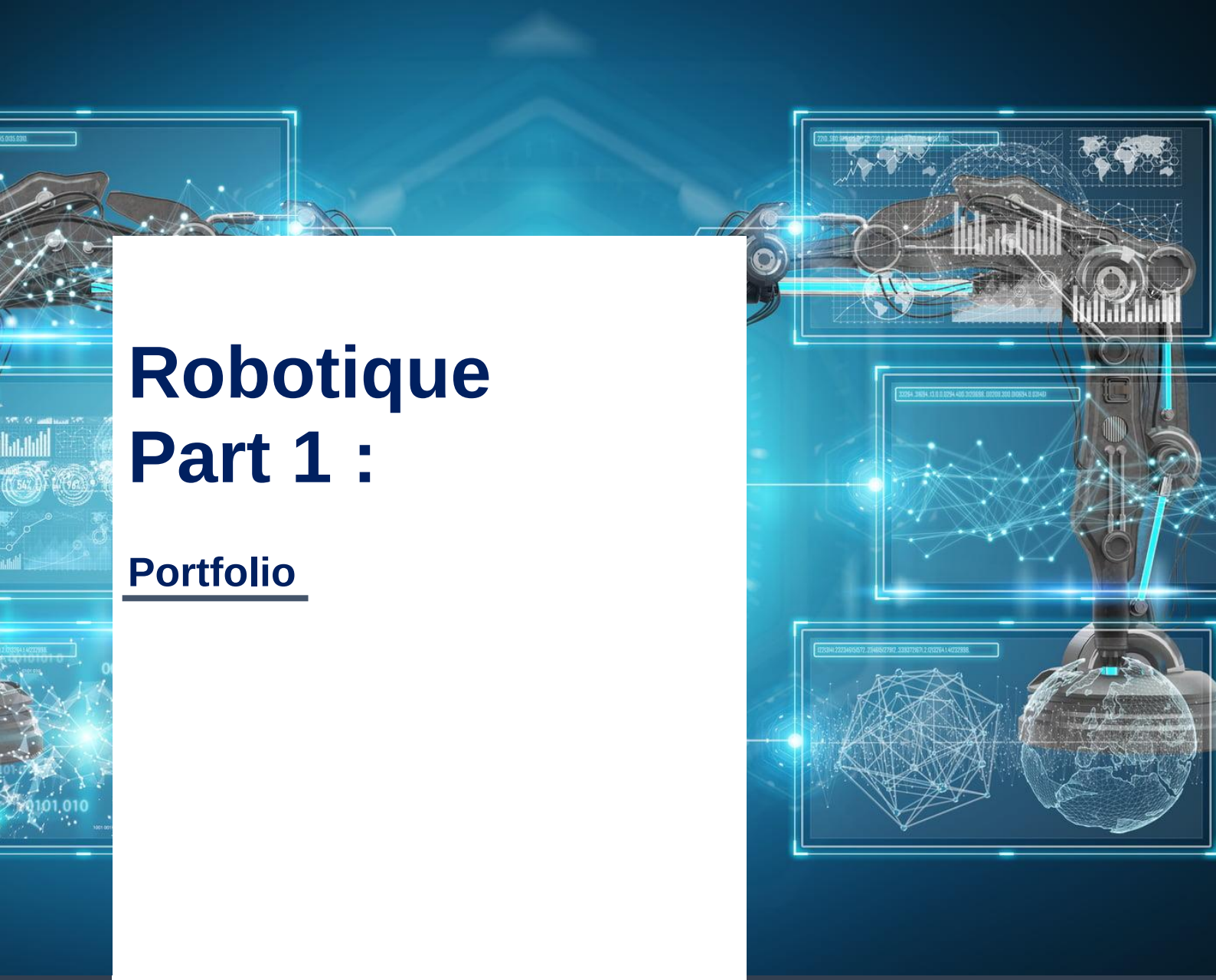


Robotique Part 1 :

