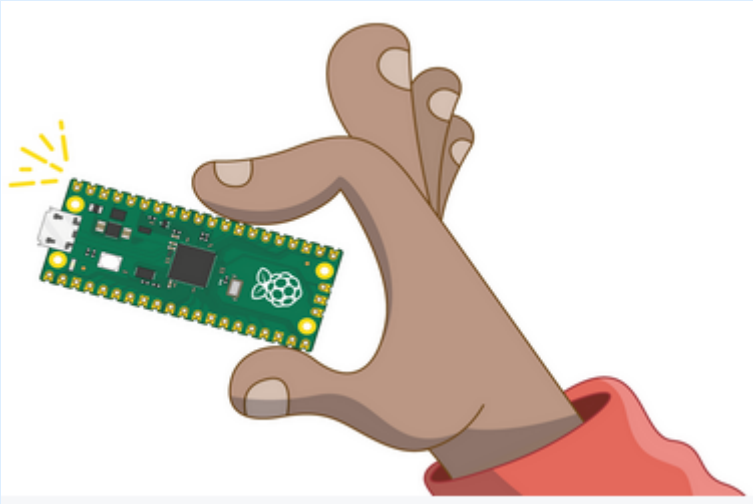
Apprendre à programmer en MicroPython avec la carte Raspberry Pi Pico

Par Fondation Raspberry Pi - f-leb (traducteur)

Date de publication : 19 juillet 2021

DÉBUTANT

NDLR : **la fondation Raspberry Pi fait son entrée dans l'univers des microcontrôleurs avec sa nouvelle carte à 4 $ dénommée Pi Pico** .



Dans ce tutoriel, vous connecterez une carte Rapsberry Pi Pico à un ordinateur et vous apprendrez à programmer cette carte avec le langage Python adapté aux microcontrôleurs, à savoir MicroPython.

La Raspberry Pi Pico est un microcontrôleur à très bas coût. Ces petits systèmes informatiques ont tendance à disposer d'un faible volume de stockage et manquent de périphériques que vous pourriez connecter (comme un clavier ou un écran). Mais la Raspberry Pi Pico a des broches GPIO, comme le nano-ordinateur Raspberry Pi, et on peut donc s'en servir pour piloter divers composants électroniques, capteurs ou actionneurs.

Commentez

En complément sur Developpez.com

- La fondation Raspberry Pi fait son entrée dans l'univers des microcontrôleurs avec sa nouvelle carte à 4 $ dénommée Pi Pico

I - Qu'allez-vous faire ?

Vous allez connecter la Raspberry Pi Pico à votre ordinateur, installer l'EDI (Environnement de développement intégré) Python **Thonny** et écrire un premier programme en MicroPython pour faire clignoter la LED intégrée en surface de la carte. Si vous avez les composants nécessaires, vous pourrez poursuivre avec d'autres exemples.

Cliquer sur ce lien pour lancer l'animation

I-A - De quoi allez-vous avoir besoin ?

Au plan matériel, vous aurez besoin des composants suivants :

- une carte Raspberry Pi Pico avec les connecteurs à broches soudés ;
- un ordinateur avec l'EDI Thonny pour programmer la carte en MicroPython ;
- un câble micro-USB ;
- un jeu de composants électroniques : un bouton-poussoir, une LED avec une résistance électrique appropriée, éventuellement un potentiomètre ;
- une plaque de câblage rapide (ou *breadboard*) et quelques fils de connexion mâle-mâle pour relier des composants sans soudures ;
- une source d'alimentation 5 V extérieure avec un connecteur micro-USB (optionnel).

Au plan logiciel :

- le firmware MicroPython à installer dans la Raspberry Pi Pico ;
- l'EDI Python Thonny.

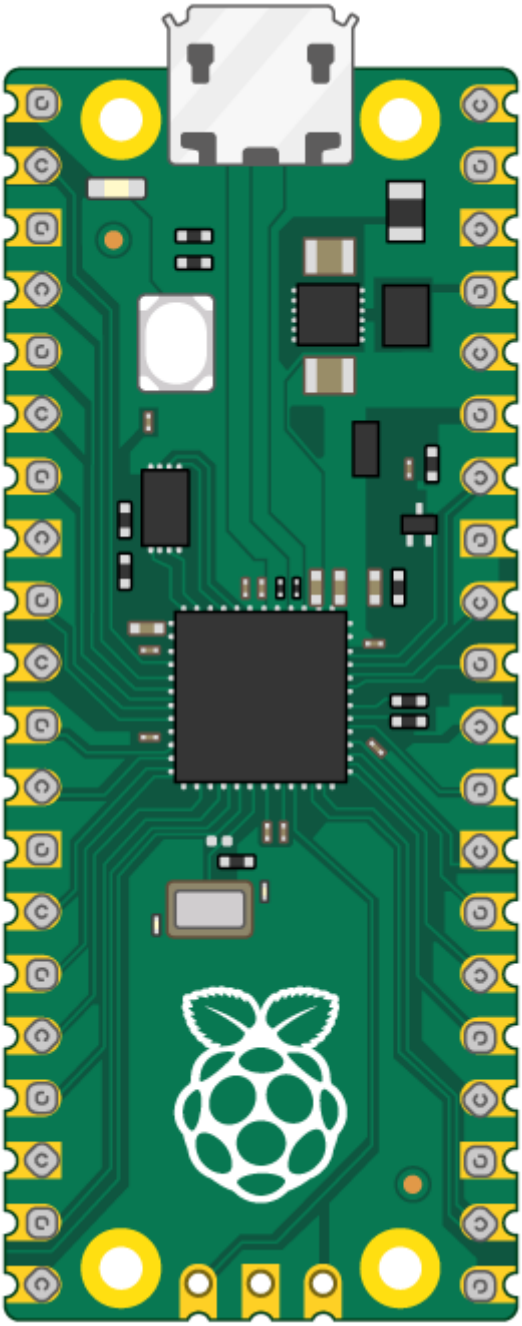
I-B - Qu'allez-vous apprendre ?

Vous allez apprendre :

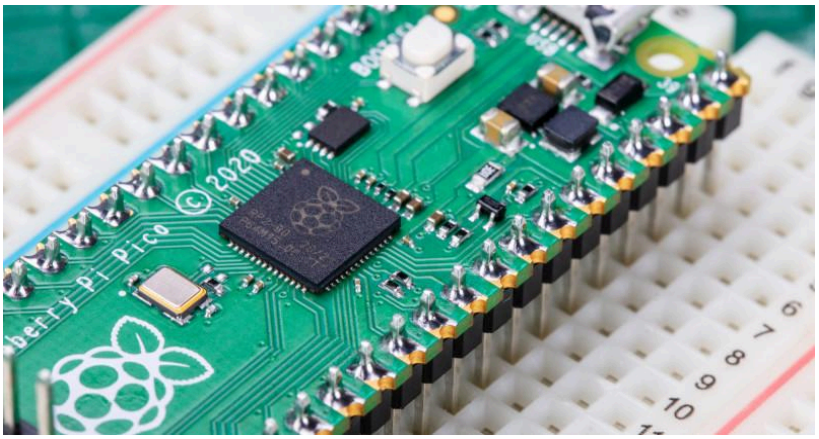
- comment téléverser le firmware MicroPython dans la Raspberry Pi Pico ;
- comment programmer la Raspberry Pi Pico en MicroPython ;
- comment connecter des composants supplémentaires à la Raspberry Pi Pico et écrire des programmes en MicroPython pour interagir avec eux.

