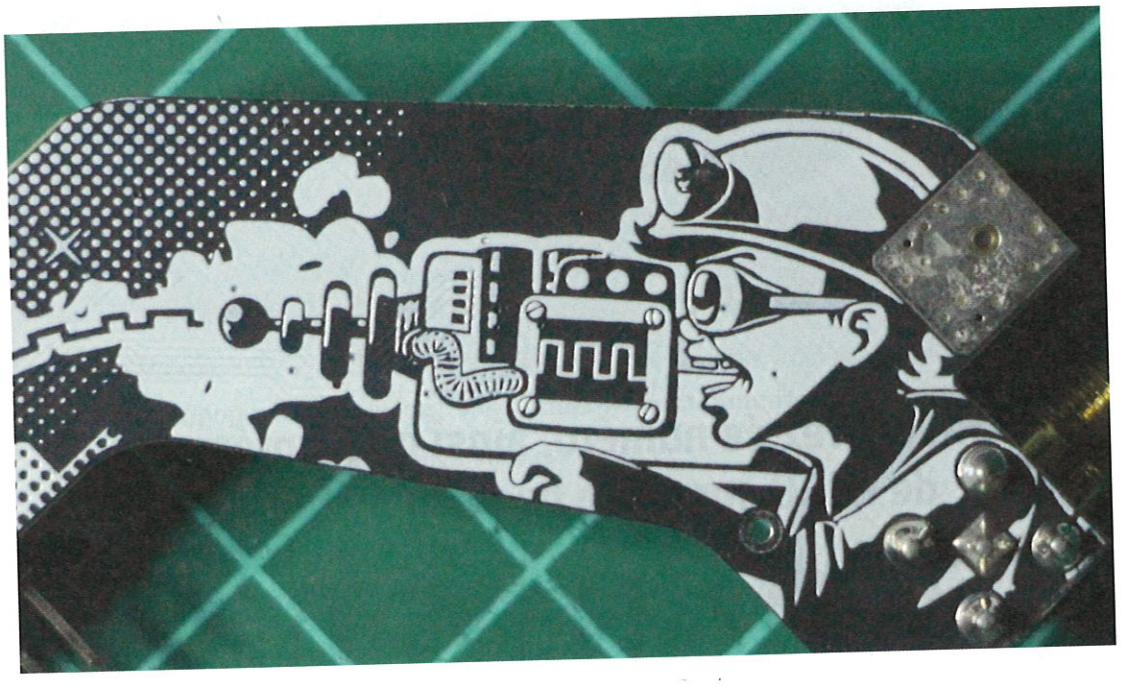
DIGILAB : TRANSFORMEZ VOTRE FLIPPER ZERO EN BOÎTE À OUTILS POUR L'ÉLECTRONIQUE

Denis Bodor

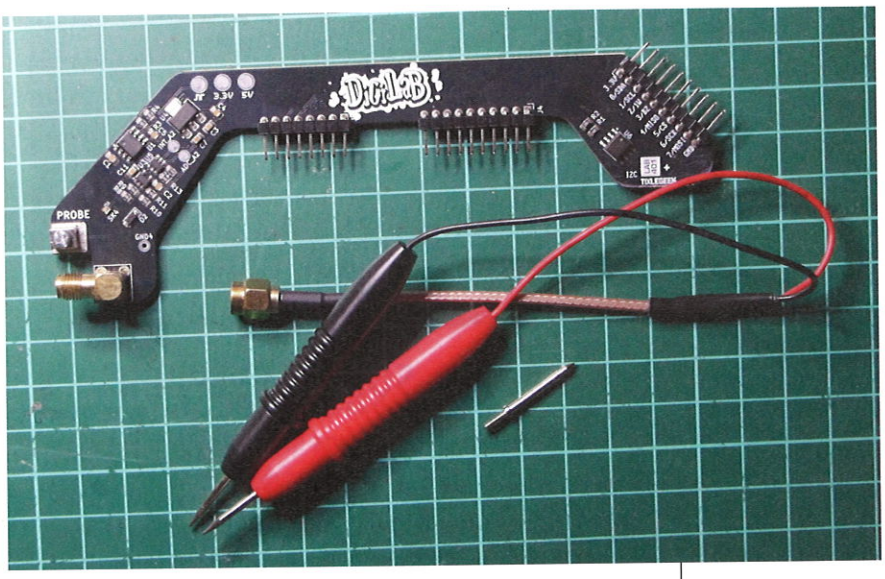
Le Flipper Zero ne nécessite presque plus d'introduction tant cet appareil de poche sachant gérer RFID/NFC, IR, RF, USB, U2F, HID, iButton, BLE, etc., a su se faire une place dans la trousse à outils de tous les pentesteurs, amateurs d'embarqué, de sans-contact, et que sais-je encore. Mais il manquait une corde à son arc, jusqu'à l'arrivée du DigiLab de Tixlegeek : faire office de multioutil pour l'électronique !



– DigiLab : transformez votre Flipper Zero en boîte à outils pour l'électronique –

Si on devait résumer à la fois les avantages et les inconvénients du Flipper Zero en une seule phrase, je pense que ce serait cette, plus ancienne qu'on le pense (18e siècle), figure de style anglaise : « *Jack of all trades, master of none* » (« touche-à-tout, maître en rien », autrement dit, capable de tout faire, mais n'excellant en rien de particulier). En effet, Le Flipper Zero est un condensé, en un seul produit, de tout ce que l'on peut vouloir faire avec, par exemple, un *laptop* équipé de nombreux périphériques dédiés. Il sait lire/écrire des *tags* NFC/RFID, mais ne vaut pas un Proxmark 3, il sait émuler des *tags* NFC, mais n'est pas un Chameleon Ultra (ou un Proxmark 3, à nouveau), il est capable de communiquer par radio, mais n'arrive pas à la cheville d'un HackRF, d'un PlutoSDR, d'un HydraSDR RFOne ou même d'un simple récepteur RTL SDR d'entrée de gamme, ou encore, il pourra se connecter à n'importe quoi, mais n'offre pas la richesse (GPIO, SPI, i2c, UART, etc.) d'un simple *devkit* ESP32 ou même d'une carte Arduino UNO.

Le DigiLab, quant à lui, produit et commercialisé par les Lyonnais de LAB401 **[1]** au prix d'environ 60 €, ajoute des fonctionnalités au Flipper Zero. L'approche n'est pas nouvelle puisque c'est précisément là l'une des fonctionnalités offertes par le Flipper en utilisant un ensemble de 18 connecteurs au pas de 2,54 mm situés sur la tranche de l'appareil. On trouve ainsi des modules de toutes sortes, allant de l'ajout du Wi-Fi (ESP32 Marauder ou Mayhem v2) à l'amélioration du support radio (Flux Capacitor ou Feberis Pro), en passant par la « peinture lumière » (Light Messenger) ou le « boost » infrarouge (IR Blaster). Ceci, sans oublier bien sûr le standard *WiFi Devboard* officiel **[2]**, équipé d'un ESP32-S2, lui aussi dédié au support Wi-Fi absent du produit « standard ».



Le DigiLab se présente sous la forme d'un simple circuit imprimé, accompagné de deux sondes ou « probes ».