Portfolio

EPHEC

Bachelier en automatisation

2AU 2022-2023

Professeur : Emile Costa

A2142

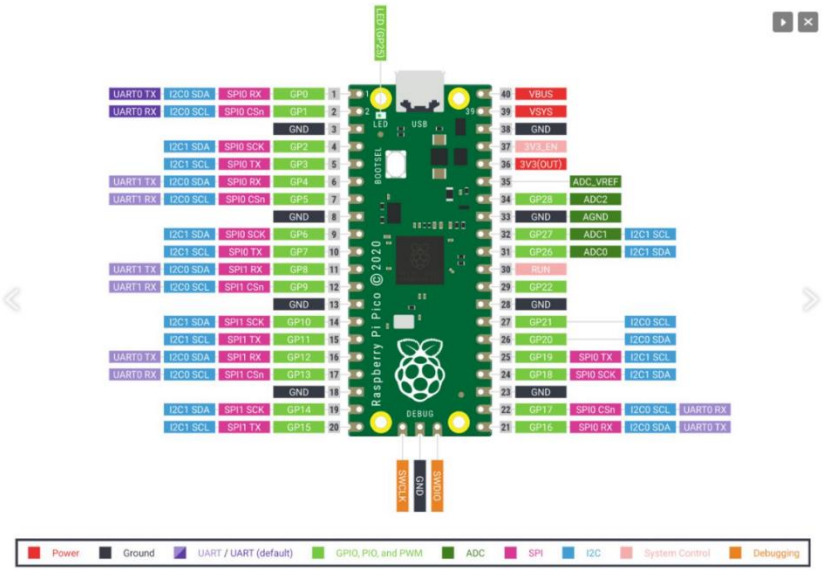
Crée par :
Gonzalez Nemo – Ribesse Vincent

Table des matières

Introduction : Programmation mycropython sur raspberry Pi Pico :

Le monde de la robotique est un monde très vaste. En constante évolution, il fait maintenant intégralement partie de nos vies. Dans ce portfolio nous reprenons les bases de la robotique et programmation micro-python, par la mise en fonctionnement de différents composants de base avec un microcontrôleur, le Raspberry PI PICO.

Le Raspberry PI PICO est un microcontrôleur, contrairement à ses collègues Raspberry PI qui sont considéré comme des microordinateurs. Il est intéressant car très performant et très polyvalent. Idéal pour l'application en temps réel, il possède des entrées analogique ADC, il est PIO, il est relativement compact et a une faible consommation d'énergie. Il ne possède néanmoins pas les fonctionnalités Bluetooth et wifi. Mais une version w de ce dernier existe et possède ces deux fonctionnalités.



Raspberry Pi Pico *Item 2 of 7*

C'est l'élément principal de notre travail effectué en robotique, qui nous permettra de tester et mettre en fonctionnement les différents composants en robotique de base. Ce portfolio est organisé en 5 parties qui reprennent les 5 TP que nous avons pu faire en cours.

LED RGB :

Introduction :

Dans ce premier TP de robotique, nous étudions la led RGB en la connectant à un Raspberry PI Pico.

Une led RGb est une led qui peut s'allumer dans toutes les couleurs grâce à 3 leds internes, qui correspondent aux 3 couleurs primaires. Celles-ci sont le bleu le vert et le rouge. La led RGB possède en tout 4 broches. Une pour chaque led interne et une commune à l'ensemble des leds.

Schémas de câblage :

