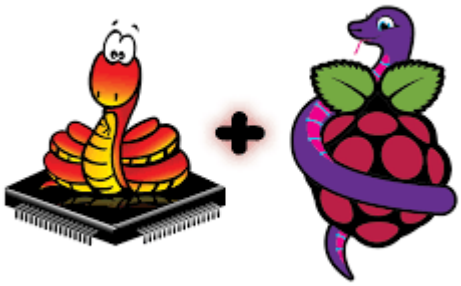
Tutoriel "Etape par Etape" - Premiers programmes en MicroPython ou CircuitPython avec une carte Raspbery Pi Pico W



[Mise à jour le :19/8/2023]

- **Ressources**
  - MicroPython.org
  - MicroPython documentation
  - Datasheet Raspberry Pi Pico Python SDK
  - IDE Thonny
  - Getting Started with Raspberry Pi Pico W using MicroPython
- **Lectures connexes**
  - **Installer MicroPython** - MicroPython - Les modules Espressif ESP32 et ESP8266
  - **Raspberry Pi Pico** - Les modules Raspberry Pi Pico et Pico W
  - **Programmez** ! Juillet/Août 2019
  - **Elektor 489** Mai/Juin 2021

Préambule

Ce tutoriel a pour objectif de vous faire découvrir la programmation de la carte Raspberry Pi Pico en MicroPython ou CircuitPython à l'aide de l'éditeur Thonny. Après une description rapide du matériel, vous le connecterez à l'IDE Thonny afin de réaliser les programmes suivants :

- 1. **blink.py** - Commande d'une sortie numérique : faire clignoter la LED de la carte Raspberry Pi Pico !
- 2. **angle** - Lecture d'une entrée analogique : afficher la position angulaire d'un axe (en °)
- 3. **afftemp.py** - Afficheur graphique : afficher la température délivrée par le capteur interne au microcontrôleur sur un afficheur graphique.
- 4. **affbme280.py** - Afficheur graphique et capteur numérique : afficher la température, l'humidité et la pression ambiantes.
- 5. **afftsl2591** - Afficheur graphique et capteur numérique : afficher la luminosité ambiante.

1. Le matériel

- **Ressource**
  - MicroPython - Les modules Raspberry Pi Pico et Pico W*

2. L'IDE Thonny

2.1 Présentation

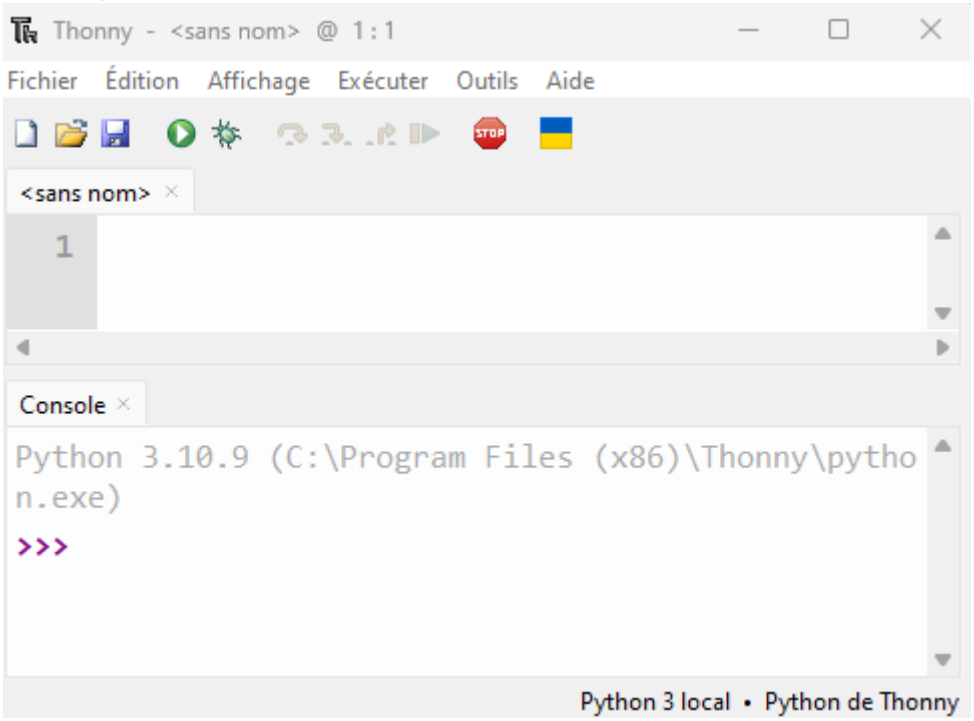
- **Ressources**
  - Site du logiciel.
  - Quick reference for the RP2¶

L'IDE **Thonny** propose aux débutants un **environnement de programmation** intuitif pour **Python** (MicroPython, CircuitPython), aussi facile à déployer qu'à utiliser. Dans un souci de simplicité, Thonny embarque la dernière version de Python qu'il installe en même temps que l'EDI. Le programme est par ailleurs préconfiguré afin d'éviter les difficultés liées au paramétrage de l'application.

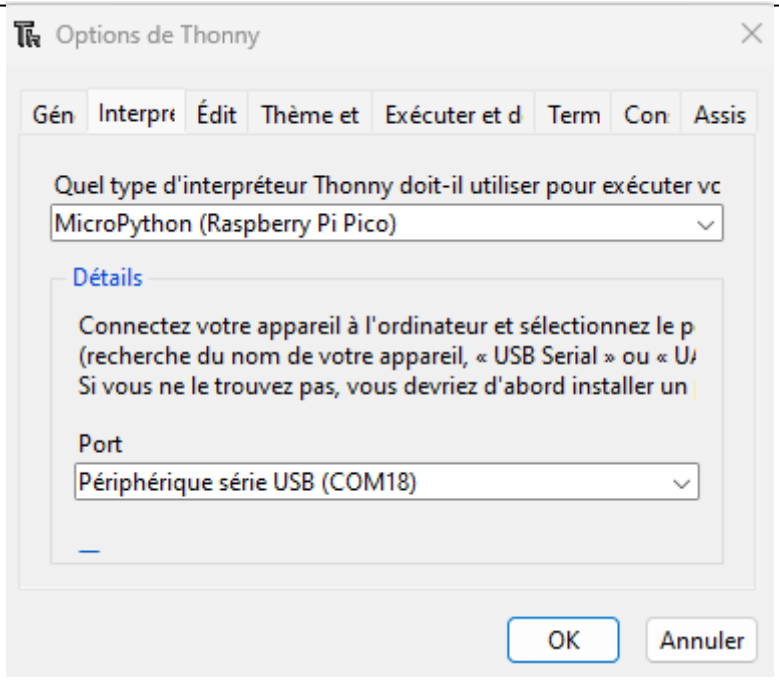
2.2 Connexion de la carte Rp2 à l'IDE

- MicroPython
- CircuitPython

1. **Brancher** la carte RP2[1)] au PC avec le câble USB.
2. **Ouvrir** l'IDE Thonny

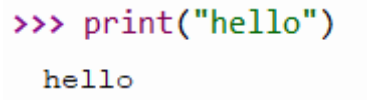


3. Dans le menu, sélectionner **outils → options… → Interpréteur**.
   Sélectionner l'interpréteur **MicroPython (Raspberry Pi Pico)** et le **port** sur lequel est connectée la carte (COM18 ci-dessous est un exemple, votre port peut être différent).