La très grande majorité des modules « clé en mains » sont orientés Wi-Fi ou radio, mais il est parfaitement possible de connecter directement le Flipper à des montages « maison » et de développer son propre support dans le *firmware*, prévu pour accueillir des applications (ou FAP pour *Flipper App Package*) en C/C++ ou même en JavaScript. Il existe même des cartes de prototypage pastillées (*perfboards* ou *protoboards*) parfaitement adaptées permettant d'obtenir un résultat plus compact, sans que tout finisse en salade de câbles.

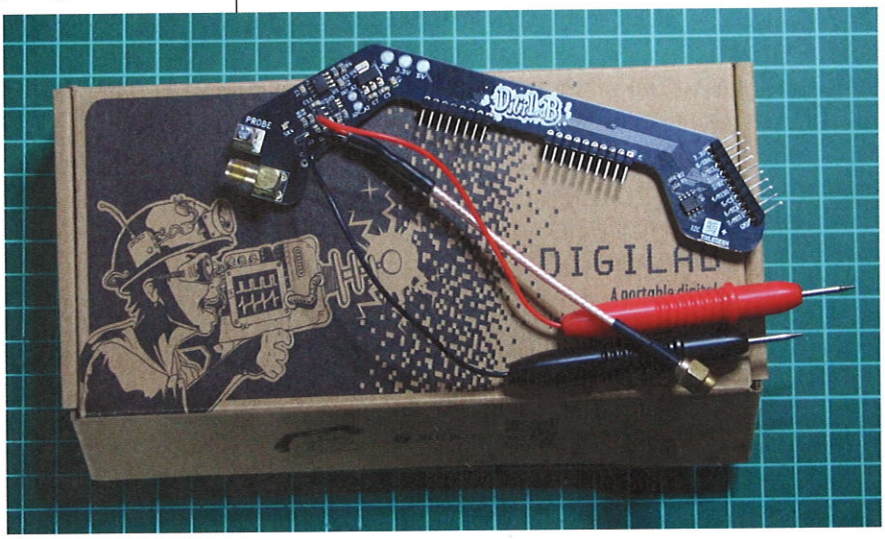
Avec le module de Tixlegeek, il ne s'agit pas de communication ou de *pentesting* Wi-Fi/radio, mais plutôt de faire du Flipper Zero un outil pratique pour le quotidien de l'amateur d'électronique en faisant office

d'oscilloscope, de multimètre (tensions 0 à +12 V et fréquences 0 à 200 kHz) et d'outil pour scanner et gérer les bus SPI et i2c. Le tout avec l'accent mis sur la mobilité, lorsqu'on a pas nécessairement sous la main ses outils plus encombrants ou qu'il n'est pas (encore) nécessaire de sortir « le gros outillage ». Ce qui est précisément la raison d'être initiale du Flipper Zero, comme précisé précédemment.

1. RAPPEL SUR LE FLIPPER ZERO

Pour beaucoup le Flipper Zero est un gadget, un jouet ou encore un dangereux outil pour méchants pirates en *hoodie* à capuche, devant être interdit à la vente (comme au Brésil), parce que les soi-disant serviteurs des citoyens écoutent de soi-disant journalistes qui ne comprennent rien à rien et ne savent plus que crier au loup à défaut d'être capable de la moindre pertinence, et ce dans le seul but de générer du trafic vers leur prose tout aussi alarmiste que médiocre (pardon, je m'emporte). Personnellement, j'y vois simplement une plateforme de développement avec une forme plus « finie et propre » qu'un *devkit* classique, avec ses modules additionnels qui pendouillent de tous les côtés, et moins encombrant qu'un ordinateur portable ou même un *cyberdeck* DIY.

On remarquera le soin tout particulier apporté au packaging. Quand on a l'habitude des commandes via AliExpress, ça fait presque un choc !



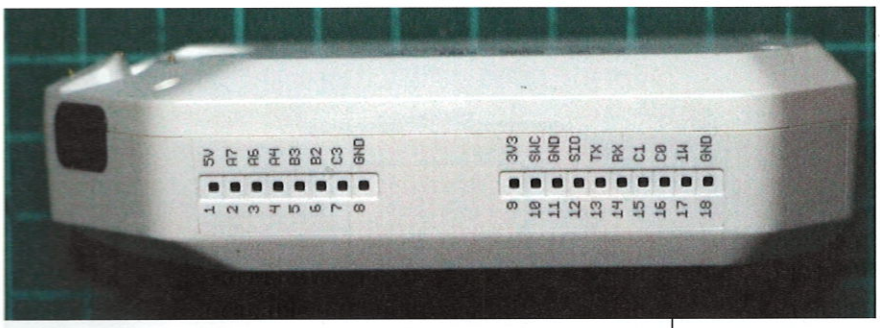
Construit autour d'un microcontrôleur STMicroelectronics STM32WB55RG **[3]** (cœur ARM Cortex-M4) embarquant quelque 1024 Kio de flash et 256 Kio de RAM, le Flipper se différencie d'un simple système de développement par son aspect général, mais aussi, et surtout par le côté « flashy » et « cartoon » qu'il met en avant (mascotte dauphin, idéogrammes japonais et texte en cyrillique, animation à l'écran, aspect général presque caricatural de style *cyberpunk*). Réunis autour de ce MCU STM32, nous avons ce qui fait la richesse du matériel, c'est-à-dire une ribambelle de périphériques :

- un contrôleur NFC ST25R3916 compatible ISO 14443A/B, FeliCa, ISO 15693 (vicinity), MIFARE Classic, etc. ;
- un circuit pour le support (purement logiciel par le STM32WB55RG) des *tags* RFID 125 kHz de type EM4100, HID H10301, HID Prox, T5577, etc. ;
- un *transceiver* Texas Instruments CC1101 travaillant par défaut (en Europe) sur la bande ISM et plus particulièrement les plages de fréquences de 433,05 à 434,79 MHz et de 868,15 à 868,55 MHz ;

- un émetteur-récepteur infrarouge avec une longueur d'onde de 800-950 nm ;
- le Bluetooth directement pris en charge par le microcontrôleur STM32WB55RG également capable de supporter Thread 1.3 et Zigbee 3.0 ;
- une interface USB-C pilotée par le MCU et capable d'apparaître comme n'importe quel périphérique USB 2.0 *FullSpeed* (voir l'article dans le numéro 60 sur le sujet avec les MCU ST32 **[4]**) ;
- un emplacement pour une carte microSD avec un support jusqu'à 64 Gio (non compatible SDIO, SPI uniquement).

Et tout ceci est complété d'une IHM composée d'un écran LCD 1,4 pouces monochrome de 128×64 pixels à rétroéclairage orange (contrôleur ST7565R en SPI) et d'un ensemble de 6 boutons disposés façon « gamepad » pour la navigation. Et enfin, nous avons une LED RGB, un *buzzer* (100-2500 Hz) et un vibreur (comme dans les smartphones) pour clore ce point.