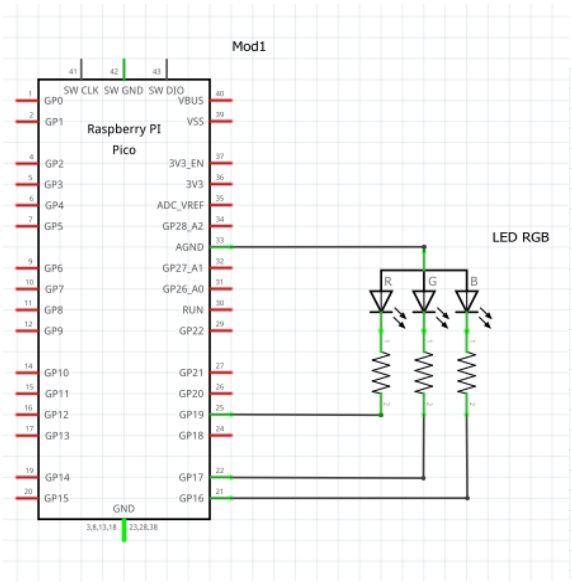


Figure 1

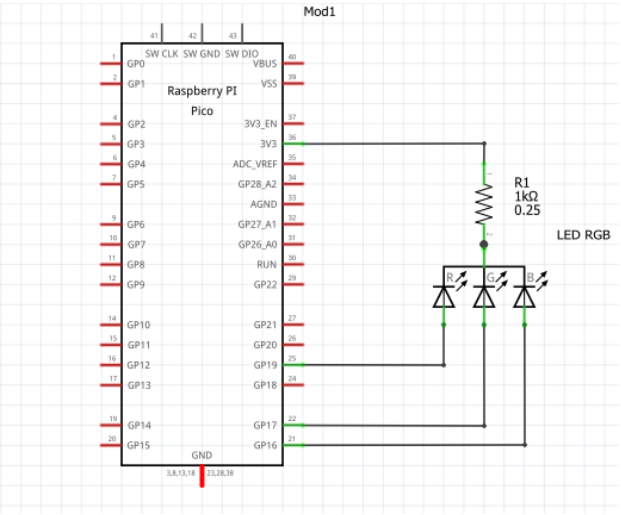
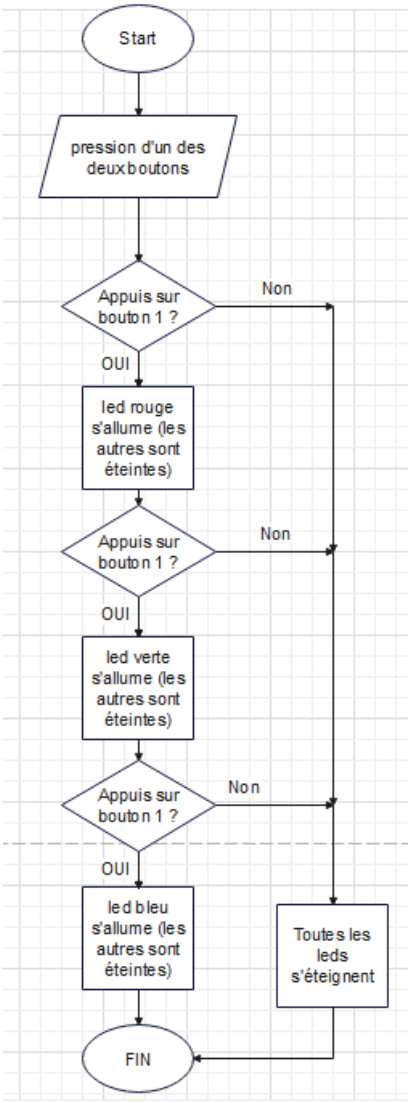


Figure 2

Nous avons constaté qu'il existait deux type de led RGB. À anode commune (Figure 1) et à cathode commune (Figure 2). Nous avons donc la possibilité de les brancher de deux manières différentes. Soit, nous branchons l'anode commune à la masse et relions les 3 leds aux pns, par l'intermédiaire de résistances (220 Ohm pour la rouge et 100 Ohm pour les deux autres). Nous sommes alors en régulation direct.

Soit, nous mettons une résistance de 1 K Ohm (Par sécurité plutôt que des valeurs plus basse de résistance) au niveau des cathodes communes, que nous relierons à l'alimentation 5V. Les anodes seront alors reliées aux Pins. (Réglage PWM).

Logigramme :



Code :

Les 3 leds sont commandées par 2 boutons. Le premier bouton permet de commencer le cycle (la première led s'allume). Chaque nouvelle pression sur celui-ci éteint la led de l'étape précédente et permet de passer à l'étape d'après. La gestion du nombre de pression sur le bouton 1 est effectuée par une variable d'incrémentation. Le deuxième bouton permet, lui, de reset le cycle et l'incrémentation qui va avec.

Le tout est organisé en différentes fonctions qu'on appelle dans la boucle principale.

On a donc le cycle suivant.