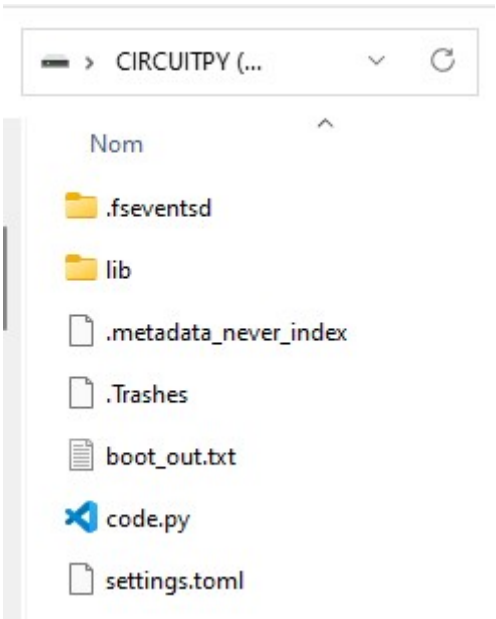


4. Un message indiquant la **version de MicroPython** sur la carte doit apparaître dans la console comme dans l'exemple ci-dessous .

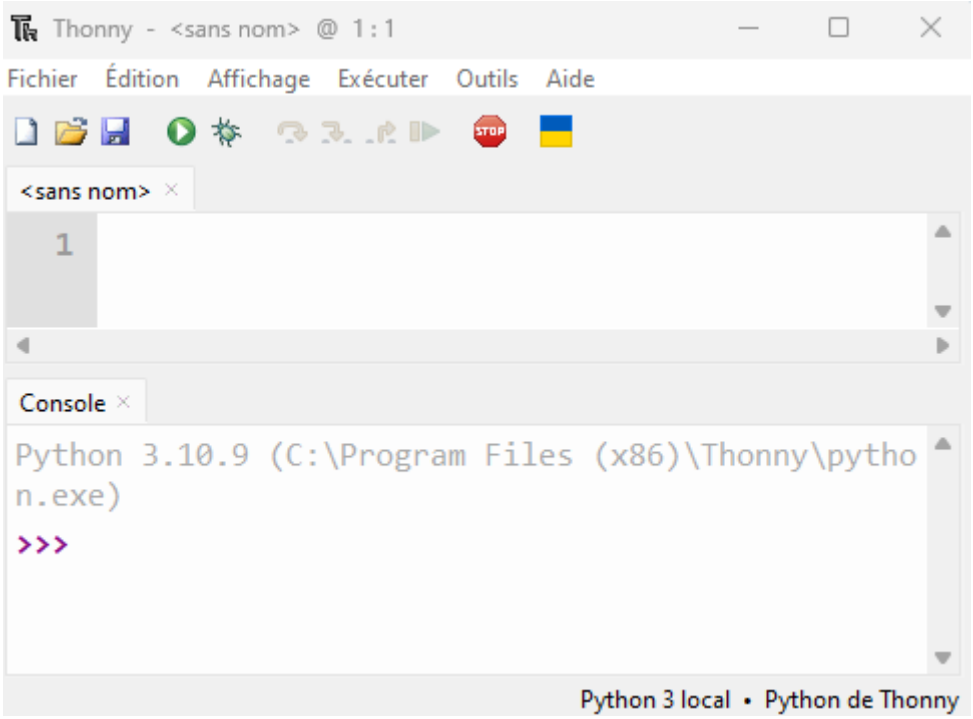
5. **Tester** la communication avec la carte en entrant print(“hello”) dans la console.



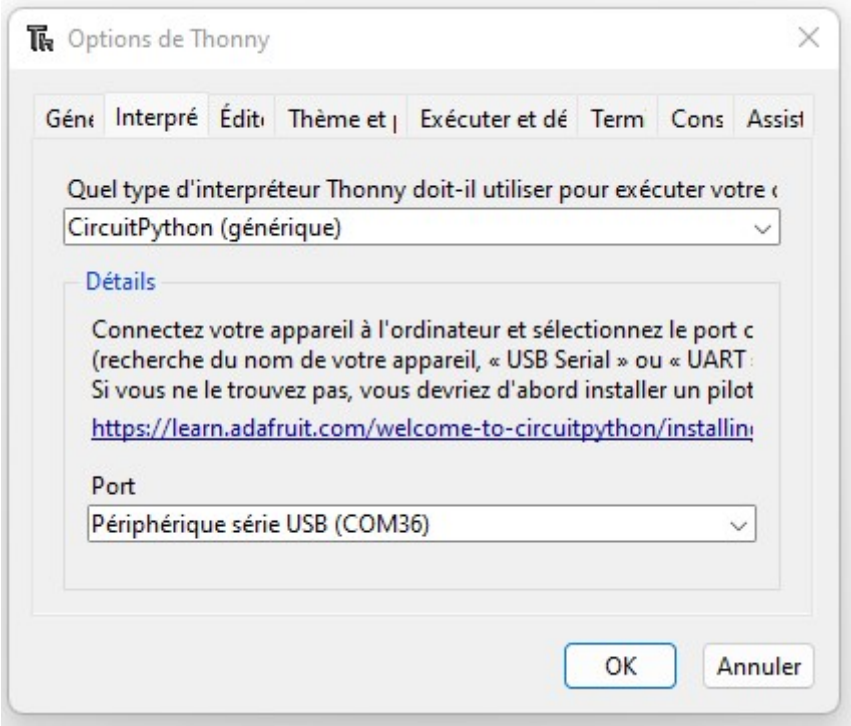
1. **Brancher** la carte RP2[2)] au PC avec le câble USB. Une fenêtre de l'explorateur de fichier s'ouvre comme ci dessous sur un nouveau lecteur USB appelé **CIRCUITPY**.
Ce lecteur est l'endroit où l'on dépose les fichiers contenant le code et les bibliothèques nécessaires au programme.



