Manuel CanSat Raspberry Pi Pico

Table des matières


CANSAT

Introduction

Les laboratoires suivants ont été conçus pour vous initier à certaines des compétences en électronique et en programmation requises pour entreprendre la mission principale de CanSat.

Ce document va droit au but sans être un guide technique complet.
Les références aux guides techniques complets sont indiquées lorsque cela est nécessaire.

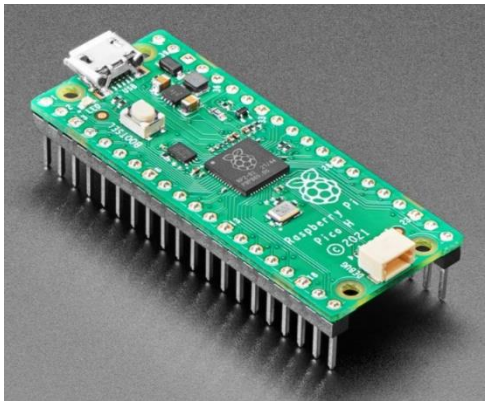
La difficulté des laboratoires est progressive, commençant par des étapes de câblage et de programmation sans aucune soudure.

Si vous n'avez jamais codé en Python, nous vous conseillons de consulter la section Apprendre Python.

Le kit CanSat

Le kit comprend du matériel standard, bon marché et facile à acheter en ligne. Cela permet aux équipes de remplacer facilement les composants cassés et de trouver du soutien et des idées dans la multitude de tutoriels d'enseignement en ligne et de ressources techniques liées à Raspberry Pi Pico et MicroPython. Le kit que nous utilisons dans les laboratoires contient les éléments suivants :

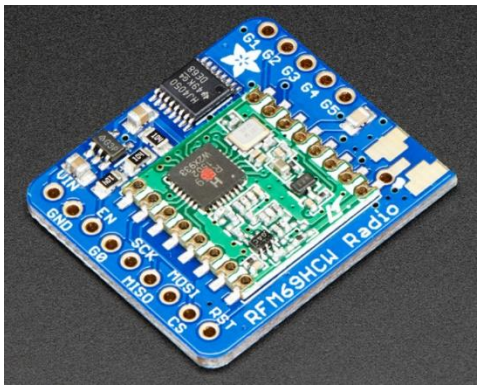
2x Raspberry Pi Pico avec connecteurs



Le Pico est vendu avec ou sans headers pré-soudés. Nous vous donnons 2 Pico avec des headers pour vous aider à vous mettre à niveau et à vous concentrer sur la programmation dès le début.

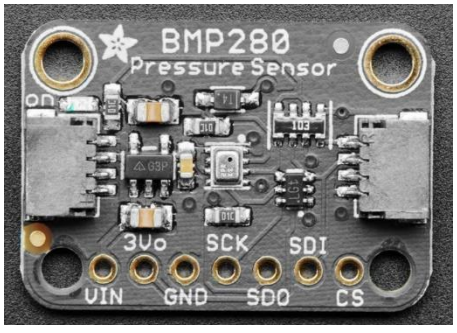
https://www.adafruit.com/product/5525

2x émetteurs-récepteurs radio RFM69HCW



https://www.adafruit.com/product/3071

1x Capteur de pression/température BMP280



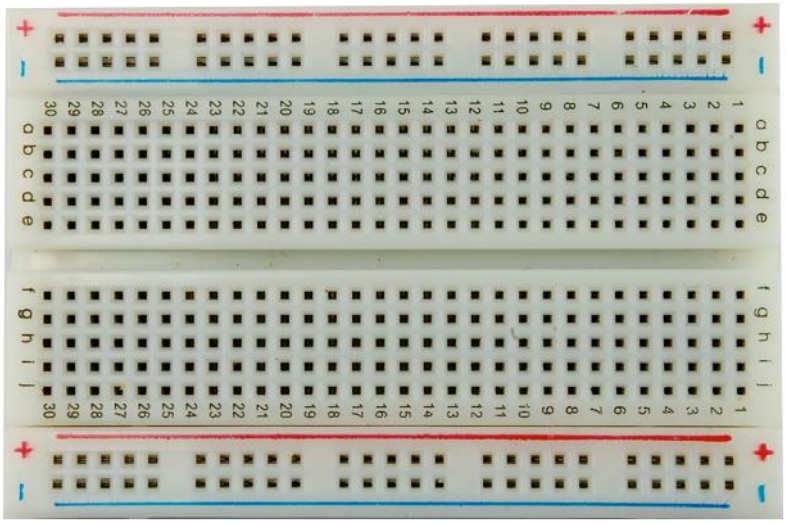
https://www.adafruit.com/product/2651

1x capteur TMP36

Alternative au BMP280, le TMP36 est un capteur de température analogique de base qui émet une tension en fonction de la température ambiante autour du capteur.

https://learn.adafruit.com/tmp36-temperature-sensor

1x breadboard pour le prototypage de circuits



Une planche à pain est une base de construction utilisée pour réaliser des prototypes semi-permanents de circuits électroniques sans aucune soudure.

1x câble USB

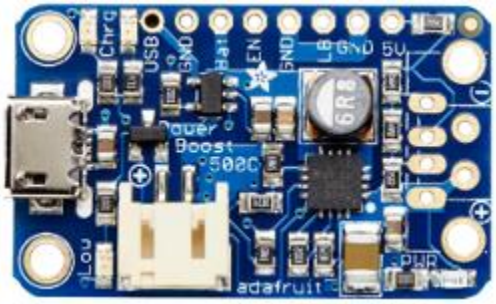
Câble utilisé pour brancher votre Raspberry Pico à un ordinateur pour le programmer ou pour charger la batterie Lipo au lithium.

1x pile au lithium

Batterie 3.7V 1300mAh pour alimenter le Raspberry Pico à l'intérieur de votre CanSat, sans le câble USB.

https://cdn-shop.adafruit.com/datasheets/Li-poly+603562-1300mAh.pdf

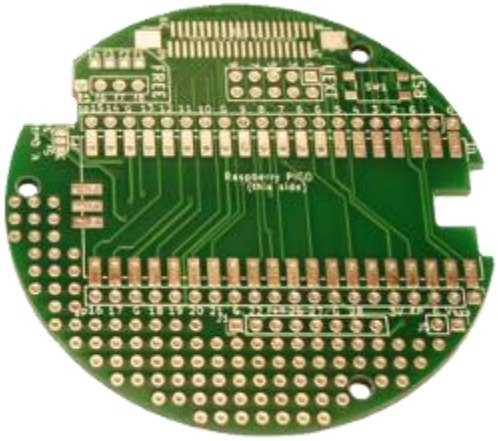
1x convertisseur 5 volts et chargeur de piles au lithium



Carte qui convertit l'alimentation de la batterie au lithium en une source d'alimentation de 5 volts adaptée au Raspberry Pico. Elle peut recharger la pile au lithium lorsque le kit est alimenté par USB.

https://www.adafruit.com/product/1944

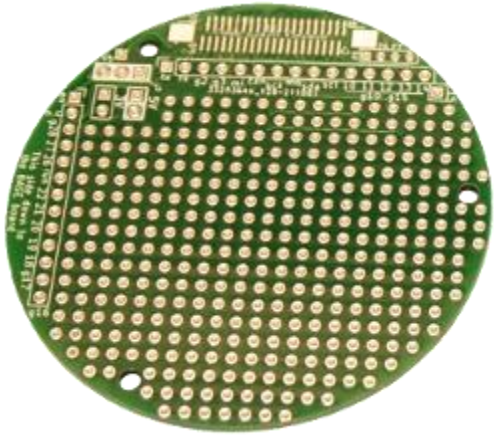
1x Carte de base Cansat pour Pico



Après avoir testé le câblage de vos composants et de leur logiciel sur la planche à pain, vous pouvez souder le Raspberry Pico, la carte convertisseur 5V et l'émetteur radio sur cette carte. Son diamètre correspond au diamètre maximal de CanSat. La carte BMP280 peut être branchée à l'aide du câble JST fourni.

https://shop.mchobby.be/fr/pico-rp2040/2275-carte-de-base-cansat-pour-pico-3232100022751.html

1x Carte d'extension Cansat



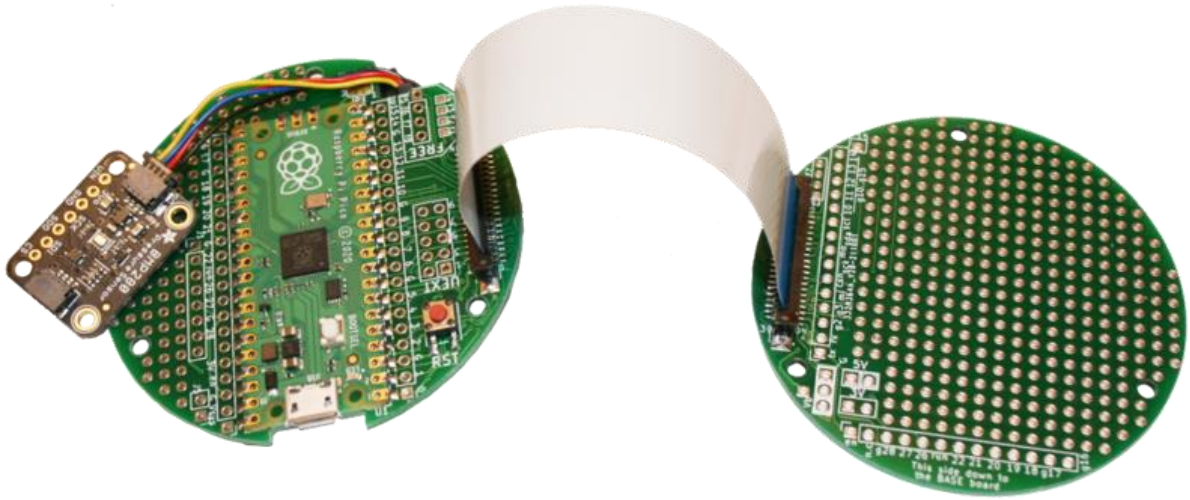
Cette carte d'extension CanSat est utile pour ajouter des composants nécessaires à la mission secondaire, comme un GPS ou d'autres capteurs.

https://shop.mchobby.be/fr/pico-rp2040/2272-carte-de-prototypage-cansat-pour-pico-3232100022720.html

1x ruban FPC

Ce ruban est utilisé pour connecter la carte de base à la carte d'extension.

https://shop.mchobby.be/en/wire-cables/2278-fpc-ribbon-40-pos-p05-1016mm-3232100022782.html



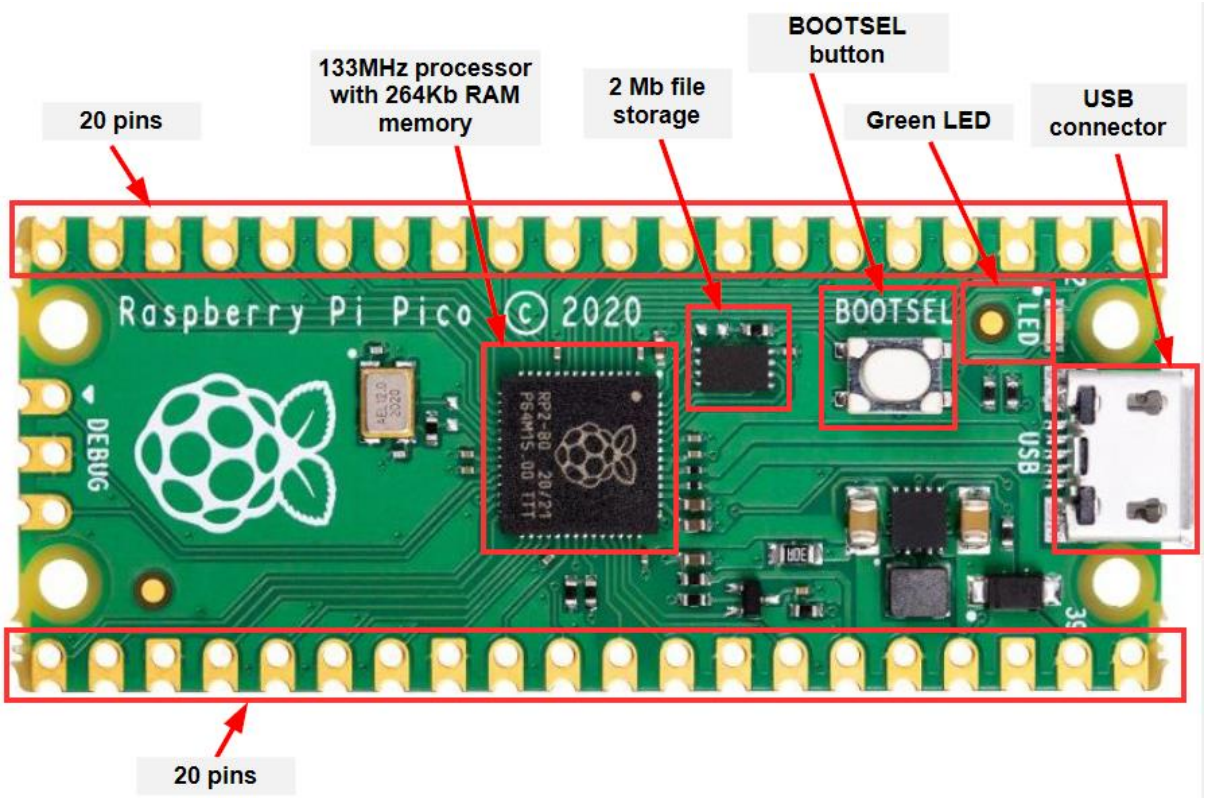
ESERO vous fournit gratuitement un kit complet, mais les différents composants peuvent être achetés séparément chez McHobby.

Rencontre avec Raspberry Pi Pico

Un Raspberry Pi Pico est un microcontrôleur à bas prix. Les microcontrôleurs sont de minuscules ordinateurs, mais ils ne disposent que d'un petit espace de stockage de fichiers (contrairement au disque dur d'un ordinateur classique) et ne disposent pas de périphériques que vous pouvez brancher (par exemple, des claviers ou des écrans).