II - À la découverte de la carte Raspberry Pi Pico

Voilà ci-dessous à quoi ressemble la Raspberry Pi Pico :

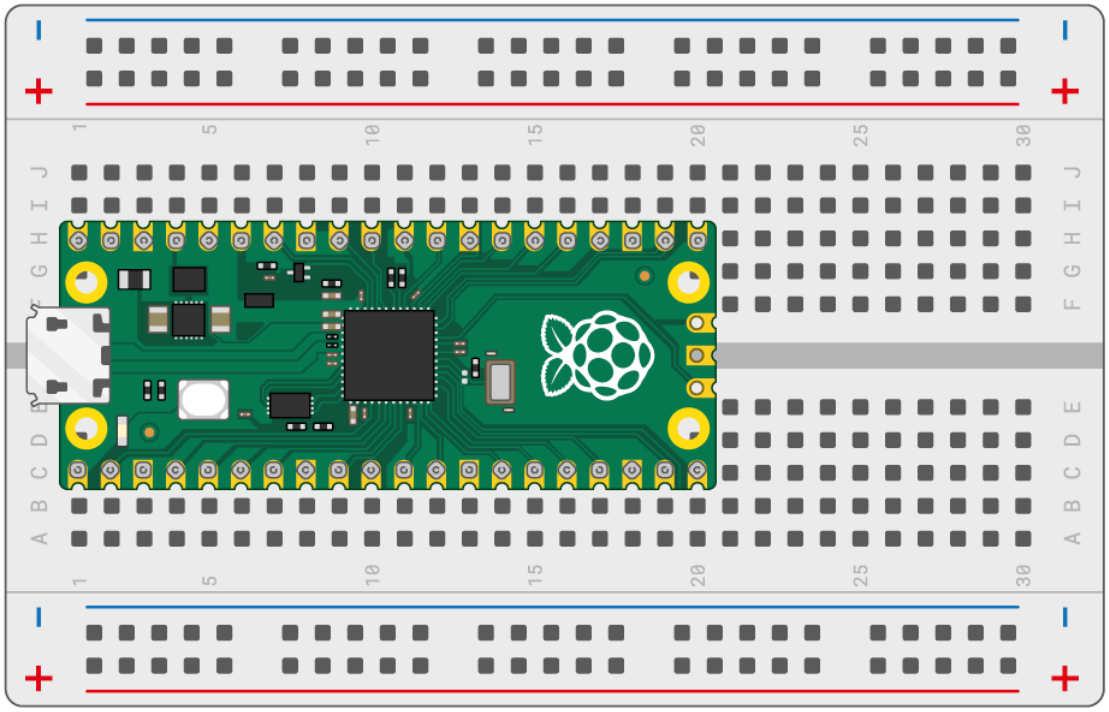


Si tout va bien, les connecteurs à broches sont déjà soudés sur la carte, mais si ce n'est pas le cas, suivez le guide : **How to solder GPIO pin headers to Raspberry Pi Pico**.

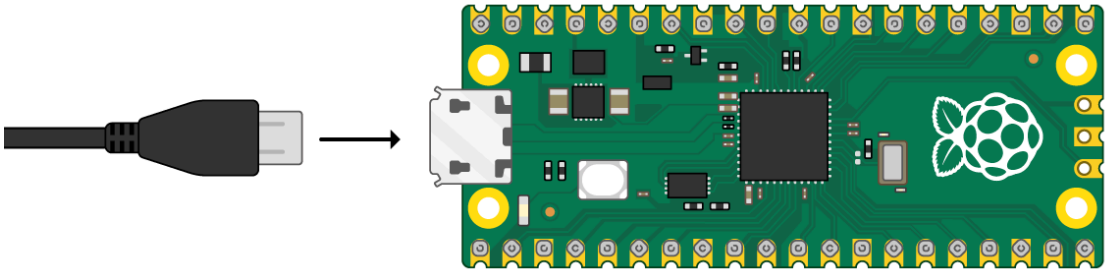


La carte Raspbery Pi Pico avec ses connecteurs à broches soudés, enfichée sur une plaque de câblage rapide.

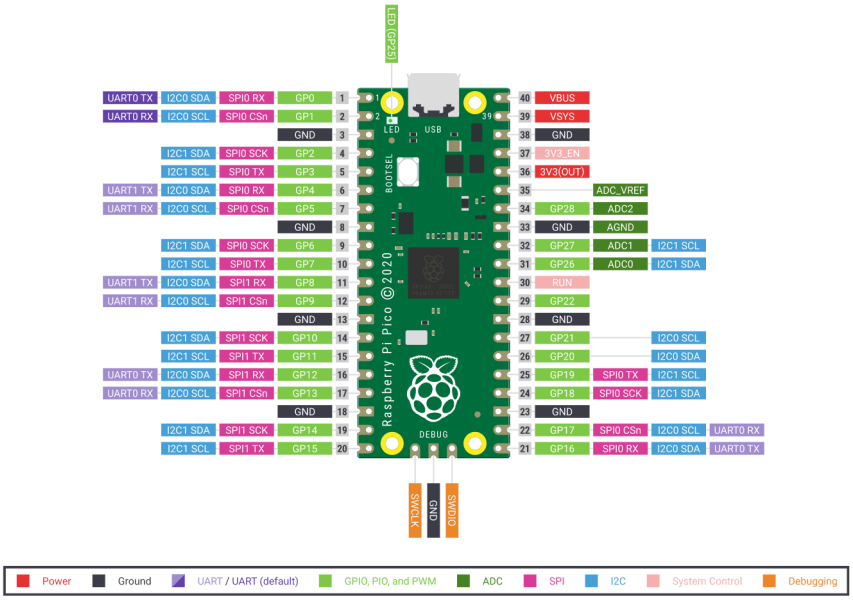
Si vous disposez d'une plaque de câblage rapide (*breadboard*), enfichez la Raspberry Pi Pico dessus. Les connecteurs à broches des deux côtés de la carte doivent être de part et d'autre de la séparation au milieu de la plaque de câblage rapide.



Branchez le câble micro-USB au port de la carte sur le côté.



Si vous avez besoin de connaître la numérotation des broches de la Raspberry Pi Pico, retrouvez-la sur le diagramme ci-dessous :



Plan de brochage

III - Installation de Thonny

Dans cette étape, vous allez installer la dernière version de Thonny. Une fois l'EDI installé, vous pourrez alors faire vos premiers pas en MicroPython.