2. **Ouvrir** l'IDE Thonny



3. Dans le menu, sélectionner **outils → options... → Interpréteur**.
   Sélectionner l'interpréteur **CircuitPython (générique)** et le **port** sur lequel est connectée la carte (COM36 ci-dessous est un exemple, votre port peut être différent).



4. Un message indiquant la **version de CircuitPython** sur la carte doit apparaître dans la console comme dans l'exemple ci-dessous .

```
  Adafruit CircuitPython 8.1.0 on 2023-05-22; Raspberry Pi Pico W with rp2040
Adafruit CircuitPython 8.1.0 on 2023-05-22; Raspberry Pi Pico W with rp2040
>>>
```

5. **Tester** la communication avec la carte en entrant print(“Hello”) dans la console.

```
>>> print("hello")
  hello
```

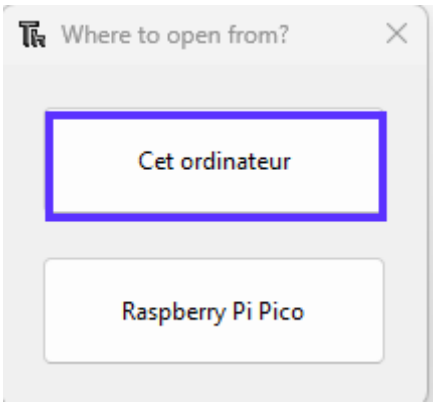
2.3 Installation des bibliothèques sur la carte Rp2

- MicroPython
- CircuitPython

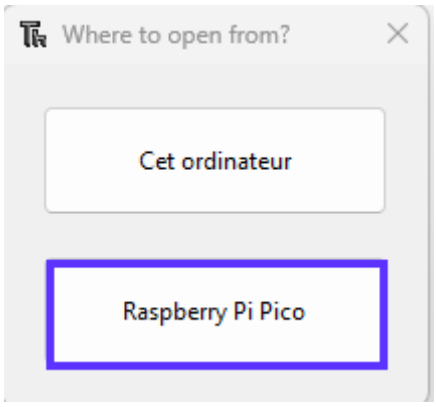
L'utilisation de certains capteurs, afficheur etc. avec le Raspberry Pi Pico nécessitent d'avoir préalablement installé des bibliothèques dans l'espace mémoire de la carte.

Exemple : installation de la bibliothèque ***ssd1306.py*** nécessaire à l'afficheur graphique utilisé dans ce tutoriel.

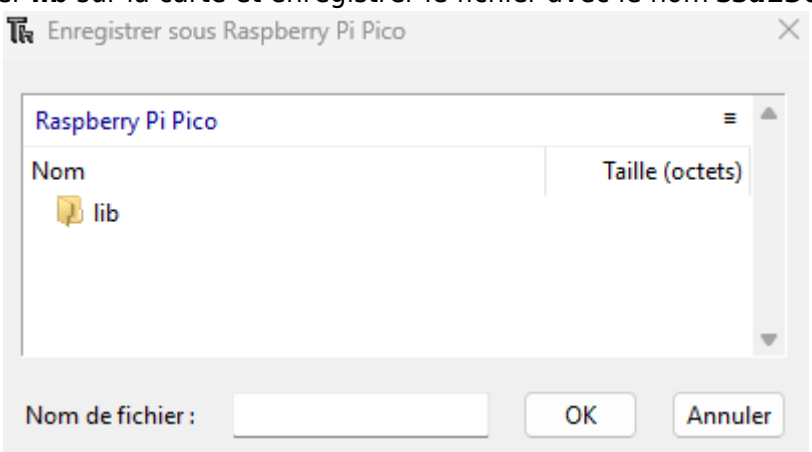
1. **Sélectionner** l'icône **ouvrir…** (Ctrl+O) et choisir **Cet ordinateur** dans la boîte de dialogue.



2. Dans l'explorateur de fichiers, entrer le **chemin** précisé dans le paragraphe **Préparation** du **document réponse** et **ouvrir *ssd1306.py*** dans Thonny.
3. **Sauvegarder** le fichier dans la carte en suivant la démarche suivante :
    1. A partir du chemin **Fichier → Enregistrer sous…**(Ctrl+Shift+S), sélectionner **Raspberry Pi Pico**



    2. Ouvrir le dossier **lib** sur la carte et enregistrer le fichier avec le nom ***ssd1306.py***

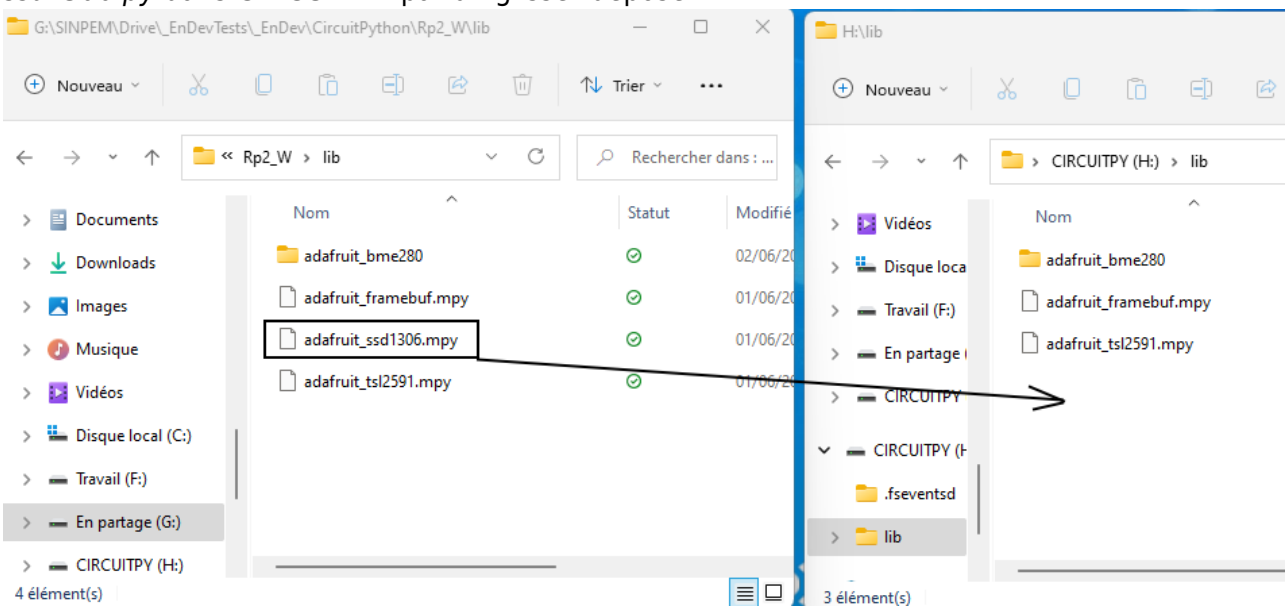


4. Fermer le fichier ***ssd1306.py*** dans Thonny.

L'utilisation de certains capteurs, afficheur etc. avec le Raspberry Pi Pico nécessitent d'avoir préalablement installé des bibliothèques dans l'espace mémoire de la carte.