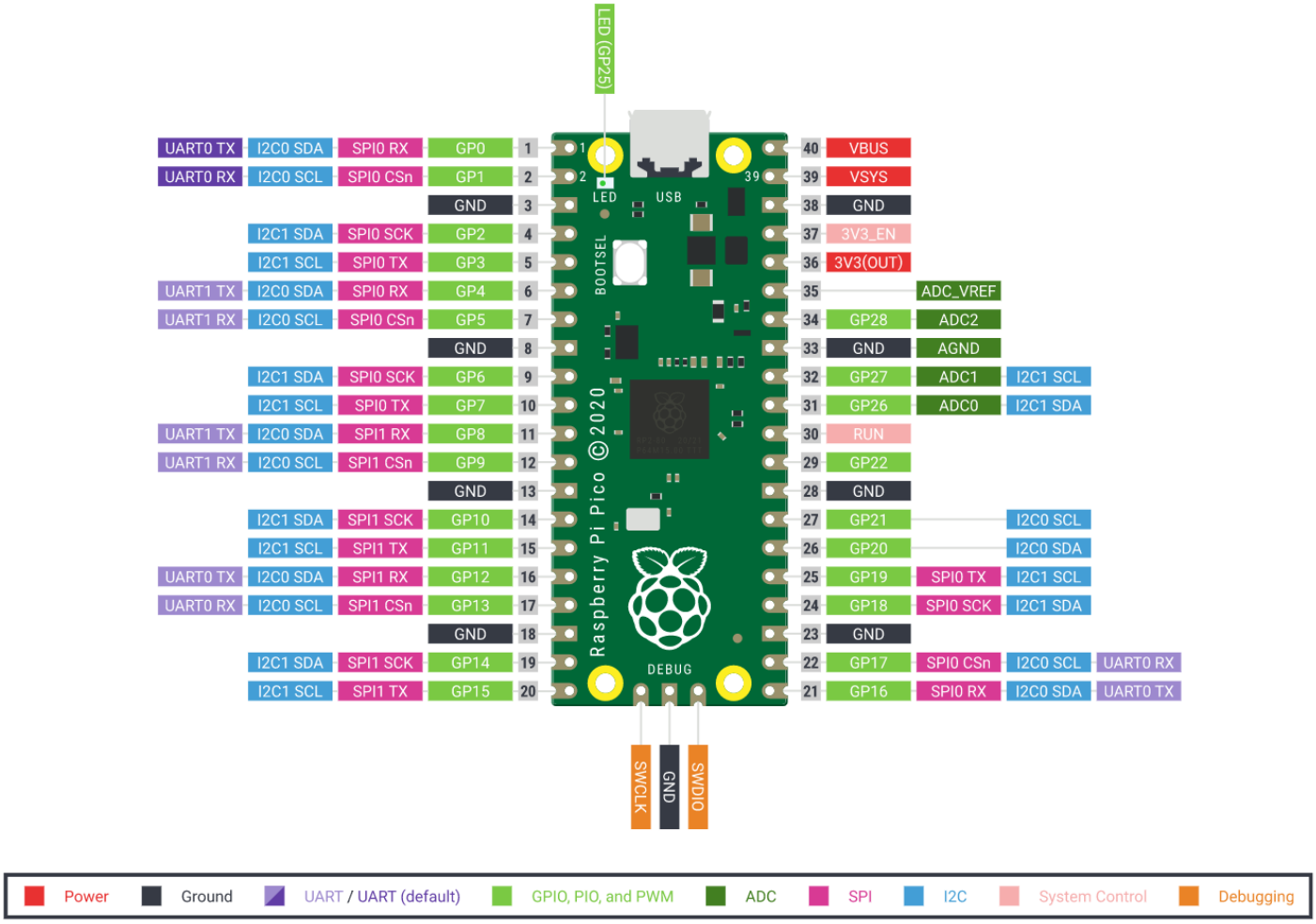
Un Raspberry Pi Pico a

1- Un processeur 133MHz avec 264 kilobytes de mémoire RAM
2- 2 mégaoctets de stockage de fichiers
3- Un bouton BOOTSEL utilisé pour installer MycroPython sur le Pico.
4- Une LED verte
5- Un connecteur USB pour alimenter le Pico et transférer des logiciels ou des données.
6- 2 x 20 broches utilisées pour alimenter le Pico ainsi que pour contrôler et recevoir des entrées de divers appareils électroniques.



Chacune des 40 broches a sa propre fonction.

Si vous avez besoin de connaître les numéros de broches d'un Raspberry Pi Pico, vous pouvez vous référer au schéma suivant ou à [ce site web interactif]



En travaillant avec le Pico, vous n'aurez besoin que de travailler avec :

1- Broches rouges et noires = broches liées à l'alimentation et à la connexion à la terre.
2- Broches violettes = broches liées à la communication UART
3- Broches roses = broches liées au protocole de communication SPI
4- Broches bleues = broches liées au protocole de communication I2C

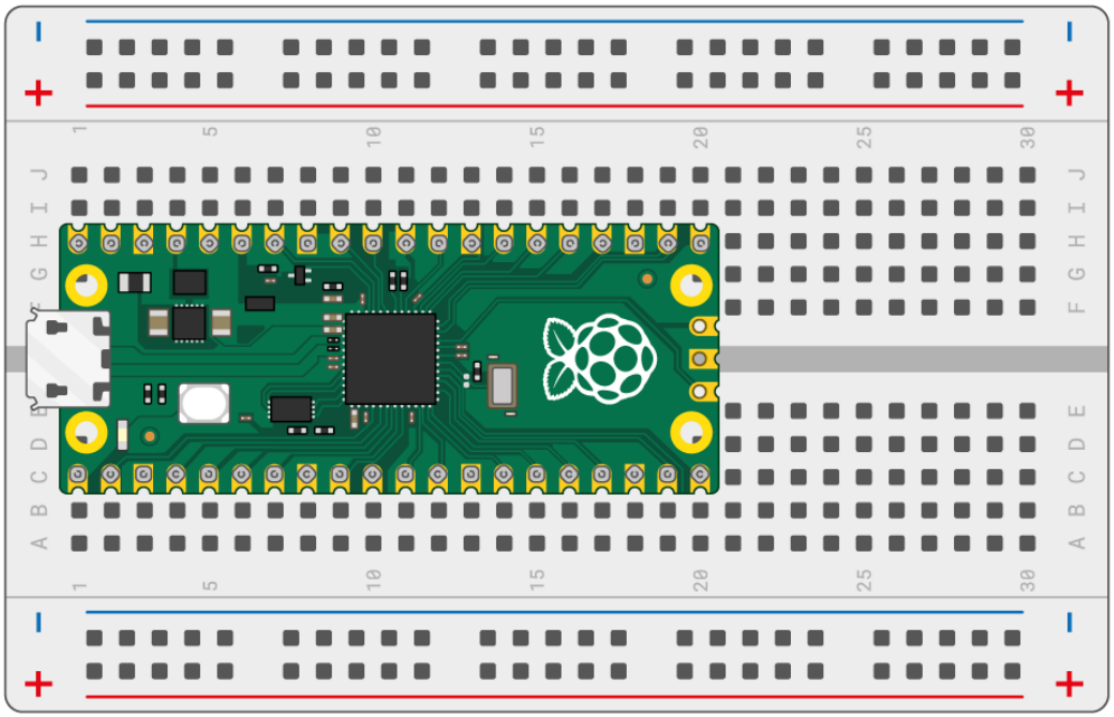
Il y a plusieurs façons d'alimenter votre Pico, en utilisant soit

1- le câble micro-USB (en phase de développement)
2- le convertisseur 5 volts et la batterie au lithium fournis (lorsque le développement sera terminé)

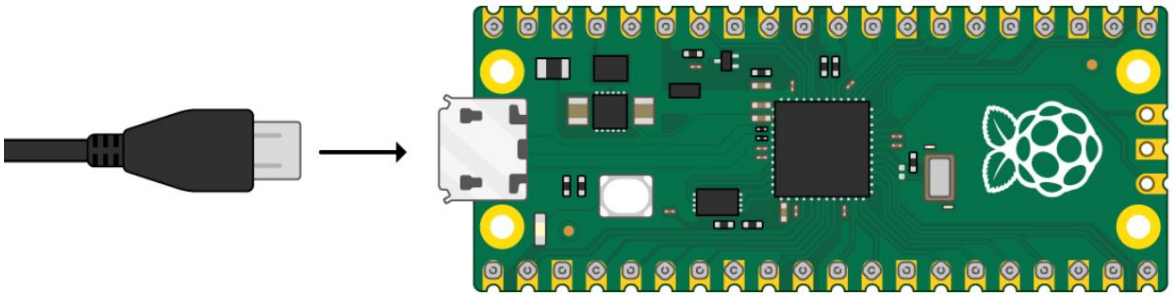
Connectez votre Raspberry Pi Pico à votre ordinateur.

Dans cette section, vous allez connecter un Raspberry Pi Pico à un autre ordinateur et apprendre à le programmer à l'aide de MicroPython.

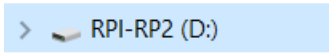
Fixez fermement votre Raspberry Pi Pico sur la planche d'essai fournie comme indiqué sur l'image suivante. Placez-le de manière à ce qu'il soit séparé par le ravin de la planche à pain au milieu.



Branchez le câble micro-USB fourni dans le port situé sur le côté gauche du Pico.



Votre Pico devrait apparaître comme un stockage USB dans votre système de fichiers.

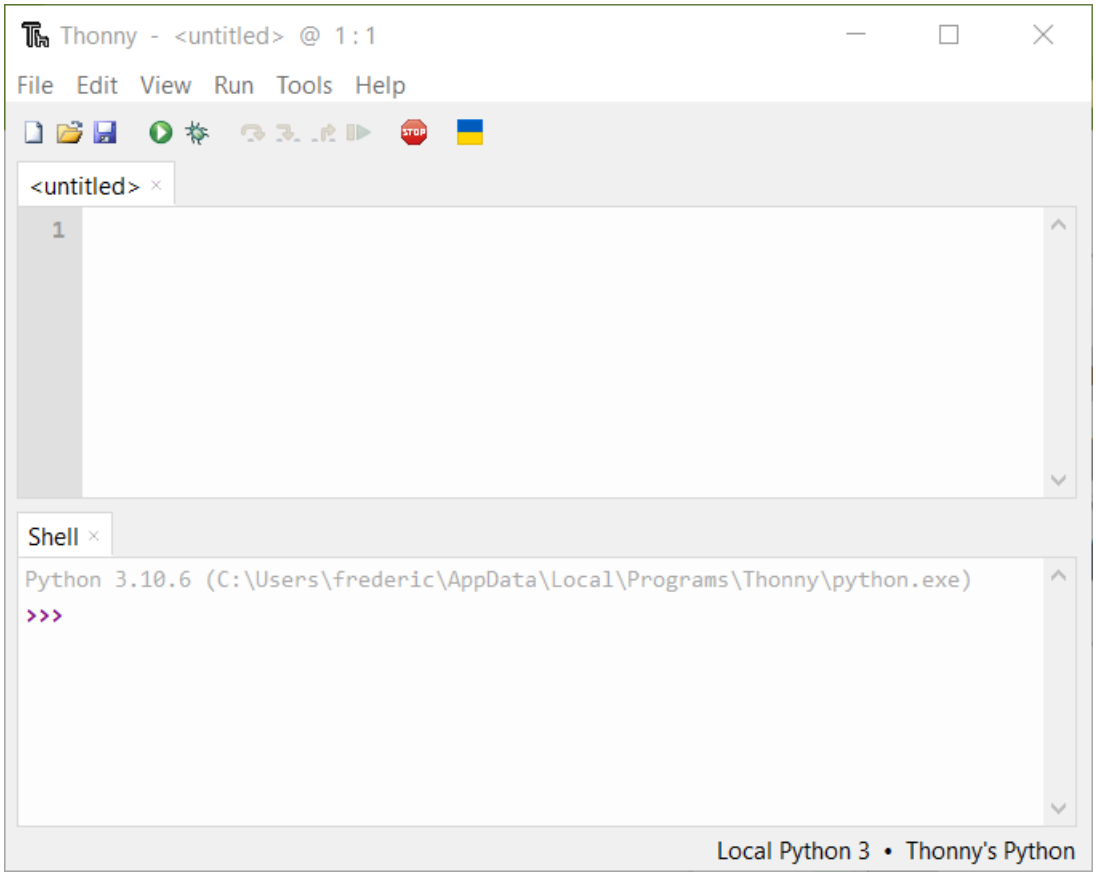


Installez Thonny sur votre ordinateur

Python est un langage généraliste utilisé dans une grande variété d'applications (sciences des données, intelligence artificielle, statistiques, ...) tandis que MicroPython est spécifiquement conçu pour les microcontrôleurs comme le Raspberry Pi Pico utilisé dans notre projet.

Pour éditer, exécuter et déboguer notre code en langage MicroPython, nous allons installer un environnement de développement intégré (IDE) appelé Thonny, disponible à l'adresse https://thonny.org.

Ouvrez Thonny à partir du lanceur d'applications. Cela devrait ressembler à ceci :



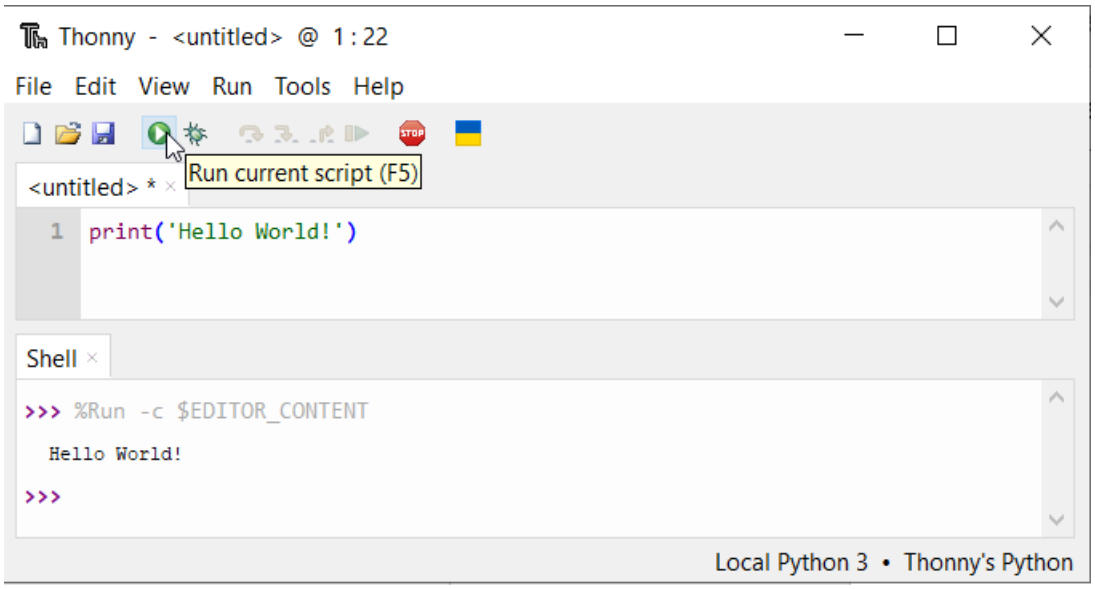
A un moment donné, nous aurons besoin que Thonny soit ouvert deux fois.