Thonny sur Raspberry Pi

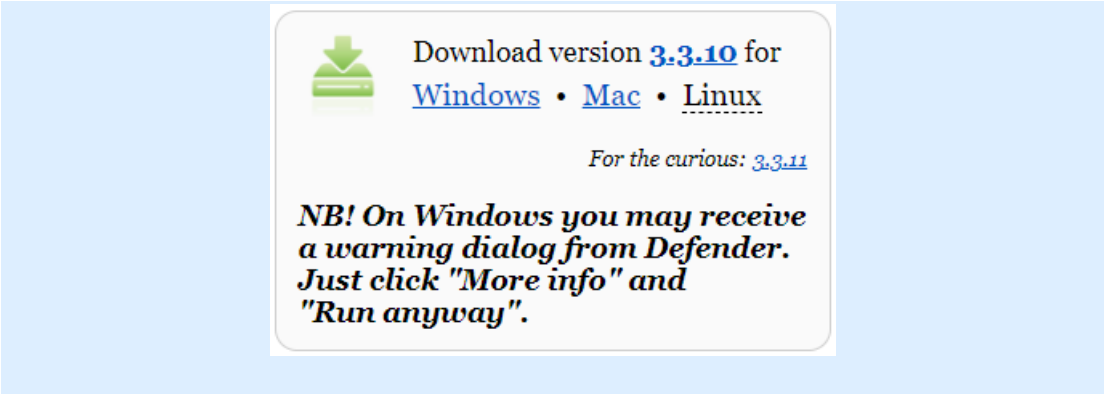
Thonny est installé par défaut sur Raspberry Pi OS, mais vous devrez faire une mise à jour. Ouvrez une fenêtre *Terminal*, soit en cliquant sur l'icône en haut à gauche de l'écran, soit par la combinaison de touches Ctrl+Alt+T. Dans le terminal, tapez la commande suivante pour mettre à jour tout le système :

```
sudo apt update && sudo apt upgrade -y
```

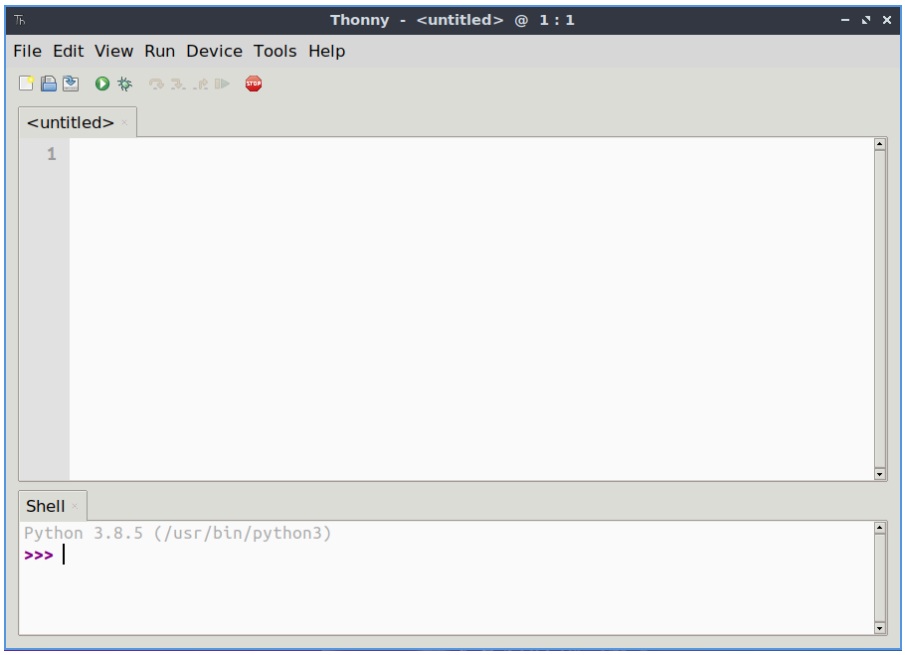
Installer Thonny sur les autres OS

Sur Windows, macOS et Linux, vous pouvez aussi installer l'EDI Thonny ou mettre à jour une version existante.

- Dans un navigateur, rendez-vous sur le site **thonny.org**.
- En haut et à droite de la page, vous devriez voir les liens de téléchargement pour Windows et macOS, ainsi que des instructions pour Linux.
- Téléchargez les fichiers correspondant à votre OS et exécutez-les pour installer Thonny.

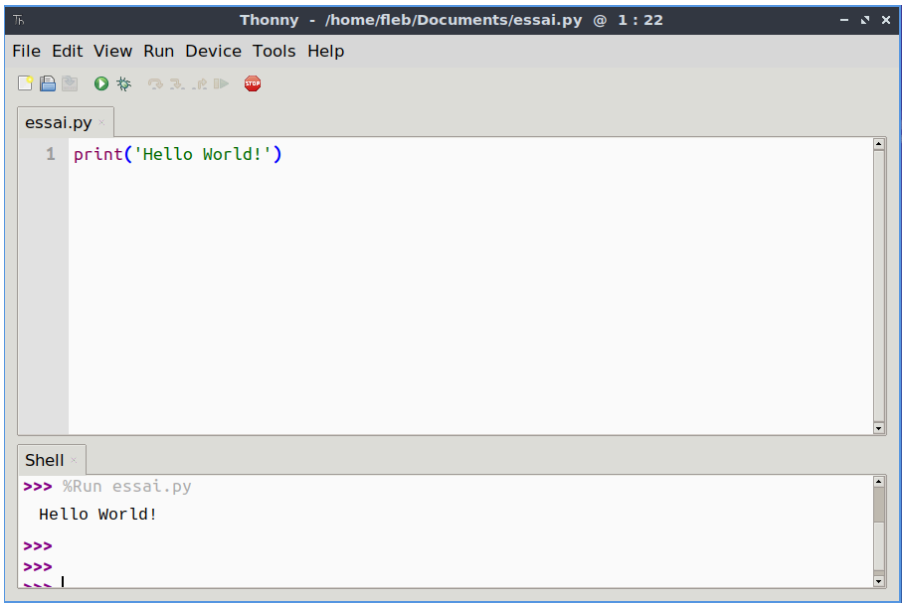


Lancez l'application Thonny, la fenêtre de l'EDI devrait ressembler à l'image ci-dessous :



Vous pouvez utiliser Thonny pour écrire du code Python standard. Tapez la ligne suivante dans la fenêtre principale de l'éditeur, puis cliquez sur le bouton *Run* (l'application vous demandera d'abord de renseigner l'emplacement et le nom du fichier à sauvegarder) :

