

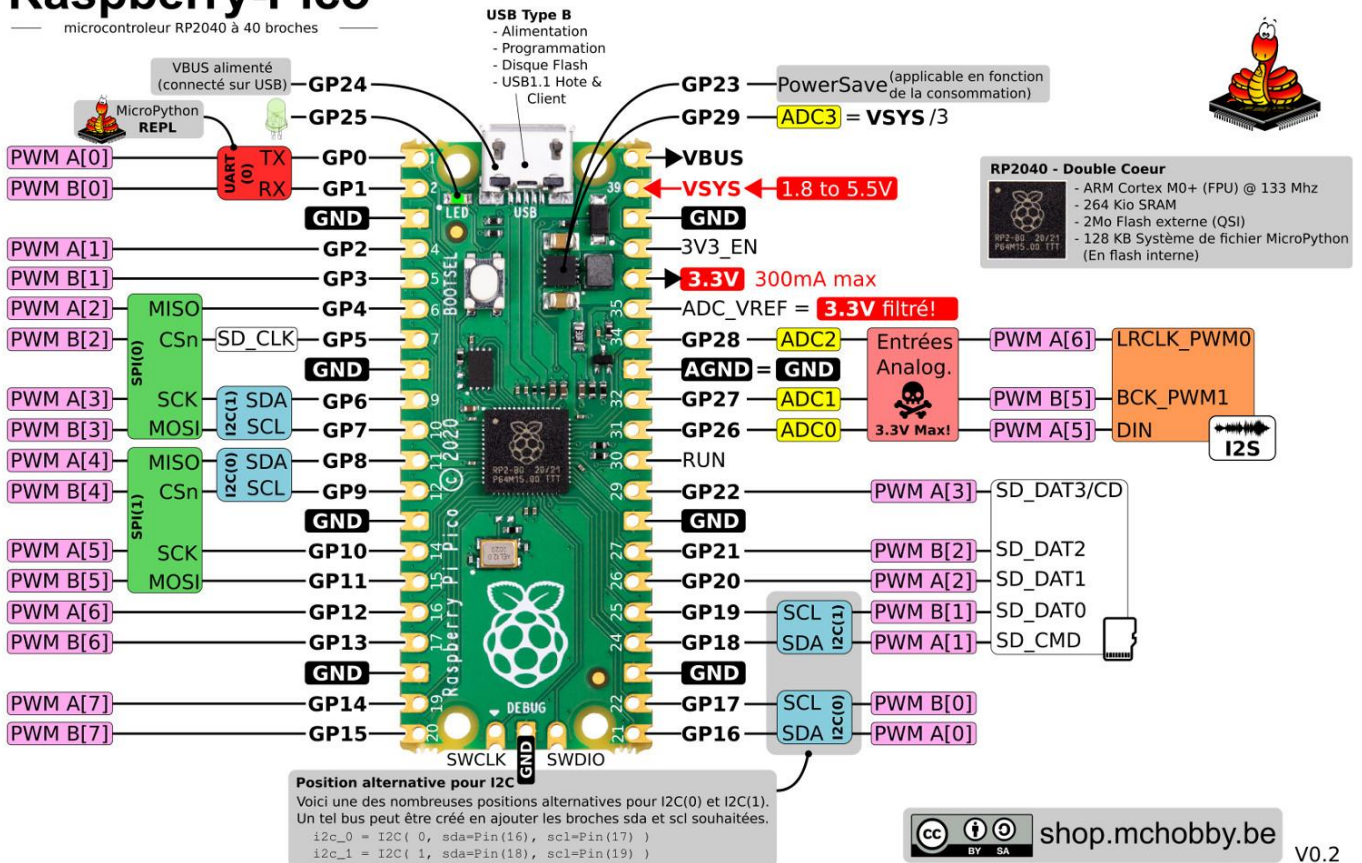
## 1. Mise en situation

L'objectif est d'apprendre à programmer notre voiture pour la faire avancer et gagner la course. Pour cela nous allons nous servir de cette carte de électronique programmable pour découvrir les bases de la programmation.