- Allez dans le menu "Outils -> Options...->Général".
- Décochez la case "Autoriser une seule instance de Thonny".
- Appuyez sur "OK".

Vous pouvez utiliser Thonny pour écrire du code Python standard. Tapez ce qui suit dans la fenêtre supérieure, puis cliquez sur le bouton Exécuter.

```
print('Hello World!')
```

Le résultat est affiché dans la fenêtre "Shell".



Installer MicroPython sur votre Pico

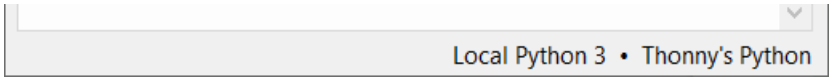
Votre nouveau Raspberry Pi Pico a besoin de MicroPython pour exécuter vos logiciels.

MicroPython est un langage de programmation largement compatible avec Python qui est optimisé pour fonctionner sur un microcontrôleur comme le Raspberry Pico.
Sa documentation peut être trouvée sur le site officiel de MicroPython.

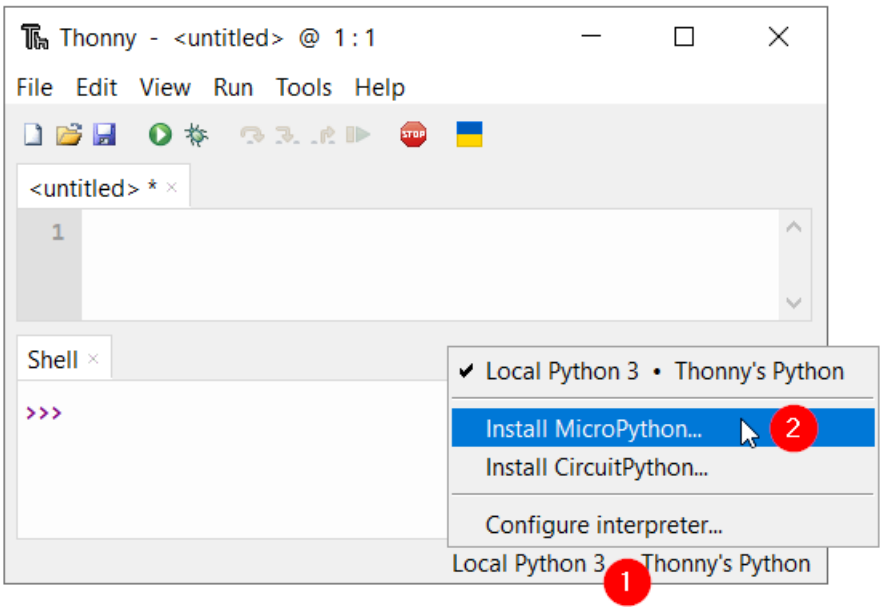
Commencez par débrancher le câble micro-USB de votre ordinateur mais laissez-le connecté à votre Pico.

Appuyez sur le bouton BOOTSEL et maintenez-le enfoncé pendant que vous connectez l'autre extrémité du câble micro-USB à votre ordinateur.

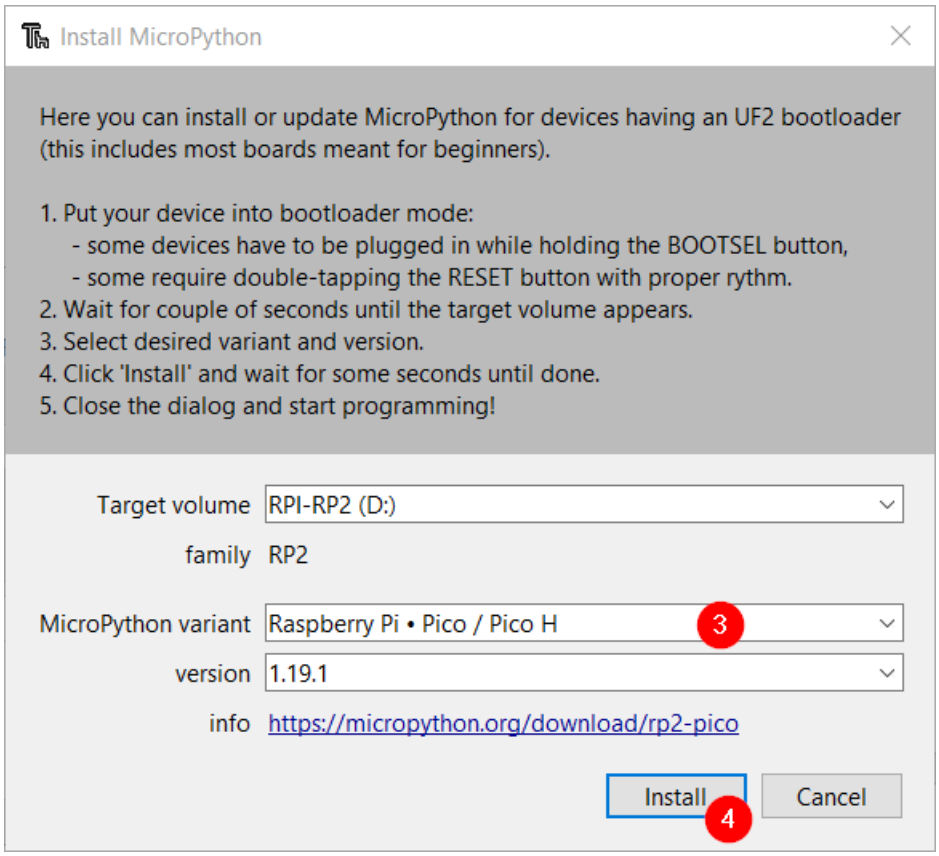
Dans le coin inférieur droit de la fenêtre Thonny, vous verrez la version de Python que vous utilisez actuellement.



Faites un clic gauche sur la version de Python et choisissez "Installer MicroPython...".



Une boîte de dialogue apparaîtra pour installer le firmware MicroPython sur votre Pico. Sélectionnez la bonne variante de MicroPython et cliquez sur le bouton Installer.



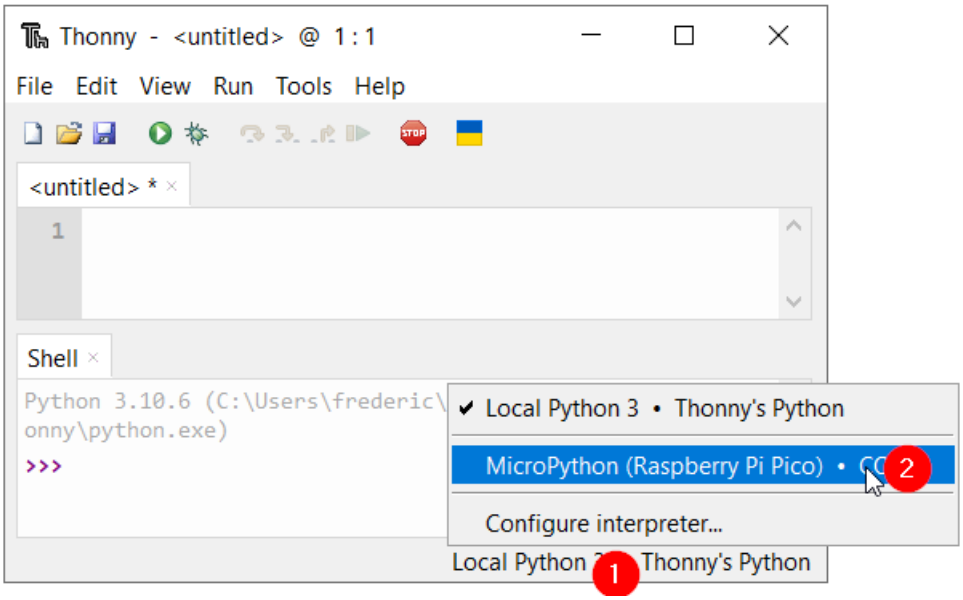
Attendez que l'installation soit terminée et cliquez sur le bouton Fermer.

Vous n'avez pas besoin de mettre à jour le micrologiciel à chaque fois que vous utilisez votre Pico.

La prochaine fois, vous pourrez simplement le brancher sur votre ordinateur sans appuyer sur le bouton BOOTSEL.

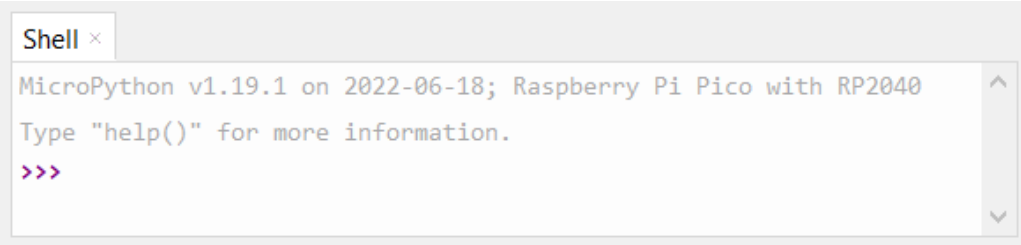
Exécuter votre premier code MicroPython sur votre Pico

Assurez-vous que votre Raspberry Pi Pico est toujours connecté à votre ordinateur. Sélectionnez l'interpréteur MicroPython (Raspberry Pi Pico) en bas à droite.



Regardez le panneau Shell au bas de l'éditeur Thonny.

Vous devriez voir quelque chose comme ceci :



Thonny est désormais capable de communiquer avec votre Pico à l'aide de la boucle REPL (read-eval-print loop), qui vous permet de taper du code Python dans le Shell et de voir directement le résultat.

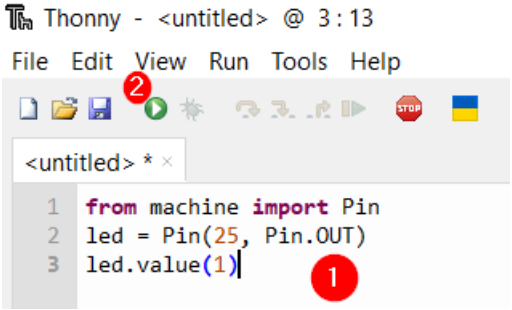
MicroPython ajoute des modules spécifiques au matériel, comme la `machine`, que vous pouvez utiliser pour programmer votre Raspberry Pi Pico.

Créons un objet `machine.Pin` pour jouer avec la LED embarquée, qui est accessible par la broche GPIO 25.

Si vous définissez la valeur de la DEL sur `1`, la DEL de la carte s'allume.

Saisissez le code suivant, en veillant à appuyer sur Entrée après chaque ligne.

```
from machine import Pin
led = Pin(25, Pin.OUT)
led.value(1)
```



Après avoir appuyé sur le bouton vert "Run" , vous devriez voir la LED de la carte s'allumer.

Changez le code et réglez la valeur de la DEL sur 0 pour éteindre la DEL.

```
led.value(0)
```

Allumez et éteignez la LED autant de fois que vous le souhaitez.

Conseil : vous pouvez utiliser la flèche vers le haut du clavier pour accéder rapidement aux lignes précédentes.

Nous avons dit que le LEP embarqué est connecté à la broche 25 du GPIO, mais qu'est-ce que le GPIO ?

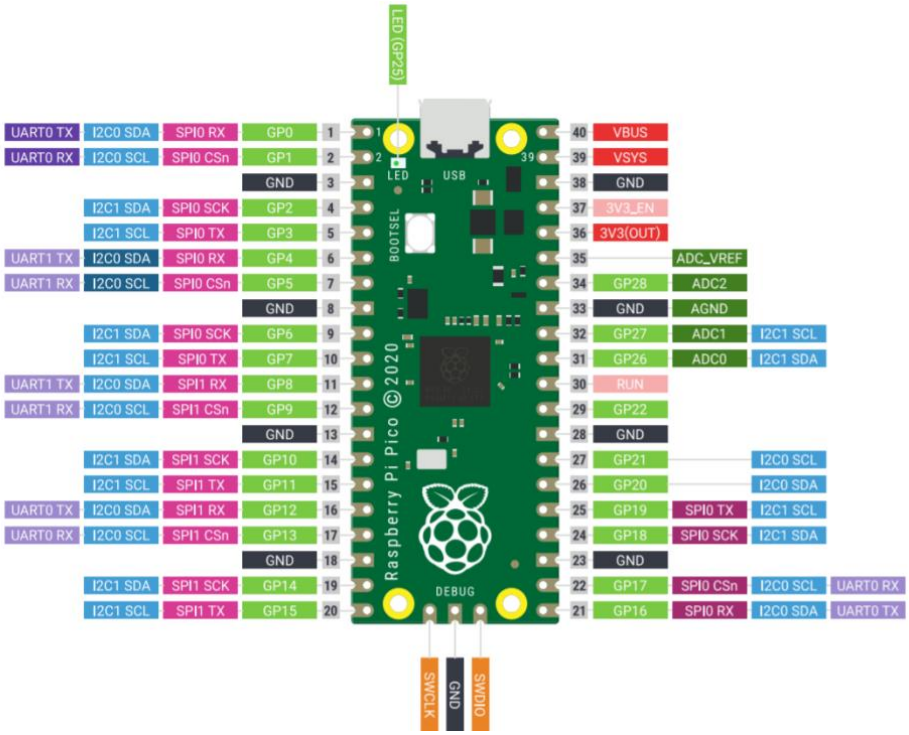
Entrée/sortie à usage général (GPIO)

Les broches GPIO du Raspberry Pi permettent aux tensions externes d'être lues par le logiciel et elles permettent également aux tensions externes d'être réglées par le logiciel. Les GPIO peuvent être numériques ou analogiques. Les broches analogiques peuvent être définies comme des broches numériques, mais les broches numériques restent numériques. Les broches analogiques sont uniquement des broches d'entrée.
Pour notre Raspberry Pi 3,3V, toute tension inférieure à 2,5V est interprétée comme "fausse" et inversement toute tension supérieure à 2,5V est interprétée comme vraie (jusqu'à 3,3V). Il en va de même pour les signaux de sortie. Une sortie "True" (vraie) réglera la tension de la broche à 3,3V et la sortie "False" (fausse) réglera la tension de la broche à 0V.

Les broches GPIO peuvent être utilisées comme des ports d'entrée ou de sortie et ceci est défini par le logiciel.