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

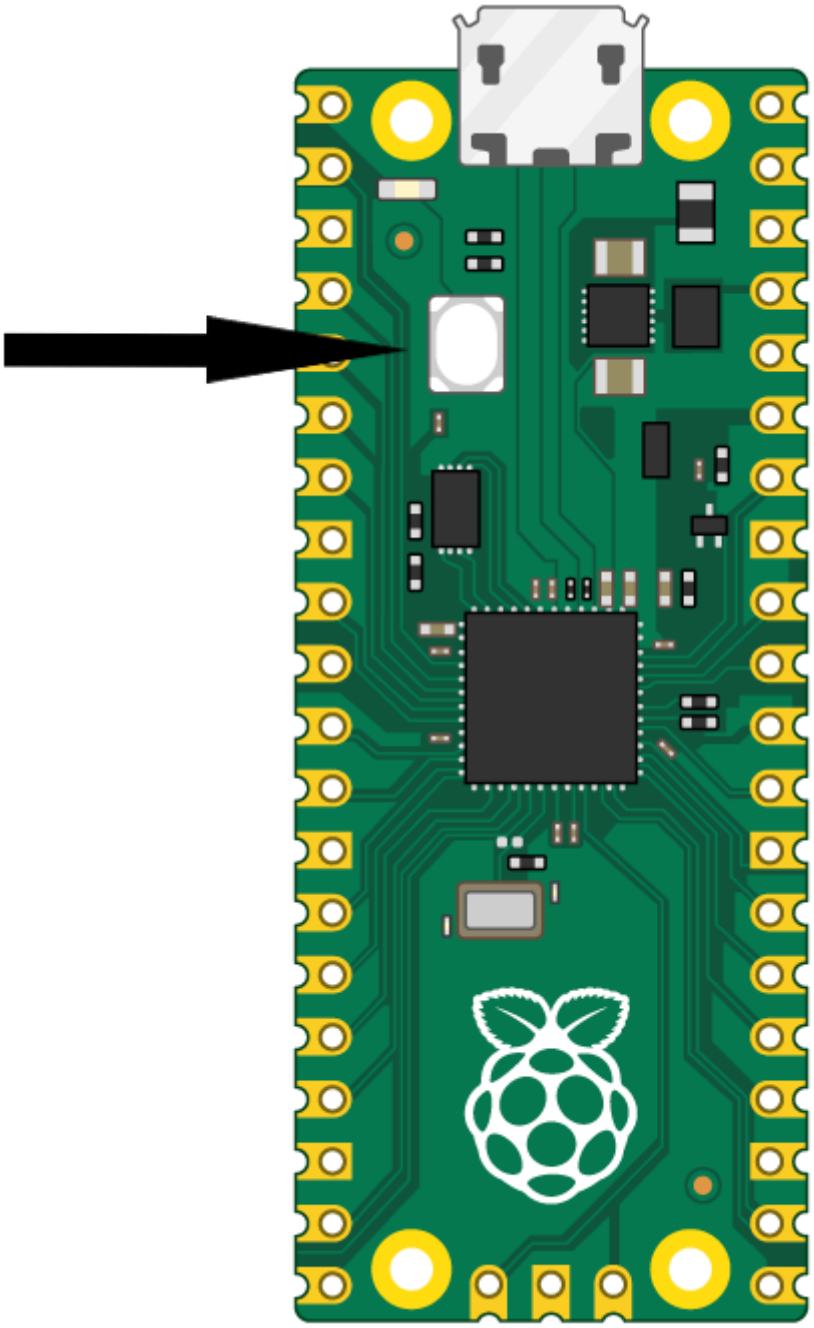


Vous êtes prêt pour l'étape suivante.

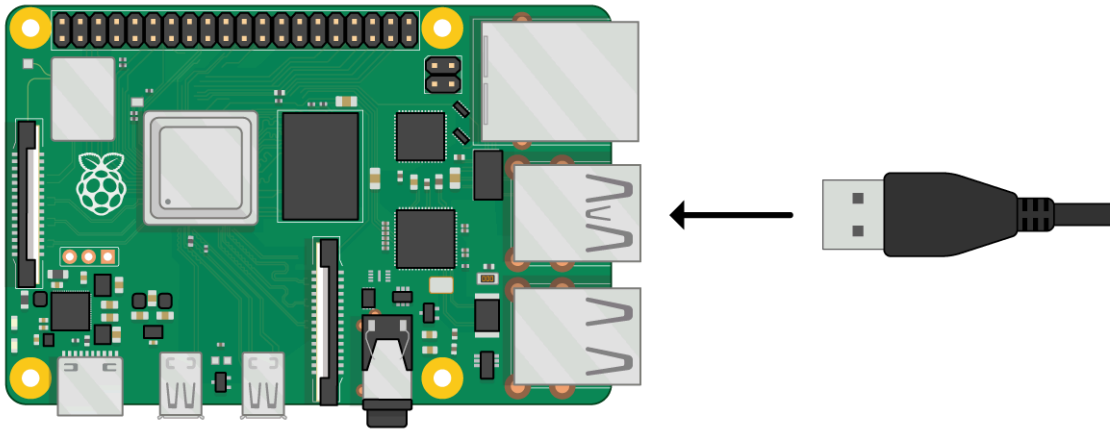
IV - Téléverser le firmware MicroPython

Avant d'utiliser pour la première fois votre Raspberry Pi Pico, il faut commencer par y téléverser le firmware MicroPython.

Repérez le bouton BOOTSEL sur la carte :



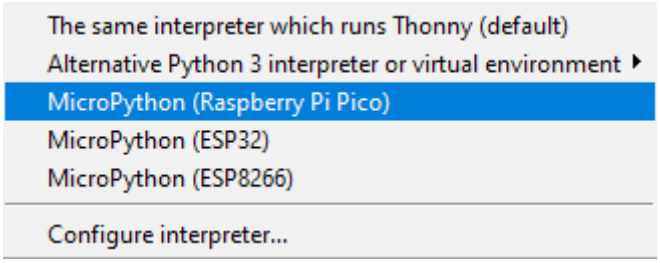
Pressez le bouton BOOTSEL et maintenez-le enfoncé pendant que vous connectez le câble USB à votre ordinateur. L'image ci-dessous montre le nano-ordinateur Raspberry Pi, mais la démarche reste la même pour tout type d'ordinateur.



La Raspberry Pi Pico est alors reconnue en tant que périphérique de stockage USB.

Dans le coin inférieur droit de la fenêtre Thonny, vous devriez voir la version de Python en cours :

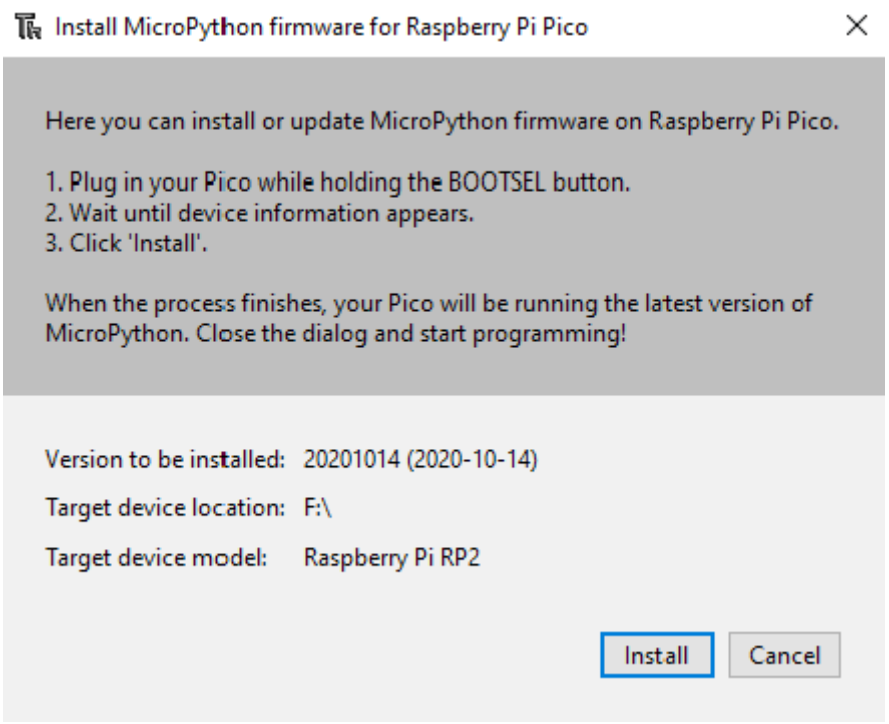
Cliquez sur le numéro de version, et choisissez l'option *MicroPython (Raspberry Pi Pico)* :



Si vous ne voyez pas cette option, vérifiez que la Raspberry Pi Pico est bien branchée.

