PREMIERE GENERALE

NUMERIQUE ET SCIENCES INFORMATIQUES

CARTE RASPBERRY PICO : COMMUNICATION UART

1 – INTERFACES UART DE LA CARTE RASPBERRY PICO

La carte Raspberry PICO intègre 2 ports UART : **`UART0`**, **`UART1`** indépendantes l'une de l'autre. Plusieurs broches GPIO permettent d'accéder aux lignes des interfaces SPI.

Interface	TX	RX
UART0	GPIO0, GPIO12, GPIO16	GPIO1, GPIO13, GPIO17
UART1	GPIO4, GPIO8	GPIO5, GPIO9

Les interfaces UART sont gérées par la classe **`UART`** de la librairie **`machine`**. Il faut créer un objet de type **`UART`**, sur lequel seront appliqués les différentes méthodes, en spécifiant le numéro de l'interface, la vitesse de communication (paramètre **`baudrate`**) et le numéro des broches **`Rx`** et **`Tx`**.

```
from machine import UART
nom_UART = UART(numero_Interface, baudrate = vitesse, tx = Pin(num_pin), rx = Pin(num_pin))
```

Les principales méthodes de cette librairie sont :

Méthode « `any` »

Cette méthode renvoie le nombre d'octets reçus sur la ligne Rx.

```
donnees = nom_UART.any()
```

Méthode « `read` »

Cette méthode lit les octets reçus sur la ligne Rx.

```
donnees = nom_UART.read()
```

Méthode « `readline` »

Cette méthode lit une ligne de caractères reçus sur la ligne Rx. Un caractère de saut de ligne termine la ligne

```
donnees = nom_UART.readline()
```

Méthode « `Write` »

Cette méthode écrit la chaine de caractères stockées dans **donnees** sur la ligne Tx.

```
nom_UART.write(donnees)
```

2 – COMMUNICATION PAR MODULE BLUETOOTH

Dans cet exemple, la carte Raspberry PICO va permettre de transmettre des par Bluetooth via le module HC-05. La communication entre la carte Raspberry PICO et le module HC-05 sera réalisée via une interface UART.

- Module HC-05 -

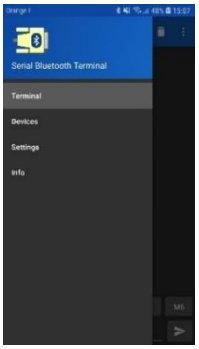
Le programme ci-dessous permet d'envoyer par Bluetooth une valeur qui s'incrémente toutes les secondes.

```
from machine import Pin, UART
import time

bluetooth = UART(1, baudrate = 9600, tx = Pin(4), rx = Pin(5))

compteur = 0

while True:
    print("Valeur du compteur: ", compteur);
    compteur = compteur + 1
    bluetooth.write("Valeur du compteur: " + str(compteur) + '\n')
    time.sleep(1)
```

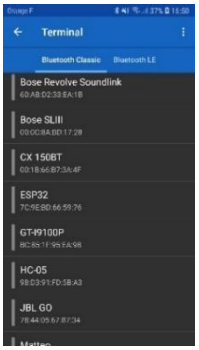
Il existe de nombreuses applications Android permettant la communication Bluetooth. Nous allons utiliser **Serial Bluetooth Terminal** .

Compiler et téléverser le programme précédent.

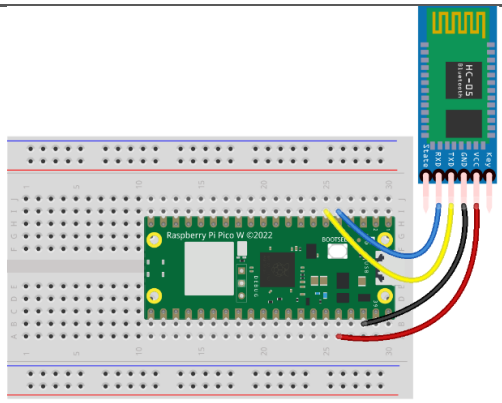
Après avoir appairé le téléphone portable avec la carte ESP32, il faut lancer l'application **Serial Bluetooth Terminal**.



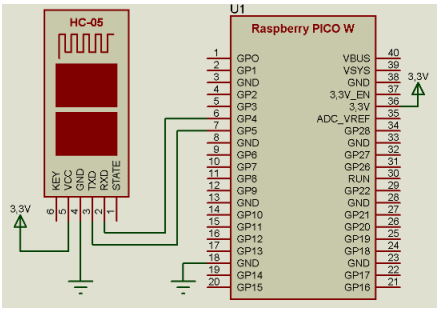
Dans le menu général choisir l'onglet « **Devices** ».



Dans la liste choisir **HC-05**.menu général choisir l'onglet « **Devices** ».



