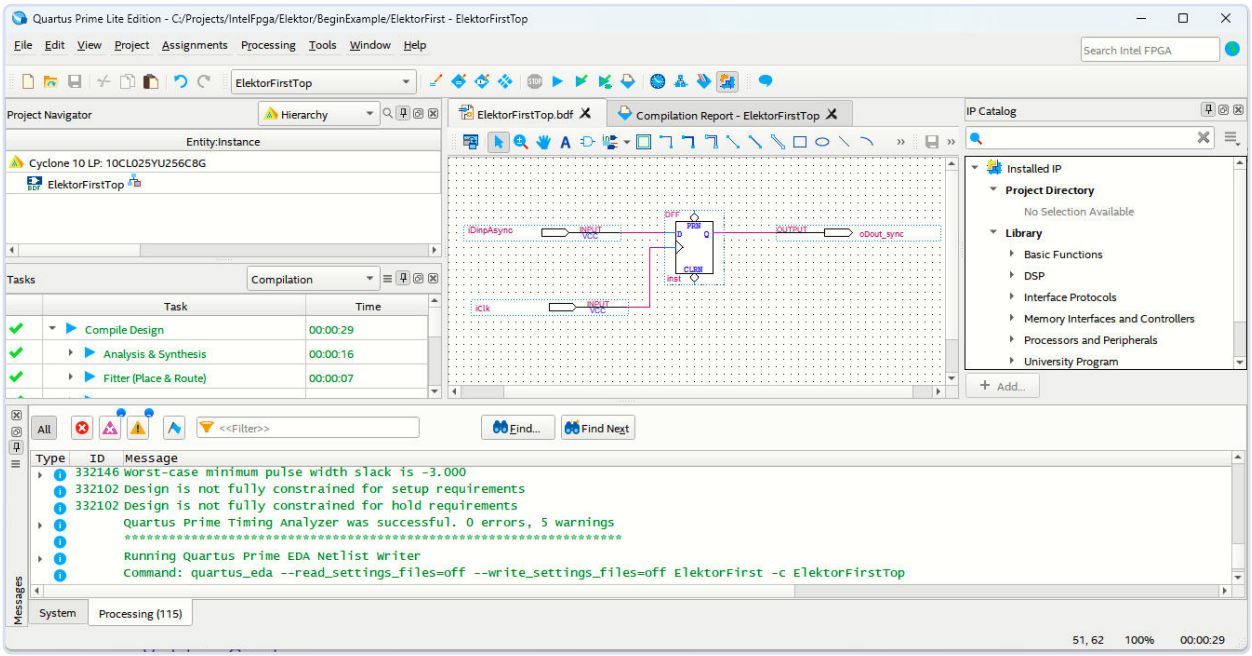
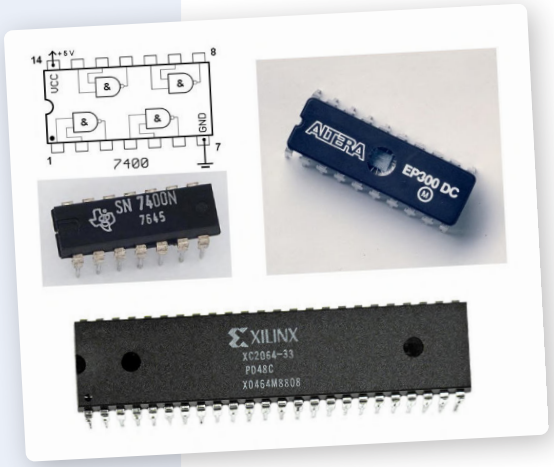


Figure 2. Quartus Lite Edition, avec ElektorFirstTop.bdf.

Historique : du TTL au FPGA

Avant l'avènement des FPGA, les circuits numériques s'appuyaient principalement sur des puces TTL et des circuits programmables tels que les FPLD, les PLA et les PAL. La série TTL SN7400 fonctionnait jusqu'à 20 MHz. Il y a 40 ans, Altera a introduit l'EP300, suivi par le XC2064 de Xilinx. L'EP300 était gravé avec une finesse de 5000 nm et avait des vitesses allant jusqu'à environ 10 MHz, tandis que les FPGA de milieu de gamme modernes sont gravés avec une finessde 20 nm et atteignent des vitesses d'horloge d'au moins 250 MHz. L'EP300 était un circuit reprogrammable à 20 broches utilisant le logiciel A+PLUS sur des PC IBM fonctionnant sous DOS. Le XC2064 de Xilinx comportait des cellules logiques de 1200 portes chacune et utilisait une technologie CMOS à faible consommation, avec un logiciel de conception comprenant Future-Net, VIEW-logic et XACT-Design-Editor. Le XC2064 était annoncé avec un taux de basculement interne de 100 MHz, mais il atteignait généralement 70 MHz.
Au départ, les outils logiciels FPGA utilisaient la capture schématique, mais celle-ci a ensuite été complétée par des langages de description de matériel (HDL) basés sur le texte, tels que AHDL pour Altera, et VHDL et Verilog pour Xilinx. Aujourd'hui, les deux sociétés prennent en charge les langages VHDL et Verilog. Les anciennes versions de l'outil Quartus d'Altera utilisaient l'AHDL avec des fonctions de simulation intégrées, nécessitant par la suite l'utilisation de ModelSim pour la simulation due au manque de prise en charge de l'AHDL. Aujourd'hui, tous les fabricants proposent une version gratuite de leurs outils.



Après avoir consulté la fenêtre de bienvenue, cliquez sur *Next*. Lorsque vous êtes invité à indiquer le répertoire de travail, choisissez le répertoire que vous avez créé. Vous devez choisir un nom de projet, ainsi qu'un nom pour l'entité de conception de premier niveau, que le compilateur utilisera comme racine du design. Dans cet exemple, nous avons utilisé *ElektorFirst* et *ElektorFirstTop*. Ensuite, choisissez *Empty Project*. Vous accéderez alors à l'étape *Add files*. Comme nous n'avons pas de fichiers, il suffit de cliquer à nouveau sur *Next*.
Vous accéderez alors à une page où vous pourrez sélectionner un appareil, ce qui peut être effectué via l'un des deux onglets : *Device* et Board. Dans ce dernier, plusieurs cartes de développement sont listées dans les familles MAX10 et Cyclone V et il est possible d'installer des cartes supplémentaires ultérieurement. Dans l'onglet *Device*, vous pouvez choisir la famille et la référence exacte du FPGA que vous utilisez, par exemple le *10LC025YU256C8G* dans la famille Cyclone 10 LP si vous avez la carte CYC1000. La famille Cyclone 10 est constituée de modules à faible coût et à faible consommation d'énergie, dérivés du Cyclone V.
Si vous n'avez pas de carte de développement, vous pouvez toujours sélectionner cette carte pour avancer dans le tutoriel et expérimenter le processus de compilation. Cliquez sur *Next*, *Next* à nouveau et *Finish*.

Structure du répertoire du projet

Jetez un coup d'œill à votre répertoire de travail. Vous remarquerez qu'il y a un répertoire db que Quartus utilisera plus tard. Le fichier nommé *ElektorFirst.qpf* est le fichier de projet *Quartus (QPF)*. Lorsque vous souhaitez reprendre votre travail plus tard, il vous suffit de double-cliquer sur ce fichier pour relancer le logiciel Quartus. Il existe également un fichier appelé *ElektorFirst.qsf*, qui est le *Quartus Settings File (QSF)*. Ces deux fichiers sont des fichiers texte, vous pouvez donc les examiner si vous êtes curieux - il suffit de cliquer avec le bouton droit de la souris et de choisir *Open with* pour choisir un éditeur de texte.

Ouvrir un projet

Dans le coin supérieur gauche, on voit *Project Navigator* avec l'option *Hierarchy* sélectionnée. Recherchez *ElektorFirstTop*, que nous avons nommé plus tôt. Lorsque vous cliquez dessus, une fenêtre contextuelle apparaît et indique « Can't find design entity ElectorFirstTop », car nous n'avons pas encore créé de fichier. Cliquez sur *OK*, puis allez dans le menu *File*. Ici, vous pouvez voir les différents types de fichiers que vous pouvez créer. Choisissez *Block Diagram/Schematic File* et cliquez sur *OK*.
Vous avez maintenant un canevas blanc dans le volet central, intitulé *Block1.bdf* en haut. Allez dans *File*, sélectionnez *Save As* et le logiciel vous proposera d'enregistrer le fichier sous *ElectorFirstTop.bdf*. Parfait, cliquez sur *Save*. Si vous cliquez sur *ElektorFirstTop* dans le *Project Navigator* et *Hierarchy*, vous arrivez sur cette feuille de schéma vide.

Ajouter des composants à notre premier schéma

Afin que tous nos lecteurs, même ceux qui ne maîtrisent pas les langages Verilog et VHDL, puissent suivre ce tutoriel, nous allons utiliser la programmation schématique. Cliquez sur l'onglet *ElektorFirstTop.bdf* pour ouvrir la feuille, faites un clic droit n'importe où sur la feuille et choisissez *Insert Symbol*. Le menu Bibliothèques apparaît. Cliquez sur le petit triangle à côté de *Libraries* pour développer la liste, où vous verrez des catégories comme *Megafunctions*, *Other*, *Primitives*, et plus encore. Il y a beaucoup à découvrir ici, alors n'hésitez pas à naviguer.
Faites défiler jusqu'au bas de la fenêtre

Libraries et dans le champ *Name* tapez « Input ». Un symbole d'entrée apparaît ; cliquez sur *OK*. Placez ce symbole quelque part sur la page, idéalement en haut à gauche. Ajoutez ensuite une deuxième entrée en cliquant avec le bouton droit de la souris et en choisissant *Insert Symbol*. Puisque « Input » devrait toujours figurer dans le champ Name, cliquez à nouveau sur *OK* et placez cette nouvelle entrée en dessous de la première sur votre feuille.
Ensuite, nous allons ajouter une bascule D. Cliquez avec le bouton droit de la souris, sélectionnez *Insert Symbol* et, dans le champ *Name*, tapez « dff ». Choisissez *dff* parmi les options proposées et cliquez sur *OK*, puis placez-le sur la feuille. Pour terminer cette étape, cliquez avec le bouton droit de la souris, sélectionnez *Insert Symbol*, saisissez « Output » dans le champ *Name* et cliquez sur *OK*. Placez également cette sortie sur la feuille.

Renommer les broches

Nous devrons attribuer des noms significatifs aux broches. Il est possible d'ajouter des préfixes ou des suffixes pour donner des détails selon nos besoins, comme i pour entrée (*input*), o pour sortie (*output*), sync ou async pour synchrone ou asynchrone, respectivement, etc. Nous allons donc renommer l'entrée *pin_name1* en *iDinpAsync*. Pour ce faire, double-cliquez sur *pin_name1* et saisissez le nouveau nom. Renommez également *pin_name2* en *iClk* et *pin_name3* en *oDout_sync*. Cliquez ensuite sur *File* et *Save*.

Ajouter des fils

Cliquez sur la broche *iDinpAsync*. Ceci activera automatiquement l'outil *Orthogonal Node* utilisé pour dessiner des fils. Commencez à dessiner à partir de cette broche et connectez-la à l'entrée D de la bascule D (DFF). Ensuite, procédez à la connexion de *iClk* à l'entrée d'horloge de la DFF, qui est indiquée par un triangle dans le symbole. Enfin, créez une connexion de la sortie Q de la DFF au symbole de sortie *oDout_sync*. Sauvegardez à nouveau.

Compiler

Pour compiler le dessin ou modèle, il suffit de cliquer sur l'icône du triangle bleu dans la barre d'icônes. Gardez un œil sur le volet des tâches à gauche pour suivre la progression et surveillez les messages qui défilent au bas de l'écran. Il s'agit d'un processus assez intéressant.
Après la compilation, le volet central affiche le volet *Flow Summary*. Ce résumé indique qu'un seul des 24 624 éléments logiques a été utilisé. Il est rare d'utiliser 100 % des éléments logiques d'un composant ; dans les circuits les plus complexes et les plus optimisés, environ 80 % des éléments logiques peuvent être utilisés. Au-delà de ce seuil, vous pouvez rencontrer des difficultés de routage ou le composant peut ne pas répondre aux exigences de synchronisation en raison de l'encombrement.
Le résumé montre également qu'un seul registre a été utilisé et que trois des 151 broches possibles sont utilisées, ce qui équivaut à 2 % du nombre total de broches. Il faut toutefois garder à l'esprit que certaines broches peuvent être réservées et non disponibles pour un usage général.
Vérifiez à nouveau votre répertoire de projet. Vous y trouverez de nouveaux sous-dossiers. Jetez un coup d'œil dans le dossier *db* pour vous faire une idée du travail de fond effectué. Le fichier *Elektor-FirstTop.sof* que vous utiliserez pour programmer votre FPGA se trouve dans le dossier *output_files*.

Choisir le FPGA idéal pour votre projet

Lorsqu'on choisit des FPGA et des cartes, il y a plusieurs paramètres à prendre en compte. Le nombre d'éléments logiques, qui donne une idée approximative de la taille du FPGA, en est un. Il faut également tenir compte de la nécessité de disposer d'entrées/sorties à faible ou à grande vitesse. Les FPGA haut de gamme dotés d'une mémoire rapide et d'émetteurs-récepteurs sont coûteux, tout comme les outils logiciels associés. Bien entendu, les classifications de haut, moyen et bas de gamme sont subjectives et peuvent varier d'un fournisseur à l'autre. Pour les deux principaux fournisseurs de FPGA, ce qui est considéré comme une performance de milieu de gamme peut être qualifié de haut de gamme par Lattice.
En ce qui concerne les E/S, les émetteurs-récepteurs sont souvent mentionnés. Il s'agit de connexions série qui offrent des taux de transfert élevés (par exemple de 4 à 32 Gbps), mais qui affectent considérablement le coût. Ils ne sont pas nécessaires pour toutes les applications. Au lieu de connexions PCIe ou COM Express à haut débit, une solution plus économique et plus pratique pour de nombreuses applications consiste à utiliser l'USB, éventuellement avec un dispositif FTDI pour l'interfaçage. Pour connecter des ADC ou DAC à grande vitesse à un FPGA, il est possible d'utiliser l'interface JESD204B, mais cela nécessite des émetteurs-récepteurs à la fois du côté FPGA et du côté ADC/DAC, ce qui peut s'avérer coûteux. Une attention particulière à l'intégrité des signaux est nécessaire lors du routage de la carte, mais la disposition du circuit est quelque peu simplifiée par le nombre réduit de pistes. Une autre option lorsque des débits importants sont nécessaires est d'utiliser plusieurs paires LVDS (Low Voltage Differential Signaling) plus lentes en parallèle. Dans tous les cas, il est utile de faire un calcul assez précis de vos besoins en termes de débit pour pouvoir choisir un composant adéquat.
J'ai tendance à considérer que les interfaces sont à bas débit lorsqu'elles fonctionnent à moins de 1000 Mbit/s, et à haut débit au-delà. Le coût des FPGA augmente avec le nombre d'interfaces à haut débit. Cependant, dans de nombreux cas, même un LVDS à vitesse modérée n'est pas nécessaire : des broches d'E/S à usage général sont suffisantes.
Sur les FPGA, les broches d'E/S à usage général sont les plus lentes des interfaces disponibles, mais elles ont toujours des taux de basculement plus rapides que les broches d'E/S à usage général sur les microcontrôleurs. Elles sont généralement compatibles avec une logique de 3,3 V, 2,5 V ou 1,8 V. Si vous pensiez utiliser un microcontrôleur pour votre prochain projet, mais avec des E/S plus rapides que ce que les microcontrôleurs typiques peuvent produire, les FPGA constituent une solution compacte, car il est possible d'implémenter un processeur (RISC-V ou autre) dans la structure FPGA.

VHDL ou Verilog?

Pour les débutants, le choix entre Verilog et VHDL dans le cadre de l'apprentissage des FPGA dépend de plusieurs facteurs. Pour ceux qui souhaitent travailler dans les secteurs de la défense ou de l'avionique, VHDL est le langage prédominant, en particulier dans les pays européens comme l'Allemagne et la France. En revanche, aux États-Unis et dans de nombreux secteurs autres que la défense, Verilog est plus couramment utilisé.

Cependant, toute personne souhaitant faire carrière dans le domaine de la conception des circuits numériques aura probablement besoin de maîtriser les deux langages. Il est important de vous adapter à votre objectif d'apprentissage ; s'il s'agit d'un objectif académique, alignez-vous sur le langage enseigné dans vos cours, tandis que pour les environnements professionnels, il est préférable d'utiliser ce que vos collègues utilisent.

D'un point de vue technique, Verilog a tendance à être plus compact et à ressembler au langage C à certains égards, ce qui n'est pas toujours avantageux car cela pourrait conduire à une vision axée sur le logiciel plutôt que sur le matériel. Le caractère textuel de VHDL peut permettre d'éviter certaines erreurs que Verilog pourrait manquer. Cette distinction peut faire du VHDL un meilleur point de départ pour ceux qui préfèrent un langage qui impose un processus de pensée centré sur le matériel, ce qui est crucial pour la conception de FPGA.

Enfin, examinez le fichier *ElektorFirstTop.pin* en l'ouvrant avec un éditeur de texte. Ce fichier montre les connexions de broches que le compilateur a configurées. Comme nous n'avons pas spécifié de conditions particulières, Quartus a sélectionné les broches au hasard. En règle générale, en tant que concepteur, vous devriez définir vos affectations de broches. Recherchez les broches *iDinpAsync*, *iClk* et *oDout_sync*. Dans ma situation, *iDinpAsync* est connecté à la broche T4, fonctionne à un niveau de 2,5 V, et se trouve dans le *bank* 3 du FPGA.

Programmer le FPGA

Bien sûr, nous sommes curieux de voir si cela fonctionne avec un vrai FPGA, alors essayons ! Tout d'abord, connectez votre carte à un port USB disponible sur votre ordinateur et suivez les étapes pour charger le fichier de configuration dans la SRAM du FPGA. Vous pouvez également le charger sur la mémoire flash, ce qui permet au FPGA de conserver le circuit même après les cycles d'alimentation.

Pour ce faire, allez dans *Tools* puis dans Programmer, ou double-cliquez simplement sur *Program Device* dans la liste des tâches. Cliquer sur Hardware Setup, et sélectionner *USB-Blaster [USB-0]*. Cliquez ensuite sur *Add File* et choisissez le fichier objet *SRAM ElektorFirstTop.sof* qui a été généré dans le répertoire *output_files* après la compilation. Cliquez sur *Start*. Pensez à sauvegarder la configuration du programmateur, par exemple sous le nom de *example1.cdf*, afin qu'il s'ouvre avec ces paramètres lors de sa prochaine utilisation. Après quelques secondes, le transfert est terminé ! Nous avons réussi à charger le circuit sur le FPGA, où il restera jusqu'à ce que l'alimentation soit déconnectée. Félicitations, vous avez maintenant une bascule D très sophistiquée, vous pouvez la tester en branchant des fils et des interrupteurs sur les broches appropriées de la carte CYC1000 !

Design hiérarchique

Il peut être très utile de concevoir une partie d'un système une fois pour toutes, et de pouvoir ensuite l'utiliser à plusieurs emplacements du système. Pour ce faire, nous pouvons intégrer une partie du schéma dans ce que l'on appelle un symbole. À titre d'exemple, créons un symbole pour notre circuit.

Tout d'abord, créez un nouveau fichier de diagramme fonctionnel/schéma comme nous l'avons fait dans la section « Ouvrir un projet » : *File*, *New*, etc. Nommez-le *Synchronizer.bdf*. Ensuite, dans la fenêtre *ElektorFirstTop.bdf*, cliquez et maintenez la souris enfoncée pour sélectionner tous les composants dans un rectangle. Copiez et collez ensuite le contenu dans le fichier *Synchronizer.bdf* vide et enregistrez. Ensuite, la fenêtre *Synchronizer.bdf* étant toujours ouverte, allez dans File, sélectionnez *Create/Update*, sélectionnez *Create Symbol Files for Current File* et cliquez sur *Save* (avec le nom suggéré *Synchronizer.bsf*). Dans la fenêtre contextuelle, cliquez sur *OK*.

Un symbole pour un DFF a été créé sous le nom de *Synchronizer.bsf*. Vous pouvez fermer la fenêtre Synchronizer. Dans *ElektorFirstTop.bdf*, vous pouvez tester et utiliser ce symbole nouvellement créé. Enlevez la bascule D, déplacez les entrées et les sorties sur les côtés pour faire de la place, et faites un clic droit dans l'espace vide pour insérer le symbole comme nous l'avons fait dans la section « Ajouter des composants à notre premier schéma ». Le symbole *Synchronizer* nouvellement créé sera disponible dans le menu *Insert Symbol*. Vous pouvez maintenant connecter le symbole aux entrées et sorties appropriées et enregistrer à nouveau. Vous pouvez à nouveau compiler ce design hiérarchique.

Utiliser un langage de description du matériel

La plupart des utilisateurs préfèrent utiliser un langage de description du matériel (ou HDL), tel que Verilog ou VHDL, plutôt que des schémas de blocs. De nombreuses ressources sont disponibles pour ces langages. En outre, les schémas sont notoirement difficiles à maintenir et à mettre à jour à mesure qu'ils prennent de l'ampleur. Convertissons le fichier *Synchronizer.bdf* en un fichier Verilog. Cliquez sur *Synchronizer.bdf*, accédez à *File*, puis à *Create/Update*, et choisissez *Create HDL Design File from Current File*. Sélectionnez *Verilog HDL* et cliquez sur *OK*. Un fichier *Synchronizer.v* devrait maintenant apparaître dans votre répertoire de conception. Vous pouvez ouvrir ce fichier avec n'importe quel éditeur de texte ou avec Quartus lui-même. Allez dans File, sélectionnez Open, trouvez *Synchronizer.v*, et ouvrez-le. Voilà, vous avez maintenant la description Verilog de votre circuit de synchronisation en main. Cela peut être utile à des fins d'apprentissage ! Bien entendu, il est également possible de créer un fichier VHDL.

Vous devrez créer un nouveau symbole pour ce nouveau fichier *Synchronizer.v*. Suivez les mêmes étapes que celles décrites dans la section « Design hiérarchique » et utilisez la commande *Create Symbol for Current File*.

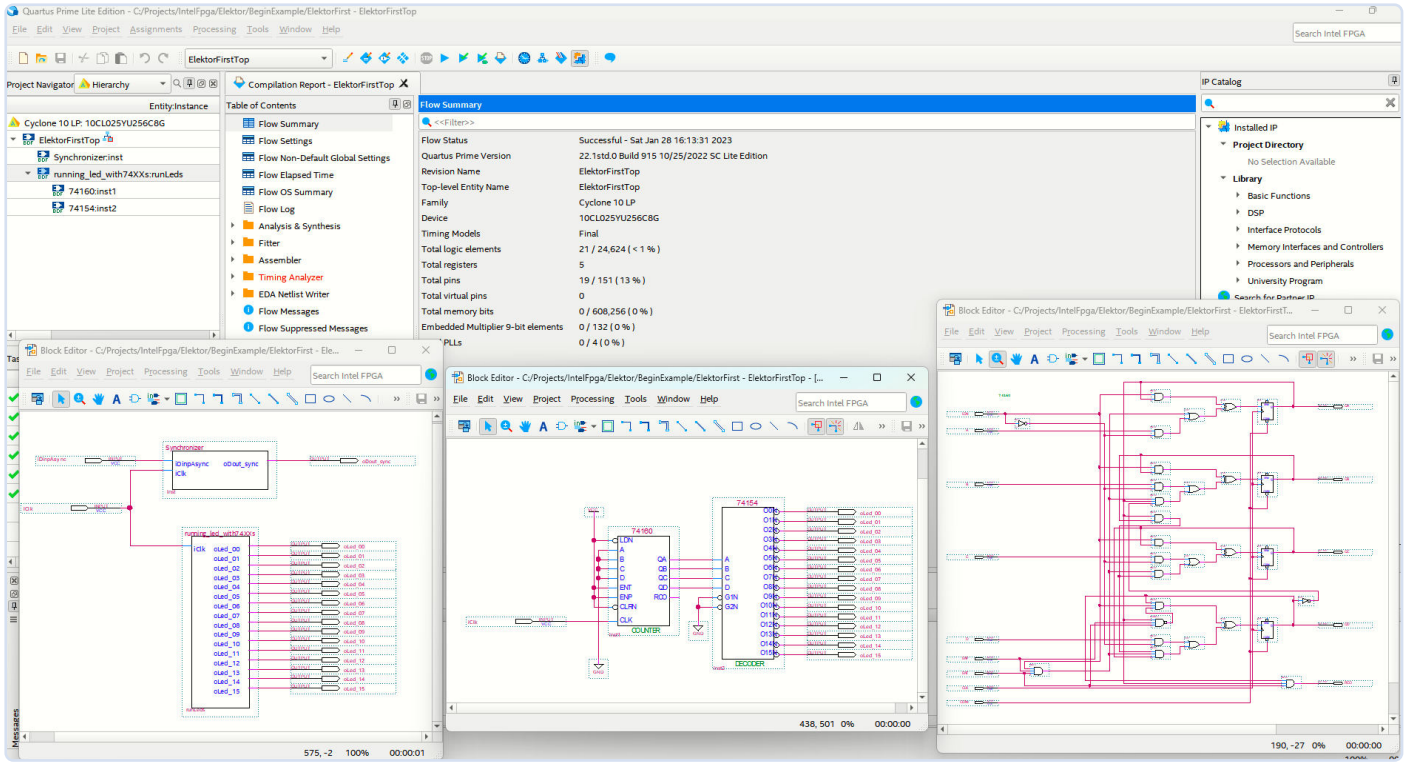


Figure 3. Quartus Lite Edition, avec l'exemple running_led_with74XXs.

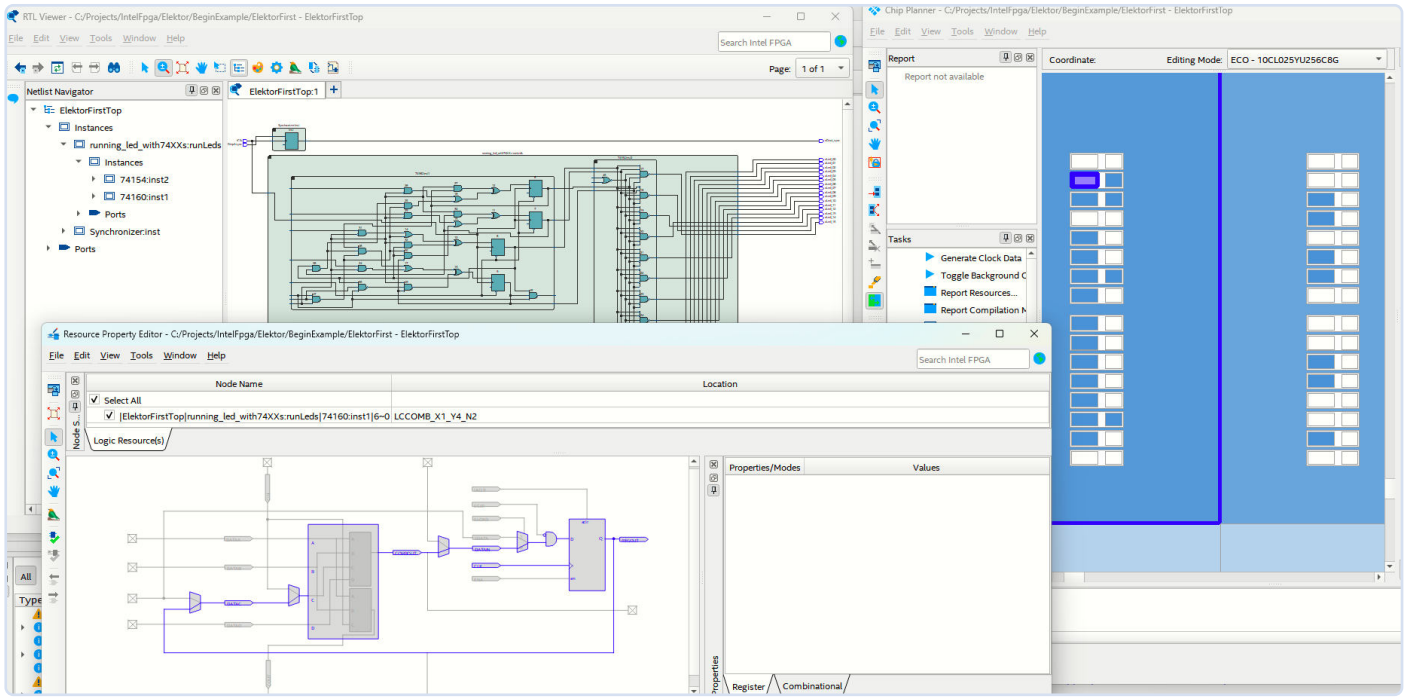


Figure 4. Quartus Lite Edition, montrant certaines des implémentations de bas niveau de l'exemple running_led_with74XXs.

Expérimenter par vous-même

N'hésitez pas à explorer et à expérimenter. Dans *ElectorFirstTop.bdf*, cliquez avec le bouton droit de la souris sur la zone schématique, choisissez *Insert Symbol* et cliquez sur le petit triangle situé à côté de la rubrique inférieure. Explorez les primitives, puis la logique, et vous découvrirez une série de composants logiques de base tels que AND, OR, XOR, etc. Allez-y et créez votre propre circuit !

De plus, pour retrouver les circuits logiques de la série 7400 dans la bibliothèque, cliquez avec le bouton droit de la souris sur la zone schématique du fichier *ElectorFirstTop.bdf*, sélectionnez *Insert Symbol* et tapez « 7400 » dans le champ *Name*. Vous trouverez une large sélection de composants de la série 7400. Vous pouvez les parcourir et prévisualiser les symboles dans le volet de droite. Il existe de nombreux circuits logiques fascinants utilisant la logique de la série 7400 disponibles en ligne, que vous pouvez émuler sur un FPGA. Ce n'est peut-être pas la méthode la plus efficace, mais cela peut certainement être une expérience divertissante et éducative ! Vous pouvez voir un exemple de circuit utilisant la logique de la série 74 à la **figure 3**.

Vous pouvez également jeter un coup d'œil au circuit complet à l'intérieur du FPGA. Dans la barre supérieure, choisissez *Tools*, *Netlist viewers*, *RTL viewer*. Vous pouvez voir le résultat à la **figure 4**. Les schémas sont visibles en haut à gauche. Pour examiner les éléments placés dans le FPGA, sélectionnez Tools, *Chip planner*. Ici, vous pouvez zoomer sur la zone contenant les cellules logiques. De plus, dans l'image, en haut à droite, une petite cellule violette est marquée. En sélectionnant une cellule logique et en cliquant dessus, vous pouvez visualiser son contenu, comme indiqué en bas à gauche.

Pour aller plus loin

Nous n'avons fait qu'effleurer la surface de la programmation FPGA en présentant la version gratuite de Quartus Prime Lite. Il offre une gamme complexe de fonctions, dont beaucoup n'ont pas pu être abordées dans cette introduction, telles que la simulation, l'analyseur logique Signal Tap, le Platform Designer, et bien plus encore. Les cartes FPGA accessibles aux débutants sont nombreuses. Pour les amateurs de logiciels libres, le projet Icestorm et l'EDI graphique Icestudio sont des points d'entrée intéressants dans ce domaine.

En outre, il existe une pléthore de ressources d'apprentissage en ligne. Des sites web comme fpga4fun [3], NandLand [4] et VHDLwhiz [5] sont d'excellents points de départ. La liste de Joel des cartes FPGA abordables [6] peut également vous aider à sélectionner le matériel adéquat.

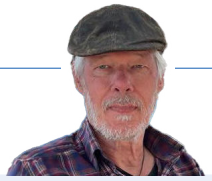
Pour les livres, la version numérique de *Free Range VHDL* de Bryan Mealy et Fabrizio Tappero est offerte gratuitement par les auteurs, et *Getting Started with FPGA*, de Russel Merrick (l'auteur des tutoriels NandLand) est également un complément intéressant à votre bibliothèque.

Le monde des FPGA est à portée de main, avec plus d'outils d'apprentissage et de communautés disponibles que jamais auparavant. C'est le moment idéal pour commencer votre voyage dans la conception numérique !

230067-04

Questions ou commentaires ?

Envoyez un courriel à l'auteur (tf.mulder@outlook.com) ou contactez Elektor (redaction@elektor.fr).



À propos de l'auteur

Theo Mulder est ingénieur en électronique. Il est expérimenté dans le traitement des signaux et a travaillé dans le domaine de l'électronique analogique et numérique, en particulier dans l'électronique médicale pour les appareils à ultrasons. Il est spécialiste du C++, des DSP et des FPGA. Il transforme les idées de recherche théoriques en application électroniques. Certains de ses propres concepts ont été brevetés. Il pense en termes de systèmes, au niveau de la carte électronique, pour trouver un bon équilibre entre le matériel analogique/numérique et le logiciel, afin de réaliser des systèmes qui fonctionnent bien avec un bon rapport prix/performance.

Produits

- **M. Dalrymple, *Microprocessor Design Using Verilog HDL* (E-book, Elektor)**
  www.elektor.fr/18518
- **Carte de développement Alchitry Au FPGA (Xilinx Artix 7)**
  www.elektor.fr/19641

LIENS

[1] Trenz Electronic — CYC1000 board: https://tinyurl.com/trenzcyc1000
[2] Intel Quartus Prime Design Software: https://tinyurl.com/downloadquartus
[3] fpga4fun: http://fpga4fun.com
[4] NandLand: https://nandland.com
[5] VHDLwhiz: https://vhdlwhiz.com
[6] Joel's list of FPGA boards: https://joelw.id.au/FPGA/CheapFPGADevelopmentBoards

MISE À JOUR :

STM32 Wireless Innovation Design Contest 2024

La rédaction d'Elektor

Avec 5 000 € en prix à gagner dans le concours *STM32 Wireless Innovation Design Contest 2024*, des innovateurs du monde entier ont travaillé dur ces derniers mois avec les solutions STM32 pour créer toutes sortes d'applications sans fil créatives. Les nominés seront annoncés en mars 2024. L'annonce des gagnants aura lieu à *embedded world* 2024.

Le concours STM32 Wireless Innovation Design Contest offre aux ingénieurs et aux électroniciens une occasion unique de démontrer leurs compétences en conception et de créer des applications sans fil basées sur les puissantes cartes de développement et d'évaluation de STMicroelectronics. Que vous soyez passionné par l'IdO, la robotique, les jeux, la domotique ou l'IA, les possibilités sont infinies. 5 000 € de prix sont à gagner !

Les gagnants : restez à l'écoute !

La date limite de soumission pour le concours était le 1er mars 2024. Au moment de la publication de ce numéro, le jury est en train d'évaluer les projets soumis. Les gagnants seront annoncés en direct lors de la conférence embedded world 2024 à Nuremberg, en Allemagne, à 17 h 00 (CEST) le mercredi 10 avril, ainsi qu'en ligne sur *elektormagazine.com/st-contest*. Si vous prévoyez de participer à la conférence embedded world 2024, venez visiter les stands d'Elektor et de STMicroelectronics (*embedded-world.de*).

Évaluation

Un jury indépendant sélectionnera les trois premiers gagnants selon les critères suivants :

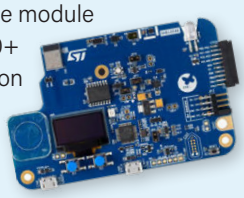
- **Créativité et innovation :** le caractère unique et l'originalité de la conception de l'application sans fil.
- **Excellence technique :** la compétence technique et la maîtrise démontrées dans l'utilisation de la carte de développement choisie.
- **Fonctionnalité et praticité :** l'efficacité et la praticité de l'application sans fil pour résoudre un problème réel ou améliorer l'expérience utilisateur.
- **Aspect visuel et expérience utilisateur :** l'attrait visuel, la conception de l'interface utilisateur et l'expérience globale de l'utilisateur de l'application.
- **Documentation et présentation :** la clarté, l'exhaustivité et la qualité de la documentation et de la présentation du projet.

Nous souhaitons bonne chance à tous les participants !

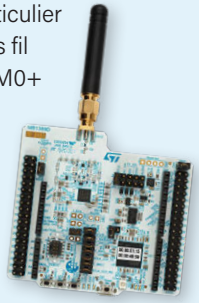
STM32 Technology

Pour participer au concours, les participants doivent réaliser un projet utilisant l'une des cartes proposées NUCLEO-WBA52CG, STM32WB-5MM-DK ou NUCLEO-WL55JC. L'idée est de tirer parti des capacités d'une carte pour concevoir et développer des applications sans fil dans n'importe quel domaine. Les participants peuvent utiliser des protocoles standardisés tels que LoRaWAN, Sigfox et Bluetooth Low Energy (BLE), ou créer leurs propres protocoles propriétaires.

- **NUCLEO-WBA52CG** est une carte Bluetooth Low Energy sans fil et ultra-basse consommation intégrant une radio puissante et ultra-basse consommation conforme à la norme Bluetooth Low Energy SIG v5.3. La prise en charge de la connectivité ARDUINO Uno V3 et les connecteurs ST morpho permettent d'étendre facilement les fonctionnalités de la plateforme de développement ouverte STM32 Nucleo grâce à un large choix de shields dédiés.



- **STM32WB5MM-DK Discovery Kit** est une plateforme de démonstration et de développement pour le module STM32W5MMG de STMicroelectronics. Ce module 32 bits à double cœur Arm Cortex-M4/M0+ intègre une radio ultra-basse consommation compatible avec Bluetooth Low Energy (BLE) 5.2, 802.15.4 avec Zigbee, Thread et des protocoles propriétaires.



- **Nucleo-WL55JC** est une carte d'évaluation pour les microcontrôleurs de la famille STM32WL, et en particulier pour le STM32WL55. Ce microcontrôleur sans fil dit sub-GHz est basé sur un Arm Cortex-M4/M0+ 32 bits à double cœur avec une fréquence d'horloge de 48 MHz. Elle se caractérise par une très faible consommation d'énergie, un émetteur-récepteur RF avec une plage de fréquences de 150 MHz à 960 MHz, 256 Ko de mémoire Flash et 64 Ko de SRAM.

Pour en savoir plus, rendez-vous sur la page web du concours STM32 Wireless Innovation Design Contest - **elektormagazine.com/st-contest** - pour plus de détails sur les projets gagnants et bien plus encore.

240010-04

Bluetooth LE avec MAUI

applications de contrôle pour Android et cie.

Tam Hanna (Hongrie)

Une application mobile pour est idéale pour contrôler vos propres appareils électroniques. La communication peut être établie via Bluetooth LE pour économiser de l'énergie. Cependant, le développement et la mise à jour de logiciels pour Android et iOS demandent un peu d'effort. C'est là que le cadre MAUI déjà présenté dans un article d'Elektor entre en jeu. Vous pouvez l'utiliser pour programmer des applications indépendamment de la plateforme. Dans cet article, nous allons vous montrer comment utiliser l'interface BLE d'un smartphone.

Le cendrier BopSync, développé par Icy Beats LLC, est contrôlé par une interface Bluetooth LE.

Ceux qui ont une connaissance approfondie du développement .NET trouveront que l'utilisation de MAUI est très pratique pour développer des applications Android et iOS - une introduction générale est disponible dans cet article [1]. Ici, nous voulons « s'approfondir » encore plus en programmant une interface Bluetooth pour Android. Il convient de noter que la bibliothèque utilisée ici est également compatible avec iOS, cependant, les modifications nécessaires dépasseraient le cadre de cet article, c'est pourquoi nous ne les aborderons pas ici.

Configuration de l'environnement de développement

Cet article est basé sur un projet pratique de l'auteur, un développeur indépendant. Un ESP32 sert de « poste distante », qui exécute un programme de contrôle basé sur l'exemple du code [2]. Visual Studio 2022 est utilisé pour développer l'application mobile ; l'exemple de projet actuel est basé sur le modèle d'application .NET MAUI.

Depuis longtemps, Microsoft a décidé de ne pas proposer « directement » une API Bluetooth sous MAUI (ou Xamarin, sur lequel MAUI est basé). C'était une sage décision, car Microsoft a dû avoir une expérience directe du processus de développement des API Bluetooth pour Android, qui était loin d'être « fluide ». Cependant, puisqu'il existe une interface qui permet à Xamarin d'accéder aux éléments natifs du système d'exploitation hôte, il est possible d'utiliser divers paquets NuGet avec des bibliothèques qui visent à compenser cette lacune. Dans les étapes suivantes, nous souhaitons nous appuyer sur la bibliothèque *Plugin.Ble* disponible sur [3]. La première étape consiste à ouvrir la console NuGet, où l'on télécharge la bibliothèque manquante.

Modification du fichier Manifest

En principe, les environnements multiplateformes ne peuvent que partiellement isoler le développeur de la plateforme sous-jacente. Pour la bibliothèque utilisée ici, cela se traduit par le fait que des modifications aux fichiers Manifest respectifs sont nécessaires pour Android et iOS (ce qui n'est pas abordé davantage ici). Ces fichiers indiquent aux systèmes d'exploitation les fonctionnalités que l'application est censée utiliser.

Pour la version 17.6.0 Preview de Visual Studio utilisée ici, un clic sur *AndroidManifest.xml* ouvre un éditeur graphique, qui n'est cependant pas encore tout à fait complet, c'est pourquoi un clic droit sur le fichier et l'ouverture du fichier *Manifest* dans un éditeur de texte est l'approche la plus pratique.

Dans l'étape suivante, l'auteur a remplacé ou ajouté certaines structures aux déclarations existantes (**listage 1**).
À cause du « système de permission » introduit dans Android 6.0, il n'est plus suffisant de modifier le fichier Manifest à ce stade. Au lieu de cela, l'application doit demander à l'utilisateur l'autorisation d'accéder aux fonctions sensibles au moment de l'exécution.
Afin d'éviter le jeu du chat et de la souris entre le système d'exécution MAUI et la gestion des nouvelles autorisations par Google, Microsoft met en œuvre une interface générale pour l'obtention des autorisations.

En principe, il s'agit d'une structure de classe qui contient une liste de constantes de permission. Pour utiliser un nouvel attribut de permission, le développeur n'a qu'à inclure la constante dans une instance *Permissions.BasePlatformPermission*. Le reste de l'interaction avec l'interface utilisateur, qui suit une procédure normalisée, est géré par le *runtime* inclus dans MAUI.
Notre prochaine tâche consiste donc à créer une nouvelle classe `Permissions.BasePlatformPermission`, qui contient les différentes permissions requises (**listage 2**).

Listage 1. Déclarations.

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android">
              <application android:allowBackup="true" android:icon="@mipmap/appicon" android:roundIcon
               ="@mipmap/appicon_round" android:supportsRtl="true"></application>
              <uses-permission android:name="android.permission.ACCESS_NETWORK_STATE" />
              <uses-permission android:name="android.permission.INTERNET" />
              <uses-permission android:name="android.permission.ACCESS_COARSE_LOCATION" />
              <uses-permission android:name="android.permission.ACCESS_FINE_LOCATION" />
              <uses-permission android:name="android.permission.BLUETOOTH" />
              <uses-permission android:name="android.permission.BLUETOOTH_ADMIN" />
              <uses-permission android:name="android.permission.BLUETOOTH_SCAN" />
              <uses-permission android:name="android.permission.BLUETOOTH_CONNECT" />
              <uses-permission android:name="android.permission.BLUETOOTH_ADVERTISE" />
              <uses-feature android:name="android.hardware.bluetooth_le" android:required="true"/>
</manifest>
```

Listage 2. Création d'une nouvelle classe Permissions.BasePlatformPermission.

```
public class BluetoothLEPermissions : Permissions.BasePlatformPermission
{
  public override (string androidPermission, bool isRuntime)[] RequiredPermissions
  {
    get
   {
     return new List<(string androidPermission, bool isRuntime)>
   {
     (Manifest.Permission.Bluetooth, true),
     (Manifest.Permission.BluetoothAdmin, true),
     (Manifest.Permission.BluetoothScan, true),
     (Manifest.Permission.BluetoothConnect, true),
     (Manifest.Permission.AccessFineLocation, true),
     (Manifest.Permission.AccessCoarseLocation, true),
        //(Manifest.Permission.AccessBackgroundLocation, true),
    }.ToArray();
    }
 }
}
```

Remarque importante : cette sélection de permissions est valable pour Android 13. Les versions plus anciennes et le Kindle Fire nécessitent un complément de permissions différent.

Recherche d'appareils BLE

Nous pouvons maintenant passer à la recherche d'appareils Bluetooth accessibles. Dans les étapes suivantes, l'auteur suppose que le lecteur est déjà familiarisé avec l'utilisation de Bluetooth LE. Une brève introduction est disponible sur [4].
Pour afficher la liste de tous les périphériques de l'environnement de l'émetteur, l'auteur utilisera une *ListView* en suivant les étapes suivantes : ouvrir le fichier *MainPage.xaml* et ajouter la ListView suivante à un emplacement convenable du code :

```
<ListView x:Name="deviceList">
  <ListView.ItemTemplate>
   <DataTemplate>
    <ViewCell>
    <StackLayout Margin="20,0,0,0"
          Orientation="Horizontal"
          HorizontalOptions="FillAndExpand">
    <Label Text="{Binding}"
          VerticalOptions="Start"
          TextColor="White"/>
    </StackLayout>
   </DataTemplate>
  </ListView.ItemTemplate>
</ListView>
```

Les développeurs habitués à Visual Basic 6 et les environnements similaires doivent changer de perspective lorsqu'ils travaillent avec des boîtes à liste MAUI. Une telle zone de liste MAUI ne consiste pas seulement en une collection d'éléments à afficher - des informations supplémentaires, nommées `ItemTemplate`, sont nécessaires pour l'affichage. Il s'agit d'un balisage XAML simple utilisé par la pile GUI pour chaque ligne de données à afficher dans la ListView. Notre exemple est simple et se contente d'afficher une étiquette à l'écran. La chaîne `Text=""` établit une liaison permettant à l'analyseur XAML d'obtenir directement les données à afficher dans l'élément.
Le fichier de balisage XAML nécessite également un bouton ordinaire qui permettra à l'utilisateur de lancer le processus de balayage.
Dans l'étape suivante, nous pouvons revenir au *Code Behind*, où nous créons quelques classes de support comme suit :

```
public partial class MainPage : ContentPage
{
   int count = 0;
   IBluetoothLE myBLE = CrossBluetoothLE.Current;
   IAdapter myAdapter =
    CrossBluetoothLE.Current.Adapter;
   public ObservableCollection<String> BTLEDevices;
```

Outre les variables requises par la bibliothèque, nous avons également besoin d'une classe `ObservableCollection`. Elle sert de « source de données » au moteur de liaison des données pour la population du ListView. Il convient de noter que le choix d'`ObservableCollection` n'a pas été fait au hasard ; il s'agit d'une classe de collection capable d'envoyer des notifications lorsque des modifications sont apportées à la base de données qu'elle contrôle.
Pour compléter la relation de liaison de données, il faut ensuite modifier le constructeur de la page principale :

```
public MainPage() {
   InitializeComponent();
   BTLEDevices = new ObservableCollection<String>();
   deviceList.ItemsSource = BTLEDevices;
}
```

Ensuite, la technique Code Behind, se charge du clic sur le bouton de recherche. Android intègre un cache de permissions, c'est pourquoi, dans la première étape, nous vérifions si notre application a déjà les permissions nécessaires pour interagir avec l'émetteur Bluetooth. Si c'est le cas, nous appelons la méthode `goScan`, qui se charge de l'exécution du scan.