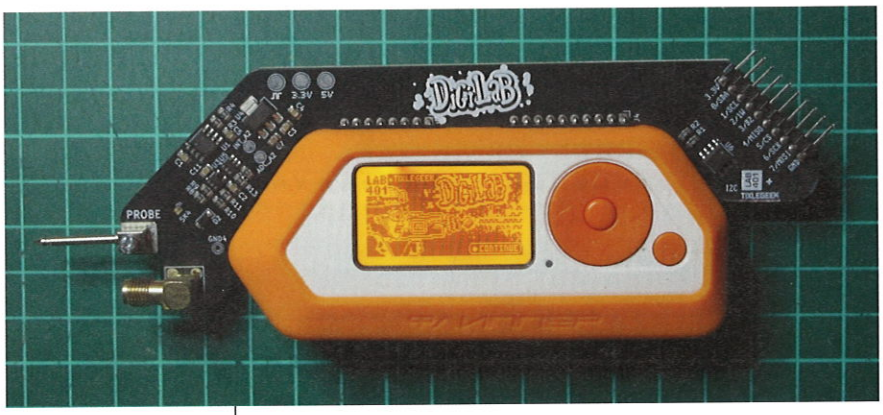
L'ensemble est compressé dans un boîtier de 10 cm de long par 4 cm de large (et 2,5 cm d'épaisseur), parfaitement adapté pour se glisser dans une poche, donnant une impression de robustesse et de qualité (qu'on pourra compléter d'une gaine en silicone et d'une protection anti-rayure pour l'écran).



Les Flipper Zero permettent d'accueillir des modules externes s'enfichant sur ce connecteur sur la tranche. Il ne s'agit que d'une toute petite partie des signaux disponibles sur un STM32, mais c'est suffisant dans la plupart des cas.

Cet ensemble d'éléments ne serait pas d'un grand intérêt sans un *firmware* pour l'utiliser et celui du Flipper Zero est, bien entendu, *open source*. Il s'accompagne également de tout un écosystème d'outils et d'applications permettant de tenir son Flipper à jour, mais également de gérer les applications (FAP) qu'on peut y ajouter. Plusieurs options existent pour ce faire :

- Activer le Bluetooth sur le Flipper et installer l'application sur son smartphone, que ce soit un périphérique Android **[5]** ou un iPhone **[6]**.
- Connecter le matériel en USB à un PC et utiliser l'interface web disponible à l'URL **https://lab.flipper.net/** (il est également possible de la faire fonctionner localement dans un conteneur Docker comme je l'ai expliqué dans le numéro 51 **[7]**). Ceci ne fonctionnera cependant qu'avec les navigateurs supportant l'accès au matériel (API Web Serial) et donc ceux basés sur Chromium (Google Chrome, Chromium, Edge, Opera, etc.). Firefox est hors jeu.
- En USB également, via l'application qFlipper disponible **[8]** pour GNU/Linux (APPimage), Windows et macOS. Il s'agit d'une application graphique assez simple et moins ergonomique, concernant la gestion d'applications, que l'interface web, mais elle permet à minima d'installer des FAP et de maintenir le *firmware* à jour.
- Toujours via USB, on pourra également utiliser uFBT (*Micro Flipper Build Tool*), un outil prévu pour le SDK, mais qui permettra également de



Le DigiLab se connecte et devient une partie intégrante du Flipper Zero en adoptant le même profil. Ceci permet d'utiliser l'ensemble d'un seul bloc comme un testeur, même sans les sondes (à condition de connecter la masse à la cible, bien sûr).

manipuler tout cela en ligne de commande (`ufbt`) et qui s'installera, de préférence dans un environnement virtuel Python, en invoquant simplement `python3 -m pip install --upgrade ufbt` (une déclinaison existe également pour macOS et Windows). Il s'agit premièrement d'un outil pour construire et déboguer des applications, mais l'installation et le lancement sont possibles via `ufbt launch APPSRC=chemin/vers/application.fap`.

Notez également qu'il existe un « *store* », facilement accessible avec l'interface web, qui permet de naviguer dans une énorme collection d'applications, classées par type (NFC, Wi-Fi, GPIO, jeux, outils, etc.), et d'installer ce qui vous chante. Ceci est également valable pour l'application mobile et les deux options seront, très probablement, celles qu'il vaudra mieux utiliser si vous ne comptez pas vous pencher sur le développement, la création d'applications ou la contribution à une application existante.

Enfin, concernant l'aspect générique du *firmware*, il est important de préciser que la version officielle n'est pas la seule existante, développée et maintenue. Certaines limitations, souvent légales (bande de fréquences utilisables, par exemple), ont poussé à la création de *firmwares* alternatifs. À ce jour, les plus utilisés/populaires sont *Unleashed*, *Momentum* et *Roguemaster*, découlant plus ou moins tous du *firmware* Xtreme dont le développement a cessé fin 2024. Si vous êtes intéressé par ce genre de choses, ou que les animations « cartoon » vous insupportent, visiter l'excellent site *Awesome Flipper* **[9]** est un excellent point de départ.

2. PRISE EN MAIN DU DIGILAB

Ce petit rappel indispensable concernant le Flipper Zero dûment fait, nous pouvons nous pencher sur ce qui nous importe aujourd'hui : le DigiLab et son application 401/DigiLab, installable, vous vous en doutez, en quelques clics depuis l'interface web **[10]** ou mobile.

Le matériel se présente sous la forme d'un PCB adoptant le profil du Flipper Zero et se connectant à l'ensemble des broches femelles situées sur la tranche de l'appareil, tout en s'arrangeant parfaitement de la présence éventuelle de la gaine en silicone officielle. Il s'accompagne d'une petite pointe en métal à fixer à l'emplacement « PROBE » sur le circuit et d'une paire de sondes, type multimètre, à brancher au connecteur RP-SMA juste à côté. Notez qu'en utilisant un adaptateur RP-SMA vers BNC femelle, vous pourrez parfaitement utiliser une sonde d'oscilloscope, voire, en ajoutant un adaptateur BNC

vers double fiche banane, des sondes de multimètre (attention aux capacités parasites induites). La pointe fixée au PCB partage la même connexion, et la même fonction, que la sonde rouge, c'est juste un autre moyen « mécanique » de tester un circuit (à condition de mettre le DigiLab à la masse via, par exemple, la sonde noire ou le via « GND 4 » à proximité du connecteur RP-SMA).

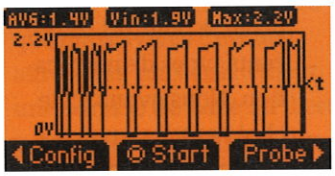
Toujours côté matériel, le circuit comporte plusieurs éléments intéressants :

- une LED RGB pilotée par l'application et permettant de remonter rapidement une information (tension seuil, alerte, déclencheur pour l'oscilloscope, etc.) ;
- un circuit « probe », composé d'un amplificateur opérationnel (ampli-op ou *opamp* en anglais) et d'un comparateur, permettant de gérer des tensions jusqu'à +12 V et de « nourrir » le convertisseur analogique/numérique (ADC) du STM32 du Flipper Zero ;
- une EEPROM i2c de 256 octets (2 kbits) 24C02 ;
- un circuit à base de NE555 (mode astable) générant un signal carré à 4800 Hz récupérable sur l'un des points de test (à gauche de « 3.3 V » et « 5 V ») ;
- et une série de broches, au pas de 2,54 mm, proposant les signaux du bus i2c (SDA, SCL) et du bus SPI (MISO, MOSI, SCK, CS), ainsi que 1-wire (1W) et GPIO B2.