Pression 1 BP1 = led rouge

Pression 2 BP1 = led verte

Pression 3 BP1 = led bleu

Pression 4 BP1 = Pression 1 BP1

Pression BP2 = Clear

```
from machine import Pin
import time

R = Pin(19, Pin.OUT)
B = Pin(16, Pin.OUT)
G = Pin(17, Pin.OUT)

Bt1 = Pin(16, Pin.IN)
Bt2 = Pin(17, Pin.IN)

increment = 0 #Incrémentation qui varie de 0 à 2. Cela permettra d'obtenir 3 états (allumage de chaque led)."


def val_Bt(): #"Fonction qui vient vérfier la valeur des boutons."

    val_Bt1 = Bt1.value()
    val_Bt2 = Bt2.value()

    if val_Bt1:
        time.sleep(500)  #"Modification du temps d'arrêt du bouton pour pas que ça aille trop vite."

    return val_Bt1, val_Bt2

def led_on(increment):  #"Fonction qui commande l'allumage des leds "
    if increment == 0:
        R.value(1)      #"led rouge on"
        G.value(0)
        B.value(0)

    if increment == 1:
        R.value(0)      #"led verte on"
        G.value(1)
        B.value(0)

    if increment == 2:
        R.value(0)      #"led Bleue on"
        G.value(0)
        B.value(1)




    increment += 1
    if increment > 2:    #"Reset de l'incrementation"
        increment = 0

    return increment

def led_off():     #"fonction qui s'occupe d'éteindre les leds"
    R.value(0)
    G.value(0)
    B.value(0)
    return

while True:            # "Boucle principale du programme"
    time.sleep_ms(50)

    Btv1 = val_Bt()[0]    #"on vient chercher les valeur des deux boutons"
    Btv2 = val_Bt()[1]
    if Btv1 == True:     # "appel à la fonction d'allumage"
        led_on(increment)

    if Btv2 == True:
        led_off()        #"appel à la fonction de mise en off des leds"
        increment = 0
```

Conclusion :

La led RGB est un composant qui peut être construit de deux manières différentes. Comme vu ci-dessous les 3 leds internes peuvent soit être montées en anode commune soit en cathode commune. Nous avons vu que la manière de les câbler n'est pas la même en fonction du type de led RGB.

Le Joystick :

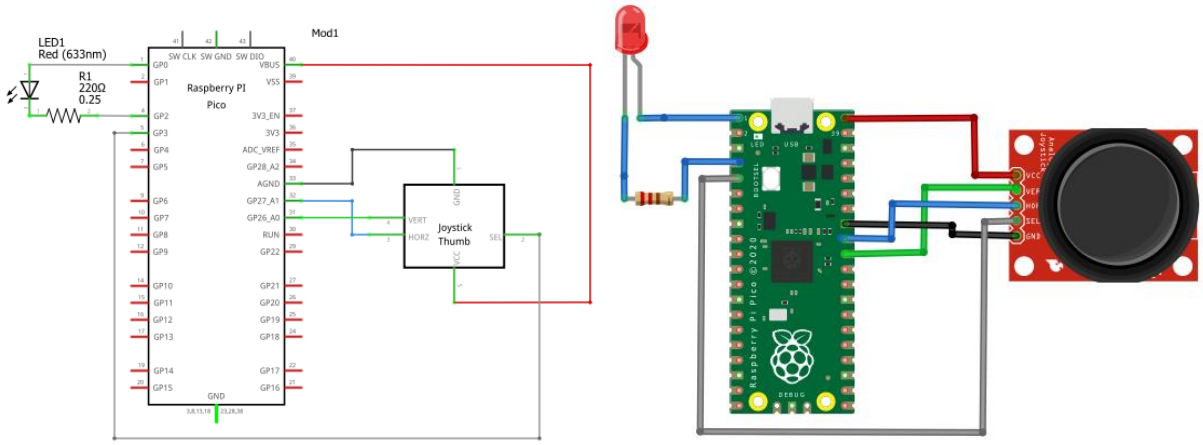
Introduction :

Le joystick est un composant qui est principalement utilisé pour « diriger » des choses. Il est très présent sur les manettes de jeux vidéo ou encore sur les télécommandes des voitures ou autres appareils télécommandés.

Le joystick est en fait un capteur composé de deux potensiomètres. Ces derniers captent les postions X et Y, ce qui donne au joystick une position bidirectionnelle entre 4 extrémités, par ailleurs il possède un bouton-Poussoir pour détecter les clics d'un utilisateur extérieur. On a donc pas 1 ni 2 mais 5 actions différentes qui peuvent être effectuées par le même composant.

Tous ces éléments en font donc un thème indispensable à analyser avant la réalisation d'un potentiel projet en robotique.