```
private async void OnScanClicked
   (object sender, EventArgs e) {
   PermissionStatus status = await
     Permissions.CheckStatusAsync
     <BopSyncNetPOC.Platforms.
       Android.BluetoothLEPermissions>();
   if (status == PermissionStatus.Granted){
     goScan();
   }
```

Si la valeur de retour de `CheckStatusAsync` n'est pas positive, nous demandons la permission à l'utilisateur :

```
   else {
   await DisplayAlert("Permission required",
          "Google requires the declaration
           of this permission to perform a BTLE scan.
           Please grant it!", "OK");
   status = await
         Permissions.RequestAsync
           <BopSyncNetPOC.Platforms.
              Android.BluetoothLEPermissions>();
   if (status == PermissionStatus.Granted) {
      goScan();
   }
   else {
      await DisplayAlert("Alert",
           "Permission denied.
            Please reinstall application!", "OK");
    }
  }
}
```

Les textes descriptifs détaillés utilisés ici sont nécessaires pour convaincre l'utilisateur d'autoriser les permissions. L'expérience pratique

a montré que les utilisateurs « confrontés à des difficultés techniques », en particulier, ont tendance à refuser les demandes d'autorisation inattendues dans les fenêtres pop-up, conséquence d'une insécurité alimentée par les médias. En outre, les versions modernes d'Android enregistrent parfois les refus de manière « permanente » - si l'utilisateur clique une fois sur « No », il devra, la plupart du temps, désinstaller et réinstaller l'application avant de pouvoir autoriser à nouveau la demande de permission.
La méthode `goScan` est utilisée ensuite :

```
private void goScan() {
   if (myBLE.State == BluetoothState.On) {
     myAdapter.ScanMode = ScanMode.LowLatency;
     myAdapter.DeviceDiscovered += FoundDevice;
     myAdapter.ScanTimeout = 12000;
     //MUST be called last or ignores settings
     myAdapter.StartScanningForDevicesAsync();
   }
```

Si l'émetteur Bluetooth est déjà actif, nous paramétrons l'objet adaptateur et lançons une recherche en appelant `StartScanningForDevicesAsync`. Il est important de noter que tous les paramètres requis pour l'exécution du balayage doivent être écrits dans la classe avant l'appel de la méthode - les modifications ultérieures sont ignorées.
Si l'émetteur Bluetooth n'est pas actif, une fenêtre s'affiche à la place, invitant l'utilisateur à activer l'émetteur :

```
   else {
   DisplayAlert("Alert",
        "Please switch Bluetooth on!", "OK");
   }
}
```

Le délégué se charge de la localisation des appareils. Nous nous concentrons sur le remplissage de la liste :

```
private void FoundDevice(object sender,
                         DeviceEventArgs e){
   try
   {
     BTLEDevices.Add(e.Device.Name.ToString());
   }
   catch(Exception ex)           {
     var aD = e.Device.NativeDevice;
     //Dont muck around with the GUID in ID
     PropertyInfo aProp =
          aD.GetType().GetProperty("Address");
    BTLEDevices.Add("N/A " + aProp.GetValue(aD, null));
   }
}
```

La majorité des appareils BLE n'ont pas de nom. La routine présentée ici utilise un bloc *try-catch* pour écrire une chaîne par défaut « plus primitive » dans ListView dans ce cas.

Figure 1. La détection Bluetooth LE fonctionne.

À ce stade, le programme est prêt à être testé. La **figure 1** montre à quoi ressemble une exécution du balayage.
Il convient de noter que le placement direct d'éléments natifs dans la partie générale de la solution entraine des résultats bizarres - en particulier, des erreurs signalant l'inexistence de l'espace de nommage `Platforms.Android` se produisent. En effet, certaines opérations de Visual Studio tentent de compiler non seulement la version Android, mais aussi les divers autres projets de platesformes cibles créés dans le squelette du projet MAUI. Pour contourner ce problème, il suffit d'ajouter une protection au niveau du préprocesseur pour le code spécifique à la plate-forme :

```
private async void OnScanClicked
      (object sender, EventArgs e) {
   #if ANDROID
   PermissionStatus status . . .
  #endif
}
```

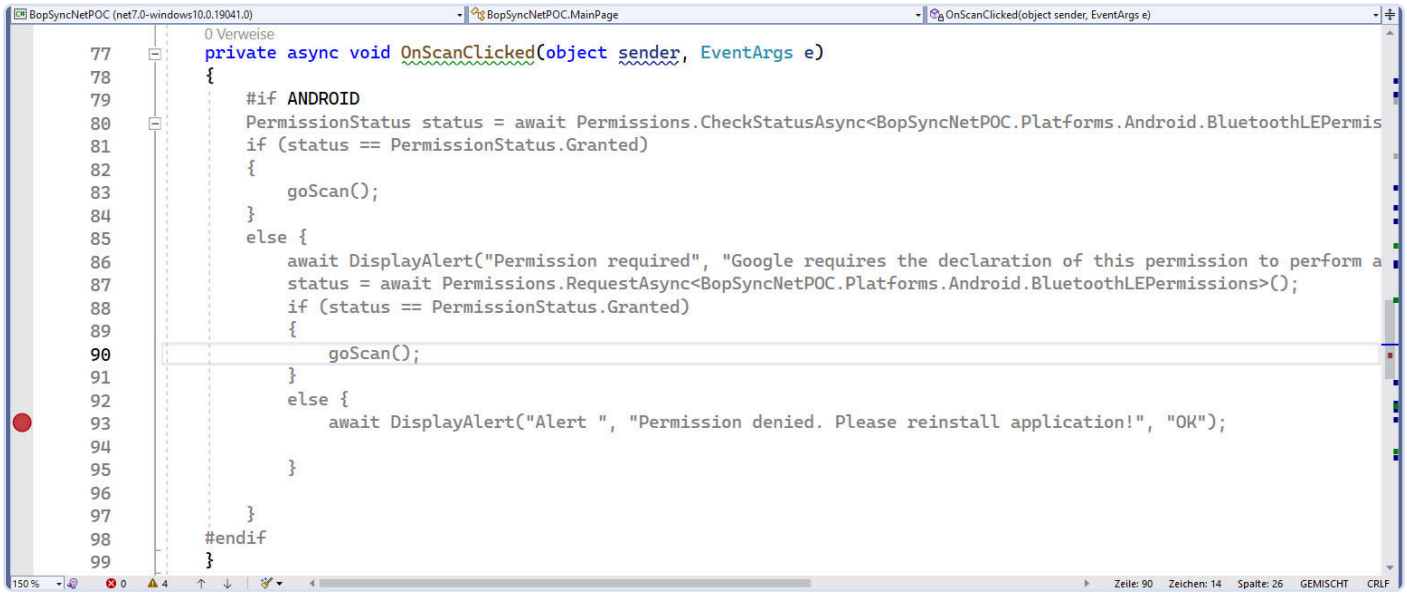


Figure 2. La bataille quotidienne : L'homme contre Visual Studio.

Il est gênant que Visual Studio rencontre des difficultés avec de tels éléments et désactive la complétion syntaxique, comme le montre la **figure 2**.

Identification et établissement de la connexion

Nous sommes maintenant prêts à contrôler le périphérique. L'auteur a ajouté à la page principale une liste supplémentaire dans laquelle il a intégré les instances individuelles de périphériques Bluetooth - elles représentent les stations distantes trouvées par le balayage Bluetooth LE.

Pour l'application de l'auteur, un seul appareil doit être traité à la fois. C'est pourquoi il est judicieux de conserver l'instance dans la classe *Application*. Ouvrez le fichier *App.xaml.cs* et insérez une variable globale :

```
public partial class App : Application
{
   public Plugin.BLE.Abstractions.
   Contracts.IService myBTLEService;
```

La qualification complète est raisonnable parce qu'il y a souvent plus d'une classe portant le nom IService dans l'environnement *.NET*. Si vous déclarez « completely », vous êtes en sécurité.

L'établissement de la connexion commence alors par l'extraction de la classe `IDevice` à partir de la liste des appareils mentionnée ci-dessus :

```
private async void deviceList_ItemSelected
  (object sender, SelectedItemChangedEventArgs e)
{
   IDevice myDevice =
   BTLEDeviceClasses[e.SelectedItemIndex];
```

L'étape suivante est l'obtention d'une instance d'appareil et l'établissement d'une liste de tous les services :

```
try
{
   await myAdapter.ConnectToDeviceAsync(myDevice);
   var services = await myDevice.GetServicesAsync();
   services = services;
   foreach (IService serv in services) {
     String aString = serv.Id.ToString();
     if (aString.CompareTo
        ("000000ff-0000-1000-8000-00805f9b34fb") == 0)
     { //Swap view
       App curApp = (App) Application.Current;
       curApp.myBTLEService = serv;
       await Navigation.
         PushModalAsync(new MainWorkPage()) ;
     }
   }
}
```

Ici, l'auteur a décidé d'identifier la poste distante en vérifiant la disponibilité d'un service spécifique. Si un service doté de l'interface graphique requise est trouvé, l'activité actuellement affichée est modifiée.

Il est important d'utiliser la méthode `Navigation.PushModalAsync`. En effet, l'auteur utilise une page dérivée de [5] dans son application commerciale - l'utilisation de la méthode normale `PushAsync`, recommandée dans la documentation de Microsoft, conduit à d'étranges plantages de la fenêtre.

Le bloc try-catch est nécessaire car il protège contre les problèmes de communication :

```
   catch (DeviceConnectionException ex)
   {
   // ... could not connect to device
   }
}
```

La tâche suivante de l'application consiste à communiquer avec les caractéristiques. Afin de réduire la logique contenue dans les vues, l'auteur s'appuie sur la structure présentée dans l'organigramme de la **figure 3**.

Si vous placez la fonction de communication dans une classe (idéalement statique), vous ne devez pas la créer dans les pages individuelles. L'auteur a de nouveau décidé d'utiliser la classe `App` dans son application ; l'étape suivante est la déclaration :

```
public async void setLightMode(int _what)
{
   var x = await myBTLEService.
      GetCharacteristicAsync(Guid.Parse
         ("0000ff01-0000-1000-8000-00805f9b34fb"));
```

La première étape consiste à utiliser la méthode `GetCharacteristicAsync`, qui détermine une caractéristique spécifique sur la base de son GUID respectif. Comme nous avons effectué une « identification » ci-dessus, l'auteur s'abstient de sauvegarder la valeur de `x` dans cet extrait de code - en pratique, il serait raisonnable de le faire.
Il convient de noter que la bibliothèque serait également capable d'effectuer une recherche « globale ». Le code nécessaire à cette tâche ressemblerait à ce qui suit :

```
public async void setLightMode(int _what)  {
  var ibx = await myBTLEService.GetCharacteristicsAsync();
  foreach (var chara in ibx) {
     var str = chara.Id.ToString();
     str = str;
  }
...
```

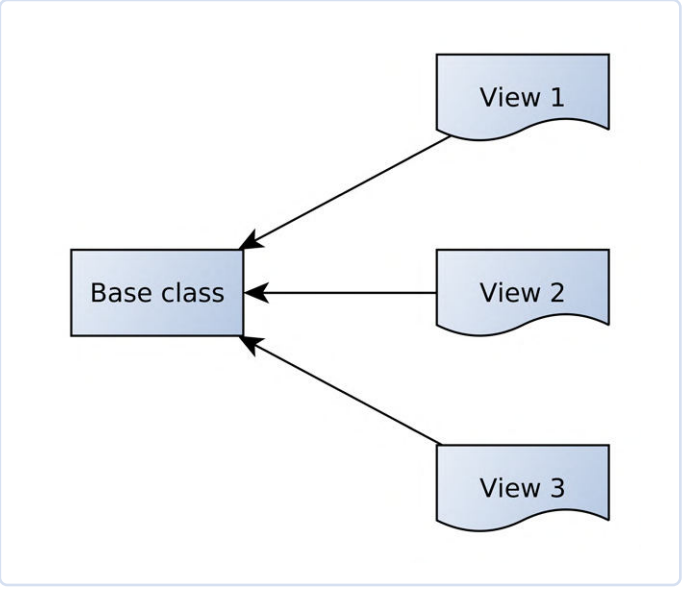


Figure 3. L'isolation de la logique de communication réduit la redondance dans le système.

L'instruction tautologique utilisée dans cet extrait est utile parce qu'elle permet de « récolter » les GUI des caractéristiques individuelles. Si vous ne pouvez pas extraire la caractéristique de l'interface graphique du programme ESP32 sous la forme d'une chaîne, vous pouvez placer un point d'arrêt sur l'instruction tautologique et exécuter le programme associé.
Pour chaque exécution, vous pouvez alors extraire les caractéristiques respectives dans le débogueur - comme le montre la **figure 4** - et (si nécessaire) les comparer aux informations affichées dans le scanner Bluetooth.

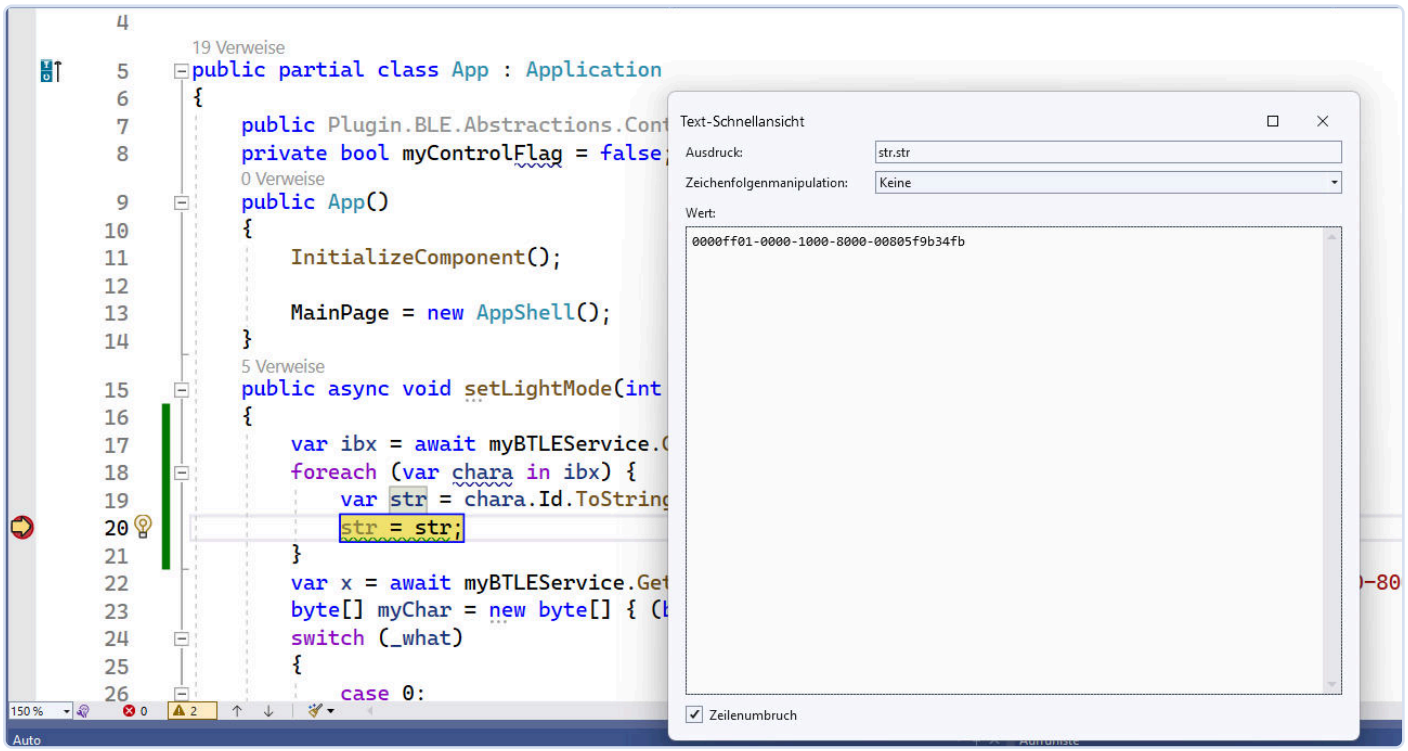


Figure 4. Les instructions tautologiques permettent d'économiser de la frappe !

Les scanners Bluetooth sont utiles

Il serait judicieux de conseiller à toute personne en charge d'une application Bluetooth LE d'installer un scanner Bluetooth LE sur son smartphone Android. L'application permet de se connecter à toutes les stations distantes et de lister les services et caractéristiques qui s'y trouvent. L'auteur utilise l'application nRF Connect dans son entreprise, qui peut être téléchargée gratuitement depuis le Play Store [7].

Pour utiliser l'application scanner, il est recommandé d'avoir un Android ou un ancien iPhone sous iOS 15. À partir d'iOS 16, des modifications ont été apportées à la pile Bluetooth, rendant les appareils moins visibles et limitant la fonctionnalité du scanner!

L'étape suivante consiste à rassembler le playload. Dans l'exemple de l'auteur, le protocole de communication propriétaire est basé sur des flux de données qui transmettent des valeurs ASCII. Le playload est ensuite rassemblé en fonction de la valeur entière fournie avec des constantes de tableau :

```
byte[] myChar = new byte[] { (byte)'0' };
switch (_what)
   {
   case 0:
     myChar= new byte[] { (byte)'0' };
     break;
   case 1:
     myChar = new byte[] { (byte)'1' };
     break;
   case 2:
     myChar = new byte[] { (byte)'2' };
     break;
   case 3:
     myChar = new byte[] { (byte)'3' };
     break;
   case 4:
     myChar = new byte[] { (byte)'4' };
     break;
...
}
```

Enfin, une commande d'établissement de la connexion est nécessaire, dont la structure est la suivante :

```
     var y = await x.WriteAsync(myChar);
   }
```

Envoi de données de l'ESP32 vers le téléphone Android

Enfin, nous voulons réaliser la transmission des données de l'ESP32 via l'interface sans fil. À ce stade, l'auteur aimerait donner un peu plus de détails sur le code ESP-IDF. Les exemples basés sur le système de tables GATT sont, selon l'auteur, très mal documentés ; il y a quelques problèmes confus et des réponses partielles dans le forum.

La question la plus importante à ce sujet est de savoir comment les caractéristiques sont initialisées. Si la constante `ESP_GATT_AUTO_RSP` est transmise, comme spécifié par défaut dans l'exemple, la pile traite les valeurs contenues dans la caractéristique.
L'application reçoit également un événement de lecture comme indiqué ci-dessous ; toutefois, la pile Bluetooth reçoit les valeurs renvoyées d'une mémoire interne et les achemine normalement avant de déclencher l'événement :

```
case ESP_GATTS_READ_EVT:
   ESP_LOGI(GATTS_TABLE_TAG, "ESP_GATTS_READ_EVT");
   break;
```

En théorie, la méthode `esp_ble_gatts_set_attr_value` fournit une fonction qui permet de modifier les valeurs de mémoire stockées dans la pile Bluetooth. Une exécution simple ressemblerait à ce qui suit :

```
void updateBTLECache() {
   esp_err_t ret;
   ret = esp_ble_gatts_set_attr_value
     (heart_rate_handle_table[IDX_CHAR_VAL_A],
        1, (const uint8_t *)"1");
}
```

Le problème avec cette méthode est qu'elle renvoie une valeur nulle même si la chaîne transmise n'est pas valide ou n'a pas encore été initialisée par la pile Bluetooth.
Cela complète déjà la « description du problème » - la configuration ou le remplissage du tableau a lieu relativement tard. Une méthode que la société de l'auteur a trouvée efficace consiste à mettre à jour les valeurs du cache BTLE à chaque fois qu'un nouveau périphérique se connecte à l'ESP32. Le code suivant est idéal à cette fin :

```
case ESP_GATTS_CONNECT_EVT:
   ESP_LOGI(GATTS_TABLE_TAG, "ESP_GATTS_CONNECT_EVT,
            conn_id = %d", param->connect.conn_id);
   updateBTLECache();
   esp_log_buffer_hex(GATTS_TABLE_TAG,
            param->connect.remote_bda, 6);
```

Enfin, voici un extrait de code qui illustre la lecture des valeurs du côté MAUI :

```
public async void setLightMode(int _what)
{
   var x = await myBTLEService.
        GetCharacteristicAsync(Guid.Parse
             ("0000ff01-0000-1000-8000-00805f9b34fb"));
   byte[] z = await x.ReadAsync();
   z = z;
...
```

Il est alors possible de traiter les valeurs fournies sous la forme d'un tableau d'octets.

Accélérer le développement

Nos essais montrent que l'API Bluetooth LE de MAUI peut interagir avec d'autres systèmes cibles facilement. Si vous avez besoin d'une application pour votre système embarqué et que vous la développez avec MAUI, vous pouvez la programmer en C# ou Visual Basic et éviter la complexité des plates-formes natives. En particulier si votre entreprise dispose déjà d'une expertise avec le cadre .NET, cela peut conduire à une accélération significative du processus de développement. Bien entendu, le contrôle Bluetooth LE sur un smartphone ne constitue que la « moitié du travail ». Dans un article publié précédemment [6], l'auteur montre comment programmer un microcontrôleur STM32 avec une interface BLE.

230381-04

Questions ou commentaires ?

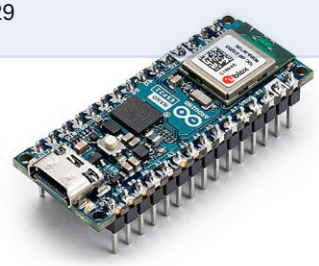
Envoyez un courriel à l'auteur (tamhan@tamoggemon.com) ou contacter Elektor (redaction@elektor.fr).

À propos de l'auteur

Ingénieur dans le domaine de l'électronique, de l'informatique et du logiciel depuis plus de 20 ans, Tam Hanna est designer indépendant, auteur de livres et journaliste (@tam.hanna on Instagram). Dans son temps libre, Tam conçoit et produit des solutions imprimées en 3D et, parmi d'autres activités, il a une passion pour le commerce et la dégustation de cigares haut de gamme.

Produits

> **Arduino Nano ESP32 avec des connecteurs**
> www.elektor.fr/20529



LIENS

[1] V. Krypczyk, « MAUI : Programmation pour PC, tablettes et smartphones «, Elektor 1-2/2024: https://elektormagazine.fr/220442-04
[2] GATT Server Service Table: https://bit.ly/45KupIU
[3] Plugin.Ble library: https://github.com/dotnet-bluetooth-le/dotnet-bluetooth-le
[4] Brief introduction to Bluetooth Low Energy (BLE): https://developer.android.com/develop/connectivity/bluetooth/ble/ble-overview
[5] Navigation.PushModalAsync method: https://github.com/dotnet/maui-samples/tree/main/8.0/Navigation/FlyoutPageSample
[6] T. Hanna, « Bluetooth LE sur le STM32 « Elektor 1-2/2024 : https://elektormagazine.fr/230698-04
[7] Application nRF Connect : https://play.google.com/store/apps/details?id=no.nordicsemi.android.mcp

carte d'extension de ports

augmentez le nombre d'E/S de votre carte de développement

Alessandro Sottocornola (Italie)

Elettronica In
WWW.ELETTRONICAIN.IT

Êtes-vous fatigué de devoir contourner le manque de ports d'E/S lors du développement de vos projets sur votre plateforme préférée ? Vous en voulez plus pour gérer une multitude de signaux et d'actionneurs ? Grâce au bus I²C, chacune de ces cartes d'extension permet d'ajouter jusqu'à 16 E/S pour atteindre un maximum de 128 sur votre système existant.

On a souvent besoin de gérer plusieurs signaux et actionneurs avec un microcontrôleur ne disposant pas de suffisamment de broches d'E/S. Dans ces situations, on peut profiter de circuits intégrés appelés « extensions d'E/S », dont la fonction est d'ajouter un certain nombre de lignes d'entrée et de sortie et de les commander via une ligne de communication série I²C ou SPI.
Afin de couvrir ce besoin de multiplier les lignes d'E/S, nous avons fabriqué une carte de liaison basée sur le Microchip MCP23017 [1], une extension d'E/S à 16 bits. Nous avons complété la carte avec le matériel minimal nécessaire à son fonctionnement, ici un commutateur DIP et quelques résistances de rappel. Enfin, pour illustrer les possibilités d'une extension d'E/S, nous vous proposons un exemple d'application où la carte de liaison pilote avec des signaux TTL ou avec des boutons une platine équipée d'une série de huit relais. Mais allons-y dans l'ordre et voyons d'abord comment fonctionne cette puce.

Le MCP23X17

Cette puce intégrée polyvalente permet une extension générique des E/S série/parallèle à 16 bits et est disponible en deux versions : celle utilisée ici, le MCP23017, équipée d'une interface de bus I²C, et le MCP23S17, une variante SPI. La puce est une extension d'E/S à 16 bits, divisée en deux ports de 8 bits chacun, interfacés via le bus I²C. C'est ainsi qu'avec seulement deux fils, référencés à la masse, on peut scruter l'état de jusqu'à 16 lignes (en mode entrée) ou de définir l'état logique de chacune d'entre elles (en mode sortie). Les lignes E/S fonctionnent par défaut en entrées.
Le MCP23017 se compose de plusieurs registres de 8 bits pour la sélection des entrées, des sorties et de la polarité. Le système principal peut activer les E/S en tant qu'entrées ou sorties en écrivant les bits de configuration correspondants (IODIRA/B). Les données de chaque entrée ou sortie sont stockées dans le registre d'entrée ou de sortie correspondant. Le registre d'inversion de polarité permet d'inverser la polarité du registre du port d'entrée. Tous les registres peuvent être lus à partir du système principal. Le port d'E/S 16 bits est structurellement composé de deux ports 8 bits, à savoir le port A et le port B, pilotés respectivement par les broches 21...28 et 1...8. Le MCP23X17 peut être configuré pour fonctionner en mode 8 bits ou 16 bits. En outre, il dispose de deux broches d'interruption, INTA et INTB, que l'on peut affecter de deux manières :

- Les broches d'interruption fonctionnent indépendamment. INTA reflète les conditions d'interruption sur le port A et INTB reflète les conditions d'interruption sur le port B.
- Les deux broches d'interruption sont actives lorsqu'une interruption se produit sur l'un ou l'autre des ports.

Schéma du circuit

On peut constater sur le schéma-bloc de la **figure 1** que la carte de liaison est très basique, puisqu'on y trouve le

circuit intégré MCP23017 en version DIP, avec toutes ses broches connectées à des barrettes (qui sur le PCB ont un pas de 2,54 mm pour l'insertion dans d'autres cartes ou platines d'expérimentation). Un commutateur DIP à trois voies (SW1 sur le schéma de câblage) permet de positionner les trois bits de poids faible de l'adresse I²C du périphérique. Les broches A0, A1 et A2 sont maintenues au niveau haut par les résistances R1 à R3 lorsque les interrupteurs sont ouverts.

Les broches d'E/S des registres A et B ont été disposées de part et d'autre afin de faciliter la connexion de notre carte. Vous pouvez connecter tout ce que vous voulez sur ces broches d'E/S numériques, dans les limites du courant et de la tension supportés par le circuit intégré MCP23017. En plus des barrettes latérales au pas de 2,54 mm, quatre autres broches ont été exposées sur une barrette nommée *I2C* (également au pas de 2,54 mm), afin de pouvoir se connecter au bus directement sur le dessus du module pour plus de flexibilité. Ces broches, qui comprennent l'alimentation positive de 5 V et la masse, sont connectées en parallèle aux broches correspondantes sur les connecteurs latéraux, et qui peuvent donc aussi être utilisées. SDA et SCL sont tous deux dotés de résistances de rappel.

Notez que sur le circuit imprimé, par commodité, les broches du registre A ont été placées d'un côté et les broches B du côté opposé, toujours dans le but de simplifier les connexions. Comme nous l'avons dit, les broches appartenant au bus I²C ont été renvoyées sur la barrette marquée ***I²C***, de même que le positif 5 V et la masse ; les deux sont dotés d'une résistance de rappel. Dans notre carte de liaison, la broche *reset* de l'extension d'E/S n'est pas utilisée ; par conséquent, pour la désactiver, nous avons placé la broche correspondante (18, RST) au niveau logique haut par l'intermédiaire de la résistance R4.

Chaque barrette latérale de la carte comporte également une duplication des lignes 5 V et GND. L'ensemble du circuit est alimenté par le contact 5 V (il y a en réalité deux contacts : 1 et 24, situés sur les côtés longs de la carte de liaison) référencé à la masse (contacts GND, c.à.d. 2 et 23 des rangées latérales).

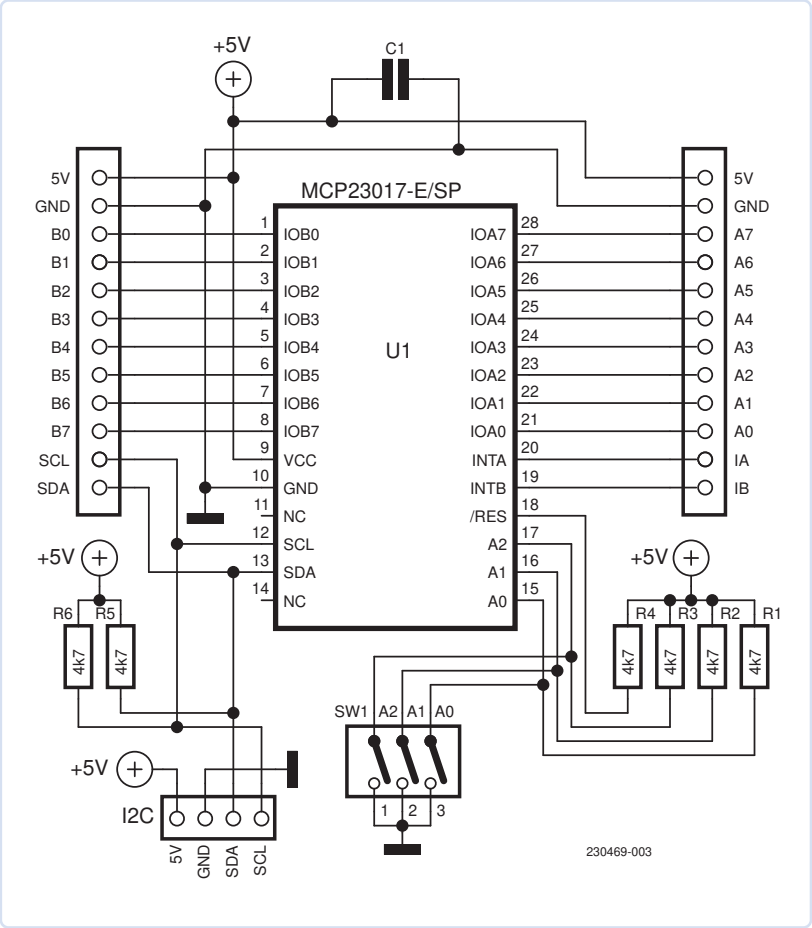


Figure 1. Schéma de la carte de liaison.

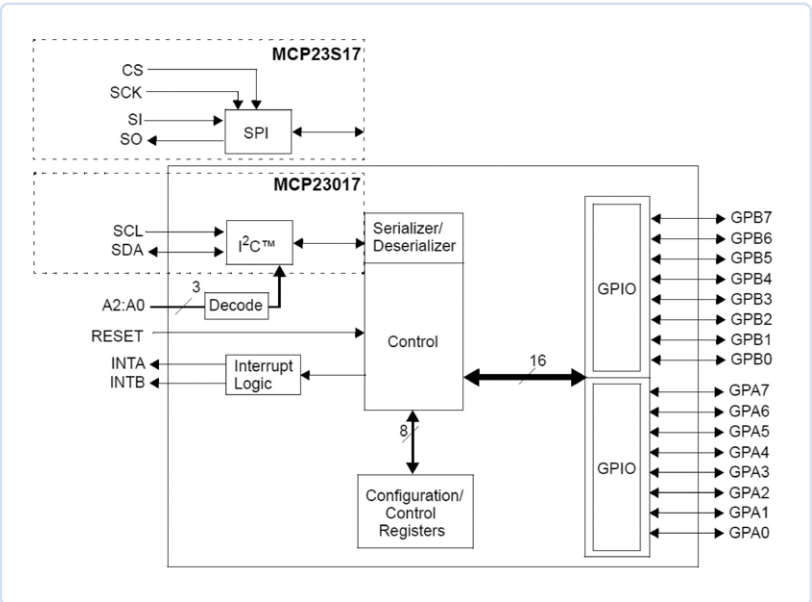


Figure 2. Schéma-bloc du MCP23017. La variante SPI, appelée MCP23S17, est également représentée. (Source : Microchip [3])

L'extension d'E/S

L'élément principal du circuit est, bien sûr, le MCP23017 fabriqué par Microchip (marqué U1), que l'on peut considérer comme un convertisseur bus I²C/parallèle. Le circuit intégré, dont on voit le schéma-bloc interne à la **figure 2**, fonctionne comme un périphérique (agent) du bus I²C et prend en charge deux modes : entrée et sortie. Dans le premier, il permet de transférer les états des E/S des registres A et B sur le bus en format série, un octet pour chaque registre, à la demande du dispositif principal du bus I²C ; dans le second, il va régler les lignes d'E/S en convertissant les données entrantes sur le bus I²C à l'état correspondant des lignes des registres A et B. La sortie d'interruption peut être configurée pour

Le CI MCP23017 en bref

La puce au cœur de la carte décrite ici est une extension d'E/S à 16 bits, divisée en deux ports de 8 bits chacun, interfacée via le bus I²C. Cela signifie qu'avec seulement deux fils, référencés à la masse, il vous permet d'acquérir l'état, ou de le définir en sortie, de pas moins de 16 lignes. Ses caractéristiques techniques sont les suivantes :

- Interface de données I²C à grande vitesse, fonctionnant à 100 kHz, 400 kHz ou 1,7 MHz
- Adresse du bus I²C réglable à 8 combinaisons
- Broches d'interruption configurables, par niveau et fonction logique
- Source d'interruption configurable Registre d'inversion de polarité

Pour les entrées :

- Entrée de réinitialisation externe
- Courant de veille de 1 µA max.
- Tension d'alimentation de 1,8 V à 5,5 V

se déclencher sous deux conditions (mutuellement exclusives) :

- Lorsqu'un état d'entrée diffère de l'état du registre du port d'entrée correspondant. Cette condition est utilisée pour indiquer au système principal qu'un état d'entrée a changé.
- Lorsque l'état d'une entrée est différent de la valeur préconfigurée du registre (registre DEFVAL).

Les lignes d'interruption INTA et INTB peuvent être configurées comme actives-hautes, actives-basses ou à drain ouvert. Le registre de capture d'interruption capture les valeurs des ports au moment où l'interruption est déclenchée, stockant ainsi la condition qui a provoqué l'interruption. La réinitialisation au démarrage (POR) ramène les registres à leurs valeurs par défaut et initialise la machine à états de l'appareil. La nécessité d'un fonctionnement bidirectionnel est due au fait que chaque périphérique du bus I²C doit à la fois être capable de lire (par exemple des commandes) et d'envoyer des données acquises sur 8+8 bits sur le bus.

Comme toutes les unités d'un bus I²C, le MCP23017 autorise le réglage de son adresse parmi une plage de huit adresses. Il dispose pour cela des broches A0, A1 et A2 qui permettent de fixer l'adresse de l'appareil pour y accéder directement à partir du bus I²C. Chacune de ces lignes est positionnée par le commutateur DIP SW1 : un DIP fermé définit l'état logique 0 sur la ligne correspondante, tandis qu'inversement un DIP ouvert définit l'état logique 1. La possibilité de définir huit adresses permet de placer jusqu'à huit extensions d'E/S sur le même bus et de commander ainsi un maximum de 128 E/S avec seulement trois lignes. Pour l'affectation de la carte de liaison à votre application, le **tableau 1** fournit la correspondance entre les adresses et le réglage des commutateurs DIP.

Tableau 1. Configuration de l'adresse de périphérique du MCP23017.

ADDR	A2	A1	A0
0x20	ON	ON	ON
0x21	ON	ON	OFF
0x22	ON	OFF	ON
0x23	ON	OFF	OFF
0x24	OFF	ON	ON
0x25	OFF	ON	OFF
0x26	OFF	OFF	ON
0x27	OFF	OFF	OFF

Avec ce matériel, la logique de fonctionnement est la suivante : chaque fois qu'il reçoit une chaîne sur la ligne SDA du bus I²C (rythmée par le signal d'horloge sur la ligne SCL), le circuit intégré MCP23017 exécute la commande qu'elle contient (dans ce cas, celle qui indique de charger l'octet de données) et dispose les huit lignes de sortie IOA0...IOA7 et IOB0...IOB7 comme les bits correspondants. Par exemple, IOA0 prendra l'état du premier bit de l'octet 1, IOA1 celui du deuxième bit, et ainsi de suite. Il en va de même pour les IOB0...IOB7, qui reproduiront exactement les bits du deuxième octet de données.

Bien entendu, la conversion et la sortie ne se produisent que si la chaîne reçue contient l'adresse du bus I²C correspondant à celle définie, via les commutateurs DIP de SW1, pour U1. À la réception de chaque chaîne, le circuit intégré met à jour l'état de ses sorties, et les niveaux logiques respectifs déterminent si les circuits en aval (par ex. des LED ou des segments d'afficheurs) sont activés ou restent éteints ; si aucune chaîne n'est envoyée par la suite, l'état de sortie conserve le dernier profil de l'octet car les sorties du MCP23017 sont maintenues. Ce qui précède s'applique au mode sortie, c.à.d. à l'écriture de l'état des deux octets du bus I²C dans les registres de sortie A et B. Si, en revanche, la commande provenant du bus est une lecture, le MCP23017 acquiert l'état E/S de chaque registre et génère deux octets, le premier contenant l'état de IOA0...IOA7 et le second celui de IOB0...IOB7 ; il les envoie ensuite comme réponse sur le bus I²C.

Réalisation pratique

Bien, maintenant que nous avons décrit le schéma de câblage, nous pouvons passer aux instructions de construction. Nous proposerons ensuite un exemple d'application basé sur l'interfaçage avec une carte Arduino accompagné du croquis correspondant. Comme d'habitude, nous avons dessiné un circuit imprimé dont nous mettons à disposition pour téléchargement les deux couches (il s'agit d'un circuit double face) à la page de ce projet sur Elektor Labs [2]. À partir de ces schémas, vous pouvez procéder à la préparation du circuit imprimé par photogravure. Après l'avoir gravé et

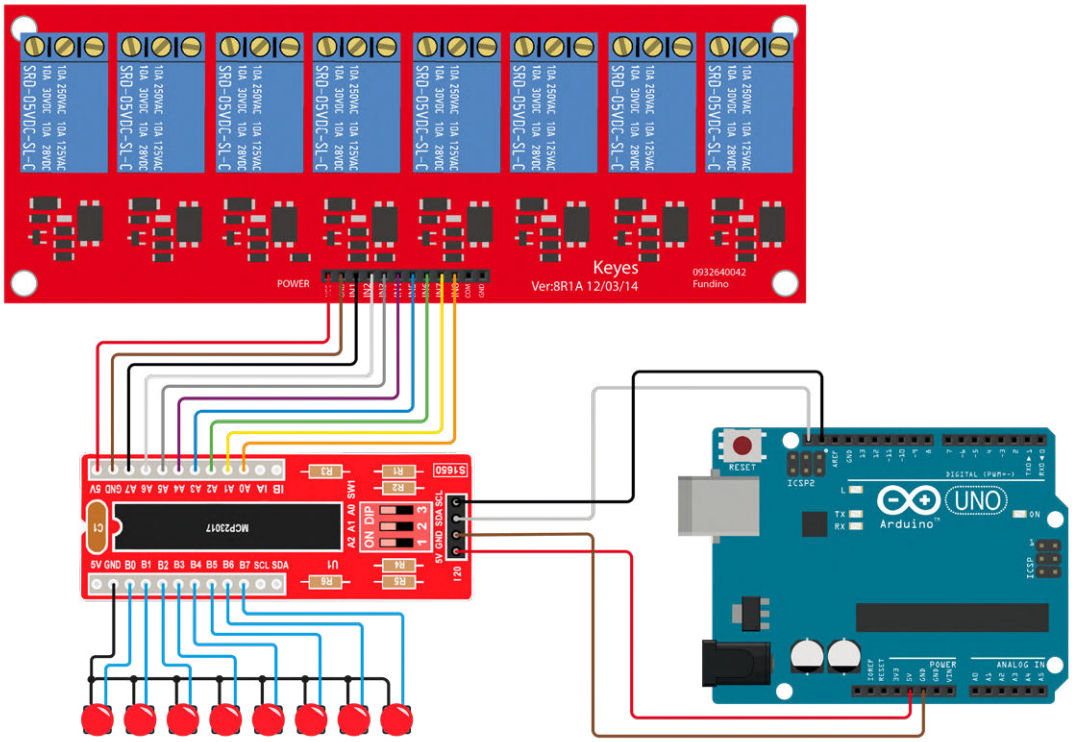


Figure 3. Schéma de câblage de l'application de commande de relais actionnés par boutons-poussoirs.

Liste des composants

Résistances

R1…R6 : 4,7 k

Condensateur

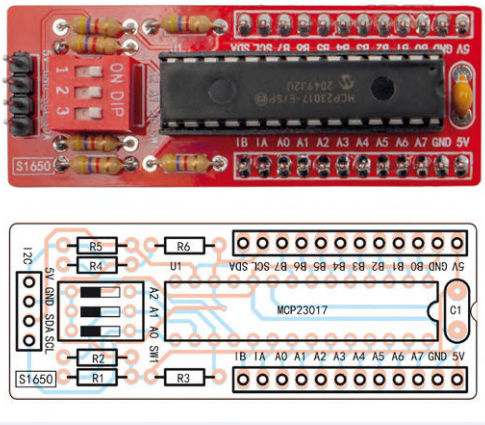
C1 = 100 nF, céramique

Semi-conducteurs

U1 = MCP23017-E/SP

Divers

SW1 = Commutateur DIP à 3 voies
Embase DIL 2 x 14 broches
2 x barrettes mâles à 12 broches
1 x barrette mâle à 4 broches
1 x circuit imprimé (voir texte)



percé, vous pouvez assembler les quelques composants nécessaires, qui, dans ce projet, sont tous traditionnels, de type traversant, pour ceux qui n'aiment pas trop la technique CMS. Commencez par insérer et souder les résistances, puis passez à l'embase pour le circuit intégré (à placer avec l'encoche orientée comme indiqué dans le plan de montage que vous voyez dans cet article) et au commutateur DIP à trois voies, à monter avec l'interrupteur 1 orienté vers la gauche, en regardant la carte avec l'embase pour U1 sur le dessus.

Enfin, insérez et étamez dans les trous respectifs la barrette à quatre broches marquée *I2C*, puis, de l'autre côté du PCB, insérez et étamez deux rangées de barrettes à 12 broches qui permettront le montage sur des platines ou l'insertion sur d'autres cartes, par exemple pour finalement se connecter à Arduino en utilisant de classiques cavaliers mâle/femelle. Une fois les composants soudés,

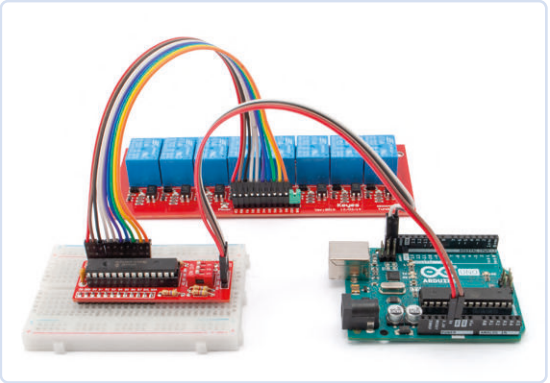


Figure 4. Le matériel pour tester le croquis d'exemple.

Listage 1. Projet de démonstration

```
#include <Adafruit_MCP23X17.h>.
#include <Adafruit_MCP23X17.h>
Adafruit_MCP23X17 mcp;
int i = 0;
int OUT[] = ;      //Represents MCP23017 PIN (A7...A0)
int IN[] = ;       //Represents MCP23017 PIN (B0...B7)
int STATO[] = ;    //For each output, every toggle the status is being saved
void setup()
{
 Serial.begin(9600);
 Serial.println("MCP23017 INPUT/OUTPUT");
 if (!mcp.begin_I2C(0x20))         //0x20 is MCP23017's address with A0=A1=A2 > ON(GND)
 {
    Serial.println("MCP Error!");  //If MCP is not found, the error is visualized
    while (1);
 }
 //bank A Pin set as outputs and B bank as inputs
 //The STATO variable to 0 to indicate idling outputs
 for (i=0; i<8; i=i+1)
 {
    mcp.pinMode(OUT[i], OUTPUT);
    mcp.pinMode(IN[i], INPUT_PULLUP);
    STATO[i] = 0;
 }
}

//******************* L O O P **********************
void loop()
{
 String Testo_Debug = "";
 for (i=0; i<8; i=i+1)
 {
    //If button pressed or output not activated, I activate it
    if ((mcp.digitalRead(IN[i])==0) && (STATO[i]==0))
    {
      STATO[i] = 1;
      Testo_Debug = "Pulsante " + String(i+1) + " premuto";
      Serial.println(Testo_Debug);
      mcp.digitalWrite(OUT[i], HIGH);
    }
    //If button released and output is active,I de-activate it
    if ((mcp.digitalRead(IN[i])==1) && (STATO[i]==1))
    {
      STATO[i] = 0;
      Testo_Debug = "Pulsante " + String(i+1) + " rilasciato";
      Serial.println(Testo_Debug);
      mcp.digitalWrite(OUT[i], LOW);
    }
 }
 delay(10);
}
```

insérez le MCP23017 dans son embase, en le tenant avec l'encoche orientée comme indiqué dans le plan de montage sur ces pages. Ceci fait, votre carte de liaison est prête pour l'expérimentation ou le prototypage.

Utilisons-la avec Arduino

La carte de liaison a été créée pour être reliée à un microcontrôleur, étant donné qu'on utilise généralement les extensions d'E/S avec des appareils équipés d'une interface série avec le bus I²C. Comme Arduino prend en charge ce bus, nous avons créé un exemple de code téléchargeable à [2] pour lire et gérer les E/S du MCP23017 via Arduino. Ce croquis permet essentiellement d'écrire l'état du registre du Port A en fonction d'un octet envoyé par Arduino sur le bus, dont les bits correspondent à l'état lu sur le Port B, qui sert cette fois d'entrée. Pour donner à l'exemple une application concrète, nous avons décidé d'utiliser les états logiques des E/S du Port A, qui fonctionneront ici comme des sorties numériques, pour piloter une carte à relais. Plus précisément, nous devons connecter les huit lignes de commande de sortie des relais d'une carte à relais à 8 canaux à la banque d'E/S du port A, tandis que huit boutons-poussoirs normalement ouverts doivent être connectés au port B, avec en commun le pôle connecté à GND. Pour réaliser cette application, il est nécessaire de connecter Arduino UNO, la carte de liaison, la carte à relais et les boutons, comme le montre le schéma de câblage proposé à la **figure 3**, tandis que la **figure 4** montre le prototype réel de cette application. Puisqu'il s'agit de boutons poussoirs assez communs (normalement ouverts) et qu'ils n'ont pas d'électronique externe, les résistances de rappel internes du MCP ont été activées (via la bibliothèque) pour les gérer et reconnaître le changement d'état, de sorte que nous avons activé la sortie respective lorsque le bouton est amené à la masse (GND).

Pour réaliser ce petit projet de démonstration, un code simple basé sur l'Arduino UNO a été écrit, en tirant parti de la bibliothèque Adafruit, qui, comme vous le voyez dans le **listage 1**, est incluse à la première ligne du croquis.

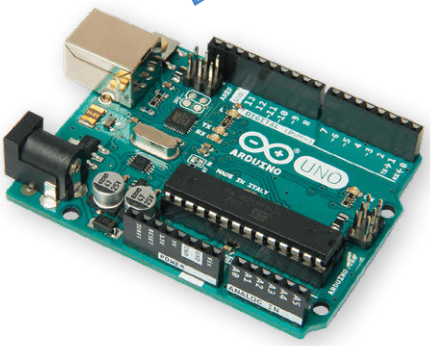
Avant de charger le code dans la mémoire programme de notre carte Arduino, il est essentiel de télécharger la bibliothèque à partir de *www.adafruit.com* et de l'installer en utilisant le gestionnaire de bibliothèque inclus dans l'EDI, ou simplement d'extraire le contenu du fichier ZIP et de copier le dossier *Adafruit_MCP23017_Arduino_Library* dans le répertoire *libraries* que l'on trouve normalement dans le système d'exploitation dans *Documents\Arduino\libraries*. Après avoir chargé la bibliothèque, il suffira de charger notre code d'exemple et de le télécharger dans la carte après avoir choisi le port COM correct dans le menu *Outils* de l'EDI.

Dans le code, un octet contenant l'état des boutons est envoyé au MCP23017 et les données correspondantes sont traitées puis écrites dans un octet dirigé vers le circuit intégré, qui définira l'état des lignes du port A de manière stable, jusqu'au rafraîchissement. Pour que l'interfaçage fonctionne, les commutateurs DIP A0, A1 et A2 doivent être réglés correctement, car si l'adresse 0x20 n'était pas attribuée au capteur, un message d'erreur serait affiché sur le port série ; l'adresse de la carte de liaison est attribuée avec la combinaison 000 des trois bits A0, A1, A2 (c'est-à-dire en fermant les trois commutateurs DIP à la masse). Si vous souhaitez modifier l'adresse, reportez-vous au tableau 1, étant entendu que vous devez également modifier l'adresse indiquée dans le croquis

VF : Denis Lafourcade — 230469-04

Produits

- **Arduino UNO Rev3**
  www.elektor.fr/15877



LIENS

[1] Page Web du MCP23017 de Microchip : https://microchip.com/en-us/product/mcp23017
[2] Ce Projet sur Elektor Labs : https://tinyurl.com/9sh3ct5t
[3] Fiche technique de Microchip : https://tinyurl.com/yc39n93v

carte spéciale IA

apprentissage automatique avec le Jetson Nano

Tam Hanna (Hongrie)

L'intelligence artificielle sur les GPU n'est pas nécessairement synonyme de cartes graphiques coûteuses et de consommation d'énergie énorme. NVIDIA s'oriente depuis un certain temps vers des systèmes plus petits, notamment en raison de la concurrence avec les fabricants de microcontrôleurs. Dans cet article, nous examinons le Jetson Nano et une petite application de démonstration.

L'accélération de l'intelligence artificielle (IA) dans les systèmes embarqués est un marché en pleine évolution. Des entreprises telles que Canaan et Maxim Integrated se livrent une lutte acharnée pour la suprématie. ARM s'aventure également sur ce terrain avec l'annonce du Cortex M52. L'objectif est de proposer de petits systèmes IA qui exécutent des tâches d'IA de base en périphérie, sans connexion à Internet. L'architecture système est ainsi beaucoup plus stable car les tâches de ML ou d'IA peuvent être poursuivies même si la connexion au serveur principal est perdue. NVIDIA essaie de se positionner dans ce domaine depuis un certain temps avec la série Jetson.

Par souci de transparence, il convient de noter que les cartes personnalisées de ces systèmes ne sont pas vraiment réalisables pour les petites entreprises, ne serait-ce qu'en raison de l'extrême largeur de bande nécessaire pour accéder à la mémoire vive DDR. La société NVIDIA est consciente de ce problème. Son aperçu de portefeuille (disponible sur [1]), qui donne aux électroniciens une vue d'ensemble de tout l'écosystème Jetson, comprend donc plusieurs « cartes de calcul », telles que le Jetson TX2 présenté dans **la figure 1**.
Il convient également de noter qu'avec la prédominance de NVIDIA dans le secteur des GPU, l'écosystème tiers bien développé a également découvert la gamme de produits Jetson. Une vue d'ensemble de divers produits tiers est disponible à l'adresse [2], dont l'intégration dans une solution interne peut faire gagner beaucoup de temps de développement (pensez, par exemple, à la conception du boîtier et à la sélection des caméras, etc.).

Démarrage du système et fonctionnement

Pour les amateurs, NVIDIA propose le kit de développement NVIDIA Jetson Nano présenté dans la **figure 2** (à côté d'un Raspberry Pi et d'un Orange Pi 5+). Au moment de la rédaction de cet article,

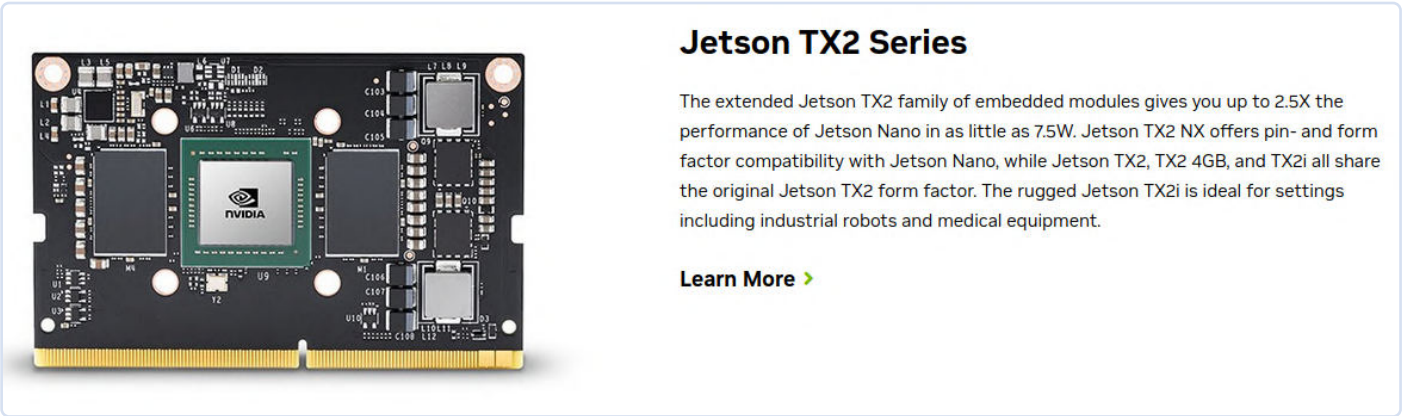


Figure 1. L'utilisation de ce module prêt à l'emploi facilite grandement l'intégration du Jetson dans des circuits propriétaires (source : NVIDIA).

le meilleur prix d'OEMSecrets (voir [3]) était de 155 €. Bien que légèrement plus cher qu'un Raspberry Pi, la technologie NVIDIA est mieux adaptée à l'écosystème général de l'IA. Lors de l'achat du NVIDIA Jetson Nano, il est recommandé de commander également un bloc d'alimentation avec un connecteur d'alimentation aux dimensions habituelles de 5,5/2,1 mm. Dans les étapes suivantes, l'auteur utilise un MeanWell GST25E05-P1J.
Un examen attentif du système (voir aussi la **figure 3**) montre qu'il s'agit d'un produit en deux parties. Outre la carte de support, qui contient les différentes interfaces, on trouve le module de calcul lui-même avec le dissipateur thermique. Il est nécessaire d'insérer une carte microSD contenant le système d'exploitation dans le module de calcul.

Micro-USB est également possible en théorie

Si vous disposez d'une alimentation Micro-USB très puissante et que vous n'utilisez pas le port Micro-USB pour connecter du matériel externe, vous pouvez également choisir cette méthode d'alimentation. Cependant, en raison de « conflits » avec le Raspberry Pi, l'auteur, comme un enfant brûlé, redoute le feu et se contente d'une alimentation CC classique.

Le SoC intégré est un système multicœur doté de quatre cœurs Arm Cortex A57, en plus de l'accélérateur d'IA lui-même. Il en résulte une configuration qui rappelle les ordinateurs de traitement classiques fonctionnant généralement sous Linux embarqué. NVIDIA recommande d'utiliser une carte MicroSD d'une capacité d'au moins 32 Go et d'une vitesse minimale de UHS1. Le fichier image est disponible à l'adresse [4], vous pouvez l'extraire sur votre carte mémoire avec un lecteur de carte comme à l'accoutumée.
Vous aurez également besoin d'une souris et d'un clavier USB pour le démarrage initial ; le Jetson prend en charge à la fois HDMI et DisplayPort pour la sortie écran. Nous supposerons qu'un câble Ethernet est également connecté dans les étapes suivantes.

Pas pour les maladroits !

Le support microSD du module Jetson est basé sur le principe du « form-fit ». Il est inséré et retiré en l'enfonçant - si vous utilisez une pince électronique et un tournevis fin, vous pouvez effectuer l'opération sans démonter la carte d'évaluation.

L'absence de caméra s'est avérée problématique lors de l'installation. Au lieu d'un capteur CCD, NVIDIA utilise deux connecteurs ordinaires des Raspberry Pi 4 et cartes similaires. Il est possible de connecter certaines caméras Raspberry Pi directement au Jetson. Dans les étapes suivantes, l'auteur utilise une caméra Raspberry Pi 2 connectée au port CAM0 via un câble FPC conçu pour le Raspberry Pi. Il faut noter que la version du câble destinée au Raspberry Pi 5 n'est pas compatible avec le Jetson. Différents modèles 3D pour connecter la caméra sont disponibles chez ThingiVerse, et leur impression simplifiera la tâche aux développeurs.

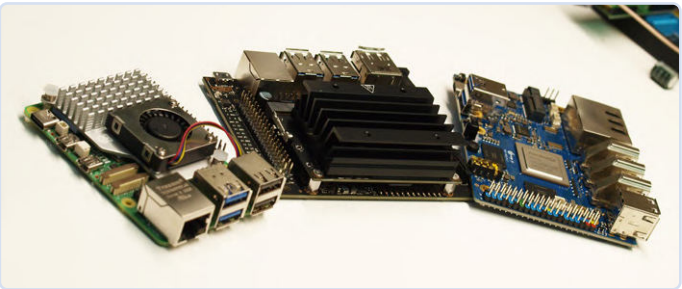


Figure 2. Le Jetson à côté d'un Raspberry Pi 5 et d'un OrangePi 5 Plus.

Figure 3. Le desserrage de deux vis permet de démonter le kit de développement.

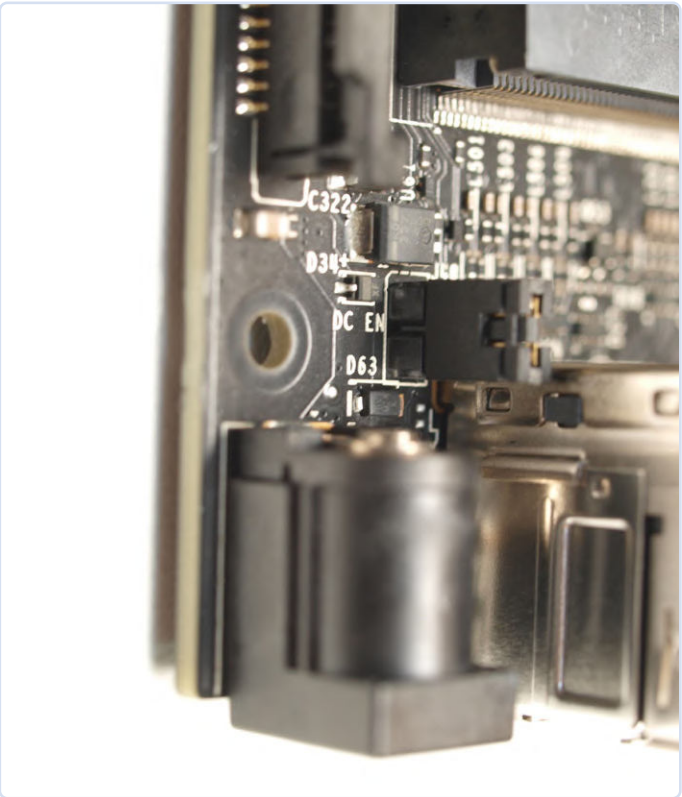


Figure 4. Ce cavalier est crucial.

Si vous souhaitez alimenter le NVIDIA Jetson avec l'alimentation MeanWell susmentionné, vous devez éviter un petit piège de débutant. Le cavalier montré branché dans la **figure 4** n'est pas branché à la livraison : dans ce cas, le connecteur DC n'est alors pas relié à l'alimentation. Lorsque l'alimentation est connectée, la LED d'état ne s'allume pas, ce qui peut prêter à confusion.
Lors du premier démarrage, l'ordinateur allume immédiatement l'écran HDMI - la reconfiguration de la carte microSD et d'autres initiations prennent un certain temps. Le système présente ensuite un bureau Ubuntu et vous demande de suivre l'assistant de bienvenue.
En général, il s'agit de l'assistant d'installation Ubuntu habituel, mais NVIDIA a ajouté une étape pour approuver les licences logicielles internes, entre autres choses. L'assistant de sélection du profil d'alimentation est important : il est recommandé de garder l'option par défaut. Elle garantit que le Jetson reçoit la puissance maximale.
Une fois cette étape terminée, il y en aura d'autres avant que la carte Jetson ne soit prête à l'emploi - des invites peuvent également apparaître sur le bureau pour redémarrer le système.

Premiers essais

NVIDIA s'appuie sur un système Ubuntu 18.04 plus ou moins compatible avec le Jetson. L'auteur aime configurer de tels processeurs pour l'accès à distance afin d'épargner la peine de passer du clavier du PC à celui de l'ordinateur de traitement.
Cependant, l'application *Settings* d'Ubuntu n'est pas adaptée à cette tâche, c'est pourquoi nous exécutons `sudo apt-get update` et `sudo apt-get upgrade` pour mettre à jour l'inventaire de paquets lors de la première étape. Toute requête doit être validée en appuyant sur *Enter* ; vous devez autoriser le redémarrage du système Docker.
Vous devez ensuite redémarrer le Jetson. En entrant `sudo apt install vino`, vous vous assurez que le serveur VNC est prêt à être utilisé.
Enfin, les commandes ci-dessous sont nécessaires pour que la configuration soit utilisable. Bien entendu, vous devez remplacer la chaîne de caractères `thepassword` par un mot de passe adapté.