Exemple : installation de la bibliothèque ***ssd1306.py*** nécessaire à l'afficheur graphique.

1. **Ouvrir** une **nouvelle fenêtre** de l'explorateur de fichier et entrer le **chemin** précisé dans le paragraphe **Préparation** du **document réponse**.
   Placer cette fenêtre avec celle de CIRCUITPY en côte à côte comme ci-dessous. Placer le fichier *ssd1306.py* dans **CIRCUITPY** par un glisser-déposer.



3. Programmation

3.1 Prog. 1 - Commande d'une sortie numérique

> **Objectif**
> Faire clignoter une LED externe à la carte Raspberry Pi Pico !

3.1.1 Généralités

*« Dans un système à base de **microcontrôleur**, on appelle **entrées-sorties** les échanges d'informations entre le processeur et les périphériques qui lui sont associés. Ainsi, le système peut réagir à des modifications de son environnement, voire le contrôler. Elles sont parfois désignées par l'acronyme **I/O**, issu de l'anglais **I**nput/**O**utput ou encore **E/S** pour **E**ntrées/**S**orties. »* Wikipédia

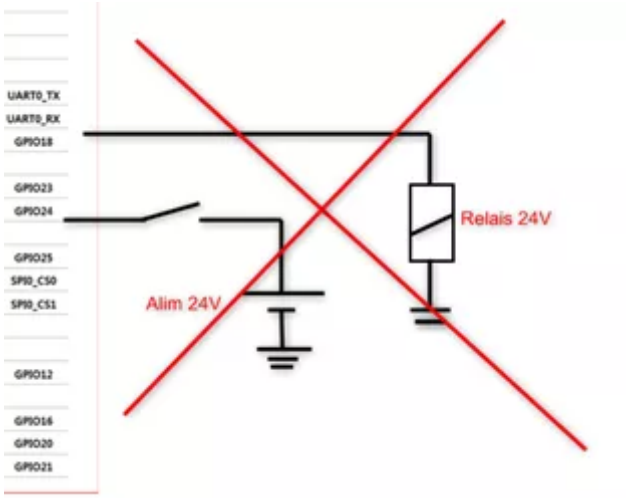
Un **microcontrôleur** dispose de broches pouvant être contrôlées par un logiciel. Elles peuvent se comporter comme des entrées ou des sorties, d'où le nom “entrée / sortie à usage général”, ou **GPIO** (**G**eneral **P**urpose **I**nput **O**utput. Le nombre de broches d’un microcontrôleur étant limité, il est fréquent d’avoir plusieurs fonctionnalités sur une même broche.

Exemple : la **broche** (pin) 38 du microcontrôleur est reliée à la broche **31** de la carte RP2. C'est une entrée/sortie à usage général, identifiée par **GP26**. Elle peut être **configurée** comme une entrée analogique (**ADC0**), une entrée/sortie numérique général ou I2C (**I2C1 SDA**).

Le choix (configuration) est fait dans le logiciel. La **configuration** de la broche 38 du microcontrôleur en tant que sortie numérique s'écrit : Pin(26,Pin.OUT)

Pour éviter de faire référence à des valeurs électriques (tension ou intensité), on définit souvent l’état d’un signal numérique en utilisant la logique booléenne.
- **true** (« 1 » logique) correspondra par exemple à 5V ou 3,3V
- **false** (« 0 » logique) correspondra à 0V.

Une sortie numérique est fragile. Ne **JAMAIS** la relier à un générateur.
Une sortie numérique délivre **très peu de puissance** (quelques centaines de mW). Il n’est donc pas possible de la relier directement à un actionneur (moteur). Il est nécessaire de placer une interface de puissance (hacheur, relais) entre elle et l’actionneur à commander.



3.1.2 Activités de programmation

- MicroPython
- CircuitPython

- **Ressources** sur Micropython.org.
  - module machine | module time | class Pin – control I/O pins
  - Quick reference for the RP2, Pins and GPIO

- **Etape 1** - Créer un nouveau programme
  1. Cliquer sur l'icône **Nouveau** (Ctrl+N).
  2. Copier-coller le code de l'exemple ci-dessous dans l'espace de travail de l'éditeur.

blink.py

```
# -----------------------------------------------------------------
# Exemple de code pour faire clignoter la led de la carte
# Matériels : Raspberry Pi Pico, Shield Grove
# Langage : MicroPython
# IDE : Thonny
# Source : blink.py
# -----------------------------------------------------------------
from machine import Pin
import time

# Configuration de l'entrée sortie connectée à la led de la carte
led = Pin('LED', Pin.OUT)  # 'LED' <=> 25 (GPIO25)

while (True):
    led.on()
    time.sleep(0.5)  # Attente 0,5s
    led.off()
    time.sleep(0.5)
```

- **Etape 2** - Enregistrer le programme
  - Enregistrer le programme dans votre compte sur le serveur en le nommant comme ci-dessus.
- **Etape 3** - Tester le programme
  - Exécutez le programme en cliquant sur  .

MODIFICATION du PROGRAMME 1

Une led externe à la carte étant placée sur **GP16** (accessible sur le **connecteur D16** du Shield Groove), **modifier** le code de l'exemple de telle sorte que la led externe s'éclaire pendant 100ms toutes les secondes.
Remarque : en MicroPython, le texte 'LED' de l'exemple doit être remplacé par le numéro de l'entrée/sortie à usage général.

- **Ressources** sur docs.circuitpython.org.
  - **digitalio** - Basic digital pin support
- **Etape 1** - Créer un nouveau programme

1. Cliquer sur l'icône **Nouveau** (Ctrl+N)
2. Copier-coller le code de l'exemple ci-dessous dans l'espace de travail de l'éditeur.

blink.py

```
# -----------------------------------------------------------------
# Exemple de code pour faire clignoter la led de la carte
# Matériels : Raspberry Pi Pico, Shield Grove
# Langage : CircuitPython
# IDE : Thonny
# Source : blink.py
# -----------------------------------------------------------------
import board
import digitalio
import time

# Configuration de l'entrée sortie connectée à la led de la carte
led = digitalio.DigitalInOut(board.LED)
led.direction = digitalio.Direction.OUTPUT

while True:
    led.value = True
    time.sleep(0.5)  # Attente de 0,5s
    led.value = False
    time.sleep(0.5)
```

- **Etape 2** - Enregistrer le programme
  - Enregistrer le programme avec le nom de l'exemple ci-dessus dans votre compte sur le serveur.
- **Etape 3** - Tester le programme
  - Exécutez le script en cliquant sur  .