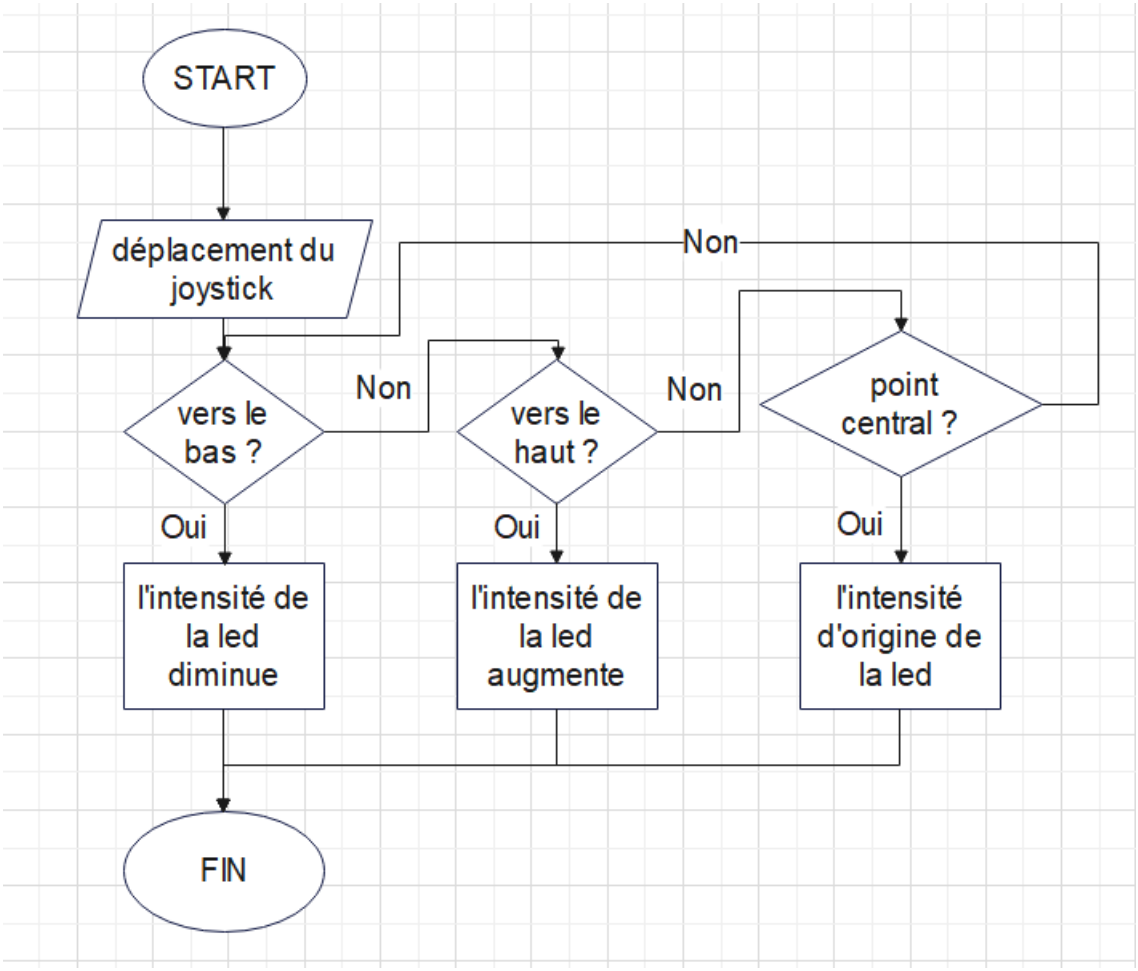
Schéma de Cablage :



Le joystick possède 5 pins. La 1ère est la pin d'alimentation Vcc en 5V, la 2ème renvoie la valeur du potentiommètre en X et la 3ème renvoie la valeur du potentiomètre en Y. Comme il s'agit de deux valeurs analogiques, il faut les connecter à des entrées ADC du Raspberry Pi Pico. La 4ème pin est celle du bouton poussoir du joystick qui est booléen et peut donc être branché dans une simple entrée GP. Et pour finir la 5ème est celle de la masse.

Nous testons içi le joystick en faisant varier l'intensité d'une led.