Côté logiciel, une fois l'application installée, la première exécution, via le menu ***Apps***, ***GPIO*** et ***401/DigiLab***, nécessitera une calibration. C'est une opération que vous pourrez reproduire par la suite si vous constatez des mesures inexactes ou douteuses. L'opération est relativement aisée puisque le programme vous demande simplement de vous assurer que les sondes sont connectées et, à l'étape de calibration, de mettre en contact la sonde rouge avec le point de test « 5 V », puis d'appuyer sur le sélecteur central du « pad » du Flipper. Une petite mélodie et un message vous informeront du bon déroulé de l'opération, et votre DigiLab sera alors prêt à être utilisé.

Tout le reste se passe dans le menu de l'application dont les entrées, en dehors de « Calibration », « About » (à propos) et « Splash » (image d'accueil), listent les fonctionnalités à votre disposition (pour l'instant) et que nous allons traiter une à une…

2.1 Oscilloscope



L'entrée « Scope » nous donne accès à un oscilloscope assez spartiate et sans prétention. Ne vous attendez pas à quelque chose de réellement utilisable en tant que tel, car même en dehors des fonctionnalités limitées de base (ADC 12 bits du STM32 avec un *sample rate* de 4,26 Msps max), l'écran de 128×64 pixels du Flipper sera bien incapable de fournir une interface digne de ce nom. L'affichage arrive tout de même à donner une vague idée de la forme du signal et à indiquer la tension instantanée, la tension moyenne et la tension maximum. Le niveau de déclenchement (*trigger*) est automatique et si vous trouvez qu'il est bien dommage de ne pas avoir d'estimation de la fréquence détectée, ne vous inquiétez pas, l'information est simplement ailleurs.

Un appui sur la droite du *pad* vous donne accès

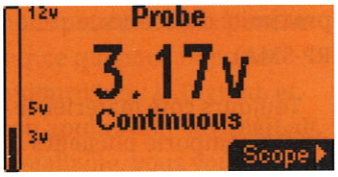
au menu de configuration et c'est là que les choses deviennent intéressantes. Traitons les quatre entrées dans le désordre :

- « Alert » : permet de régler la condition d'alerte pour les différents types de notification. Nous avons ici le choix entre : « u<3.3 » pour une tension mesurée inférieure à 3,3 V, « u>3.3 » pour supérieure à 3,3 V, « u<5v » pour inférieure à 5 V, « u>5v » pour supérieure à 5 V, « Osc » dans le cas d'une tension oscillant autour de 0 V (typiquement un signal d'horloge), « ~MAX » pour une tension qui dépasse 12 V, et enfin « ~0v » pour une tension proche de 0 V/masse. Ce qui se passe lorsqu'une alerte est déclenchée dépendra des entrées du menu qui suivent.
- « Sound » : peut être activé (« ON ») pour avoir un retour audio en fonction de la tension mesurée, désactivé (« OFF ») ou réglé pour émettre un son si la condition d'alerte est vérifiée.
- « Vibro » : déclenche le vibreur en cas d'alerte (« Alert ») ou pas (« OFF »).
- « LED » : propose davantage d'options, avec l'arrêt complet (« OFF »), le déclenchement en fonction de la condition d'alerte (« Alert »), une teinte changeant linéairement suivant la tension mesurée entre 0 et 12 V (« Follow »), une teinte en fonction de la variance **[11]** du signal (« Variance »), et « Trigger » qui allume la LED lorsque la tension dépasse la valeur du déclencheur autocalculé.

Certains de ces paramètres ne présentent, à mon sens, que peu d'intérêt, mais avoir un retour sonore et visuel (LED) lorsqu'on cherche une masse, un +5 V ou un +3,3 V sur un circuit est furieusement intéressant. Il en va de même pour le retour audio concernant un signal oscillant, qui peut être une fonctionnalité très pratique lorsqu'on cherche un bus série ou un signal d'horloge en faisant un peu de *reverse*.

Une amélioration possible de ce point de vue pourrait être le fait d'avoir un réglage plus précis que le simple supérieur/inférieur à 3,3 V et 5 V, en pouvant choisir arbitrairement une tension de seuil déclenchant l'alerte, voire une série de tensions (plus ou moins une certaine tolérance réglable) avec une correspondance avec une fréquence sonore. Ceci faciliterait non seulement la recherche d'une tension précise sur une carte (comme du 1,2 V pour un *devkit* FPGA), mais également la vérification d'une alimentation propre et stable en différents points d'un circuit.

2.2 Probe



Cette fonction est accessible depuis le menu, mais également en appuyant sur « droite » du *pad* lorsqu'on est sur l'oscilloscope. Et inversement, en mode « probe », la même action renvoie au « scope », montrant clairement que les deux fonctionnalités ont été pensées pour fonctionner de concert.

De but en blanc, ceci ressemble à un simple voltmètre, ce qui est déjà très pratique si tout ce qu'on a sur soi est un Flipper Zero (et non son bien aimé Agilent U1272A de six fois sa taille), mais la fonctionnalité va un peu plus loin. Gardez cependant bien à l'esprit que ceci ne remplacera pas un multimètre de qualité, ni en termes de précision ni d'exactitude (sauf peut-être

les appareils pour bricoleurs du dimanche vendus en magasins de bricolage).

La surprise arrive lorsqu'on mesure une tension non continue, avec l'affichage automatique de la fréquence mesurée (en lieu et place du mot « *continuous* » sous la valeur). C'est, bien entendu, quelque chose qu'on retrouvera éventuellement sur un multimètre, mais qui pourra être ici bien utile dans certaines situations.

Bien entendu, la fréquence mesurée est limitée (à 200 kHz max), ce qui ne saurait satisfaire à tous les besoins, surtout pour l'embarqué. La jauge sur la gauche est intéressante, mais mériterait, je pense, une petite amélioration : pourquoi ne pas maintenir un indicateur en place de la plus haute tension mesurée les 10 dernières secondes, à la manière des VU-mètres hi-fi ?

Autre point perfectible à mon sens, le système d'alerte n'est actif que sur la partie « scope », mais est désactivé en « probe », alors qu'il serait d'une grande aide dans un cas comme dans l'autre.

2.3 i2c

Le DigiLab « exporte » non seulement le bus i2c accessible sur la tranche du Flipper Zero sur la droite du PCB, mais intègre également une EEPROM 24C02 qui y est connectée. Ainsi, pour tester la fonctionnalité « I2C Probe », il n'est pas vraiment nécessaire d'avoir un lot de composants à disposition. La fonction cependant est relativement mal nommée, car on pourrait supposer qu'il ne s'agit que d'une fonctionnalité de scan du bus, un peu comme ce que propose l'application « i2c Tools » de NaejEL (qui permet aussi d'envoyer des trames i2c).