MODIFICATION du PROGRAMME 1

Une led externe à la carte étant placée sur **GP16** (accessible sur le **connecteur D16** du Shield Groove), **modifier** le code de l'exemple de telle sorte que la led externe s'éclaire pendant 100ms toutes les secondes.
Remarques : dans le langage CircuitPython, le texte *LED* de l'exemple doit être remplacé par l'entrée, sortie à usage général (GPIO) sur laquelle est connectée la led externe. La liste des GPIO connues de CircuitPython peut être visualisée dans la console avec la commande *dir(board)*.

3.2 Prog. 2 - Lecture d'une entrée analogique

> **Objectif**
> Afficher la position angulaire (en °) d'un axe dans la console de l'IDE.

3.2.1 Matériels

- **Capteur** : Rotary angle sensor Grove

3.2.2 Généralités

- Un signal analogique : petits rappels sur le site Zeste de savoir.

3.2.3 Activités de programmation

- MicroPython
- CircuitPython

- **Ressources** sur Micropython.org.
  - ADC (analog to digital conversion) | module time | class Pin – control I/O pins.

- **Etape 1** - Créer un nouveau programme
  1. Cliquer sur l'icône **Nouveau** (Ctrl+N).
  2. Copier-coller le code de l'exemple ci-dessous dans l'espace de travail de l'éditeur.

angle.py

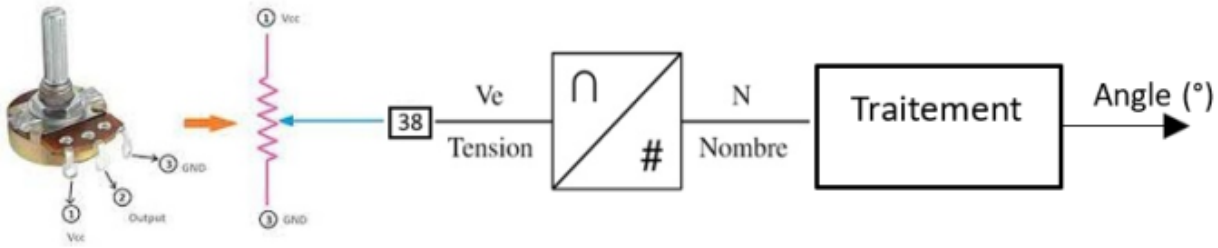
```
# -------------------------------------------------------------------
----------
# Exemple de code pour afficher la valeur N délivrée par le
# convertisseur analogique numérique ADC0
# Matériels : Raspberry Pi Pico, Shield Grove, potentiomètre 10k
# Langage : MicroPython
# IDE : Thonny
# Source : angle.py
# -------------------------------------------------------------------
----------
# Identification des entrées analogiques disponibles sur le Shield
Grove
# Shield    :         RP2040
#           : ADC       : GPIO
# ----------------------------------
#   A0      : ADC0      : GP26
#   A1      : ADC0,ADC1 : GP26, GP27
```

```
#   A2      : ADC1,ADC2 : GP27,GP28
# ------------------------------------------------------------------------
----------
from machine import ADC, Pin
import time

# Le potentiomètre 10k est connecté à l'entrée analogique A0 du shield.
pot = ADC(Pin(26))

while (True):
    # Lecture de la valeur délivrée par le convertisseur analogique
numérique
    # N est un entier positif codé sur 16 bits
    N = pot.read_u16()
    print(f"N = {N}")  # Affichage dans la console
    time.sleep(1) # Attente 1s
```

- **Etape 2** - Enregistrer le programme
  - Enregistrer le programme dans votre compte sur le serveur en le nommant comme ci-dessus.
- **Etape 3** - Tester le programme
  - Exécutez le programme en cliquant sur  .

MODIFICATION du PROGRAMME 2

Sachant que la tension Ve maximale à la sortie du potentiomètre est égale à **3,3V** lorsque l'angle maximum parcouru par son axe est **300°**, modifier l'exemple ci-dessus pour qu'il affiche la tension (en V) et l'angle (en °) dans la console de l'IDE.



On considère que les différentes grandeurs sont proportionnelles entre elles.
Exemple d'affichage attendu : **Ve = 1.2V ⇒ N=23909 ⇒ Angle = 109,4°**.
Remarque : arrondir les résultats à 1 décimale avec la fonction *round(valeur,1)* et utiliser une f-string pour l'affichage.

- **Ressources** sur docs.circuitpython.org.
  - Module analogio
  - class busio.i2c
- **Etape 1** - Créer un nouveau programme

1. Cliquer sur l'icône **Nouveau** (Ctrl+N)
2. Copier-coller le code de l'exemple ci-dessous dans l'espace de travail de l'éditeur.

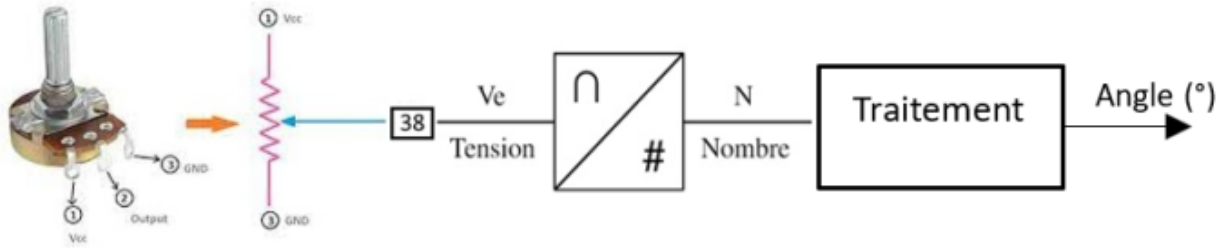
angle.py

```
# -------------------------------------------------------------------
----------
# Exemple de code pour afficher la valeur N délivrée par le
convertisseur
# analogique numérique ADC0.
# Matériels : Raspberry Pi Pico, Shield Grove, potentiomètre 10k
# Langage : CircuitPython
# IDE : Thonny
# Source : angle.py
# -------------------------------------------------------------------
----------
# Identification des entrées analogiques disponibles sur le Shield
Grove
# Shield    :         RP2040
#           : ADC       : GPIO
# ----------------------------------
#   A0      : ADC0      : GP26
#   A1      : ADC0,ADC1 : GP26, GP27
#   A2      : ADC1,ADC2 : GP27,GP28
# -------------------------------------------------------------------
----------
import board
from analogio import AnalogIn
import time

# Le potentiomètre 10k est connecté à l'entrée analogique A0 du shield.
pot = AnalogIn(board.A0)

while (True):
    # Lecture de la valeur délivrée par le convertisseur analogique
numérique
    # N est un entier positif codé sur 16 bits
    N = pot.value
    print(f"N = {N}")  # Affichage dans la console
    time.sleep(1)   # Attente 1s
```

- **Etape 2** - Enregistrer le programme
  - Enregistrer le programme avec le nom de l'exemple ci-dessus dans votre compte sur le serveur.
- **Etape 3** - Tester le programme
  - Exécutez le script en cliquant sur  .