Une boîte de dialogue vous invite à installer la dernière version du firmware MicroPython.

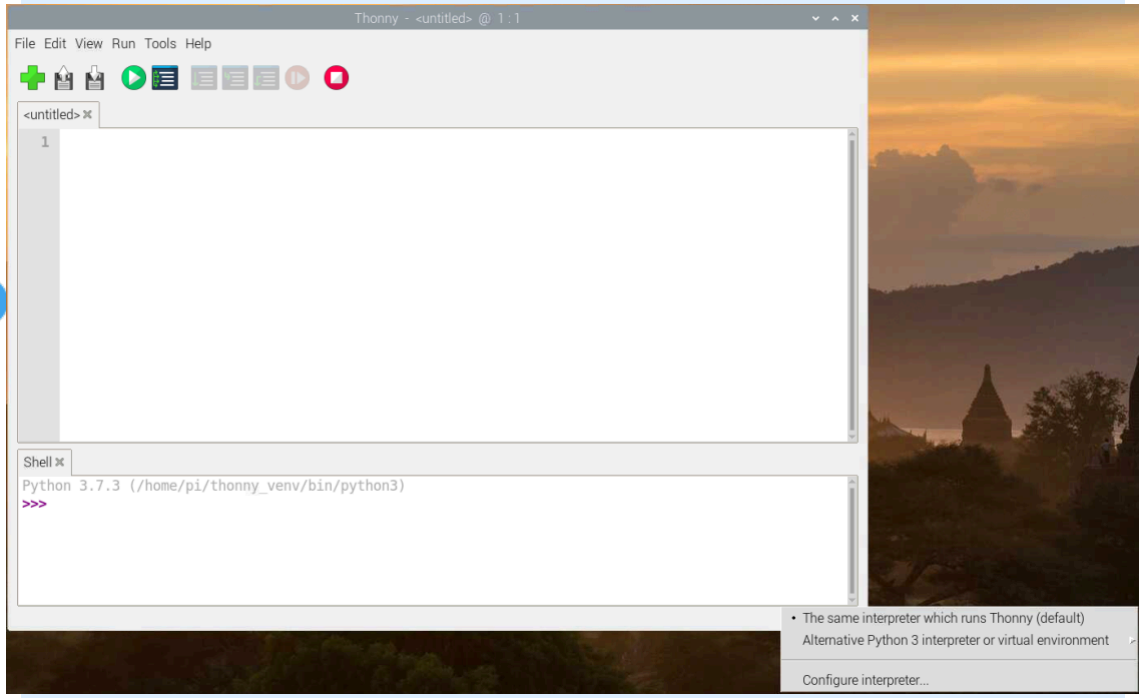
Cliquez sur le bouton d'installation pour copier le firmware dans la Rapsberry Pi Pico.



Une fois l'installation complétée, cliquez sur *Close*.

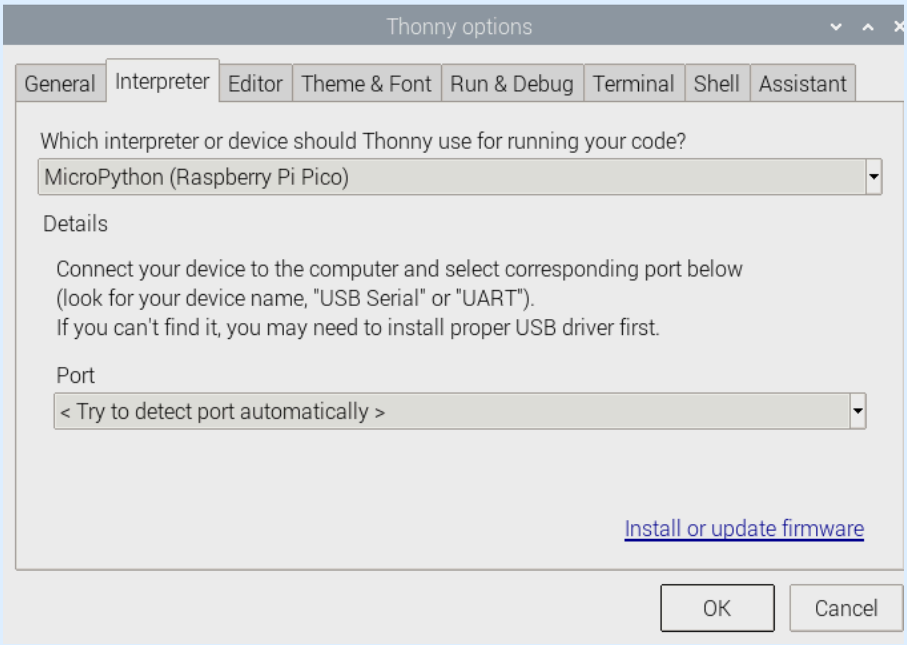
Menu d'installation du firmware

Vous pouvez aussi accéder au menu d'installation du firmware en cliquant sur *MicroPython (Raspberry Pi Pico)* dans la barre de statut, et en choisissant l'option *Configure interpreter…*



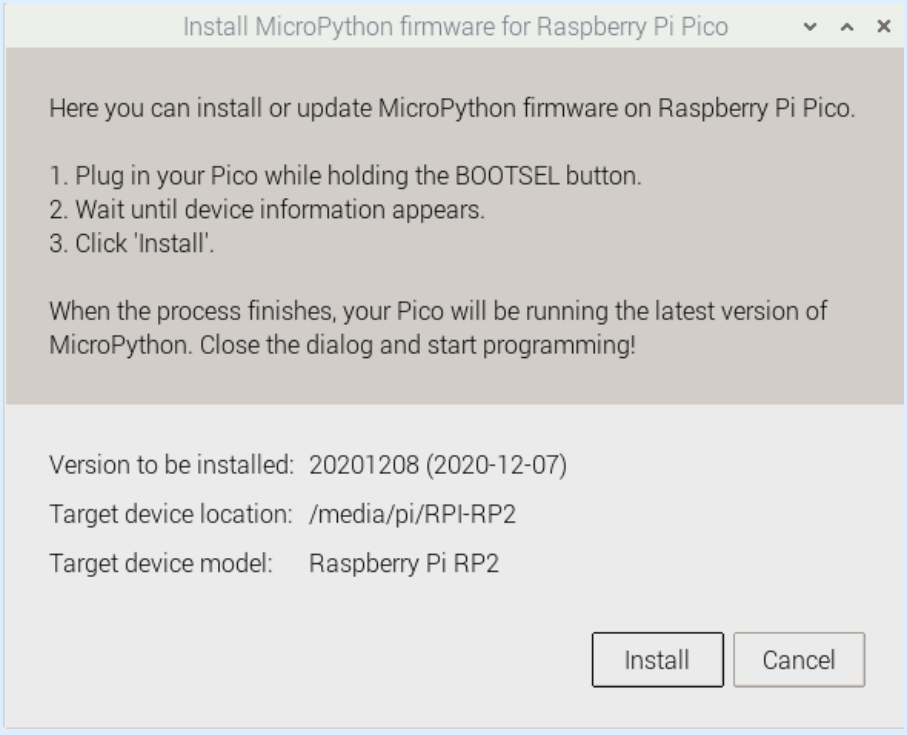
Sur Raspberry Pi, configuration de l'interpréteur dans Thonny

