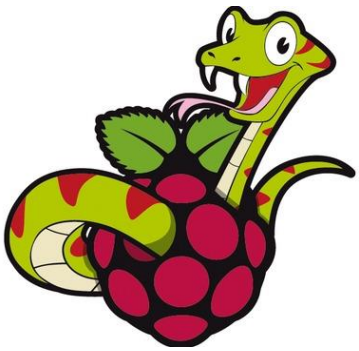
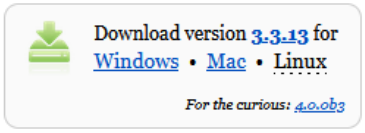
Raspberry PI PICO



Nous allons utiliser la Raspberry PI PICO (appeler PICO par la suite) pour faire notre projet, avec ce document nous allons voir les bases pour l'utiliser.

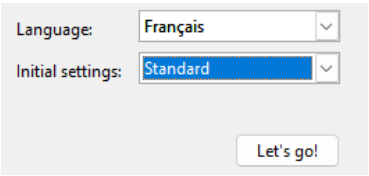
1. Nous utiliserons un IDE (Integrated Developpement Environment ou environnement de développement intégrer (l'endroit où on mettra notre code)), *Thonny* téléchargeable sur https://thonny.org/
   Choisir votre distribution, (Linux dispo dans une autre version) ex : Windows



   a. Première chose à faire l'installer sur votre PC il y a une version Mac, Linux et Windows
   b. Deuxième lancer *Thonny* et le paramétrer pour faire fonctionner la PICO

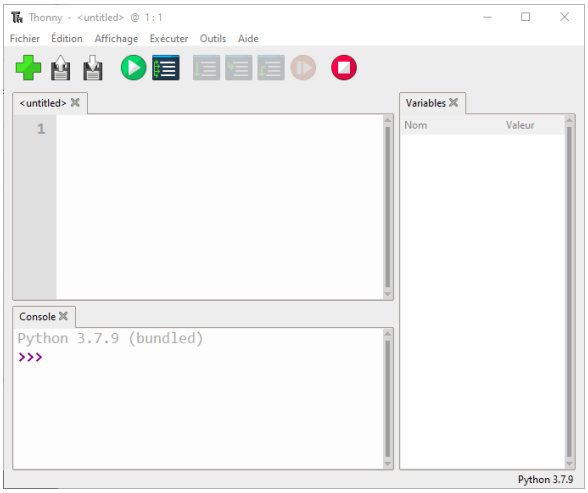
   Français ou à votre convenance

   Et bien mettre standard

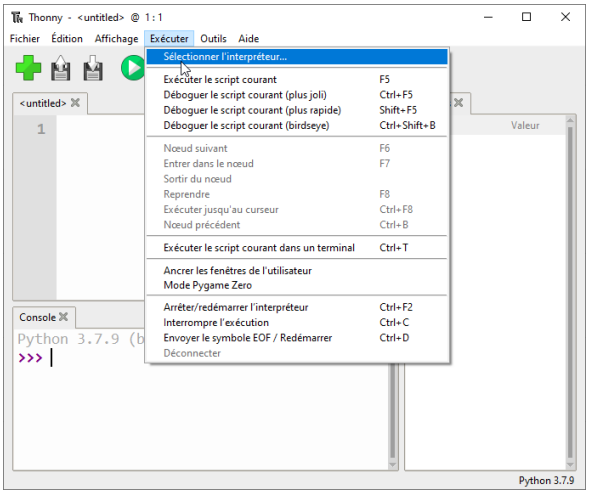


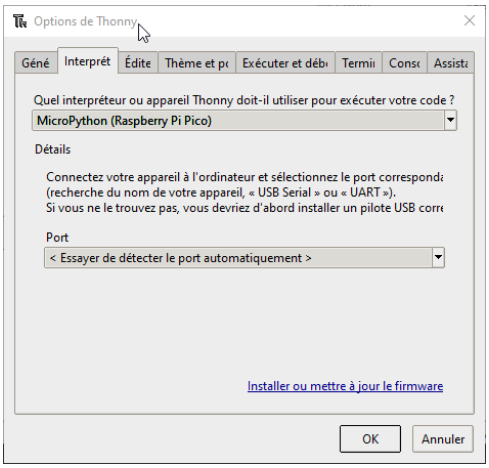
   c. Se familiariser avec les mots clés (voir annexe)