```
tamhan@tamhan-desktop:~$ mkdir -p ~/.config/autostart
tamhan@tamhan-desktop:~$ cp /usr/share/applications/
vino-server.desktop ~/.config/autostart
tamhan@tamhan-desktop:~$ gsettings set org.gnome.Vino
prompt-enabled false
tamhan@tamhan-desktop:~$ gsettings set org.gnome.Vino
require-encryption false
tamhan@tamhan-desktop:~$ gsettings set org.gnome.Vino
authentication-methods "['vnc']"
tamhan@tamhan-desktop:~$ gsettings set org.gnome.Vino
vnc-password $(echo -n 'thepassword'|base64)
```

Après le redémarrage suivant, il est possible d'accéder au Jetson à l'aide du client VNC Reminna si un utilisateur est connecté. Notez toutefois que l'applet d'accès à distance dans l'application *Settings* ne fonctionne toujours pas.
Ensuite, vous pouvez effectuer un test initial de la connexion de la caméra en entrant la commande suivante :

```
tamhan@tamhan-desktop:~$ gst-launch-1.0 nvarguscamerasrc
! nvoverlaysink
```

L'aperçu de la caméra qui peut être activé avec cette commande de base n'apparaît que sur un moniteur connecté à la carte Jetson - le système communiquant avec l'ordinateur via VNC voit plutôt la sortie du terminal. Pour mettre fin au processus, il suffit d'envoyer un événement d'interruption en appuyant sur *Control* + *C*.
L'accès à la caméra est alors - en général - assuré par des méthodes connues du PC ou de la station de travail. Le fichier [5], qui montre la configuration d'un pipeline basé sur Open CV, est particulièrement intéressant.
Les commandes suivantes sont nécessaires pour le rendre exécutable localement :

```
tamhan@tamhan-desktop:~$ git clone https://github.com/
JetsonHacksNano/CSI-Camera
tamhan@tamhan-desktop:~$ cd CSI-Camera/
tamhan@tamhan-desktop:~/CSI-Camera$ python3 simple_
camera.py
```

La fenêtre de la caméra ouverte est alors également visible via VNC, car les informations ne sont pas envoyées directement au framebuffer du GPU Tegra.
Si la fenêtre de la caméra est à l'envers, vous pouvez modifier le paramètre `flip_method` dans le fichier :

```
def show_camera():
    window_title = "CSI Camera"
    print(gstreamer_pipeline(flip_method=2))
     video_capture = cv2.VideoCapture(gstreamer_
            pipeline(flip_method=2), cv2.CAP_GSTREAMER)
    if video_capture.isOpened():
    . . .
```

Interagir avec les broches d'E/S (via Python)

Les systèmes d'IA nécessitent des compétences complètement différentes de la part du développeur, qui ne correspond généralement que très peu au développement classique des systèmes embarqués. Dans la pratique, les personnes extérieures au domaine, ayant une formation en informatique, maîtrisent souvent mieux l'IA que les développeurs de systèmes embarqués. La partenaire de l'auteur, qui programme très efficacement en Java, résout les tâches d'IA plus rapidement - mais elle n'a que peu de connaissances en langage assembleur.
Le but de ce bref aperçu de l'architecture logicielle est de souligner que nous décrivons intentionnellement l'accès aux GPIO sur le Jetson en Python dans les étapes suivantes. La raison en est que la plupart des développements de systèmes d'intelligence artificielle orientés vers l'utilisateur sont faites avec Python : si l'on s'appuie ici sur le langage C, on se retrouve avec une interface native.
La distribution standard de Python n'inclut pas la gestion de paquets sur le Jetson, c'est pourquoi vous devez entrer les commandes

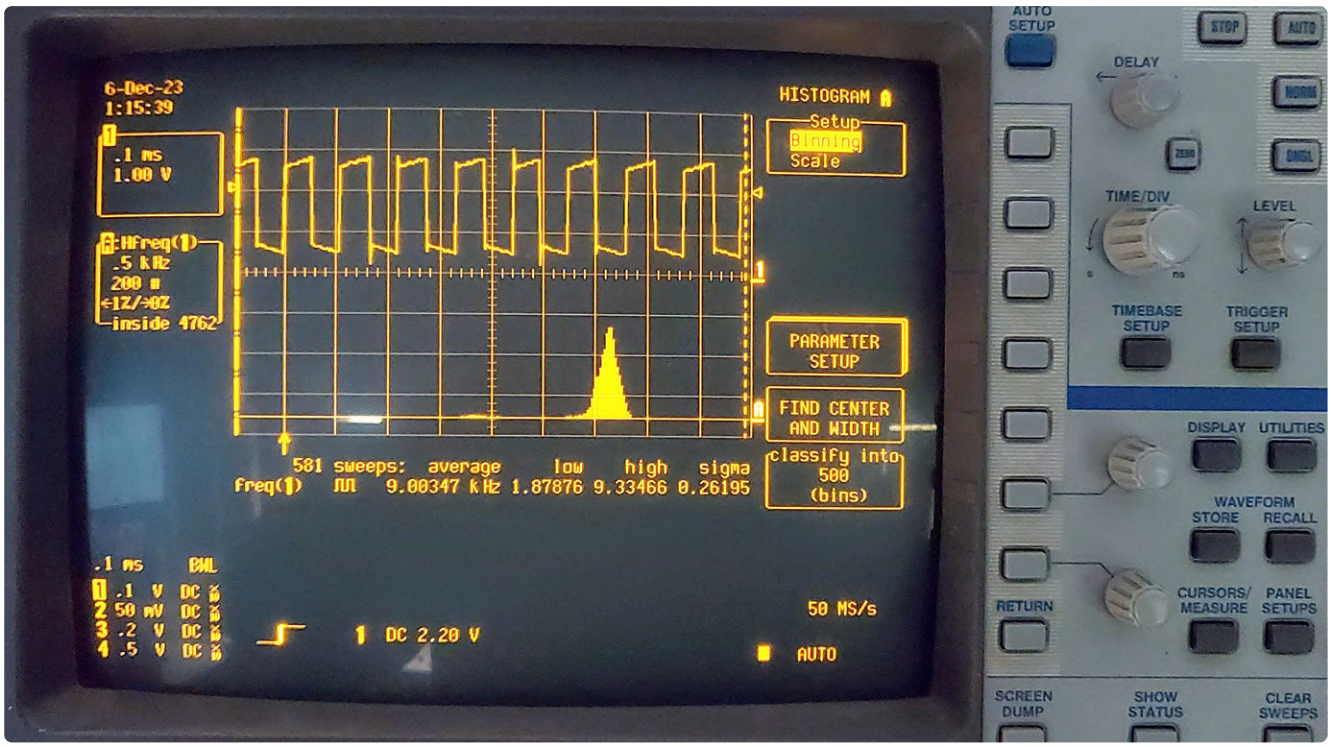


Figure 5. Le Jetson de NVIDIA produit des ondes carrées avec Python.

`sudo apt install python-pip` et `sudo apt install python3-pip`. L'étape suivante consiste à vérifier que les modules GPIO ont été correctement installés. Deux commandes sont également nécessaires ici, car les environnements Python ne sont (bien entendu) pas vraiment capables de partager leur référentiel de bibliothèques :

```
tamhan@tamhan-desktop:~$ sudo pip install Jetson.GPIO
tamhan@tamhan-desktop:~$ sudo pip3 install Jetson.GPIO
```

Nous ne pouvons pas présenter l'API GPIO Jetson dans son intégralité dans la limite de cet article. Sous [6], vous trouverez des exemples prêts à l'emploi qui expliquent l'API.
Le programme suivant est suffisant pour notre test :

```
import RPi.GPIO as GPIO
import time

output_pin = 18  # BCM pin 18, BOARD pin 12

def main():
    GPIO.setmode(GPIO.BCM)
# BCM pin-numbering scheme from Raspberry Pi
    GPIO.setup(output_pin, GPIO.OUT, initial=GPIO.HIGH)

    print("Starting demo now! Press CTRL+C to exit")
    try:
        while True:
            GPIO.output(output_pin,  GPIO.HIGH)
            GPIO.output(output_pin,  GPIO.LOW)
            GPIO.output(output_pin,  GPIO.HIGH)
            GPIO.output(output_pin,  GPIO.LOW)

    finally:
        GPIO.cleanup()

if __name__ == '__main__':
    main()
```

Ce qui est particulièrement intéressant ici, c'est que NVIDIA propose au développeur Jetson débutant plusieurs façons d'adresser les broches - nous utilisons ici la fonction `GPIO.BCM`, qui est basée sur le brochage du Raspberry Pi. La récompense de nos efforts est la capture d'écran présentée dans la **figure 5** - la stabilité de la fréquence n'est peut-être pas élevée, mais elle est plus que suffisante pour exécuter des fonctions IdO.
Il convient également de noter que l'exécution d'exemples de programmation de GPIO sans droits de superutilisateur peut parfois poser des problèmes. Voir [7] pour plus de détails à ce sujet.

Essais avec la fonction ML

En théorie, la disponibilité d'un système d'exploitation Linux complet et d'une interface USB3 (très rapide) facilite la réalisation de tests. Un exemple possible serait le modèle Stable Diffusion : en pratique, il est possible d'utiliser le Jetson comme générateur d'images à l'aide d'un runner (modifié).
Cependant, cette approche n'est pas recommandée dans la pratique

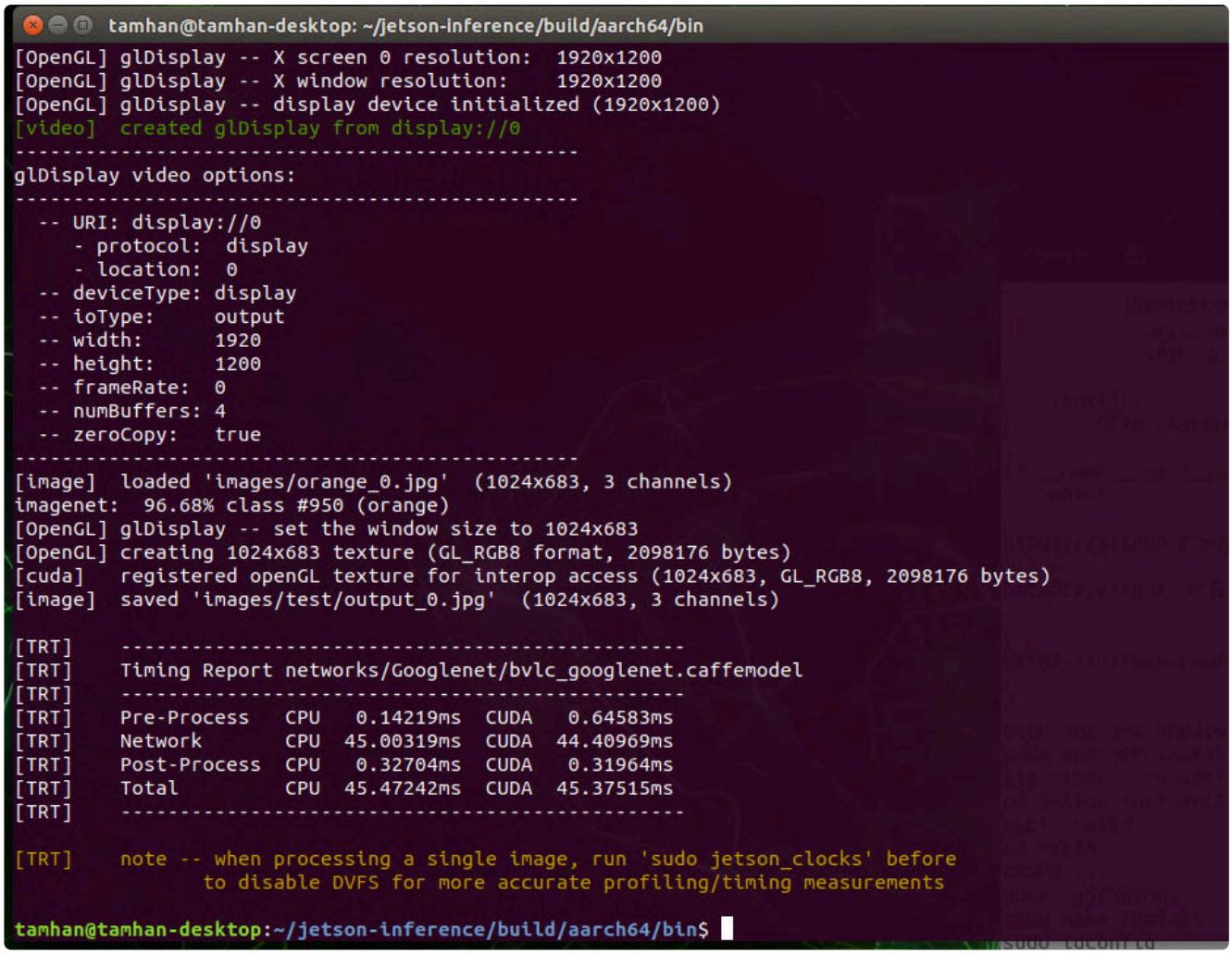


Figure 6. Le programme de démo fourni avec le Jetson n'est pas très « communicatif » au niveau de la ligne de commande.

: tant la mémoire graphique relativement faible de seulement 4 GB VRAM que le faible nombre de cœurs 128 (seulement) rendent la génération d'images lente. Sur Reddit, on trouve des rapports d'expérimentateurs de ML qui estiment jusqu'à 5 heures de calcul par image avec une taille de cluster de 512 × 512 pixels.

Un coup de pouce pour les frameworks les plus répandus

NVIDIA s'efforce de faciliter autant que possible l'implémentation de divers frameworks ML largement utilisés par les développeurs. Une liste des produits pris en charge, y compris des instructions d'installation optimisées, est disponible à l'adresse suivante : [8].

Il en va de même pour les solutions « de bout en bout » qui sont souvent présentées - l'entraînement d'un modèle nécessite tellement de puissance de calcul et de ressources que NVIDIA recommande de l'externaliser sur un ordinateur de bureau ou un ordinateur central dans la plupart des tutoriels - le Jetson est ensuite configuré avec les tailles des modèles prêt à l'emploi.
Sur [9], vous trouverez des exemples plus ou moins prêts à l'emploi qui illustrent les performances du Jetson de manière simple.
Pour utiliser ce modèle « powershow », vous devez télécharger et implémenter un logiciel NVIDIA. En théorie, l'utilisation d'un conteneur Docker est également possible ici - pour ceux qui ne sont pas familiers avec la technologie des conteneurs, la compilation locale est une alternative, qui peut être réalisée en entrant les commandes suivantes :

```
sudo apt-get update
sudo apt-get install git cmake libpython3-dev
python3-numpy
git clone --recursive --depth=1 https://github.com/
dusty-nv/jetson-inference
cd jetson-inference
mkdir build
cd build
cmake ../
make -j$(nproc)
sudo make install
sudo ldconfig
```

N'hésitez pas à répondre (par oui ou non) à la question concernant l'installation des composants d'entrainement - l'auteur a répondu Non dans les étapes suivantes pour économiser de la mémoire sur sa carte microSD, dont la taille n'est que de 32 GB.
Une fois la compilation terminée avec succès, vous trouverez une structure de projet relativement complexe dans le répertoire `~/jetson-inference/build/aarch64$/bin`, qui contient des

fichiers Python ainsi que divers fichiers binaires. Il est intéressant de noter que NVIDIA y a même placé des exemples de test prêts à l'emploi.
Tout d'abord, nous voulons utiliser le classificateur - il analyse les fichiers d'images et détermine ce qu'il faut voir dans l'image fournie :

```
tamhan@tamhan-desktop:~/jetson-inference/build/aarch64/
bin$ ./imagenet.py images/orange_0.jpg images/test/
output_0.jpg
```

L'exécution de cette commande pour la première fois prend un peu plus de temps car les modules fournis sont optimisés pour les besoins du système Jetson. Ensuite nous voyons les infos de timing dans la **figure 6**.
Pour visualiser les résultats du processus de ML, visualisez le fichier image généré - la commande suivante ouvre le dossier de sortie directement dans le gestionnaire de fichiers Nautilus :

```
tamhan@tamhan-desktop:~/jetson-inference/build/aarch64/
bin$ nautilus images/test/output_0.jpg
```

La récompense de vos efforts est l'image d'écran montrée dans la **figure 7**.

Analyse du fichier Python

Jetons maintenant un coup d'œil rapide à l'exemple de programme que nous venons d'utiliser. Le code source est accessible en entrant la ligne de commande suivante :

```
tamhan@tamhan-desktop:~/jetson-inference/build/aarch64/
bin$ gedit imagenet.py
```

Même à première vue, il est clair que la bibliothèque utilisée ici est liée au classique *ImageNet* - NVIDIA regroupe plusieurs systèmes d'IA couramment utilisés avec le kit de démarrage Jetson.
Le cœur de l'exemple de programme est une boucle infinie qui

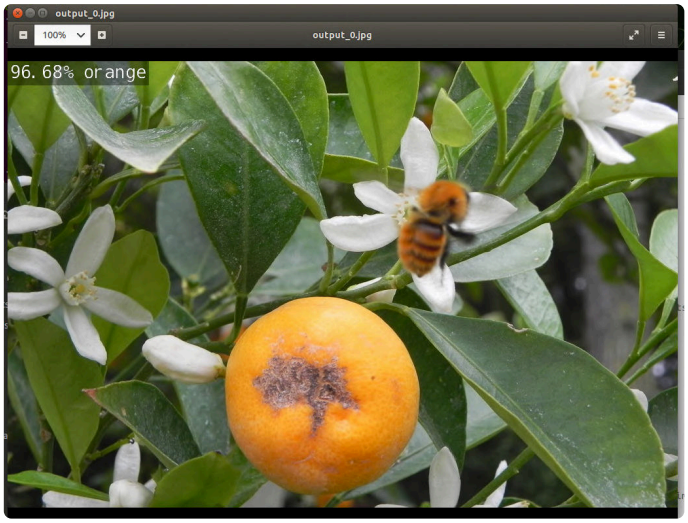


Figure 7. L'exemple de programme de NVIDIA s'est avéré être un détecteur de fruits exemplaire.

reçoit une image du flux d'entrée, la transmet au modèle de ML et affiche enfin les infos « calculées » (**listage 1**).
Pour cela, l'initialisation des flux de données et du modèle à utiliser est effectuée comme suit :

```
net = imageNet(args.network, sys.argv)
input = videoSource(args.input, argv=sys.argv)
output = videoOutput(args.output, argv=sys.argv)
font = cudaFont()
```

Un autre aspect intéressant est d'obtenir ou de compléter le paramètre `network`, qui fournit le nom du modèle à traiter. Ici, NVIDIA s'appuie sur la classe `ArgParser`, qui - normalement - se spécialise dans le traitement des paramètres fournis via la ligne de commande.
Dans le cas de cette déclaration, une valeur par défaut est introduite qui active et charge GoogLeNet :

```
parser.add_argument("--network", type=str,
default="googlenet", help="pre-trained model to load
(see below for options)")
```

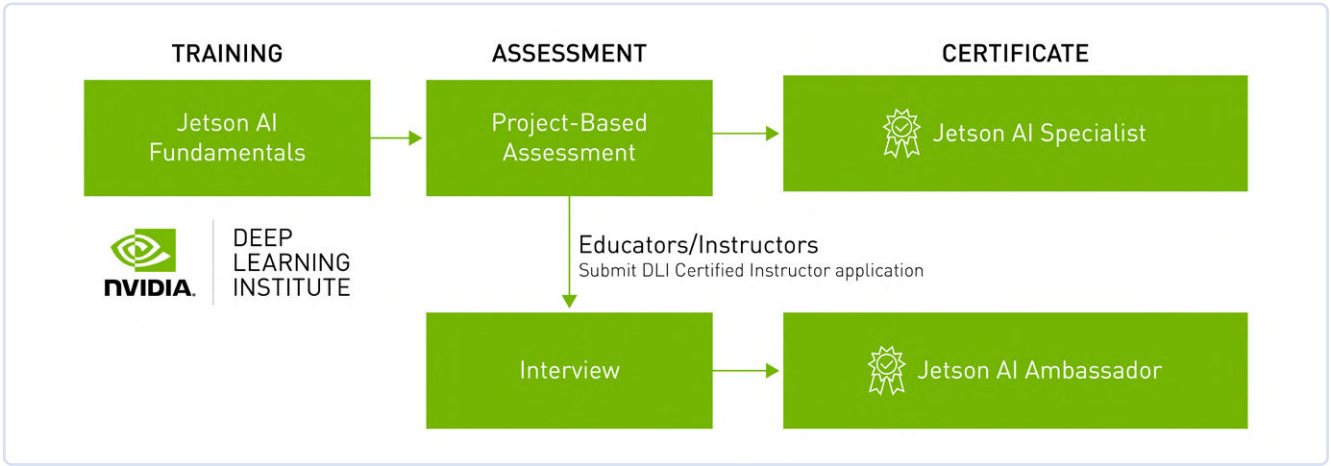


Figure 8. Apprentissage à deux voies de NVIDIA (source : NVIDIA [10])

Listage 1. Classification.

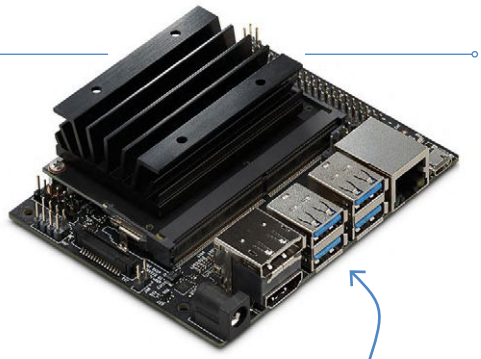
```
while True:
    img = input.Capture()

    if img is None: # timeout
        continue
        predictions = net.Classify(img, topK=args.topK)
        for n, (classID, confidence) in enumerate(predictions):
            classLabel = net.GetClassLabel(classID)
            confidence *= 100.0

            print(f"imagenet: % class # ()")

            font.OverlayText(img, text=f"% ",
                             x=5, y=5 + n * (font.GetSize() + 5),
                             color=font.White, background=font.Gray40)

    output.Render(img)
```



Produits

- **NVIDIA Jetson Nano Developer Kit (B01)**
  www.elektor.fr/19001

Note : formation (guidée) en ligne avec une certification

Le succès de l'ancienne Union soviétique dans plusieurs pays arabes et africains peut être en partie attribué au partenariat étroit en matière de formation - ce que le cadet apprend est ce qu'il utilise plus tard dans son travail. NVIDIA est clairement conscient de cette situation, c'est pourquoi la certification Jetson AI - comme le montre la **figure 8** - consiste en un cours de ML entièrement gratuit en deux parties. Il convient également de souligner que Nvidia offre un certificat récompensant l'achèvement de ce cours. Si vous êtes intéressé, nous vous encourageons à visiter le site [10]. Vous y trouverez de plus amples informations sur la manière d'organiser au mieux votre participation à la formation NVIDIA.

Avantages de Linux

Avec le Jetson, NVIDIA lance un hybride dans cette lutte entre tous les acteurs du domaine de l'IA. D'une part, les microcontrôleurs basse consommation dédiés, tels que le Maxim MAX78000, consomment beaucoup moins d'énergie. D'autre part, ces contrôleurs ne prennent pas en charge CUDA : un modèle pouvant fonctionner sur un PC ou un ordinateur central doit donc être modifié avant de pouvoir être utilisé avec ces puces dans des applications IdO.

En revanche, le NVIDIA Jetson ne constitue pas un GPU à part entière : tant en termes de consommation d'énergie que des versions CUDA prises en charge (CUDA 11 ne fonctionne pas), le module n'est pas un substitut à part entière d'un RTX 4000.

En fin de compte, l'implémentation est réussie si un modèle qui fonctionne sans problème sur un ordinateur ou une unité centrale doit être utilisé avec peu d'efforts et s'il dispose de suffisamment de ressources fournies par le Jetson. La disponibilité d'un système d'exploitation Linux signifie que relativement peu de travail de conversion est nécessaire pour un scientifique des données - se familiariser avec les API intégrées utilisées par Maxim et consorts demande beaucoup plus d'efforts. Le coût plus élevé du matériel peut donc être amorti rapidement et efficacement, en particulier pour les petites séries.

230740-04

Questions ou commentaires ?

Envoyez un courriel à l'auteur (tamhan@tamoggemon.com), ou contactez Elektor (redaction@elektor.fr).

LIENS

[1] Aperçu du portefeuille : https://www.nvidia.com/en-eu/autonomous-machines/embedded-systems/
[2] Liste de divers produits tiers : https://developer.nvidia.com/embedded/ecosystem
[3] Meilleur prix d'OEMSecrets : https://www.oemsecrets.com/compare/%20945-13450-0000-100
[4] Téléchargement du fichier image : https://developer.nvidia.com/jetson-nano-sd-card-image
[5] Fichier du pipeline basé sur un Open CV : https://github.com/JetsonHacksNano/CSI-Camera/blob/master/simple_camera.py
[6] Exemples prêts à l'emploi : https://github.com/NVIDIA/jetson-gpio/tree/master/samples
[7] Exécution d'exemples de programmes GPIO sans droits de superutilisateur : https://github.com/NVIDIA/jetson-gpio/issues/20
[8] Liste des produits pris en charge : https://elinux.org/Jetson_Zoo
[9] Exemples prêts à l'emploi : https://github.com/dusty-nv/jetson-inference
[10] Cours et certificats Jetson AI : https://developer.nvidia.com/embedded/learn/jetson-ai-certification-programs

2024 l'Odyssée de l'IA

premiers essais avec TensorFlow

Brian Tristam Williams (Elektor)

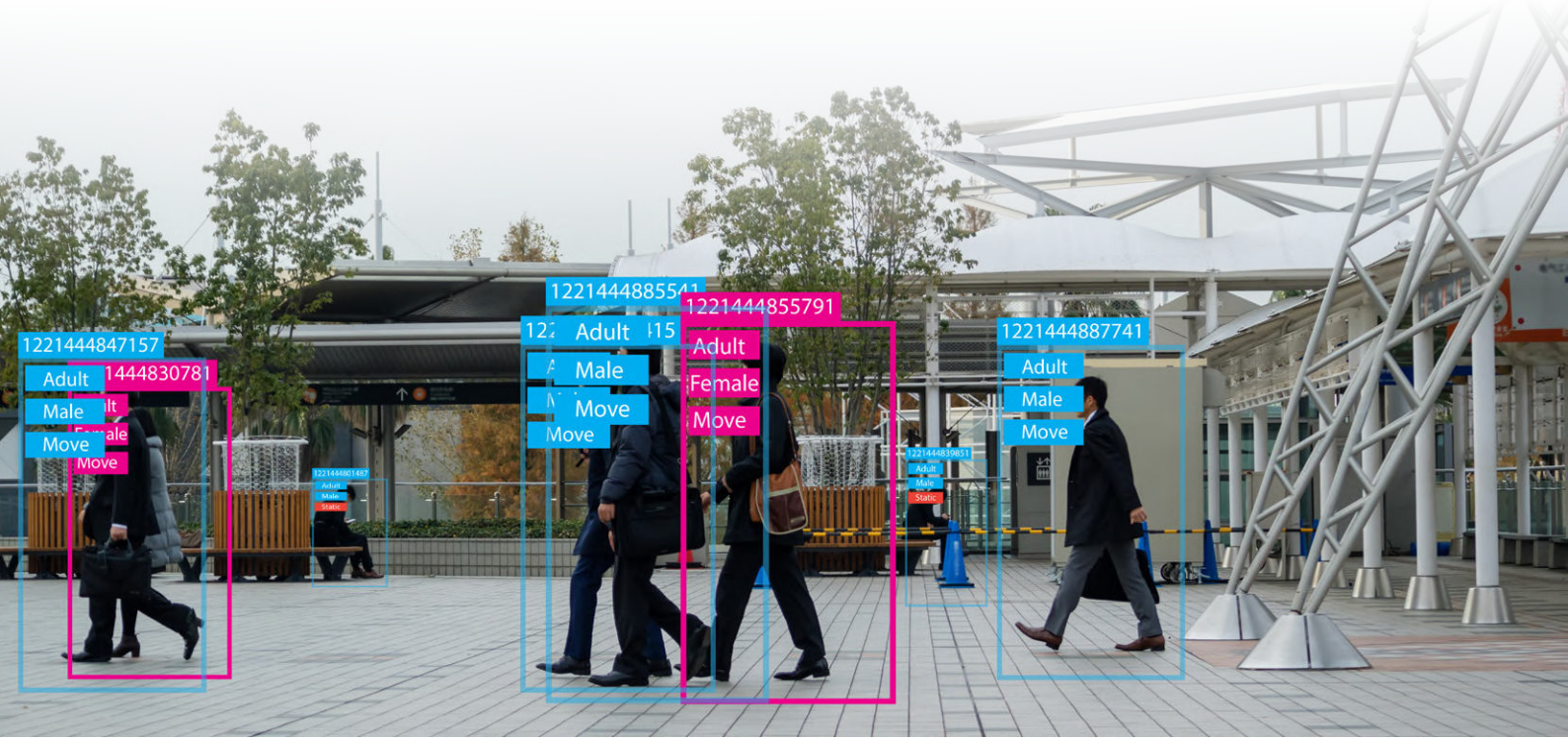
En explorant le potentiel de TensorFlow Lite sur le Raspberry Pi, cet épisode se penche sur les aspects pratiques de l'IA dans un environnement compact, en testant les limites de ce qui est possible sur des appareils de faible puissance.

TensorFlow Lite

TensorFlow : quèsaco ?

Développé par l'équipe Google Brain, TensorFlow est une bibliothèque open-source pour le calcul numérique et l'apprentissage automatique (en anglais : machine learning), jouant un rôle essentiel dans l'écosystème plus large de l'IA, qui inclut des techniques d'IA générative telles que ChatGPT. Il s'agit d'une pierre angulaire de l'apprentissage profond, qui permet de créer des modèles capables de traiter et d'apprendre des « données massives » (big data).

TensorFlow et les modèles d'IA générative sont souvent complémentaires, TensorFlow fournissant la plateforme fondamentale pour créer et entraîner une variété de modèles IA, y compris ceux qui pourraient alimenter les applications génératives. Sa polyvalence lui permet de s'adapter à un large éventail de tâches, allant de simples classifications à des processus décisionnels complexes, ce qui en fait un choix populaire dans les domaines académiques et industriels. Sa capacité à traiter de grands ensembles de données, à reconnaître des modèles et à en apprendre le rend un outil puissant dans le cadre plus général de l'IA générative, qui se concentre sur la création de nouveaux modèles de contenu et de données.



TensorFlow Lite : moins c'est plus

Dans ce nouvel épisode de notre voyage dans l'IA, j'emprunte un chemin quelque peu inhabituel. Alors que j'ai accès à un PC assez performant équipé d'un GPU RTX 4070 approprié et de suffisamment de RAM, je me trouve attiré par le monde de TensorFlow Lite [1] sur le Raspberry Pi. Il y a un certain charme dans le défi d'optimiser les processus d'IA pour un appareil aussi compact et moins puissant. Mais il ne s'agit pas seulement de faire plus avec moins ; il s'agit de créer des solutions intelligentes, autonomes, portables et efficaces.

Parfois, le projet à réaliser ne nécessite pas, ou n'est pas en mesure de supporter, l'encombrement et la puissance d'une tour de bureau. Dans de tels cas, le Raspberry Pi constitue une alternative parfaite. L'exécution de TensorFlow Lite sur cette petite mais performante carte permet de rendre l'IA plus accessible et adaptable à divers environnements, que ce soit à des fins éducatives, pour des projets amateurs ou pour des applications pratiques où l'encombrement et la consommation d'énergie sont primordiaux.

Avec l'essor de l'*edge computing* (informatique en périphérie), le besoin d'outils d'IA à la fois puissants et efficaces s'est accentué. C'est là que TensorFlow Lite, une version plus légère et plus efficace de TensorFlow, entre en jeu, notamment pour des appareils tels que le Raspberry Pi. Le Raspberry Pi étant un nano-ordinateur abordable, mais puissant, fournit la plateforme parfaite pour exécuter TensorFlow Lite, apportant des capacités d'apprentissage automatique au bout des doigts des amateurs, des éducateurs et des professionnels.

L'intégration de TensorFlow Lite dans un petit ordinateur monocarte offre des avantages significatifs pour les applications d'IA sur des systèmes plus grands. TensorFlow Lite est optimisé pour fonctionner sur du matériel léger, ce qui le rend adapté à un large éventail d'applications pratiques autres que la détection d'objets. Il s'agit notamment du traitement de données en temps réel, de la prise de décision locale dans les appareils IdO et de l'exécution de modèles IA pour des tâches telles que la reconnaissance des gestes, les capteurs environnementaux et la surveillance de la santé. Les fabricants de Raspberry Pi peuvent ainsi déployer des modèles d'IA complexes de manière rentable et accessible, ce qui ouvre un monde de possibilités pour l'innovation et l'exploration de l'IA, même dans le cadre de petits projets.

Dans les parties suivantes, nous allons voir les étapes de l'installation de TensorFlow Lite sur un Raspberry Pi et nous préparer aux futures applications innovantes qu'il permet de réaliser.

Pour commencer

J'ai un tiroir plein de Raspberry Pi, allant du O.G. en passant par le Zero jusqu'au 5, mais je vais utiliser un Raspberry Pi 4 pour ce projet. C'est un bon choix en termes d'accessibilité pour nos lecteurs, parmis les trois, que je vois encore en vente de temps en temps. (D'ailleurs, je n'ai qu'un seul Raspberry Pi 5).

Donc, voici ce que j'ai pour commencer :

- **Raspberry Pi 4 Model B :** Nous allons voir comment il fonctionne avec ses 4 Go de RAM. (J'ai utilisé la commande `free -h` pour me rappeler la quantité de mémoire embarquée).
- **Carte microSD de 32 Go :** Je ne sais pas si j'ai nécessairement besoin d'autant d'espace, mais c'est la première que j'ai trouvée quand j'en ai cherché une.

Figure 1. Raspberry Pi Imager dispose d'une interface simple et conviviale.

- **Raspberry Pi OS Lite (64-bit) :** Je ne suis pas vraiment fan de la complexité introduite par les interfaces graphiques, et je préfère le oui/non ou on/off rassurant que j'obtiens de l'interface texte, il s'agit d'un portage de Debian Bookworm sans environnement de bureau. Ayant longtemps travaillé avec des serveurs web, je n'ai jamais vraiment trouvé d'utilité à une interface graphique sous Linux. Faites-moi savoir si je devrais me mettre à la page !

J'ai installé la version 1.8.5 de Raspberry Pi Imager, la version disponible au moment de la rédaction de cet article depuis [2]. Il suffit de télécharger la version correspondant à votre système d'exploitation et de l'exécuter. C'est vraiment génial pour simplifier le processus d'installation. Il suffit de choisir la version de votre carte Raspberry Pi, le système d'exploitation que vous souhaitez, et la carte que vous avez connectée au lecteur de carte de votre ordinateur. L'interface utilisateur est vraiment simple (**figure 1**). Mieux encore, cela fonctionne. J'ai vérifié mon numéro de version une fois installé, en utilisant `cat /etc/os-release`. Le vôtre peut être différent, mais voici le résultat que j'ai obtenu :

```
briantw@raspberrypi:~ $ cat /etc/os-release
PRETTY_NAME="Debian GNU/Linux 12 (bookworm)"
NAME="Debian GNU/Linux"
VERSION_ID="12"
VERSION="12 (bookworm)"
VERSION_CODENAME=bookworm
ID=debian
HOME_URL="https://www.debian.org/"
SUPPORT_URL="https://www.debian.org/support"
BUG_REPORT_URL="https://bugs.debian.org/"
```

En ce qui concerne les accessoires, j'ai simplement un clavier (pas besoin de souris - youpi !), une alimentation Raspberry Pi 4, et les deux sorties Micro HDMI connectées (**figure 2**). Actuellement, l'une est reliée à un petit moniteur portable et l'autre à un périphérique de capture sur mon PC, que, pour l'instant, j'utilise pour quelques captures d'écran. L'idée à terme est d'avoir un code qui s'exécute sur un écran et une sortie vidéo (par exemple d'un script de détection d'objet) visible sur l'autre.

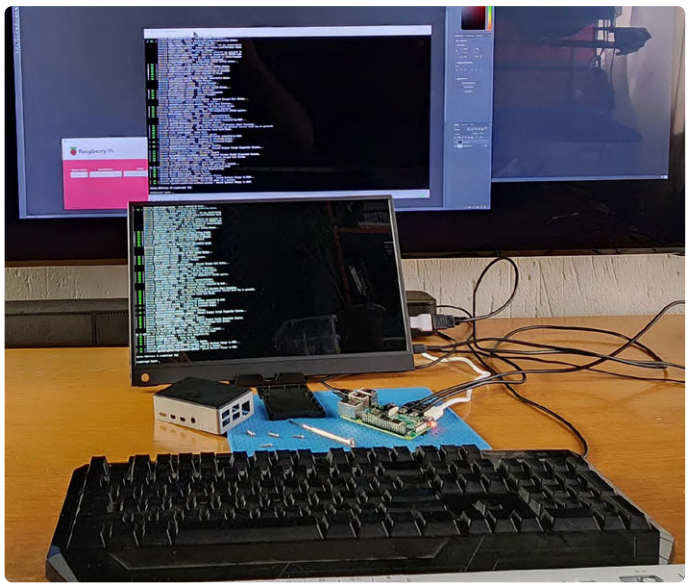


Figure 2. La configuration de base du Raspberry Pi que j'utilise pour ce projet.

Installation de TensorFlow Lite

Maintenant que nous avons tout ce qu'il faut pour commencer, exécutons TensorFlow Lite sur le Raspberry Pi.
Tout d'abord, quelques réglages. Lançons un cycle de mise à jour/mise à niveau sur le Raspberry Pi, juste pour nous assurer que nous avons toutes les dernières versions de tout ce qui est installé. Commençons par :

```
sudo apt-get update
```

Le temps d'attente dépend de votre retard côté mises à jour. Je n'ai eu que 12 éléments à mettre à jour. Ensuite :

```
sudo apt-get upgrade
```

Pour moi, cela a résulté en un écran entier rempli de mises à jour, ce qui me surprend toujours un peu puisque je viens juste d'installer la dernière version officielle du système d'exploitation. Soyez patient avec le processus - la durée peut varier.
En ce qui concerne l'installation de TensorFlow Lite, j'ai plongé dans divers résultats de recherche Google, et le guide le plus accessible que j'ai trouvé était celui d'EdjeElectronics sur GitHub [3]. Bien entendu, un certain temps s'est écoulé depuis la présentation initiale du processus par Edge et les nouvelles versions et mises à jour du système d'exploitation, et j'ai donc rencontré quelques difficultés mineures, que je vais documenter ici.
La première étape a été de cloner le dépôt GitHub en utilisant :

```
git clone https://github.com/EdjeElectronics/TensorFlow-Lite-Object-Detection-on-Android-and-Raspberry-Pi.git
```

La réponse encourageante de mon Raspberry Pi :

```
-bash: git: command not found
```

Ah oui, c'est une nouvelle installation. Oups. Installez *git* en utilisant :

```
sudo apt install git
```

Cela a fonctionné pour moi. Maintenant que git est installé, revenons à la commande `git clone...` mentionnée plus haut. Si vous réussissez, vous obtiendrez un sous-répertoire dans votre répertoire personnel appelé *TensorFlow-Lite-Object-Detection-on-Android-and-Raspberry-Pi*. C'est un peu compliqué, alors renommez-le en *tflite1*, avec la commande `mv` (move) :

```
mv TensorFlow-Lite-Object-Detection-on-Android-and-Raspberry-Pi tflite1
```

Ensuite, accédez à ce répertoire avec :

```
cd tflite1
```

EdjeElectronics suggère de créer un environnement virtuel, nous allons donc le suivre :

```
sudo pip3 install virtualenv
```

La réponse du Raspberry Pi, comme on pouvait s'y attendre :

```
-bash: pip3: command not found
```

L'installation de *pip3* s'est faite presque aussi facilement que celle de *git*. J'ai d'abord utilisé `sudo apt-get update`, où neuf mises à jour ont été effectuées, puis j'ai utilisé la commande :

```
sudo apt-get install python3-pip
```

De nombreuses installations ont eu lieu à ce stade, ce qui a abouti à un autre écran rempli de texte.
Ensuite, j'ai réessayé de lancer la commande `sudo pip3...`. Debian n'a pas aimé. Après quelques recherches, j'ai découvert que je devais créer mon propre environnement virtuel. En supposant que vous êtes toujours dans le répertoire *tflite1*, utilisez :

```
python3 -m venv tflite1-env
```

Activez ensuite cet environnement virtuel :

```
source tflite1-env/bin/activate
```

Il faut maintenant installer les dépendances de TensorFlow Lite et OpenCV. Comme nous l'avons vu précédemment, TensorFlow Lite a de nombreux domaines d'application, mais cette installation va l'adapter à un domaine très populaire : la vision artificielle et la reconnaissance d'objets. Ainsi, OpenCV, une bibliothèque open-source destinée aux tâches de vision artificielle et d'apprentissage automatique, sera bien pratique.
Heureusement, vous n'aurez pas besoin de `curl`, `wget`, ou `git clone` ici, parce que le créateur du dépôt a écrit un script shell utile de 40 lignes,

appelé *get_pi_requirements.sh*. Si vous êtes curieux de découvrir ce que le script est censé faire avant de l'exécuter, vous pouvez voir son contenu en exécutant un `cat get_pi_requirements.sh`, mais nous allons l'exécuter en entrant :

```
bash get_pi_requirements.sh
```

J'ai fait l'erreur d'essayer ceci après avoir redémarré le Raspberry Pi, ce qui a généré des messages d'erreur et des avertissements. Pourquoi ? La commande source que nous avons exécutée plus tôt dans le répertoire *tflite1* doit être exécutée chaque fois que vous redémarrez ou que vous démarrez une nouvelle session du Terminal. De plus, cela a pris quelques minutes pour s'exécuter, alors soyez patient, encore une fois ! Maintenant, nous avons la possibilité d'utiliser un modèle TensorFlow Lite de Google, ou d'utiliser un modèle que nous avons développé nous-mêmes. En tant que débutant, j'ai choisi d'utiliser celui de Google :

```
wget https://storage.googleapis.com/download.tensorflow.org/models/tflite/coco_ssd_mobilenet_v1_1.0_quant_2018_06_29.zip
```

Cette commande est longue et ne tient pas ici sans saut de ligne, mais notez que le seul espace que vous devez saisir est celui qui suit `wget` - il n'y en a pas à partir de `https`.
Ensuite, décompressez le fichier que vous venez de télécharger :