Logigramme :



Code :

```
from machine import ADC, Pin, PWM
import time

#Variables

adc = ADC(Pin(26))
adc1 = ADC(Pin(27))

pwm0 = PWM(Pin(0))
pwm0.freq(1000)
SW= Pin(2, Pin.IN, Pin.PULL_UP)

#Boucle principale du code

while True:

    print(SW.value())
    val_x = adc.read_u16() #Lecture valeur x
    val_y = adc1.read_u16() #lecture valeur y
    print(val_x)
    print(val_y)
    time.sleep_ms(1000)
```

Conclusion :

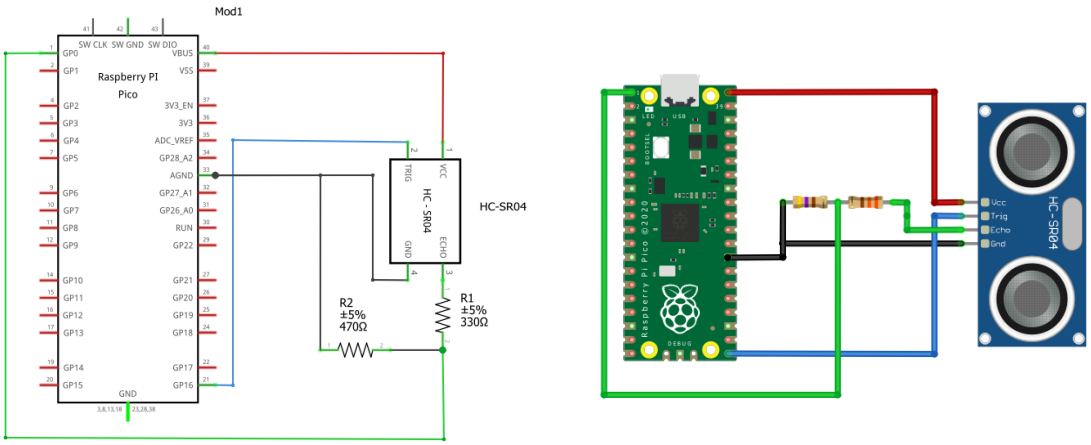
On peut conclure que le joystick est le bouton qui contient le plus de fonctionnalités, 5 niveaux d'actions. Ses deux potentiomètres renvoient des valeurs de résistances qui sont analogiques. On ne peut donc pas utiliser toutes les entrées du Raspberry Pi Pico. Comme nous avons de l'analogique, les entrées à utiliser sont des entrées type ADC.

Capteur à ultrason :

Introduction :

Le capteur à ultrason est un capteur qui permet de définir la distance d'un objet. Comme son nom l'indique il fonctionne avec des ultrasons. Cette fonctionnalité est très intéressante et peut être utilisée dans divers projets de robotique. Son étude nous sera donc utile pour la deuxième partie de ce cours de robotique.

Schéma de câblage :