Vous pouvez, bien entendu, scanner un bus et avoir ainsi une idée à peu près correcte des périphériques qui le composent (l'opération n'est, par définition, pas fiable puisque les spécifications i2c ne proposent rien dans ce sens). Ils apparaîtront alors chacun sur un écran et on pourra basculer de l'un à l'autre avec gauche/droite :

Ce qui fait la différence avec les autres applications, c'est la prédiction. En effet, ce bus n'a rien à voir avec quelque chose de plus étoffé comme l'USB ou le PCI/PCIe où un mécanisme d'énumération existe pour que chaque périphérique informe le système de son identité (classe, vendeur, modèle, etc.). En i2c, tout ce que nous avons, c'est une adresse et, plus exactement, une adresse **de base** qu'il est parfois possible de changer en ajustant l'état de jusqu'à trois broches dédiées (A0 à A2). Pour déterminer quels périphériques sont supposément présents sur le bus, il faut donc comparer les adresses où « quelque chose répond » à une base de puces/adresses pour prédire une éventuelle correspondance. Et c'est précisément ce que fait l'application DigiLab, listant des types possibles au bas de l'écran et proposant ensuite, via le bouton central du *pad*, d'afficher davantage d'informations :

À partir de cet écran, il est alors possible d'utiliser une nouvelle fois le bouton central pour basculer sur une interface permettant de spécifier un registre auquel accéder, pour enfin aboutir à un éditeur hexadécimal basique pour lire/écrire des valeurs :

L'ensemble fonctionne relativement bien et offre des fonctionnalités bien plus intéressantes que ce qui existait auparavant en termes d'application pour l'i2c. Bien entendu, c'est relativement simpliste par rapport aux solutions qui existent, ou qui peuvent être rapidement développées sur PC ou avec un MCU, mais je pense que c'est aussi le maximum de ce qu'il est possible d'obtenir avec un Flipper Zero. Arriver à faire tenir un éditeur hexadécimal utilisable sur un si petit écran est déjà un tour de force, ce serait abuser que de réclamer davantage...

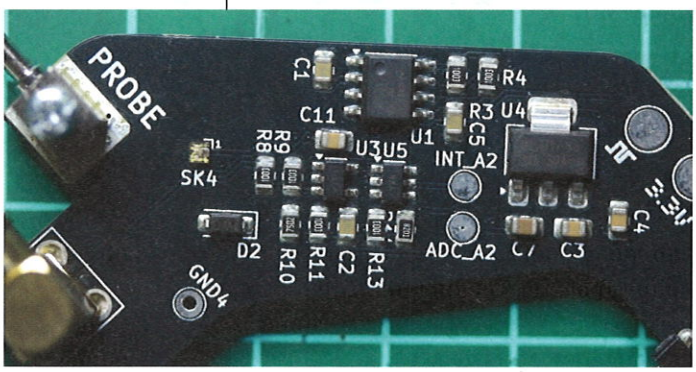
Sauf, bien sûr, si l'on imagine pousser le concept beaucoup plus loin, en imaginant un système de greffons, et/ou de fonctionnalités modulaires à la compilation de l'application, permettant de prendre en charge des puces i2c spécifiques comme des RTC Maxim Dallas, des capteurs de température, pression, hygrométrie, etc. Mais là, ceci relève du « gros œuvre », non plus de la petite contribution et il peut tout aussi bien suffire de développer une application dédiée par composant, comme BME680 **[12]** qui existe déjà dans le *store*.

Notez que le DigiLab seul, connecté au Flipper Zero, détecte non seulement l'EEPROM 24c02 présente sur le circuit en 0x57, mais également un périphérique inconnu fantôme en 0x5F. Ceci n'a rien de surprenant et arrive avec d'autres composants **[13]**. C'est un problème connu et, ne l'oubliez pas, le scan i2c est quelque chose qui n'est absolument pas couvert par les spécifications du bus.

2.4 SPI

Ici, malheureusement, il n'y a pas grand-chose à dire. Les fonctionnalités SPI à ce jour se limitent à la détection de périphériques et la prédiction du type de puce sur le bus. Pour l'heure, seuls 6 composants sont détectés : 3 types de flash/EEPROM (25Q80, 25LC256 et AT45DB1016), un ADC (MCP3008) et deux capteurs (MPU9250 et TMP102). N'ayant aucun exemplaire de ces composants sous la main, et ayant failli honteusement dans ma tentative d'ajout rapide et brouillonne (sinon sauvage) du support pour une flash Winbond W25Q32BV, je n'ai pu tester cette fonctionnalité.

Et ceci clôt le tour de ce qu'offre le DigiLab à ce jour, même si, vous l'aurez compris, les évolutions possibles sont surtout dépendantes du support logiciel, dont les sources sont disponibles sur GitHub **[14]**.



Le circuit gérant les sondes se charge de convertir le signal en 0 et 12 V vers des tensions adaptées au convertisseur analogique/numérique du microcontrôleur du Flipper Zero.

3. JETONS UN ŒIL AUX FAP

Si cet article vous a inspiré et que vous souhaitez contribuer au développement du support DigiLab du Flipper Zero, la première chose à faire sera d'en récupérer les sources et de reconstruire l'application vous-même.

Pour ce faire, vous devez disposer du SDK et de l'environnement de construction adapté, c'est-à-dire du *Micro Flipper Build Tool (uFBT)* fournissant, presque clé en main, tout le nécessaire (outils, chaîne de compilation, bibliothèques, etc.). Le tout étant basé sur Python, nous pourrons alors commencer par créer un environnement virtuel dédié avec :

```
$ cd kkpart
$ mkdir FLIPPER
$ cd FLIPPER
$ python3 -m venv flipenv
$ source ./flipenv/bin/activate
```

Cette dernière commande active l'environnement, qui restera actif jusqu'à ce que vous fermiez l'émulateur de terminal utilisé, ou que vous appeliez la commande `deactivate`. Vous devrez réactiver l'environnement avant chaque session de travail avec `source` (ou « `.` »). L'invite de la ligne de commande reflétera l'activation en affichant un « `(flipenv)` » en début de ligne.

Ceci fait, nous pouvons installer la base d'uFBT en utilisant le gestionnaire de paquets `pip` :

```
$ pip install --upgrade ufbt
Collecting ufbt
  Using cached ufbt-0.2.6-py3-none-any.whl (25 kB)
Collecting oslex>=0.1.3
  Using cached oslex-0.1.3-py3-none-any.whl (3.5 kB)
Collecting mslex
  Using cached mslex-1.3.0-py3-none-any.whl (7.8 kB)
Installing collected packages: mslex, oslex, ufbt
Successfully installed mslex-1.3.0 oslex-0.1.3 ufbt-0.2.6
```

Vous disposerez alors de la commande `ufbt` que nous utilisons immédiatement pour installer le SDK :

```
$ ufbt update
05:12:16.209 [I] Deploying SDK for f7
05:12:16.209 [I] Fetching version info for UpdateChannel.
   RELEASE from
   https://update.flipperzero.one/firmware/directory.json
05:12:16.442 [I] Using version: 1.4.2
05:12:16.442 [I] uFBT SDK dir: /home/denis/.ufbt/current
05:12:17.738 [I] Deploying SDK
05:12:17.941 [I] SDK deployed.
```

Notez qu'en dehors des commandes `ufbt`, `ufbt-bootstrap` et `mslex-split` (ainsi que des modules Python dépendants), qui sont installés dans l'environnement virtuel, tout le reste arrivera dans un sous-répertoire `.ufbt/` de votre répertoire personnel (`$HOME`). Si vous souhaitez supprimer le *Micro Flipper Build Tool* ou réinstaller le tout pour une raison ou une autre, il faudra veiller à supprimer ce répertoire (environ 1,3 Gio en tout).