```
unzip coco_ssd_mobilenet_v1_1.0_quant_2018_06_29.zip -d Sample_TFLite_model
```

Et voilà, nous avons installé tout ce dont nous avons besoin pour commencer à nous amuser avec la détection d'objets et la classification d'images en utilisant TensorFlow Lite sur le Raspberry Pi.
Nous pouvons maintenant commencer à faire des essais.
Dans notre prochain épisode : en tant que collectionneur de données, en particulier d'images et de vidéos historiques qui ne sont souvent disponibles nulle part ailleurs, je souhaite partager ce que j'ai à ma disposition avec le monde entier. Mais je n'aurai jamais le temps de passer en revue chaque photo et chaque image vidéo pour ajouter des métadonnées et indiquer aux moteurs de recherche ce que j'ai. J'ai donc l'intention d'utiliser ces outils pour m'aider à analyser et à classer ma grande collection de fichiers multimédias. Je vous tiendrai au courant de mes succès et de mes échecs la prochaine fois !

230181-E-04

Questions ou commentaires ?

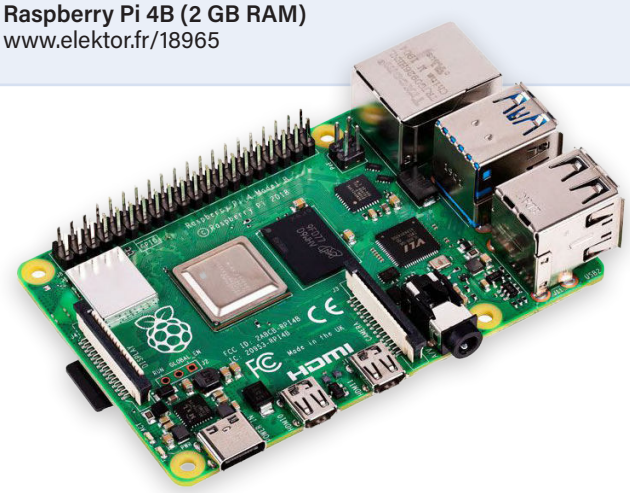
Envoyez un courriel à l'auteur (brian.williams@elektor.com).

À propos de l'auteur

Brian Tristam Williams est fasciné par les ordinateurs et l'électronique depuis qu'il a eu son premier « micro-ordinateur » à l'âge de 10 ans. Son aventure avec le magazine Elektor a commencé lorsqu'il a acheté son premier numéro à 16 ans, et depuis lors, il suit le monde de l'électronique et de l'informatique, ne cessant d'explorer et d'apprendre. Il a rejoint Elektor en 2010 et, aujourd'hui, il s'attache à suivre les dernières tendances technologiques, en se concentrant notamment sur l'intelligence artificielle et les ordinateurs monocartes tels que le Raspberry Pi.

Produits

- **Raspberry Pi 4B (2 GB RAM)**
  www.elektor.fr/18965



LIENS

[1] Site officiel de TensorFlow Lite : https://tensorflow.org/lite
[2] Téléchargement de Raspberry Pi Imager : https://raspberrypi.com/software
[3] TensorFlow Lite Object Detection on Android and Raspberry Pi [GitHub] : https://tinyurl.com/edjetflite

262 144 façons de jouer au Jeu de la Vie

un projet de lecteur en bref

Brian White (États-Unis)

Le Jeu de la Vie de Conway me fascine depuis longtemps. Suivez-moi dans mon parcours de conversion de ce casse-tête informatique des années 1970 en un spectacle visuel captivant, en utilisant une matrice de LED RVB associée à des méthodes de codage uniques qui nous permettent de simuler toutes les règles imaginables du jeu !

Ce projet est l'histoire de plusieurs pièces, dont certaines m'ont trotté dans la tête pendant 50 ans, qui ont fini par s'assembler pour aboutir à un résultat fort satisfaisant. Il commence avec le Jeu de la Vie de John Horton Conway, dont j'ai entendu parler pour la première fois à la fin des années 1970. À l'époque, je suivais mon premier (et unique) cours de programmation – en langage BASIC sur un terminal DECwriter connecté à un PDP-11 dans mon lycée – et l'écriture du code pour le Jeu de la Vie était l'un des exercices. C'est un exemple amusant de boucles imbriquées (`FOR...NEXT` en BASIC), un excellent exercice d'initiation à la programmation, qui produit des motifs changeants fascinants. Dans les années qui ont suivi, cet exercice a continué à me fasciner alors que l'informatique – et donc le jeu de Conway – passait des terminaux papier aux écrans vidéo, puis aux ordinateurs portables et aux iPads.

Le jeu de la vie

En bref, le Jeu de la Vie de Conway [1] (en anglais « The Game of Life », GoL) est une simulation simple d'automate cellulaire basée sur des règles. Il se joue sur une matrice rectangulaire de taille quelconque. Les pièces du jeu sont des cellules qui occupent chacune une case de la matrice. Dans le jeu, les cellules peuvent survivre, mourir ou naître d'une génération à l'autre. L'état de vie d'une cellule occupant une case donnée dans la génération suivante

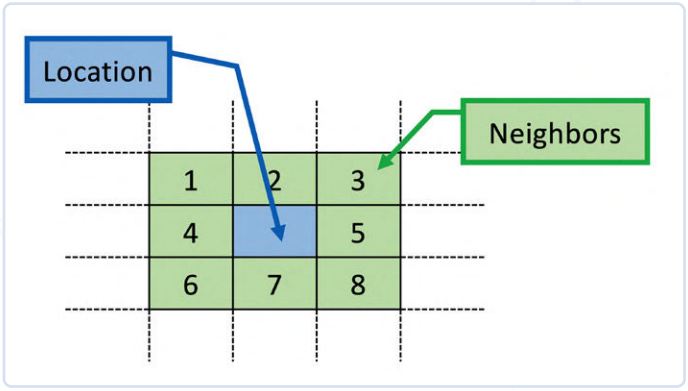


Figure 1. Une case dans le « monde » et ses huit cases voisines.

dépend uniquement du nombre de voisines vivantes qu'elle a dans la génération actuelle. Dans une matrice rectangulaire, chaque cellule a entre zéro et huit voisines (**figure 1**).
Les règles de Conway sont les suivantes :

- Une cellule survit au changement de génération si elle a deux ou trois voisines vivantes. Les cellules qui en ont moins « meurent de solitude » ; celles qui en ont plus « meurent de surpopulation ».
- Une cellule naît dans une case vide à la génération suivante si elle a exactement trois voisines vivantes.

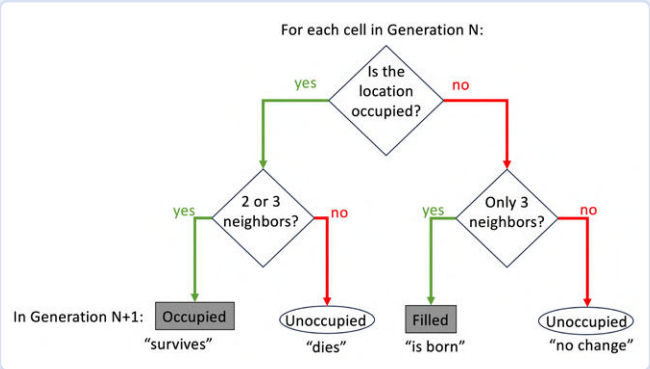
Conway a retenu ces règles après une période d'expérimentation afin de répondre aux critères suivants (tels que décrits dans [2]) :

- Il ne devrait pas y avoir de motif initial pour lequel il existe une preuve simple que la population peut croître sans limite.
- Il devrait y avoir de motifs initiaux qui *semblent* croître sans limite.
- Il devrait y avoir des motifs initiaux simples qui se développent et changent pendant une durée considérable avant de prendre fin de trois manières possibles : en s'éteignant complètement (en raison d'une surpopulation ou d'une trop faible densité), en s'installant dans une configuration stable qui reste inchangée par la suite, ou en entrant dans une phase d'oscillation dans laquelle ils répètent un cycle sans fin de deux périodes ou plus.

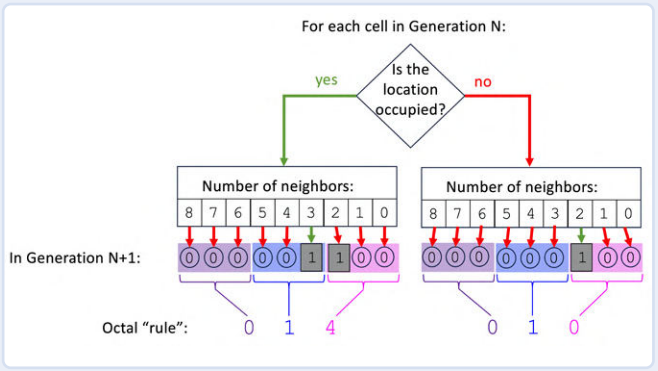
Codage des règles en six chiffres octaux

Les règles originelles du jeu de la vie peuvent être décrites brièvement :

- Une cellule survit si elle a 2 ou 3 voisines, sinon elle meurt.
- Une cellule naîtra dans une case vide avec exactement trois voisines.



Ces mêmes règles peuvent également être représentées de la manière suivante : un « 1 » dans la règle indique une case occupée à la génération suivante (survie ou naissance) pour ce nombre de voisins et un « 0 » dans la règle indique une case vide à la génération suivante (mort ou déjà vide).



Le motif binaire spécifie l'état de la cellule dans la génération suivante ; l'index dans le motif binaire est le nombre de voisins combiné avec le fait que la cellule est actuellement occupée ou non. Le décodage de la règle 256020 est présenté ci-dessous à titre d'exemple :

Tableau 1. Représentation octale du jeu de règles « 256020 ».

Case d'origine	**Occupée**									**Vide**								
Voisines	8	7	6	5	4	3	2	1	0	8	7	6	5	4	3	2	1	0
Génér. suiv.	0	1	0	1	0	1	1	1	0	0	0	0	0	1	0	0	0	0
Octal	2			5			6			0			2			0		

Traduit en texte, cela donnerait

- Actuellement, les cellules vivantes survivent si elles ont 1, 2, 3, 5 ou 7 voisines ; sinon, elles meurent.
- Une cellule naîtra dans chaque case actuellement vide ayant exactement 4 voisines.

Ce format d'encodage permet de spécifier n'importe lequel des 262 244 jeux de règles possibles pour le Jeu de la Vie.

Dans la version originale de 1970, le jeu se jouait avec des pions physiques sur un damier. Peu après, le processus a été informatisé et les innovations se poursuivent encore aujourd'hui [3].

Algorithmes intéressants

La pièce suivante du puzzle a été le livre « A New Kind of Science » de Stephen Wolfram, dans lequel il présente divers algorithmes d'automates cellulaires qui produisent des motifs visuels très intéressants. L'un des éléments clés de l'analyse de Wolfram était le codage des règles de transition d'une génération à l'autre sous forme de nombres binaires. En codant les règles de cette manière, il est possible d'explorer les conséquences de tous les ensembles de règles possibles d'une manière bien définie et reproductible [4]. Cela m'a amené à réfléchir à un moyen d'encoder les règles du GoL afin qu'il soit possible d'explorer tous les ensembles de règles possibles. J'ai compris qu'il était possible d'encoder les règles dans un nombre binaire de 18 bits (voir l'encadré **Encoder les règles en six chiffres octaux** pour plus de détails). Chacun des neuf bits de poids faible (0 à 8) indique l'état de la case dans la génération suivante (1 = occupée ; 0 = vide) pour une case vide avec zéro à huit voisines – le bit 0 indique l'état pour la génération suivante si la case a actuellement zéro voisine ; le bit 1 pour une voisine, etc. De même, les neuf bits de poids fort (9 à 17) donnent l'état de la génération suivante pour une case occupée avec zéro à huit voisines - bit 9 pour zéro voisine ; bit 10 pour une voisine, etc. En utilisant ce codage, il est possible de spécifier n'importe quel ensemble de règles basées sur le nombre de voisines d'une case.

Il est intéressant de noter qu'il n'est pas nécessaire que ces règles aient un « sens biologique ». Par exemple, la règle 256020, selon laquelle les cellules survivent si elles ont 1, 2, 3, 5 ou 7 voisines. Ce jeu de règles donne un comportement intéressant même s'il n'y a aucune justification biologique à ce que 4, 6 et 8 voisines soit un surpeuplement alors que 3, 5 et 7 ne le serait pas.

Étant donné que tous les jeux de règles possibles peuvent être encodés en 18 bits binaires et 2^{18} = 262 144, cela signifie qu'il y a 262 144 façons de jouer au GoL. Il est donc possible de construire un appareil qui permettrait à un utilisateur d'explorer toutes ces possibilités et d'observer comment une population de chaque « espèce » différente se développe au fil du temps.

Le dernier élément de la conception du logiciel est le fruit de

nombreuses conversations avec l'artiste électronique Kelly Heaton [5], que j'ai rencontrée grâce à un article paru dans le magazine *Elektor* [6]. Elle m'a encouragé à sortir des sentiers battus et à réaliser des œuvres qui soient à la fois fidèles à l'algorithme et visuellement attrayantes. Cela m'a amené à modifier le schéma de coloration standard des simulations du GoL, où les cases occupées sont colorées et les cases vides sont noires. Je voulais donner aux motifs une plus forte impression de changement, j'ai donc colorié les cellules nouveau-nées en bleu ; les cellules qui survivent pendant plus d'une génération sont vertes ; et les cellules qui meurent laissent un « fantôme » violet foncé pendant une génération (les « fantômes » ne sont pas comptés comme des voisins). Ce schéma est détaillé dans la **figure 2**.

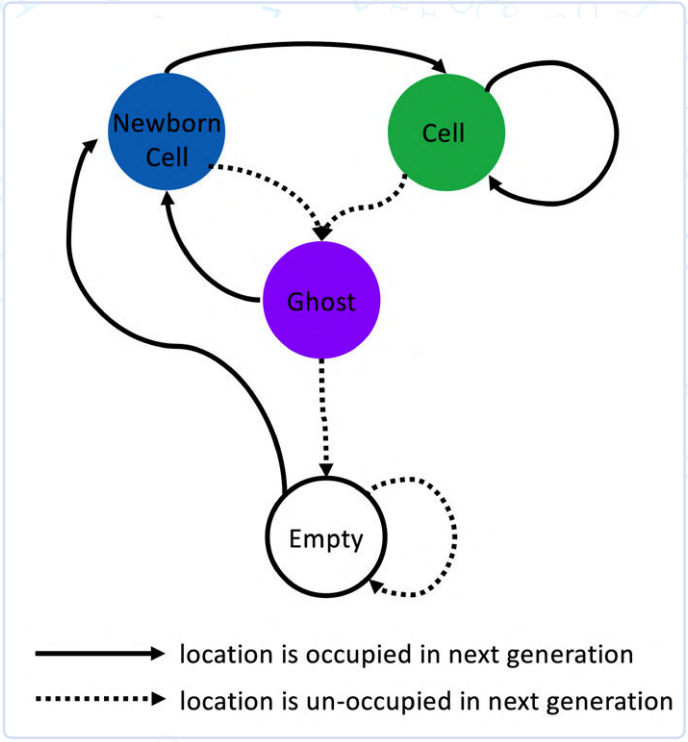


Figure 2. Diagramme d'état pour un motif de cellule multicolore.

Assemblage du matériel

La dernière pièce, le matériel, est un cadeau de Noël de mes fils : un Matrix Portal d'Adafruit [7] et une matrice de 32×64 LED RGB [8]. J'ai ensuite assemblé toutes les pièces dans un boîtier en acrylique pour rendre toutes les parties internes visibles. Le produit final est illustré à la **figure 3**. Sur la gauche se trouvent trois séries de trois roues codeuses de sélection des règles et des paramètres pour la prochaine exécution du programme. Celles du haut et du milieu définissent les règles sous la forme de six chiffres octaux – chaque chiffre représente 3 bits ; multipliés par 6 chiffres, on obtient les 18 bits requis. Les trois chiffres du haut spécifient les règles pour les cases occupées, c'est-à-dire les règles qui déterminent quelles cellules survivent à la génération suivante. Les trois chiffres du milieu spécifient les règles pour les cases vides, c'est-à-dire les règles pour la naissance de nouvelles cellules. À titre d'exemple, les règles canoniques de Conway seraient exprimées sous la forme 014010. Pour plus de détails, lire l'encadré **Codage des règles en six chiffres octaux**. Sous l'écran, de gauche à droite, se trouvent la carte d'extension de port MCP23017, la carte MCU Matrix Portal, le bouton-poussoir de réinitialisation et l'interrupteur à bascule du mode d'affichage.

Chaque exécution commence par une distribution aléatoire de cases occupées et vides, puis se poursuit pendant un nombre fixe de générations avant de redémarrer avec une nouvelle distribution aléatoire.

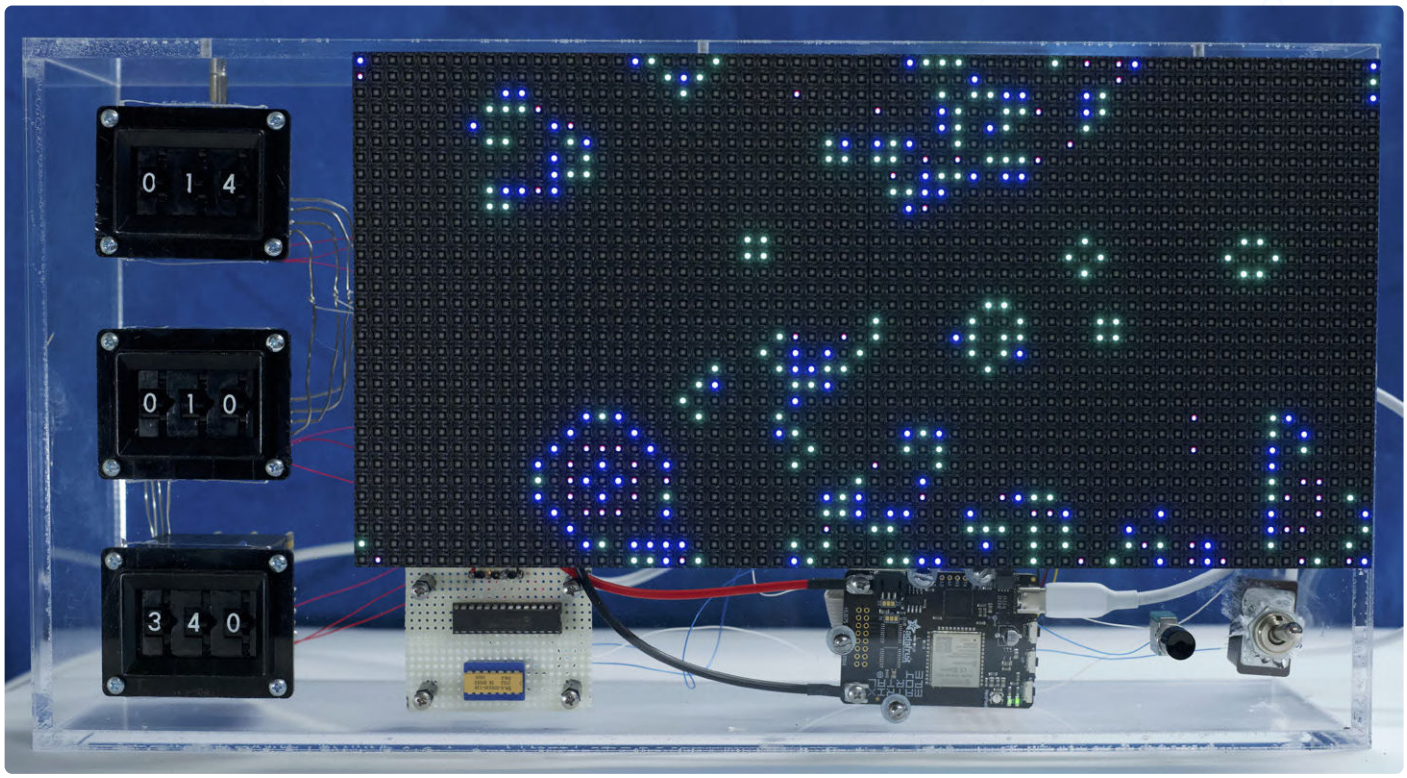


Figure 3. Démonstration du projet assemblé.

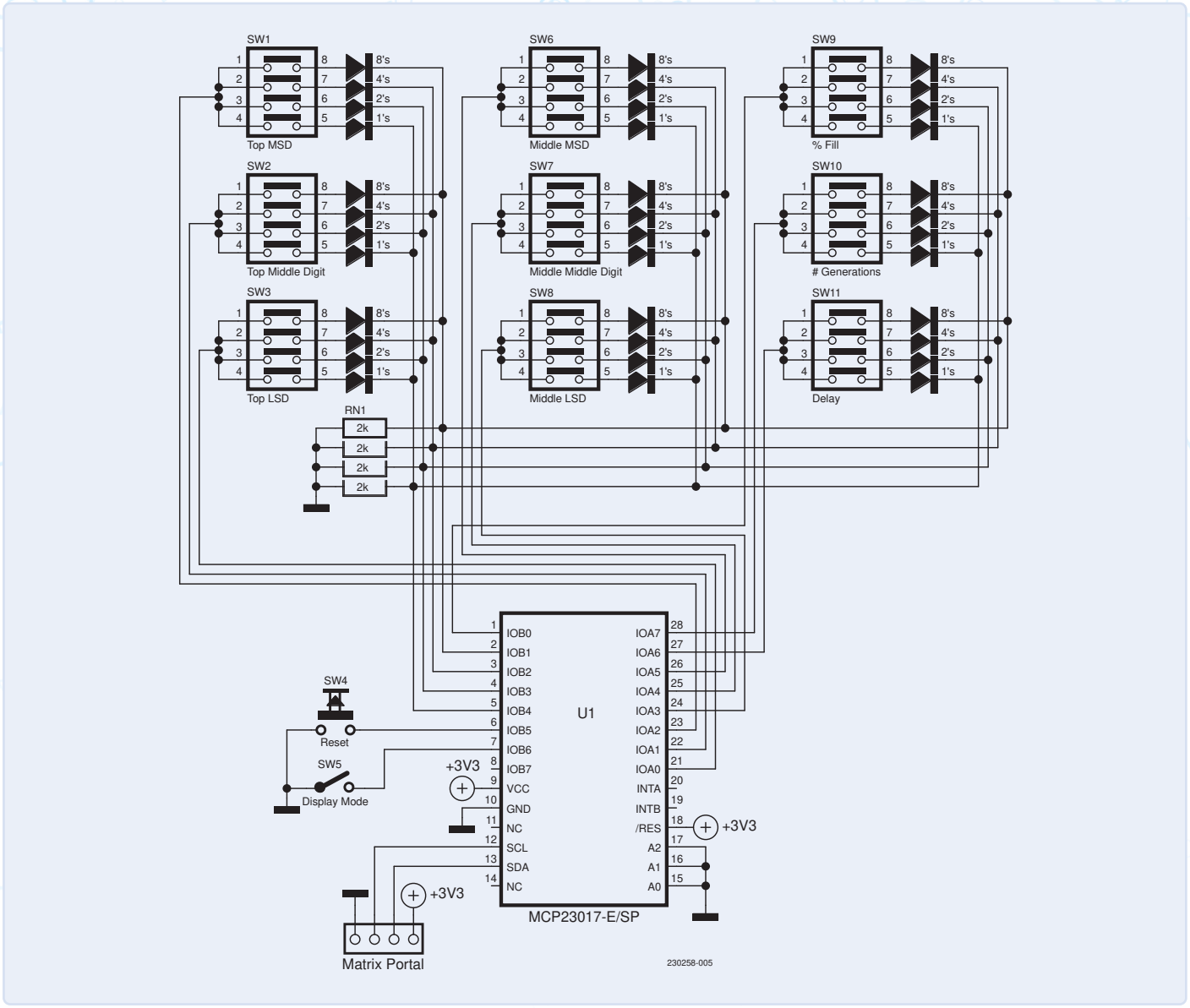


Figure 4. Schéma sans le Portal Matrix, l'afficheur 64×32 et l'alimentation USB.

Les trois chiffres inférieurs précisent les détails de ce processus. Le chiffre de gauche spécifie la densité de cellules vivantes dans la population initiale ; plus précisément, la probabilité qu'une case soit occupée au départ est égale à 10 % de la valeur de ce chiffre. Ainsi, un réglage de «3» signifie qu'au commencement, environ 30 % des cellules sont occupées. Étant donné que certains ensembles de règles donnent des résultats plus intéressants avec plus ou moins de cases occupées, cela permet d'explorer les effets de la variation de la densité cellulaire initiale. Le chiffre central indique le nombre N de générations à exécuter avant de redémarrer. J'ai choisi une échelle exponentielle pour permettre une large gamme de temps d'exécution. Plus précisément, il s'agit de $2\times10^{(N/2)}$ générations ; ainsi, un réglage de 4 correspond à 200 générations. Certains motifs se terminent rapidement, tandis que d'autres ne se terminent jamais ; ce réglage permet de maintenir l'affichage dans un état intéressant. Enfin, le chiffre le plus à droite indique le délai en secondes entre les générations ; ainsi, un réglage de 0 permet d'obtenir un défilement maximal. En ralentissant ce défilement, on peut suivre l'effet des règles en détail. Les deux dernières commandes sont un bouton-poussoir qui redémarre immédiatement l'exécution et randomise à nouveau la population et un interrupteur à bascule qui permet de sélectionner la coloration standard (bleu = occupé ; éteint = vide) ou mon schéma de coloration multicolore des cases dans la matrice.

Le réaliser soi-même

Le schéma de la **figure 4** est très simple. Le Matrix Portal et l'écran sont connectés comme décrit dans les instructions d'Adafruit pour le microcontrôleur et la matrice LED. Les extensions représentées reçoivent leur alimentation +3.3 V et la communication I²C du Portal Matrix via son connecteur Stemma QT. Le gros du travail est effectué par un port d'expansion MCP23017 I²C. Les roues codeuses DCB (SW1 à SW6, SW9 à SW11) sont activées individuellement par les sorties du MCP23017, et les sorties DCB activées de chaque roue sont ensuite lues. Comme le MCP23017 n'a pas d'option de rappel au niveau bas, les lignes DCB sont ramenées à la masse par un réseau de résistances. Les deux dernières entrées du MCP23017 sont utilisées avec les résistances de rappel internes au niveau haut pour lire les commandes de mode de réinitialisation et d'affichage. L'alimentation est fournie par un bloc mural de 5 V, 4 A qui alimente le Matrix Portal via un câble USB-C.

Le code est basé sur le code GoL fourni par Adafruit pour le Portal et l'écran Matrix [9] qui utilise de remarquables astuces de programmation pour exécuter très rapidement le code des boucles

Liste des composants

Adafruit Matrix Portal – Afficheur Internet alimenté par CircuitPython (Adafruit 4745)
64x32 RVB LED Matrix – au pas de 5 mm (Adafruit 2277)
Roue codeuse DCB à 3 chiffres (exemple : Littlefuse 3P-3-23-3-1-0-2)
MCP23017 port d'expansion
4 x 2K résistances de rappel
Bouton-poussoir
Interrupteur à bascule SPST
Bloc d'alimentation USB
Boîtier acrylique

emboîtées typiques du GoL. J'ai modifié le code pour remplacer les règles « standard » par celles basées sur les nombres saisis sur les roues codeuses. En bref, je lis les valeurs binaires des roues et les utilise pour créer une liste de 18 bits à 1 ou à 0 représentant l'état de la case dans la génération suivante, en fonction de l'état présent de la case et du nombre de voisines non vides et non fantômes. Je compte ensuite les voisines et j'utilise ce nombre, auquel j'ajoute 9 si la case est actuellement occupée, comme index dans la liste de règles. J'ai également réalisé le motif multicolore décrit plus haut. Le code est disponible sur la page du projet Elektor Labs [10].

Pour aller plus loin

Les résultats ont été fascinants. J'ai créé une YouTube playlist [11] avec une collection de courtes vidéos montrant les motifs qui émergent avec différents réglages des roues codeuses. Il est remarquable de constater que certaines modifications des règles ont un effet important alors que d'autres n'apportent que des changements subtils dans la manière dont les motifs se développent, changent, se déplacent et s'éteignent. Je n'ai fait que commencer à étudier les possibilités (262 144 est un nombre assez important !), mais voici ce que j'ai remarqué :

- Le réglage des bits de poids faible (bits 0 et 9), qui permet les naissances et/ou les survies avec zéro voisine, entraîne des inversions positives-négatives spectaculaires à chaque génération.
- Le réglage de bits d'ordre supérieur - contrôler les naissances et les survies avec un plus grand nombre de voisines - tend à avoir moins d'effet, peut-être parce que les cases ayant beaucoup de voisines sont plus rares.
- La règle originale, 014010, évolue généralement vers des structures relativement stables ou des figures oscillantes. D'autres règles, comme la 114110, sont les préférées de ma femme car elles semblent ne jamais s'arrêter.
- Vous pouvez créer des règles dans lesquelles les structures sont créées et se développent progressivement de manière intéressante si vous autorisez la survie pour de nombreuses voisines et les naissances pour quelques-unes seulement ; par exemple, 256020.

En ce moment, l'appareil repose sur notre table de cuisine et je constate souvent que quelqu'un de la maisonnée a trouvé un nouvel ensemble de règles qui donne lieu à un nouveau motif fascinant. Depuis que nous vivons avec cet appareil et que nous le partageons avec d'autres, nous avons eu plusieurs idées pour ceux qui voudraient en construire un à leur tour. Tout d'abord, on pourrait ajouter d'autres couleurs au schéma multicolore. Par exemple, les couleurs pourraient passer progressivement à d'autres teintes au fur et à mesure que les cellules « mûrissent ». Kelly Heaton a suggéré que les couleurs se fondent l'une dans l'autre plutôt que de changer brusquement, afin d'obtenir un effet plus fluide. Enfin, les membres de mon foyer qui ignorent l'octal suggèrent de trouver un moyen de montrer plus explicitement la correspondance entre les bits des règles et le nombre de voisines. On pourrait ajouter une rangée de 18 LED pour afficher l'équivalent binaire des paramètres octaux ou, plus simplement encore, utiliser 18 interrupteurs individuels, un pour chaque bit de la règle.
J'espère que des lecteurs de cet article se lanceront dans la réalisation de leur version de ce projet, soit avec un écran et un MCU dédiés, soit entièrement sur ordinateur, et qu'ils partageront les ensembles de règles qu'ils trouvent intéressants. Qui aurait cru que partager des nombres octaux à 6 chiffres pouvait être aussi intéressant ?

VF : Helmut Müller — 230258-04

À propos de l'auteur

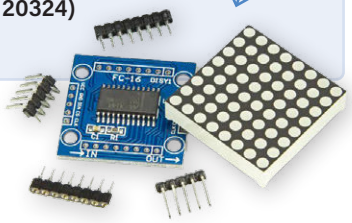
Brian White est professeur au département de biologie de l'université du Massachusetts, à Boston. Il a étudié la biologie au MIT et à Stanford. En tant qu'enseignant, il mène également des recherches sur la biologie et l'enseignement de la biologie et développe des logiciels connexes. Brian est également un passionné d'électronique, un musicien et un radioamateur (KA1TBQ). De plus amples informations sur ses projets électroniques sont disponibles sur son site [12].

Questions ou commentaires ?

Envoyez un courriel à l'auteur (brian.white@umb.edu) ou contactez Elektor (redaction@elektor.fr).

Produits

- **MAX7219 Dot Matrix Module (Set of 8)MAX7219 Module matriciel à points (lot de 8)**
  www.elektor.fr/18422
- **Adafruit Feather RP2040 (SKU 19689)**
  www.elektor.fr/19689
- **ESP32-C3-DevKitM-1 (SKU 20324)**
  www.elektor.fr/20324



LIENS

[1] Wikipédia : Le jeu de la vie de Conway : https://en.wikipedia.org/wiki/Conway%27s_Game_of_Life
[2] Martin Gardner, «MATHEMATICAL GAMES : The fantastic combinations of John Conway's new solitaire game 'life'», Scientific American 223 (octobre 1970) : 120-123 : https://web.stanford.edu/class/sts145/Library/life.pdf
[3] LifeWiki : https://conwaylife.com/wiki
[4] Stephen Wolfram, Un nouveau genre de science, voir en particulier cette section : https://wolframscience.com/nks/p53--more-cellular-automata
[5] Kelly Heaton Studio: https://kellyheatonstudio.com
[6] C.J. Abate, « Faire de l'art avec l'électricité : Q/A avec Kelly Heaton », Elektor Circuits de Vacances 2022 : https://www.elektormagazine.fr/magazine/elektor-264/60889
[7] Matrix Portal - Affichage Internet alimenté par CircuitPython : https://adafruit.com/product/4745
[8] Matrice de 64×32 LED RVB – Au pas de 5 mm : https://adafruit.com/product/2277
[9] Exemple Adafruit : Le « Jeu de la Vie » de Conway : https://learn.adafruit.com/rgb-led-matrices-matrix-panels-with-circuitpython/example-conways-game-of-life
[10] Page du projet sur Elektor Labs : https://elektormagazine.fr/labs/262144-ways-to-play-the-game-of-life
[11] Vidéos du Jeu de la Vie : https://youtube.com/playlist?list=PL2wCLIpn1MkRpBHeZ2svMs-CdLvf2B_9N
[12] Site web de l'auteur : https://brianwhite94.wixsite.com/electronics

sur le vif

le souffle du dragon sur mon cou

Ilse Joostens (Belgique)

Importée en Europe au début du XVIIe siècle par la Compagnie néerlandaise des Indes orientales, la porcelaine chinoise dite kraak était un objet de décoration très prisé des foyers aisés, notamment aux Pays-Bas. Après la chute de la dynastie Ming en 1644 et l'interruption du commerce avec la Chine, les manufactures de Delft se lancèrent dans la production de copies reprenant l'aspect délicat de cette porcelaine, mais en usant d'une glaçure blanche à l'étain moins coûteuse à produire. Près de quatre siècles plus tard, les rôles semblent inversés puisque ce sont désormais les Chinois qui copient les produits occidentaux.

Au matin du 12 octobre 1654, Cornelis Soetens détruisit par mégarde une grande partie de la ville de Delft en allant chercher un échantillon d'explosif à la poudrière appelée « Secret de la Hollande ». L'industrie de la faïence était déjà florissante à l'époque, mais la reconstruction de la ville lui donna un nouvel élan. La faïence de Delft est aujourd'hui encore reconnue dans le monde entier. Quant à la poudrière, elle aussi fut reconstruite, mais à distance plus prudente de la ville.

Mettez m'en 5 pour le prix de 2

À vrai dire la faïence de Delft [1] était bien plus qu'une simple copie de la porcelaine chinoise. Les peintres céramistes des manufactures de Delft étaient notamment réputés pour la finesse de leur décor et leur constant recours à des techniques nouvelles. On ne saurait en dire autant de la Chine et de sa propension à copier l'Occident : inauguré à Dongguan en 2019, le campus Ox Horn du fabricant Huawei est une photocopie de l'Europe [2]. Le train qui y conduit ressemble à s'y méprendre à celui du Chemin de fer de la Jungfrau, et les zones du campus qu'il déverse, comme Heidelberg, Bologne et Grenade, comprennent des répliques de monuments célèbres de ces villes. Ailleurs en Chine, les Champs-Élysées et le pittoresque village autrichien de Hallstatt [3] ont été recréés pour servir d'attractions touristiques. Bien d'autres produits font l'objet de copies, y compris des œuvres d'art. Les magasins chinois n'hésitent pas non plus à reprendre l'aspect et le sens du service de magasins occidentaux à succès, comme ceux d'Ikea ou d'Apple pour ne pas les nommer.

Des batailles juridiques ont certes conduit à la fermeture de ces magasins, mais la copie de produits et d'idées se poursuit. Alors que je naviguais sur le site d'oncle Ali, j'ai failli perdre ma mâchoire en tombant sur une réplique de deux mètres de haut de la sculpture *Kindred Spirits* d'Alex Pentek [4]. L'électronique n'est pas épargnée. Contrefaçons de CI, composants recyclés vendus comme neufs, cartes, gadgets et autres machines vendues à prix très bas, le dossier est épais. Même les produits de niche sont visés : Dalibor Farný, le fabricant tchèque des tubes Nixie R|Z568M, a de quoi craindre cette concurrence redoutable. En 2016, j'ai écrit pour Elektor un article sur les afficheurs à 7 segments à filaments de LED (alors relativement nouveaux) [5]. Il semble désormais possible de commander auprès d'oncle Ali des kits pour horloges à filaments de LED à des prix inférieurs à ce que me coûterait l'achat de leurs composants. Les frais d'expédition sont eux aussi très bas, car la Chine bénéficie du statut de pays en développement auprès de l'Union Postale Universelle [6]. Il semble que cette agence de l'ONU ait manqué l'épisode où la Chine a réussi à poser une sonde spatiale sur la face cachée de la lune. Qu'une idée soit réutilisée ne me gêne pas, mais il est fâcheux de se voir évincer du marché par sa propre idée.

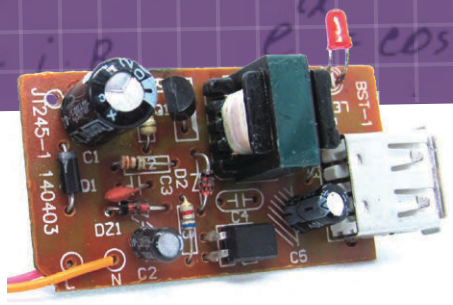


Figure 1. Adaptateur bon marché, face composants...

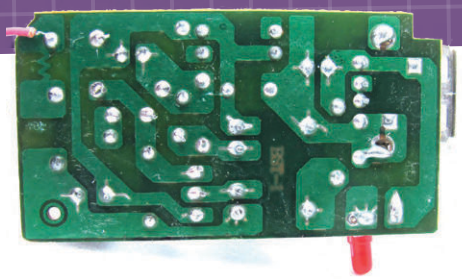


Figure 2. ...et face cuivre. (Figures 1 et 2 : photos de Leo Potjewijd).

Figure 3. L'éclairage de ma lampe en cristal de sel fabriquée en Chine.

Danger : restes de tofu

Comme dans de nombreuses sociétés, corruption, mensonge et tricherie existent à tous les niveaux en Chine, même si la majorité des citoyens sont aimables et honnêtes. Ces machinations sournoises ont des conséquences graves, comme l'effondrement de bâtiments construits avec des matériaux de mauvaise qualité. Par métaphore, les Chinois appellent « restes de tofu » les projets résultant d'un travail bâclé [7]. De nombreux biens de consommation chinois importés et vendus chez nous peuvent eux aussi être de mauvaise qualité, et même mettre des vies en danger. Les adaptateurs et chargeurs USB bon marché construits avec un nombre minimal de composants sont légion. La distance d'isolation entre les sections primaire et secondaire est pratiquement inexistante, et seule une mince trace de circuit imprimé sert de fusible. Rien d'étonnant à ce que des personnes soient parfois électrocutées dans leur bain par leur smartphone. Les médias n'en font qu'un fait divers, et à l'exception d'une mise en garde sur les dangers à utiliser un smartphone relié à son chargeur, nul ne se soucie des raisons. J'ai trouvé les photos des **figures 1** et **2** sur un forum. Outre les défauts précédents, l'optocoupleur de cet adaptateur n'a visiblement aucune fonction...

Pas mieux pour cette lampe en cristal de sel trouvée l'an dernier sous mon sapin de Noël. Elle était réglable en intensité, mais clignotait de manière gênante lorsque le variateur n'était pas au maximum. J'ai mis le nez dedans : le circuit de l'éclairage (**fig. 3**) comprenait juste un pont redresseur, un CI toujours en surchauffe et une matrice de LED – pas même un condensateur de lissage là-dedans. J'ai pu ouvrir le boîtier du variateur (**fig. 4**) sans outil, et le cordon d'alimentation s'est révélé être de l'aluminium cuivré avec des brins de moins d'un dixième de mm² (**fig. 5**). Ajoutez à cela un faux disjoncteur *made in China* [8], et les effets pyrotechniques sont presque garantis (**fig. 6**). Même si ce n'est pas toujours facile, acheter davantage de produits locaux et donner une chance à nos concitoyens ne semble donc pas une mauvaise idée.

VF : Hervé Moreau — 230609-04

Figure 4. Le variateur de ma lampe en cristal de sel.

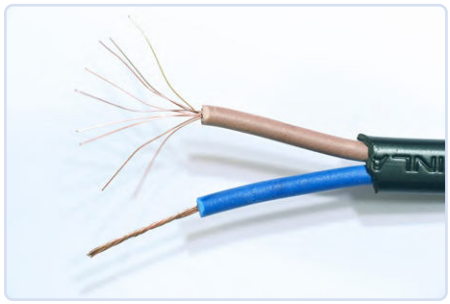


Figure 5. Le cordon d'alimentation de la lampe. Effroyablement fin...

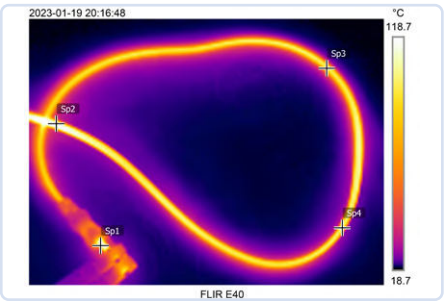


Figure 6. ...et terriblement chaud sous 6 A (118 °C d'après la caméra thermique).

LIENS

[1] Wikipédia : Faïence de Delft : https://fr.wikipedia.org/wiki/Fa%C3%AFence_de_Delft
[2] Photos of Huawei's European-Themed Campus in China, The Atlantic : https://tinyurl.com/atlanticoxhorn
[3] Wikipedia : Hallstatt (China) : https://en.wikipedia.org/wiki/Hallstatt_(China)
[4] Wikipédia : Sculpture Kindred Spirits : https://fr.wikipedia.org/wiki/Kindred_Spirits
[5] LEDitron, Peter S'heeren, Elektor 4/2016 : https://www.elektormagazine.fr/magazine/elektor-201604/28899
[6] How AliExpress Can Ship Packages So Cheaply, RTL News [hollandais] : https://tinyurl.com/rtlaliexpress
[7] Fragile steel bars/Tofu-dreg project in China/Shaky building/Collapsing buildings/Poor quality, China Insights [YouTube] : https://youtu.be/s-2DtL-Wjkc
[8] Inside a fake un-trippable circuit breaker, bigclivedotcom [YouTube] : https://youtu.be/2TJEzdqtXIQ

faites tourner votre moteur CC

exemples de projets du livre Motor Control Development (offre groupée)

Dogan Ibrahim (Royaume-Uni)

La meilleure façon de s'initier à la programmation et à l'électronique pratique est d'étudier comment on peut faire tourner des composants, les faire vibrer ou clignoter. Le mouvement implique souvent l'utilisation d'un moteur, et un petit moteur à courant continu à balais est idéal pour démarrer avant de se lancer dans la robotique et la mécatronique. Ici, nous proposons une introduction à la commande intelligente d'un moteur à courant continu, en nous appuyant sur un kit de développement exceptionnel d'Elektor qui combine le matériel, le logiciel et un guide pratique.

Note de l'éditeur : *cet article est un extrait du livre de 192 pages Motor Control Development Kit (Elektor, offre groupée) formaté et légèrement modifié pour correspondre aux normes éditoriales et à la mise en page du magazine Elektor. L'auteur et l'éditeur ont fait de leur mieux pour l'éviter et seront heureux de répondre aux questions – pour les contacter, voir l'encadré* **« Questions ou commentaires ? »**.

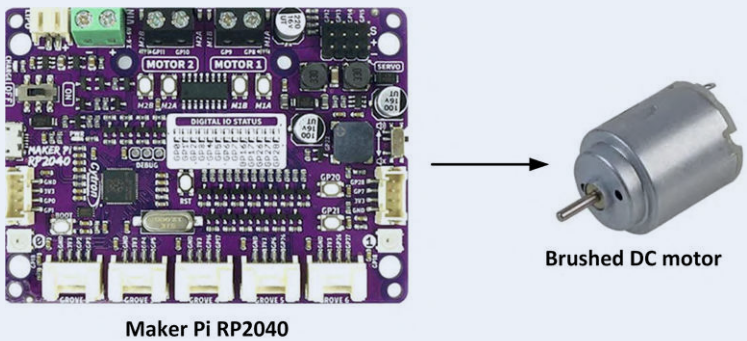


Figure 1. Les deux principaux composants du projet.

Dans cet article, trois projets simples basés sur un petit moteur à courant continu, à balais et à aimant permanent, commandés par la carte de développement Cytron/Maker Pi RP2040, sont décrits. Les deux composants sont inclus dans l'offre groupée disponible sur l'e-choppe Elektor [1]. Les projets constituent des exemples exploratoires et éducatifs plutôt que des projets complets.

Commande marche/arrêt d'un moteur CC

Ce projet de base montre comment connecter un petit moteur CC à balais fonctionnant de 1...6 V CC à l'un des borniers DC MOTOR de la carte Maker Pi RP2040. Le moteur est commandé en marche avant pendant 5 secondes, puis arrêté pendant 5 secondes. Ensuite, il tourne en sens inverse pendant 5 secondes et s'arrête à nouveau pendant 5 secondes. Ce processus est répété indéfiniment jusqu'à ce qu'il soit arrêté manuellement.
Le but de ce projet est de se concentrer sur la connexion, le fonctionnement et la commande d'un moteur CC avec la carte Maker Pi RP2040. La **figure 1** montre le schéma fonctionnel du projet, et la **figure 2** présente le moteur utilisé. Il s'agit d'un petit moteur CC, à balais, conçu pour fonctionner de +1 V à +6 V avec une tension de fonctionnement recommandée de +3 V. Voyons comment le faire tourner.
Un double pilote de moteur à pont en H, commandant jusqu'à deux moteurs CC à balais, M1 et M2, ou un moteur pas à pas, est intégré sur la carte de développement. La disposition des broches d'E/S est la suivante

- M1A : GP8
- M1B : GP9
- M2A : GP10
- M2B : GP11

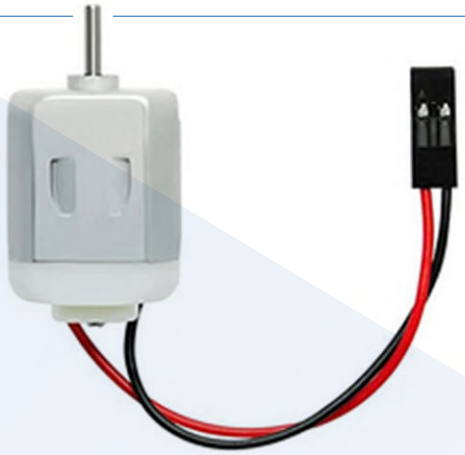


Figure 2. Le petit moteur à courant continu utilisé.

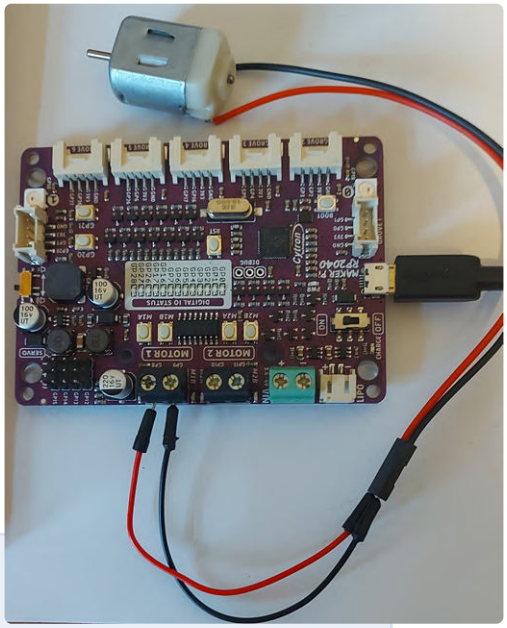


Figure 3. Connexion du moteur à la carte RP2040 Maker Pi.

La table de vérité du pilote de moteur est présentée dans le **tableau 1**. Dans ce projet, les entrées PWM1A (port GP8) et PWM1B (port GP9) du moteur sont utilisées avec les sorties du moteur sur M1A et M1B. Ces sorties (*Output A*) sont disponibles sur les bornes à vis MOTOR 1 situées au milieu en haut de la carte de développement. Comme le montre le tableau 1, le moteur est commandé par les ports GP8 et GP9. Par exemple, le fait de mettre GP8 au niveau haut (HIGH) et GP9 au niveau bas fait (LOW) tourner le moteur en avant. De même, le réglage de GP8 au niveau bas et GP9 au niveau haut fait tourner le moteur en arrière.

La **figure 3** montre le projet avec le moteur connecté aux sorties MOTOR 1.

Le programme de la démo de démarrage est donné dans le **listage 1**. Le programme appelé *DCMotor1.py* est contenu dans un fichier d'archive logiciel volumineux disponible gratuitement sur le site web d'Elektor [1]. Sur cette page, défilez vers *Téléchargements Software_Motor Control Development Kit.*

Le moteur est commandé par des formes d'ondes PWM comme décrit ailleurs dans le guide cité en référence. La bibliothèque intégrée de moteurs CC appelée *adafruit_motor* est utilisée dans ce projet. Au début du programme, les modules de bibliothèque *board*, *time*, *pwmio* et *motor* sont importés dans le programme. Les ports GP8 et GP9 de Motor 1 sont alors configurés pour fonctionner avec une forme d'onde PWM à une fréquence de 50 Hz. Le reste du programme s'exécute dans une boucle infinie.

La propriété de la classe `throttle` peut prendre une valeur comprise entre -1 et +1 et contrôle le moteur comme suit :

`throttle = 0` idle
`throttle = 1` le moteur tourne en avant à pleine vitesse
`throttle = 0.5` le moteur tourne en avant à 50 % de sa vitesse maximale
`throttle = -1` le moteur tourne en arrière à pleine vitesse
`throttle = -0.5m` le moteur tourne en arrière à 50 % de sa vitesse maximale

Dans ce programme, le moteur tourne à 25 % de sa vitesse maximale en réglant l'accélérateur à +0,25 dans un sens et à -0,25 dans l'autre.

Listage 1. DCMotor1.py

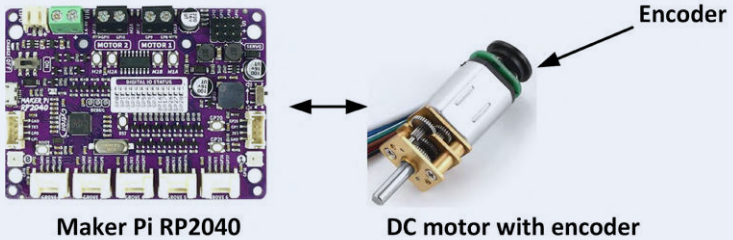
```
#-------------------------------------------------------------
#                BRUSHED DC MOTOR CONTROL
#
# In this program a brushed DC moor is controlled as follows:
# The motor is turned ON in forward direction for 5 seconds
# and then stopped for 5 seconds, then in reverse direction
# for 5 seconds and then stopped again for 5 seconds. This
# process is repeated forever
#
# Author: Dogan Ibrahim
# File  : DCMotor1.py
# Date  : February, 2023
#-------------------------------------------------------------
import board
import time
import pwmio
from adafruit_motor import motor
#
# Initialize DC Motor 1
#
m1a = pwmio.PWMOut(board.GP8, frequency=50)
m1b = pwmio.PWMOut(board.GP9, frequency=50)
motor1 = motor.DCMotor(m1a, m1b)
while True:
    motor1.throttle = 0.25              # 25% of full speed
    time.sleep(5)
    motor1.throttle = 0                 # Idle
    time.sleep(5)
    motor1.throttle = -0.25             # 25% of full speed
    time.sleep(5)
    motor1.throttle = 0                 # Idle
    time.sleep(5)
```

Tableau 1. Motor Driver Truth Table.

Input A	Input B	Output A	Output B	Motor Action
Low	Low	Low	Low	Brake
High	Low	High	High	Forward
Low	High	Low	High	Backward
High	High	Hi-Z (Open)	Hi-Z (Open)	Coast

Listage 2. DCMotor2.py.

```
#-------------------------------------------------------
#        TWO SPEED BRUSHED DC MOTOR CONTROL
#
# In this program a brushed DC moor is controlled as follows:
# The motor normally rotates at 25% of its full speed. Pressing
# button at port GP20 increses the speed to 50% of its full
# speed. Releasing the botton returns the speed to 25%
#
# Author: Dogan Ibrahim
# File  : DCMotor2.py
# Date  : February, 2023
#-------------------------------------------------------
import board
import time
import pwmio
import digitalio
from adafruit_motor import motor
#
# Initialize DC Motor 1
#
m1a = pwmio.PWMOut(board.GP8, frequency=50)
m1b = pwmio.PWMOut(board.GP9, frequency=50)
motor1 = motor.DCMotor(m1a, m1b)
#
# Configure button at port GP20. The button state is
# normally at logic 1
#
btn = digitalio.DigitalInOut(board.GP20)
btn.direction = digitalio.Direction.INPUT
btn.pull = digitalio.Pull.UP
while True:
    motor1.throttle = 0.25              # 25% of full speed
    while btn.value == 0:               # If button pressed
        motor1.throttle = 0.5           # Increase speed
```



▲
Figure 4. Schéma fonctionnel du projet.

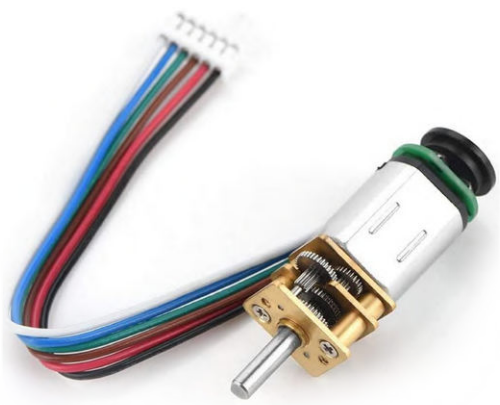


Figure 5. Moteur/encodeur, GBMQ-GM12BY20.

▶

Commande de moteur CC à deux vitesses

Il s'agit d'un projet très simple où un petit moteur CC à balais est connecté à la carte Maker Pi RP2040. Ici, nous utilisons également le bouton poussoir embarqué sur le port GP20. Normalement, le moteur tourne à 25 % de sa vitesse maximale. En appuyant sur le bouton, il passe à 50 % de sa vitesse maximale.

Les composants et les connexions entre le moteur et la carte électronique pour ce projet sont les mêmes que pour le projet précédent.

Le **listage 2** montre le programme *DCMotor2.py*. Le programme principal s'exécute dans une boucle, où l'état du bouton sur GP20 est vérifié. Tant que le bouton est pressé, la vitesse du moteur est augmentée à 50% de sa vitesse maximale.

Mesure de la vitesse d'un moteur à courant continu avec un encodeur rotatif

Ce projet montre un exemple d'utilisation d'un encodeur rotatif pour mesurer la vitesse d'un moteur CC. La vitesse est ensuite affichée à l'écran. Dans une «représentation en diagramme», cela ressemble à la **figure 4**. Notez que le moteur, avec son encodeur rotatif intégré, n'est pas inclus dans le kit de développement et vous devez l'acheter séparément. (Disponible sur eBay, AliExpress et autres.)

Nous utilisons un petit moteur CC à engrenages et à balais avec un encodeur rotatif intégré. La **figure 5** montre le moteur où l'encodeur rotatif est fixé à l'arbre arrière. Dans ce projet, le moteur suivant est utilisé : Moteur à engrenages CC 6 V, 2 W, GBMQ-GM12BY20 avec encodeur, 70 RPM. La vitesse à vide du moteur est spécifiée comme étant de 70 RPM. Dans ce projet, vous utiliserez une alimentation externe de +5 V pour le moteur et, par conséquent, vous vous attendrez à ce que la vitesse à vide soit inférieure à 70 RPM. Notez que la consommation de courant du moteur peut aller jusqu'à 200 mA.

Un encodeur rotatif (ou encodeur d'arbre) permet de convertir la position angulaire d'un composant en rotation, tel qu'un moteur, en un signal électrique. Les encodeurs rotatifs sont utilisés dans les applications de commande de moteur pour détecter la vitesse du moteur ou la position de l'arbre du moteur. Il existe essentiellement deux types d'encodeurs rotatifs : les encodeurs optiques et les encodeurs à effet Hall. Les encodeurs rotatifs optiques fonctionnent selon les principes de l'optique, où la lumière éclaire une photodiode à travers les fentes (ou trous) d'un disque métallique ou de toute autre forme. En comptant le nombre de trous passant devant la photodiode en un temps donné, vous (ou votre microcontroleur !) pouvez calculer la vitesse du moteur. Dans ce projet, nous utilisons un encodeur rotatif à effet Hall. Nous utilisons deux capteurs magnétiques statiques (appelés Phase A et Phase B) ainsi qu'un disque magnétique rotatif. Les capteurs fournissent des impulsions lorsque le moteur tourne. En comptant les impulsions

sur un temps donné, vous pouvez calculer la vitesse du moteur. La **figure 6** montre les formes d'ondes de sortie du moteur utilisé dans ce projet. La forme d'onde située en haut représente la phase A et celle située en bas la phase B. Ces types d'encodeurs sont également connus sous le nom d'encodeurs en quadrature car les capteurs magnétiques sont placés à 90° l'un par rapport à l'autre et il y a quatre états de sortie possibles. Le sens de rotation est facilement déterminé en trouvant l'ordre dans lequel les signaux de sortie passent de 0 à 1. Par exemple, si le signal Phase A remonte à partir de son état BAS, alors le moteur tourne dans un sens. Si, en revanche, le signal Phase B monte alors que le signal Phase A est au niveau BAS, alors le moteur tourne dans le sens opposé - voir **figure 7**.
Les caractéristiques du moteur utilisé dans ce projet sont les suivantes (en fonction de la tension appliquée) :

- Fonctionnement en courant continu (6 V)
- Vitesse de l'arbre après les engrenages : 70 RPM (à 6 V)
- Puissance : 2 W
- Courant 170…200 mA
- Deux capteurs à effet Hall intégrés
- Poids : 14 grammes

Un connecteur à 6 fils est fixé au panneau arrière du moteur. La configuration des broches du moteur est la suivante :

Fil noir	alimentation du moteur GND
Fil rouge	alimentation du moteur (+5 V dans ce projet)
Jaune	alimentation encodeur (+3.3 V dans ce projet)
Vert	alimentation encodeur GND
Bleu	sortie encodeur Phase A
Blanc	sortie encodeur Phase B

La **figure 8** montre le schéma de câblage du projet. Notez que le moteur fonctionne avec une alimentation externe de +5 V capable de fournir jusqu'à 500 mA. L'encodeur fonctionne sur une ligne de +3,3 V provenant de la carte de développement afin que la tension soit compatible avec les niveaux de tension d'entrée du processeur RP2040. Notez également que seule la sortie Phase A du moteur est utilisée dans ce projet.
Les fabricants et distributeurs de moteurs électriques n'indiquent pas le rapport d'engrenage et le nombre d'impulsions de l'encodeur émises par le moteur par seconde. Il s'agit pourtant de spécifications importantes. Dans cette section, nous écrivons un programme qui affiche ces paramètres importants du moteur utilisé. Vous devrez peut-être effectuer ces calculs pour trouver les spécifications importantes du moteur que vous utilisez. Le **listage 3** montre le programme MotorPulses.py écrit pour afficher le nombre d'impulsions du moteur en sortie de l'encodeur Phase A toutes les secondes. Dans ce programme, le port GP6 est assigné à l'objet `Encoder` et est configuré comme une entrée, et le nombre d'impulsions reçues de l'encodeur est calculé et affiché à la fin de chaque seconde. Notez que ce programme utilise la fonction `time.monotonic()` pour calculer le temps, et qu'il n'est pas très précis. L'idéal serait de recevoir les impulsions sous forme d'interruptions externes et d'utiliser des interruptions de temporisation d'une seconde pour calculer et afficher le nombre d'impulsions reçues chaque seconde. Malheureusement, CircuitPython ne prend pas en charge les interruptions externes ou temporisées.

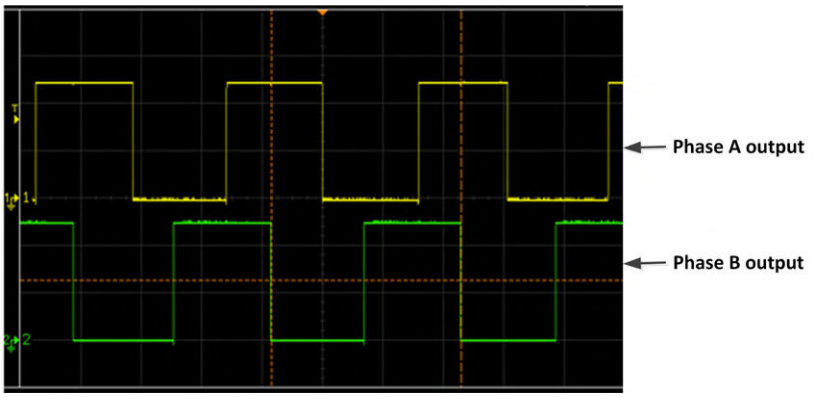


Figure 6. Formes d'onde de sortie de l'encodeur rotatif.

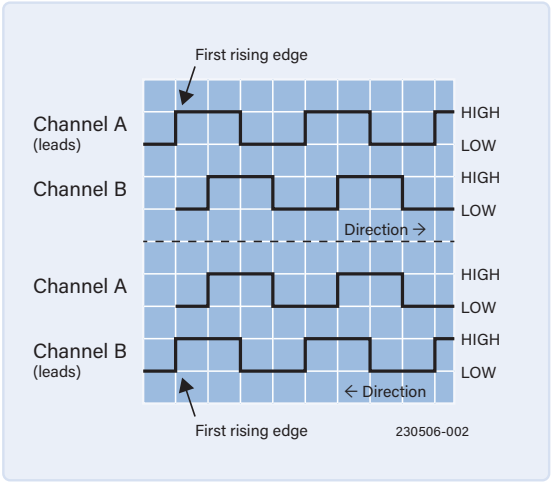


Figure 7. Détermination du sens de rotation (Source : robotoid.com).

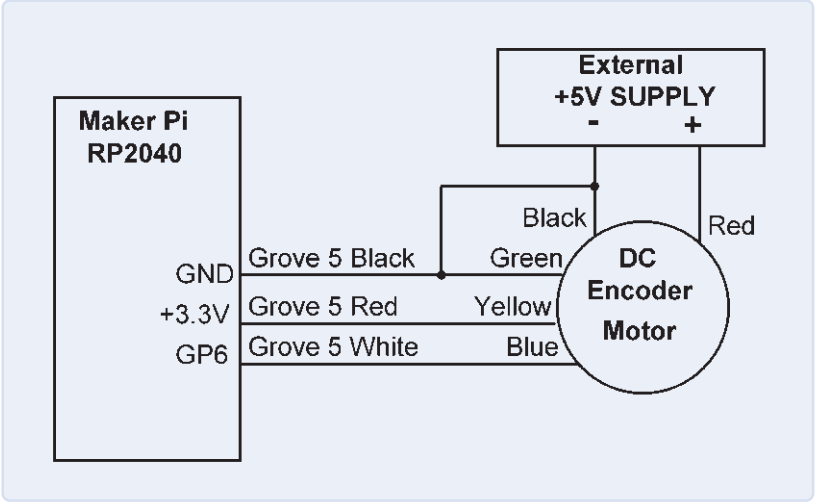


Figure 8. Le schéma de câblage du projet.

Listage 3. MotorPulses.py.

```
#-------------------------------------------------------
#       DISPLAYING THE NUMBER OF ENCODER PULSES
#
# In this program a geared DC motor with Hall Effect sensors
# is connected to the Maker Pi. This program calculates
# and displays the number of pulses received from the encoder
# every second. Only Phase A encoder output is used here
#
# Author: Dogan Ibrahim
# File  : MotorPulses.py
# Date  : March, 2023
#-------------------------------------------------------
import board
import digitalio
import time
Encoder = digitalio.DigitalInOut(board.GP6)
Encoder.direction = digitalio.Direction.INPUT
while Encoder.value == 0:          # Wait while 0
    pass
strtime = time.monotonic()         # Start time,rising edge
                                   # detected
count = 0                          # Intialize count
#
# Main program loop
#
while True:
    while Encoder.value == 1:      # Wait while 1
        pass
    while Encoder.value == 0:      # Wait while 0
        pass
    endtime = time.monotonic()     # End time
    if endtime-strtime > 1.0:      # Just over a second
        print(count)               # Display count
        strtime = time.monotonic()# Start time
        count = 0
    else:
        count = count + 1          # Increment count
    while(Encoder.value) == 1:     # Wait while 1
        pass
```

Listage 4: MotorSpeed.py.