MODIFICATION du PROGRAMME 2

Sachant que la tension Ve maximale à la sortie du potentiomètre est égale à **3,3V** lorsque l'angle maximum parcouru par son axe est **300°**, modifier l'exemple ci-dessus pour qu'il affiche la tension (en V) et l'angle (en °) dans la console de l'IDE.



On considère que les différentes grandeurs sont proportionnelles entre elles.
Exemple d'affichage attendu : **Ve = 1.2V ⇒ N=23909 ⇒ Angle = 109,4°**.
Remarque : arrondir les résultats à 1 décimale avec la fonction *round(valeur,1)* et utiliser une f-string pour l'affichage.

3.3 Prog. 3 - Afficheur graphique

Objectif

Afficher la température délivrée par le **capteur interne** au microcontrôleur sur un afficheur graphique.

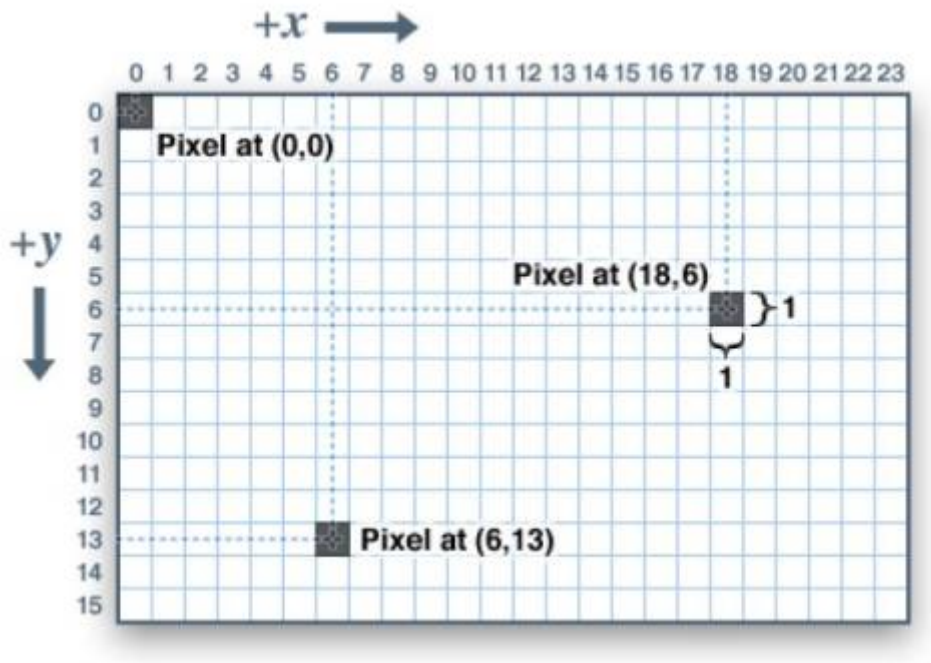
3.3.1 Matériels

- **Afficheur graphique** : 0,96" 128x64 OLED 2864 Display module - SSD1306 (I2C)

3.3.2 Généralités

Sur un écran, les **pixels** constituent une **image numérique**. On y accède par l'intermédiaire de leurs **coordonnées horizontales** (**X**) et **verticales** (**Y**). Le système de coordonnées place l'origine (0,0)

dans le coin supérieur gauche de l'écran, avec X positif croissant vers la droite et Y positif croissant vers le bas. L'axe Y est inversé par rapport au repère cartésien utilisé en mathématiques, mais c'est une pratique établie dans de nombreux systèmes graphiques informatiques. Si nécessaire, l'affichage peut être pivoté.



Les **coordonnées** sont **toujours exprimées en pixels**; il n'y a pas d'échelle implicite au monde réel exprimé en millimètres ou en pouces, la taille d'un dessin sera fonction de la densité, en pixels, de l'afficheur. Si vous visez une représentation du monde réel, vous aurez besoin de mettre vos coordonnées à l'échelle. Le pas des points peut être trouvé dans la fiche technique de l'écran, ou en mesurant sa largeur et en divisant le nombre de pixels par cette mesure.

Pour les affichages **monochromes** (unicolores), les “couleurs” sont toujours spécifiées comme étant simplement **1** (**afficher**) ou **0** (**effacer**). La sémantique **set/clear** est spécifique au type d'affichage : avec un affichage lumineux **OLED**, un pixel “set” est allumé, alors qu'avec un écran **LCD** réfléchissant, pour “set” le pixel est généralement sombre. Il peut y avoir des exceptions, mais généralement vous pouvez compter sur 0 (effacer) représentant l'état d'arrière-plan par défaut pour un affichage récemment initialisé.

3.3.3 Activités de programmation

- MicroPython
- CircuitPython

- **Ressources**
  - **Wiki** - MicroPython - Afficheurs à circuit SSD1306

- **Etape 1** - Créer un nouveau programme
  1. Cliquer sur l'icône **Nouveau** (Ctrl+N).
  2. Copier-coller le code de l'exemple ci-dessous dans l'espace de travail de l'éditeur.

afftemp.py

```
# -----------------------------------------------------------------------
----------
# Exemple de code pour afficher la température délivrée par le capteur
interne au
# microcontrôleur sur un afficheur graphique.
# Matériels : Raspberry Pi Pico, Shield Grove, afficheur à SSD1306
# Langage : MicroPython
# IDE : Thonny
# Source : afftemp.py
# Bibliothèque(s) : /lib/ssd1306.mpy
# -----------------------------------------------------------------------
----------
from machine import Pin, I2C
import ssd1306  # code de la bibliothèque à placer dans un dossier /lib
sur le RP2

# BUS I2C0
i2c = I2C(0, sda=Pin(8), scl=Pin(9), freq=400_000)

# Afficheur oled (ssd1306) connecté au bus I2C0
oled_width = 128  # px
oled_height = 64  # px
oled = ssd1306.SSD1306_I2C(oled_width,oled_height, i2c)

# 1er affichage
oled.text("Mesure temp.", 5, 0)  # un caractère occupe 8px de haut
oled.show()

while True:
    pass
```

- **Etape 2** - Enregistrer le programme
  - Enregistrer le programme dans votre compte sur le serveur en le nommant comme ci-dessus.
- **Etape 3** - Tester le programme
  - Exécutez le programme en cliquant sur  .