À ce stade, nous avons presque le nécessaire. Il nous manque encore la chaîne de compilation, mais celle-ci s'installera automatiquement lors d'une première construction d'application. Nous enchaînons donc sur la récupération des sources de l'application DigiLab depuis GitHub avec :

```
$ git clone https://github.com/lab-401/fzDigiLab.git
Clonage dans 'fzDigiLab'...
remote: Enumerating objects: 309, done.
remote: Counting objects: 100% (309/309), done.
remote: Compressing objects: 100% (171/171), done.
remote: Total 309 (delta 155), reused 276 (delta 133),
        pack-reused 0 (from 0)
Réception d'objets: 100% (309/309), 10.05 Mio |
        23.60 Mio/s, fait.
Résolution des deltas: 100% (155/155), fait.
```

Ce dépôt contient plus que les sources de l'application, vous y trouverez aussi un modèle 3D STL du DigiLab sur un Flipper Zero, ainsi que le schéma au format PDF du circuit (non à jour, il présente un emplacement U8 pour une EEPROM SPI 25LCxx qui n'a finalement pas été ajouté dans la version de production et les valeurs de résistance du circuit NE555 sont fausses). Pour construire l'application, nous devons nous déplacer dans le sous-répertoire `401DigiLabApp` et y utiliser à nouveau `ufbt` :

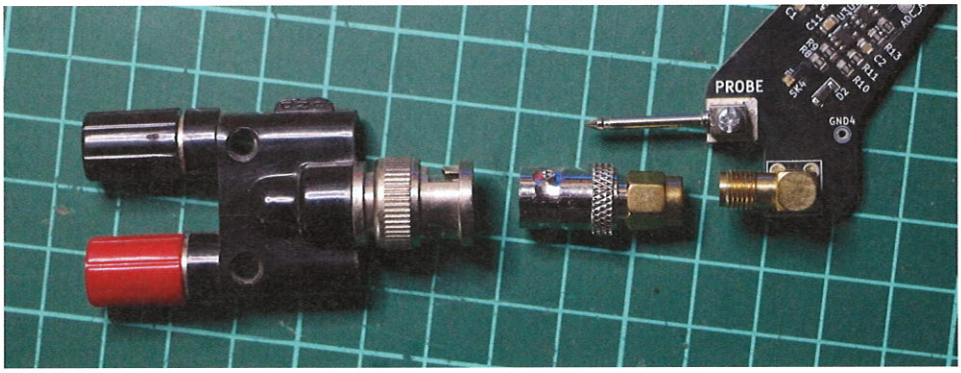
```
$ cd fzDigiLab/401DigiLabApp
$ ufbt build
Checking for tar..yes
Checking if downloaded toolchain tgz exists..no
Checking curl..yes
Downloading toolchain:
######################################## 100.0%
done
Removing old toolchain..done
Unpacking toolchain to '/home/denis/.ufbt/toolchain':
######################################## 100.0%
linking toolchain to 'current'..done
Cleaning up..done

scons: Entering directory `/home/denis/.ufbt/current/scripts/ufbt'
CC      [...]401DigiLabApp/cJSON/cJSON.c
CC      [...]401DigiLabApp/cJSON/cJSON_helpers.c
CC      [...]401DigiLabApp/drivers/sk6805.c
CC      [...]401DigiLabApp/fonts/u8g2_font_logisoso20_tr.c
CC      [...]401DigiLabApp/fonts/u8g2_font_tom_thumb_4x6_mr.c
CC      [...]401DigiLabApp/ringbuffer/ringbuffer.c
ICONS   /home/denis/.ufbt/build/401_digilab/401_digilab_icons.c
CC      [...]401DigiLabApp/401_gui.c
CC      [...]401DigiLabApp/board.c
CC      [...]401DigiLabApp/devicehelpers.c
CC      [...]401DigiLabApp/osc.c
```

```
CC      /home/denis/.ufbt/build/401_digilab/401_digilab_icons.c
CC      [...]401DigiLabApp/scenes/spi/401DigiLab_spitool_scanner.c
CC      [...]401DigiLabApp/401DigiLab_probe.c
CC      [...]401DigiLabApp/scenes/spi/401DigiLab_spitool.c
CC      [...]401DigiLabApp/401DigiLab_main_menu.c
CC      [...]401DigiLabApp/401_sign.c
CC      [...]401DigiLabApp/scenes/scope/401DigiLab_scope_config.c
CC      [...]401DigiLabApp/401DigiLab_main.c
CC      [...]401DigiLabApp/scenes/scope/401DigiLab_scope.c
CC      [...]401DigiLabApp/401DigiLab_config.c
CC      [...]401DigiLabApp/401_config.c
CC      [...]401DigiLabApp/scenes/i2c/401DigiLab_i2ctool_scanner.c
CC      [...]401DigiLabApp/401DigiLab_splash.c
CC      [...]401DigiLabApp/scenes/i2c/401DigiLab_i2ctool_reader.c
CDB     [...]401DigiLabApp/.vscode/compile_commands.json
LINK    /home/denis/.ufbt/build/401_digilab_d.elf
INSTALL [...]401DigiLabApp/dist/debug/401_digilab_d.elf
APPMETA /home/denis/.ufbt/build/401_digilab.fap
APPFILE /home/denis/.ufbt/build/401_digilab.fap
FAP     /home/denis/.ufbt/build/401_digilab.fap
FASTFAP /home/denis/.ufbt/build/401_digilab.fap
INSTALL [...]401DigiLabApp/dist/401_digilab.fap
APPCHK  /home/denis/.ufbt/build/401_digilab.fap
        Target: 7, API: 87.1
```

L'application résultante, au format FAP (qui est en réalité un binaire ELF ARMv5TE 32 bits EABI version 5), se retrouve dans `~/.ufbt/build/401_digilab.fap`, mais il ne vous est pas nécessaire de désigner directement ce fichier pour l'installer. Encore une fois, `ufbt` fera tout le travail à votre place en installant et lançant l'application :

```
$ ufbt launch
scons: Entering directory `/home/denis/.ufbt/current/scripts/ufbt'
python3 /home/denis/.ufbt/current/scripts/runfap.py -p auto
        -s /home/denis/.ufbt/build/401_digilab.fap
        -t /ext/apps/GPIO/401_digilab.fap
        APPCHK  /home/denis/.ufbt/build/401_digilab.fap
                Target: 7, API: 87.1
2025-12-04 06:56:33,384 [INFO] Using flip_Oect1p4 on /dev/ttyACM0
2025-12-04 06:56:33,917 [INFO] Installing
    "/home/denis/.ufbt/build/401_digilab.fap"
    to /ext/apps/GPIO/401_digilab.fap
2025-12-04 06:56:33,930 [INFO]
    Sending "/home/denis/.ufbt/build/401_digilab.fap"
    to "/ext/apps/GPIO/401_digilab.fap"
<100%, chunk 18 of 18 @ 113.78 kb/s
2025-12-04 06:56:35,190 [INFO] Closing current app, if any
2025-12-04 06:56:35,204 [INFO]
    Launching app: /ext/apps/GPIO/401_digilab.fap
```