```
#-------------------------------------------------------
#         DISPLAYING THE MOTOR SPEED
#
# This program displays the motor speed in RPM using the
# number of encoder pulses received every second and the
# formula given in the text
#
# Author: Dogan Ibrahim
# File  : MotorSpeed.py
# Date  : March, 2023
#-------------------------------------------------------
import board
import digitalio
import time
Encoder = digitalio.DigitalInOut(board.GP6)
Encoder.direction = digitalio.Direction.INPUT
while Encoder.value == 0:                Wait while 0
    pass
strtime=time.monotonic()                # Start time
count=0                                 # Intialize count
#
# Main program loop
#
while True:
    while Encoder.value == 1:           # Wait while 1
        pass
    while Encoder.value == 0:           # Wait while 0
        pass
    endtime=time.monotonic()            # End time
    if endtime-strtime > 1.0:           # Just over a second
        RPM = count * 60 / 880          # Motor speed
        print("Speed=%6.2f RPM" %RPM)   # Display speed
        strtime=time.monotonic()        # Start time
        count=0
    else:
        count=count+1                   # Increment count
    while(Encoder.value) == 1:          # Wait while 1
        pass
```

```
CircuitPython REPL
Speed=  63.95 RPM
Speed=  64.02 RPM
Speed=  64.02 RPM
Speed=  63.95 RPM
Speed=  63.89 RPM
Speed=  63.95 RPM
```

Figure 9. Sortie du programme.

Figure 10. Le tachymètre numérique DT-2234C.

Produits

- **Offre groupée Motor Control Development**
  www.elektor.fr/20534

Lorsque le moteur tourne, le programme *MotorPulses.py* affiche 938 pulses/seconde. Nous avons fixé un objet de forme ronde (un petit pneu) à l'arbre du moteur et nous avons fait une marque sur celui-ci afin de pouvoir compter les rotations, et nous avons observé que le nombre de tours de l'arbre du moteur à vide était de 64 RPM.
Or, le nombre d'impulsions reçues chaque minute est de 938 × 60 = 56,280 impulsions/minute, et cela correspond à une vitesse à vide de 64 RPM. Par conséquent, 56 280/64 = 879,38, ou 880 comme nombre entier le plus proche, puis la vitesse du moteur peut être calculée en utilisant la formule suivante :

vitesse (RPM) = (nombre d'impulsions mesurées par seconde × 60) / 880

Par exemple, si le nombre d'impulsions reçues est compté à 450 par seconde, la vitesse du moteur est la suivante :

vitesse = 450 × 60 / 880 = 30.68 RPM

La formule ci-dessus sera utilisée pour calculer la vitesse du moteur.
Le programme MotorSpeed.py présenté dans le **listage 4** est utilisé pour calculer puis afficher la vitesse du moteur en utilisant la formule ci-dessus. Ce programme est essentiellement le même que dans le listage 3, mais ici la vitesse du moteur est calculée et affichée à l'écran. La **figure 9** montre un exemple de sortie du programme.
Comme nous l'avons vu précédemment, nous aurions pu obtenir des résultats plus précis si des interruptions externes avaient été utilisées pour compter les impulsions de l'encodeur, et des interruptions de la minuterie pour mesurer le temps.

RP2040 ?

Vous pouvez facilement mesurer et **vérifier** la vitesse du moteur à l'aide d'un tachymètre numérique. Il existe de nombreux appareils de ce type à des prix différents. Celui utilisé par l'auteur était un modèle DT-2234C (**figure 10**) qui lui a coûté environ 8 £ plus le coût d'une seule pile de 9 V de type PP3. Avant d'utiliser cet appareil, il faut fixer un morceau de papier réfléchissant sur l'arbre du moteur, comme le montre la **figure 11**.
La vitesse du moteur est mesurée avec le tachymètre DT-2234C comme suit :

- Mettez le moteur en marche.
- Appuyez sur le bouton TEST du tachymètre et dirigez la lumière de l'appareil vers le papier réfléchissant.
- La vitesse du moteur s'affiche en continu sur l'écran LCD.
- Vous pouvez sauvegarder les relevés en appuyant sur le bouton MEM.

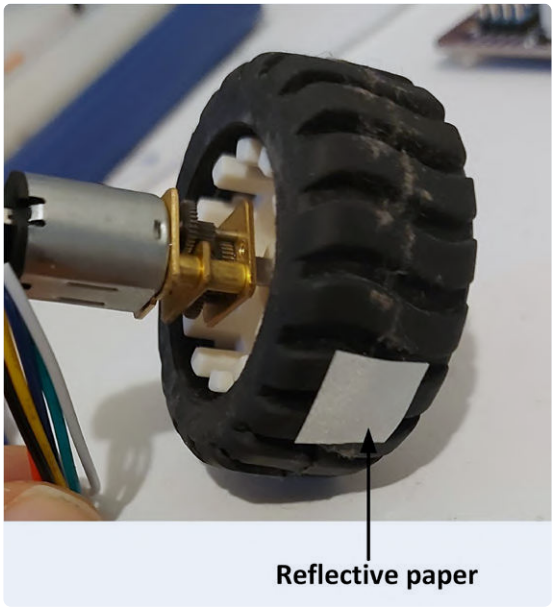


Figure 11. Fixation d'un morceau de papier réfléchissant sur l'arbre du moteur.

Note : Dans ce projet, nous n'utilisons qu'une seule phase de sortie de l'encodeur (Phase A). Il est possible d'obtenir des résultats plus précis sur la vitesse du moteur si les deux phases de l'encodeur (c'est-à-dire Phase A et Phase B) sont utilisées. Il est également possible de commander le moteur à partir des bornes à vis MOTOR 1 ou MOTOR 2 de la carte de développement Maker Pi RP2040. Cela limitera la tension maximale du moteur à +3,3 V et exigera également que la vitesse du moteur soit contrôlée en envoyant des formes d'ondes de tension PWM au moteur par les ports GP8/GP9 (MOTOR 1) ou les ports GP10/GP11 (MOTOR 2).

230506-04

Questions ou commentaires ?

Envoyez un courriel à l'auteur (d.ibrahim@btinternet.com) ou contactez Elektor (redaction@elektor.fr).

À propos de l'auteur

Dogan Ibrahim est titulaire d'une licence (avec mention) en ingénierie électronique, d'une Master en ingénierie du contrôle automatique et d'un doctorat en traitement des signaux numériques et microprocesseurs. Dogan a travaillé dans de nombreuses organisations et est membre de l'Institution of Engineering and Technology (IET) au Royaume-Uni et ingénieur électricien agréé. Dogan est l'auteur de plus de 100 livres techniques et de plus de 200 articles techniques sur l'électronique, les microprocesseurs, les microcontrôleurs et les domaines connexes. Dogan est un expert certifié d'Arduino et a de nombreuses années d'expérience avec presque tous les types de microprocesseurs et de microcontrôleurs.

LIEN

[1] Motor Control Development Bundle: https://elektor.fr/motor-control-development-bundle

Adaptateur ESP32-RS-232

liaison sans fil pour les équipements de test classiques

Georg Hofstetter (Allemagne)

RS-232 devient sans fil ! Un module ESP32 moderne et peu coûteux donne des ailes à une interface série. En plus de sa capacité de communication par liaison série, ce petit adaptateur permet aux équipements de test SCPI de recevoir des commandes et d'envoyer des données sur le réseau domestique via MQTT.

En travaillant sur différents projets électroniques au fil des ans, les équipements de test sur votre établi ne cessent de s'accumuler. Dans mon cas, j'utilise un SourceMeter 2400 de Keithley pour détecter les défauts dans les appareils électroniques, les alimenter ou même mesurer les propriétés de divers composants. C'est également un instrument très utile pour tester le courant de fuite des condensateurs électrolytiques ou mesurer leurs capacités nominales. J'utilise également un récepteur AOR AR5000 pour communiquer avec les radioamateurs du monde entier. Ces deux appareils, comme beaucoup d'autres dans le labo, peuvent être contrôlés par un PC via leurs interfaces série RS-232 intégrés. La disponibilité de câbles adaptateurs USB-RS-232 rend la connexion à un PC personnel assez simple. Le câble adaptateur que j'ai utilisé a bien fonctionné avec l'AR5000, mais la communication avec le SourceMeter de Keithley s'est avérée beaucoup moins stable. Des caractères se perdent parfois lors de l'envoi de commandes SCPI et la connexion a été souvent interrompue.

La connexion RS-232

La norme RS-232, établie dans les années 1960, permet de transmettre des bits de données sérielles d'un caractère en utilisant des changements de niveaux de tension. Les propriétés électriques ont été définies dans la norme V.28, tandis que le fonctionnement général a été réglementé par la norme V.24. La norme V.28 spécifie que, selon la valeur du bit logique, un « 0 » logique est transmis comme un front montant allant jusqu'à 5...12 V et un « 1 » logique comme un front descendant allant jusqu'à -5...-12 V. Le récepteur a une plage de tension légèrement plus large et interprétera un signal dans la plage de 3 à 12 V comme un «0» logique et un signal dans la plage de -3 à -12 V comme un «1» logique. Cela tient compte de la dégradation du signal dans le câble due au bruit, à la chute de tension et à la déviation des fronts du signal causée par l'impédance du câble.

Dans les anciens adaptateurs USB-série peu coûteux et dans certains ordinateurs portables, la variation du niveau du signal du côté de l'émetteur dépasse à peine ±5 V, voire ±3 V, en raison des simplifications apportées à la conception du pilote du signal RS-232 visant à minimiser les coûts de développement et de composants. Cela a conduit à la croyance populaire que les ordinateurs portables et les dispositifs RS-232 ne fonctionnent souvent pas bien ensemble. L'image de la **figure 1**, extraite d'un document de TI, montre les niveaux de tension conformes aux normes.

Que le problème de communication susmentionné avec l'instrument Keithley soit dû à l'utilisation d'un câble d'extension USB, à une boucle de masse ou aux caractéristiques électriques de l'émetteur-récepteur intégré, il n'a pas d'importance ici. Après tout, un câble USB tendu à travers le labo n'est pas seulement encombrant, mais il présente également un risque de trébuchement. Ce dont nous avons besoin, c'est une solution sans fil compacte ! J'ai cherché en vain une solution prête à l'emploi à mon problème, mais je n'ai trouvé que des modules assez volumineux montés

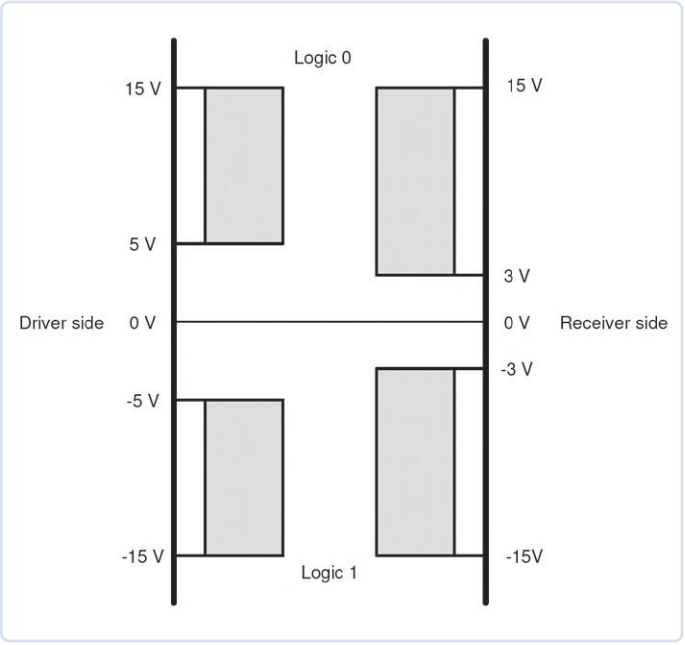


Figure 1. Niveaux de signal RS-232 spécifiés à la sortie de l'émetteur et à l'entrée du récepteur. (Source : Texas Instruments [9])

sur rail DIN, ou des circuits *DIY* plus conçus pour résoudre un problème spécifique et présentant trop peu de fonctionnalités pour moi. Inutile : si je voulais une unité de communication radio RS-232 propre et universelle, je devais prendre le temps de la concevoir moi-même.

Critères de conception

Ce projet est destiné aux labos de *maker* et ne sera pas un produit joliment emballé que l'on peut acheter dans un magasin d'électronique. Cependant, il est préférable d'examiner attentivement la conception de l'appareil afin qu'il puisse effectuer toutes les tâches pour lesquelles il est prévu.

Un ESP32 fait tout le travail

La passerelle RS2-32 est principalement destinée à transmettre des données vers et depuis le réseau. Cela se fait généralement via le port TCP 23 (Telnet). Cependant, des modifications logicielles spécifiques à l'application devraient également être possibles, que l'utilisateur peut adopter en fonction des exigences de l'application. Pour moi, un service SCPI-to-MQTT était important, pour que l'équipement de test SourceMeter puisse transmettre ses mesures pendant la nuit à Grafana (voir ci-dessous) à des fins d'évaluation.

L'ESP32 offre une puissance de calcul suffisante (en plus de l'envoi de simples paquets TCP) pour que l'appareil connecté puisse être facilement intégré à Home Assistant ou à un service similaire via MQTT ou pour le partager via un frontal dédié via le serveur web de l'ESP32.

Une alternative serait l'ESP8266, qui prend un peu moins de place et est un peu moins cher. Cependant, l'encombrement réduit se fait au prix d'un processeur considérablement plus sollicité et d'une mémoire RAM réduite. En outre, l'ESP8266 ne dispose pas d'un module Bluetooth (avantageux dans cette application) en plus de la fonction Wifi souhaitée.

L'inconvénient de l'ESP32 est, bien entendu, sa consommation de courant considérablement plus élevée, qui atteint un pic d'environ 500 mA sur le rail de 3,3 V. Le CPU ne se chargera évidemment pas de calculer en permanence, mais selon le degré d'optimisation finale du logiciel, le courant de crête atteindra probablement cette valeur, comme l'indique le tableau 4.2 de la fiche technique [1]. Une comparaison avec la fiche technique de l'ESP8266 [2] montre que la consommation d'énergie de l'ESP32 est supérieure d'environ 35 % en émission et d'environ 80 % en réception.

Un design soigné et compact

La plupart des gens conviendront probablement qu'il y a déjà suffisamment de câbles qui traînent sur et sous les établis. Il est possible de souder ce circuit imprimé directement à un connecteur D-sub DE-9. Il n'est pas beaucoup plus grand qu'une prise secteur IEC C13 standard (y compris le rayon de courbure). Cela signifie qu'une fois connectée, la carte n'augmente pas l'encombrement de l'appareil RS-232. Si un boîtier est nécessaire, il est possible de l'imprimer facilement en 3D ; dans l'idéal, le circuit imprimé peut même s'insérer dans un boîtier standard disponible sur le marché.

Alimentation par micro-USB

La carte est alimentée via un connecteur micro-USB (désormais le standard pour de nombreux petits appareils). La tension +3,3 V requise par l'ESP32 est fournie par le régulateur de tension linéaire Advanced Monolithic Systems [3] AMS1117 à partir de l'alimentation +5 V du port USB. Bien que l'AMS1117 ait une capacité de 1 A, il chauffe beaucoup sur ce circuit imprimé relativement petit. Une alternative plus efficace est le convertisseur abaisseur à découpage de Texas Instruments [4] TPS62291, livré dans un boîtier WSON-6 de 2 × 2 mm. Il fournit également un courant maximum de 1 A et permet de réduire l'encombrement. À moins de disposer d'un microscope et tous les outils et l'expérience nécessaires pour travailler à ce niveau, cette puce est un peu plus difficile à assembler en raison de la petite taille du boîtier. Ceux qui doutent de leurs compétences en soudage peuvent trouver des conseils utiles dans la vidéo [5].

Il convient également de noter que la disponibilité de ce composant particulier a été irrégulière au cours des deux dernières années. En conséquence son prix a grimpé à plus de 20 € chez certains distributeurs, au lieu des 1,70 € habituels. En comparant les prix sur internet, on peut trouver des exemplaires à des prix qui rappellent le bon vieux temps. Pour simplifier les choses, j'ai utilisé le AMS1117 dans cette version, ce qui facilite grandement la soudure à la main.

Alimentation via le connecteur sub-D à 9 broches

Si vous n'êtes pas opposé à l'idée de sortir des sentiers battus, vous pouvez également alimenter la carte via le connecteur D-Sub à 9 broches. En modifiant légèrement le ou les terminaux, il est possible de fournir les 5 V requis via le connecteur RS-232 (et d'éviter ainsi l'utilisation d'une alimentation supplémentaire). Cependant, la norme ne prend pas en charge une telle option d'alimentation via le connecteur, nous devons donc noter que cette modification n'est pas non plus conforme à la norme.

Selon la norme V.28, toutes les lignes de signal peuvent supporter des tensions de +15 V à -15 V, pour l'alimentation, il est logique de remplacer l'une des fonctions des broches qui est rarement utilisée. La broche 9 est ce qu'on appelle l'indicateur de sonnerie (RI) et n'est utilisée par un modem que pour signaler un appel entrant. Les modems RS-232 d'aujourd'hui sont pratiquement redondants, ce qui fait de la broche RI un candidat idéal pour l'alimentation.

Toutefois, nous devons d'abord vérifier ce qui se passe si nous appliquons subitement la

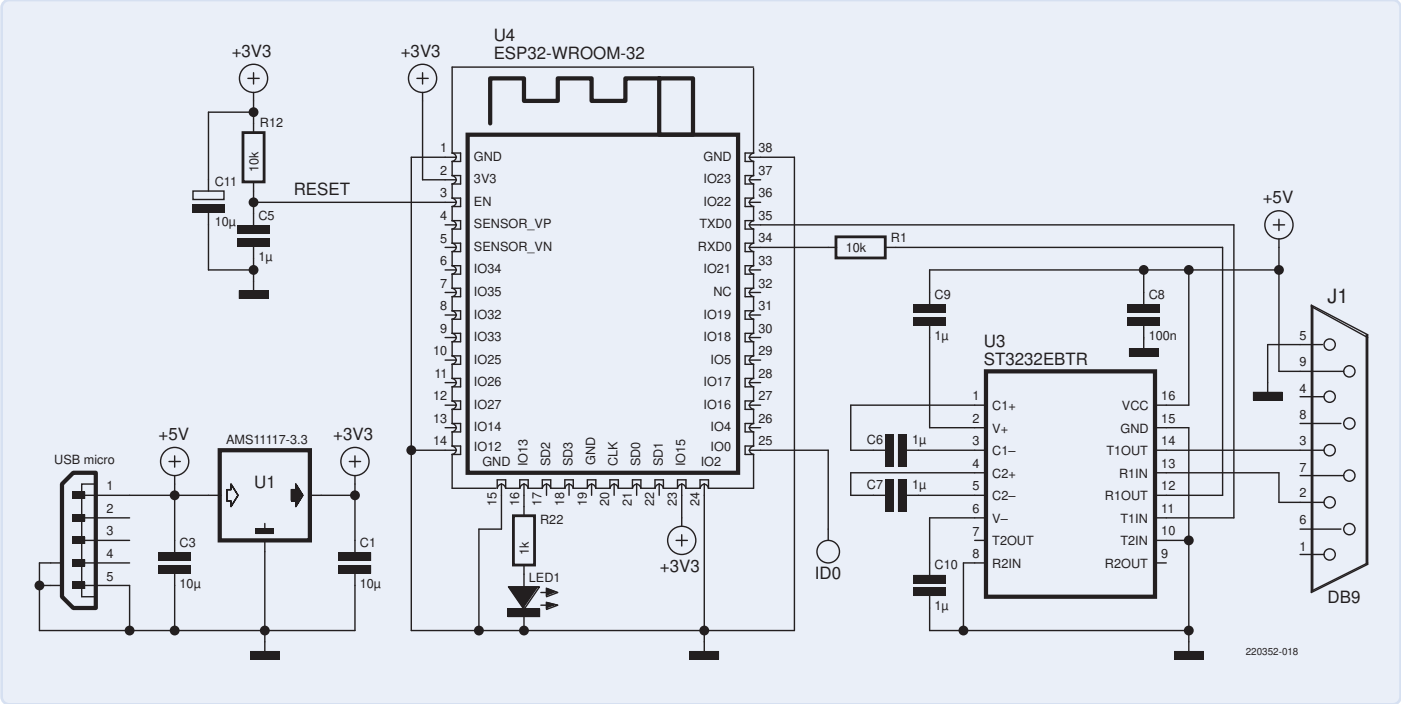


Figure 2. Le schéma simple du dongle ESP32/RS-232.

tension maximale, par exemple en connectant un terminal qui alimente notre ligne d'alimentation conformément à la norme V.28.
La tension +15 V ne poserait pas de problème à l'AMS1117 lui-même ; cette tension est dans ses limites spécifiées. Cependant, si une tension -15 V y est appliquée, les diodes de protection internes du régulateur AMS conduiront les 10 à 20 mA du signal RS-232 à la masse. Empiriquement, ces diodes supportent cette tension et, selon leurs spécifications, les pilotes RS-232 sont protégés contre les courts-circuits, pour qu'aucun dommage n'est à déplorer ici non plus.
La valeur de la tension d'alimentation VDD de notre émetteur-récepteur RS-232 est spécifiée jusqu'à 5,5 V et est également connectée au +5 V. Dans ce cas, la tension chute à environ 3,5 V en raison de la limitation du courant par les pilotes RS-232. Certes, nous sommes un peu courts ici, mais nous n'avons pris en compte tous ces facteurs que pour le cas rare où RI sur la broche 9 est effectivement piloté par l'appareil final. La plupart du temps, cette broche n'est pas connectée.

True RS-232

Comme nous l'avons déjà mentionné, notre objectif est de respecter le plus possible la norme V.28, c'est pourquoi un émetteur-récepteur de signaux de données est absolument nécessaire. Le ST232EBTR de ST [6] fera l'affaire et est disponible dans un boîtier TSSOP-16. Ce CI est abordable (environ 1 €) et comporte un doubleur de tension interne et des circuits inverseurs, il génère des tensions d'alimentation de ± 10 V, qui sont utilisées pour les signaux RS-232 (qui sont typiquement d'environ 9 V dans des conditions normales de fonctionnemen).
Le schéma du dongle ESP32-RS-232 avec le module ESP32-WROOM (ou le WROVER avec PSRAM) et les deux CI (ST232EBTR et AMS1117), ainsi que tous les composants périphériques, sont représentés dans la **figure 2**.

Placement des composants

Pour un circuit imprimé aussi petit et comportant aussi peu de composants, il est peu judicieux de recourir à des professionnels pour le montage des composants ou d'engager des dépenses supplémentaires pour un pochoir de pâte à souder et un assemblage avec une plaque de refusion. Nous pouvons souder à l'ancienne (à la main) le circuit imprimé illustré à la **figure 3**, puisque le nombre de composants est faible et les pastilles sont accessibles.
Le premier composant à assembler est le ST232 (U3). Il est relativement plat, et quelques retouches peuvent être nécessaires pour obtenir des connexions nettes. Ensuite, vous pouvez souder les condensateurs, les résistances et la LED, puis l'AMS1117 (U1) et le connecteur USB. Il ne reste plus qu'à souder l'ESP32 et, enfin, le connecteur D-Sub à 9 broches.

Micrologiciel

La fonction principale du micrologiciel est d'acheminer les caractères reçus sur le réseau vers le port RS232 et vice versa, mais grâce à la polyvalence de l'ESP32, il est possible d'aller plus loin. Le micrologiciel, disponible dans le dépôt [7], offre actuellement deux modes de fonctionnement - Telnet et MQTT/SCPI - que nous expliquerons ci-dessous.

Telnet / Socket TCP brut

Comme nous l'avons déjà évoqué dans les critères de conception, la fonction principale est de transférer des données du réseau vers le port RS-232. Pour de tels adaptateurs, il est courant de transmettre des caractères bruts via le port TCP 23. Ce port était auparavant utilisé pour les connexions Telnet à un serveur mais est rarement utilisé aujourd'hui (sauf dans les systèmes embarqués). Pour notre objectif : l'envoi de commandes ou de mesures vers et depuis des périphériques série, il est suffisant dans un réseau domestique.

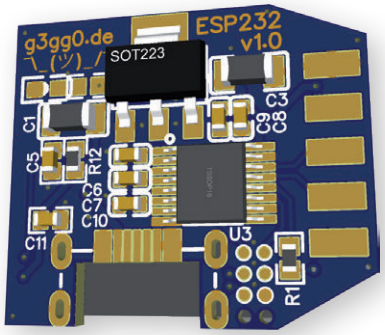


Figure 3. Ce petit circuit imprimé avec des pastilles facilement accessibles est idéal pour souder à la main des CMS.

Figure 4. Exemple de tableau de bord Grafana, montrant des formes d'onde générées à partir de valeurs mesurées au fil du temps (sans rapport avec ce projet).

Telnet est un protocole de communication réseau développé pour permettre d'accéder à distance à un ordinateur sur un même réseau. Il fournit un canal de communication textuel permettant à un utilisateur d'accéder à un ordinateur distant et d'exécuter des commandes. La connexion est bidirectionnelle, de sorte que les entrées et les réponses sont toutes deux transmises sur le réseau. Développé à l'origine à l'époque plus primitive des premiers standards Internet, Telnet est souvent utilisé dans des applications où une simple communication textuelle est suffisante. Il est cependant considéré comme peu sécurisé car il n'est pas crypté, ce qui permet potentiellement l'interception de toutes les données transférées, y compris les mots de passe et d'autres informations sensibles.
Pour les premiers tests de communication, vous pouvez utiliser l'utilitaire *telnet.exe*, fourni par défaut dans les versions antérieures de Windows. S'il n'est pas disponible, vous pouvez utiliser l'alternative gratuite et plus universelle PuTTY [8].

MQTT/SCPI

Pour les applications qui nécessitent des relevés de mesures réguliers sur des périodes de plusieurs heures, vous pouvez développer des scripts Python appropriés ou même lancer des outils prêts à l'emploi sur votre PC et les lire via le port Telnet. Une alternative pratique pour ceux qui ont déjà mis en place une infrastructure appropriée dans leur réseau domestique consiste à se passer complètement du PC gourmand en énergie et à utiliser une solution de mesure plus simple.
De nombreux ménages utilisent déjà des services tels que MQTT, InfluxDB et Grafana pour collecter les données de capteurs avec un Raspberry Pi ou via des conteneurs Docker à l'aide de leur NAS. Sans entrer dans les détails et en termes simples :

- **MQTT** est un protocole de messagerie léger conçu pour une communication efficace entre les appareils IdO et les réseaux à faible bande passante et à latence élevée. Il utilise un courtier en informations avec un modèle de publication/abonnement pour permettre un échange de données transparent avec une surcharge minimale.
- **InfluxDB** est une base de données de séries temporelles conçue pour gérer efficacement des charges d'écriture/demande élevées pour des données horodatées. Les balises sont des paires clé/valeur pour l'indexation, tandis que les champs contiennent les données réelles.
- **Grafana** extrait les valeurs de données d'une base de données et affiche les informations de manière conviviale sous la forme de graphiques définis par l'utilisateur.

Pour ceux qui disposent d'une telle installation chez eux, ce micrologiciel offre la possibilité de publier les relevés générés par l'appareil

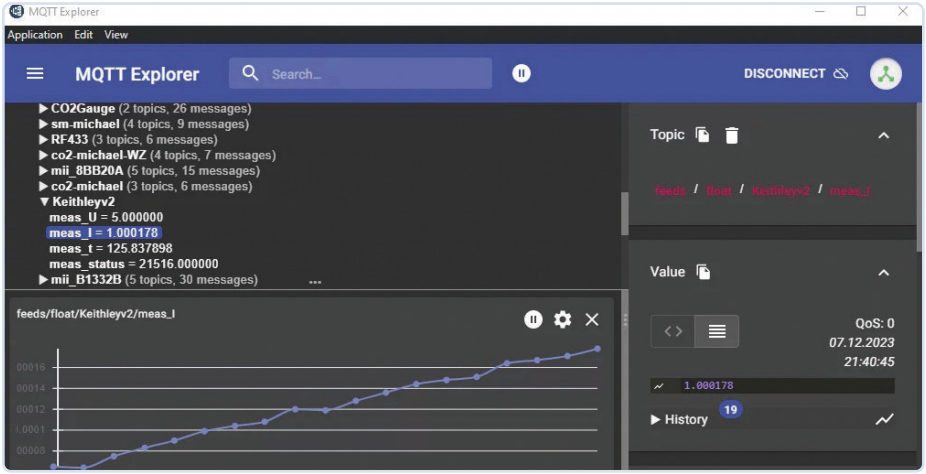


Figure 5. MQTT Explorer montre les mesures publiées par l'adaptateur ESP32/RS-232 envoyées via MQTT.

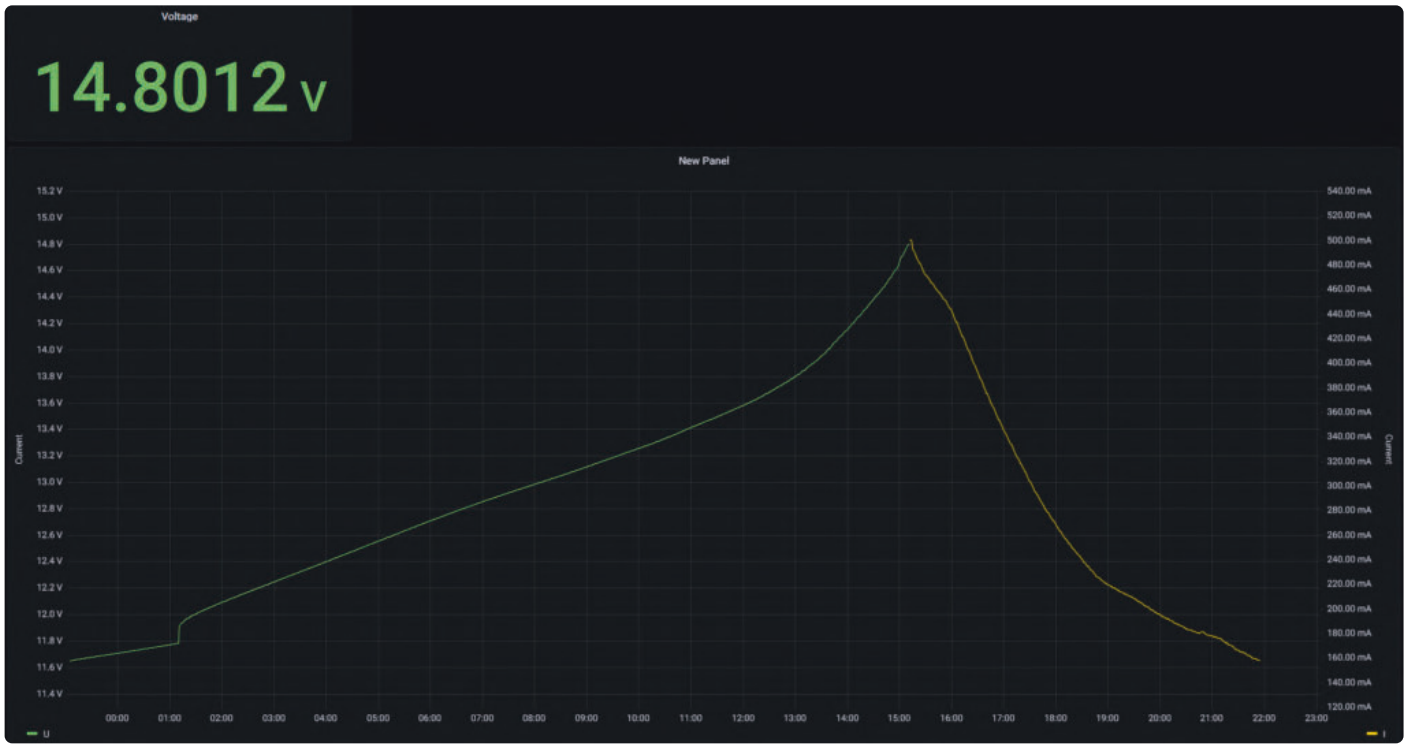


Figure 6. Tension (verte) et courant (jaune) de la batterie pendant la charge d'une batterie au plomb de 12 V, mesurés via un adaptateur ESP32/RS232.

compatible SCPI directement vers un courtier MQTT pour une diffusion ultérieure.
L'ensemble des commandes utilisées par les différents appareils SCPI varie considérablement. Le seul appareil actuellement pris en charge par le code est le SourceMeter 2400 de Keithley Toutefois, il s'agit d'un bon point de départ permettant à chacun d'ajouter ses propres versions de micrologiciels. Les développeurs embarqués sont invités à étendre les fonctionnalités du logiciel et à les partager avec la communauté.
Pour voir les représentations des mesures dans différentes applications, voir la **figure 4** (Grafana), la **figure 5** (MQTT Explorer, un client MQTT complet pour différentes plateformes [10]) et la **figure 6** (Grafana également).
L'ESP32 doit donc effectuer certaines tâches et pour cela, il a besoin d'informations que nous devons lui fournir via une interface web.

Configuration du Wifi

Si aucun point d'accès configuré n'est trouvé ou si aucun n'est configuré, l'ESP32 devient actif et se propose comme point d'accès nommé *esp232-config*. Il est préférable de se connecter à ce point d'accès avec un smartphone. Après s'être connecté, vous pouvez configurer les paramètres principaux en allant sur *http://192.168.4.1/* dans un navigateur. La page web hébergée sur l'ESP32 est présentée dans la **figure 7**. Dans le **tableau 1**, vous pouvez vérifier ce que vous doit être saisi dans chaque ligne.

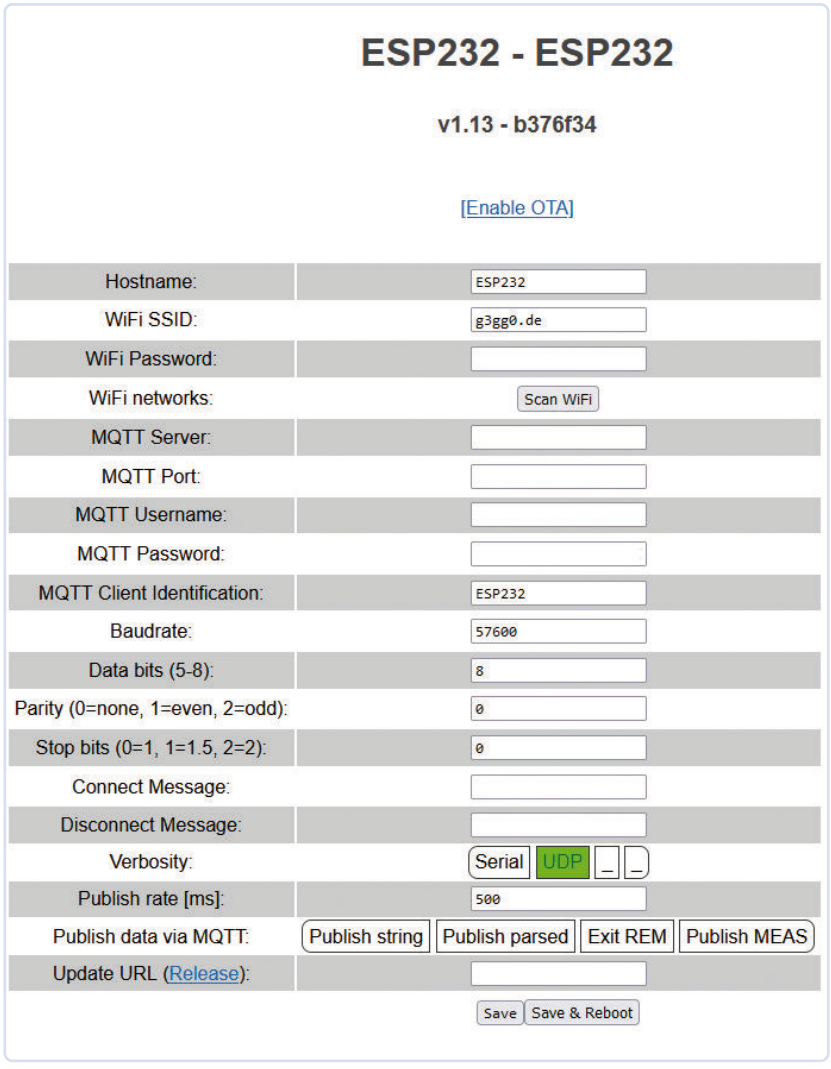


Figure 7. Configuration Wifi de l'ESP32.

Tableau 1. Données de la configuration Wifi de l'ESP32.

Nom d'hôte	Le nom que l'appareil utilisé pour s'identifier via MDNS sur le réseau.
WiFi SSID/mot de passe Wifi	Le réseau Wifi auquel l'appareil doit se connecter au démarrage.
Réseaux Wifi	Affiche une liste des réseaux Wifi trouvés ; en cliquant sur un réseau, son nom s'affiche.
MQTT [...]	Paramètres de configuration pour le client MQTT, en conjonction avec le SCPI ci-dessous.
Débit en bauds / Bits de données / Bits d'arrêt	Paramètres correspondants pour la communication RS232.
Dis-/Connect Message	[cas particuliers] Message à envoyer via RS232 pour une connexion TCP.
Verbosity	[Experts] Communication de débogage via UDP.
Publier des données via MQTT	[Experts] Pour le Source Meter 2400 de Keithley : analyse les données d'affichage ou lance une mesure et publie les résultats via MQTT.
URL de mise à jour	Pour les mises à jour du micrologiciel, il suffit d'entrer les URL HTTP et le fichier .bin attendu sera chargé et mis à jour.

Tableau 2. Affectation des broches de l'interface de programmation.

Broche	Rôle	Niveau
IO0	Définit le mode de démarrage, n'a d'effet qu'après une réinitialisation.	Flash : GND (normal : flottante, 3,3 V)
RESET	Broche reset de l'ESP32 (CHIP_PU).	active : GND, inactive : flottante (3,3 V)
TXD0	Signal d'émission de l'ESP32	0V / 3,3 V
RXD0 (U3_34)	Signal de réception de l'ESP32	0V / 3,3 V
+5V	Tension d'alimentation	5 V
GND	Masse	GND

OTA : mises à jour à distance

Dans l'EDI Arduino ou dans PlatformIO, que je préfère, il est possible de mettre à jour l'ESP32 « à distance «. Le protocole sous-jacent est implémenté par le composant ArduinoOTA. Bien que cette fonctionnalité facilite les mises à jour fréquentes du microcontrôleur, en pratique, des inconvénients significatifs sont apparus. Il semble que l'allocation et la libération constantes de tampons pour chaque paquet UDP reçu fragmente la mémoire de l'ESP32, ce qui peut conduire à des réinitialisations imprévisibles. De plus, cela se produit indépendamment du fait que des paquets OTA soient en cours d'acheminement. Comme vous pouvez l'imaginer, un message de crash exécuté via l'UART de l'ESP32 est rarement bien reçu par l'appareil connecté. Heureusement, il existe des améliorations [11] qui réduisent de manière significative le taux de crash. À cause de ce niveau d'incertitude, la fonction OTA n'est activée que brièvement et à la demande de l'utilisateur. C'est pourquoi un champ (*Enable OTA*) est prévu à cet effet dans l'interface web.

Flashage

Si vous ne disposez pas déjà d'un adaptateur Pogo pour les modules ESP32, vous devez programmer le microcontrôleur avec micrologiciel correct après l'assemblage en utilisant les broches TX et RX de l'ESP32. Pour ce faire, téléchargez le code source à partir de [7], importez-le dans PlatformIO, compilez-le et flashez-le sur la carte.

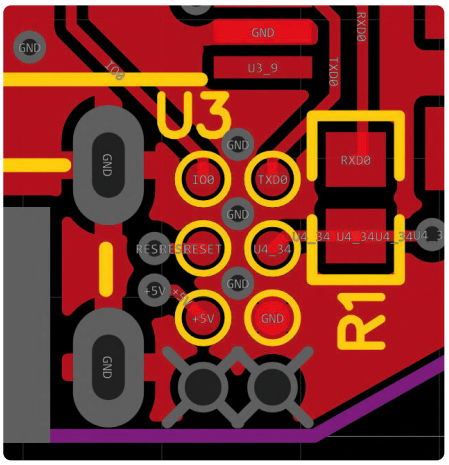


Figure 8. Connexions pour le programmateur sur le circuit imprimé.

Un port de programmation est fourni en bas à droite de la carte. Pour une utilisation sporadique, vous pouvez simplement mettre à la masse le signal IO0 et souder les TXD0 et RXD0 (broche U4_34 du microcontrôleur, voir **figure 8**) à un adaptateur USB/UART.

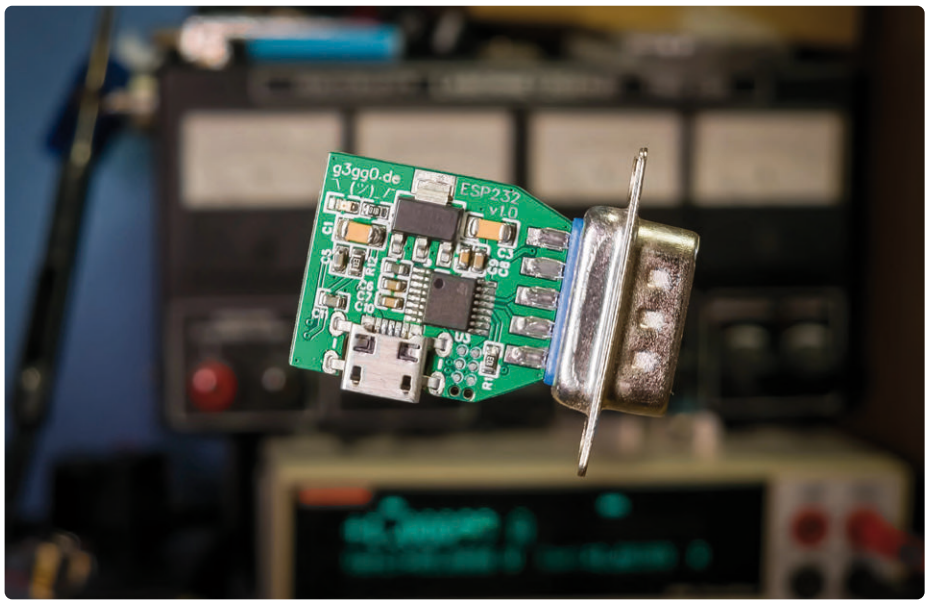
Comme le montre le **tableau 2**, l'ESP32 possède quelques broches de «strapping « que vous devez configurer lorsque vous chargez le code ROM en utilisant cette méthode. Dans cette configuration, nous n'avons à nous préoccuper que de l'une d'entre elles (IO0). Une fois le micrologiciel chargé, une LED devrait commencer à clignoter après environ 30 secondes.

Tout n'est pas beau

L'ESP32 a prouvé ses capacités dans mes projets. Cependant, il y a eu au départ divers problèmes avec la bibliothèque Arduino concernant l'utilisation de la mémoire tampon UART, liés au code accédant directement aux bibliothèques du cadre IDF. Des bogues dans la bibliothèque *ArduinoOTA* m'ont également amené à passer plusieurs nuits à essayer d'identifier les problèmes. En conséquence, il existe certaines solutions de contournement dans le code source. Mais comme ce micrologiciel est open source, il continuera à évoluer et à bénéficier des améliorations apportées par la communauté.

La prochaine version

Ce qui m'a gêné dans la version v1.0 du projet jusqu'à présent, c'est la lourdeur du processus de flashage un peu (ou le flashage pour corriger des erreurs dans le code). Grâce à un adaptateur de flash avec des broches Pogo que j'utilise pour tous mes projets, l'effort est considérablement réduit, mais tout le monde n'a pas un adaptateur approprié à portée de



main ou n'aime pas travailler avec des câbles de connexion.

Pour cette raison, la version 2.0 de l'adaptateur ESP32-RS-232 est en cours de développement et utilise un ESP32-S3-PICO. Quelques prototypes initiaux ont déjà été produits et semblent prometteurs. Le PICO est un module complet avec une mémoire flash SPI et une mémoire RAM, le tout intégré dans un boîtier LGA-56. Bien Bien qu'à première vue, le PICO ressemble à un QFN, qui est généralement facile à souder à la main, la masse (GND) doit être connectée à une pastille visible, ce qui nécessite de souder la pastille à travers un trou lorsqu'il est assemblé manuellement. Ce boîtier n'a pas non plus de broche de connexion et il est difficile de souder les broches à partir du bord. En fin de compte, une (mini) plaque de refusion semble être la meilleure solution dans ce cas [12].

Le grand avantage du S3 est qu'il dispose de toute la périphérie USB à bord pour le flashage ou le débogage. Si nécessaire, il est possible d'utiliser l'ESP232 v2.0 comme adaptateur USB vers RS232.

La question de savoir si cela nous permet de sortir de notre point de départ est une question rhétorique. Si l'on considère que l'ESP8266 était idéal pour construire un adaptateur WiFi/série dès le départ, on peut se demander pourquoi un produit similaire n'est pas encore sur le marché aujourd'hui.

220352-04

Questions ou commentaires ?

Envoyez un courriel à l'auteur (elektor@g3gg0.de), ou contactez Elektor (redaction@elektor.fr).

À propos de l'auteur

L'intérêt de Georg Hofstetter pour l'électronique est né à l'époque du Commodore 64 et des cartes perforées. Après une formation d'ingénieur en électronique, il a obtenu un diplôme en informatique. Depuis lors, il développe ou fait de la rétro-ingénierie de logiciels et de systèmes embarqués et publie des projets sélectionnés sur son site web www.g3gg0.de. Parmi les projets les plus connus, on peut citer des modifications de micrologiciels pour les DSLR Canon appelées Magic Lantern (*www.magiclantern.fm*) et pour le Toniebox (*https://gt-blog.de/toniebox-hacking-how-to-get-started*).

Produits

- **ESP32-WROOM-32**
  www.elektor.fr/18421
- **ESP32-S2-WROVER**
  www.elektor.fr/19693

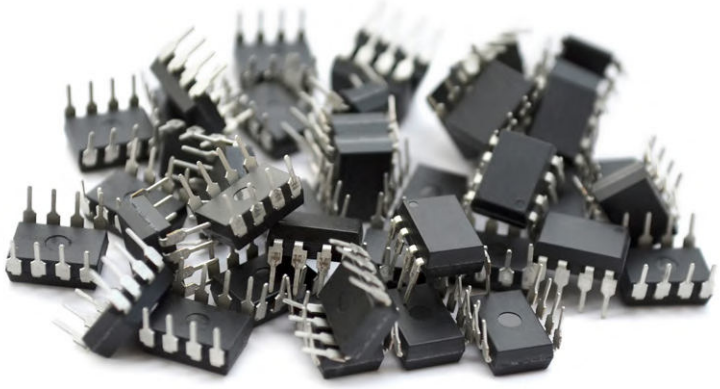
LIENS

[1] Fiche technique de l'ESP32 : https://www.espressif.com/sites/default/files/documentation/esp32_datasheet_en.pdf
[2] Fiche technique de l'ESP8266 : https://www.espressif.com/sites/default/files/documentation/0a-esp8266ex_datasheet_en.pdf
[3] AMS1117 : http://www.advanced-monolithic.com/pdf/ds1117.pdf
[4] TPS62291 : https://www.ti.com/product/de-de/TPS62291
[5] SMD soldering by hand, vidéo d'Elektor : https://www.elektormagazine.de/news/uberzeugendes-video-loten-von-smd-bauteilen
[6] ST232 : https://www.st.com/resource/en/datasheet/st202eb.pdf
[7] Dépôt du projet : https://github.com/g3gg0/ESP232
[8] PuTTY : https://www.putty.org/
[9] MQTT Explorer : https://mqtt-explorer.com/
[10] WiFiUDP Crash issue : https://github.com/espressif/arduino-esp32/issues/4104
[11] Texas Instruments Application Report: RS-232 Frequently Asked Questions : https://www.ti.com/lit/an/slla544/slla544.pdf
[12] M. Divito, « Mini plaque de refusion », Elektor 11-12/2023 : https://www.elektormagazine.fr/230456-04

démarrer en électronique

...en savoir plus sur les ampli-op

Eric Bogers (Elektor)



Dans l'épisode précédent, nous avons examiné les transistors à effet de champ discrets et nous avons abordé ce qui est probablement le composant le plus important de l'électronique analogique : l'amplificateur opérationnel (ampli-op). Le sujet est vaste, comme vous le verrez, et il existe de nombreuses formules en rapport avec les amplificateurs que nous pouvons utiliser (et que nous utiliserons !).

L'amplificateur opérationnel comme boîte noire

Un amateur d'électronique débutant souhaitant utiliser un ampli-op dans une application pratique s'intéresse peu à l'aspect intérieur du composant. En fait, il suffit de connaître quelques paramètres importants pour commencer à utiliser un ampli-op. Vous pouvez donc visualiser l'ampli-op comme une sorte de boîte noire avec des entrées et des sorties (**figure 1**).

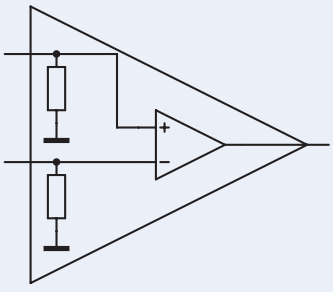


Figure 1. L'amplificateur opérationnel comme une boîte noire.

Le gain en boucle ouverte, qui se situe généralement entre 10 000 et 100 000, est le principal élément auquel vous devez prêter attention. Ce paramètre permet de calculer la tension de sortie grâce à la formule suivante, où U_{ni} est la tension sur l'entrée non inverseuse et U_i est la tension sur l'entrée inverseuse :

$$U_{out} = V_0 \cdot (U_{ni} - U_i)$$

Vous pourriez donc avoir l'impression que la valeur exacte du gain en boucle ouverte (c'est-à-dire le gain sans rétroaction négative) est d'une importance capitale. Or, ce n'est *pas* le cas, comme vous le verrez plus loin.

Pour les ampli-op constitués de transistors bipolaires, l'impédance d'entrée est de l'ordre de 1 MΩ. Si des transistors à effet de champ (FET) sont utilisés dans l'étage d'entrée de l'amplificateur, l'impédance d'entrée sera supérieure de plusieurs ordres de grandeur.

La tension d'alimentation maximale est généralement comprise entre ±16 V et ±22 V, mais elle peut atteindre ±150 V avec des ampli-op à haute tension. Le courant de sortie maximal est typiquement de l'ordre de 20 mA, bien que les ampli-op de puissance puissent supporter plusieurs ampères s'ils sont suffisamment refroidis. La fréquence de coupure sera discutée en détail plus loin.

Montages amplificateurs

Les deux montages amplificateurs, avec des ampli-op, couramment utilisés sont l'amplificateur inverseur et l'amplificateur non inverseur. Comme son nom l'indique, l'amplificateur inverseur inverse la polarité du signal d'entrée, alors que l'amplificateur non inverseur ne l'inverse pas.

Amplificateur inverseur

La **figure 2** montre le montage de base de l'amplificateur inverseur. Il ne nécessite que deux résistances externes. Pour les applications audio, on ajoute généralement une paire de condensateurs.
Quel est le gain de ce circuit ? L'entrée non inverseuse est reliée à la masse, de sorte que le gain en boucle ouverte est le paramètre initial qui détermine le gain :

$$V_0 = -\frac{U_{out}}{U_V} \quad \Rightarrow U_V = -\frac{U_{out}}{V_0}$$

L'équation suivante s'applique à la tension d'entrée :

$$U_{in} = U_{R1} + U_V = I_{R1} \cdot R_1 - \frac{U_{out}}{V_0}$$

Le gain est défini en divisant la tension de sortie par la tension d'entrée. Si vous utilisez le résultat obtenu ci-dessus dans cette formule, vous obtenez :

$$V = \frac{U_{out}}{U_{in}} = \frac{U_{out}}{I_{R1} \cdot R_1 - \frac{U_{out}}{V_0}}$$

La formule suivante s'applique aux courants d'entrée et de sortie du nœud en amont de l'entrée inverseuse :

$$I_{R1} = I_{R2} + I_{Ri} = I_{R2} + \frac{U_V}{R_i} = I_{R2} - \frac{U_{out}}{R_i \cdot V_0}$$

En utilisant cette valeur dans la formule de calcul du gain, on obtient :

$$V = \frac{U_{out}}{U_{in}} = \frac{U_{out}}{I_{R2} \cdot R_1 - \frac{U_{out} \cdot R_1}{R_i \cdot V_0} - \frac{U_{out}}{V_0}}$$

Le réseau comportant la résistance R_2 est régi par la formule suivante :

$$U_{out} = U_V - U_{R2} = -\frac{U_{out}}{V_0} - I_{R2} \cdot R_2$$

Pour calculer le courant I_{R2} :

$$I_{R2} = \frac{U_{out}}{V_0 \cdot R_2} - \frac{U_{out}}{R_2}$$

Enfin, si vous introduisez ce résultat dans la formule du gain et que vous annulez U_{out} partout où il apparaît, vous obtenez la formule exacte du gain de l'amplificateur inverseur :

$$V = \frac{1}{-\frac{R_1}{V_0 \cdot R_2} - \frac{R_1}{R_2} - \frac{R_1}{R_i \cdot V_0} - \frac{1}{V_0}}$$

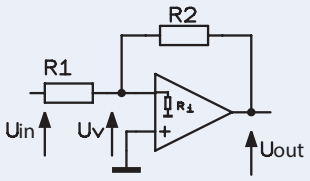


Figure 2. Amplificateur inverseur.

Les choses commencent à devenir intéressantes. Ici, vous pouvez raisonner comme suit : le gain en boucle ouverte est généralement beaucoup plus élevé que le rapport R_2/R_1, tandis que l'impédance interne de l'amplificateur est également beaucoup plus élevée que les valeurs des résistances externes. En tenant compte de cela, vous pouvez obtenir une formule simplifiée permettant de calculer le gain de façon approximative :

$$V \approx -\frac{R_2}{R_1}$$

Cette formule est très pratique pour calculer le gain de l'amplificateur opérationnel inverseur. Prenons un exemple de calcul pour voir la différence dans la pratique. En considérant un montage amplificateur avec R_2 = 100 kΩ et R_1 = 10 kΩ et en utilisant la formule simplifiée, nous obtiendrons un gain de -10. Selon la formule exacte dérivée précédemment (en supposant un gain en boucle ouverte de 100 000 et une impédance d'entrée de 1 MΩ), nous obtenons un gain d'environ -9,99889 (si cela vous intéresse, vous pouvez faire le calcul vous-même). Cela signifie que la différence est inférieure de plusieurs ordres de grandeur à la tolérance des résistances de précision, et qu'elle peut donc être facilement ignorée. (Par ailleurs, nous avons totalement ignoré plusieurs autres facteurs, notamment l'impédance de sortie, la dérive en température, etc.)

Amplificateur non inverseur

La **figure 3** montre le schéma de base de l'amplificateur non inverseur. De même, dans cette version simplifiée, vous n'avez besoin que de deux résistances externes.
La formule suivante s'applique à la tension de sortie d'un ampli-op :

$$U_{out} = (U_{in} - U_V) \cdot V_0 \quad \Rightarrow \quad (U_{in} - U_V) = \frac{U_{out}}{V_0}$$

Le gain du circuit peut être défini comme suit :

$$V = \frac{U_{out}}{U_{in}} = \frac{U_{out}}{(U_{in} - U_V) + U_V}$$

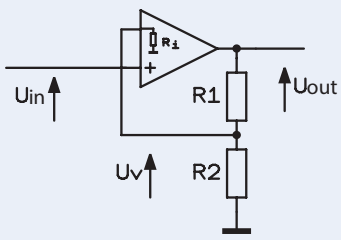


Figure 3. Amplificateur non inverseur.

En calculant la dérivée, nous comprendrons pourquoi nous avons inclus le terme U_V (apparemment dénué de sens), dans la formule ci-dessus. Si l'on inverse cette formule et que l'on utilise la tension de sortie de l'amplificateur, on obtient la formule suivante (notez que nous avons omis quelques étapes pour faciliter l'explication) :

$$\frac{1}{V} = \frac{U_{out}}{V_0 \cdot U_{out}} + \frac{U_V}{U_{out}} = \frac{1}{V_0} + \frac{U_V}{U_{out}}$$

Ici, R_1 et R_2 forment un diviseur de tension qui est chargé par la résistance R_i. Pour U_V, cela signifie

$$U_V = \frac{U_{out} \cdot (R_2 \| R_1)}{R_i + (R_2 \| R_1)} = \frac{U_{out}}{R_1 \cdot \left(\frac{1}{R_2} + \frac{1}{R_i}\right) + 1}$$

Si vous introduisez cette valeur dans la formule (inversée) du gain, vous obtenez :

$$\frac{1}{V} = \frac{1}{V_0} + \frac{1}{R_1 \cdot \left(\frac{1}{R_2} + \frac{1}{R_i}\right) + 1}$$

En inversant ce résultat, on obtient la formule exacte du gain :

$$V = \frac{1}{\frac{1}{V_0} + \frac{1}{R_1 \cdot \left(\frac{1}{R_2} + \frac{1}{R_i}\right) + 1}}$$

Dans ce cas également, il faut tenir compte que le gain en boucle ouverte est beaucoup plus élevé que le rapport R_2/R_1, et qu'en outre la valeur de Ri est beaucoup plus élevée que celle de R_2. Si vous appliquez ces considérations à la formule ci-dessus, vous arrivez finalement à la formule simplifiée du gain de l'ampli-op non inverseur :

$$V \approx \frac{R_1}{R_2} + 1$$

Il s'agit également d'une formule très pratique et réduite, et vous n'aurez souvent même pas besoin d'une calculatrice pour l'utiliser. Dans le prochain épisode, nous aborderons plus en détail la théorie des amplificateurs, puis nous nous pencherons enfin sur des montages amplificateurs pratiques.

230756-04

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

Questions ou commentaires ?
Contactez Elektor (redaction@elektor.fr).

Produits

- **Livre « *Basic Electronics for Beginners* », B. Kainka (Elektor, 2020)**
  Version papier : www.elektor.fr/19212
  Version numérique : www.elektor.fr/19213

Bibliothèques ESP recommandées

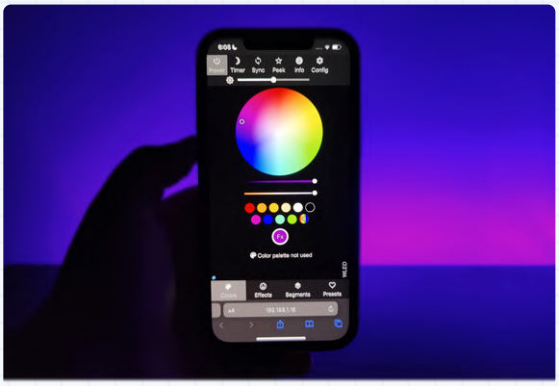
Saad Imtiaz et Jean-François Simon (Elektor)

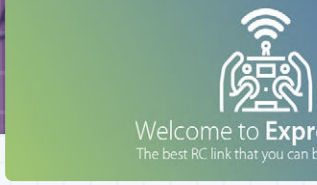
Vous cherchez des bibliothèques liées aux ESP ? Consultez ces recommandations de l'équipe d'ingénieurs d'Elektor.

WLED (github.com/Aircoookie)

WLED, de Christian Schwinne, est une implémentation rapide et riche en fonctionnalités d'un serveur web ESP8266/ESP32 pour contrôler les LED NeoPixels (WS2812B, WS2811, SK6812). Elle supporte également les puces SPI comme le WS2801, l'APA102 et bien d'autres. La bibliothèque propose plus de 100 effets spéciaux pour les NeoPixels, 50 palettes, des effets de bruit FastLED, et bien plus encore ! Elle dispose d'une interface utilisateur moderne avec des contrôles avancés. La configuration s'effectue via le réseau. Parmi les avantages notables, cette bibliothèque prend en charge jusqu'à 10 sorties LED par instance et peut fonctionner avec des bandes RGBW. Jusqu'à 250 préréglages utilisateur peuvent être utilisés pour sauvegarder et charger facilement des couleurs/effets, et permettent de les faire défiler. Les préréglages peuvent également être utilisés pour exécuter automatiquement les appels API. Il existe également une fonction « veilleuse », qui réduit progressivement l'intensité lumineuse des LED. En ce qui concerne les mises à jour, *Over the Air* (HTTP + ArduinoOTA) est pris en charge et peut être protégé par un mot de passe. Si vous cherchez à contrôler vos lumières RGB ou à donner un nouveau thème à votre pièce, vous allez adorer celle-ci ! Voir aussi Adafruit_NeoPixel ci-dessous.

https://github.com/Aircoookie/WLED





ExpressLRS (github.com/ExpressLRS)

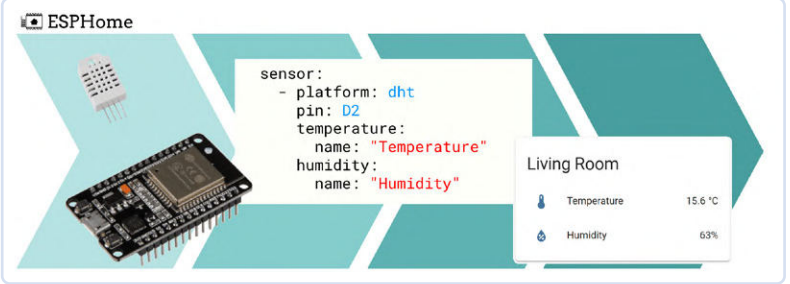
ExpressLRS est une liaison radio open-source pour les applications RC (radiocommande). Elle est conçue pour offrir d'excellentes performances en utilisant la puce LoRa SX127x/SX1280 combinée à un microcontrôleur Espressif ou STM32. ExpressLRS permet aux utilisateurs de bénéficier d'une bonne portée et d'une très faible latence, grâce à la modulation LoRa et à une taille de paquets réduite. ExpressLRS prend en charge le matériel de nombreux fabricants : AxisFlying, BETAFPV, Flywoo, FrSky, HappyModel, HiYounger, HGLRC, ImmersionRC, iFlight, JHEMCU, Jumper, Matek, NamimnoRC, QuadKopters et SIYI. Il offre un débit de paquets de 1 kHz, la télémétrie, des mises à jour WiFi, deux fréquences pour la liaison RC (2,4 GHz ou 900 MHz) et bien d'autres choses encore. C'est sans aucun doute très intéressant pour de nombreux projets RC, avec différents matériels adaptés pour différents besoins.

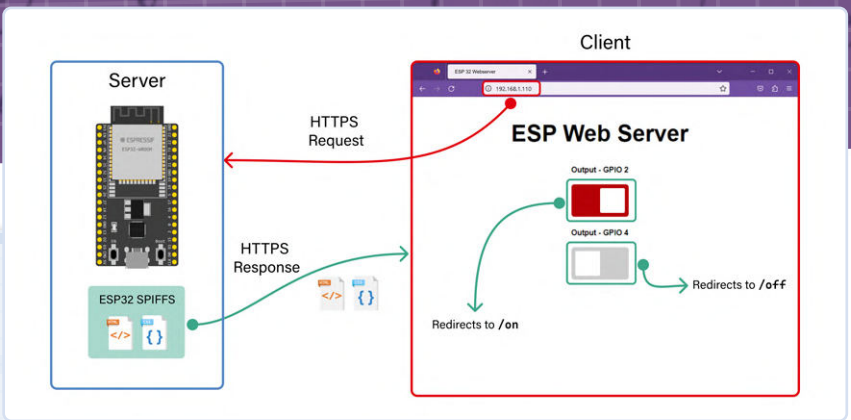
https://github.com/ExpressLRS/ExpressLRS

ESPHome (github.com/esphome)

ESPHome est un système de développement open-source qui simplifie le processus de configuration et de gestion des appareils basés sur ESP8266 et ESP32. Il permet aux utilisateurs de créer des microprogrammes personnalisés pour ces appareils sans exiger de programmation avancée. Avec ESPHome, vous pouvez définir les fonctionnalités et les paramètres des appareils à l'aide d'un fichier de configuration YAML facile à utiliser, ce qui le rend accessible aux débutants comme aux développeurs expérimentés. Les principales caractéristiques d'ESPHome sont la prise en charge d'un large éventail de capteurs et de composants, la détection et l'intégration automatique avec les plateformes domotiques les plus répandues comme Home Assistant, et les mises à jour OTA pour des mises à niveau transparentes du micrologiciel. Il favorise la création de solutions domestiques intelligentes, permettant aux utilisateurs d'adapter leurs appareils à des besoins spécifiques, tels que la surveillance de la température, le contrôle de l'éclairage ou la sécurité à domicile. ESPHome a gagné en popularité dans la communauté de la maison intelligente. Ne manquez pas de le découvrir, si vous ne le connaissez pas déjà !

https://github.com/esphome/esphome



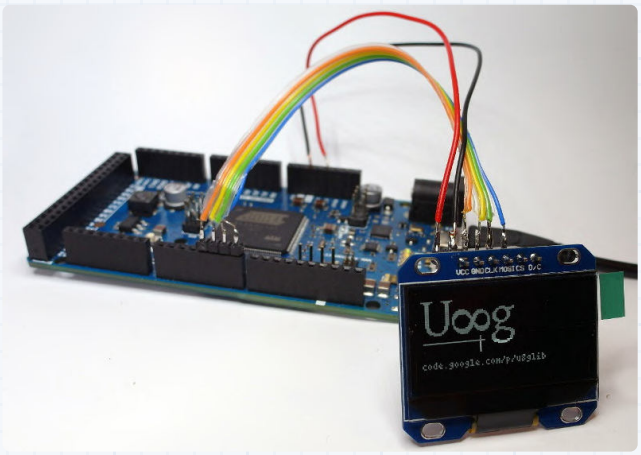


ESPAsyncWebServer (github.com/me-no-dev)

ESPAsyncWebServer est un serveur HTTP et WebSocket asynchrone pour ESP8266 utilisé dans l'environnement Arduino. Il peut être utilisé avec PlatformIO et l'EDI Arduino. L'utilisation d'un réseau asynchrone signifie que vous pouvez gérer plus d'une connexion en même temps. Un appel (*call*) est émis dès que la requête est prête et analysée. Pour envoyer la réponse, vous êtes immédiatement prêt à gérer d'autres connexions pendant que le serveur se charge d'envoyer la réponse en arrière-plan. La vitesse de traitement est très élevée ! ESPAsyncWebserver fournit une API facile à utiliser et est compatible avec les authentifications HTTP Basic et Digest MD5 (par défaut), ainsi qu'avec Chunked Response. Il existe plusieurs plugins offrant divers avantages, tels que : différents emplacements, envoi d'événements au navigateur, réécriture d'URL avancée, etc…
https://github.com/me-no-dev/ESPAsyncWebServer

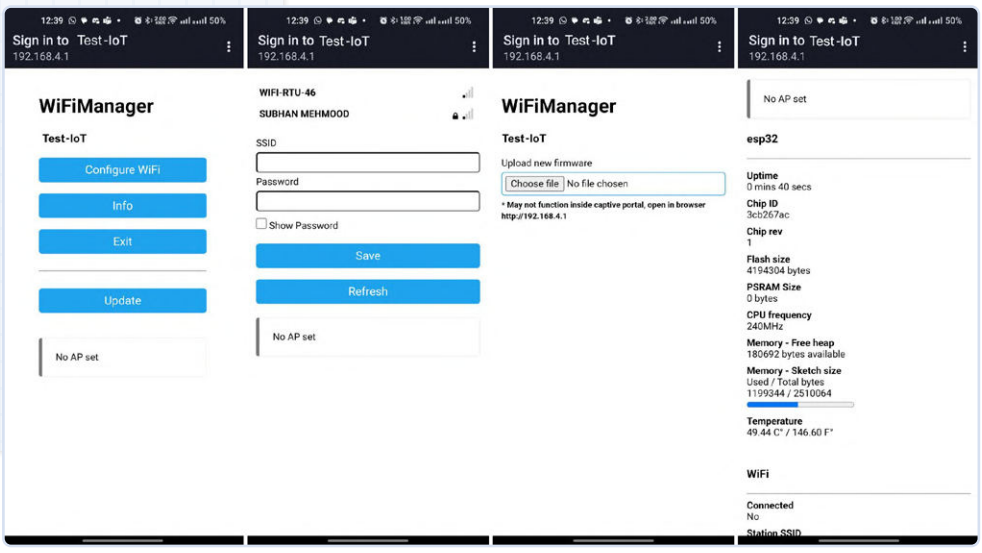
U8glib (github.com/olikraus)

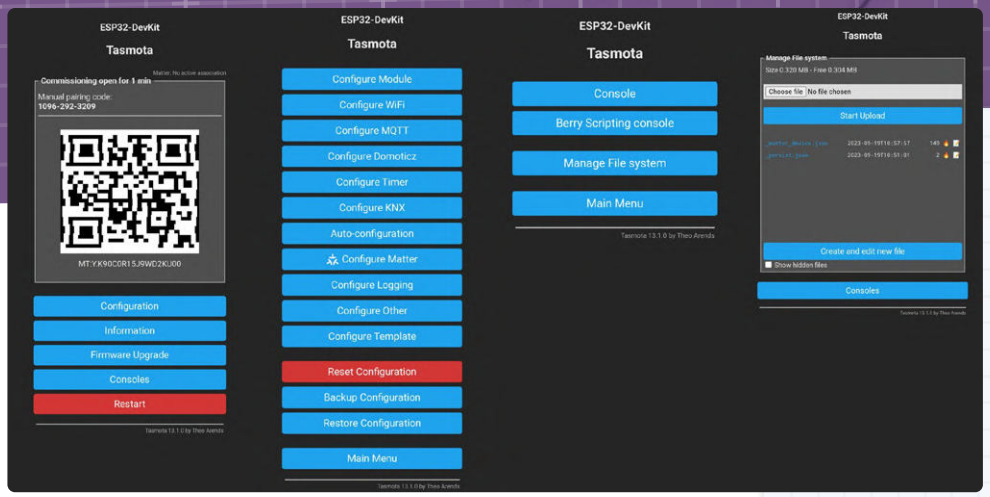
U8glib est une bibliothèque graphique qui prend en charge de nombreux écrans monochromes. La dernière version de *U8glib* pour Arduino est disponible dans le gestionnaire de bibliothèques et peut également être téléchargée depuis GitHub. Elle est compatible avec Arduino (ATmega et ARM), AVR, ARM (avec exemple pour LPC1114) et beaucoup d'autres modules tels que SSD1325, ST7565, ST7920, UC1608, UC1610, UC1701, PCD8544, PCF8812, KS0108 et plus encore. Cette bibliothèque prend en charge de nombreuses polices (monospaces et proportionnelles), le mode curseur de la souris, le mode paysage et portrait… En résumé, une bibliothèque très complète qui sera utile pour de nombreux projets
https://github.com/olikraus/u8glib



WifiManager (github.com/tzapu)

WiFiManager est un gestionnaire de connexion pour les projets basés sur ESP. Il permet à l'utilisateur de configurer les identifiants WiFi et les paramètres personnalisés au moment de l'exécution, grâce à un portail captif. Comment cela fonctionne-t-il ? Lorsque votre ESP démarre, il se met en mode *Station* et tente de se connecter à un point d'accès précédemment enregistré. En cas d'échec, le micrologiciel fait passer l'ESP en mode *Point d'accès* et lance un DNS et un serveur Web. Ensuite, à l'aide de n'importe quel appareil WiFi doté d'un navigateur (ordinateur, téléphone, tablette), vous pouvez vous connecter au point d'accès nouvellement créé et configurer les informations d'identification. L'ESP se connectera alors au réseau de votre choix !
Il existe également des options permettant de changer cette manière de faire, ou de lancer manuellement le portail de configuration ainsi que le portail web indépendamment l'un de l'autre. Il est également possible de les faire fonctionner en mode *non bloquant*.
https://github.com/tzapu/WiFiManager



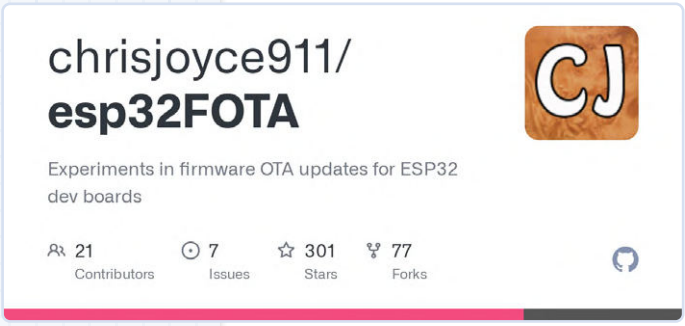


Tasmota (github.com/arendst)

Tasmota est un micrologiciel open-source conçu pour les appareils ESP, spécifiquement adaptés à la domotique et aux maisons intelligentes. Il prend en charge les microcontrôleurs ESP8266, ESP32, ESP32-S ou ESP32-C3. Theo Arends a initié ce projet en 2016 sous le nom de *Sonoff-MQTT-OTA*. Son objectif principal était d'équiper les appareils basés sur l'ESP8266 produits par l'ITEAD (Sonoff) en les rendant compatibles avec MQTT et les mises à jour *over-the-air* (OTA). Cela avait commencé comme une solution simple, pour modifier un Sonoff Basic (qui, à l'origine, requiert l'usage du cloud) en un appareil gérable localement. Le projet a fini par devenir un écosystème complet adapté à presque tous les apareils basés sur l'ESP. Tasmota est développé pour PlatformIO, ce qui facilite la configuration par le biais d'une interface web (webUI). Les mises à jour peuvent être effectuées par WIFI. Les utilisateurs peuvent automatiser les appareils à l'aide de minuteries ou de règles personnalisées. Le contrôle local complet et l'extensibilité sont possibles grâce aux protocoles MQTT, HTTP, série ou KNX
https://github.com/arendst/Tasmota

Esp32FOTA (github.com/chrisjoyce911)

esp32FOTA est une bibliothèque simple pour ajouter la compatibilité avec les mises à jour *over-the-air* (OTA) à votre projet ESP32. Cette bibliothèque va essayer d'accéder à un fichier JSON hébergé sur un serveur web, analyser son contenu pour déterminer si une nouvelle version du firmware est disponible, et si c'est le cas, la télécharger et l'installer. Pour que ce processus de mise à jour fonctionne, vous avez besoin d'un serveur web avec un fichier JSON valide (compressé en option avec zlib ou gzip). La liste complète des conditions requises (y compris les détails concernant HTTPS) est disponible sur GitHub. Cette bibliothèque est assez complète et inclut la prise en charge des firmwares compressés, des mises à jour de partitions SPIFFS/LittleFS, la compatibilité avec divers systèmes de fichiers pour le stockage des certificats et des signatures. En outre, esp32FOTA permet également des mises à jour basées sur le web (avec un serveur web), la synchronisation des firmwares par lots et la possibilité de forcer les mises à jour. Enfin, la bibliothèque dispose également de fonctionnalités avancées telles que la vérification des signatures pour les images de firmwares téléchargées, la vérification des signatures et la prise en charge du versionnage sémantique. En résumé, c'est une bibliothèque qui mérite d'être découverte si vous souhaitez utiliser les mises à jour OTA pour votre prochain projet.
https://github.com/chrisjoyce911/esp32FOTA



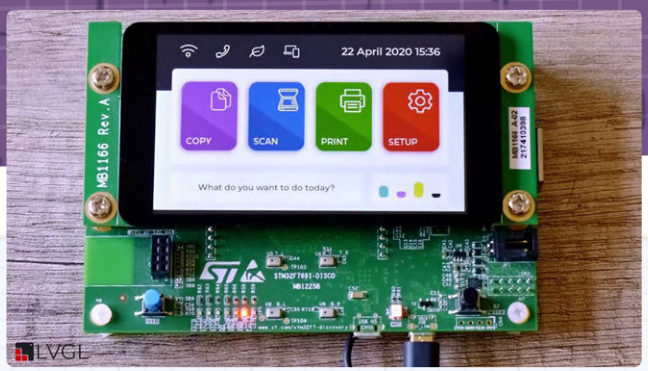
Willow (github.com/toverainc)

Willow est un un système de développement IdO d'Espressif (*Espressif IDF, IoT Development Framework*) utilisé pour transformer une ESP32-S3-BOX en un assistant vocal polyvalent doté de fonctionnalités diverses. Il peut être déclenché par des mots clés personnalisés (*wake words*) tels que « Hi ESP » ou « Alexa » et écoute les commandes vocales, détectant automatiquement quand commencer et arrêter d'écouter. Willow s'intègre bien aux plateformes domotiques les plus courantes, telles que *Home Assistant*, *openHAB* et les *API REST* génériques. Cet assistant vocal fonctionne parfaitement dans des environnements difficiles, reconnaissant les *wake words* et la parole à des distances d'environ sept m. Il garantit un son de la meilleure qualité possible grâce à des fonctions telles que le contrôle automatique du gain, l'annulation de l'écho, la réduction du bruit et la séparation des sources. Dans les environnements WiFi encombrés, Willow peut utiliser la compression audio pour optimiser l'utilisation du temps de transmission. En outre, Willow offre une reconnaissance vocale sur l'appareil, ce qui vous permet de configurer jusqu'à 400 commandes localement. Vous pouvez également utiliser le serveur d'inférence open-source Willow pour obtenir des capacités de transcription vocale plus étendues. En conclusion, c'est un projet très impressionnant !
https://github.com/toverainc/willow

Adafruit_NeoPixel (github.com/adafruit)

Il s'agit d'une bibliothèque Arduino pour contrôler les NeoPixels. Ce sont, dans le jargon d'Adafruit, des pixels et des bandes de LED RVB adressables individuellement, basés sur les WS2812, WS2811 et SK6812. Ces LED ont chacune un driver intégré et utilisent un protocole de contrôle à un seul fil. Elles ont tendance à être un peu difficiles à utiliser car les exigences de synchronisation sont strictes et à cause du protocole spécifique. Avec cette bibliothèque, tout devient plus faciles ! Adafruit la développe et la maintient gratuitement. En retour, les utilisateurs peuvent décider d'acheter certains de leurs produits. Les principaux objectifs d'Adafruit_NeoPixel sont d'être facile à utiliser et d'être flexible en termes de chipsets supportés. La bibliothèque supporte les AVR (ATmega et ATtiny), Teensy, Arduino Due, Arduino 101, ATSAMD21/51, Adafruit STM32 Feather, ESP8266, ESP32, Nordic nRF51/52 ainsi que certains ICs de la série XMC d'Infineon. Entre autres, les nombreuses fonctions bien connues telles que `setBrightness()`, `setPixelColor()` et 12 autres rendent cette bibliothèque très utile pour un prototypage rapide.

https://github.com/adafruit/Adafruit_NeoPixel



LVGL (github.com/lvgl)

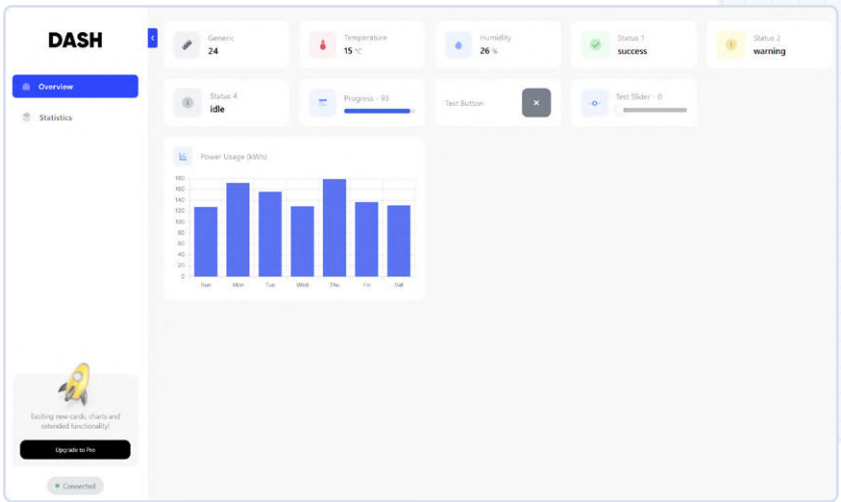
Toujours sur le sujet des tableaux de bord et des graphiques, LVGL (**figure 12)** est une bibliothèque graphique embarquée, permettant la création d'interfaces graphiques pour une large gamme de microcontrôleurs, y compris, bien sûr, ESP32 et Arduino (la liste complète est sur GitHub). Cette bibliothèque est écrite en C et est portable, c'est-à-dire qu'elle ne dépend d'aucune dépendance externe. Elle peut être utilisée avec ou sans système d'exploitation en temps réel (RTOS) et supporte un grand nombre d'afficheurs : monochrome, ePaper, OLED, etc. Elle nécessite 32 kB de RAM et 128 kB de mémoire Flash. LVGL dispose d'un large éventail de widgets intégrés, de styles, de présentations, et plus encore ; elle est très personnalisable. Animations, anticrénelage, contrôle de l'opacité, défilement fluide, ombres, transformation d'images… La liste est longue. Diverses méthodes de saisie comme la souris, le pavé tactile et le clavier sont prises en charge. Make et CMake sont tous deux supportés, ce qui vous permet de développer sur un PC et d'utiliser le même code d'interface utilisateur sur du matériel embarqué, ce qui peut être une fonctionnalité intéressante pour certains utilisateurs.

https://github.com/lvgl/lvgl

ESP-DASH (github.com/ayushsharma82)

ESP-DASH est une bibliothèque à haute vitesse conçue pour développer un tableau de bord fonctionnel et en temps réel adapté aux microcontrôleurs ESP8266 et ESP32. Elle est développée par l'utilisateur GitHub ayushsharma82 et d'autres contributeurs, et en est actuellement à la version 4. Cette bibliothèque comporte de nombreuses fonctionnalités, notamment des graphiques, des cartes d'affichage, des boutons interactifs et de nombreux autres composants, permettant de créer de beaux tableaux de bord. Ceux-ci seront accessibles localement et ne nécessiteront pas de connexion internet, toutes les données étant stockées sur la puce. ESP-DASH propose de générer automatiquement des pages web et de les mettre à jour en temps réel pour tous les clients connectés. Les utilisateurs n'ont pas besoin de se plonger dans le HTML, le CSS ou le JavaScript, car elle offre une interface C++ facile à utiliser. Elle fournit des composants préconfigurés pour la gestion de vos données et offre une grande souplesse, car vous pouvez ajouter ou supprimer des composants sans effort, directement à partir de la page web. De plus, elle est dotée d'une prise en charge intégrée des graphiques, ce qui améliore sa fonctionnalité.

https://github.com/ayushsharma82/ESP-DASH



VF : Laurent Rauber — 230588-04

composants piézoélectriques

drôle de composant

David Ashton (Australie)

L'effet piézoélectrique est connu depuis les années 1880, mais la façon dont il est passé d'une curiosité scientifique en une notion importante de l'électronique moderne est une histoire d'innovation et de découverte. Cette propriété unique de certains cristaux a conduit à leur utilisation généralisée dans une grande variété d'applications, des transducteurs audio aux instruments de précision en astronomie et en imagerie numérique.

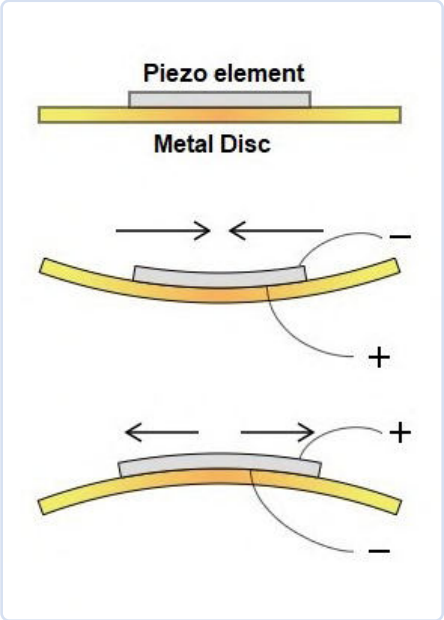


Figure 1. Effet piézoélectrique. (Source : Sonitron Support - CC BY-SA 3.0, commons.wikimedia.org/w/index.php?curid=115322872)

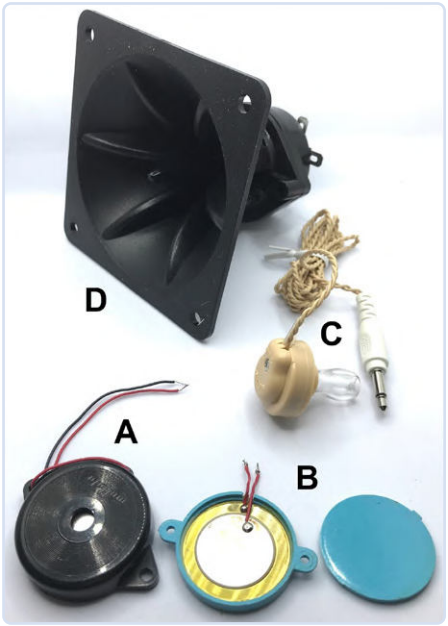


Figure 2. Une sélection de générateurs de signaux piézoélectriques. Il s'agit simplement d'éléments piézoélectriques commandés par une tension alternative. A : un buzzer piézo ordinaire. B : Idem, mais ouvert pour montrer la structure. C : un écouteur à cristal. D : un tweeter à pavillon piézo de 100 W pour les applications hi-fi.

Lorsqu'on dépose des électrodes sur certains matériaux céramiques cristallins, sur deux faces opposées du cristal, et qu'on les soumet à une tension, le cristal se déforme (légèrement). Cet effet a été démontré pour la première fois par les frères Pierre et Jacques Curie en 1880, bien qu'il ait déjà été prédit mathématiquement par Gabrielle Lippmann en 1861. Au cours des décennies suivantes, il n'a toutefois été considéré que comme un phénomène curieux dans les laboratoires.

Cet effet a de nombreuses applications en électronique. Les transducteurs acoustiques sont probablement les mieux connus par les passionnés d'électronique. Je me souviens d'avoir fabriqué, il y a plus de 50 ans, mon premier récepteur à cristal à l'aide d'un écouteur à cristal - qui est simplement un transducteur piézoélectrique monté sur un disque métallique. La **figure 1** montre le principe de fonctionnement. Ils ont l'avantage d'avoir une impédance assez élevée, ce qui est important pour un récepteur à cristal, car la charge ne sera pas élevée. Ils sont également très fins, ce qui explique leurs utilisations dans

Figure 3. Divers buzzers piézo-électriques (beepers). Ils sont équipés d'un système électronique permettant de contrôler l'élément. À l'arrière : une sirène piézoélectrique de 12 V. Au centre : trois buzzers piézo différents. Avant : un buzzer piézoélectrique CA/CC destiné à être utilisé dans les tableaux de distribution industriels (adapté à plusieurs tensions).

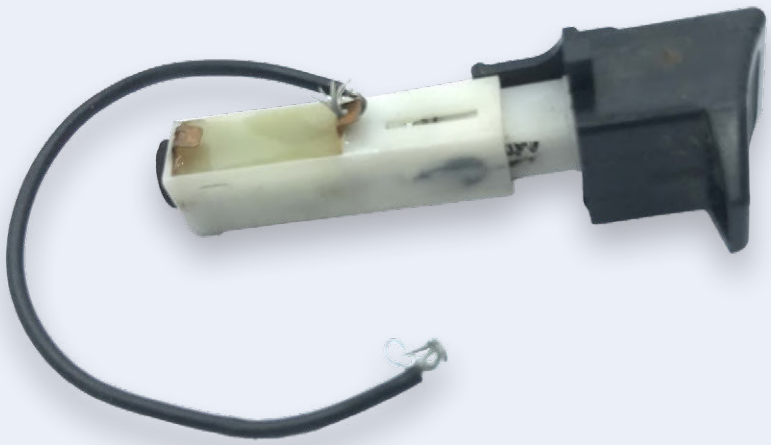


Figure 4. Allumeur piézoélectrique provenant d'un briquet à gaz. En appuyant sur le bouton situé à l'extrémité, il émet un « clic » et génère une impulsion haute tension qui produit une étincelle entre deux électrodes.

les cartes de vœux qui jouent une mélodie lorsque vous les ouvrez. Il existe des transducteurs plus grands avec une réponse en fréquence assez acceptable ; certains tweeters Hi-fi fonctionnent avec la technologie piézo (**figure 2**). La plupart de ces composants ont une fréquence de résonance à laquelle ils sont plus efficaces, et cette fréquence est utilisée dans les sonneurs, qui intègrent un oscillateur qui fait vibrer le disque à la fréquence de résonance ou à une fréquence voisine. Ils sont utilisés comme « beepers » et sonneries d'alarme (**figure 3**) pour produire des signaux d'avertissement. Ils ont l'avantage d'être très efficaces et de nécessiter très peu d'énergie pour le son qu'ils produisent.

Parmi les autres applications des transducteurs piézoélectriques figurent les imprimantes à jet d'encre (un petit mouvement est utilisé pour « jeter » une goutte d'encre) et les télescopes astronomiques - les actionneurs piézoélectriques sont utilisés pour apporter de très petites corrections à la forme du lourd miroir utilisé pour collecter la lumière, afin de minimiser les distorsions. Ils peuvent appliquer des forces relativement importantes sur de très petites distances. Une autre application est la stabilisation de l'image dans les appareils photo numériques, où des mouvements légers mais précis avec un temps de réponse rapide sont nécessaires.

L'effet est réversible, c'est-à-dire que lorsqu'un élément piézo est plié ou déformé, il génère une tension au niveau des électrodes. Cette technique est utilisée dans les allumeurs de gaz, où un élément est lentement déformé puis reprend rapidement sa forme initiale par une sorte de mécanisme à cliquet (**figure 4**). Au cours de ce processus, une impulsion à très haute tension est générée - suffisamment pour produire une étincelle à travers deux électrodes utilisées pour enflammer le gaz. Ils sont également utilisés dans les capteurs de vibrations, où leur petite taille et leur faible coût constituent un grand avantage.

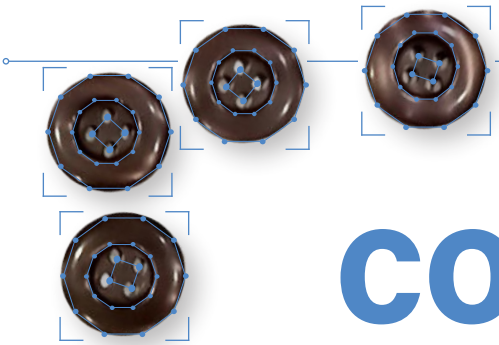
230684-04

Questions ou commentaires ?

Contactez Elektor (redaction@elektor.fr).

À propos de l'auteur

David Ashton est né à Londres, a grandi en Rhodésie (aujourd'hui Zimbabwe), a vécu et travaillé au Zimbabwe et vit aujourd'hui en Australie. Il s'intéresse à l'électronique depuis qu'il est haut comme trois pommes. La Rhodésie n'étant pas le centre du monde de l'électronique, l'adaptation, la substitution et la recherche de composants ont été des compétences qu'il a acquises très tôt (et dont il est toujours fier). Il a dirigé un labo d'électronique, mais a essentiellement travaillé dans le domaine des télécommunications.



compteur d'objets intelligent

reconnaissance d'images simplifiée avec Edge Impulse

Somnath Bera (Inde)

Découvrez comment vous pouvez transformer un Raspberry Pi et une caméra en un outil intelligent de comptage d'objets à l'aide de la plateforme Edge Impulse ! Ce projet amusant et accessible démontre la facilité de débuter avec Edge Impulse sur un Raspberry Pi, parfait pour les débutants comme pour les passionnés plus expérimentés.

Edge Impulse est spécialisé dans la fourniture d'outils et de plateformes pour le développement de modèles d'apprentissage automatique pour l'*edge computing*, en particulier pour les systèmes embarqués. L'*edge computing* implique le traitement des données à proximité de la source de données, au lieu de dépendre d'un serveur distant. C'est parfait pour une implantation sur un Raspberry Pi ! Dans cet exemple, nous allons compter les petits objets, dans le cas présent les boutons ordinaires que l'on trouve sur les vêtements. Les plateformes d'apprentissage automatique telles que Edge Impulse utilisent ce que l'on appelle des modèles, qui sont des types spécifiques d'algorithmes utilisés pour l'analyse des données et la reconnaissance des formes. Ces modèles sont entraînés à reconnaître des formes, à faire des prédictions ou à effectuer des tâches basées sur les données d'entrée.

Les modèles Edge Impulse permettent de classer facilement les objets. Vous pouvez faire la distinction entre un homme et un animal, un vélo et une voiture, etc. En outre, vous pouvez facilement compter une sorte d'objet parmi d'autres types d'objets. Tout ce dont vous avez besoin, c'est d'un module caméra de bonne qualité, d'une lumière suffisante, d'une mise au point correcte, et enfin, d'un ordinateur de taille moyenne (un Raspberry Pi 3 ou Raspberry Pi 4 est suffisant), et vous êtes prêt à compter.

Dès le départ, ce projet a été destiné à une installation au niveau microcontrôleur (MCU), comme un Espressif ESP32, un Arduino Nicla vision, etc. C'est pourquoi il a été conçu pour une très petite zone de comptage (120 × 120 pixels) avec un bouton relativement petit en tant qu'objet à compter. En fin de compte, il s'est avéré que même pour une si petite zone, les microcontrôleurs n'étaient pas du tout adaptés. Les modèles d'apprentissage automatique sont pré-entraînés sur les serveurs Edge Impulse et un fichier modèle est généré pour être stocké sur l'appareil embarqué. Ici, le fichier modèle lui-même pèse environ 8 Mo ! Par conséquent, le projet a finalement été installé sur un ordinateur Raspberry Pi, où il fonctionne parfaitement.

Connaissance et sagesse

Si vous connaissez Edge Impulse [1], alors croyez-moi, votre travail est à moitié terminé. Pour le reste, il vous suffit de peaufiner votre modèle afin d'obtenir un niveau de performance acceptable. Un modèle d'IA fonctionne comme un enfant. Imaginez comment vous avez appris des choses comme « A est comme Abeille (A comme *apple*) » et « B comme Balle ». On vous a montré une pomme sous différents angles, puis on vous a appris à la nommer « pomme ». Il en va de même pour la « balle ». À présent, un enfant identifiera assez facilement une pomme et une balle sous tous les angles possibles ! Il en va de même pour l'IA, qui peut les identifier facilement.

Considérons maintenant qu'il y a un panier dans lequel sont mélangées des balles de la taille d'une pomme et des pommes de la taille d'une balle, et qu'elles se ressemblent de là où on les regarde. En tant qu'enfant, que feriez-vous ? Avec votre seule connaissance de la pomme et de la balle, vous vous tromperiez tout simplement ! L'IA passerait à côté également. Mais considérons que la corbeille de fruits est exposée par un vendeur de fruits et légumes. Selon toute probabilité, aucun de ces fruits n'est une balle ! Et certains d'entre eux, voire tous, pourraient être des pommes. Cette « astuce » consistant à associer une pomme à un vendeur de fruits et légumes s'appelle la sagesse, que l'on ne peut attendre ni d'un enfant ni d'une IA, à moins qu'on ne lui enseigne spécifiquement le contraire. Cependant, les êtres humains ont appris, au fil des ans, de nombreuses autres choses associées qui nous ont finalement donné assez de sagesse pour relier une pomme à un vendeur de fruits et légumes.

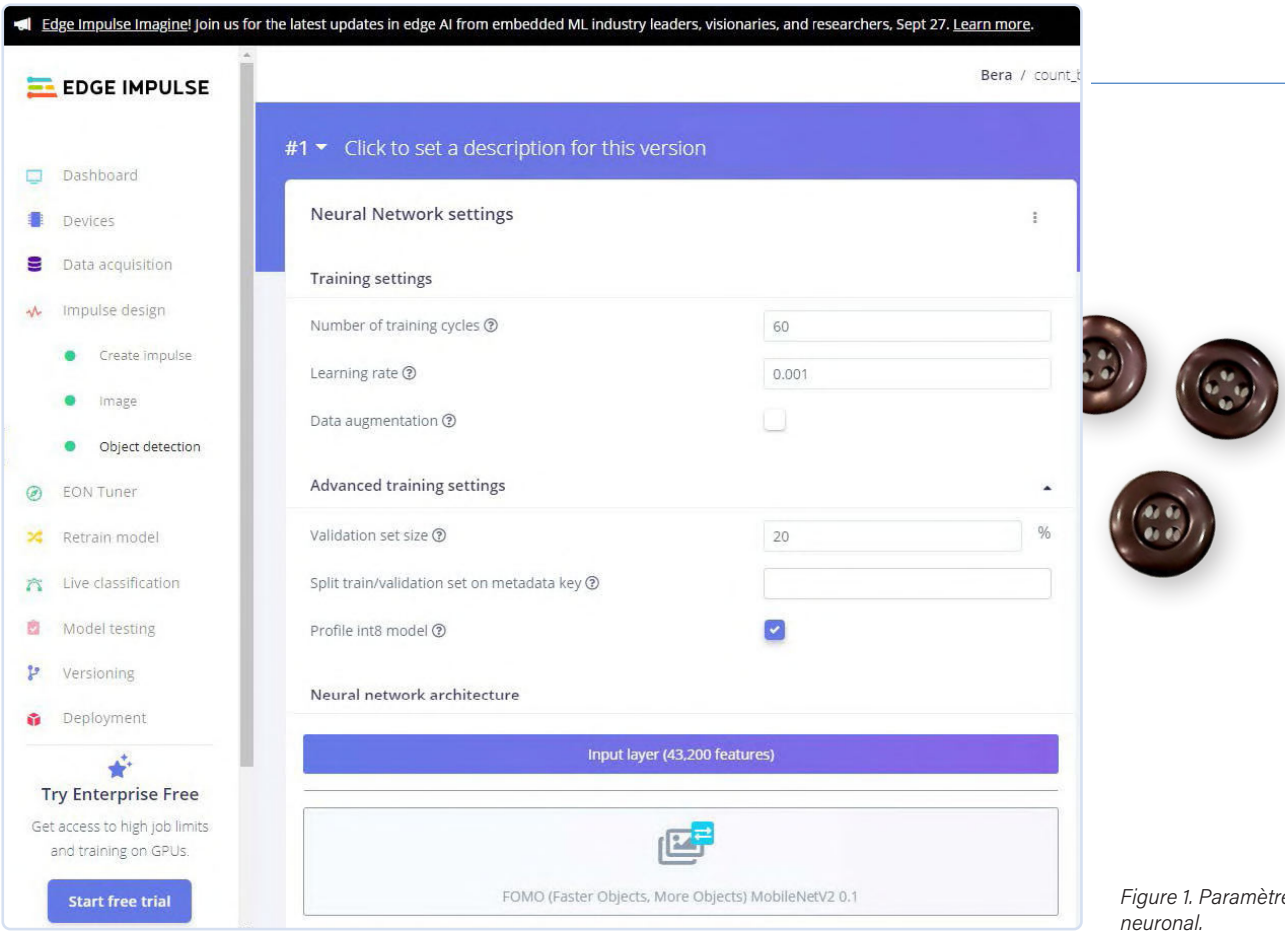


Figure 1. Paramètres du réseau neuronal.

Pourtant, l'IA progresse si rapidement qu'un jour, elle pourra travailler sur la sagesse. Pour l'instant, vous devez apprendre la pomme et la balle au modèle ML sous tous les angles possibles afin de les exécuter sans aucune confusion (par exemple, le profil de texture d'une pomme, sa tige, ses plis sur son corps, l'aspect du haut et du bas et bien d'autres choses encore). Quoi qu'il en soit, de nombreux modèles aux capacités différentes sont disponibles dans Edge Impulse pour tester et expérimenter.

Démarrer avec Edge Impulse

Tout d'abord, ouvrez un compte dans Edge Impulse [1], qui nécessite un identifiant par email. Rassemblez quelques boutons du même type. Si vous ouvrez le site à partir d'un ordinateur Raspberry Pi, en utilisant la caméra de l'ordinateur Raspberry Pi (soit connectée en USB, soit connectée au port Cam), vous pouvez collecter des images de boutons sous plusieurs angles (ce qui est nécessaire lorsque le modèle est déployé dans un environnement de travail réel). Edge Impulse permet également de connecter votre téléphone portable ou votre ordinateur portable en tant que périphérique d'entrée pour la collecte de données, ce qui est également plus pratique pour l'acquisition de données dans le cadre du projet Edge Impulse.

Le projet

Le projet Edge Impulse est divisé en plusieurs étapes, qui doivent toutes être suivies sur le site web Edge Impulse.

1. L'acquisition de données : il peut s'agir d'images, de sons, de températures, de distances, etc. Une partie des données est utilisée comme données de test, tandis que toutes les autres données sont utilisées comme données d'apprentissage.
2. Impulse design : l'élément central est nommé *Create Impulse*. Dans ce contexte, une « impulsion » fait référence à un pipeline ou à un workflow pour la création d'un modèle d'apprentissage automatique. Cette impulsion comprend plusieurs étapes, notamment l'ajustement des paramètres d'entrée associés aux données qui viennent d'être collectées, le traitement du signal, l'extraction des caractéristiques et le modèle d'apprentissage automatique lui-même. Les « caractéristiques » sont les propriétés ou les particularités individuelles mesurables d'un phénomène observé. En principe, les caractéristiques sont les attributs de données utilisés par les modèles pour détecter des modèles et prendre des décisions. Le pipeline d'Impulse est subdivisé comme suit :

   - Paramètres d'entrée (*Input parameters*) : image (largeur, hauteur), son (paramètres du son)
   - Bloc de traitement (*Processing block*) : comment traiter les données d'entrée
   - Bloc d'apprentissage (*Learning block*) : données de l'objet de ce mode d'apprentissage

   Vous devez sélectionner et configurer ces trois étapes.

3. Traitement des images (*Image processing*) : générer des caractéristiques à partir des images collectées.
4. Détection d'objets (Object detection) : sélectionnez votre modèle de réseau neuronal et entraînez-le.

Pour la partie finale – la détection d'objets – votre expertise est nécessaire, ou je dirais plutôt un effort d'expérimentation, afin que la précision du modèle atteigne 85 % ou plus. À un moment donné, vous devez supprimer certaines mauvaises images (dites aberrantes) du modèle afin d'améliorer son efficacité.

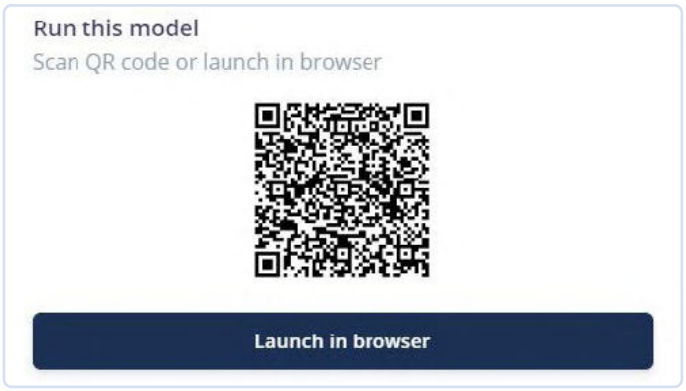


Figure 2. Scannez le code QR pour lancer ce modèle.

Il existe quelques modèles dans lesquels vous pouvez essayer de voir le niveau de précision du modèle. Tout ce qui est supérieur à 90 % est excellent, mais il ne sera certainement pas précis à 100 % ! Si c'est le cas, c'est que quelque chose ne va pas avec vos données. Il se peut qu'il y ait très peu de données ou que les caractéristiques soient insuffisantes. Revérifiez et réessayez dans ce cas ! Pour ce projet, la précision était de 98,6 %. Certes, notre nombre de données (environ 40) était faible. Cependant, pour un projet de débutant, c'est plutôt bien (voir la **figure 1**). Les fichiers de ce projet sont disponibles sur la page *Elektor Labs* de ce projet [4].

Test du modèle

Vous pouvez d'abord tester votre modèle sur les données d'essai. Commencez par là, puis dirigez votre système vers les données réelles et voyez si cela fonctionne !

Dans le tableau de bord de la page d'ouverture de Edge Impulse, la fonction de test est disponible. Vous pouvez directement exécuter le modèle dans le navigateur ou utiliser votre smartphone pour le tester. Pour cela, Edge Impulse propose un code QR à scanner avec votre smartphone **figure 2)**. Dirigez l'appareil photo vers les boutons (**figure 3**, **4** et **5**) et voyez s'il peut les compter ou non !

Déploiement sur un Raspberry Pi

Pour exécuter le modèle sur un ordinateur Raspberry Pi, vous devez télécharger le fichier **.eim*. Mais contrairement à d'autres matériels (Arduino, Nicla Vision, ou ESP32 où vous pouvez télécharger directement), dans le cas du Raspberry Pi, vous devez d'abord installer Edge Impulse sur l'ordinateur Raspberry Pi. À l'intérieur du logiciel *edge-impulse-daemon*, vous devez télécharger ce fichier. Mais ne vous inquiétez pas, Edge Impulse a consacré une page entière à l'installation de Edge Impulse sur Raspberry Pi. Il y a quelques dépendances à installer en premier. Regardez [2]. C'est assez simple. La marche à suivre est bien détaillée.

Donc après avoir installé Edge Impulse sur l'ordinateur Raspberry Pi, les choses sérieuses commencent. N'oubliez pas de garder le Raspberry Pi connecté à Internet.

Exécuter la commande `edge-impulse-linux-runner` à partir du terminal du Raspberry Pi. Cela lancera un assistant qui vous demandera de vous connecter et de choisir un projet Edge Impulse. Si vous souhaitez passer d'un projet à l'autre par la suite, exécutez à nouveau cette commande avec l'option `--clean`. Cette commande compilera et téléchargera automatiquement le modèle d'IA de votre projet et l'exécutera sur votre Raspberry Pi. Montrez les boutons à la caméra connectée à votre Raspberry Pi et elle devrait les compter. C'est bien ! Dans ce qui suit, nous allons modifier le système en utilisant Python et un synthétiseur vocal qui, après avoir compté, énonce le nombre de boutons qu'il a réussi à compter.

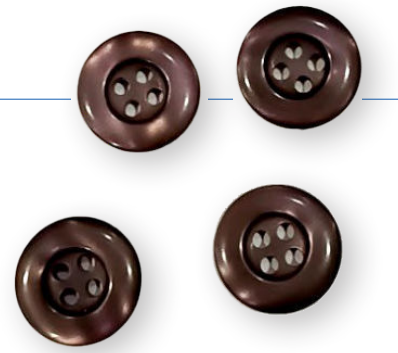


Figure 3. Acquisition des données : Échantillon 1 (Sample-1).

Figure 4. Acquisition des données : Échantillon-2.

Figure 5. Acquisition des données : Échantillon-3.

Déploiement du modèle en Python

Dans le déploiement ci-dessus, il fonctionnerait comme prévu dans le modèle Edge Impulse. Pour le faire fonctionner dans un but particulier, par exemple pour déclencher une alarme sonore ou allumer une LED lorsque le comptage atteint « 2 ou plus » , vous devez trouver un autre moyen ! C'est ici que Python 3 est là pour vous aider. Le *Linux-sdk-python* doit être installé sur votre ordinateur Raspberry Pi.

Le kit de développement logiciel (SDK) Edge Impulse est disponible pour de nombreux modèles, notamment Python, Node.js, C++, etc. Consultez la page SDK Python [3].

Une fois que *linux-sdk-python* est installé, allez dans le répertoire *linux-sdk-python/examples/image* directory et exécutez le fichier Python pour l'identification de l'image. Ne vous y trompez pas. Le répertoire des exemples contient trois sous-répertoires, un pour les données audio, un pour les données d'image et un pour les données personnalisées. Dans le répertoire image, le fichier de classification vidéo est également disponible pour les données d'entrée vidéo. Le répertoire custom permet de personnaliser d'autres types de données (pour les experts uniquement !).

Ensuite exécutez la commande :

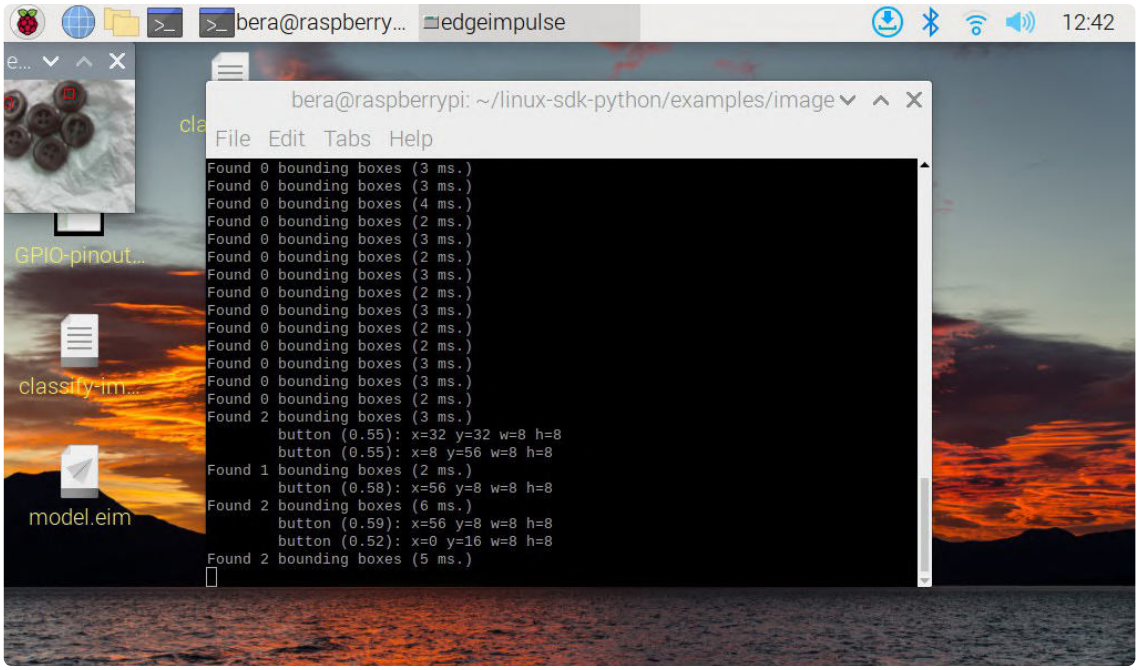


Figure 6. Quatre boutons ne sont pas comptés en raison d'une mise au point et d'une luminosité inadaptées.

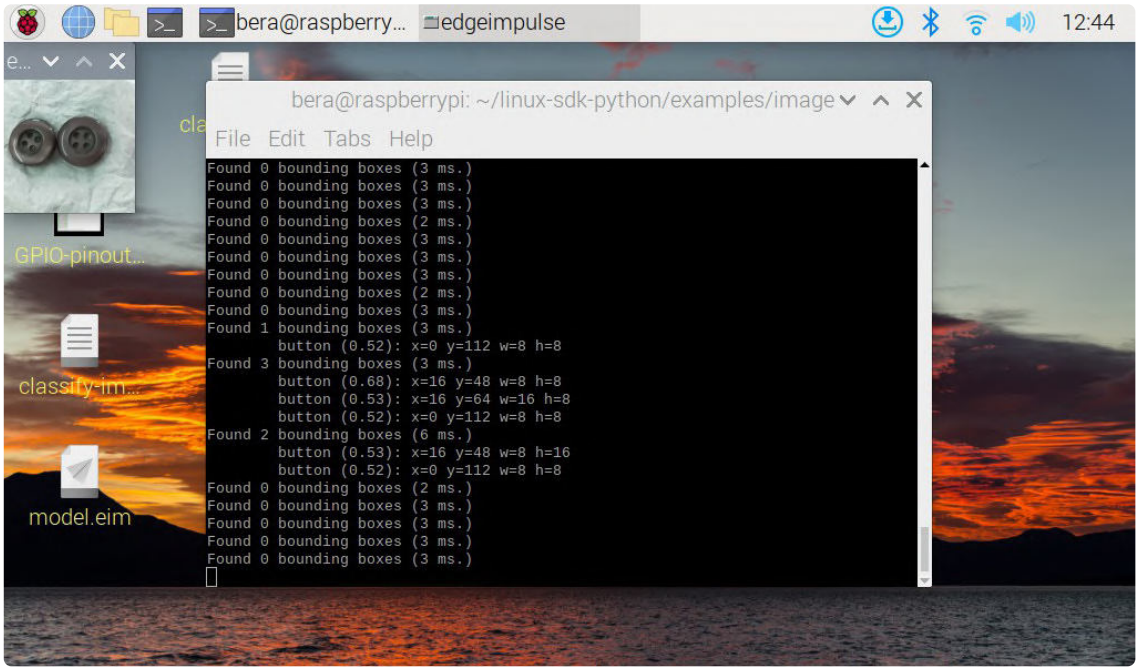


Figure 7. Après avoir retiré un bouton, le modèle l'a compté correctement du premier coup.

