DIY Meshtastic avec un Raspberry PI PICO-Zéro

Anthony Le Cren F4GOH - KF4GOH

INTRODUCTION

Meshtastic (1) est un projet open-source qui vise à créer des réseaux de communication décentralisés et autonomes. J'ai déjà écrit plusieurs articles sur le sujet en utilisant la plateforme de programmation officielle (2).

Mais si l'on veut utiliser un microcontrôleur avec un modem LoRa qui n'est pas dans la liste, la seule solution est de recompiler une version personnalisée en modifiant le paramétrage d'un fichier. Seulement voilà, la recompilation nécessite un environnement de développement précis (platform.io couplé la plupart du temps avec Visual Studio Code).

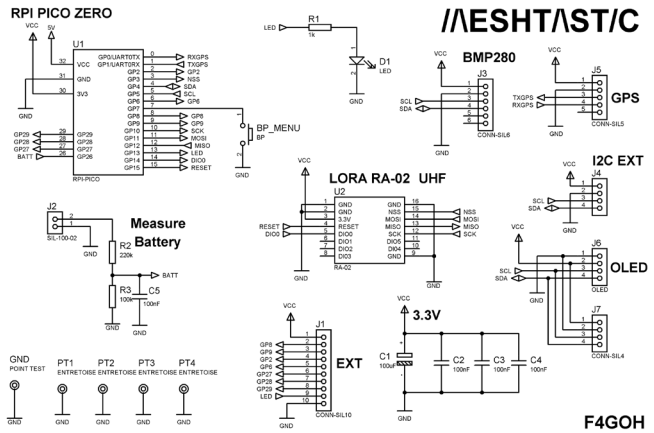
Ce travail d'installation et de recompilation est la plupart du temps réalisé par des OM qui ont une expérience solide de l'environnement de développement.

Heureusement il existe une solution pour recompiler de façon personnalisée le code source du logiciel Meshtastic sans rien installer sur son PC. C'est ce que je propose d'expliquer aujourd'hui à travers la réalisation d'une carte réalisée autour d'un Raspberry PI PICO-Zéro.

Caractéristiques matérielles de la carte :

- Raspberry PI PICO-Zéro RP2040 ;
- Module LoRa RA-02 (QRP 433 MHz) ;
- Capteur I²C BME 680 en option ;
- GPS ATGM336H en option ;
- OLED SSD 1306.

DESCRIPTION DU SCHÉMA STRUCTUREL



Le schéma structurel de la carte

J'ai choisi d'utiliser le microcontrôleur RP2040 et plus particulièrement le Raspberry PI PICO-Zéro. En effet j'ai déjà utilisé ce circuit pour le projet dongle USB CW publié récemment. De plus le tarif est très intéressant, on en trouve pour moins de 2 €. Couplé à un modem Lora RA-02, l'ensemble simple à fabriquer et à programmer est imbattable en termes de coût de revient. Le bus I²C est évidemment présent pour gérer l'afficheur OLED 128x64 ainsi que le capteur BME680. Il est possible d'utiliser un capteur BMP280 ou BME280. La liaison série est aussi disponible pour le récepteur GPS. À noter qu'il n'est pas nécessaire de câbler ces deux modules externes (capteur et GPS) pour une utilisation simple d'envoi et de réception de message.

Le connecteur J2 toujours optionnel permet de mesurer une tension externe. Cela peut être une batterie ou un autre élément d'alimentation. Veillez à changer le diviseur de tension formé par R2 et R3 en conséquence.

Le bouton poussoir permet de naviguer dans les différents menus de l'afficheur.

L'alimentation du module est fournie par la fiche USB type C du microcontrôleur.

En fonction du fournisseur, l'afficheur OLED a deux types de brochages :

- GND, VCC SCL, SDA -> utiliser le connecteur J6
- VCC, GND, SCL, SDA -> utiliser le connecteur J7

En effet, l'ordre des deux broches de l'alimentation est inversé. Une petite vérification évite les mauvaises surprises.

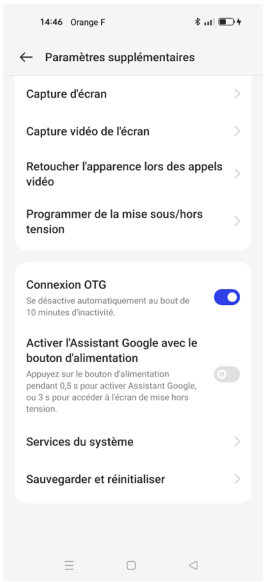
Les fichiers de fabrication sont sur mon Github (5).