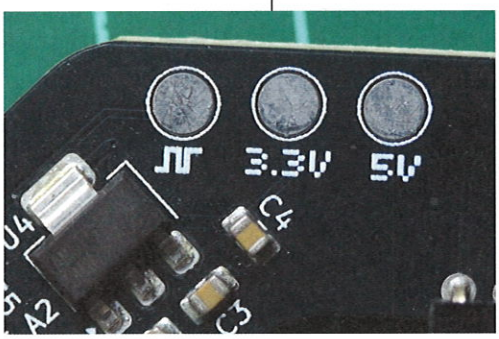


Les sondes se connectent au DigiLab en RP-SMA. En ajoutant un adaptateur vers BNC, on peut alors utiliser une sonde d'oscilloscope et même compléter d'un autre adaptateur pour obtenir deux fiches banane pour sondes de multimètre.

Pour que cela fonctionne, il faudra, bien sûr, que votre Flipper Zero soit connecté en USB à la machine de développement (et que les permissions sur le port série soient correctes). Notez également que si vous avez le Flipper Lab (interface web) ouvert dans un onglet de navigateur et que celui-ci est connecté au matériel, un conflit surviendra. Ceci est également valable pour l'application qFlipper, étant donné qu'un seul outil peut ouvrir le port série (généralement `/dev/ttyACM0`) à la fois.

Et voilà, vous avez une version construite à partir des sources qui fonctionne sur votre Flipper Zero. Il est important de relever que les applications Flipper Zero, bien qu'étant des binaires ELF, s'accompagnent de données qui se retrouvent généralement ensuite placées sur la carte microSD (tout comme l'application elle-même), dans un répertoire (caché) `apps_assets/`. Dans le cas de l'application DigiLab, ce répertoire `401_digilab/` contient les données de calibration, la configuration (alertes, etc.) ainsi que, dans `targets/`, un ensemble de fichiers JSON utilisés pour la prédiction de composants sur les bus i2c et SPI. Le répertoire `401_digilab/` et son contenu sont automatiquement créés, à partir des métadonnées embarquées dans l'ELF, lors de la première exécution de l'application, mais ne sont pas supprimés en la désinstallant avec l'interface web, par exemple. Ceci peut parfois poser problème si vous travaillez justement sur la détection i2c/SPI (même si une nouvelle installation de l'application met théoriquement à jour les fichiers). Pensez donc à la supprimer avant d'installer et exécuter une nouvelle version fraîchement compilée en cas de problèmes.

Les points de test présents sur le circuit proposent les deux tensions de base, 3,3 V et 5 V (utilisé pour la calibration), mais également un signal carré de 5 kHz pour tester la fonction oscilloscope.



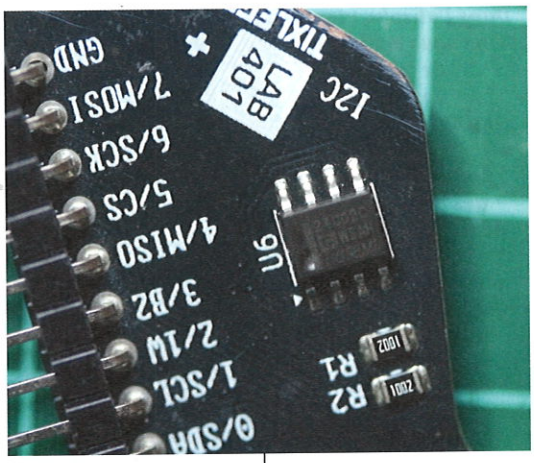
Les sources de l'application DigiLab possèdent une structure relativement facile à suivre. Une telle application repose sur un *framework* interne appelé Furi, fonctionnant à la manière d'un *toolkit* graphique (type GTK+ ou Qt) et permettant au développeur de ne pas avoir à se soucier de l'affichage. Furi utilise les notions de scènes et de vues pour gérer l'interface utilisateur dont le point d'entrée est ici `digilab_app` (`401DigiLab_main.c`), tel que spécifié par la ligne `entry_point` dans le

`application.fam` accompagnant les sources. FAM signifiant *Flipper App Manifests*, un fichier qui décrit les propriétés des composants du *firmware* et ses relations avec le reste du système.

Une application Flipper Zero fonctionne comme une sorte de machine à états où chaque scène est un état, qui peut être un écran, un menu, etc. Les vues sont les éléments graphiques fournis par le *framework* pour construire l'interface, il peut s'agir de menu/sous-menu, de listes, de boîte de texte, d'image, etc. Une action sur un de ces éléments déclenche un événement qui sera traité dans le code sous la forme de fonctions *callback*. C'est également vrai pour l'entrée dans une scène ou la sortie d'une scène.

À partir de cette structure, commune à toutes les applications Flipper Zero, et qu'il est indispensable de clairement comprendre, vous pouvez commencer à explorer les sources divisées en plusieurs fichiers intéressants :

- `401DigiLab_main.c` : le point de départ où se trouve `digilab_app()` et toutes les fonctions d'initialisation ;
- `401DigiLab_main_menu.c` : gestion des menus ;
- `401DigiLab_splash.c` : affichage de l'image d'accueil ;
- `401DigiLab_config.c` : gestion de la configuration ;
- `401DigiLab_probe.c` : gestion de la fonctionnalité « probe » ;
- `drivers/sk6805.c` : pilotage de la LED RGB ;
- `scenes/scope/` : gestion de l'oscilloscope ;
- `scenes/spi/` : fonctions en lien avec le bus SPI ;
- `scenes/i2c/` : gestion de l'i2c ;
- `resources/targets/*.json` : les descriptions au format JSON pour les puces SPI et i2c.