Cette fenêtre devrait apparaitre :



On va ensuite sélectionner l'interpréteur :





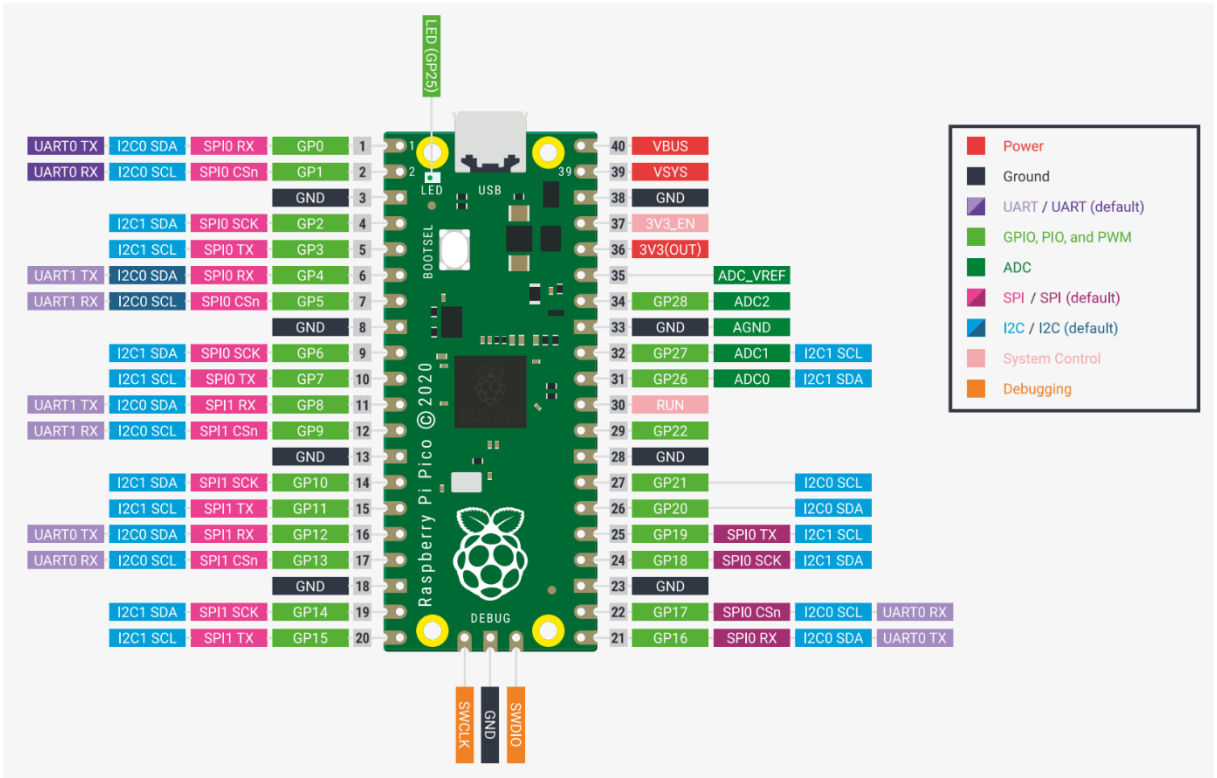
On met l'interpretteur MicroPython (Raspbery PI Pico), on peut laisser le port en automatique, on valide avec OK.

Si ultérieurement, il y a des problèmes et que le port n'est pas détecté automatiquement, il faudra revenir ici pour le sélectionner.

Un message dans la console indique que le composant n'est pas trouvé (device), c'est normal, il faut relancer *Thonny*. Ensuite, on peut brancher et débrancher la PI PICO quand on le souhaite, pour faire les branchements des composants par exemple.

Les différents codes qui seront utilisés par la suite sont disponibles en ligne. À vous de les télécharger si vous le souhaitez, sinon il est possible de les réécrire, à vous de choisir (préconisation : réécrire les choses importantes C-a-d faire du copier/coller de son propre code).