Le Pi Pico possède 28 ports GPIO, comme le montrent les cases vertes du diagramme

suivant. De nombreuses broches sont polyvalentes et peuvent également être utilisées pour d'autres interfaces (UART, SPI, I2C), elles sont représentées par les cases multicolores à côté des cases vertes dans le diagramme. Le
lien suivant contient le brochage : https://datasheets.raspberrypi.org/pico/Pico-R3-A4-Pinout.pdf



https://datasheets.raspberrypi.org/pico/Pico-R3-A4-Pinout.pdf

Utilisations typiques de GPIO CanSat :
Entrées : Capteurs basés sur la marche/arrêt, interrupteurs, boutons, capteurs de déploiement
Sorties : LEDs d'état, servos de base (PWM est mieux), activation de capteurs, réinitialisation de capteurs connectés par d'autres signaux.

Si vous voulez écrire un programme plus long, il est préférable de l'enregistrer dans un fichier. C'est ce que vous ferez dans la section suivante.

Ecrire votre premier fichier microPython pour contrôler la LED embarquée

Le Shell est utile pour tester des commandes rapides. Cependant, il est préférable de placer les programmes plus longs dans un fichier.

Thonny peut enregistrer et exécuter des programmes MicroPython directement sur votre Raspberry Pi Pico.

Dans cette étape, vous allez créer un programme MicroPython pour allumer et éteindre la LED de la carte dans une boucle, en utilisant les broches GPIO.

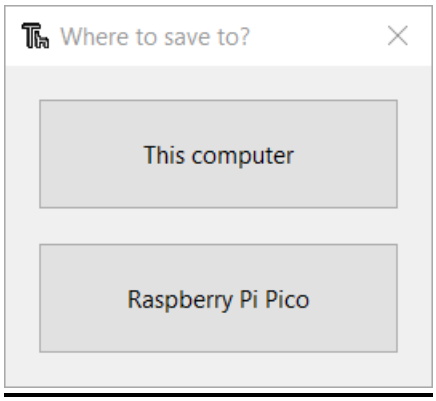
Retournez dans Thonny et cliquez dans le volet principal de l'éditeur de .

Entrez le code suivant pour faire basculer la LED.

```
# importer les bibliothèques préexistantes nécessaires
from machine import Pin
# Déclarer une variable nommée "led", la relier à la broche numéro 25 et la
définir comme sortie
led = Pin(25, Pin.OUT)
# Changez l'état de la led de led.value(0) à led.value(1) et vice versa.
led.toggle()
```

La documentation complète de la bibliothèque "Pin" se trouve dans la documentation officielle de MicroPython.

Cliquez sur le bouton **Save** pour enregistrer votre code et l'écran suivant apparaîtra :



Choisissez "Raspberry Pi Pico" et nommez le fichier "blink.py".

Conseil : vous devez saisir l'extension de fichier . py pour que Thonny reconnaisse le fichier comme un fichier Python. Thonny peut enregistrer votre programme sur votre Raspberry Pi Pico et l'exécuter.

Vous devriez voir la LED embarquée passer de l'état allumé à l'état éteint chaque fois que vous cliquez sur le bouton Run.

Mais que faire si vous voulez voir la LED clignoter sans avoir à cliquer sur le bouton Run à plusieurs reprises ?

Pour cela, nous allons utiliser une boucle "while" et la fonction "sleep".

Prenez soin d'indenter le code avec 4 espaces à l'intérieur de la boucle while pour que MicroPython sache que ces lignes font partie de la boucle while.

Mettez à jour votre code pour qu'il ressemble à ceci :

```
# importer les bibliothèques préexistantes nécessaires
from machine import Pin
from time import sleep
# Déclarer une variable nommée "led", la relier à la broche numéro 25 et la
définir comme sortie
led = Pin(25, Pin.OUT)
```

```
# Prenez soin d'indenter le code avec 4 espaces à l'intérieur de la boucle while
pour que MicroPython sache quelles lignes font partie de la boucle while.
while True:
    # Changez l'état de la led de led.value(0) à led.value(1) et vice versa.
    led.toggle()
    # Arrêter l'exécution du programme pendant une demi-seconde
    sleep(0.5)
```

Vous pouvez également utiliser le module Timer (première ligne ci-dessous) pour définir une minuterie qui exécute une fonction à intervalles réguliers. Mettez à jour votre code pour qu'il ressemble à ceci :

```
# importer les bibliothèques nécessaires préexistantes
from machine import Pin, Timer

# Déclarer une variable nommée "led", la relier à la broche numéro 25 et la
définir comme sortie
led = Pin(25, Pin.OUT)
# Déclarer une variable timer pour gérer le timing des périodes et des événements
timer = Timer()

# Déclarez une fonction nommée "blink" qui fait basculer l'état de la LED.
def blink(timer) :
    # # Changer l'état de la led de led.value(0) à led.value(1) et vice versa.
    led.toggle()

# Configurez le timer pour qu'il appelle la fonction de clignotement prédéfinie
toutes les 2,5 secondes.
timer.init(freq=2.5, mode=Timer. PERIODIC, callback=blink)
```

La documentation complète de la bibliothèque "Timer" se trouve dans la documentation officielle de MicroPython.

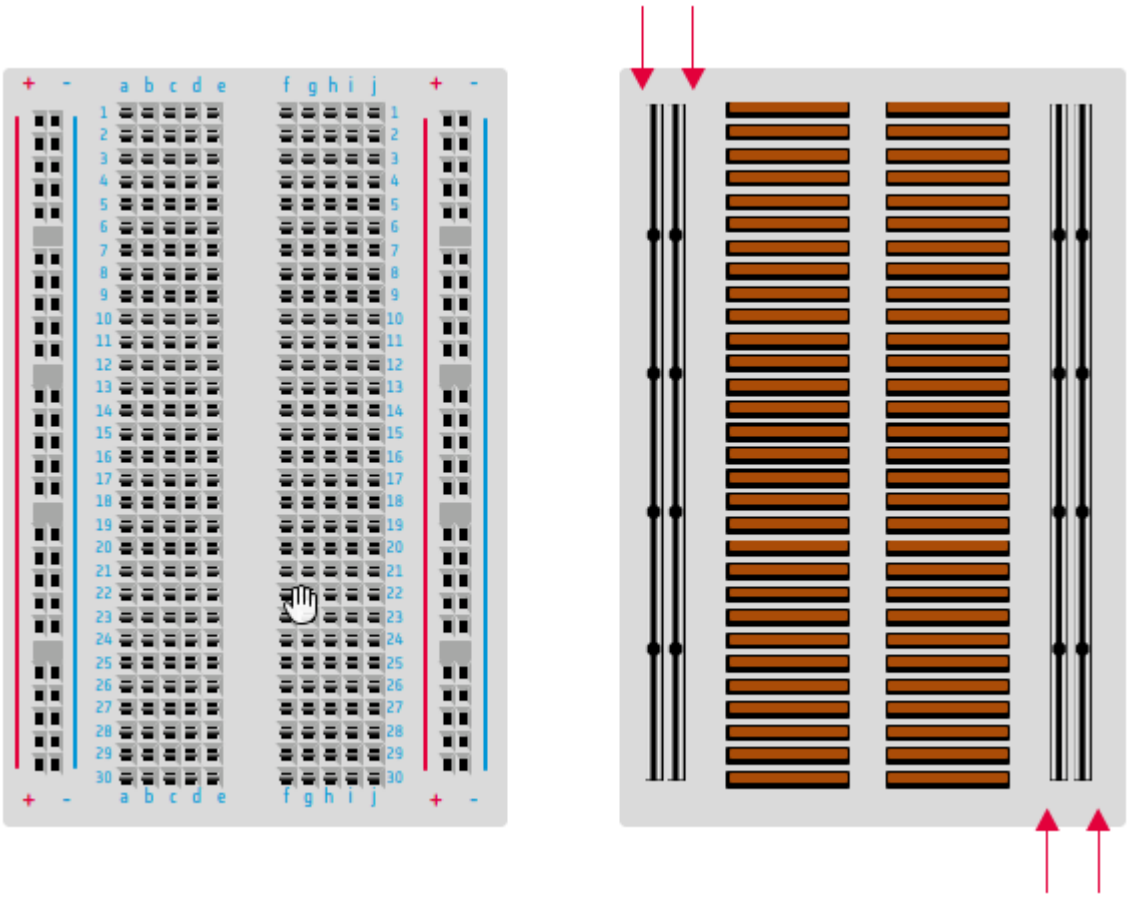
Cliquez sur **Run** et votre programme fera clignoter la LED jusqu'à ce que vous cliquiez sur le bouton **Stop**.

Conseil : Si vous débranchez/rebranchez votre Pico, vous devrez appuyer à nouveau sur le bouton **Run** pour exécuter le programme. Mais si vous nommez votre programme "**main.py**" sur votre Pico, il s'exécutera automatiquement lorsque le Pico sera mis sous tension.

Développement de logiciels et test de câblage

Rencontre avec une planche à pain

Pendant que vous apprenez les bases du Pico et des capteurs, il est préférable d'utiliser une planche d'essai sans soudure, car toute erreur commise lors de la construction de votre circuit peut être facilement corrigée.
Une planche d'essai est un outil simple qui peut être utilisé pour relier des composants électriques entre eux.



Les broches des composants électriques peuvent être placées dans les bornes du tableau. Au centre, les rangées sont reliées horizontalement. Cela signifie par exemple que les deux broches d'une résistance doivent être placées dans des rangées différentes, sinon elle formera un circuit fermé avec elle-même.

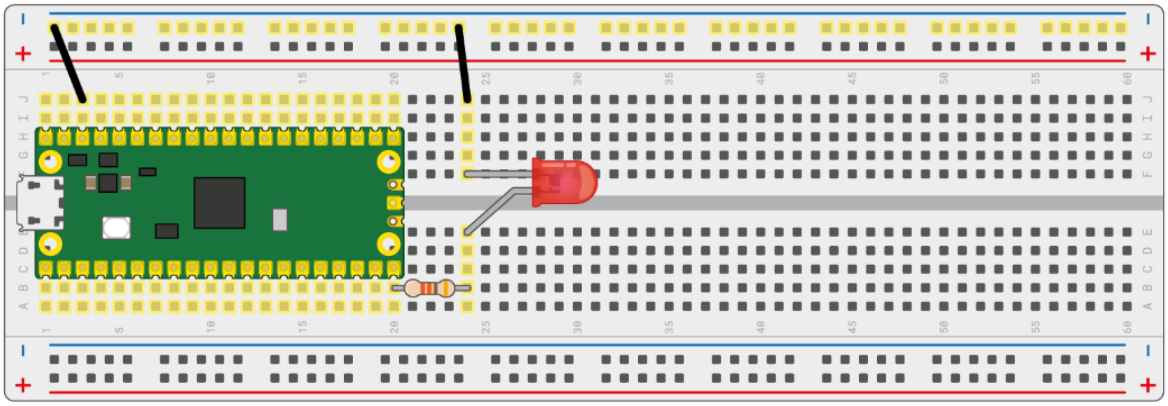
Il est très important de faire un croquis de votre circuit avant de le connecter et de l'alimenter, car vous risqueriez de casser les composants. Les colonnes extérieures de la carte sont connectées en colonnes, plutôt qu'en rangées. En général, elles sont utilisées pour fournir des connexions de masse et de tension.

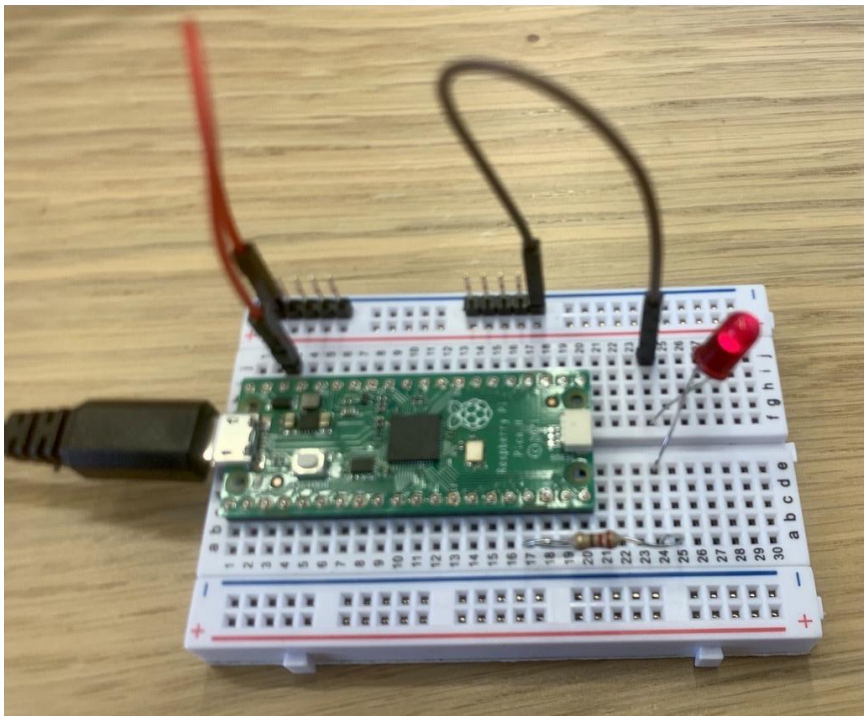
Faire clignoter une LED sur la planche à pain

Vous apprendrez à contrôler une LED externe.

Utilisez une résistance de 220 ohms, une LED, des cavaliers femelles et quelques connecteurs pour connecter votre Raspberry Pi Pico sur votre planche à pain comme le montre l'image ci-dessous.

Notez comment la LED est connectée à GPIO 15 d'un côté (le dernier en bas à gauche comme vous pouvez le voir dans le diagramme de brochage du Pico) et à la broche de masse du Pico de l'autre côté.





Dans Thonny, réutilisez le code de la section précédente, mais au lieu de GPIO 25, utilisez GPIO 15.

```
....
# Déclarer une variable nommée "led", la relier à la broche numéro 15 et la
définir comme sortie
led = Pin(15, Pin.OUT)
....
```

Dans cet exemple, nous avons choisi de connecter la LED au GPIO 15 mais vous pouvez utiliser un autre GPIO si vous le souhaitez.

Testez vos cavaliers

Votre kit contient plusieurs cavaliers mâles et femelles pour tester le câblage de vos composants sur votre planche à pain avant de souder vos composants sur la carte de base CanSat.

Parfois, à cause de problèmes d'usine, il arrive que certains de ces cavaliers soient cassés.

Pour vous épargner quelques maux de tête et du temps, nous vous conseillons vivement de tester tous vos cavaliers avec l'exercice précédent "faire clignoter une LED externe" en remplaçant les 2 cavaliers par les autres cavaliers fournis avec le kit.