- Schéma de cablâge -



- Schéma structurel -

Les valeurs transmises par la carte Raspberry PICO via le module HC-05 s'affichent alors sur le terminal de l'application Serial Bluetooth Terminal et sur la console.

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

MPY: soft reboot
Valeur du compteur:  0
Valeur du compteur:  1
Valeur du compteur:  2
Valeur du compteur:  3
Valeur du compteur:  4
Valeur du compteur:  5
Valeur du compteur:  6
Valeur du compteur:  7
Valeur du compteur:  8
```

- Résultats après exécution du programme -

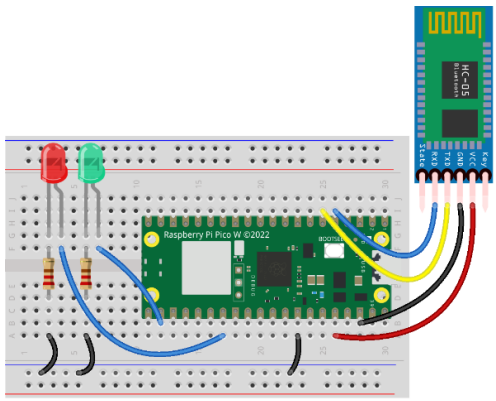
3 – RECEPTION D'UNE DONNEE PAR BLUETOOTH

Le programme ci-après permet de contrôler deux LED à partir d'une commande envoyée par un smart phone :

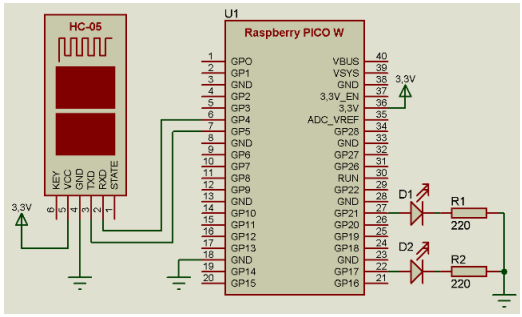
- La commande **'a'** allume uniquement la première LED
- La commande **'b'** allume uniquement la deuxième LED
- La commande **'c'** allume les deux LEDs
- La commande **'d'** éteint les deux LEDs
- La commande **'e'** change l'état des deux LEDs

```
from machine import Pin, UART
import time

bluetooth = UART(1, baudrate = 9600, tx = Pin(4), rx = Pin(5))

bluetooth_msg = ""
etatLED1 = 0
etatLED2 = 0
pin_led1 = Pin(17, mode=Pin.OUT)
pin_led2 = Pin(21, mode=Pin.OUT)

while True:
    if bluetooth.any() > 0:
        bluetooth_msg = str(bluetooth.read().strip().decode()):
        if bluetooth_msg == 'a':
            pin_led1.value(1)
            pin_led2.value(0)
            etatLED1 = 1
```

```
            etatLED2 = 0
            bluetooth.write("LED 1 allumée, LED 2 éteinte \n")
        elif bluetooth_msg == 'b':
            pin_led1.value(0)
            pin_led2.value(1)
            etatLED1 = 0
            etatLED2 = 1
            bluetooth.write("LED 1 éteinte, LED 2 allumée \n")
        elif bluetooth_msg == 'c':
            pin_led1.value(1)
            pin_led2.value(1)
            etatLED1 = 1
            etatLED2 = 1
            bluetooth.write("Les 2 LEDs allumées \n")
        elif bluetooth_msg == 'd':
            pin_led1.value(0)
            pin_led2.value(0)
            etatLED1 = 0
            etatLED2 = 0
            bluetooth.write("Les 2 LEDs éteintes \n")
        elif bluetooth_msg == 'e':
            etatLED1 = not etatLED1
            etatLED2 = not etatLED2
            pin_led1.value(etatLED1)
            pin_led2.value(etatLED2)
            bluetooth.write("Les LEDs ont changé d'état \n")
        bluetooth_msg = ""
        time.sleep(1)
```

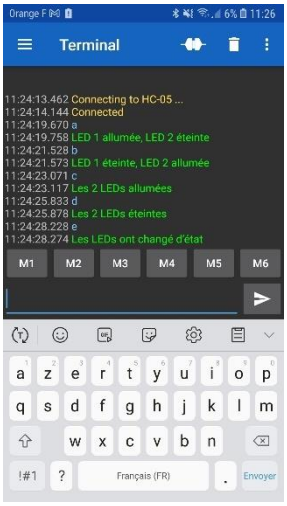


- Schéma de cablâge -



- Schéma struturel -

Après avoir téléversé le programme, il devient possible de commander à distance par Bluetooth l'allumage des 2 LEDs à l'aide de l'application Serial Bluetooth Terminal par exemple.



- Résultats après exécution du programme -