2. Voici les différentes PIN de notre PICO :

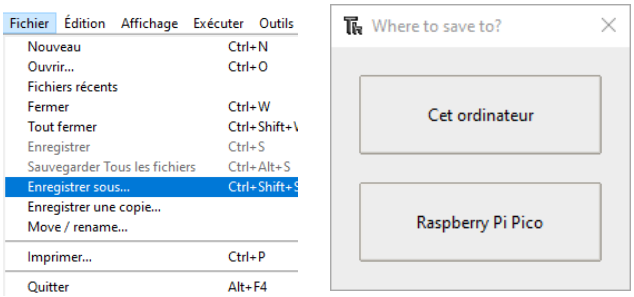


3. Faire clignoter la LED (GP25) présente sur la PICO.

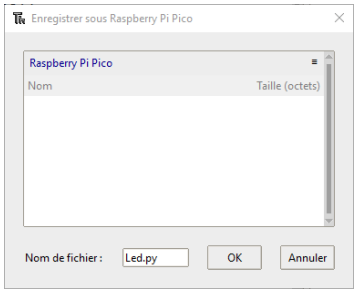
Pour cela on doit écrire le code suivant :

```
import machine
import utime

led = machine.Pin(25, machine.Pin.OUT)

while True :

    led.value(1)
    utime.sleep(2)
    led.value(0)
    utime.sleep(2)

```

Ensuite on l'enregistre sur la PICO :

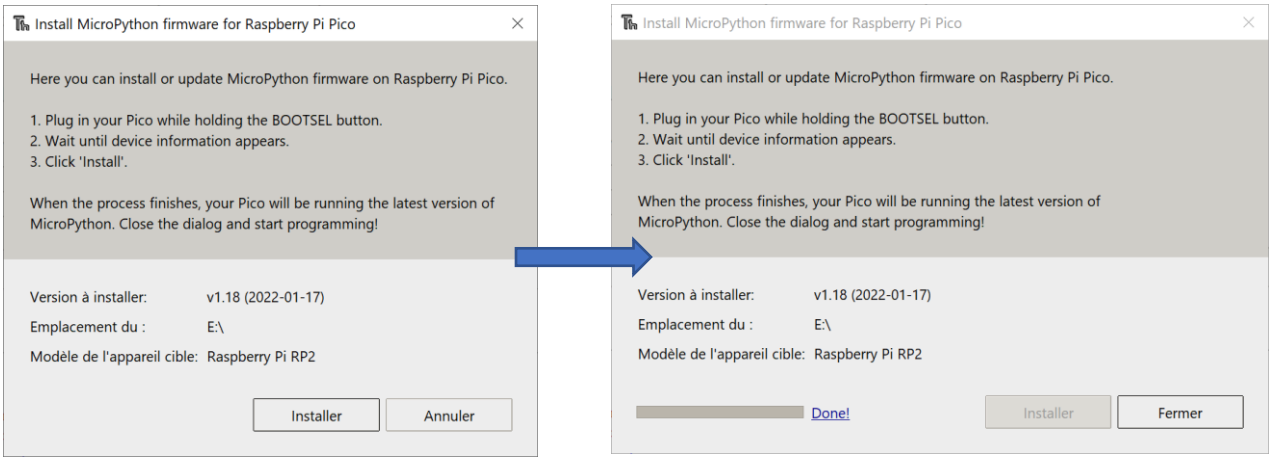


Et on la renomme led.PY :
On n'oublie pas de valider avec OK.



Il n'y a plus qu'à appuyer sur la flèche verte pour faire fonctionner le programme. Pour le stopper on utilise le carré rouge

Il est possible qu'une fenêtre d'erreur apparaisse, si c'est le cas il faut simplement appuyer sur installer et attendre que la barre de chargement se termine pour finir appuyer sur fermer.



Cette étape ne devrait arriver qu'une seule fois, c'est pour « initialiser » la carte et faire en sorte qu'elle puisse communiquer avec THONNY

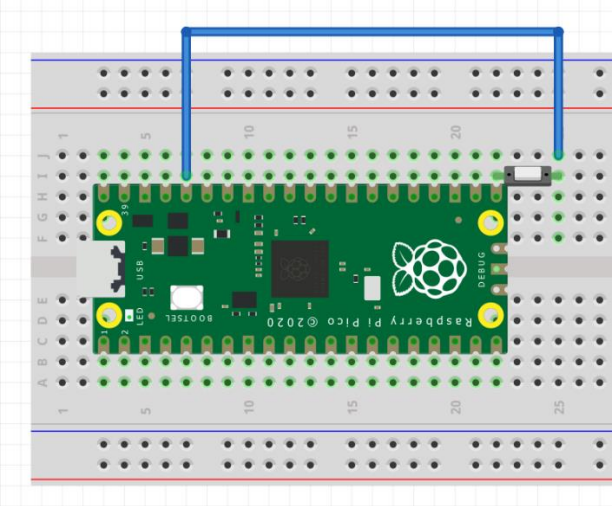
Quelques explications :

```
import machine                          # On ajoute les bibliothèques machine et utime
import utime

led = machine.Pin(25, machine.Pin.OUT)  # On initialise la variable led pour dire que c'est une
                                        # sortie sur la PIN 25 (sur le schéma précédent le
                                        # numéro des pins correspond au nombre après le GP

while True :                            # La boucle infinie, le corps du programme

   led.value(1)                         # On allume la led
   utime.sleep(2)                       # On attend 2sec
   led.value(0)                         # On éteint la led
   utime.sleep(2)                       # on attend 2 sec
```