Vous pouvez également tester les cavaliers avec le testeur de continuité d'un multimètre.

Mesure de la température et de la pression

La puce BMP280 fournie avec le kit mesure la pression atmosphérique ainsi que la température.

Installer la bibliothèque python BMP280

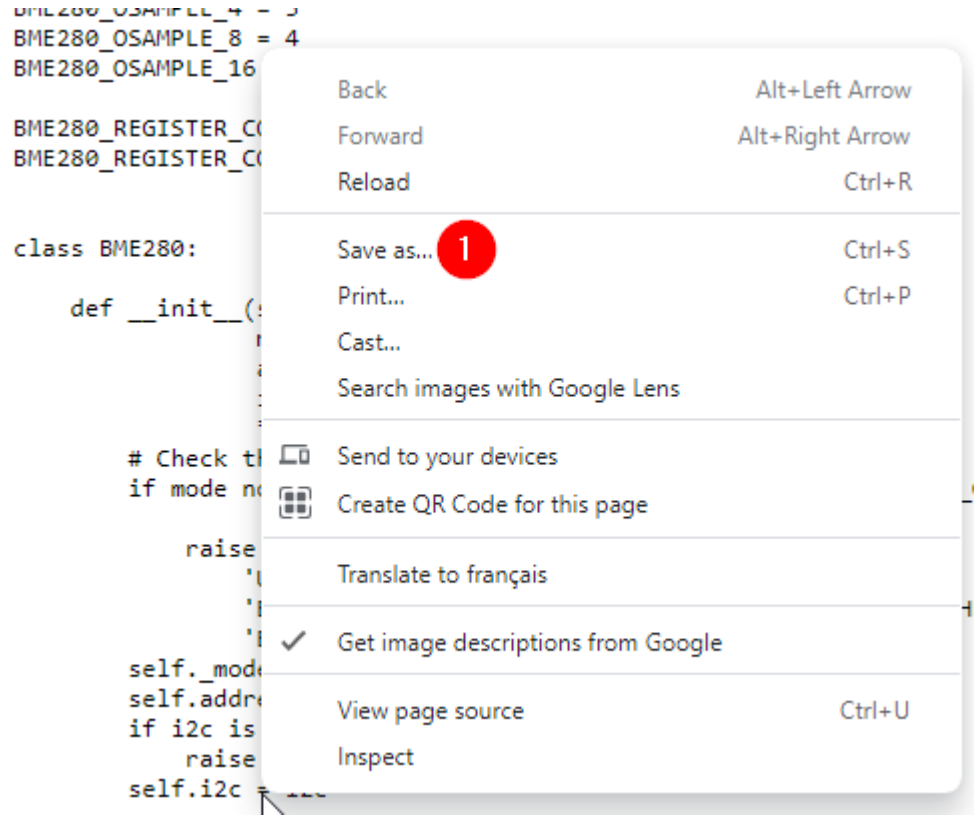
Micro Python fournit des fonctionnalités intégrées pour gérer le Pi Pico, mais celles-ci peuvent être étendues grâce à l'utilisation de bibliothèques tierces. Il s'agit de bibliothèques produites par des fabricants, des fournisseurs et la communauté Micro Python dans le but d'utiliser des périphériques supplémentaires avec le Pi Pico. Ces bibliothèques réduisent la complexité de l'utilisation de périphériques externes en fournissant des fonctions de haut niveau pour interagir avec les périphériques qu'elles prennent en charge.

Le kit est composé de plusieurs capteurs pour lesquels nous utiliserons des bibliothèques Micro Python spécifiques que nous pouvons trouver sur internet.

Pour interagir avec la BMP280, nous avons besoin d'une bibliothèque python dédiée qui peut être téléchargée via ce lien :

https://raw.githubusercontent.com/mchobby/esp8266-upy/master/bme280-bmp280/bme280.py

Cliquez avec le bouton droit de la souris sur le fichier, et choisissez "Enregistrer sous..." comme indiqué dans la capture d'écran ci-dessous :

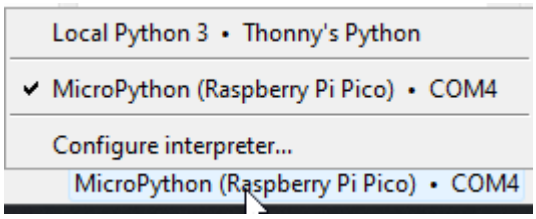


Enregistrez le fichier sur votre PC et gardez le nom "bme280.py" en minuscules.

En utilisant Thonny, la bibliothèque doit ensuite être transférée de votre PC au répertoire /lib du Raspberry Pi Pico.

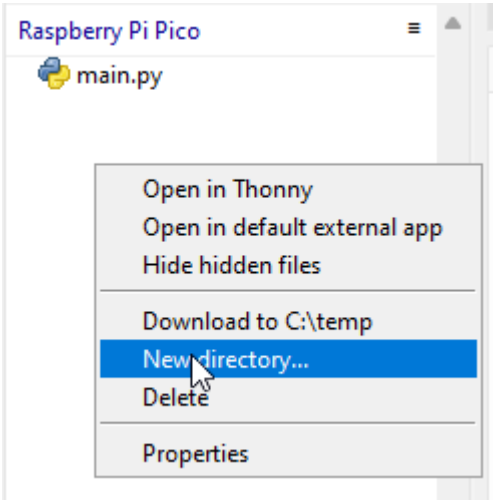
Assurez-vous que le Pico est connecté à votre PC et allumé.

Dans Thonny, assurez-vous que vous êtes connecté au Pico en cliquant sur le menu en bas à droite.

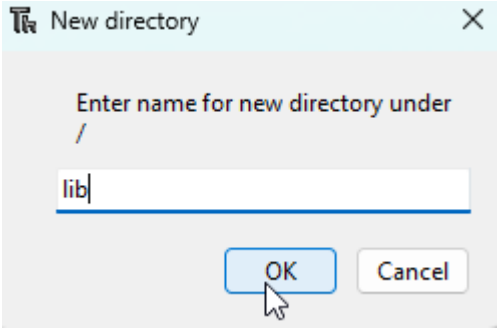


Dans le menu "View", assurez-vous que l'option "Files" est cochée.

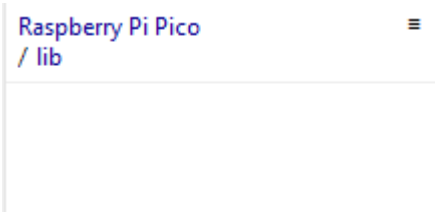
Ensuite, dans la fenêtre "Raspberry Pi Pico" en bas à droite, faites un clic droit et sélectionnez "nouveau répertoire...".



Entrez "lib" (**en minuscules**) et appuyez sur ok.

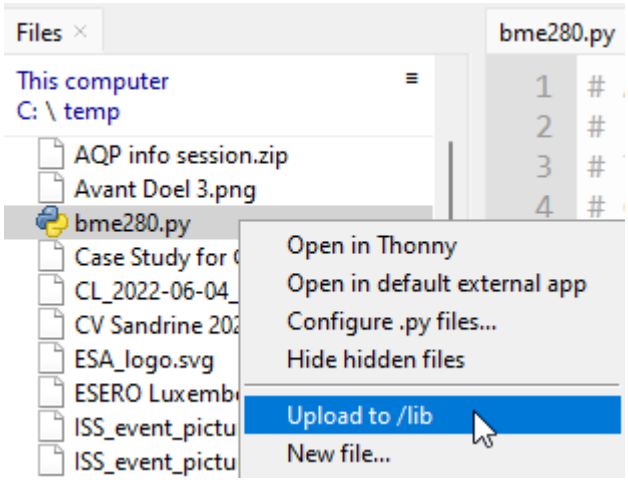


Puis double-cliquez sur le répertoire "lib" qui est vide pour l'instant



Dans la fenêtre "Fichiers" en haut à gauche, naviguez jusqu'à l'emplacement où vous avez enregistré le fichier bme280.py.

Faites un clic droit sur le fichier bme280.py et appuyez sur "upload to /lib" pour installer la bibliothèque sur le Pico.

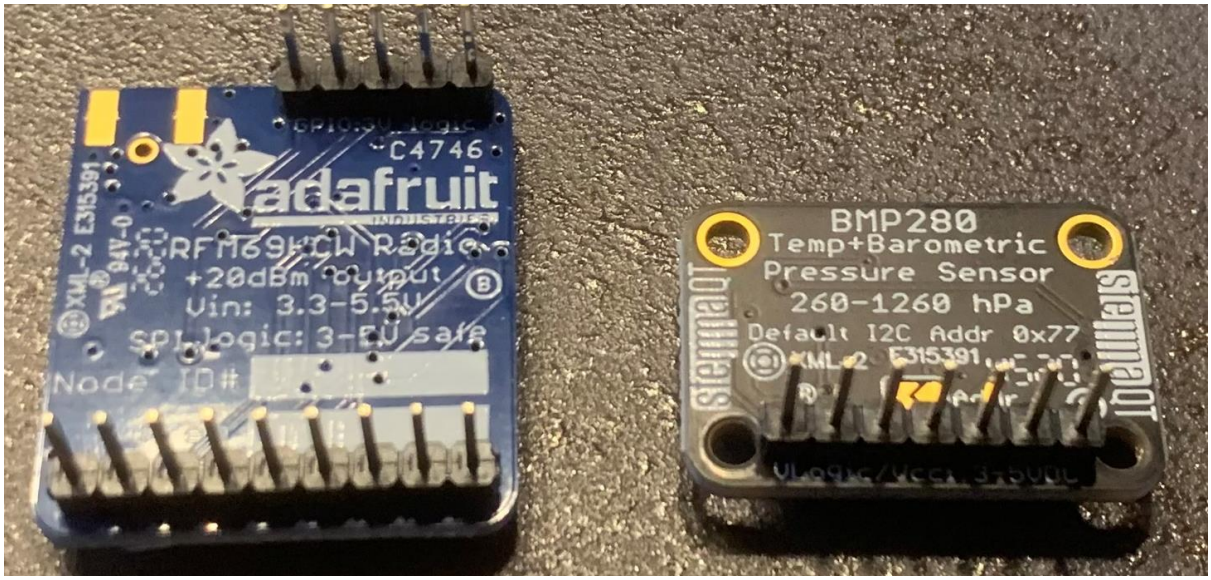


Souder les broches des composants

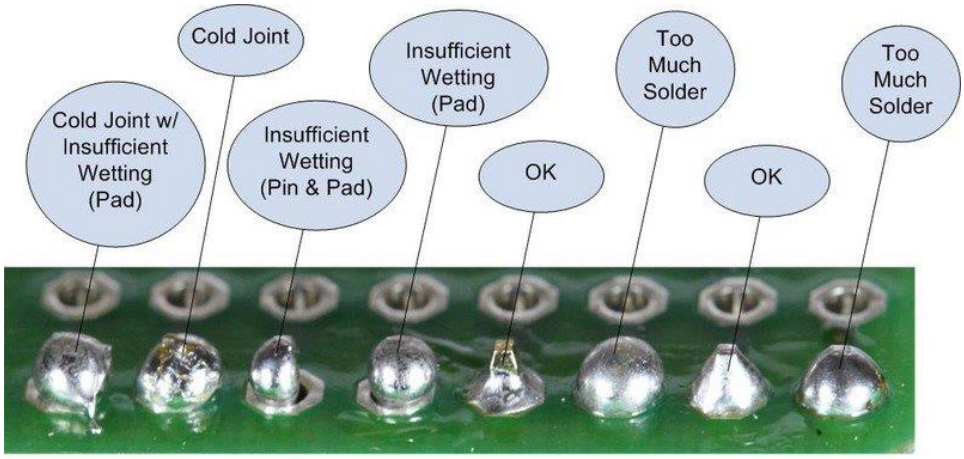
AVERTISSEMENT : Veuillez d'abord lire attentivement le guide de l'excellente soudure d'Adafruit.

Le BMP280 ainsi que le module radio RFM96W pour la station au sol ont besoin de broches d'en-tête soudées afin de pouvoir être connectés aux planches d'essai.

Les broches de la carte BMP280 et du module radio RFM96W qui sera utilisé pour la station au sol doivent être soudées avec des broches d'en-tête.



Pour le deuxième module radio RFM96W, soudez **des cavaliers femelles** sur ses broches, et **non sur les têtes**, afin de pouvoir facilement connecter le module sur la planche à pain ainsi que sur la carte de base CanSat par la suite.



https://learn.adafruit.com/adafruit-guide-excellent-soldering/common-problems

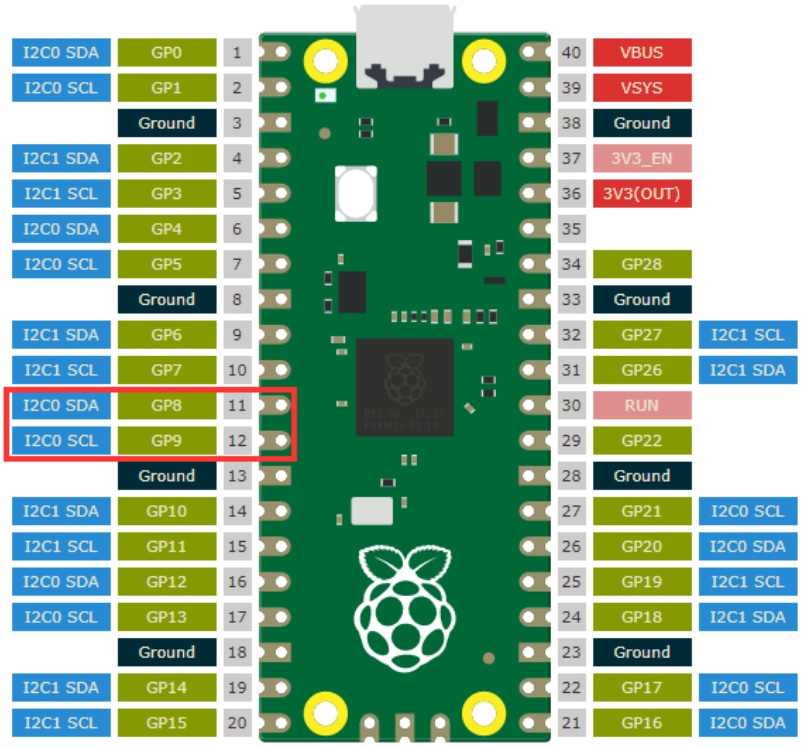
Connexion du capteur de pression/température BMP280 à l'aide d'I2C

Avant de pouvoir lire les données du capteur, nous devons le connecter au Pi.
Pour ce faire, nous devons d'abord souder des connecteurs sur ses broches, comme expliqué à la page 8 de la documentation du BMP280.