PAS DE CONNEXION BLUETOOTH

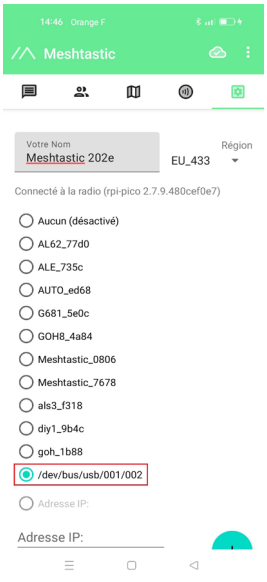
La plupart du temps on configure un module Meshtastic via la connexion Bluetooth du smartphone. Le RP2040 ne possède pas de communication sans fil intégré comme l'ESP32.

Il faudra alors utiliser un câble OTG entre le smartphone et le Raspberry PI PICO. C'est également le smartphone qui fournit l'énergie au module (attention : la plupart du temps il faudra activer le mode OTG dans le smartphone).

Pour rappel, un câble OTG (On-The-Go) est un adaptateur qui permet à un smartphone ou une tablette de se comporter comme un hôte USB, exactement comme un ordinateur. Cela signifie qu'il est possible de connecter des périphériques externes directement à un appareil mobile.

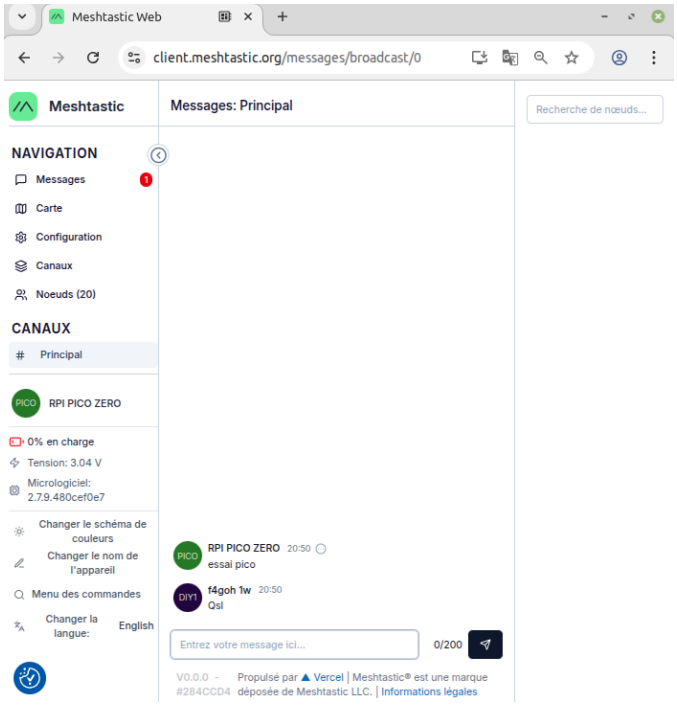


Activation du mode OTG dans le smartphone.



La carte connectée apparaît sous forme d'un périphérique USB connecté.

La deuxième solution est de relier le Raspberry PI PICO à l'ordinateur PC et d'utiliser l'application en ligne (3). L'envoi des messages se fera par une émulation d'une liaison série standard via le connecteur USB type C. Dans ce cas c'est le PC qui fournit l'énergie au module.



Utilisation du Module sur un PC avec client.meshtastic.org

PROGRAMMATION FACILE

Le fichier firmware uf2 tout prêt est disponible sur mon site Github (5).

- Maintenir le bouton BOOTSEL du Pico enfoncé.
- Brancher le câble USB à l'ordinateur tout en gardant le bouton BOOTSEL appuyé.
- Une fois branché, relâcher le bouton. Le Pico apparaîtra comme un disque USB nommé RPI-RP2.
- Glisser-déposer le fichier firmware.uf2 dans ce disque.
- Le Pico redémarre automatiquement et exécute le logiciel.
- L'afficheur OLED doit afficher le menu de sélection de fréquence. (EU_433 pour le RA-02).

RECOMPILATION EN LIGNE

Je recommande de regarder la vidéo de RichBlackHat (4).

La recompilation nécessite que vous ayez au préalable un compte personnel sur Github.

Saisir le lien suivant dans votre navigateur :

https://urls.r-e-f.org/ze356up

Après enregistrement en lien avec votre compte Github, le site gitpod.io va « récupérer » automatiquement les fichiers sources de Meshtastic. Une fois chargé, un bandeau latéral apparaît avec tous les fichiers sources.