Derrière les # on peut mettre du commentaire qui ne sera pas regardé par le programme c'est pour que nous humain nous comprenions et donnions des informations aux autres humains qui lisent le code

4. Le micro-rupteur

Branchements :



Programme :

```
import machine                                  # On ajoute les bibliothèques
                                                # machine et utime
import utime

microRupteur = machine.Pin(16, machine.Pin.IN)  # On initialise la variable led
                                                # pour dire que c'est une
                                                # entrée sur la PIN 16


while True:
    utime.sleep_ms(200)                         # On attend 200ms
    print(microRupteur.value())                 # On affiche la valeur lue
                                                # sur microRupteur
```

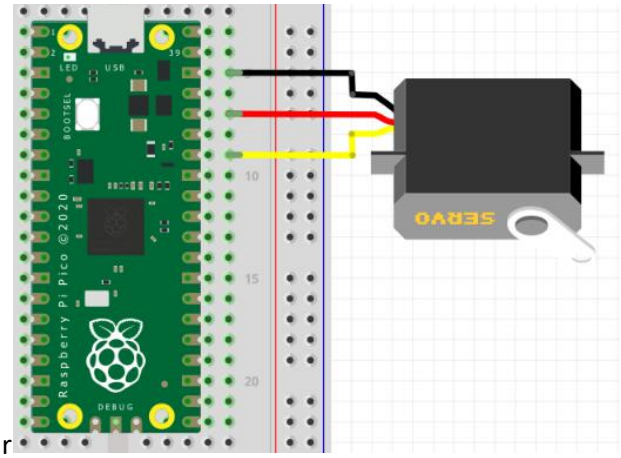
Objectif voir dans la console ce qui se passe quand on appuis sur le « bouton » 0 ou 1 apparait.

Petit exercice :

Allumer la led quand vous appuyez sur le micro-rupteur. (Réponse à télécharger)

5. Le Servo moteur

Branchements :



r

Programme :

```
import machine
import utime

stop = 1500000                          # valeur permettant de stopper le servo moteur
back = 1000000                          # valeur permettant de faire tourner le servo moteur dans un sens
go = 2000000                            # valeur permettant de faire tourner le servo moteur dans l'autre sens

servo = machine.PWM(machine.Pin(28))   # on déclare le servo moteur comme PWM brancher sur la PIN 1

servo.freq(50)                          # on donne la fréquence (50 par défaut)
servo.duty_ns(stop)                     # on arrête le servo moteur

while True :
    servo.duty_ns(stop)                 # on arrête le servo moteur
    utime.sleep(1)                      # on attend 1 sec (servo arrêté 1 sec)
    servo.duty_ns(back)                 # on arrête le servo moteur
    utime.sleep(2)                      # on attend 2 sec (servo tourne dans un sens 2 sec)
    servo.duty_ns(stop)                 # on arrête le servo moteur
    utime.sleep(1)                      # on attend 1 sec (servo arrêté 1 sec)
    servo.duty_ns(go)                   # on arrête le servo moteur
    utime.sleep(2)                      # on attend 2 sec (servo tourne dans l'autre sens 2 sec)
```

Actuellement il fait des allers-retours avec une pause, il y a la possibilité de changer les valeurs.

Petit exercice :

Avec 2 boutons, si j'appuie sur un bouton je tourne dans un sens, si j'appuie sur l'autre, dans l'autre sens sinon si j'appuie sur les deux OU sur aucun je m'arrête. Possibilité de s'amuser avec « and » et « or ».