La documentation explique que nous avons besoin d'une alimentation de 3,3 V pour l'alimenter. Nous pouvons utiliser les broches 3V3 et GND (masse) du Pico pour cela. La documentation explique également que nous pouvons nous y connecter en utilisant le bus I2C. Le bus I2C nécessite la connexion de deux câbles de données, ce qui signifie que nous devons connecter quatre câbles au total.

1. En utilisant le site web de brochage interactif, localisez la broche d'alimentation 3V3 sur le Pico.

2. Toujours à l'aide du site Web de brochage interactif, localisez les broches I2C (0) sur le Pico (voir ci-dessous).



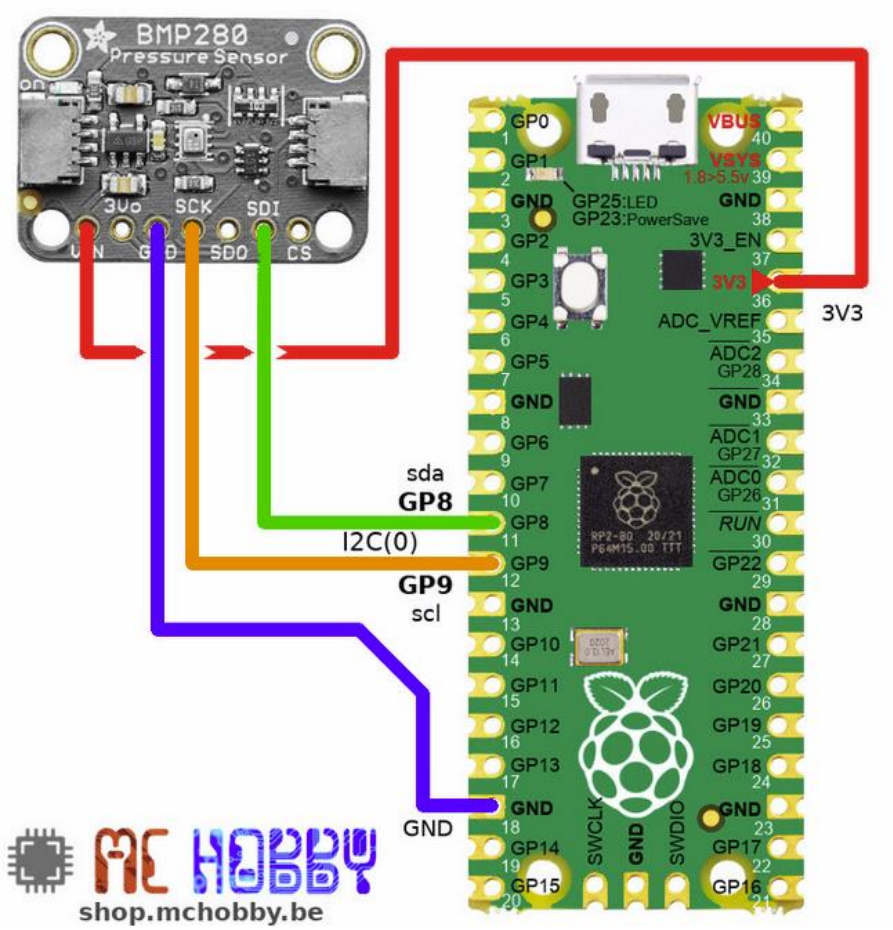
3. Connectez la broche d'entrée 2-6V du BMP280 à la broche 3V3 du Pi (3,3 volts 300 mA en sortie).

4. Connectez la broche GND du BMP280 à une broche GND du Pi.

5. Connectez l'I2C 0 SDA (broche 11) à la broche SDI du BMP280.

6. Connectez le I2C 0 SCL (broche 11) à la broche SCK du BMP280.

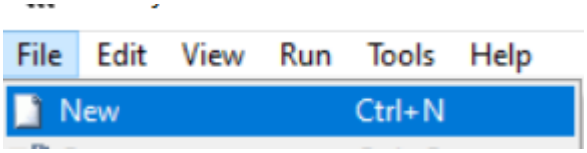
Voir le schéma de connexion ci-dessous :



Lecture de la température et de la pression

Maintenant que le BMP280 est connecté et configuré, nous pouvons lire les données qu'il contient.

1. Créer un nouveau fichier



2. Enregistrez ce fichier sous le nom de votre choix, **terminé par . py**

3. Remplissez le fichier avec le code suivant

```
from machine import I2C
# La bibliothèque BME280 fonctionne également pour le capteur BMP280.
from bme280 import BME280, BMP280_I2CADDR
from time import sleep
# Initie un nouveau connecteur I2C sur le bus 0, sda=GP8, scl=GP9 @ 400 KHz
(par défaut)
i2c = I2C(0)
# Créer une nouvelle variable BME280 connectée au bus i2c 0 communiquant
avec l'adresse BMP280_I2CADDR
bmp = BME280( i2c=i2c, address=BMP280_I2CADDR)
while True:
    # imprimer un tuple avec (température, pression et humidité)
    print( bmp. raw_values)
    sleep(1)
```

4. En appuyant sur le bouton "Run", la température, la pression et l'humidité seront imprimées toutes les secondes.

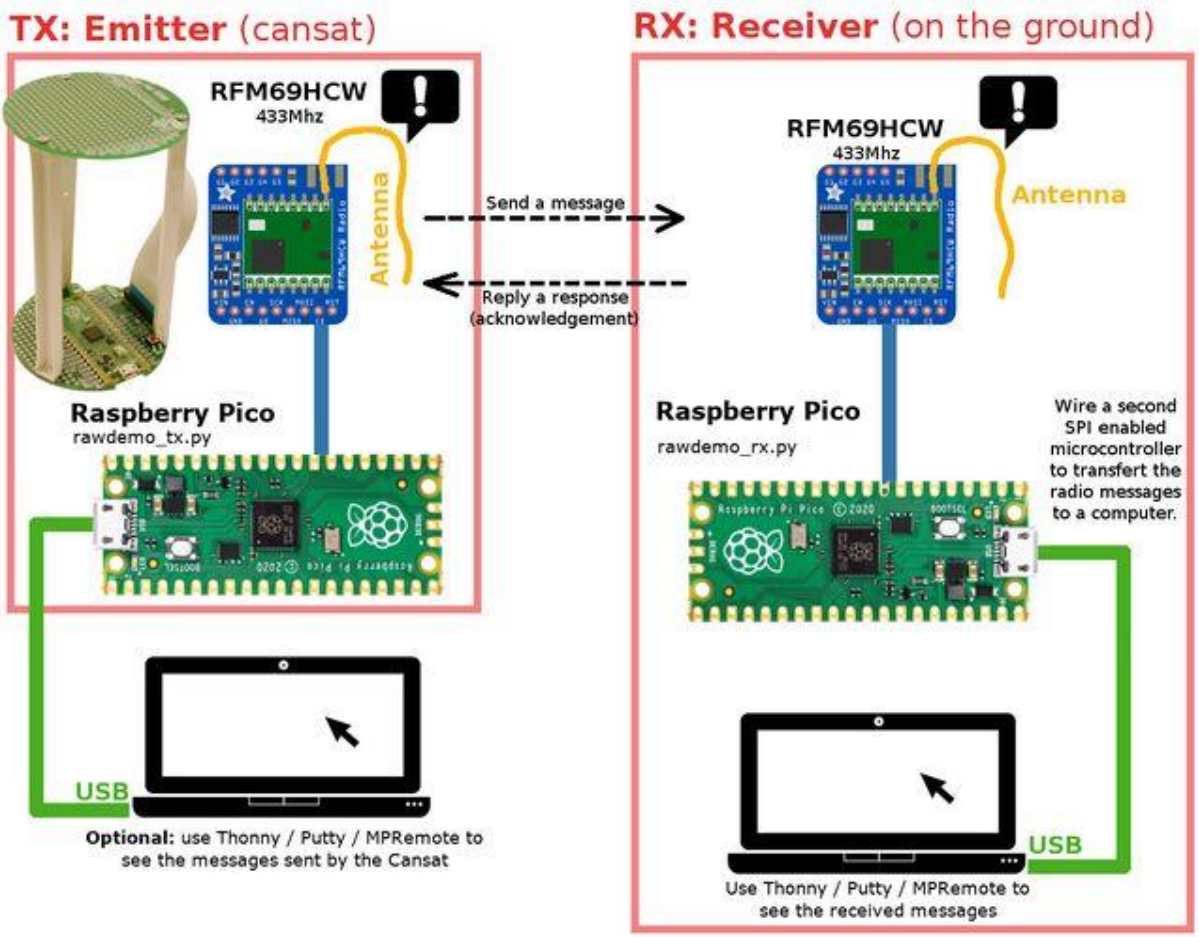
```
(22.28, 1017.68, 0.0)
(22.27, 1017.66, 0.0)
(21.87, 1017.67, 0.0)
(21.83, 1017.73, 0.0)
(21.83, 1017.68, 0.0)
(21.81, 1017.68, 0.0)
(21.81, 1017.68, 0.0)
```

L'altitude peut être calculée en utilisant la pression atmosphérique comme expliqué dans ce document.

Réception de données radio

Le kit contient 2 microcontrôleurs Pico et 2 modules RFM69HCW pour créer un "émetteur de données" (le CanSat) ainsi qu'un "récepteur de données" (la station terrestre).

Dans cette section, nous allons configurer la station au sol (à droite ci-dessous).



Une communication réussie exige des deux côtés :

- Fréquences identiques (par exemple : 433,1 MHz).
- Clés de cryptage identiques.
- Antennes bien conçues, sauf lors des tests sur les planches à pain où les 2 RFM69HCW sont très proches l'une de l'autre.

Installer la bibliothèque python RFM69HCW

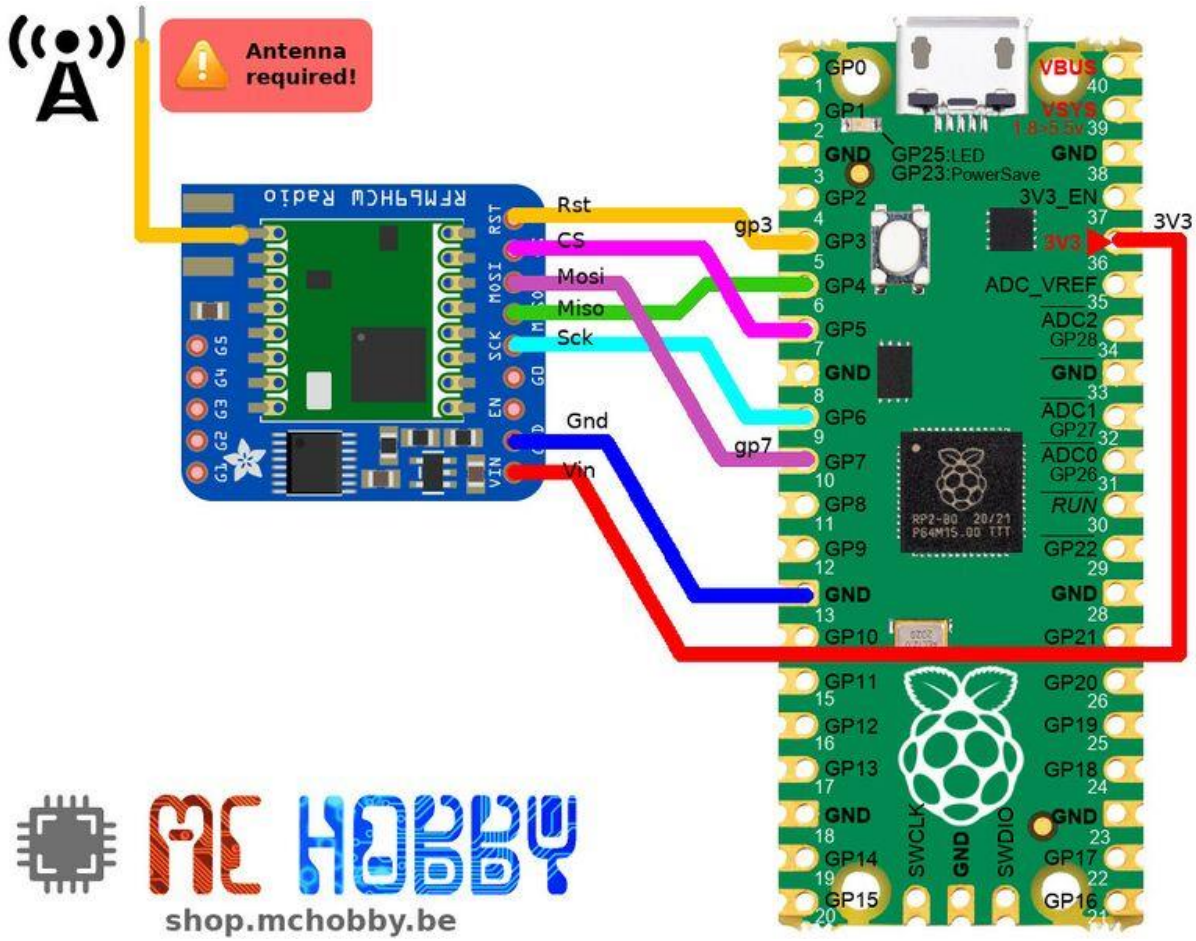
En utilisant Thonny, suivez le même processus que pour l'installation de la bibliothèque BMP280 pour télécharger le fichier rfm69.py suivant dans le dossier /lib de votre 2 Pico.

https://raw.githubusercontent.com/mchobby/esp8266-upy/master/rfm69/lib/rfm69.py

Connexion du RFM69HCW à l'aide de SPI

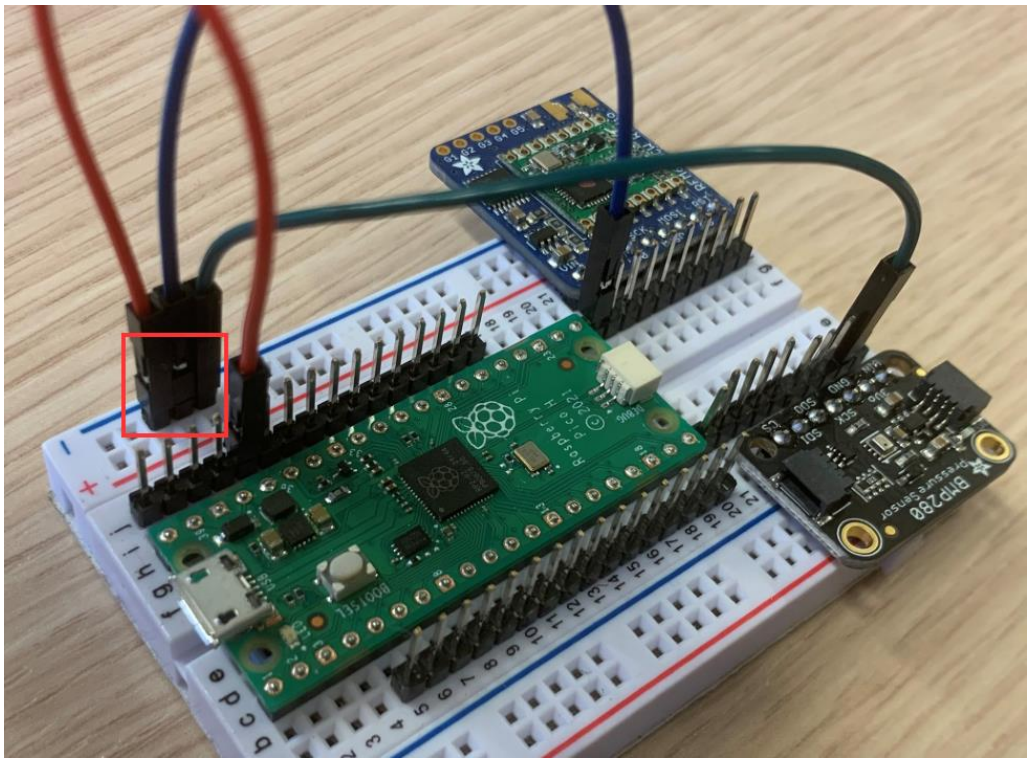
À l'aide d'embouts et de câbles de liaison, nous devons connecter la station de terre Pico et le RFM69HCW ensemble sur une planche à pain.