MODIFICATION du PROGRAMME 3

Le processeur du Raspberry Pi Pico intègre un capteur de température. Un exemple de code permettant de lire la température ambiante à l'aide du capteur de température interne au processeur est donné **ici**. Ouvrir cet exemple et modifier le programme précédent pour qu'il affiche la température ambiante dans la console et sur l'afficheur graphique.
Exemple d'affichage attendu : **T=22,9C**
Remarque : Il faut effacer l'écran graphique avec une méthode *fill(0)* avant d'écrire dessus.

- **Ressources**

- Adafruit : CircuitPython Wiring | Adafruit_CircuitPython_DisplayIO_SSD1306

- **Etape 1** - Créer un nouveau programme
  1. Cliquer sur l'icône **Nouveau** (Ctrl+N)
  2. Copier-coller le code de l'exemple ci-dessous dans l'espace de travail de l'éditeur.

afftemp.py

```
# -----------------------------------------------------------------------
----------
# Exemple de code pour afficher la température délivrée par le capteur
interne au
# microcontrôleur sur un afficheur graphique.
# Matériels : Raspberry Pi Pico, Shield Grove, afficheur à SSD1306
# Langage : CircuitPython
# IDE : Thonny
# Source : afftemp.py
# Bibliothèque(s) : /font5x8.bin, /lib/adafruit_ssd1306.mpy,
# /lib/adafruit_framebuf.mpy
# -----------------------------------------------------------------------
----------
import board
import busio  # code de la bibliothèque à placer dans un dossier /lib
sur le RP2
import adafruit_ssd1306
import time

# BUS I2C0
i2c = busio.I2C(board.GP9, board.GP8)

# Afficheur oled (ssd1306) connecté au bus I2C0
oled_width = 128  # px
oled_height = 64  # px
oled = adafruit_ssd1306.SSD1306_I2C(oled_width, oled_height, i2c)

# 1er Affichage
oled.text("Mesure temp.", 5, 0, 1)  # un caractère occupe 8px de haut
oled.show()

while True:
    pass
```

- **Etape 2** - Enregistrer le programme
  - Enregistrer le programme avec le nom de l'exemple ci-dessus dans votre compte sur le serveur.
- **Etape 3** - Tester le programme
  - Exécutez le script en cliquant sur  .

MODIFICATION du PROGRAMME 3

Le processeur du Raspberry Pi Pico intègre un capteur de température. CircuitPython rend très simple la lecture de la température grâce au module ***microcontroller***. Pour cela, il suffit d'écrire *microcontroller.cpu.temperature*. Modifier le programme précédent pour qu'il affiche la température ambiante dans la console et sur l'afficheur graphique.
Exemple d'affichage attendu : **T=22,9C**.
Remarque : Il faut effacer l'écran graphique avec une méthode *fill(0)* avant d'écrire dessus.

3.4 Prog. 4 - Capteur atmosphérique

Objectif

Afficher la température, l'humidité et la pression délivrées par un capteur à circuit **BME280** sur un afficheur graphique.

3.4.1 Matériels

- **Afficheur graphique** : Wiki - 0,96" 128x64 OLED 2864 Display module - SSD1306 (I2C)
- **Capteur atmosphérique** : Wiki - Capteurs - Environnement

3.4.2 Généralités

Le capteur utilisé est basé sur un circuit BME280 et mesure la température, l'humidité et la pression atmosphérique. Il communique avec un microcontrôleur via un bus I2C ou SPI.

- **Caractéristiques**
  - Alimentation: 3,3 à 5 Vcc
  - Plages de mesure:
    - température: -40°C à 85°C
    - humidité: 0 à 100% HR
    - pression: 300 à 1100 hPa
  - Précision:
    - température: ±1°C (±0,5°C pour le BME680)
    - humidité: ±3%
    - pression: ±1 hPa (0,12hPa pour le BME680)
  - Interfaces:

- I2C: sur connecteur Qwiic de Sparkfun ou Stemma QT d'Adafruit. **Adresse I2C: 0x77** (0x76 via cavalier à connecter entre SDO et GND)
- SPI: sur pastilles femelles au pas de 2,54 mm (connecteurs mâles à souder inclus)

3.4.3 Activités de programmation

- MicroPython
- CircuitPython

- **Ressources** sur Micropython.org.
  - Hardware I2C bus

- **Etape 1** - Créer un nouveau programme
  1. Cliquer sur l'icône **Nouveau** (Ctrl+N)
  2. Copier-coller le code de l'exemple ci-dessous dans l'espace de travail de l'éditeur.

affbme280.py

```
# -----------------------------------------------------------------------
----------
# Exemple de code pour afficher la température, l'humidité et la
pression
# mesurées par un capteur BME280 sur un afficheur graphique .
# Matériels : Raspberry Pi Pico, Shield Grove, capteur BM280, afficheur
à SSD1306
# Langage : MicroPython
# IDE : Thonny
# Source : affbme280.py,
# Bibliothèque(s) :  /lib/ssd1306.mpy, /lib/bme280.py
# -----------------------------------------------------------------------
----------
from machine import Pin, I2C
from time import sleep
import bme280  # code de la bibliothèque à placer dans un dossier /lib
sur le RP2

# BUS I2C1
i2c = I2C(1, scl=Pin(7), sda=Pin(6), freq=400_000)

# Capteur BME280 connecté à I2C1 du shield Grove
bme = bme280.BME280(i2c=i2c)

while True:
    temp = bme.temperature
    hum = bme.humidity
    pres = bme.pressure
    print('Température: ', temp)
    print('Humidité: ', hum)
    print('Pression: ', pres)
    print('')
```

```
    sleep(2)
```

- **Etape 2** - Enregistrer le programme
  - Enregistrer le programme dans votre compte sur le serveur en le nommant comme ci-dessus.
- **Etape 3** - Tester le programme
  - Exécutez le programme en cliquant sur  .