```
python3 classify-image.py /home/bera/downloads/model.eim
```

Le fichier modèle **.eim* doit être chargé à partir du répertoire correspondant à son emplacement. Si vous préférez, vous pouvez également le copier dans le répertoire du SDK !

C'est ainsi que vous devez charger le fichier Python avec le fichier *model.eim* téléchargé. Le programme trouvera automatiquement le module caméra (connecté par USB ou par Cam-Port) et commencera à fonctionner ! Dans le coin supérieur gauche, une petite fenêtre de caméra de 120 × 120 pixels s'ouvrira, et les boutons identifiés seront marqués d'un petit marqueur rouge. Les numéros identifiés s'affichent sur le terminal. Veillez à ce qu'il y ait suffisamment de lumière et que la caméra soit correctement mise au point sur les boutons. Ceci est particulièrement important pour les caméras bon marché. C'est pourquoi, si vous exécutez le modèle sur votre smartphone, il produit des images de bien meilleure qualité et compte beaucoup plus rapidement. Assurez-vous tout de même que la lumière et la mise au point sont correctes, et vous obtiendrez de meilleurs résultats.

Dans les captures d'écran suivantes de l'ordinateur Raspberry Pi, en haut à gauche, on peut voir la petite fenêtre de prise de vue de 120 × 120 pixels, dans laquelle les trois boutons sont identifiés et comptés par le modèle. Vous pouvez observer le marqueur rouge visible sur les trois boutons.

Dans la **figure 6**, quatre boutons sont manquants en raison d'une mise au point défaillante et d'un problème d'éclairage. L'appareil photo n'était pas non plus fixé sur un pied pour ce travail ! C'est pourquoi je recommande de fixer l'appareil sur un pied, comme dans le cas d'une utilisation avec un microscope. Veillez à ne pas regarder en biais depuis le haut de l'appareil.

Dans la **figure 7**, j'ai enlevé un bouton et le modèle l'a d'abord compté correctement. Il a compté deux boutons, mais au moment

Figure 8. Mon prototype.

où j'ai appuyé sur le bouton *pr-screen*, il s'est désaligné et n'a pas compté. Veillez également à ce que la caméra dispose d'un câble long (un câble plat, voir mon prototype à la **figure 8**) pour les positionnements. Ce câble est disponible sur Amazon. Toutefois, une fois que la caméra est fixée sur un support avec une bonne quantité de lumière, elle fonctionnera de manière infaillible.

Personnalisez votre modèle

Jetez un coup d'œil au fichier *classify-image.py* file. Il s'agit d'un simple fichier Python qui peut être adapté sans grande difficulté. Dans ce fichier Python, j'ai ajouté un module *espeak* de sorte qu'au moment où il trouve un ou plusieurs boutons, il énonce le nombre de boutons qu'il trouve. Pour installer *espeak*, sur votre Raspberry Pi, lancez la commande :

```
sudo apt-get install espeak
```

Reportez-vous au **listage 1** avec le fichier Python incluant mes modifications.
Espeak est un module autonome de synthèse vocale pour Python. Il ne nécessite pas de connexion Internet pour fonctionner.

Exécution modifiée

Vous avez maintenant modifié le programme Python. Si vous exécutez le fichier Python maintenant, il localisera le bouton (en haut à gauche, une petite fenêtre de caméra de 120 × 120 pixels s'ouvrira), les nombres s'afficheront dans la fenêtre du terminal et le haut-parleur associé énoncera le nombre : « Trouvé cinq boutons / Trouvé deux boutons, » etc. Si vous souhaitez faire fonctionner un relais, allumer une LED, etc., importez la bibliothèque *GPIO* de Python, puis activez le GPIO associé pour faire fonctionner le relais, etc. Par contre, pour faire fonctionner un relais, vous devez utiliser un transistor de commutation pour augmenter la quantité de courant nécessaire pour enclencher le relais.

Et ensuite

Edge Impulse a démarré en 2019, avec pour objectif de permettre aux développeurs de créer la prochaine génération d'appareils intelligents. Depuis, des programmes et des appareils basés sur l'IA sont apparus sur ESP32, Jetson Nano, Raspberry Pi, Orange Pi, Maixduino, OpenMV, Nicla Vision, et bien d'autres encore. Cette tendance va encore s'accentuer dans les jours à venir ! L'époque des superordinateurs ou des ordinateurs de grande marque est révolue. Les petits appareils modulaires à faible consommation sont en train de conquérir rapidement cet espace. Et qui sait, peut-être aurons-nous bientôt la sagesse intégrée et prête à l'emploi à portée de main !

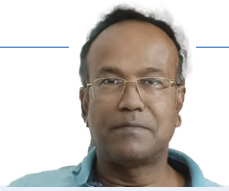
VF : Laurent Rauber — 230575-04

Questions ou commentaires ?

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

Listage 1. Programme Python pour exprimer le nombre de boutons avec l'outil espeak.