La fenêtre de configuration de l'interpréteur s'ouvre alors :

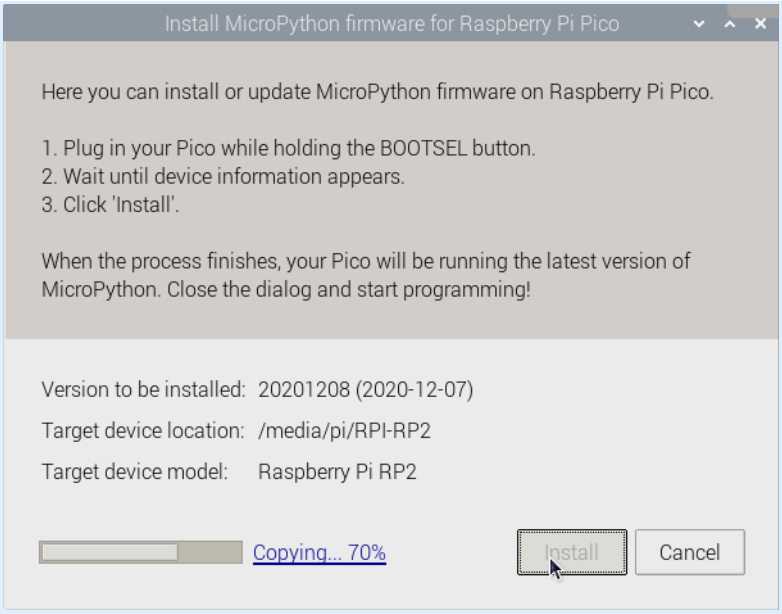


Cliquez sur *Install or update firmware*.

Vous serez invité à connecter votre Raspberry Pi Pico pendant que vous maintenez appuyé le bouton BOOTSEL.



Cliquez alors sur *Install*.



Attendez la fin de l'installation puis cliquez sur *Close*.

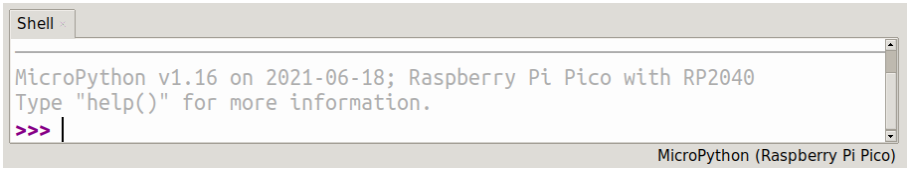
Vous n'avez pas besoin de mettre à jour le firmware à chaque fois que vous utilisez votre Raspberry Pi Pico. Les fois suivantes, vous pouvez donc vous contenter de connecter la carte à votre ordinateur sans presser le bouton BOOTSEL.

V - Utiliser le Shell

Vous pouvez maintenant utiliser le *shell* de Thonny pour exécuter quelques instructions Python en ligne de commande.

Assurez-vous d'avoir auparavant connecté votre Raspberry Pi Pico à votre ordinateur et de sélectionner l'interpréteur *MicroPython (Raspberry Pi Pico)*.

Observez le panneau du *shell* dans la partie inférieure de la fenêtre de l'éditeur Thonny. Vous devriez voir ceci :



Thonny est maintenant capable de communiquer avec votre Raspberry Pi Pico grâce au REPL (*read–eval–print loop*, un acronyme qui signifie *:* lire les entrées utilisateur, évaluer le code, afficher les résultats, le tout en boucle pour poursuivre la conversation). Il suffit de taper du code dans le *shell* et constater le résultat produit par la Raspberry Pi Pico en sortie.

Par exemple, tapez le code suivant dans le *shell* :

```
print("Hello")
```

Après avoir appuyé sur la touche [Entrée], vous devriez voir ceci :

```
Shell
MicroPython v1.16 on 2021-06-18; Raspberry Pi Pico with RP2040
Type "help()" for more information.
>>> print("Hello")
 Hello
>>>
MicroPython (Raspberry Pi Pico)
```

Dans le code suivant, on commence par importer le sous-module machine.Pin pour accéder aux broches d'entrées-sorties.

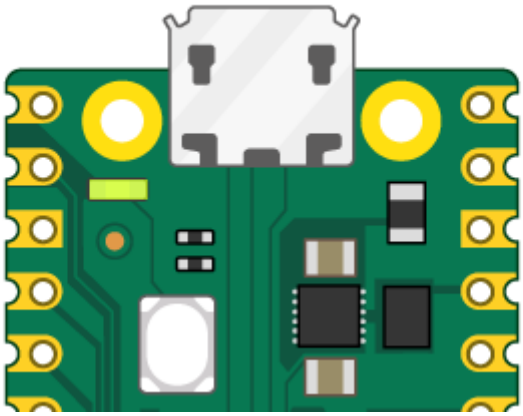
Ensuite, la variable led est associée à la broche GPIO 25 configurée en sortie. Si cette sortie est mise à 1, la LED en surface de la Raspberry Pi Pico connectée à la broche GPIO 25 s'allume.

Le code est à saisir dans le *shell* au niveau de l'invite de commande >>>, et ligne par ligne, en appuyant sur [Entrée] à chaque fin de ligne :

```
from machine import Pin

led = Pin(25, Pin.OUT)

led.value(1)
```

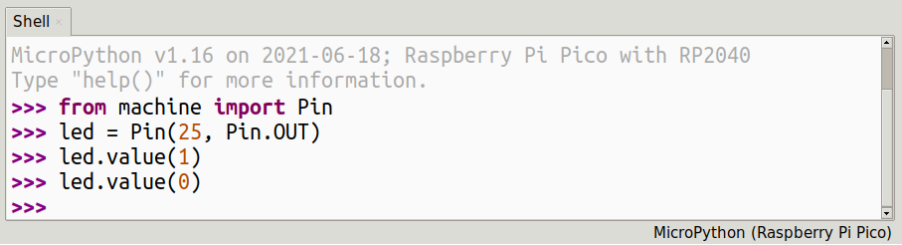
```
Shell
MicroPython v1.16 on 2021-06-18; Raspberry Pi Pico with RP2040
Type "help()" for more information.
>>> from machine import Pin
>>> led = Pin(25, Pin.OUT)
>>> led.value(1)
>>>
MicroPython (Raspberry Pi Pico)
```

Après avoir validé la dernière ligne, la LED devrait s'allumer :



Si à la suite on écrit led.value(0) puis [Entrée], la LED s'éteint.

Au lieu de taper entièrement cette dernière instruction, vous pouvez rappeler la ligne précédente avec la touche fléchée vers le haut, et la modifier.



Pour des programmes plus longs, sauvegarder l'ensemble des instructions du programme dans un même fichier est évidemment préférable, ce que nous ferons dans la prochaine étape.