## 2. Présentation de la carte Raspberri-Pico:

### Raspberry-Pico

microcontrôleur RP2040 à 40 broches



- 1) Le cœur de la carte, un ARM.
- 2) Le port USB, qui permet entre autre d'alimenter la carte en +5V.
- 3) DELS de visualisation. L'une (orange) permet de tester la carte, les 2 autres (vertes) permettent de visualiser la liaison série en émission Tx et en réception Rx.



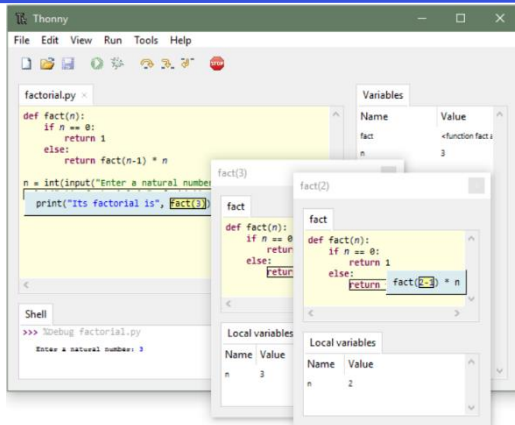
- 4) Connecteur permettant une liaison filaire avec des composants extérieurs (capteurs, boutons poussoir, Dels, transistor...)

Nous utiliserons le shield Grove pour câbler les composants externes qui se trouvent sur les différents modules

## 3. Le logiciel de programmation :

Nous allons utiliser le logiciel de programmation Thonny





3 le « **Bootloader** », de « **Firmware** », et de « **Programme micropython** ».

- Tout d'abord, il faut savoir que tous les Raspberry Pi Pico disposent d'un « **bootloader intégré** », dans leur mémoire ROM ; c'est à dire un programme de démarrage « gravé dans le dur », dans la mémoire morte du microcontrôleur. Ce programme permet entre autre :

- => la communication entre le Raspberry Pico et un ordi, via le port USB
- => et l'exécution de code programme, stocké en mémoire FLASH

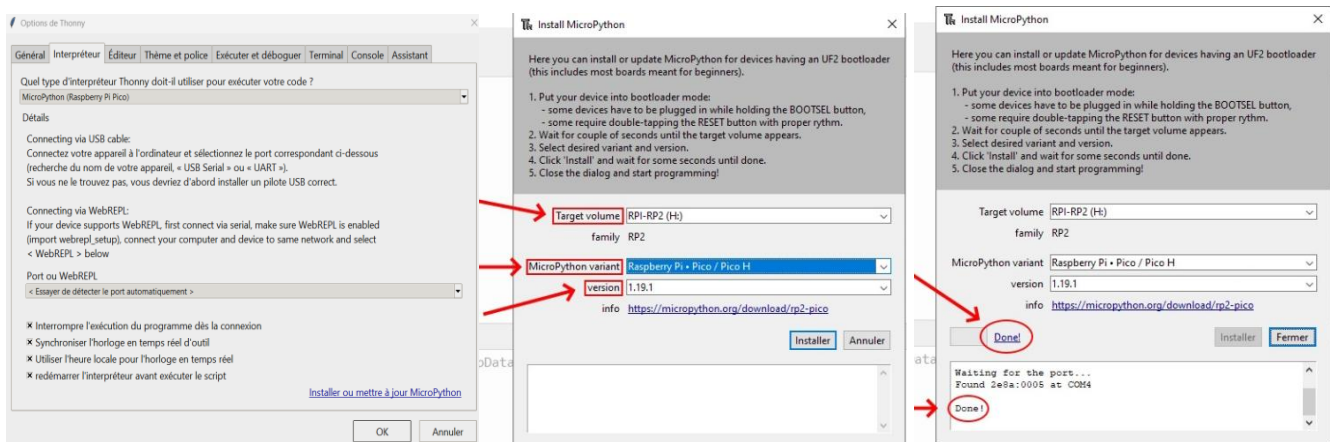
La mémoire FLASH, peut contenir :

