PREMIERE GENERALE

NUMERIQUE ET SCIENCES INFORMATIQUES

CARTE RASPBERRY PICO : COMMUNICATION I2C

1 – INTERFACES I2C DE LA CARTE RASPBERRY PICO

La carte Raspberry PICO dispose de 2 interfaces I2C, nommés `I2C0` et `I2C1` indépendantes l'une de l'autre. Plusieurs broches GPIO permettent d'accéder aux lignes SDA et SCL de ces 2 interfaces.

Interface	Ligne SDA	Ligne SCL
I2C0	GPIO0, GPIO4, GPIO8, GPIO12, GPIO16, GPIO20	GPIO1, GPIO5, GPIO9, GPIO13, GPIO17, GPIO21
I2C1	GPIO2, GPIO6, GPIO10, GPIO14, GPIO18, GPIO26	GPIO3, GPIO7, GPIO11, GPIO15, GPIO19, GPIO27

Les interfaces I2C sont gérées par la classe **`I2C`** de la librairie **`machine`**. Il faut créer un objet de type **`I2C`** sur lequel seront appliqués les différentes méthodes en spécifiant le numéro de l'interface, le numéro des broches **`scl`** et **`sda`** et éventuellement la fréquence des signaux I2C.

```
from machine import I2C
nom_I2C = I2C(numero_Interface, scl=Pin(num_broche), sda=Pin(num_broche), freq = frequence)
```

Les principales méthodes de cette librairie sont :

Méthode « `start` »

Cette méthode génère une condition de START sur le bus. Le signal SDA passe au niveau bas pendant que SCL est au niveau haut.

```
nom_I2C.start()
```

Méthode « `stop` »

Cette méthode génère une condition de STOP sur le bus. Le signal SDA passe au niveau haut pendant que SCL est au niveau haut.

```
nom_I2C.stop()
```

Méthode « `scan` »

Scanne toutes les adresses des composants connectés sur l'interface I2C et retourne une liste Python contenant ces adresses.

```
nom_I2C.scan()
```

Méthode « `writeto` »

Cette méthode envoi les octets disponibles dans `data` vers l'esclave dont l'adresse `adr` est mentionnée en paramètre. Si le paramètre `stop` est placé à `True`, la condition d'arrêt (STOP) est générée à la fin du transfert.

```
nom_I2C.writeto(adr, data, stop = True)
```

Méthode « readfrom »

Cette méthode permet la lecture des nboctets octets transmis par l'esclave identifié par son adresses addr. Si le paramètre stop est à True alors la condition de STOP est générée à la fin du transfert

```
nom_I2C.readfrom(adr, nboctet, stop = True)
```

Certains périphériques I2C fonctionnent comme des mémoires (ou ensemble de registres) qu'il est possible de lire ou dans lesquels il est possible d'écrire. Dans ce cas, il y a deux adresses associées avec le périphérique I2C : une adresse I2C et une adresse mémoire. Un certain nombre de méthodes sont conçues pour communiquer avec ce type de périphériques. Parmi lesquelles on trouve :

Méthode « readfrom_mem »

Cette méthode permet la lecture des **nboctets** octets dans la mémoire **memadr** du périphérique identifié par son adresse **addr**.

```
nom_I2C.readfrom_mem(adr, memadr, nboctet)
```

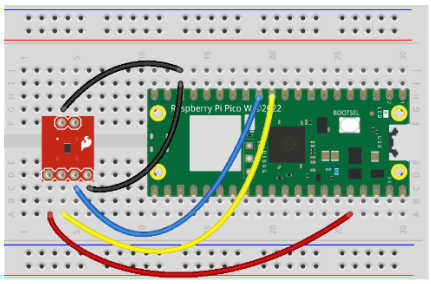
2 – AFFICHAGE DE LA TEMPERATURE MESUREE PAR LE CAPTEUR TMP102

Le programme ci-dessous permet de lire la température mesurée par le capteur de température tmp102 et de l'afficher dans la console.

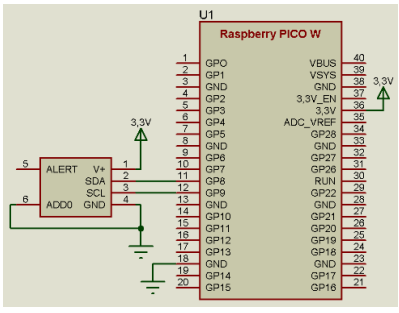
```
from machine import I2C, Pin
import time

tmp102_I2C = I2C(0, scl = Pin(9), sda = Pin(8), freq = 400000) # Création d'un objet I2C
tmp102_adresse = 0x48;
ptr_registre = 0;

while True:
    data = tmp102_I2C.readfrom_mem(tmp102_adresse, ptr_registre, 2) # Lecture des données
    octets_tmp = list(data)          # Conversion des données en tableau
    temp = ((octets_tmp[0] * 256 + octets_tmp[1]) >> 4) * 0.0625;    # Calcul température
    print("Température : ", temp, " °C")
    time.sleep(2)
```



- Schéma de cablâge -



- Schéma struturel -

Après avoir exécuter le programme, la température mesurée par le capteur TMP102 est affichée toutes les 2 secondes dans la console.

```
Console
>>> %Run -c $EDITOR_CONTENT

MPY: soft reboot
Température :  27.75  °C
Température :  27.6875  °C
Température :  27.625  °C
Température :  27.5  °C
Température :  27.4375  °C
```

- Résultat d'exécution du programme -