Il faudra localiser le fichier variant.h dans le répertoire

`variants/rp2040/rpipico/.`

On trouve la configuration des broches du connecteur GPIO et leurs liaisons avec les différents périphériques conformément au schéma structurel. Il faudra remplacer son contenu par la nouvelle définition de la carte (faire un copier-coller).

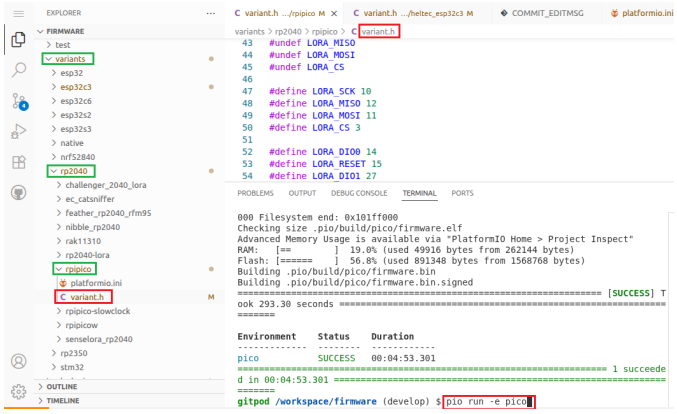
Dans l'invite de commande suivante, taper la commande de compilation en rouge :

```
gitpod /workspace/firmware (develop) $
pio run -e pico
```

Après plusieurs minutes, le fichier tant attendu firmware.uf2 se trouve dans le répertoire :

`.pio/build/pico/src`

Il suffit de le télécharger (clic droit avec la souris sur le fichier firmware.uf2 puis « Download »). Ensuite vient la programmation du fichier dans le RP2040 comme indiqué précédemment dans la partie programmation facile.



Le navigateur Chrome avec gitpod et son firmware Meshtastic.

Ci-dessous : configuration du fichier variant.h en lien avec le schéma structurel.

```
#define ARDUINO_ARCH_AVR
#define SERIAL_MODULE
#define SERIAL_RX_PIN 9
#define SERIAL_TX_PIN 8
#define SERIAL_BAUD 9600
#define UART_PORT uart1
#define HAS_GPS 1
#define GPS_RX_PIN (1u)
#define GPS_TX_PIN (0u)
#define EXT_NOTIFY_OUT 6
#define BUTTON_PIN 7
#define BUTTON_NEED_PULLUP
#define LED_PIN 13
#define BATTERY_PIN 26
#define ADC_MULTIPLIER 3.1
#define
BATTERY_SENSE_RESOLUTION_BITS
ADC_RESOLUTION
```

```
#define USE_RF95
#undef LORA_SCK
#undef LORA_MISO
#undef LORA_MOSI
#undef LORA_CS
#define LORA_SCK 10
#define LORA_MISO 12
#define LORA_MOSI 11
#define LORA_CS 3
#define LORA_DIO0 14
#define LORA_RESET 15
#define LORA_DIO1 27
#define LORA_DIO2 RADIOLIB_NC
#define LORA_DIO3 RADIOLIB_NC

#define HAS_NEOPIXEL
#define NEOPIXEL_COUNT 1
#define NEOPIXEL_DATA 16
#define NEOPIXEL_TYPE (NEO_GRB +
NEO_KHZ800)
```

CONCLUSION

Voilà donc un nouveau projet autour du LoRa agréable à réaliser. La recompilation offre beaucoup de possibilités de personnalisation non seulement logicielle mais aussi matérielle.

Cela encourage les réalisations, ce qui est un domaine prépondérant dans l'activité radio-amateur. Il y aura prochainement une autre carte basée sur le circuit HT-CT62.

L'activité Meshtastic s'agrandit en Sarthe. De nombreux essais sont en cours, menés par toute une équipe d'OM motivés. Le réseau est devenu très fiable. J'espère que cela prendra de plus en plus d'ampleur comme chez nos amis Allemands. Dans tous les cas, ce n'est pas le coût de l'investissement en composants qui freine l'expérimentation de ce réseau en France.

(1) **https://meshtastic.org/**

(2) **https://flasher.meshtastic.org/**

(3) **https://client.meshtastic.org**

(4) **https://urls.r-e-f.org/vi635iu**

(5) **https://urls.r-e-f.org/zg926ds**