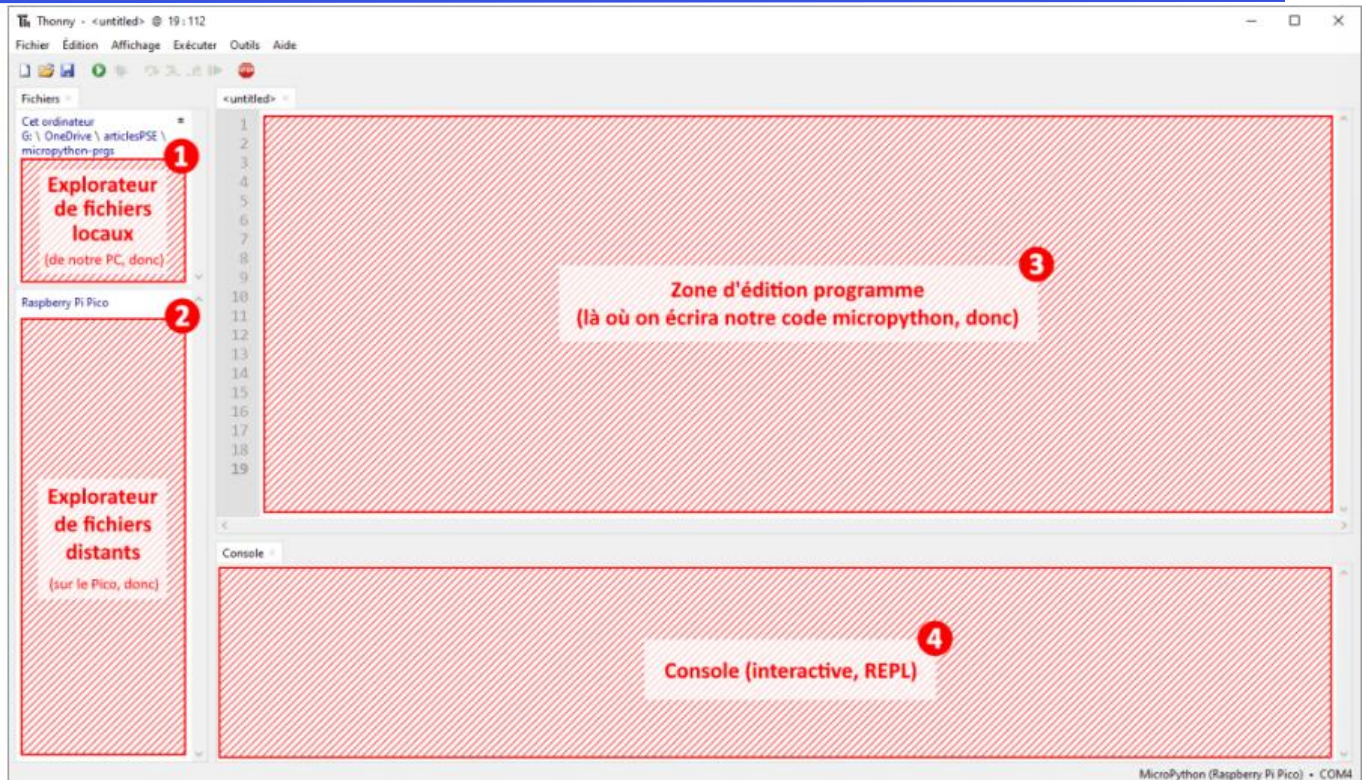
- un programme (écrit en langage machine, directement compréhensible par le  $\mu C$ )
- et d'éventuelles données

pour qu'un Raspberry Pi Pico puisse exécuter des scripts écrits en MicroPython, nous allons utiliser la mémoire FLASH de cette manière là :