La documentation du RFM69HCW explique que nous pouvons nous y connecter en utilisant le bus SPI.
SPI nous oblige à connecter 4 câbles de données, une connexion de réinitialisation plus 2 câbles d'alimentation et, donc, nous devons connecter sept câbles au total.

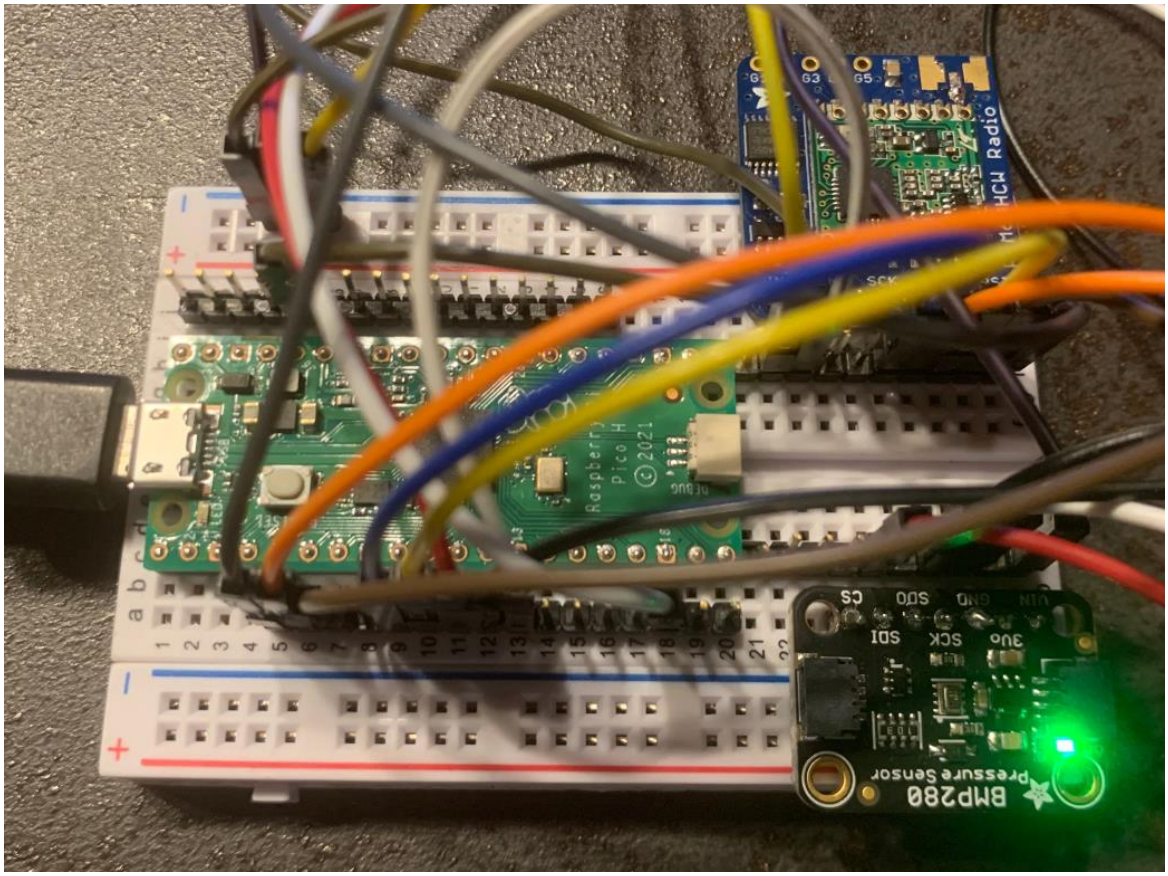


RFM69HCW	PICO
RST	GP3
CS	GP5 (Slave Select)
MOSI	GP7 (Miso)
MISO	GP4 (Mosi)
SCK	GP6 (Clock)
GND	GND
VIN	3V3

Notez que puisque la broche d'alimentation 3V3 est déjà utilisée pour alimenter le BMP280, vous pouvez la partager pour alimenter le module radio comme indiqué en rouge ci-dessous.



Après le câblage de la radio, la planche d'essai devrait ressembler à ceci



En attente de données radio

1- Téléchargez et ouvrez le script du récepteur dans Thonny.

2- Mettre à jour la fréquence et la clé de cryptage sur les lignes 17 et 30

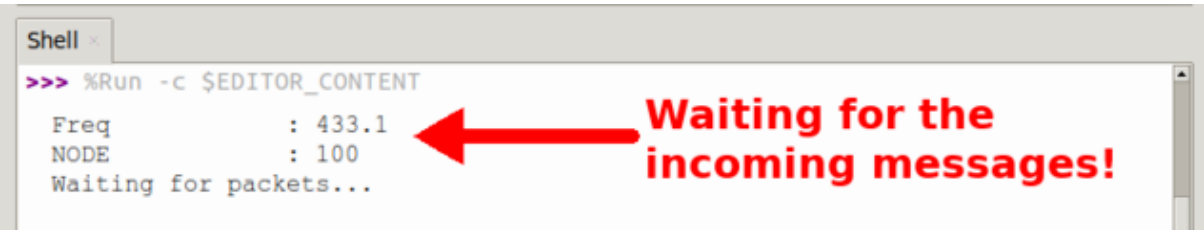
   a. La fréquence est définie sur la ligne 17, comprise entre 430 et 440Mhz,

   b. La clé de cryptage peut être définie à la ligne 30 en utilisant le code suivant

```
rfm.encryption_key = bytes( [1,2,3,4,5,6,7,8,1,2,3,4,5,6,7,8] )
```

   Chaque équipe recevra sa propre fréquence et sa clé de cryptage avant le lancement de la fusée.

3- Exécuter le script



Envoi de données radio

Dans cette section, nous allons construire l'émetteur de données qui sera à bord de votre CanSat.

Laissez Thonny ouvert avec le script du récepteur en cours d'exécution et ouvrez une seconde fois Thonny.

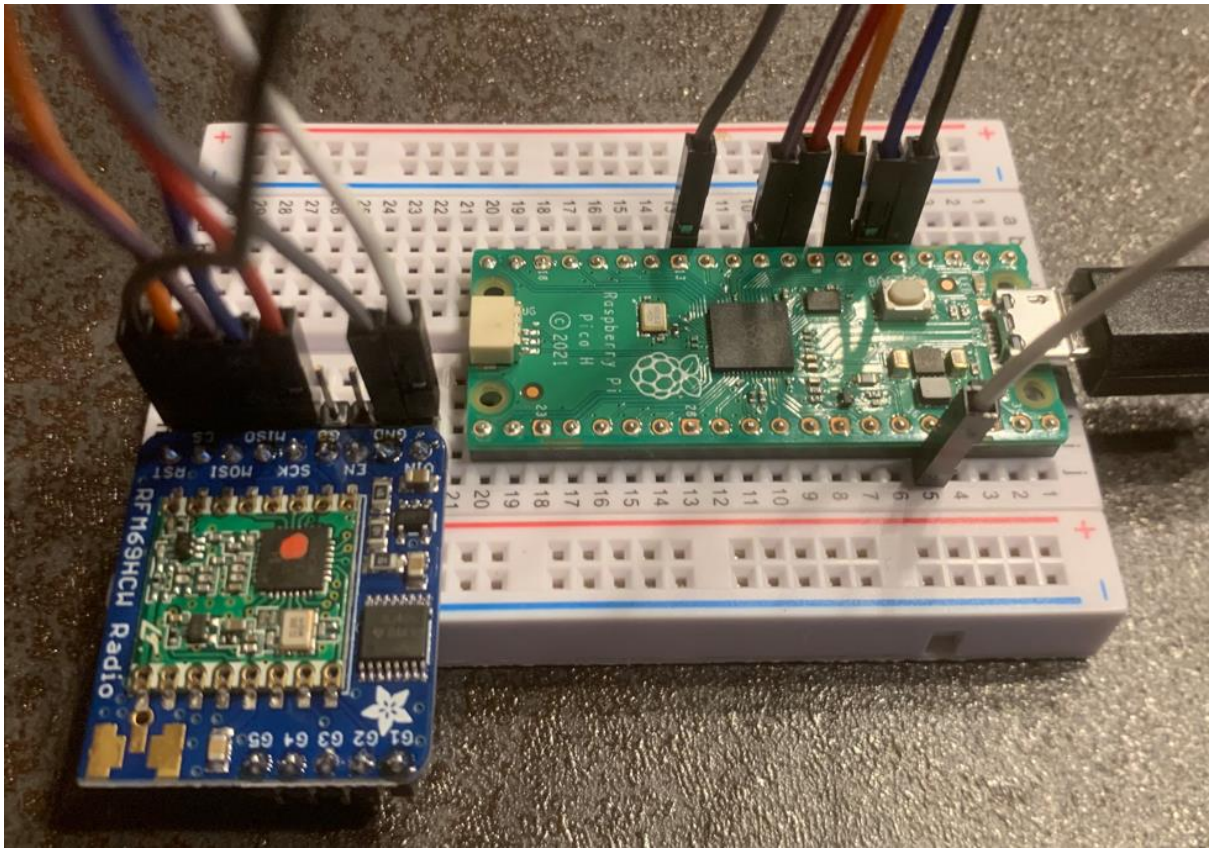
À l'aide de connecteurs et de câbles de liaison, nous devons connecter le CanSat Pico et le RFM69HCW sur une autre planche d'essai.

Sur le deuxième Pico :

1- Installer MicroPython

2- Installer la bibliothèque python RFM69HCW si ce n'est pas déjà fait (voir section précédente)

3- Connecter le RFM69HCW en utilisant SPI (voir section précédente)

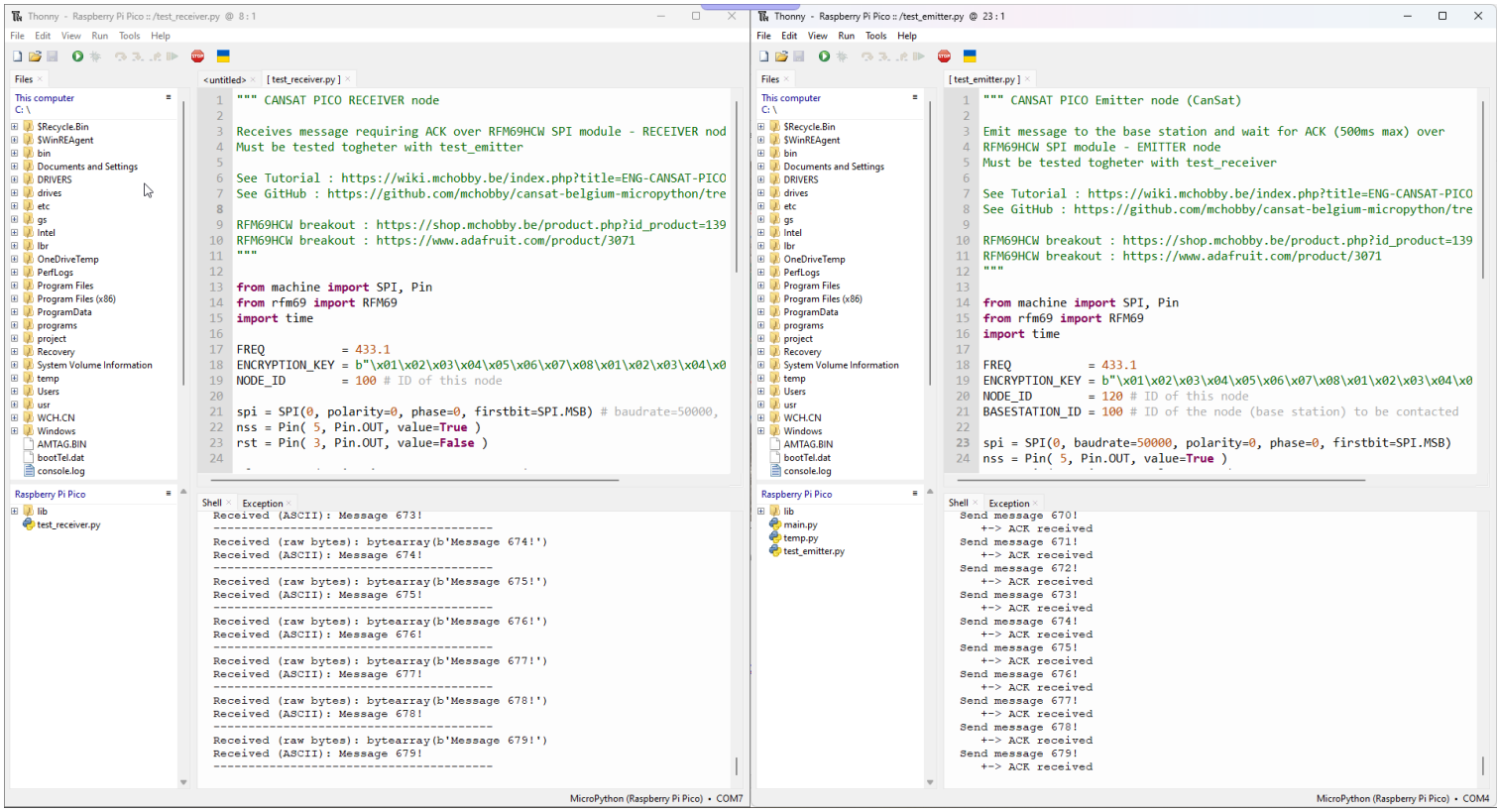


4- Téléchargez et ouvrez le script de l'expéditeur dans Thonny.