```
#!/usr/bin/env python
import device_patches    # Device Specific patches - taken care by the software
import cv2  #import Computer Vision
import os
import sys, getopt
import signal
import time
from edge_impulse_linux.image import ImageImpulseRunner
import subprocess    #this one have been added by Bera
…
        elif "bounding_boxes" in res["result"].keys():
                print('Found %d bounding boxes (%d ms.)' % (len(res["result"]["bounding_boxes"]),
                res['timing']['dsp'] + res['timing']['classification']))
                if  (len(res["result"]["bounding_boxes"])>0):
                        exitCode = subprocess.call(["espeak","-ven+f3","-a200"," Found %d Buttons" %
                        len(res["result"]["bounding_boxes"])  ])   #This one have been added by Bera
…
```



À propos de l'auteur

Somnath Bera, ingénieur en mécanique du Jalpaiguri Govt. Engg. College de Jalpaiguri, en Inde, travaille comme directeur général chez NTPC, le plus grand producteur d'électricité du pays. Il a une profonde passion pour l'électronique, comme en témoignent ses plus de 60 projets innovants sur *Elektor Labs*, dont plus de 10 ont fait l'objet d'un article dans *Elektor Mag*. Ses projets sont souvent axés sur la résolution de problèmes dans des domaines tels que la gestion des déchets et des ressources naturelles. Somnath aime utiliser des approches innovantes et des plateformes comme Arduino, Raspberry Pi et ESP32, associées à différents types de capteurs et de systèmes sans fil, pour créer des solutions efficaces et rentables.

Produits

- **Raspberry Pi 4 B (1 GB RAM)**
  www.elektor.fr/18966
- **G. Spanner, *Machine Learning with Python for PC, Raspberry Pi, and Maixduino* (E-book, Elektor)**
  www.elektor.fr/20150
- **D. Situnayake, Jenny Plunkett, *AI at the Edge* (O'Reilly)**
  www.elektor.fr/20465

LIENS

[1] Edge Impulse : https://edgeimpulse.com/
[2] Installation d'Edge Impulse sur un Raspberry Pi 4 : http://tinyurl.com/ysc6mtuz
[3] Linux Python SDK : http://tinyurl.com/2bat4w6z
[4] Page du projet sur Elektor Labs : https://www.elektormagazine.fr/labs/count-speak-number-of-buttons

des solutions à vos problèmes de développement de systèmes embarqués les plus délicats

Par Stuart Cording, pour Mouser Electronics

Le développement de systèmes embarqués offre aux ingénieurs de nombreuses opportunités… et bien davantage de défis aussi divers et variés ! La combinaison de l'électronique, des logiciels et des systèmes nous ouvre la possibilité de nous plonger dans le domaine analogique pour résoudre un problème de synchronisation numérique, examiner des problèmes complexes de boucle de contrôle ou évaluer la qualité des relevés réalisés par des capteurs. Ainsi, pour mieux répondre aux besoins des développeurs et si possible leur faciliter la vie, les fabricants d'équipements de test n'ont de cesse de proposer de nouvelles fonctions intelligentes, davantage de connectivité et même des applications pour smartphones.

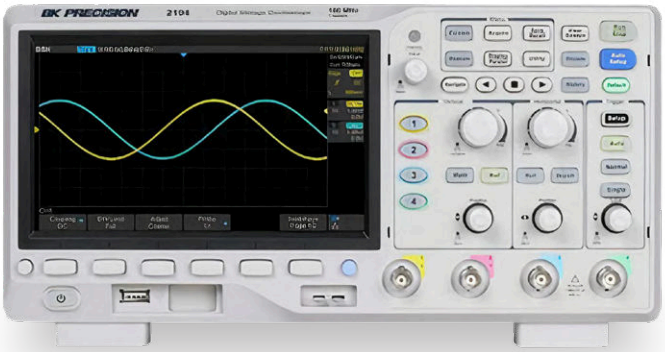


Figure 1. Le modèle 2194 de B&K Precision dispose d'une profondeur de mémoire impressionnante de 14 Mpts. Une fonction de recherche permet de trouver facilement les anomalies de signal.

Par exemple, les oscilloscopes ne se contentent plus d'afficher tout simplement la tension par rapport à un axe temporel. Ils sont désormais capables de réaliser des calculs complexes sur la base des signaux entrants à des fins d'analyse, de décoder des communications de données en série et de déclencher des signaux complexes. Il existe également des oscilloscopes « soft » qui reposent sur la technologie FPGA et qui ressemblent davantage à des laboratoires miniatures. Ces appareils offrent toute la gamme standard de mesures prêtes à l'emploi, mais ils peuvent aussi être configurés pour des tâches plus spécifiques grâce à leurs fonctions programmables. Si vous n'avez rien contre le fait d'utiliser votre ordinateur portable comme écran et si vous manquez de place dans votre laboratoire, ce type d'oscilloscope est un choix tout à fait judicieux.

Recherche de runts avec un oscilloscope

L'avènement de l'oscilloscope numérique a rendu la recherche des causes de problèmes beaucoup plus simple en comparaison à l'ancienne méthode qui consistait à prendre une photo de l'écran CRT. Les équipements modernes se connectent facilement à des réseaux ou à des ordinateurs, ce qui permet de partager et d'analyser les signaux relevés et simplifie la configuration de l'appareil et les procédures de tests automatisés. À cet égard, le modèle 2194 [1] de B&K Precision ne semble pas très différent de la plupart des autres oscilloscopes à quatre voies du marché (voir **figure 1**). Cependant, comme pour tout investissement important, nous recommandons de consulter son manuel d'utilisation afin de découvrir toutes les possibilités qu'offre cet appareil.

Avec une belle largeur de bande d'entrée de 100 MHz, un taux d'échantillonnage maximal de 1 Géch/s et un écran TFT de 7 pouces avec une résolution de 800 × 480 pixels, le 2194 est aussi bien équipé que des appareils similaires valant parfois plus du double de prix. Ne pesant que 2,6 kg pour des dimensions de 31 × 15 × 13 cm, il est facile à déplacer et sera peu encombrant dans votre laboratoire. Les connecteurs BNC et USB pour supports de stockage sont facilement accessibles sur la face avant, tandis que le panneau arrière comporte un port USB pour le contrôle, un port Ethernet et la sortie de déclenchement BNC (qui sert également de sortie pass/fail

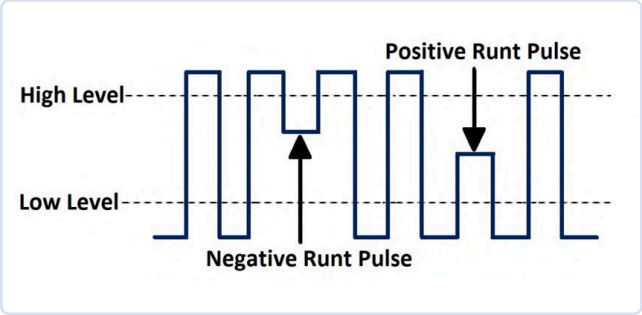


Figure 2. Les impulsions runt, provoquées par des problèmes avec les circuits de commande, peuvent être utilisées comme déclencheur ou identifiées dans les signaux capturés grâce à la fonction de recherche du 2194.

Figure 3. Le Moku:Go est un véritable laboratoire miniature regroupant les capacités de mesure de huit appareils de test dans un boîtier à peine plus large que la main.

pour l'automatisation des tests). L'appareil est également doté d'un point de fixation pour câbles antivol comme ceux utilisés avec les ordinateurs portables. Pour l'analyse du signal, l'appareil propose 38 types de paramètres de mesure et de statistiques, dont les valeurs comptage, écart type, temps de montée et de descente et crête à crête. Il décode aussi les protocoles de bus série I²C, SPI, UART, CAN et LIN.

Son taux de rafraîchissement de 100 000 wfms/s permet de détecter les glitches et autres événements peu fréquents. La profondeur de sa mémoire est elle aussi impressionnante. Avec 14 Mpts disponibles, elle permet de capturer de longues périodes de signaux qui peuvent ensuite être analysées à l'aide de la fonction zoom afin d'y détecter d'éventuelles anomalies. Si la recherche d'anomalies dans des ensembles de données aussi volumineux tient du défi, la fonction de recherche du 2194 permet de détecter toute une gamme d'événements grâce aux options « Front » (Edge), « Pente » (Slope), « Impulsions » (Pulse) et « Intervalle » (Interval).

Les développeurs sont souvent mis au défi de trouver les signaux numériques incorrectement transmis. Appelés *impulsions runt* (voir **figure 2**), ces signaux apparaissent lorsque la vitesse de balayage est insuffisante pour atteindre le niveau logique souhaité dans le temps requis. Le 2194 peut retrouver les impulsions runt grâce à sa fonction de recherche, mais il peut aussi les retrouver en les utilisant comme déclencheur. Il faut pour cela définir un niveau de déclenchement supérieur et inférieur. Lorsqu'un signal franchit un seuil mais pas l'autre, le déclencheur capture la forme d'onde à des fins d'analyse. Cette démarche permet de résoudre beaucoup plus simplement les pannes intermittentes autrement difficiles à trouver.

Le Moku:Go de Liquid Instruments

Pour les personnes qui sont toujours en déplacement ou qui travaillent sur le terrain, un oscilloscope traditionnel est généralement trop lourd et trop encombrant à transporter. Heureusement, il existe des alternatives intéressantes pour peu que l'on consente à utiliser son ordinateur portable comme interface graphique. L'une d'entre elles est le Moku:Go [2] de Liquid Instruments. Il s'agit d'une unité compacte de 24 × 3,8 × 13 cm pour un poids de seulement 750 g (voir **figure 3**). Grâce à sa conception basée sur la technologie FPGA, le Moku:Go rassemble huit instruments en un avec, entre autres, une fonction oscilloscope de 30 MHz et un analyseur de spectre en temps réel, un générateur de forme d'onde de 20 MHz, un contrôleur PID et un générateur de forme d'onde arbitraire.

Le Moku:Go M0 dispose de deux entrées analogiques, deux sorties analogiques et 16 E/S numériques. Une connectique USB-C permet de le connecter facilement à tout ordinateur Windows ou Mac. Par rapport au modèle M0, le M1 dispose en plus d'un bloc d'alimentation (PSU) programmable à deux canaux pour -5 V à +5 V et 0 V à 16 V à 150 mA livré avec les câbles nécessaires. Le M2 est quant à lui équipé de deux blocs d'alimentation programmables supplémentaires de 0,6 V à 5 V jusqu'à 1 A par canal ainsi que d'une connectivité Ethernet.

Bien que ces blocs d'alimentation programmables ne semblent pas constituer une option absolument indispensable, leur présence s'avère tout à fait intéressante lorsque l'on se rend compte que le courant consommé peut être utilisé comme paramètre dans les configurations de test, comme on peut le voir dans la note d'application « Évaluation de convertisseur à l'aide du TI TPS63802EVM » [3] expliquant comment mesurer l'efficacité d'un convertisseur buck-boost. Les blocs d'alimentation couplés aux deux canaux analogiques sont plus que suffisants pour mesurer l'efficacité, tandis que le canal Math, lorsqu'il est équipé de la résistance de charge, fournit aisément la nécessaire puissance en watts. Il est également possible d'automatiser les mesures à l'aide d'API de programmation Python, MATLAB et LabView.

Nous ne pouvons pas examiner tous les aspects du Moku:Go ici, mais nous pouvons tout de même mentionner ses impressionnantes notes d'application ainsi [4] que l'application elle-même [5] avec son interface graphique claire et ordonnée. Un mode démo vous permet en plus d'essayer toutes les différentes fonctionnalités. Vous pourrez ainsi vous rendre compte du zoom avant et arrière sur les axes X et Y ainsi que de la configuration des canaux et des fonctions mathématiques. Si vous hésitez encore à acquérir cet outil, l'application pourra donc vous aider à prendre une décision.

Tester et compartimenter les composants

La conception d'une alimentation nécessite une excellente compréhension des inductances et des condensateurs utilisés. Il se peut que par souci de précision, chaque composant doive être testé et compartimenté. Que ce soit en laboratoire ou à l'usine, la série T3LCR [6] de compteurs LCR de Teledyne LeCroy pourra vous être d'une aide précieuse

Figure 4. Dans le laboratoire de conception, le T3LCR de Teledyne LeCroy peut être utilisé pour caractériser les composants tandis que, dans un environnement de production, il peut être utilisé pour compartimenter les composants.

Figure 5. Les instruments de mesure de la série 250W d'Extech permettent de mesurer toute une série de paramètres environnementaux comme le niveau sonore, le régime, le flux lumineux, l'humidité relative et la vitesse de l'air.

(voir **figure 4**). Doté d'un grand écran TFT de 3,5 pouces, cet appareil est facilement configurable pour des mesures à quatre fils par le panneau avant ou via ses ports USB ou RS-232C. Les trois modèles T3LCR prennent en charge des fréquences de test de 10 Hz à 2 kHz, 100 kHz ou 300 kHz avec une précision de base de 0,05 %.

Des fonctionnalités comme le contrôle automatique de niveau (ALC) permettent de fournir aux dispositifs MLCC une tension de test constante, tandis que le courant de test réglable convient aux mesures d'inductance. Une polarisation CC interne réglable de ±2,5 V peut être utilisée pour simuler simultanément le CA et le CC afin d'évaluer la variation de capacité. En mode de *mesure de liste*, il est possible de collecter jusqu'à 10 paramètres de mesure automatisés de façon à permettre la caractérisation des composants et les éventuelles tendances de variation. Le T3LCR dispose à l'arrière d'un connecteur DB-25 pour assurer l'interface avec un gestionnaire lors du compartimentage des composants et prend en charge jusqu'à dix compartiments. Des mesures de modèles équivalents en série et en parallèle sont disponibles pour la résistance, l'inductance et la capacité. L'appareil permet de relever une douzaine de paramètres supplémentaires, notamment le facteur de dissipation (D), le facteur de qualité (Q), l'angle de phase et la résistance au courant continu.

Assurer le suivi de son environnement

Lorsque l'on développe un système embarqué, il est souvent nécessaire de comparer les données de capteurs collectées par le microcontrôleur avec des résultats de mesures réalisées par des équipements de test professionnels. D'autres fois, il est nécessaire de procéder à des relevés de mesures sur une longue période dans le cadre de tests de l'environnement. La série d'instruments de mesure de paramètres environnementaux connectés Bluetooth 250 W [7] d'Extech est l'outil idéal pour ce type de situation (voir **figure 5**). Ces appareils permettent en effet de mesurer un large éventail de paramètres : régime (tr/min), intensité lumineuse, humidité relative, niveau sonore ou encore vitesse de l'air. Grâce à leurs dimensions compactes (54 × 28 × 120, 176 mm), les unités tiennent facilement dans la main. Un grand écran LCD éclairé en facilite la lecture.

Le RPM250W est un tachymètre laser avec une plage de mesure de 10 à 99 999 tr/min et une précision de 0,04 %. Le LT250W mesure l'intensité lumineuse jusqu'à 100 000 lux. Le débitmètre d'air compact AN250W affiche les résultats en pieds/min, m/s et nœuds ainsi que la température de l'air ambiant. Pour le niveau de pression acoustique, le SL250W propose des mesures de fréquence « audition humaine » pondérées « A » avec enregistrement des valeurs max/min. Enfin, l'humidité relative et la température peuvent être mesurées et enregistrées avec le RH250W.

Chaque appareil est doté de la connectivité Bluetooth, ce qui leur permet d'être appariés à un appareil iOS ou Android et commandés depuis l'application Extech ExView. L'application peut collecter simultanément des données provenant d'un maximum de huit compteurs de la série 250 W et peut être configurée pour générer des alarmes sonores à des niveaux élevés/bas. Les données sont enregistrées localement et peuvent être partagées sous la forme d'un fichier au format csv. Il est en outre possible de créer des rapports de mesure au format PDF intégrant des photos des lieux où les mesures tests sont réalisées.

Un outil de mesure pour chaque développeur de systèmes embarqués

Développer des systèmes embarqués est une tâche diversifiée et même plutôt amusante, mais cela n'en fait pas une chose facile pour autant. Il est parfois indispensable de disposer d'un équipement de test de pointe, par exemple pour trouver la cause d'une défaillance, définir l'efficacité réelle de convertisseurs de puissance ou comparer des mesures de paramètres environnementaux avec les relevés des capteurs intégrés. Dans d'autres cas, il s'agira de comprendre la valeur précise des composants utilisés ou d'effectuer un compartimentage lors de la production. Heureusement, quelle que soit la nature du problème auquel vous êtes confronté durant le développement de votre système embarqué, l'une des solutions présentées ci-dessus devrait vous aider à le résoudre.

230750-04

À propos de l'auteur

Stuart Cording est un journaliste indépendant qui écrit notamment pour Mouser Electronics. Il se spécialise dans la création de contenu vidéo et se concentre sur les nouvelles et les approfondissements techniques. Il s'intéresse particulièrement à la technologie elle-même, à la façon dont elle s'intègre dans les applications finales et aux perspectives d'évolution.

Mouser Electronics est un distributeur agréé de semi-conducteurs et de composants électroniques qui se spécialise dans le lancement de nouveaux produits de ses partenaires fabricants leaders.

LIENS

[1] B&K Precision Model 2194: https://eu.mouser.com/new/bk-precision/bk-2194-oscilloscope/
[2] Liquid Instruments Moku:Go: http://tinyurl.com/Moku-Go
[3] Converter Evaluation Using TI TPS63802EVM: http://tinyurl.com/AppNoteConverter
[4] Application Notes by Liquid Instruments: https://www.liquidinstruments.com/blog/category/application-notes/
[5] Windows & macOS apps by Liquid Instruments: https://www.liquidinstruments.com/products/desktop-apps/
[6] Teledyne LeCroy T3LCR: http://tinyurl.com/TeledyneT3LCR
[7] Extech 250W: https://eu.mouser.com/new/extech/extech-250w-meters/

Terminal ESP32

un appareil portable avec un écran tactile

Johan van den Brande (Belgique)

Le Terminal ESP32 d'Elecrow est un appareil portable piloté par un ESP32-3 avec un écran tactile capacitif TFT de 3,5 pouces, 480 × 320 pixels et une multitude de possibilités. Cet appareil peut être un complément intéressant à vos projets si vous avez besoin d'un écran tactile avec des possibilités d'interfaçage.

L'écran basé sur un pilote d'affichage ILI9488 a une profondeur de couleur de 16 bits, ses capacités tactiles sont gérées par un FT6236. Les deux puces sont très bien prises en charge par la communauté Arduino. Le Terminal ESP32 est logé dans une coque acrylique noire, ce qui lui donne une impression de robustesse, et certainement pas dans un boîtier en plastique rudimentaire. Un connecteur de batterie est présent, avec un circuit de charge LiPo, mais la portabilité serait meilleure si une batterie était présente dans l'appareil. Si vous voulez rendre votre projet portable, vous devrez ajouter une batterie à l'arrière. Il y a deux trous de montage M3 à l'arrière que vous pouvez utiliser pour attacher des batteries ou le fixer au mur.

Le microcontrôleur ESP32-S3

En regardant ce qui alimente l'appareil, je suis toujours étonné par l'incroyable puissance de calcul des microcontrôleurs modernes. Le terminal ESP est alimenté par un ESP32-S3 d'Espressif [2], qui contient un microcontrôleur XTensa LX7 32 bits à double cœur cadencé à 240 MHz. Comparez cela à l'Arduino UNO original avec son ATMega328 8 bits de Microchip cadencé à 16 MHz !

Le microcontrôleur dispose de 512 Ko de SRAM à bord, ainsi que de 8 Mo de PSRAM. PSRAM signifie pseudo-statique RAM, une sorte de RAM statique externe qui, dans le cas de l'ESP32-S3, est connectée au microcontrôleur via une interface SPI.
En ce qui concerne la connectivité sans fil, nous avons le Wifi 2,4 GHz (2,4 GHz 802.11 b/g/n) et le Bluetooth 5 LE. Je ne le savais pas, mais Bluetooth 5 a des capacités de longue portée, et ce module prétend les prendre en charge. Le mode longue portée étend la portée de 10 à 30 mètres à plus d'un kilomètre.

Se connecter au monde physique

Le terminal ESP32 dispose d'un connecteur de batterie et d'un chargeur LiPo embarqué. Vous pouvez utiliser le connecteur USB-C pour alimenter l'appareil, mais aussi pour charger la batterie LiPo. Il y a également un emplacement pour carte micro-SD, utile si vous souhaitez stocker quelques images ou d'autres ressources (graphiques) comme des pages web pour votre application.
Les côtés du module comportent un total de quatre ports « Crowtail » [3]. Il s'agit d'interfaces 4 fils compatibles avec Grove de Seeed Studio. Il y a un connecteur

Figure 1. Le Terminal ESP32 dans son boîtier noir affichant les données météo.

Figure 2. Les quatre ports « Crowtail » sont visibles à l'arrière.

numérique, un analogique, un série (UART) et un I²C, ce qui est suffisant pour des projets simples. Si vous avez besoin de plus, vous pouvez utiliser le port I²C comme une extension.

Programmation du terminal ESP32

Le téléchargement du micrologiciel se fait via le câble USB-C, sur une connexion série rendue possible par un convertisseur USB-série CH340 [4]. Je n'ai pas eu de chance en essayant de faire fonctionner cela avec mon MacBook, pour lequel j'ai essayé d'installer le pilote officiel. Heureusement, j'ai toujours mon vieil ordinateur portable Linux, qui tourne sous Ubuntu et qui fonctionne parfaitement !

La première chose que je voulais essayer, c'était d'exécuter une version de Python sur l'appareil. Cela s'est avéré difficile. Bien que la documentation indique qu'il peut être programmé avec Python et MicroPython, je n'ai pas trouvé de micrologiciel téléchargeable prêt à l'emploi.

En lisant leur site web, on y trouve de nombreux exemples et un tutoriel pour Arduino [5], et c'est donc ce que j'ai commencé à utiliser pour mes applications de démonstration.

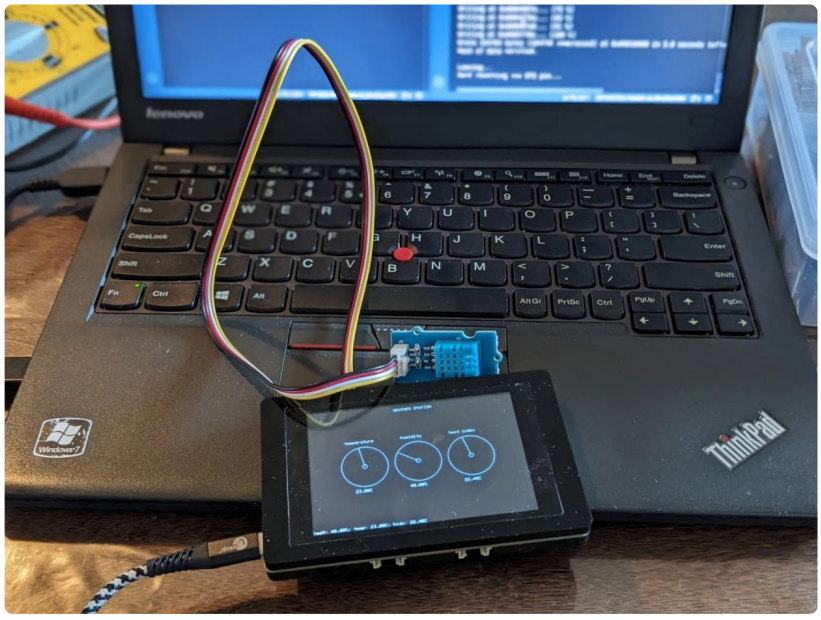


Figure 3. Ajoutez un capteur DHT-11 et un peu de code et vous obtiendrez une station météorologique simple.

Conception de l'interface utilisateur

Selon la page web du Terminal ESP32, l'appareil est certifié LVGL [6][7]. LVGL signifie *Light and Versatile Graphics Library*, qui est une bibliothèque graphique embarquée open source permettant de créer des interfaces utilisateur pour des microcontrôleurs et des affichages divers. Bien que la bibliothèque soit open source en elle-même, un éditeur de mise en page (par glisser-déposer) à logiciel propriétaire, Squareline Studio [8], est également disponible.

J'ai installé la version d'essai et bien qu'elle soit un peu déroutante à première vue, j'ai rapidement réussi à dessiner une interface utilisateur simple pour le projet de servomoteur ci-dessous. Si votre objectif est de créer des interfaces utilisateur pour des systèmes embarqués, cela vaut la peine d'investir un peu de temps dans l'apprentissage de ce produit.

Deux petits projets

J'ai décidé de créer deux projets d'exemple. Le premier projet est une petite station météorologique, où nous dessinons notre propre interface utilisateur en utilisant les primitives ligne et cercle. Le second projet est un exemple où nous contrôlons un servo à partir d'une interface utilisateur conçue avec Squareline Studio.

Station météo

Pour la station météorologique, nous utilisons un capteur DHT-11, qui peut détecter la température et l'humidité. Il possède une interface numérique à un fil et est bien supporté par Arduino. Pour ce projet, j'ai choisi la bibliothèque *Adafruit DHT [9]*. Elle fait partie d'un ensemble de bibliothèques de capteurs partageant un code commun, et par conséquent nous devons également installer la bibliothèque *Adafruit Unified Sensor Driver [10]*.

Dans ce projet, j'ai décidé de créer ma propre interface utilisateur à partir de rien, en utilisant des primitives pour imprimer du texte et créer quelques lignes graphiques. Pour cela, j'ai utilisé la bibliothèque graphique *LCD* et *e-Ink* de LovyanGFX [11][12], une autre bibliothèque graphique open-source.

Après avoir initialisé l'écran, pour lequel j'ai simplement copié un des exemples du site de Terminal ESP32, nous lisons la mesure du capteur DHT-11. Celui-ci nous donne trois valeurs : la température, l'humidité et l'indice de chaleur. L'indice de chaleur est une température ajustée par l'humidité réelle. Ces trois valeurs sont affichées dans un cadran simple, dessiné à l'aide de deux cercles et d'une ligne comme pointeur. L'écran est mis à jour toutes les deux secondes avec une nouvelle valeur.

Contrôleur de servo

Pour le contrôleur de servo, j'ai voulu essayer l'outil Squareline Studio pour créer l'interface utilisateur. Un curseur en forme d'arc est le seul widget utilisé. La plage du curseur est réglée de 0 à 180, ce qui correspond à l'angle du servo.

Lorsque vous concevez une interface utilisateur pour l'ESP32 à l'aide de cet outil, vous devez commencer par le modèle *Arduino with TFT_eSPI* et définir la profondeur de couleur à 16 bits et la résolution à 480 × 320 pixels. Après avoir exporté le code généré, vous devrez le décompresser dans un répertoire nommé *ui* dans votre dossier *libraries* d'Arduino. Normalement, vous trouverez ce répertoire *libraries* dans le même répertoire que celui où se trouvent vos croquis Arduino… Il m'a fallu un certain temps pour m'en rendre compte.

Le servo est piloté par la bibliothèque *ESP32Servo*. La bibliothèque standard *Arduino servo* ne fonctionne pas avec l'ESP32.

Le code est assez basique, dans la fonction `loop`, nous lisons l'angle du curseur d'arc, qui renvoie un nombre entre 0 et 180, puis nous envoyons cette valeur à la fonction `servo.write`.

Vous pouvez trouver le code source des deux projets dans la section de téléchargement de cet article. Je n'ai pas inclus le dossier *libraries* dans le téléchargement, car il est (beaucoup) trop volumineux. Installer les dépendances soi-même n'est pas compliqué.

Conclusion

Le Terminal ESP32 d'Elecrow n'est pas aussi facile à apprivoiser que je l'espérais, mais il a beaucoup de potentiel. Si vous voulez investir du temps pour apprendre la bibliothèque LVGL ou une autre bibliothèque d'interface utilisateur, et que vous êtes à l'aise avec Arduino, alors avec le Terminal ESP32 vous pouvez ajouter une interface utilisateur avec un contrôle tactile à votre prochain projet.

Cela aurait été plus facile si le micrologiciel MicroPython [15] avait été disponible en téléchargement sur le site du produit. J'ai cherché un peu sur internet, mais je n'ai rien trouvé. Il est certainement possible de compiler votre propre micrologiciel, et c'est peut-être une bonne idée pour votre prochain projet ?

VF : Laurent Rauber — 230755-04

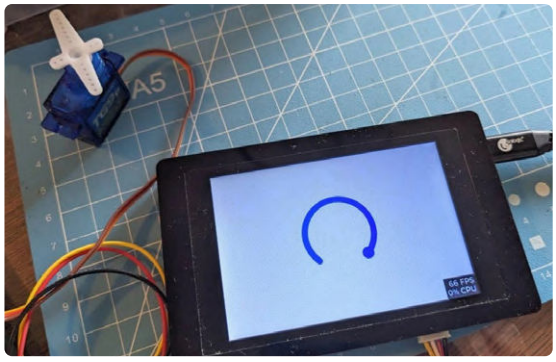


Figure 4. Le curseur circulaire contrôle la position du servo.

Questions ou commentaires ?

Contactez Elektor (redaction@elektor.fr).

Produits

- **ESP Terminal (carte de développement ESP32-S3 avec écran tactile TFT capacitif de 3,5 pouces)** www.elektor.fr/20526

LIENS

[1] Terminal ESP32 d'Elecrow : https://www.elektor.fr/esp-terminal-esp32-s3-based-development-board-with-3-5-capacitive-tft-touch-display
[2] Fiche technique de l'ESP32-S3 : https://www.espressif.com/sites/default/files/documentation/esp32-s3_datasheet_en.pdf
[3] Ports « Crowtail » : https://www.elecrow.com/wiki/index.php?title=Main_Page#Crowtail
[4] CH340 convertisseur USB-série : https://cdn.sparkfun.com/datasheets/Dev/Arduino/Other/CH340DS1.PDF
[5] Tutoriel pour Arduino : https://www.elecrow.com/wiki/index.php?title=Lesson01_Introducing_the_ESP32_Display_series_and_environment_configuration
[6] LVGL : https://lvgl.io/
[7] Documentation LVGL : https://docs.lvgl.io/latest/en/html/index.html
[8] Squareline Studio : https://squareline.io/
[9] Bibliothèque DHT-11 : https://github.com/adafruit/DHT-sensor-library
[10] Unified Sensor Library : https://github.com/adafruit/Adafruit_Sensor
[11] LovyanGFX LCD and e-Ink graphics library: https://github.com/ropg/LovyanGFX
[12] Documentation LovyanGFX : https://lovyangfx.readthedocs.io/en/latest/index.html
[13] Bibliothèque ESP32Servo : https://www.arduino.cc/reference/en/libraries/esp32servo/
[14] Téléchargement du micrologiciel : https://www.elektormagazine.fr/review/terminal-esp32-test
[15] « MicroPython pour l'ESP32 et ses copains - Partie 1 : installation et premiers programmes », Günter Spanner, 2021: https://www.elektormagazine.fr/articles/micropython-pour-l-esp32-et-ses-copains-partie-1

commencer avec le RTOS Zephyr

aussi puissant que difficile à maîtriser

Clemens Valens (Elektor)

Zephyr est un système d'exploitation en temps réel (RTOS) petit et évolutif, optimisé pour les appareils à ressources limitées, sur plusieurs architectures. Hébergé par la Fondation Linux (voir [1]), le projet est un effort de collaboration open-source dont l'objectif est de fabriquer un RTOS de premier ordre. Le système d'exploitation (SE) Zephyr a gagné en popularité ces dernières années dans l'informatique embarquée et aujourd'hui, de nouveaux microcontrôleurs et cartes de développement arborent fièrement le support de Zephyr. Il est temps d'y regarder de plus près.

Zephyr est évolutif, ce qui lui permet de s'adapter à une large gamme d'appareils ayant des contraintes de ressources différentes. L'évolutivité est obtenue grâce à une architecture modulaire qui permet aux développeurs de n'inclure que les composants dont ils ont besoin, minimisant ainsi l'empreinte du système. Le site web de Zephyr indique qu'il fonctionne sur des systèmes dotés d'une mémoire de 8 KO jusqu'à plusieurs gigaoctets.

Prise en charge d'un grand nombre de supports

Zephyr supporte une large gamme d'architectures, y compris ARM, x86, RISC-V, Xtensa, MIPS et plus encore. Les FPGA sont également pris en charge avec les noyaux logiciels Nios2 et MicroBlaze. À l'heure où nous écrivons ces lignes, Zephyr répertorie plus de 600 cartes utilisables, dont les Arduino UNO R4 Minima, GIGA R1 WIFI et Portenta H7, de multiples saveurs de l'ESP32, les deux versions du BBC micro:bit, le Raspberry Pi Pico (et même le Raspberry Pi 4B+), les cartes nRF51 et nRF52, la famille NXP MIMXRT1010-EVK, et les familles STM32 Nucleo et Discovery. Je n'ai cité que les cartes couramment rencontrées dans Elektor, mais il en existe bien d'autres.
Outre les cartes processeurs, Zephyr prend également en charge de nombreuses cartes d'extension (connues sous le nom de *shields*) et est livré avec des pilotes pour toutes sortes d'interfaces et plus de 150 capteurs.

Multitâche, mise en réseau et gestion de l'énergie

En tant que système d'exploitation en temps réel (RTOS), Zephyr offre des fonctionnalités telles que le multitâche préemptif, la communication inter-thread et la prise en charge de l'horloge en temps réel. Le système d'exploitation est également doté de technologies et de protocoles de réseau tels que TCP/IP, Bluetooth, IEEE 802.15.4 (utilisé par exemple dans Zigbee), MQTT, NFS et LoRaWAN. Associées à ses capacités de mise en réseau, les fonctionnalités de gestion de l'énergie intégrées à Zephyr le rendent adapté aux applications IdO économes en énergie et aux appareils fonctionnant sur batterie.
Un ensemble de bibliothèques et d'intergiciels simplifient les tâches courantes, telles que les protocoles de communication, les systèmes de fichiers et les pilotes de périphériques.
Sachez que Zephyr est également conçu pour être compatible avec les certifications de sécurité telles qu'ISO 262, ce qui le rend adapté aux applications critiques en matière de sécurité.

Inspiré par Linux

Zephyr n'est pas Linux, mais il utilise des concepts, des techniques et des outils utilisés par ce dernier. Par exemple, Kconfig est utilisé pour configurer le système d'exploitation, et les propriétés et configurations matérielles sont décrites à l'aide de la spécification de l'arbre des périphériques (DTS) [2]. Les développeurs Linux se sentiront donc rapidement à l'aise lorsqu'ils coderont pour Zephyr.

Open Source

Enfin, Zephyr est publié sous la licence Apache 2.0, qui permet une utilisation commerciale et non commerciale. Sa communauté d'utilisateurs fournit de l'aide et de la documentation. Vous pouvez également la rejoindre.

Essai de Zephyr

Essayer Zephyr est sur ma liste de choses à faire depuis plusieurs années, mais mes premières expériences avec cet SE n'étaient pas très encourageantes, et je n'ai donc jamais approfondi. À l'époque,

Figure 1. La minuscule carte BBC micro:bit est une excellente cible pour essayer le RTOS Zephyr. Qui aurait pensé que cette petite carte destinée à enseigner la programmation à des enfants de 10 ans à l'aide de MakeCode, un langage graphique de type Scratch, serait aussi un super outil pour les développeurs chevronnés de logiciels embarqués désireux d'apprendre un système d'exploitation en temps réel de qualité industrielle ?

l'un de ses principaux problèmes (outre le fait de le compiler sans erreur) était qu'il nécessitait un adaptateur de programmation pour programmer le contrôleur cible, ce qui le rendait moins adapté aux *makers*. Grâce à Arduino et son chargeur d'amorçage (*bootloader*), nous nous sommes habitués à ne pas avoir besoin d'outils de programmation spéciaux, et en exiger un me semblait un pas en arrière.

Choisir une carte

Les choses ont évolué depuis. Comme mentionné plus haut, Zephyr supporte aujourd'hui plus de 600 cartes de microcontrôleurs. Il y a de fortes chances que vous possédiez déjà une ou plusieurs compatibles. En regardant la liste, j'ai découvert que j'en avais plus d'une douzaine différentes à ma disposition.

Vive la BBC micro:bit !

J'ai essayé la plupart d'entre elles pour finalement me fixer sur la BBC micro:bit pour mes expériences (**figure 1**, connu par Zephyr sous le nom de `bbc_microbit` ou `bbc_microbit_v2`, selon la version de la carte). Comparé à mes autres options, en plus d'être facilement disponible, elle a probablement le meilleur support Zephyr, ce qui signifie que tous ses périphériques sont accessibles et supportés par quelques exemples. Mieux encore, il peut être programmé *et* débogué sans avoir besoin d'outils supplémentaires.

La populaire ESP-WROOM-32 (connu par Zephyr sous le nom `esp32_devkitc_wroom`) est également une candidate appropriée, mais le débogage nécessite un outil externe.

L'Arduino GIGA R1 WIFI est une autre bonne option, mais son *bootloader* est détruit lors de l'utilisation de Zephyr. On peut le restaurer, bien sûr, mais c'est un effet collatéral non souhaité.

Officiellement, l'Arduino UNO R4 Minima nécessite une sonde de programmation compatible SWD (comme beaucoup d'autres cartes, dont la Raspberry Pi Pico), mais j'ai trouvé un moyen de contourner ce problème en utilisant *dfu-util* (voir ci-dessous). Cependant, comme la GIGA R1, son chargeur d'amorçage Arduino est piétiné par Zephyr.

Utilisez plutôt un émulateur

Si vous n'avez pas de carte appropriée, mais que vous voulez vraiment essayer Zephyr, sachez qu'il dispose d'un support d'émulateur intégré pour QEMU (sur Linux/macOS uniquement). Cela vous permet d'exécuter et de tester des applications virtuellement. Renode d'Antmicro est capable de réaliser des prouesses similaires, bien que je ne l'aie pas essayé.

Installer Zephyr

J'ai installé Zephyr sur un ordinateur fonctionnant sous Windows 11, je n'ai pas essayé Linux ou macOS. Des instructions d'installation détaillées et fonctionnelles sont disponibles en ligne [3]. Les étapes à suivre sont clairement indiquées et ne nécessitent pas d'explications supplémentaires. J'ai utilisé un environnement Python virtuel, comme suggéré. Cela signifie que vous devez noter la commande d'activation de l'environnement virtuel quelque part, car vous en aurez besoin à chaque fois que vous commencerez à travailler. Si vous utilisez Windows PowerShell, vous devez lancer le *script activate.ps1* ; dans l'invite de commandes, il s'agit du fichier *batch activate.bat*. Windows PowerShell est plus performant pour traiter les messages émit par le compilateur et l'éditeur de liens (**figure 2**).

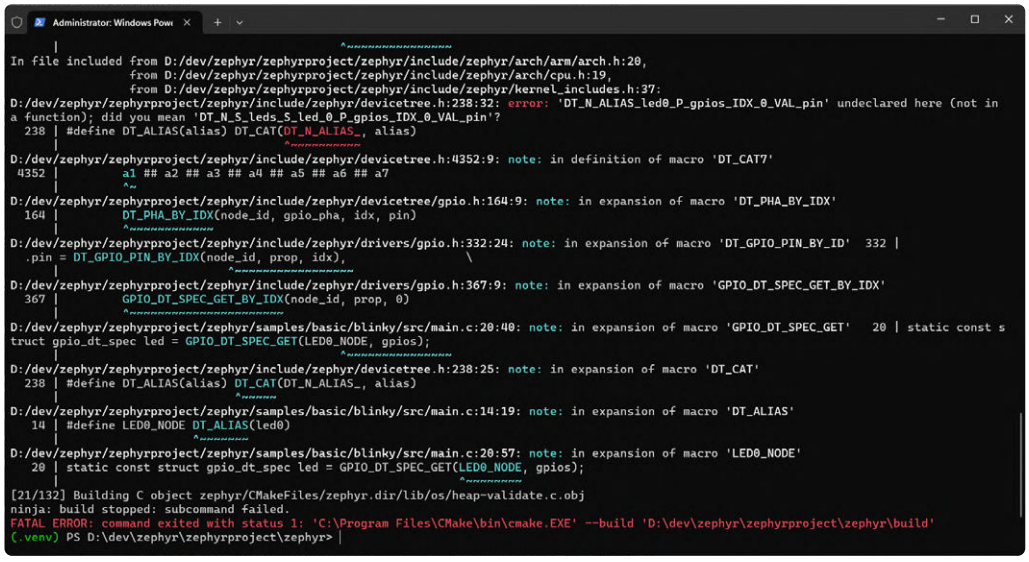


Figure 2. L'outil de construction west de Zephyr est destiné à fonctionner dans un terminal. Le cmd.exe de Windows peut être utilisé, mais ce n'est pas un terminal. Windows PowerShell, alias Terminal, est donc plus adapté.

Figure 3. Dans de nombreuses situations (mais pas toutes), un programmateur/débogueur JTAG ou SWD est nécessaire pour vos expériences Zephyr.

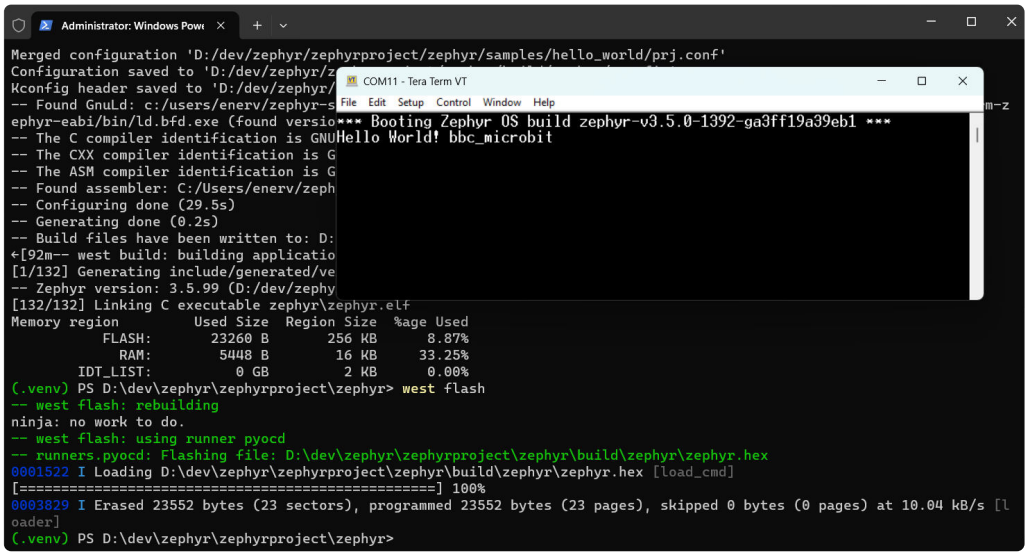


Figure 4. L'exemple de Hello World n'est pas très étendu. Si vous êtes trop lent à ouvrir un terminal série, vous ne verrez pas le message de bienvenue.

Zephyr se compose de deux parties, le SE lui-même et un SDK contenant une collection de chaînes d'outils microcontrôleur (21 au moment de la rédaction…). Le SE et le SDK n'ont pas besoin d'être installés au même endroit. Au total, pour moi, cela a consommé environ 12 GB d'espace précieux sur le disque dur. Pour récupérer de l'espace, vous pouvez supprimer les chaînes d'outils dont vous n'avez pas besoin.

Après l'installation, vérifiez qu'il fonctionne en compilant un exemple (*sample*) et en le téléchargeant sur la carte avec la commande ci-dessous. Remplacez `<ma_carte>` par le nom de votre carte, par exemple `arduino_uno_r4_minima` :

```
west build -p always -b <my_board> samples/basic/blinky
```

Si vous ne souhaitez pas modifier le chemin d'accès à l'exemple, vous devez exécuter cette commande à l'intérieur du dossier (où `(.venv)` indique que vous vous trouvez dans un environnement virtuel) :

```
(.venv) <my_path_to_zephyr>\zephyrproject\zephyr
```

Si l'exemple se construit sans erreur, vous pouvez le télécharger sur la carte à l'aide de la commande :

```
west flash
```

et la LED « default » de la carte commence à clignoter à une fréquence de 0,5 Hz.

Comme mentionné précédemment, le clignotement peut nécessiter un outil de programmation externe, tel qu'un adaptateur J-Link ou un autre programmateur (compatible JTAG ou SWD) (**figure 3**) et le logiciel pilote pour celui-ci doit être accessible (c'est-à-dire dans le chemin de recherche, *%PATH%* sous Windows). Si ce n'est pas le cas, vous en serez informé par un message (souvent long et énigmatique).

Sur la BBC micro:bit V2, la première fois, j'ai dû copier le fichier HEX manuellement sur la carte en utilisant la procédure de programmation standard du micro:bit. Ensuite, la commande *flash* a fonctionné correctement. L'exécutable *zephyr.hex* se trouve dans *zephyrproject\zephyr\build\zephyr*.

La commande *flash* par défaut pour les cartes Arduino, UNO R4 Minima et GIGA R1 WIFI, nécessite que l'utilitaire de programmation *dfu-util* soit dans le chemin de recherche du système d'exploitation (avant d'activer l'environnement virtuel, si vous en utilisez un). Cet utilitaire est inclus dans l'EDI Arduino. Cependant, c'est à vous de déterminer où il se trouve exactement sur votre ordinateur (par défaut dans *%HOMEPATH%\AppData\Local\Arduino15\packages\arduino\tools\dfu-util\<votre version Arduino la plus récente>*). La carte doit également être mise en mode DFU. Pour ce faire, appuyez deux fois sur le bouton de réinitialisation. Lorsque la LED commence à « pulser » ou à « respirer », vous pouvez flasher le programme.

Compatibilité avec Blinky

Vous avez peut-être remarqué que j'ai suggéré l'Arduino UNO R4 Minima comme carte à utiliser pour l'exemple Blinky et non la BBC micro:bit dont j'étais si enthousiaste plus tôt. La raison est que, bien qu'elle ait 25 LED (sans compter l'indicateur de mise sous tension), elle n'a pas de LED compatible avec l'exemple Blinky. L'ESP Wroom 32 n'en a pas non plus, mais la R4 Minima en a une.

La GIGA R1 est également compatible avec Blinky. Le microcontrôleur de cette carte possède deux cœurs (Cortex-M7 et -M4) et Zephyr vous permet de choisir lequel utiliser en sélectionnant soit `arduino_giga_r1_m4` soit `arduino_giga_r1_m7` pour la commande de construction. Vous pouvez montrer que les cœurs sont effectivement indépendants en flashant l'exemple Blinky deux fois, une fois pour le -M4 et une autre pour le -M7. La GIGA possède une LED RGB et Blinky utilisera des couleurs différentes pour chaque cœur : bleu pour le M4 et rouge pour le M7. Pour rendre les deux Blinky plus distincts, vous pouvez changer le taux de clignotement de l'un d'entre eux (dans *samples\basic\blinky\src\main.c*, changez la valeur de `SLEEP_TIME_MS`).

Hello World

Pour les cartes sans Blinky LED, il y a l'exemple *hello_world* qui émet une chaîne de caractères sur le port série.

```
west build -p always -b <my_board> samples/hello_world
west flash
```

Cet exemple fonctionne avec le BBC micro:bit et le module ESP-WROOM-32. Pour voir la chaîne de sortie, ouvrez un

programme de terminal série sur votre ordinateur. Le débit est généralement de 115 200 bauds (115200,n,8,1). Il se peut que vous deviez d'abord réinitialiser la carte, car le message n'est envoyé qu'une seule fois, et vous l'avez peut-être manqué (**figure 4**).
Sur la R4 Minima et la GIGA R1, la sortie série est sur la pin 1 et non, comme vous pouvez naïvement vous y attendre, sur le port USB-C comme ce serait le cas sur l'EDI Arduino. Ceci est dû au fait que le port USB est un périphérique du microcontrôleur et non une puce séparée. Zephyr étant un système d'exploitation modulaire et évolutif, le support USB, comme tout autre module ou périphérique, doit être explicitement activé pour le projet avant de pouvoir être utilisé. Cela se fait dans les fichiers de configuration du projet. Nous y reviendrons plus tard.
Pour les cartes sans convertisseur série-USB intégré, vous devez trouver le port série (généralement le port 0 au cas où le microcontrôleur en aurait plusieurs) et le connecter à votre ordinateur par le biais d'un convertisseur série-USB externe.

Pour aller un peu plus loin

Si vous avez réussi à faire fonctionner les exemples Blinky et *hello_world* sur votre carte, vous êtes en bonne position pour créer une application fonctionnant sur Zephyr. Si un seul est fonctionnel, et que vous souhaitez que l'autre le soit aussi, les choses sont un peu plus compliquées.
J'ai choisi la BBC micro:bit pour les expériences Zephyr, même si elle n'est pas compatible avec l'exemple Blinky. Ce n'est pas vraiment un problème, car la carte est livrée avec quelques exemples (dans le sous-dossier *bbc_microbit* du dossier *samples\boards*), dont l'un (*display*) est beaucoup plus joli qu'un Blinky à une seule diode électroluminescente. Il y a aussi des exemples pour d'autres cartes, mais très peu par rapport aux plus de 600 cartes supportées (même pas 5%). De plus, la plupart de ces exemples concernent des cas d'utilisation avancés ou obscurs.
Lorsque vous essayez de construire Blinky pour la BBC micro:bit (ou l'ESP-WROOM-32 ou toute autre carte incompatible), vous obtenez un message d'erreur incompréhensible. Ce qu'il essaie de vous dire, c'est que `led0` est une entité inconnue. C'est la LED Blinky par défaut (un peu comme `LED_BUILTIN` sur Arduino). Comme la micro:bit possède un port d'extension sur lequel vous pouvez connecter des LED et d'autres choses, essayons de définir l'une des broches de ce port comme `led0`.
Avant cela, faites une copie de sauvegarde du dossier *samples/basic/blinky* ou créez une copie avec un nouveau nom et utilisez-la à la place. Dans ce qui suit, le dossier *samples/basic/blinky* est utilisé.

L'arbre des périphériques

La définition d'une `led0` nous amène à l'arbre des périphériques, déjà brièvement mentionné. Il est contenu dans un ou plusieurs fichiers texte et liste les périphériques et la mémoire disponibles sur une carte ou dans un contrôleur. Dans Zephyr, ces fichiers ont le format *.dts* ou *.dtsi* (« I » pour *include*) et les fichiers peuvent inclure d'autres fichiers. Les fichiers *.dtsi* du processeur se trouvent dans le dossier *dts*, les fichiers *.dts* et *.dtsi* de la carte se trouvent dans le dossier *boards*. Les deux dossiers sont organisés par architecture de processeur.
Pour mieux visualiser les fichiers DTS(I), vous pouvez utiliser le plugin DeviceTree pour Visual Studio Code [4]. Il permet la mise en

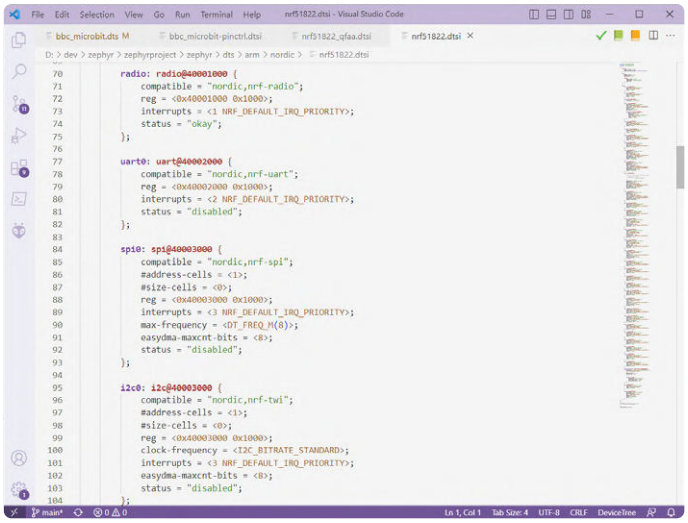


Figure 5. Un extrait du fichier nrf51822.dtsi. La vue zoomée à droite montre la longueur de ce fichier.

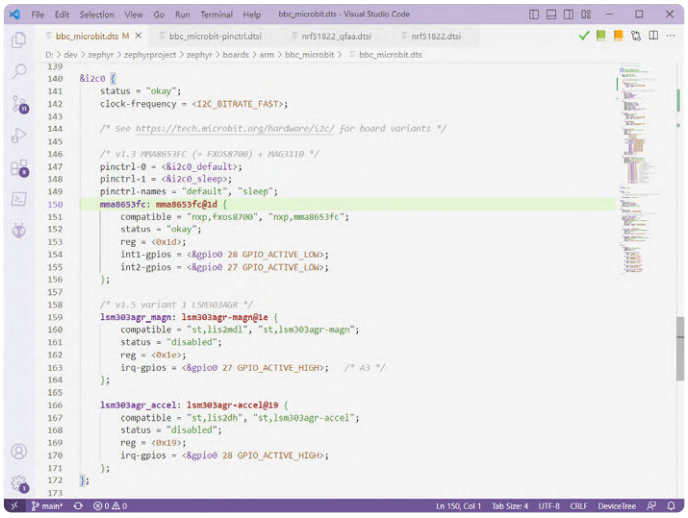


Figure 6. Cette section de l'arbre des périphériques représente le bus I²C de la carte BBC micro:bit, et non de son processeur. Tiré du fichier bbc_microbit.dts.

évidence de la syntaxe et le pliage du code, ce qui rend les fichiers plus faciles à lire (les fichiers DTS utilisent une syntaxe de type langage C). La **figure 5** montre un extrait du fichier *.dtsi* pour le nRF51822 qui est au cœur de la carte BBC micro:bit V1. Ce fichier est inclus dans le fichier DTS de la carte. À titre d'exemple, notez que le statut de l'`uart0` est réglé sur `disabled`. Ce statut est surchargé dans le fichier DTS de la carte, où il est réglé sur `okay`, ce qui signifie qu'il peut être utilisé. Il en va de même pour `gpio0` et `i2c0`.

I²C dans l'arborescence

Un autre extrait du fichier .dts pour la BBC micro:bit est visible dans la **figure 6**. Il montre l'arbre des périphériques pour le bus I²C. La micro:bit a un ou deux capteurs attachés au bus (selon la variante de la carte micro:bit V1) et ils sont représentés dans l'arbre par `mma8653fc` et `lsm303agr` (ce dernier comprend deux capteurs, c'est pourquoi il apparaît deux fois). Le premier a l'état `okay`, tandis que les deux autres sont `disabled`. Ceci est correct pour ma variante de carte, qui est un échantillon de la toute première génération de micro:bit V1.
Comme le montre l'extrait, ce capteur est compatible avec le FXOS8700 et le MMA8653FC, son adresse sur le bus I²C est 0x1d, et deux signaux `int` (*interrupt*) sont déclarés qui sont connectés à

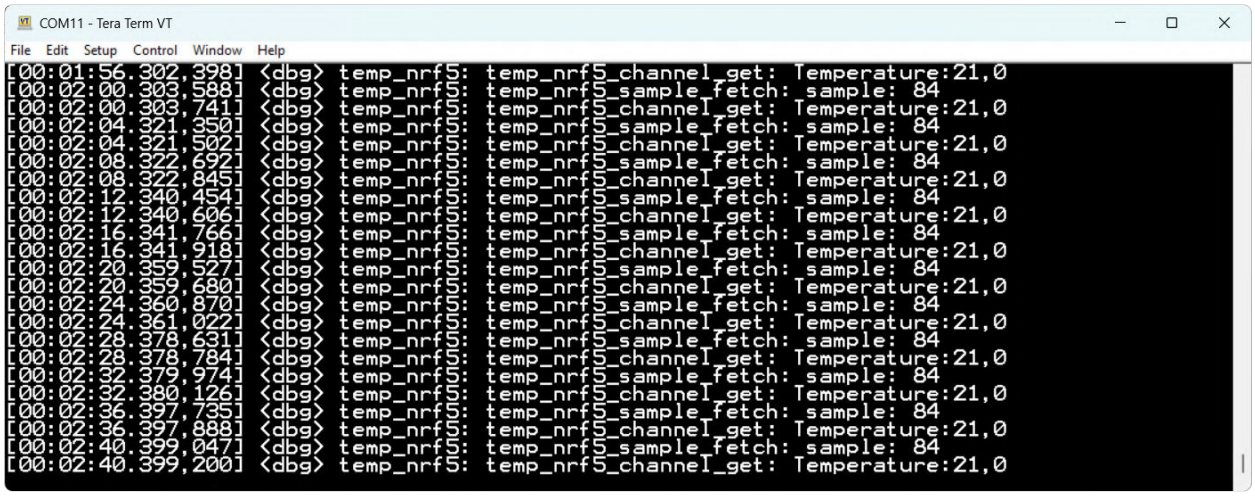


Figure 7. La sortie de la démo samples/sensor/fxos8700 fonctionnant sur la carte BBC micro:bit.

deux broches GPIO : 27 et 28. Si vous voulez l'essayer, un programme de démonstration est disponible :