- la partie programme hébergera un interpréteur micropython (c'est à dire un micrologiciel, aussi appelé firmware en anglais, permettant de transformer à la volée du code MicroPython, en langage machine, que le microcontrôleur saura comprendre)
- et la partie données hébergera nos fichiers-programmes écrits en MicroPython

**Si besoin :** Vérification du firmware : Ouvrir thonny => configurer l'interpréteur comme ci-dessous :





## 4. Premiers programmes :

### 4.1. Le code minimal :

```
# =====
# Programme BLINK (faisant clignoter la LED embarquée sur la carte Pico)
# =====
from machine import Pin
from time import sleep

led = Pin(18, mode=Pin.OUT)

while True:    # Boucle perpétuelle (celle-ci ne s'arrêtera jamais, car la condition "True" est toujours
               # "vraie")
    led.on()   # Allumage de la LED du Pico
    sleep(1)   # Mise en pause du programme, pendant 1 seconde
    led.off()  # Extinction de la LED du Pico
    sleep(1)   # Nouvelle pause programme d'une seconde, et rebouclage sur le while

# Remarque : avant led.high(), sleep(1), et led.low(), ce sont des tabulations qui sont présentes, et non
#             des espaces (qui seraient faits avec la barre d'espace). Si jamais vous copiez/collez ce code,
#             et qu'il y a des espaces à la place des tabulations, il faudra les remplacer par des tabulations !
```

#### Initialisation :

Les **librairies** sont des bibliothèques préprogrammées que nous allons utiliser.

Les **variables** sont initialisées en début ainsi que tout ce qui va se trouver connecté à la carte, broches en



entrée ou en sortie...

### Boucle while:

On écrit le contenu du programme. Cette instruction s'exécute en **boucle infinie**, lorsque la dernière ligne de programme est exécutée, le programme reprend de la première ligne.

### Tester ce programme

#### 4.2. Exercice 1 :

Quand notre voiture va s'arrêter elle doit le signaler aux autres voitures sur la piste. Pour cela **modifier** le programme pour qu'il permette de faire clignoter une DEL connectée sur la broche numérique n°20. Elle doit rester allumée pendant 500ms et éteinte pendant 500ms.

**Enregistrer** votre programme dans votre répertoire réseau.

### Tester votre programme

#### 4.3. Exercice 2 :

Pour pouvoir faire avancer notre voiture on va lui en donner l'ordre par l'intermédiaire d'un bouton poussoir. **Créer** un nouveau fichier bouton.py, comme vous

## 5. Les conditions

---

### Exercice 1 :

Ce programme doit permettre d'allumer une LED lorsqu'on appuie sur le bouton, puis s'éteint quand on relâche le bouton poussoir. Pour cela nous allons avoir besoin de condition.

En python nous utiliserons ceci :

```
if (Condition) :  
    Opération1  
else :  
    Opération2
```

La led doit être branchée sur le Pin 20 et le bouton poussoir doit être branché sur le Pin 18.

### Ecrire et Tester votre programme

#### Exercice 2 :

La led doit maintenant s'allumer quand on appuie sur le bouton poussoir et s'éteindre quand on relâche le bouton poussoir.

### Ecrire et Tester votre programme

## 6. Exercices d'application

---

### 6.1. Exercice 1 : Allumage de sécurité



On souhaite maintenant que la LED ne s'allume que si deux boutons sont appuyés simultanément.

**Question 1** - Dessiner l'algorithme décrivant le fonctionnement du processus.

**Application 1** - Écrire le programme en le commentant.

**Application 2** - transférer et tester le programme.

### 6.2. Exercice 2 : Fonctionnement du bouton en appui momentané

On veut maintenant que le système s'allume sur un appui d'un bouton et s'éteigne lors d'un nouvel appui.

**Expérimentation 1** - Mêmes questions qu'à l'exercice 1