VI - Programmer un clignotement de LED

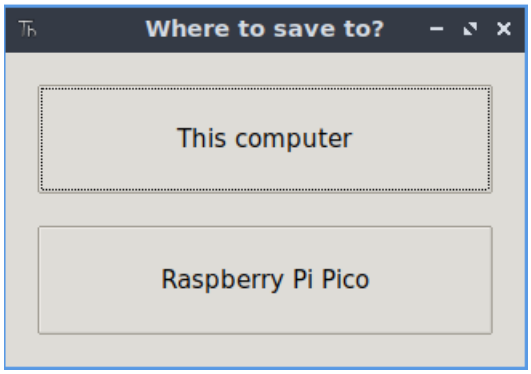
Le *Shell* est bien pratique pour s'assurer que tout fonctionne en testant quelques commandes en ligne. Pour autant, si le programme devient conséquent, il est plus pratique de le sauvegarder dans un fichier, et Thonny peut bien sûr sauvegarder et exécuter des programmes en MicroPython.

Ici, vous allez apprendre à créer un programme MicroPython pour faire clignoter la LED en surface de la carte.

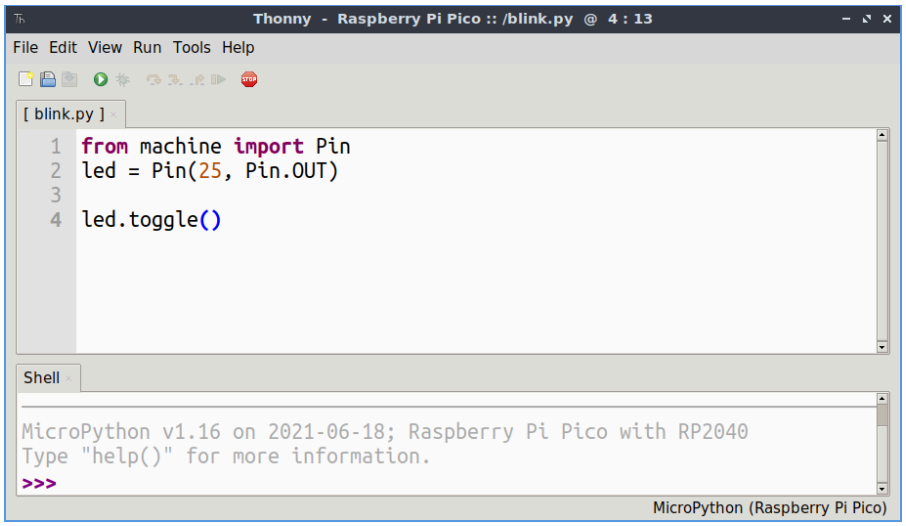
Cliquez dans la fenêtre principale d'édition de Thonny puis saisissez le code suivant qui fait basculer l'état de la LED.

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

led.toggle()
```

Cliquez sur le bouton *Run* pour exécuter le code. Thonny vous demandera d'abord si vous souhaitez sauvegarder le programme dans un fichier de votre ordinateur (*This computer*) ou dans votre carte (*Raspberry Pi Pico*). Choisissez l'option *Raspberry Pi Pico* :



Entrez *blink.py* comme nom de fichier. Vous devez préciser l'extension *.py* pour que Thonny reconnaisse le fichier comme un programme écrit en Python.



Thonny peut maintenant sauvegarder votre programme dans la Raspberry Pi Pico. Vous devriez voir la LED de la carte changer d'état allumé ou éteint à chaque fois que vous cliquez sur le bouton *Run*.

Grâce au module Timer, vous pouvez configurer un chronomètre (*timer*) pour déclencher un événement à intervalle régulier.

Pour cela, mettez votre code à jour comme suit :

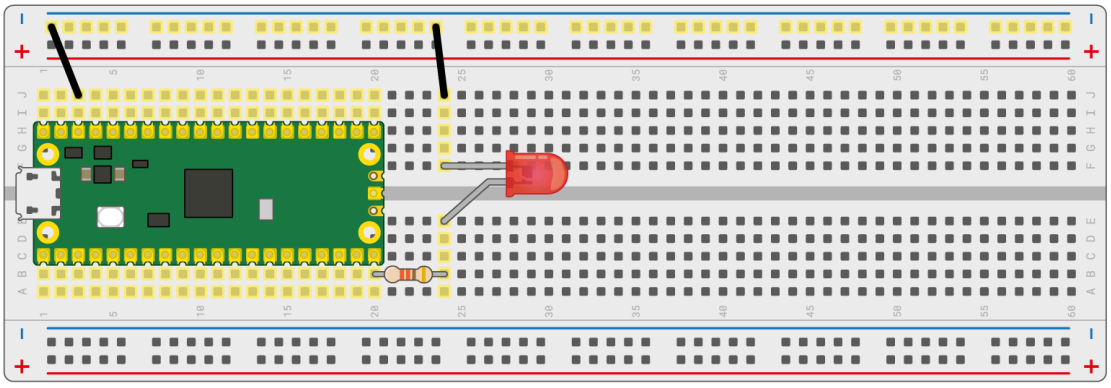
```
from machine import Pin, Timer
led = Pin(25, Pin.OUT)
timer = Timer()

def blink(timer):
    led.toggle()

timer.init(freq = 2.5, mode = Timer.PERIODIC, callback = blink)
```

Sauvegardez votre programme et cliquez sur *Run*. Vous devriez voir la LED clignoter jusqu'à ce que vous cliquiez sur le bouton *Stop*.

VII - Programmer les entrées-sorties numériques

Maintenant que vous connaissez les bases, vous pouvez poursuivre en contrôlant une LED externe avec votre Raspberry Pi Pico et apprendre à lire une entrée numérique reliée à un bouton-poussoir.

Vous aurez besoin d'une résistance électrique (entre 50 et 330 Ω), une LED et deux câbles mâle-mâle pour faire les connexions comme sur le schéma ci-dessous :



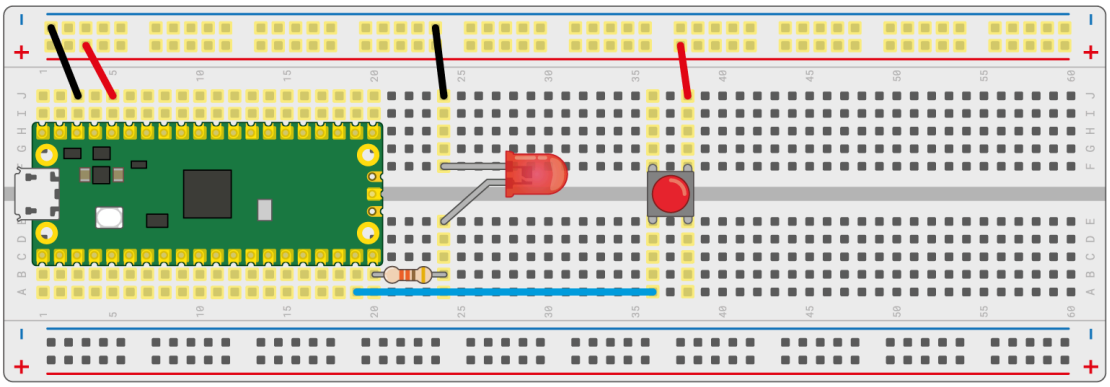
Dans cet exemple, la LED est connectée à la broche GPIO 15. Si vous voulez choisir une autre broche, pensez à consulter **le plan de brochage** (*pinout diagram*) de la Rapsberry Pi Pico.

Vous pouvez reprendre le code du **chapitre précédent** quand vous avez fait clignoter la LED en surface de la carte Raspberry Pi Pico. Il suffit de changer le numéro de la broche GPIO en le remplaçant par 15.