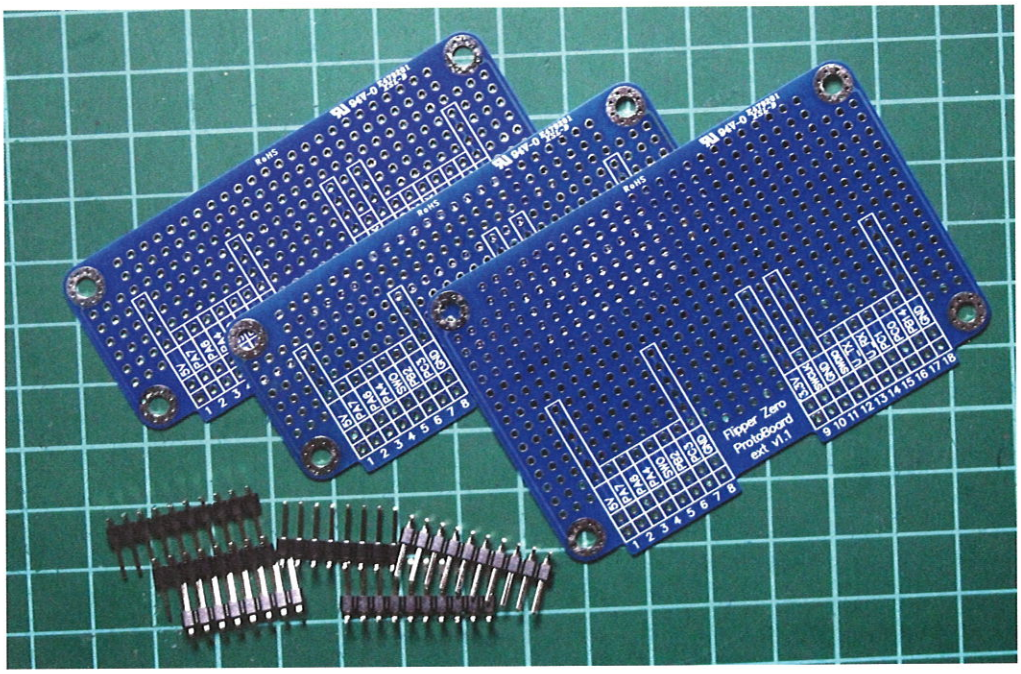


Le DigiLab intègre par défaut une EEPROM i2c de 256 octets 24C02. Celle-ci pourra être utilisée pour les tests et, pourquoi pas, servir à une évolution future de l'application...

Cet article n'étant pas un tutoriel de développement d'applications Flipper Zero, je m'arrêterai là et laisse donc à votre charge le fait de vous jeter à corps perdu dans ces sources. Je vous recommande cependant, avant d'aller dans ce sens, d'étudier les exemples, plus simples, accompagnant les sources du *firmware* officiel **[15]** et, en particulier, le contenu de `applications/examples/example_thermo` et `applications/examples/example_apps_assets`.

CONCLUSION

J'ai acheté mon DigiLab chez Lab401 presque immédiatement après avoir assisté à un *stream* Twitch de Fixoulab **[16]** **[17]** sur le sujet, faisant intervenir son créateur, Tixlegeek. Ce n'est pas un article sponsorisé, puisque ce genre de choses est une hérésie absolue aux éditions Diamond, mais il faut savoir qu'il existe un code de « TIXLEGEEK » utilisable lors de l'achat chez

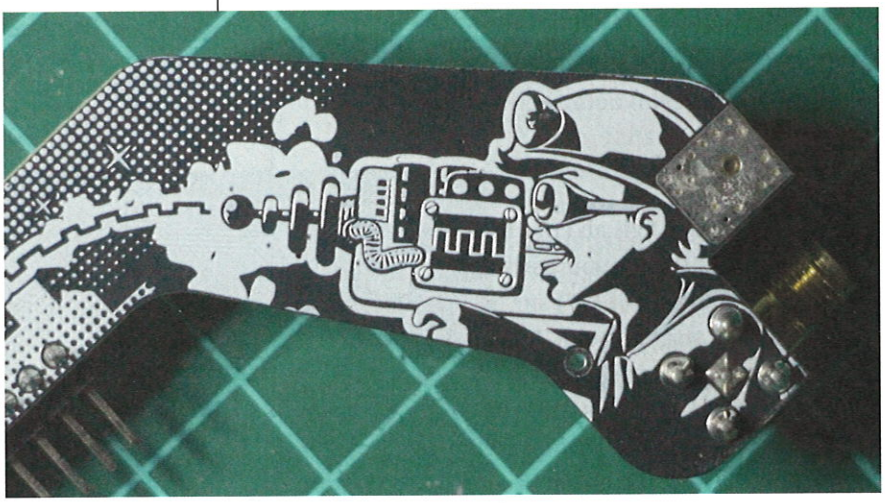


Dans l'absolu, rien ne vous empêche de fabriquer votre propre DigiLab. Plusieurs revendeurs commercialisent des protoboards dédiées au Flipper Zero pouvant servir de base à la création de modules « maison ».

Lab401, donnant droit à une petite réduction, mais, surtout, reversant une participation, bien méritée à mon sens, à Tixlegeek pour sa création.

C'était un achat impulsif, comme souvent pour ce genre de choses en ce qui me concerne, mais finalement je suis relativement satisfait de ce que j'ai réceptionné. Certes, comme précisé plusieurs fois dans ce qui précède, les fonctionnalités sont relativement limitées, mais si l'on y réfléchit, le duo Flipper Zero plus DigiLab forme un ensemble cohérent qui, avec les autres fonctionnalités du Flipper, permettent ponctuellement de se passer d'un PC et/ou d'un équipement difficile à transporter. Imaginez simplement devoir avoir avec vous un *laptop*, un Bus Pirate ou un HydraBus (voir l'article dédié **[18]** dans le numéro 53), un Proxmark 3 et un multimètre...

Tixlegeek n'est pas juste créateur de circuits en tous genres, c'est également un artiste. Tous les dessins présents en sérigraphie sur le circuit et sur le packaging sont ses créations.



Je crois que le seul reproche qu'on peut faire au DigiLab du point de vue de la transportabilité est la série de 10 broches mâles situées à côté de U6, qui vont invariablement rester coincées dans une poche ou un

sac en tissu. Mais, soyons honnêtes, j'ai suffisamment longtemps pesté intérieurement contre la connectique des Arduino pour m'abstenir de dire que des connecteurs femelles auraient été plus judicieux...

Au final, est-ce que je vais intensément utiliser ce duo lorsque je suis chez moi avec tout mon matériel à portée de main ? Probablement pas. Mais en ballade ou en déplacement, oui, absolument. Il sera très certainement avec moi. Et, si j'ai le temps, je pense aussi que c'est une excellente occasion de me pencher sérieusement sur le développement d'applications pour le Flipper Zero, en contribuant moi-même sur les améliorations possibles que j'ai ouvertement pointées du doigt ici... DB

RÉFÉRENCES

[1] **https://lab401.com/fr/products/digilab-pour-flipperzero**

[2] **https://shop.flipperzero.one/products/wifi-devboard**

[3] **https://www.st.com/resource/en/datasheet/stm32wb55rg.pdf**

[4] **https://connect.ed-diamond.com/hackable/hk-060/peripherique-usb-stm32-est-aussi-dans-la-course**

[5] **https://play.google.com/store/apps/details?id=com.flipperdevices.app&hl=fr**

[6] **https://apps.apple.com/fr/app/flipper-mobile-app/id1534655259**

[7] **https://connect.ed-diamond.com/hackable/hk-051/flipper-zero-jouet-ou-outil**

[8] **https://flipperzero.one/downloads**

[9] **https://awesome-flipper.com/firmware/**

[10] **https://lab.flipper.net/apps/401_digilab**

[11] **https://fr.wikipedia.org/wiki/Variance_(mathématiques)**

[12] **https://lab.flipper.net/apps/bme680_flipper_zero**

[13] **https://electronics.stackexchange.com/questions/397569/i2c-response-to-ghost-address-0x5f**

[14] **https://github.com/lab-401/fzDigiLab**

[15] **https://github.com/flipperdevices/flipperzero-firmware**

[16] **https://www.twitch.tv/fixoulab**

[17] **https://www.youtube.com/@fixoulab/videos**

[18] **https://connect.ed-diamond.com/hackable/hk-053/hydrabus-un-outil-pour-tous-les-bus**