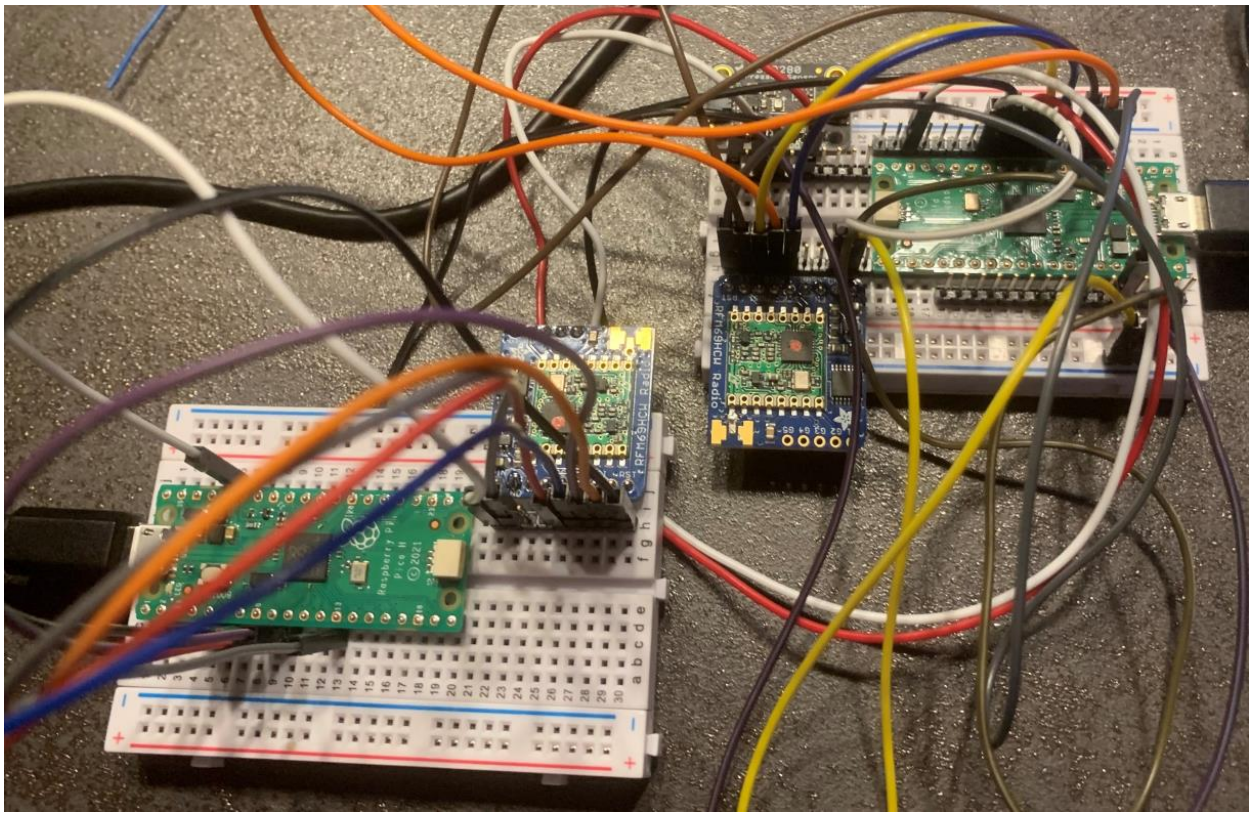
5- Mettez à jour la fréquence et la clé de cryptage sur les lignes 18 et 32 aux mêmes valeurs que la station terrestre.

6- Téléchargez le script sur le Pico

7- Exécuter le script



Les 2 radios se transmettent des données.
Elles ne doivent pas être éloignées de plus de 5 cm l'une de l'autre.



Des informations plus détaillées sont disponibles sur le [site web de McHobby.]

Assemblage de votre CanSat

Lorsque vous avez testé tous vos composants, sur les planches à pain.

Le wiki McHobby CanSat fournit toutes les informations nécessaires pour assembler les composants sur la base Cansat et les cartes d'extension fournies avec le kit.

Les étapes de montage de la mission primaire sont les suivantes :

1- Souder un Pico à la base de CanSat

   IMPORTANT : dans le kit, nous avons fourni des cartes Pico avec des connecteurs pré-soudés que vous pouvez également utiliser pour souder la base CanSat.

2- Soudez le composant PowerBoost 500 à la base du CanSat.

3- Connectez le BMP280 au câble Qwiic/StemmaQt

4- Connecter le TMP36 (optionnel)

5- Connecter le module radio

L'extension CanSat est utile si vous avez besoin d'espace pour ajouter des composants supplémentaires pour la mission secondaire, comme un GPS, un accéléromètre, une caméra, etc.

Que faire ensuite ?

- Lisez la ressource pour concevoir votre parachute
- Lisez la ressource pour concevoir vos antennes radio

Aller plus loin

- Pensez à ajouter un GPS pour augmenter vos chances de retrouver votre CanSat après le lancement.
  - Tutoriel
  - Module GPS
  - Bibliothèque GPS MicroPython
- Certaines équipes conçoivent leur propre tableau de bord en temps réel de la station terrestre.

Annexes

Apprendre Python

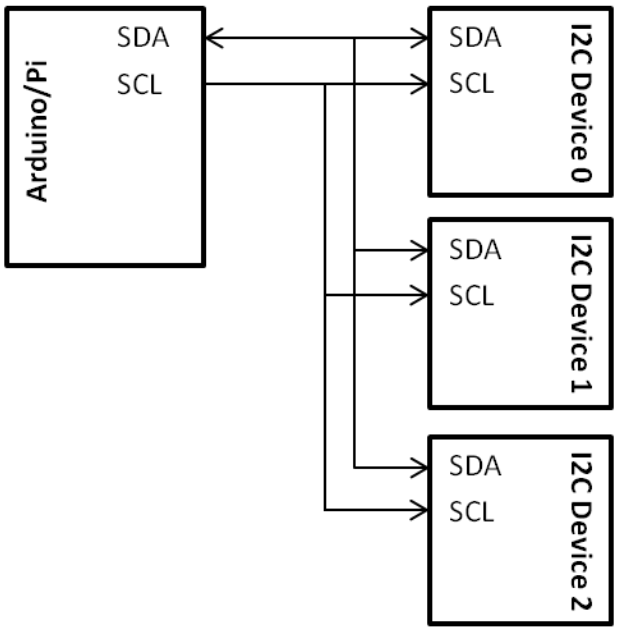
MicroPython est très similaire à Python. Si vous êtes un débutant absolu en Python, voici une liste de ressources utiles pour vous aider à démarrer :

https://wiki.python.org/moin/BeginnersGuide/NonProgrammers

Par exemple, LearnPython est un excellent tutoriel interactif qui convient aux débutants absolus.

I² C Protocole

I2C permet de connecter plusieurs dispositifs (jusqu'à 1008) à la même interface I2C avec seulement 2 fils. Elle permet également une communication bidirectionnelle sur ces deux fils et est donc idéale pour communiquer avec de nombreux capteurs. Un exemple de câblage avec trois dispositifs serait le suivant :



Le logiciel nécessaire pour communiquer avec les périphériques I2C peut être complexe, mais la plupart des périphériques disposent d'une bibliothèque logicielle qui vous offre des fonctions qui facilitent l'utilisation du périphérique. Par exemple, nous utilisons la bibliothèque BMP280 fournie pour cacher le code I2C de bas niveau.

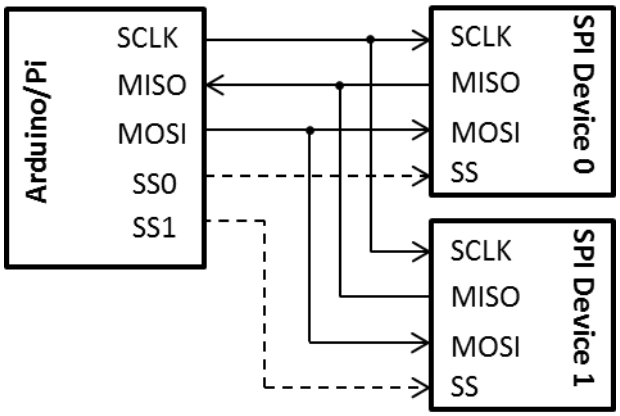
Utilisations typiques de CanSat I2C :
Capteurs "plus intelligents" (par exemple, le BMP280), accéléromètres, convertisseurs analogique-numérique, convertisseurs numérique-analogique, écrans LCD, contrôleurs de batterie.

Interface périphérique série (SPI)

SPI offre une interface avec des capacités plus puissantes que I2C au prix d'un câblage plus important. Comme l'I2C, elle prend également en charge la communication bidirectionnelle avec plusieurs dispositifs, mais offre un débit de données beaucoup plus élevé. Elle est donc adaptée à la communication avec les dispositifs les plus complexes que vous pourriez connecter au CanSat. L'interface se compose d'au moins quatre broches :

SCLK : *Serial Clock (horloge série)*. Un flux de 0-1s sur lequel les données sont alignées. Le taux d'horloge SPI est lié à la vitesse de ce flux, vous pouvez le ralentir si vous avez des problèmes d'intégrité de données.
MISO : *Master Input / Slave Output.* Les données du périphérique vers le Pi.
MOSI : *Sortie maître / Entrée esclave.* Les données du Pi vers le périphérique.
SS0/CE0 : *Sélection d'esclave / Activation de puce.* Active un périphérique et signifie que le périphérique peut sortir sur la broche MISO. Une broche SS/CE est nécessaire pour chaque périphérique.

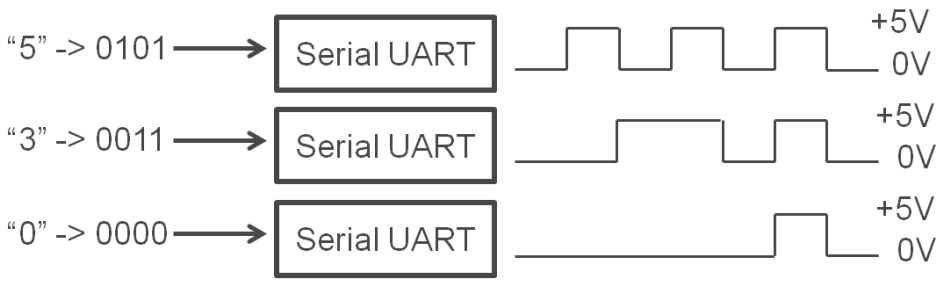
Pour utiliser SPI, vous n'avez pas besoin de vous préoccuper de la fonction de ces broches car la bibliothèque logicielle du périphérique se chargera de la plupart du code SPI de bas niveau pour vous. Cependant, il est bon d'être conscient de leur fonction lorsque vous mettez en cascade plusieurs périphériques SPI ensemble, par exemple pour connecter deux périphériques vous aurez besoin de deux broches SS/CE :



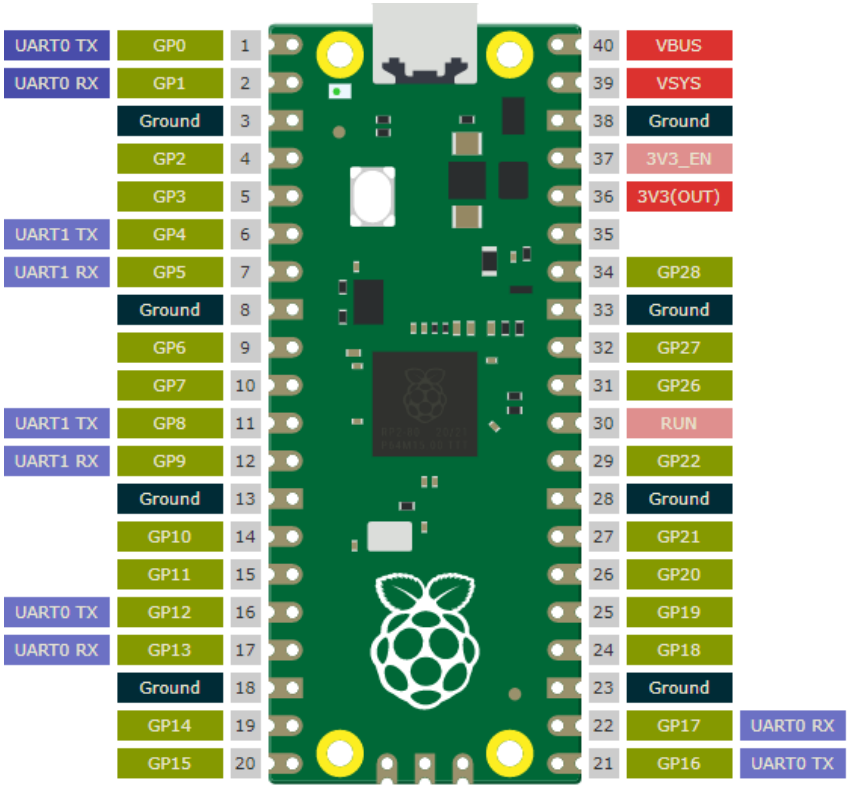
Utilisations typiques de SPI CanSat :
Caméras, cartes de stockage (par ex. cartes SD), modules GPS, modems WiFi.

Récepteur-transmetteur asynchrone universel (UART)

Malgré son nom, l'UART est une interface de communication relativement simple. Elle fonctionne de la même manière que le GPIO avec des valeurs vrai/faux représentées par 0V et 5V mais des impulsions sont envoyées sur le fil au lieu d'une tension constante. Cela permet de convertir une valeur numérique en une série d'impulsions et de les envoyer sur un seul fil :



Le Raspberry Pi Pico possède deux UART. Celles-ci peuvent être connectées à plusieurs paires de broches GP, comme indiqué en violet dans le schéma de brochage ci-dessous : TX est la transmission (c'est-à-dire les données envoyées par le Pi) et RX la réception (c'est-à-dire les données envoyées au Pi).



Utilisations typiques de l'UART CanSat :
Envoi de messages de débogage et de développement à un PC, communication avec des capteurs GPS, communication avec des modems externes WiFi et GPRS (3G).