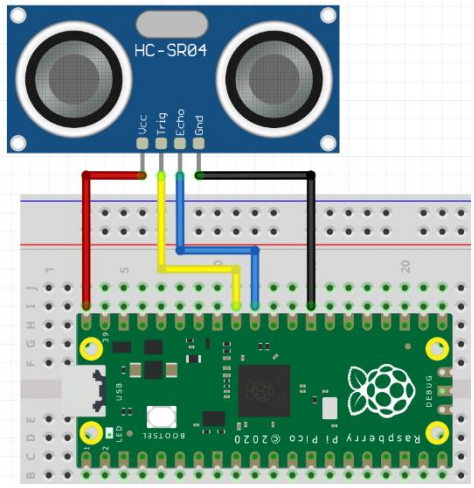
6. Ultrason

Petite subtilité il faut aller télécharger la bibliothèque HCSR04 sur le lien : (aussi dispo dans les documents fournis)
https://github.com/rsc1975/micropython-hcsr04

Il faut ensuite l'ouvrir avec *Thonny* Faire « enregistrer sous » et l'enregistrer sur la PICO attention à bien respecter le nom : hcsr04.py

Ensuite comme on commence à avoir l'habitude

Branchement :



Programme :

```
from hcsr04 import HCSR04    # https://github.com/rsc1975/micropython-hcsr04
import utime

#definition ultrason attention brancher sur 5V le VCC
sensor = HCSR04(trigger_pin=27, echo_pin=26) # On déclare le capteur sur la PIN 27 Pour le TRIGGER
                                             # et sur la PIN 26 pour le ECHO

#boucle de fonctionnement

while True:

    utime.sleep_ms(200)                      # on attend 20ms
    distance = sensor.distance_cm()          # on récupère la valeur dans la variable distance
    print(distance)                          # on affiche la distance dans la console

```

Petit exercice :

Allumer 1,2,3 LEDs en fonction de la distance de détection 100cm, 50cm, 25 cm par exemple

Infrarouge

2 étapes :

- Allumer la LED IR voir 3
- Détecter la présence d'infrarouge (même principe qu'un bouton)

Attention, la diode IR fonctionne avec une tension de 1.5V, 20mA et notre PICO fourni 3.3V il faut donc rajouter une résistance 3.3-1.5 = 1.8V R=U/I donc R= 1.8/0.02 = 90 ohms on utilisera une résistance de 100ohms.

En réalité pour la LED IR on n'est pas obligé de la commander, elle peut rester allumer continuellement.

On vous laisse faire, avec ce qu'on a vu précédemment vous devriez y arriver.

Annexes

Nous avons réellement besoin des conditions (if),

des boucles (while) puisqu'on l'utilise dans tous nos programme éventuellement « for » pour certain d'entre vous.

Et des fonctions (même si non obligatoire mais bien plus pratique pour les feignants, il faut être un feignant intelligent)

Les conditions :

Le si … sinon si … sinon …
Le principe : si j'ai faim
je mange
sinon si j'ai envie d'un bonbon
je mange un bonbon,
sinon
je ne fais rien

Si on le traduit avec des nombres et en python:

```
x = -2
if x > 0:
        print(x, "est positif")
elif x==0 :
        print(x, "est nul")
else :
        print(x, "est négatif")

print("Fin")
```

Il est possible de ne mettre que le si, ex :
if a==b :
print(a égale b)
donc tant que ce n'est pas vrai il n'y a rien d'affiché

on peut mettre si et sinon ex :
if a==b :
print(« a égale b »)
else :
print(« a différend de b »)
ou la totale

etc. par contre il faut forcément que la première comparaison soit un « si » ou if .

Les opérateurs de comparaison possible sont :

x == y x est égal à y
x != y x est différent de y
x > y x est plus grand que y
x < y x est plus petit que y
x >= y x est plus grand ou égal à y
x <= y x est plus petit ou égal à y

Le cours :

https://courspython.com/tests.html

Pour la suite autant utiliser les cours disponibles sur les liens suivants

Les boucles :

Permettant de répéter une opération sans réécrire X fois le code

https://courspython.com/boucles.html

Les fonctions :

Permettant de faire une action spécifique à plusieurs endroit dans le code et en ne l'écrivant qu'une fois (c'est du bonus)

Dispo ici :

https://courspython.com/fonctions.html

Toutes les autres bases disponibles ici : (pas forcément utile pour ce qu'on a à faire, ça peut toujours être utile.

https://courspython.com/bases-python.html