```
west build -p always -b bbc_microbit samples/sensor/
fxos8700
west flash
```

Notez que cela ne fonctionnera pas pour la BBC micro:bit V2, car elle a un capteur différent dans son arbre de périphériques.
La sortie de la démo est montrée dans la **figure 7**, mais nous nous éloignons du sujet.

Superposition de l'arbre des périphériques

Revenons à notre LED, `led0`. L'arbre des périphériques de la carte ne mentionne pas `led0`, comme prévu, et nous devons donc l'ajouter. Nous pourrions l'insérer directement dans le fichier de l'arbre des périphériques de la carte, mais ce serait incorrect, car elle n'a pas de `led0`. La bonne façon d'étendre l'arborescence d'un périphérique est d'ajouter un fichier de superposition (*overlay* en anglais). Le contenu d'un fichier de superposition y est ainsi ajouté. Les sections qui existent dans l'arborescence sont étendues (dans le cas d'un nouvel élément) ou écrasées (dans le cas où l'élément existe déjà dans l'arborescence) ; de nouvelles sections sont ajoutées.
Les fichiers de superposition doivent être placées à l'intérieur du dossier du projet, dans un sous-dossier nommé boards. Lorsque ce sous-dossier est présent, le processus de construction y recherchera un fichier portant le nom *<ma_carte>.overlay*. Dans mon cas, le nom du fichier est *bbc_microbit.overlay* (*bbc_microbit_v2.overlay* pour les utilisateurs de la V2). La **figure 8** montre le contenu du fichier.

Ajouter une LED clignotante

Zephyr a un mot-clé spécial pour l'arbre des périphériques pour les LED, qui est `leds`, donc nous créons un nœud (branche) pour cela (vous pouvez l'appeler comme vous voulez, mais tenez-vous en à `leds` s'il est supposé être superposé à un nœud `leds` existant). Il doit être ajouté à la racine de l'arbre ; c'est pourquoi une barre oblique '/' est placée devant, car elle signifie racine en langage DT. La ligne suivante indique que cette branche est compatible avec le pilote `gpio-leds` intégré à Zephyr (vous pouvez trouver l'interface de ce pilote dans *zephyr\include\zephyr\drivers\led.h*).

Nœuds fils

Vient ensuite la liste des nœuds fils des LED. Comme je n'ai qu'une LED, il n'y a qu'un seul nœud fils, `led_0`, que j'ai étiqueté `led0`. L'étiquetage d'un nœud est facultatif, mais il permet de le référencer ailleurs dans l'arbre, ce que nous ferons quelques lignes plus bas. De plus, ils peuvent être utilisés par (le développeur de) l'application pour accéder aux nœuds et à leurs propriétés.
Un nœud fils doit spécifier les propriétés de l'appareil. Dans le cas d'une LED, seule la broche GPIO est une propriété obligatoire, mais la propriété optionnelle nommée `label` peut être ajoutée. Ces étiquettes peuvent être utilisées pour fournir de la documentation ou des informations lisibles par l'utilisateur. Elles n'ont pas d'autre utilité.
Comme broche GPIO, j'ai choisi `1`, qui correspond au grand trou `2` sur le connecteur d'extension de la micro:bit. Si vous avez une BBC micro:bit V2, utilisez `4` comme broche GPIO (au lieu de `1`).

Créer un alias

L'étape suivante est nécessaire, car l'exemple Blinky l'attend. Elle consiste à créer l'alias `led0` pour notre LED. On pourrait penser qu'étiqueter le nœud fils aurait été suffisant, mais ce n'est pas le cas, car Blinky utilise la macro `DT_ALIAS` pour y accéder. Par conséquent, nous devons fournir quelque chose que cette macro peut assimiler, ce qui se trouve être un alias. Il se trouve à l'intérieur du bloc alias. Si Blinky avait utilisé la macro `DT_NODELABEL` à la place,

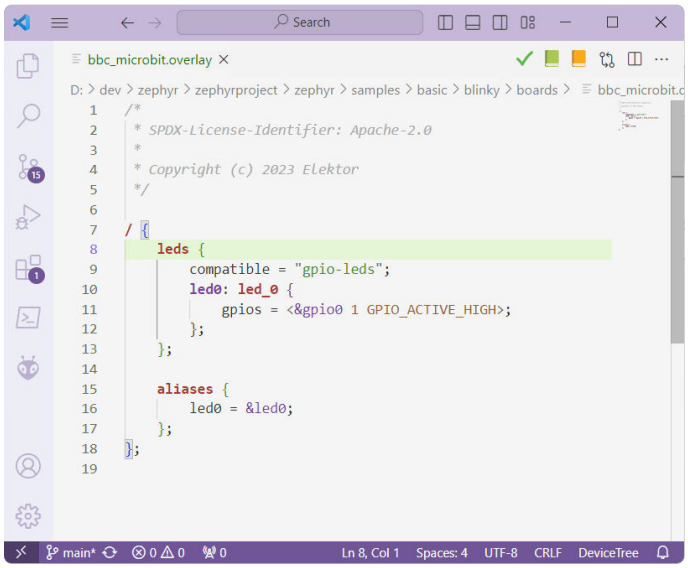


Figure 8. Ce fichier de superposition d'abre de périphériques rend la BBC micro:bit V1 compatible avec l'exemple Blinky.

un alias aurait été superflu, car `DT_NODELABEL` récupère directement l'étiquette du nœud fils `led0`. Je sais que le fait que les étiquettes et les alias portent le même nom est un peu déroutant, mais c'est nécessaire pour mon explication.

Macros Zephyr

Bien que les macros soient mal vues dans la programmation C/C++, Zephyr les utilise beaucoup. Celles telles que `DT_ALIAS` et `DT_NODELABEL` permettent à l'application et aux outils de configuration du projet d'extraire des informations de l'arbre des périphériques, et elles sont nombreuses. Vous trouverez leur description dans le manuel Zephyr, au chapitre « Devicetree API » [5].

Fait amusant : de nombreuses (toutes ?) macros Zephyr s'attendent à ce que leurs arguments soient en minuscules, les caractères qui ne sont pas des lettres ('a' à 'z') ou des chiffres ('0' à '9') étant remplacés par des tirets bas ('_'). C'est ce qu'on appelle la compatibilité *lowercase-and-underscores* (minuscules et tirets bas). Par exemple, imaginons que j'ai étiqueté le nœud fils LED d'avant `LED-0` au lieu de `led0`. L'argument à utiliser avec `DT_NODELABEL` aurait alors été led_0, c'est-à-dire `DT_NODELABEL`(led_0), car '-' n'est ni une lettre ni un chiffre, et les lettres doivent être en minuscules. En d'autres termes, pour le développeur d'applications utilisant des macros d'arborescence, le caractère de soulignement est un joker. Ainsi, `led_0` dans l'application peut faire référence à `led_0`, `led-0`, `Led_0`, `LED-0` et `ledé0` (et toutes les autres variantes possibles) dans l'arbre des périphériques. En gardant cela à l'esprit, il est fortement recommandé de lire attentivement la documentation des macros Zephyr.

Tu ne feras pas d'erreurs

Notez que faire des erreurs dans l'arbre des périphériques est puni par le compilateur qui vous crie *FATAL ERROR* sans fournir d'autres informations.

Constructions parfaites

Lorsque vous jouez avec l'arbre des périphériques ou avec votre application, il est probable que vous recompilez souvent votre projet. Pour accélérer un peu les choses, supprimez l'option `-p always`('p' de *pristine*) de la commande de compilation. Cela l'empêchera de tout reconstruire à partir de zéro. Si, en revanche, vous essayez de nombreux exemples différents les uns après les autres, gardez-le, car cela vous évitera une erreur ennuyeuse concernant le dossier de compilation qui n'est pas destiné à votre projet.

Notez que la commande *flash* déclenche également la dernière commande *build*, il suffit donc d'exécuter la commande *flash* à chaque fois que vous modifiez quelque chose.

Utiliser un pilote de périphérique

L'exemple Blinky appelle la fonction `gpio_pin_toggle_dt()` pour basculer l'état de la LED. Il s'agit d'une fonction du pilote GPIO. Bien sûr, c'est tout à fait possible, mais Zephyr inclut également une collection de pilotes de LED. Leur utilisation ne rend pas seulement le programme plus lisible, mais aussi plus flexible et portable, car un pilote de LED peut être remplacé par un autre sans modifier l'application elle-même. C'est là que l'évolutivité et la modularité de Zephyr entrent en jeu.

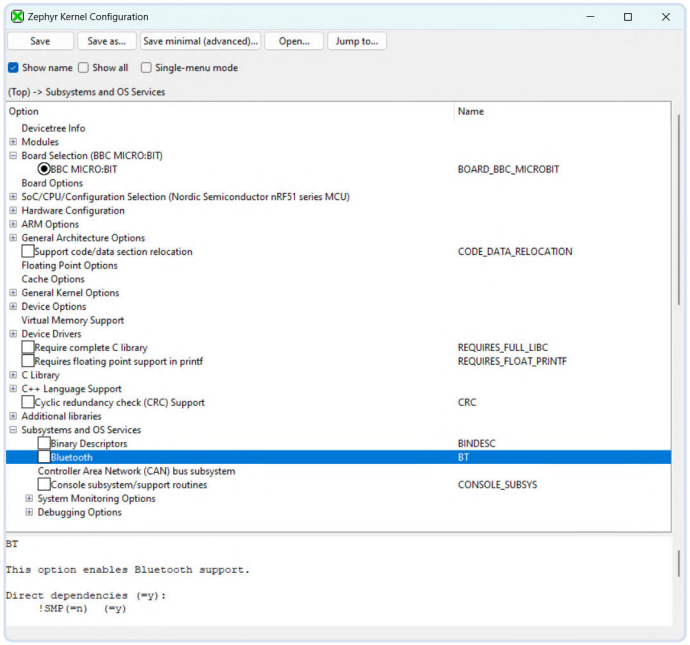


Figure 9. La GUI de l'outil de configuration de projet Kconfig. Les options sont nombreuses.

Kconfig a une interface graphique

L'intégration d'un pilote de périphérique LED dans notre programme nécessite quelques étapes. Tout d'abord, le projet doit être reconfiguré afin de l'inclure. La configuration du projet est gérée par Kconfig, le système de configuration du noyau au moment de la compilation, également utilisé par Linux. Il y a plusieurs façons d'interagir avec lui, et l'une d'entre elles est une interface utilisateur graphique (GUI). Dans Zephyr, vous l'ouvrez comme suit :

```
west build -t guiconfig
```

La GUI met un certain temps à s'ouvrir, mais lorsqu'elle le fait, elle ressemble à la **figure 9**. Elle affiche de nombreuses informations sur le projet en cours de développement. Notez la partie projet en cours de développement. Pour s'assurer que Kconfig travaille bien sur votre projet, faites une construction vierge (avec l'option `-p always`) de votre projet avant de lancer la GUI de Kconfig.

Tant d'options de configuration…

Prenez le temps d'explorer l'arbre de configuration. Déployez les branches en cliquant sur les symboles '+'. Mettez les options en surbrillance en cliquant dessus. Cela affichera des informations dans le volet inférieur. Notez que la prise en charge de la virgule flottante pour `printf()` est une option de configuration, tout comme la prise en charge du langage C++. De même, sous *Build and Link Features*, vous pouvez trouver des options d'optimisation du compilateur.

Il existe des tonnes d'options de configuration. Celle qui nous intéresse se trouve dans la branche *Device Drivers*. Déployez-la et faites défiler vers le bas tout en regardant tout ce qui est disponible. Le pilote de LED se trouve à peu près à mi-chemin : *Light-Emitting Diode* (LED) Drivers (pilotes de diodes électroluminescentes). Cochez la case. Laissez les options de sa sous-branche sur leurs valeurs par

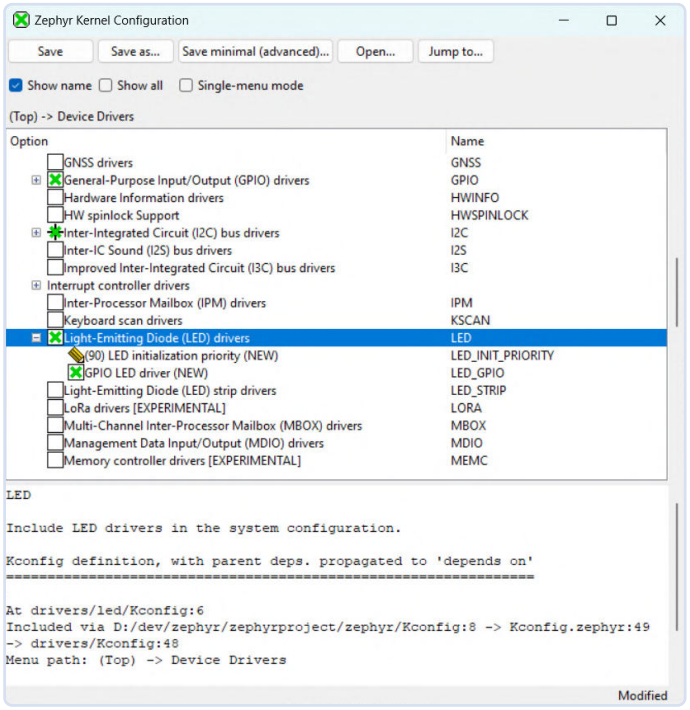


Figure 10. Sélectionnez Light-Emitting Diode (LED) Drivers et laissez les valeurs enfant telles quelles.

```c
#include <zephyr/kernel.h>
#include <zephyr/device.h>
#include <zephyr/drivers/led.h>

#define SLEEP_TIME_MS   (1000) /* 1000 msec = 1 sec */

int main(void)
{
    printf("\nBlinky with LED device driver\n");

    const struct device *const led_device = DEVICE_DT_GET_ANY(gpio_leds);
    if (!led_device)
    {
        printf("No device with compatible 'gpio-leds' found\n");
        return 0;
    }
    else if (!device_is_ready(led_device))
    {
        printf("LED device %s not ready\n", led_device->name);
        return 0;
    }

    while (1)
    {
        int result = led_on(led_device,0);
        if (result<0)
        {
            printf("led_on failed\n");
            return 0;
        }
        k_msleep(SLEEP_TIME_MS);

        result = led_off(led_device,0);
        if (result<0)
        {
            printf("led_off failed\n");
            return 0;
        }
        k_msleep(SLEEP_TIME_MS);
    }

    return 0;
}
```

Figure 11. Le programme Blinky réécrit pour être utilisé avec un pilote de périphérique LED. Notez qu'il n'y a pas de code spécifique à la carte. Ce programme devrait fonctionner sur n'importe quelle carte ayant un pilote `gpio_leds` *(ou* `gpio-leds`*) listé dans son arbre de périphériques.*

défaut (**figure 10**). Cliquez sur le bouton *Save* et notez le chemin du fichier de configuration imprimé en bas de la fenêtre. Il est instructif d'inspecter ce fichier, juste pour voir ce qu'il contient (beaucoup de choses). Fermer l'interface graphique.
A partir de maintenant, ne spécifiez plus le `-p always` dans les commandes de compilation, car cela annulerait les changements que vous avez faits ci-dessus. Je montrerai plus tard comment rendre le changement de configuration permanent.

Blinky avec le pilote de périphérique LED

Nous pouvons maintenant écrire le nouveau programme Blinky, voir **figure 11**. Il commence par inclure les fichiers d'en-tête du périphérique et du pilote de LED. Ensuite, dans le programme principal, nous employons la macro `DEVICE_DT_GET_ANY` pour obtenir une référence de périphérique pour la LED à partir de l'arbre des périphériques. Notez que l'argument `gpio_leds` de la macro est compatible avec les minuscules et les tirets bas, de sorte qu'il correspondra à la valeur `gpio-leds` de la propriété compatible du nœud leds dans l'arbre des périphériques (comme expliqué ci-dessus). S'il n'en trouve pas parce que vous avez fait une faute de frappe ou autre, le nouveau Blinky affichera le message `No device with compatible gpio-leds found`. Ce message sera également déclenché lorsque la propriété d'état d'un périphérique est réglée sur `disabled`.
Il faut un peu de temps pour s'habituer à l'utilisation du mot `compatible` comme substantif dans Zephyr. Ainsi, le message d'erreur ci-dessus ne signifie pas qu'il n'y a pas de périphérique compatible, mais qu'il n'y a pas de périphérique ayant une propriété nommée `compatible` avec la valeur `gpio-leds` (ni d'ailleurs avec `gpio_leds`, rappelons que le tiret bas '_' remplace n'importe quel caractère sauf 'a' à 'z' et '0' à '9').
Une deuxième vérification permet de s'assurer que le périphérique s'est initialisé correctement. En supposant que c'est le cas, nous continuons.
Dans la boucle infinie `while()`, la LED est allumée et éteinte à l'aide des commandes `led_on` et `led_off` fournies par le pilote [6]. L'argument `0` signifie le premier (et seul) périphérique trouvé par la macro `DEVICE_DT_GET_ANY`, qui est `led0`.

Vérifier les valeurs de retour

Comme nous utilisons un pilote de périphérique au lieu de basculer directement une broche GPIO au niveau du registre matériel, il est bon de vérifier les valeurs de retour de tous les appels de fonction du pilote, car ils peuvent échouer pour une raison ou une autre. Un pilote doit fournir certaines fonctions et certains rappels, mais il peut aussi avoir des caractéristiques optionnelles. Un pilote de LED, par exemple, doit implémenter `led_on` et `led_off`, mais `led_blink` est optionnel. Si vous appelez `led_blink` dans notre projet, il ne se passera rien parce qu'il n'est pas implémenté. Elle existe, mais elle est vide. La valeur de retour vous le montrera. En général, c'est une bonne pratique de programmation que de vérifier la valeur de retour de chaque appel de fonction.
Construisez et téléchargez le programme comme suit (notez l'absence du drapeau `-p always`).

```
west build -b bbc_microbit samples/basic/blinky
west flash
```

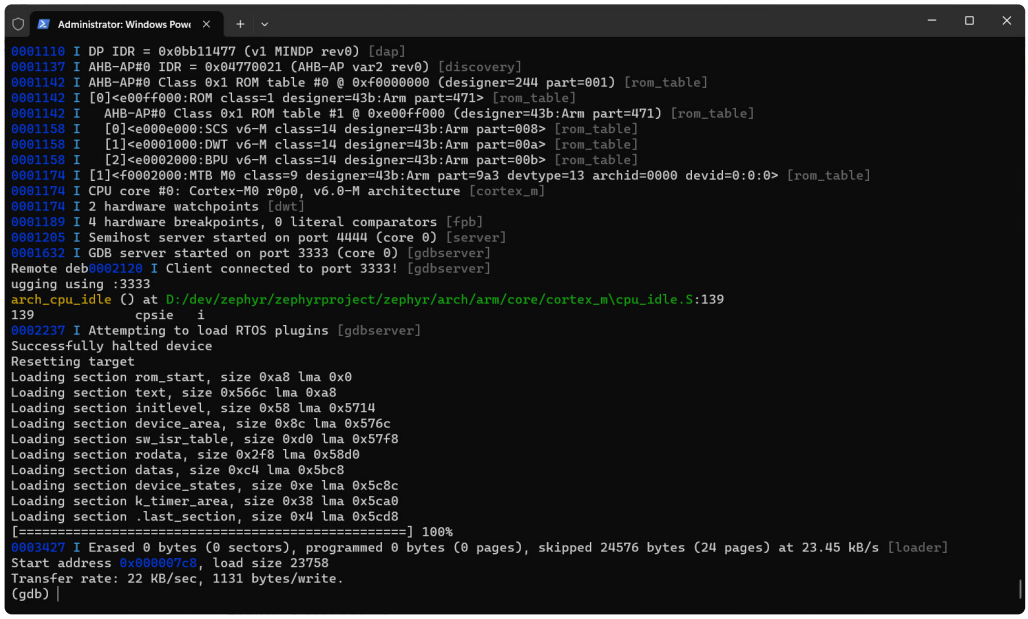


Figure 12. La BBC micro:bit supporte le débogage avec gdb sans avoir besoin d'outils supplémentaires ou quoi que ce soit.

Configurer le projet

Si la LED a commencé à clignoter à 0,5 Hz, nous avons un programme qui fonctionne. Pour qu'il le reste, nous devons rendre la configuration actuelle permanente. Pour ce faire, ouvrez le fichier *prj.conf* dans le dossier de notre Blinky et ajoutez la ligne (contrairement aux fichiers d'arbre de périphériques qui utilisent la syntaxe du langage C, les fichiers de configuration Kconfig utilisent la syntaxe du langage Python) :

```
CONFIG_LED=y
```

Pour vérifier qu'il fonctionne, exécutez une compilation *pristine* de votre projet et téléchargez l'exécutable sur la carte.

Débogage

Si votre carte le permet (comme la BBC micro:bit) ou si vous disposez d'un outil de débogage approprié, vous pouvez déboguer l'application avec

```
west debug
```

Ceci lancera un serveur gdb et ouvrira un terminal (**figure 12**). Consultez internet pour apprendre à travailler avec gdb car cela sort du cadre de cet article.

Zephyr contre Arduino

Maintenant que vous avez vu comment démarrer avec le système d'exploitation Zephyr, vous vous demandez peut-être pourquoi vous l'utiliseriez. N'est-il pas plus simple d'utiliser Arduino ? Comme Zephyr, Arduino supporte plusieurs architectures de processeurs et des centaines de cartes. Des milliers de pilotes et de bibliothèques ont été écrits pour Arduino. Si une application ou une bibliothèque utilise l'API Arduino Core, elle peut être facilement portée sur n'importe quelle autre plateforme supportée avec des périphériques similaires, tout comme une application Zephyr. Les deux sont des logiciels libres. Et donc ?

Zephyr est conçu comme un RTOS robuste de qualité industrielle, doté de fonctions telles que la planification des tâches, la gestion de la mémoire et les pilotes de périphériques. En outre, Zephyr est conçu pour s'adapter à un large éventail de complexités de projets, des petits appareils IdO aux systèmes embarqués complexes. Il offre une plus grande flexibilité, mais nécessite une compréhension plus approfondie du développement embarqué.

Arduino, bien qu'offrant certaines capacités en temps réel, n'est pas un RTOS mais un cadre de travail pour les applications à un seul fil, axé sur la simplicité et la facilité d'utilisation. Arduino fait abstraction de nombreux détails de bas niveau, ce qui le rend accessible aux débutants. Cependant, il peut être utilisé au-dessus d'un RTOS tel que Mbed OS pour des applications plus complexes. Des travaux visant à faire fonctionner l'API Arduino Core sur Zephyr sont en cours ([7]).

C'est à vous de décider si vous avez besoin de Zephyr pour votre prochain projet ou non. Vous pouvez au moins l'essayer, car il a fière allure sur le CV de tout développeur de systèmes embarqués.

Autres lectures

C'est tout pour l'instant. Comme vous l'aurez constaté, le système d'exploitation Zephyr est compliqué et la courbe d'apprentissage est quelque peu abrupte. Dans cet article, j'ai essayé de rendre ce parcours un peu plus facile. Cependant, il y a beaucoup, beaucoup plus à dire et à apprendre sur Zephyr, alors ne pensez pas maintenant que vous savez tout. Les références [8] et [9] fournissent des liens vers deux sujets qui pourraient vous intéresser.

VF : Maxime Valens — 230713-04

Questions ou commentaires ?

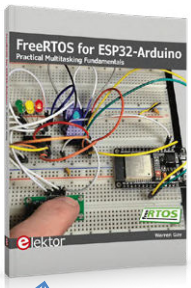
Envoyez un courriel à l'auteur (clemens.valens@elektor.com) ou contactez Elektor (redaction@elektor.fr).

À propos de l'auteur

Après une carrière dans l'électronique marine et industrielle, Clemens Valens a commencé à travailler pour Elektor en 2008 en tant que rédacteur en chef d'Elektor France. Il a occupé différents postes depuis lors et a récemment rejoint le département de développement des produits. Ses principaux centres d'intérêt sont le traitement du signal et la génération de sons.

Produits

- **BBC micro:bit V2**
  www.elektor.fr/19488
- **Get Started with the NXP i.MX RT1010 Development Bundle**
  www.elektor.fr/20699
- **Warren Gay, *FreeRTOS for ESP32-Arduino*, Elektor 2020**
  www.elektor.fr/19341



LIENS

[1] À propos du projet Zephyr - Un projet de la Fondation Linux : https://zephyrproject.org/learn-about
[2] Spécifications de l'arbre des périphériques : https://devicetree.org/specifications
[3] Démarrer avec Zephyr : https://docs.zephyrproject.org/latest/develop/getting_started/index.html
[4] Plugin de mise en évidence de la syntaxe Devicetree pour Visual Code Studio: https://github.com/plorefice/vscode-devicetree
[5] Devicetree API : https://docs.zephyrproject.org/latest/build/dts/api/api.html
[6] Le périphérique LED : https://docs.zephyrproject.org/latest/hardware/peripherals/led.html
[7] Exécuter Arduino sur Zephyr : https://dhruvag2000.github.io/Blog-GSoC22
[8] Voir le livre blanc d'Eli Hughes : « From Hardware Concept to Zephyr Bring Up » : https://zephyrproject.org/white-papers
[9] Sortie de la console série USB dans une application RTOS Zephyr : https://www.gnd.io/zephyr-console-output

Lauréat du prix Éthique

Dialogue avec Alexander Gerfer, Directeur Technique chez Würth Elektronik eiSos sur la promotion du comportement respectueux et innovant

Questions par Shenja Panik (Ethics in Electronics / Elektor)

Directeur Technique (CTO) chez Würth Elektronik eiSos, Alexander Gerfer partage son engagement dans l'innovation et l'éthique dans une interview exclusive avec Shenja Panik. L'interview aborde les initiatives du Centre d'Innovation de Haute Technologie sur la responsabilité environnementale, l'équité et la prédictibilité, et souligne comment l'éthique est intégrée avec succès dans les affaires de Würth Elektronik.

Shenja Panik : Félicitations, M. Gerfer, pour avoir gagné le prix 2023 de l'Éthique en Électronique pour votre infatigable engagement dans le développement durable, la technologie verte et l'agriculture verticale. Merci de nous avoir invité aujourd'hui dans votre Centre d'Innovation de Haute Technologie à Munich et d'avoir pris le temps de répondre à quelques questions. Pouvez-vous, s'il vous plaît, nous décrire votre rôle dans le domaine de l'éthique en électronique ?

Alexander Gerfer : Pour commencer, il est important de jeter un œil sur mon passé : en grandissant dans une ferme avec des vaches et des poules, vous devez prendre très tôt des responsabilités. Un autre aspect crucial se trouve dans un dicton allemand, « *Was du nicht willst, das man dir tut, das füg auch keinem andern zu* « (ce qui veut dire : Ne fais pas à autrui ce que tu n'aimerais pas qu'on te fasse). Voilà les bases de mon éducation. Notre objectif chez Würth Elektronik a toujours été nos clients. Nous ne faisons pas de différence entre nos clients en fonction de la taille de leur entreprise ou de leur importance. Nous conservons une très bonne relation de coopération avec chacun d'entre eux. Nous leur faisons profiter de notre expertise par des conseils, des conceptions d'échantillons, des présentations en ligne, des kits de développement et des outils logiciels qui aident dans le dimensionnement. Nous ne facturons rien de tout cela, pas plus que les échantillons de laboratoire de nos composants. Chacun est également important à nos yeux. Nous aidons toute personne qui nécessite de la connaissance sur une sélection et une application. Nous fournissons et assistons tout le monde, depuis les start-up jusqu'aux entreprises bien établies, depuis ceux qui commandent de petites quantités jusqu'aux acheteurs de grandes séries. Nous n'avons pas de minimum de commande. C'est basé sur notre culture d'entreprise. Dans une start-up, vous pouvez échouer pendant la conception d'un produit à cause d'un composant valant un centime. Je le sais d'après ma propre expérience. Voilà pourquoi nous faisons le maximum pour toujours résoudre rapidement les problèmes — et nous ne donnons pas notre préférence à celui qui nous rapportera le plus.

Shenja Panik : Toutes les start-up n'ont pas les mêmes chances et la manière dont Würth Elektronik

travaille activement pour égaliser le terrain de jeu est extraordinaire. Quelles sont les autres initiatives et stratégies implémentées à Munich pour promouvoir une économie éthique et écologique ?

Alexander Gerfer : Dans notre nouveau Centre d'Innovation de Haute Technologie à Munich, nous testons les prototypes d'appareils et ensembles électroniques pour leur compatibilité électromagnétique et nous donnons des suggestions ciblées pour leur amélioration. Après tout, la certification EMC est souvent un obstacle majeur pour les concepteurs. L'équité, la prédictibilité, l'égalité, le soutien mutuel — voilà les principes de la gestion d'entreprise et la manière dont les employés se traitent les uns les autres. En tant que partenaire du groupe Würth, nous contribuons aussi pro-activement à la préservation de notre environnement, à la fois par nos produits et par des usages économiques respectueux de l'environnement. Aujourd'hui, il est plus que jamais important de prendre soin de l'environnement.

Nous réfléchissons toujours un pas en avant. Prenez, par exemple, l'agriculture verticale : cultiver de la nourriture dans une serre à plusieurs niveaux offre déjà une contribution à la production locale de nourriture de haute qualité. Néanmoins, cette contribution a besoin de se développer de façon significative. D'après les estimations de la FAO (Organisation de l'Agriculture et de la Nourriture des Nations Unies), en 2050, 10 milliards de personnes vivront sur cette planète. Ceci nécessitera une augmentation de 50 % de la production globale de l'agriculture d'ici 2050. Et pourtant, la surface agricole a diminué durant les dernières décennies, de pratiquement 40% de la surface mondiale en 1991, à seulement 37% en 2018. Les plantes vont littéralement devoir pousser vers le ciel si nous voulons assurer une bonne nourriture à tout le monde. Les changements climatiques et démographiques posent d'énormes défis et il faut que nous travaillions sur des solutions maintenant — même si elles sont initialement perçues comme inefficaces et «trop chères».

Il y a un manque de bonnes idées pour rendre le monde meilleur. Cependant, chaque idée significative aujourd'hui nécessite de l'électronique pour sa mise en œuvre. C'est là que nous contribuons en tant que fournisseur fiable mais plus important encore, comme une main tendue pour les petites et moyennes entreprises et les start-up ambitieuses. Bien que, chez Würth Elektronik, nous devions générer des profits en tant qu'entreprise, l'accent n'est pas sur le profit à court terme mais sur la pertinence du projet.

Figure 1. Remise du prix Ethics-in-Electronics 2023 au lauréat Alexander Gerfer.

Shenja Panik : Que représente ce prix pour vous et votre réseau ?

Alexander Gerfer : Pour moi personnellement, le prix est un honneur très spécial. Nous le partageons également avec l'équipe parce que je ne fais pas ça tout seul. C'est un honneur pour toute l'équipe qui travaille sur ces projets — mais par dessus tout, c'est un encouragement. Ce serait probablement surestimer de déclarer que le monde ne s'est amélioré que récemment. Malheureusement, des signes de confrontation persistent dans beaucoup de domaines, et tristement, même les principes éthiques les plus fondamentaux sont ignorés. Il est encore plus important que nous restions fidèles à nos principes, dans l'assistance altruiste, l'action commune, le traitement respectueux de chacun, la préservation de notre espace de vie et la stimulation pro-active des innovations pour le bénéfice du public. Nous faisons tous face à des défis importants. Nous avons d'énormes problèmes à résoudre et le mieux est de les traiter ensemble à partir d'aujourd'hui.

Shenja Panik : Vous nous avez parlé de votre expérience et de la façon dont elle vous a guidé jusqu'ici. Quelles sont vos principales sources d'inspiration actuelles lorsque vous prenez des décisions éthiques ?

Alexander Gerfer: Tout d'abord et avec certitude, c'est la boussole des valeurs et ma propre éthique personnelle dont j'ai hérité chez moi. On m'a inculqué un sens des responsabilités envers la nature et les êtres humains depuis tout petit. Voilà pourquoi, par exemple, le thème de la Ferme Verticale avec des LEDs me tient beaucoup à cœur.

Nos valeurs d'entreprise dans le Groupe Würth sont importantes pour nous, ce qui détermine notre conduite économique et notre attitude dans tous les domaines clés. La confiance mutuelle, la prédictibilité, l'honnêteté et la droiture, à la fois internes et externes, sont les principes fondamentaux que le Professeur Dr. h. c. mult. Reinhold Würth a formulé dans les années 1970.

Le Code de Conformité du Groupe Würth s'étend sur 32 pages, décrivant les principes généraux de comportement, les directives pour les relations avec les partenaires commerciaux, les informations pour éviter les conflits d'intérêt et la mise en application du Code de Conformité.
Avant de prendre toute décision, n'importe quel collègue ou moi-même nous demandons: ma décision ou mon action est-elle en accord avec les lois d'application ? La décision est-elle bien dans la ligne des valeurs connues et des règles de Würth ? En supposant que mes collègues et ma famille soient au courant de ma décision, aurais-je la conscience tranquille ? Si tout le monde entérinait cette décision, pourrais-je vivre avec les conséquences? Si ma décision paraissait dans les journaux le lendemain, pourrais-je la justifier ?
Comme je l'ai dit, il faut être loyal dans vos décisions, être respectueux et reconnaissant pour chaque contribution, mais il est aussi très important d'être prévisible et fiable.

Shenja Panik : Quelle résilience est nécessaire pour l'implémentation éthique et les initiatives durables dans une grande entreprise ?
Alexander Gerfer : Dans la vie de tous les jours ? Pratiquement aucune, parce que nous sélectionnons nos employés avec beaucoup de soin. Pour nous, le respect ne veut pas seulement dire adhérer à toutes les règles d'application, les lois et les principes de l'entreprise, mais également une attitude interne conforme de la part de nos employés, ce qui représente un élément crucial pour le succès commercial durable de Würth Elektronik.
Ainsi, nous comptons sur nos dirigeants, employés et partenaires commerciaux pour qu'ils garantissent, au niveau national et international, des règles et des lois valides. De plus, dans le cadre de notre catalogue de valeurs, nous nous engageons à adhérer aux principes, directives et normes clairement définis dans le Code de Conformité. Cette vision de valeurs partagées est transmise dans tous les domaines de l'entreprise et est assimilée par tous les employés.
Pendant le processus de sélection, nous évaluons les candidats non seulement sur leurs qualifications professionnelles mais aussi particulièrement sur leurs aptitudes personnelles.
Sont-ils en accord avec nos principes ? Est-ce que les futurs employés s'identifient à nos valeurs d'entreprise ou les ont-ils simplement mémorisées ? Sont-ils plus orientés vers des résultats durables que vers des chiffres trimestriels ? Voient-ils leurs collègues comme des partenaires ou de manière arrogante ? Sont-ils conscients au niveau environnemental et global ou plutôt égocentriques et opportunistes ?
Ces quelques questions parmi beaucoup d'autres doivent être résolues avant une embauche. La connaissance peut être acquise sur le tas et l'habileté peut être transmise par des collègues expérimentés. Cependant, la base indispensable est l'orientation éthique que chacun doit apporter au groupe. Ainsi il n'y pas vraiment de résilience — les gens qui travaillent ici croient dans ce que nous faisons et comment nous le faisons. Et, chaque jour, nous avons la preuve que nos employés assument leurs responsabilités avec beaucoup de compétences. Chez nous, la direction, les cadres et les employés avancent dans la même direction. Dans notre entreprise, l'éthique n'est pas imposée mais vécue.

Shenja Panik : Ce n'est pas passer d'un projet à un autre mais maintenir une philosophie d'entreprise. Êtes-vous tous impliqués dans la problématique de comment rendre le monde meilleur ?
Alexander Gerfer : Nous ne valorisons pas seulement notre propre équipe mais nous stimulons également les personnes en dehors de l'entreprise. En me basant sur ma propre expérience de start-up et de ses défis, je me suis rendu compte de l'importance de tendre la main aux développeurs dans les premières étapes, particulièrement dans le secteur du hardware qui est l'étape la plus complexe au début. À ce stade, une mauvaise décision peut saper même la meilleure idée. Donc, nous fournissons aide et soutien. Et dans le processus, nous apprenons à nous connaître... Ainsi, notre vision ultime est une compréhension totalement prévisible des besoins de nos clients. Pour nous, l'éthique et le succès commercial ne sont pas contradictoires ; ils sont plutôt liés de manière inséparable.

VF : Chris Elsass — 240013-04

Questions ou commentaires ?

Si vous avez des questions ou des commentaires sur cet article, n'hésitez pas à prendre contact avec Shenja (shenja.panik@elektor.com).

À propos de Shenja Panik

Shenja Panik est une sociologue passionnée qui fait partie de l'équipe de rédaction de EiE, spécialisée dans l'éthique de l'industrie électronique. Son travail tourne autour de l'exploration et de la promotion des considérations éthiques dans le domaine en perpétuelle évolution de la technologie.

À propos de Alexander Gerfer

Alexander Gerfer est le directeur technique de Würth Elektronik eiSos, un important fabricant européen de composants électroniques. Il fait parti de cette entreprise depuis 27 ans, à la tête de plusieurs postes, encourageant le partenariat technologique dans des domaines tels que les recettes de lumière LED, la récupération d'énergie et la mobilité électrique. Grâce à son expertise, il se passionne dans le soutien et le conseil des start-ups innovantes. Gerfer est aussi l'auteur de livres techniques, conférencier et orateur principal dans des conférences mondiales.

elektorstore

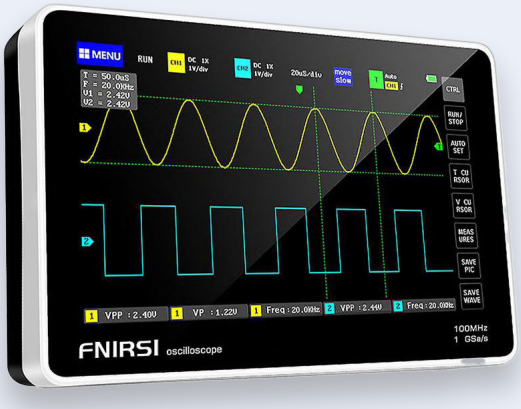
www.elektor.fr

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.

FNIRSI 1013D Oscilloscope à tablette à 2 canaux (100 MHz)

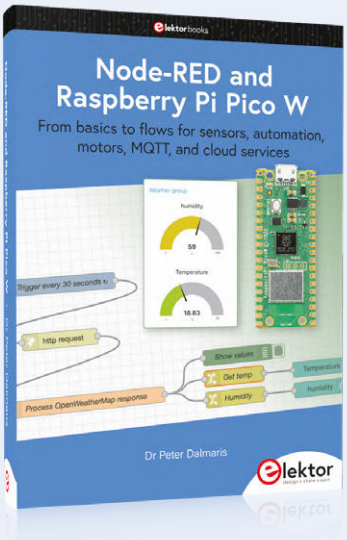


Le FNIRSI 1013D est un oscilloscope à tablette à 2 voies doté d'un écran LCD haute résolution de 7 pouces (800 x 480 pixels). L'oscilloscope a un taux d'échantillonnage en temps réel de 1 Géch/s et une bande passante analogique de 100 MHz.

Prix : 159,95 €

Prix (membres) : 143,96 €

www.elektor.fr/20644



Node-RED and Raspberry Pi Pico W

Ce livre constitue un guide d'apprentissage et une référence. Utilisez-le pour apprendre Node-RED, Raspberry Pi Pico W et MicroPython, et enrichir vos connaissances techniques. Il vous présentera les machines virtuelles, Docker et MySQL pour soutenir vos projets IdO basés sur Node-RED et le Raspberry Pi Pico W.

Prix : ~~44,95 €~~

Prix (membres) : 40,46 €

www.elektor.fr/20745

Oxocart Connect Innovator Kit

Prix : 89,95 €

Prix (membres) : 80,96 €

www.elektor.fr/20718

SunFounder Kepler Kit (Kit de démarrage ultime pour Raspberry Pi Pico W)

Prix : 69,95 €

Prix (membres) : 62,96 €

www.elektor.fr/20730

Station de soudage à température contrôlée ZD-931

Prix : 49,95 €

Prix (membres) : 44,96 €

www.elektor.fr/20623

FNIRSI DSO152 Mini Oscilloscope (200 kHz)

Prix : 39,95 €

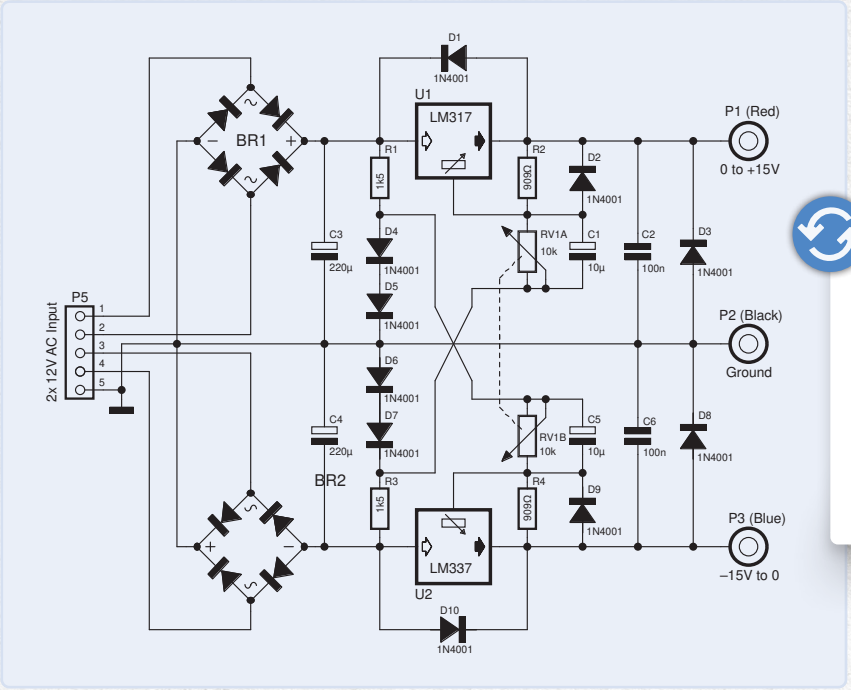
Prix (membres) : 35,96 €

www.elektor.fr/20642

projet 2.0

corrections, mises à jour et courrier des lecteurs

Compilés par Jean-François Simon (Elektor)



Alimentation linéaire variable

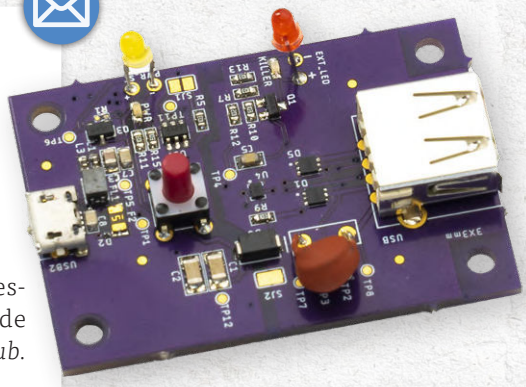
Elektor 1-2/2024, p. 26 (220457)

Il y a une erreur dans le schéma de l'alimentation variable double (p. 29). Le régulateur de tension négative situé en bas du schéma est un LM337 et non un LM317. Le texte de l'article est correct et indique bien un LM337.

Détecteur de clé USB killer

Elektor 11-12/2023, p. 32 (220549)

Le détecteur de clé USB killer peut-il être acheté assemblé, prêt à l'emploi ?
N. D. (Allemagne)



Vous trouverez tous les fichiers nécessaires à la réalisation du détecteur de clé USB killer sur GitHub (*https://github.com/instantdevices/USB-Killer-Detector*). Bien que je puisse éventuellement fournir les composants nécessaires, j'aimerais aussi une aide de la part de la communauté, en raison d'un problème persistant, où un modèle spécifique de « killer » détruit le détecteur. Je pense que le dispositif n'est pas encore prêt à être commercialisé. Vous pouvez me contacter à l'adresse *instant.devices@yahoo.com*.
Carlos Guzman (Auteur de l'article)

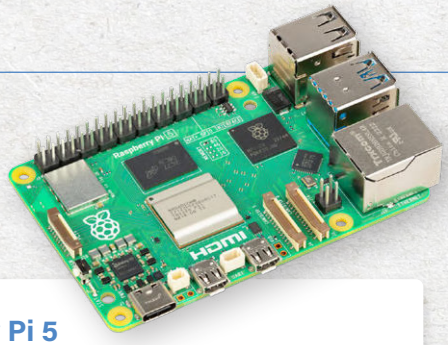
Avez-vous une idée intéressante ou un commentaire pour Elektor ? Contactez-nous à l'adresse redaction@elektor.fr – nous serons heureux de vous lire !

Mise en place une ligne d'assemblage CMS

Elektor 11-12/2023, p. 108 (230593)

Dans cet article, vous présentez le dispositif Whizoo Controleo3. Pouvez-vous m'indiquer une source d'approvisionnement possible en Allemagne ?
Wolf-Peter Kleinau (Allemagne)

Cet article a été écrit par notre partenaire Opulo, une société basée aux USA. Whizoo est également une entreprise basée aux États-Unis, et si je ne me trompe pas, il n'y a malheureusement pas de revendeurs en Europe. L'article d'Opulo indique que le dispositif « Controleo3 » est une solution clés en mains. Pour être plus précis, le Controleo3 était d'abord un kit permettant de réaliser votre propre four à refusion, et il s'avère que Whizoo commercialise également un four totalement assemblé pour environ 1 200 €. Malheureusement, ce produit n'est disponible qu'aux États-Unis et au Canada, et il contient un four fonctionnant sous une tension de 115 V. Si vous faites le choix des produits Controleo3, vous pouvez les acheter en kit, vous devrez alors régler séparément la TVA et prendre en charge l'importation, ce qui est parfois compliqué. Vous devrez approvisionner séparément et assembler vous-même le four, ce n'est donc pas une solution « clés en mains ».
Il n'y a pas de réponse simple à la question « quel four à refusion devrais-je acheter ? ». C'est un vaste domaine, et je vous encourage à faire une recherche approfondie afin de choisir ce qui correspond le mieux à vos besoins. Personnellement, je ne recommande aucune société en particulier. Toutefois, j'ai entendu parler de Beta Layout. En ce qui concerne les autres fabricants, il y a deux choix populaires qui sont le four T962 (en différentes configurations) et le ZB3530HL. Certaines personnes sur Internet ont également trouvé des moyens d'améliorer le T962 pour une meilleure répartition de la chaleur ou une interface utilisateur plus pratique.
Jean-François Simon (Elektor)



Le Raspberry Pi 5

Elektor 11-12/2023, p. 6 (230635)

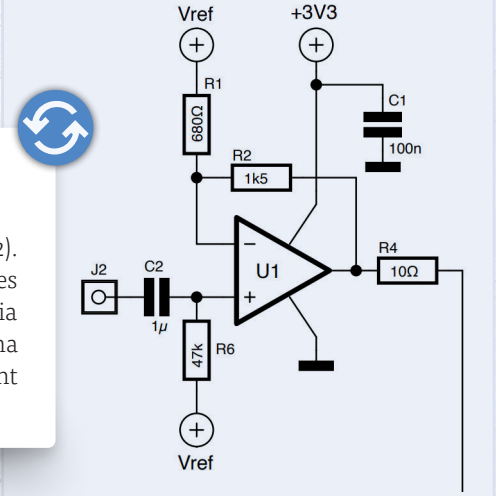
Merci pour cet article très instructif. Sur les différents Raspberry 4 que je possède, j'ai l'habitude de copier les cartes SD pour porter facilement tout ce que j'avais précédemment laborieusement installé (comme VS Code, Qt 5, CAN bus, etc.) vers différentes unités. Pouvez-vous me dire, ainsi qu'aux autres lecteurs, comment porter ces applications vers un Pi 5 ? J'ai inséré une carte SD copiée dans un Pi 5, mais cela ne fonctionne pas. Merci beaucoup.
Hanspeter Schären (Allemagne)

Merci pour votre e-mail. Cela est dû à la version de l'OS présente sur vos cartes SD utilisées avec le Raspberry Pi 4. Le Pi 5 nécessite la version *Bookworm* du système d'exploitation et il ne fonctionne pas avec les versions antérieures. Dans votre cas, malheureusement, vous devez créer une nouvelle carte en partant de zéro, par exemple à l'aide du Raspberry Pi Imager qui est facile à utiliser. Vous devrez ensuite réinstaller l'ensemble des logiciels dont vous avez besoin. Lorsque vous aurez préparé cette carte SD utilisant *Bookworm*, elle devrait en principe fonctionner indifféremment avec un Raspberry Pi 5 ou un Raspberry Pi 4, si vous devez passer d''un Raspberry à l'autre. Il vous faudra cependant faire le test par vous-même afin de vérifier que tout fonctionne. Bonne chance !
Jean-François Simon (Elektor)

Microphone MEMS

Elektor 11-12/2023, p. 70 (230188)

Nous avons commis une erreur en retraçant le schéma. (Figure 3, p. 72). Toutes nos excuses pour cette erreur ! Les entrées non inverseuses des amplificateurs opérationnels U1 à U6 sont reliées à la tension Vref via des résistances de 47 kΩ, et non pas à la masse (voir l'extrait du schéma à droite). Le schéma original disponible dans le projet *Hackaday.io* dont le lien est donné dans l'article est bien entendu correct.



Les composants à cathode froide

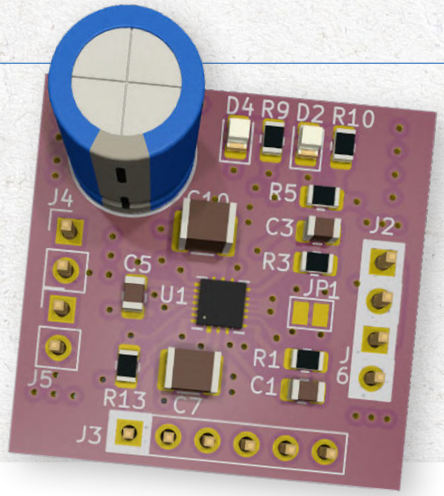
Elektor 1-2/2024, p. 74 (230373)

J'ai entendu dire que certains tubes à cathode froide étaient faiblement radioactifs. Est-ce vrai ? Par exemple, les tubes de marque Elesta ou d'autres tels que ER21A, GT21, 5823, ou le type GR16 ?
Adolf Parth (Italie)

Vous avez raison, certains de ces tubes contiennent des matériaux radioactifs favorisant l'ionisation du gaz qu'il renferment. Vous trouverez plusieurs sites web sur Internet, animés par des personnes passionnées par la radioactivité ou les compteurs Geiger, qui vous donneront davantage de détails sur les éléments radioactifs précis contenus dans tel ou tel tube.
Jean-François Simon (Elektor)

Le rôle de la cathode (dans les tubes à cathode froide ou chaude) est généralement d'émettre des électrons. Dans le cas des cathodes chaudes, elles émettent des électrons qui sont le plus souvent modulés par une grille et collectés par une anode. Dans les tubes à cathode froide, elles émettent des électrons qui ionisent un gaz, que ce soit pour la protection contre les surtensions, pour l'affichage (néons et tubes Nixies) ou pour la commutation, comme dans le cas des tubes thyratron à cathode froide, cités par Mr Parth.
Pour cette raison, les cathodes sont très souvent recouvertes de substances favorisant l'émission d'électrons. Dans le cas des tubes standards à cathode chaude, le Thorium est communément utilisé. Dans les tubes à cathode froide, certains composés métalliques et terres rares sont utilisés. Certains d'entre eux émettent une légère radioactivité. Dans le tableau périodique des éléments, les métaux rares font partie de la série des Lanthanides (éléments 57 à 71), le Thorium est dans la série des Actinides (éléments 89 à 103). Quelques éléments parmi les Lanthanides sont radioactifs, et la plupart des Actinides le sont. Le Thorium est également utilisé dans les revêtements des lampes à gaz, celles-ci sont donc légèrement radioactives. De nombreux éléments communs ont une proportion d'isotopes radioactifs, le Potassium en étant un bien connu.
David Ashton (Auteur de l'article)

Source : https://en.wikipedia.org/wiki/Nixie_tube#/media/File:ZM1210-operating_edit2.jpg



Carte d'interface pour pilote de moteur

Elektor 9-10/2023, p. 56 (210657)

Bonjour à tous ! Est-il possible de créer un contrôleur de vitesse pour ventilateur, en fonction de la température, avec le contrôleur MP6619 qui produirait, par exemple, une tension de 3 V à la température de 20°C et jusqu'à 12 V à 40°C, utilisant un capteur CTN de 10 kΩ ? Merci d'avance pour toute information ou exemples de circuits. Ou bien peut-être existe-t-il une autre possibilité de réaliser un tel contrôleur de vitesse ?
Matthias Seesemann (Allemagne)

Il est possible de réaliser ce que vous décrivez avec un MP6619, mais il vous faut également d'autres composants. Le circuit MP6619 est un pont en H, il commute ses transistors de sortie (état passant ou bloqué) selon l'état de ses broches d'entrée IN1, IN2, et ENABLE (actif ou inactif). La broche EN devra être maintenue au niveau haut par une résistance de tirage, vous pourrez alors maintenir IN2 au niveau haut ou bas, selon la direction de rotation voulue. Il faudra ensuite envoyer un signal PWM sur IN1, afin de faire varier la vitesse du moteur. Pour générer le signal PWM, vous pouvez utiliser n'importe quel microcontrôleur, un Arduino ou autre. Je vous recommande l'utilisation d'un Arduino Uno R3. De cette façon, vous trouverez de nombreux exemples de programmes en ligne. Vous apprendrez ainsi comment lire la température d'un capteur CTN, et mettre en œuvre les fonctionnalités PWM avec Arduino. Par ailleurs, vous pourriez également être intéressé par certains projets utilisant le « contrôleur de vitesse de ventilateur » TC648. Je crois que ce composant est conçu pour faire ce que vous recherchez. Vous devrez l'adapter en changeant la valeur d'une ou deux résistances afin qu'il se comporte exactement comme vous voulez. Vous devriez publier votre projet sur la plateforme Elektor Labs (*www.elektormagazine.fr/labs*), ainsi d'autres membres de la communauté Elektor pourront apporter leurs commentaires et vous aider. Amusez-vous bien !
Jean-François Simon (Elektor)

VF : Jean Boyer — 240020-04

Rejoignez la communauté Elektor

Devenez membre **maintenant !**

elektor GOLD membership

- accès à l'archive numérique depuis 1978 !
- 8x magazine imprimé Elektor
- 8x magazine numérique (PDF)
- 10 % de remise dans l'e-choppe et des offres exclusives pour les membres
- accès à plus de 5000 fichiers Gerber

Également disponible

abonnement elektor GREEN membership sans papier !

- accès à l'archive numérique d'Elektor
- 10 % de remise dans l'e-choppe
- 8x magazine Elektor (PDF)
- accès à plus de 5000 fichiers Gerber

www.elektormagazine.fr/membres

elektor design > share > earn

Renforcez vos compétences

Accédez à des conseils rapides, des outils et des articles pour les acheteurs professionnels

mou.sr/purchasing-resources

MOUSER ELECTRONICS