MODIFICATION du PROGRAMME 4

Modifiez le programme précédent pour qu'il affiche les 3 grandeurs physiques sur l'afficheur graphique.
Exemple d'affichage attendu :
BME280
- **T = 23.94C**
- **H = 59.59%**
- **P = 992.54hPa**

- **Ressources** sur doc.circuitpython.org.
  - class busio.i2c
- **Etape 1** - Créer un nouveau programme
  1. Cliquer sur l'icône **Nouveau** (Ctrl+N)
  2. Copier-coller le code de l'exemple ci-dessous dans l'espace de travail de l'éditeur.

affbme280.py

```
# -----------------------------------------------------------------------
----------
# Exemple de code pour afficher la température, l'humidité, la pression
et
# l'altitude mesurées par un capteur BME280.
# Matériels : Raspberry Pi Pico, Shield Grove, capteur BM280, afficheur
à ssd1306
# Langage : CircuitPython
# IDE : Thonny
# Fichiers : affbme280.py, /font5x8.bin, /lib/adafruit_ssd1306.mpy,
# /lib/adafruit_framebuf.mpy, /lib/adafruit_bme280.py
# -----------------------------------------------------------------------
----------
import board
import busio
from adafruit_bme280 import basic as adafruit_bme280
import time

# BUS I2C1
i2c = busio.I2C(board.GP7, board.GP6)  # SCL, SDA
```

```
# Capteur BME280 connecté à I2C1 du shield Grove
bme280 = adafruit_bme280.Adafruit_BME280_I2C(i2c)

# A modifier pour que cette valeur corresponde à la pression
# (hpa) au niveau de la mer (océan atlantique)
bme280.sea_level_pressure = 1016

while True:
    temp = bme280.temperature
    hum = bme280.humidity
    pres = bme280.pressure
    alt = bme280.altitude
    print('Température=', temp)
    print('Humidité: ', hum)
    print('Pression: ', pres)
    print('Altitude', alt)
    print('')
    time.sleep(2)
```

- **Etape 2** - Enregistrer le programme
  - Enregistrer le programme dans votre compte sur le serveur en le nommant comme ci-dessus.
- **Etape 3** - Tester le programme
  - Exécutez le programme en cliquant sur  .

MODIFICATION du PROGRAMME 4

Modifiez le programme précédent pour qu'il affiche les 3 grandeurs physiques sur l'afficheur graphique.
Exemple d'affichage attendu :
BME280
- **T = 23.22C**
- **H = 50.26%**
- **P = 999.70hPa**
- **Alt = 149.51m**

3.5 Prog. 5 - Capteur de luminosité

Objectif

Afficher la luminosité ambiante sur un afficheur graphique.

3.5.1 Matériels

- **Afficheur graphique** : Wiki - 0,96" 128x64 OLED 2864 Display module - SSD1306 (I2C)
- **Capteur de luminosité** : Wiki - Capteurs - Eclairement

3.5.2 Généralités

Le capteur utilisé est basé sur un circuit TSL2591 permettant de mesurer l'éclairement lumineux. Ce capteur communique avec un microcontrôleur via le bus I2C.

- **Caractéristiques**
  - Alimentation: 3,3 à 5 Vcc
  - Interface I2C:
    - Adresse I2C: 0x29 et 0x28 (l'adresse ne peut pas être changée)
  - Plage de mesure: 188 µLux à 88000 Lux

3.5.3 Activités de programmation

- MicroPython
- CircuitPython

- **Ressources** sur Micropython.org.
  - Hardware I2C bus

- **Etape 1** - Créer un nouveau programme
  1. Cliquer sur l'icône **Nouveau** (Ctrl+N)
  2. Copier-coller le code de l'exemple ci-dessous dans l'espace de travail de l'éditeur.

tst_tsl2591.py

```
# -----------------------------------------------------------------------
----------
```

```
# Affichage de la luminosité mesurée par un capteur TSL2591 dans la
console
# Date : 7/7/2023
# Matériels : Raspberry Pi Pico, TSL2591, Shield Grove, adaptateur
Qwiic,
# câble Qwiic 5cm
# Langage : MicroPython
# IDE : Thonny
# Fichiers : demoTSL2591.py, /lib/tsl2591.py
# Source : https://github.com/mchobby/esp8266-upy/tree/master/tsl2591
# ------------------------------------------------------------------
----------

from tsl2591 import * # Pilote du module
from machine import Pin, I2C # Pilote du bus I2C
from time import sleep # pour temporiser

# Bus I2C1
i2c = I2C(1, scl=Pin(7), sda=Pin(6), freq=400_000)

sleep(1) # Pause d'une seconde pour laisser à l'I2C le temps de
s'initialiser

tsl = TSL2591( i2c ) # Instanciation du capteur

while True:
    print( "Eclairement : %.1f lx" % tsl.lux ) # Lecture d'une valeur
    ir = tsl.infrared                          # Valeur entière
proportionelle à l'éclairement infrarouge
    vi = tsl.visible                           # Valeur entière
proportionelle à l'éclairement visible
    total = ir + vi                            # Somme des deux ...

    if total !=0:
        inv_total = 100 / total
        print("Infrarouge : %.1f %%" %(ir*inv_total))
        print("Lumière visible : %1.f %%" %(vi*inv_total))
    print("")
    sleep(5) # Temporisation de 5 secondes
```

- **Etape 2** - Enregistrer le programme
  - Enregistrer le programme dans votre compte sur le serveur en le nommant comme ci-dessus.
- **Etape 3** - Tester le programme
  - Exécutez le programme en cliquant sur  .

MODIFICATION du PROGRAMME 5

Modifiez le programme précédent pour qu'il affiche les 3 grandeurs physiques sur l'afficheur graphique.
Exemple d'affichage attendu :
TSL2561
- **Ecl. = 207.4lx**
- **Ir = 0%**
- **Lum. vis. = 100%**

- **Ressources** sur doc.circuitpython.org.
  - class busio.i2c

- **Etape 1** - Créer un nouveau programme
  1. Cliquer sur l'icône **Nouveau** (Ctrl+N)
  2. Copier-coller le code de l'exemple ci-dessous dans l'espace de travail de l'éditeur.

affbme280.py

```
```

- **Etape 2** - Enregistrer le programme
  - Enregistrer le programme dans votre compte sur le serveur en le nommant comme ci-dessus.
- **Etape 3** - Tester le programme
  - Exécutez le programme en cliquant sur  .

MODIFICATION du PROGRAMME 5

3.6 Synthèse

WEB : A faire

Pour aller plus loin

- **Tutoriels** sur RANDOM NERD TUTORIALS[2)]

1)
mnémonique pour Raspberry Pi Pico
2)
Random Nerd Tutorials helps makers, hobbyists and engineers build electronics projects. We make projects with: ESP32, ESP8266, Arduino, Raspberry Pi, ...