```
from machine import Pin, Timer
led = Pin(15, Pin.OUT)
timer = Timer()

def blink(timer):
    led.toggle()

timer.init(freq = 2.5, mode = Timer.PERIODIC, callback = blink)
```

Exécutez votre programme et la LED devrait clignoter. Si ce n'est pas le cas, il faut peut-être vérifier à nouveau votre câblage.

L'étape suivante consiste à contrôler l'état de la LED en appuyant sur un bouton-poussoir. Ajoutez le bouton sur la plaque de prototypage et connectez-le comme sur la figure ci-dessous :



Le bouton est connecté à la broche GPIO 14 d'une part et à la broche d'alimentation 3,3 V de la carte Raspberry Pi Pico d'autre part. Cela signifie que vous devez configurer la broche GPIO en entrée et activer sa résistance de rappel interne (*pull-down*).

Créez un nouveau fichier avec ce code :

```
from machine import Pin
import time

led = Pin(15, Pin.OUT)
button = Pin(14, Pin.IN, Pin.PULL_DOWN)
```

```
while True:
    if button.value():
        led.toggle()
        time.sleep(0.5)
```

À l'exécution du programme, la LED devrait se mettre à clignoter tant que le bouton est maintenu appuyé.

VIII - Contrôler la luminosité d'une LED avec un signal PWM

Les signaux PWM (*Pulse Width Modulation* ou en français MLI pour **Modulation en Largeur d'Impulsions**) permettent dans certains cas de reproduire le fonctionnement qu'on aurait avec une sortie analogique et donc, par exemple, de moduler la tension moyenne aux bornes d'une LED afin de contrôler sa luminosité.

Pour cette activité, vous pouvez reprendre le montage précédent.

Créez un nouveau fichier dans Thonny et saisissez le code suivant :

```
from machine import Pin, PWM
from time import sleep

pwm = PWM(Pin(15))

pwm.freq(1000)

while True:
    for duty in range(65535):
        pwm.duty_u16(duty)
        sleep(0.0001)
    for duty in range(65535, 0, -1):
        pwm.duty_u16(duty)
        sleep(0.0001)
```

Sauvegardez le ficher et exécutez le programme. La LED devrait continuellement s'allumer puis s'éteindre progressivement.

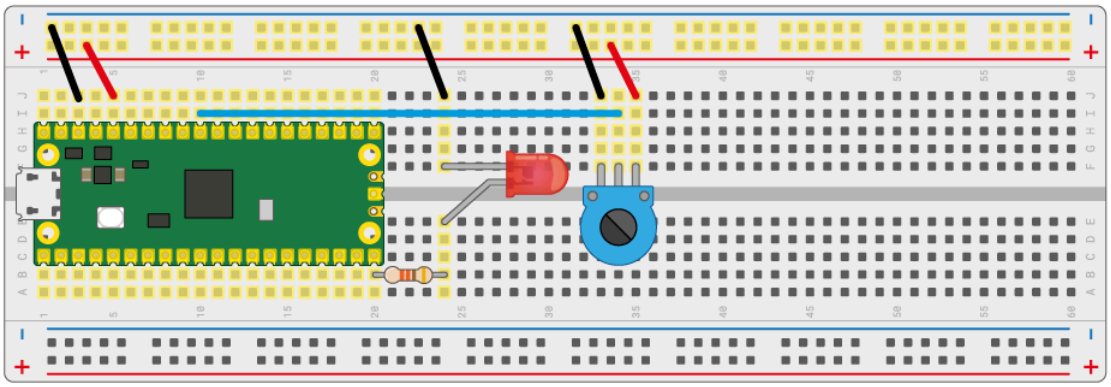
pwm.freq est la fréquence en Hz du signal PWM. La rapport cyclique (*duty cycle*) est le pourcentage de temps sur une période (*rappel : période T = 1 / fréquence*) où le signal est à l'état haut. Plus ce pourcentage est élevé, plus la LED sera lumineuse. Pour la carte Raspberry Pi Pico et en MicroPython, le rapport cyclique est entre 0 et 65 535 : 65 535 pour un rapport cyclique de 100 %, dans ce cas, la luminosité de la LED est maximale. Pour une valeur aux alentours de 32 767, la LED devrait avoir une luminosité intermédiaire, théoriquement à la moitié de la luminosité maximale.

Vous pouvez faire quelques essais en ajustant les valeurs de pwm.freq() et pwm.duty_u16 et de la durée de temporisation en paramètre de l'instruction sleep.

IX - Piloter une LED avec une entrée analogique

Votre Raspberry Pi Pico comporte aussi des entrées analogiques. Un potentiomètre est le composant analogique idéal pour cette activité.

Vous pouvez remplacer le bouton du circuit précédent par un potentiomètre. Suivez le schéma ci-dessous pour le connecter à une broche GPIO analogique.



Dans un nouveau fichier Thonny, ajoutez le code suivant qui affiche la valeur en provenance du potentiomètre et exécutez-le.

```
from machine import ADC, Pin
import time

adc = ADC(Pin(26))

while True:
    print(adc.read_u16())
    time.sleep(1)
```

Tournez le potentiomètre pour voir comment évolue cette valeur. Elle devrait évoluer entre 0 et 65 535. Cette valeur peut être utilisée pour contrôler le rapport cyclique du signal PWM envoyé vers la LED.

À l'exécution du code suivant, la luminosité de la LED devrait varier quand vous tournez le bouton du potentiomètre.

```
from machine import Pin, PWM, ADC

pwm = PWM(Pin(15))
adc = ADC(Pin(26))

pwm.freq(1000)

while True:
    duty = adc.read_u16()
    pwm.duty_u16(duty)
```

X - Alimenter la Raspberry Pi Pico

Si votre Raspberry Pi Pico doit fonctionner sans traîner un câble USB relié à votre ordinateur, vous avez besoin d'une alimentation USB externe. La carte peut être alimentée en toute sécurité entre 1,8 et 5,5 V.

Un programme MicroPython peut s'exécuter automatiquement lors de la mise en alimentation de la carte s'il est sauvegardé avec le nom *main.py*.

Dans Thonny, une fois le programme au point, cliquez sur le menu *File*, puis *Save as*. Lorsque vous y êtes invité, sélectionnez *Raspberry Pi Pico* dans le menu contextuel. Nommez votre fichier *main.py*.

Vous pouvez maintenant débrancher la Raspberry Pi Pico de votre ordinateur et la brancher à votre batterie USB extérieure avec un câble micro-USB.

Une fois la carte alimentée, le programme du fichier *main.py* devrait s'exécuter automatiquement.

XI - Et maintenant ?

Pourquoi ne pas essayer votre Raspberry Pi Pico équipée d'autres composants électroniques : un buzzer, une photorésistance ou même une commande de moteur.

Pour plus de précisions sur l'utilisation de votre Raspberry Pi Pico, vous pouvez consulter la documentation **ici**.

XII - Notes de la rédaction de Developpez.com

Ce livre est la traduction du guide **Getting started with Raspberry Pi Pico** proposé par la **Fondation Raspberry Pi**.

Nous remercions les membres de la rédaction de **Developpez.com** pour le travail de traduction et de relecture qu'ils ont effectué, en particulier : **f-leb**, et **escartefigue**.