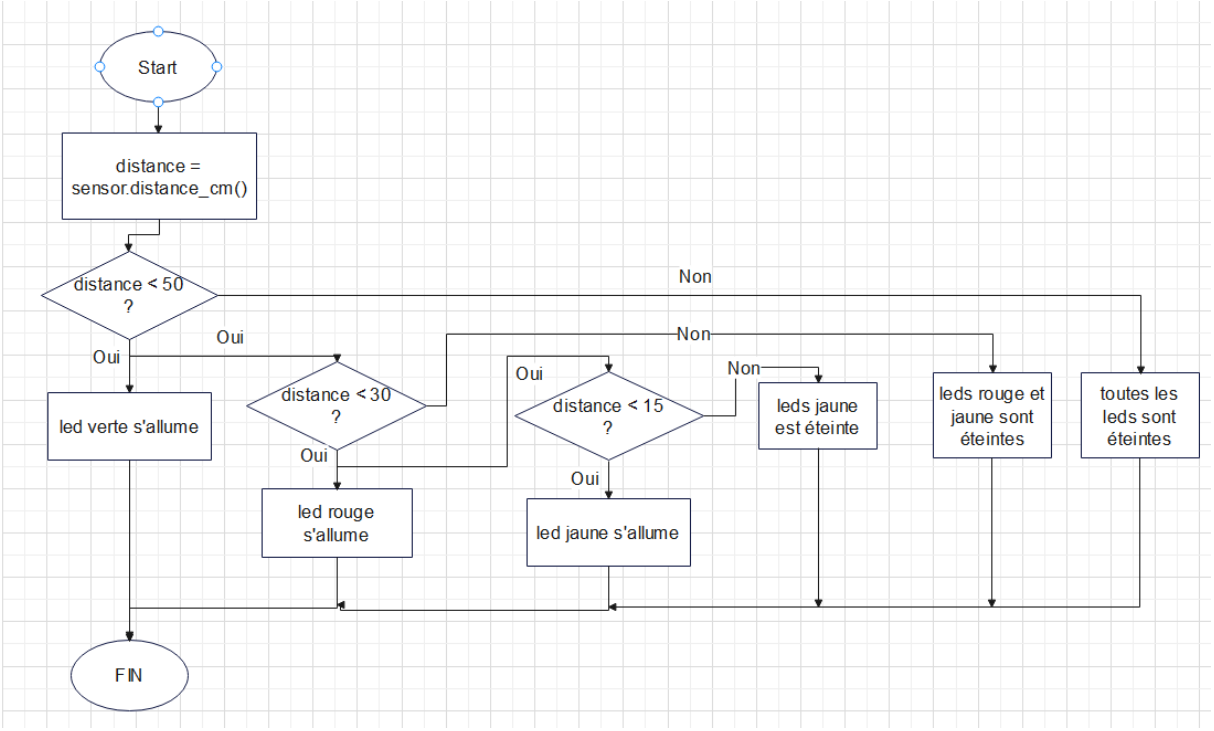


Le capteur à ultrason est en fait un transducteur qui contient un émetteur (pin Trig) et un récepteur (pin Echo) d'ondes sonores à haute fréquence. L'émetteur envoie des ondes qui vont, au contact d'un obstacle quelconque, rebondir et être captées en retour par le récepteur. Si on mesure le temps de réception des ondes, on peut alors grâce à la vitesse de propagation du son (340m/s), calculer la distance de l'obstacle en question.

Le fonctionnement du capteur HC SR04 :

Lorsque le composant reçoit une impulsion de 5 V pendant 10 micro secondes sur l'entrée trigger, il émettra des ultrasons à environ 40 kHz. Lorsque le récepteur reçoit ces ultrasons, la sortie Echo envoie une impulsion de 5V dont la durée est proportionnelle à la distance.

Logigramme :


Start
distance = sensor.distance_cm()
distance < 50 ?
Non
Oui
Oui
led verte s'allume
distance < 30 ?
Oui
Non
Oui
led rouge s'allume
distance < 15 ?
Non
Oui
led jaune s'allume
leds jaune est éteinte
leds rouge et jaune sont éteintes
toutes les leds sont éteintes
FIN

Code :

Pour tester l'ultrason nous aurions pu utiliser un oled qui permet d'afficher une distance, mais comme nous n'en avions pas, nous avons décidé de représenter cette distance captée en utilisant des leds qui s'allument suivant la proximité de l'obstacle détecté.

Pour l'utiliser correctement nous avons dû importer le module « HCSR04 » que nous avons été chercher sur GitHub et qui nous permet d'utiliser la fonction « hcsr04 ».

```
from hcsr04 import HCSR04
from machine import Pin
import time
import utime

sensor = HCSR04(trigger_pin=16, echo_pin=0) # On utilise içi une fonction dans le module hcsr04.
G = Pin(28, Pin.OUT)
R = Pin(27, Pin.OUT)
J = Pin(22, Pin.OUT)

while True:  # Boucle principale du code

    distance = sensor.distance_cm()
    print('Distance:', distance, 'cm')
    time.sleep_ms(50)

    # déifinition des conditions d'activation des leds.

    if distance < 50:
        G.value(1)
    else :
        G.value(0)

    if distance < 30:
        R.value(1)
    else :
        R.value(0)

    if distance < 15:
        J.value(1)
    else :
        J.value(0)
```

Conclusion :

Nous pouvons capter une distance. Les leds s'allument bien de manière successive au fur et à mesure que l'on se rapproche du capteur. La précision varie en fonction du rebondissement des ondes sur l'objet qu'on capte. Si la surface est plate et est composé d'une matière sur laquelle les ondes rebondissent bien, on aura une distances fiable. Mais si on a une surface oblique, tout l'echo ne sera peut-être pas récupéré par le capteur. Si la surface est composée d'une matière absorbante comme du tissu ou de la mousse, les ondes ne rebondiront peut-être pas systématiquement.

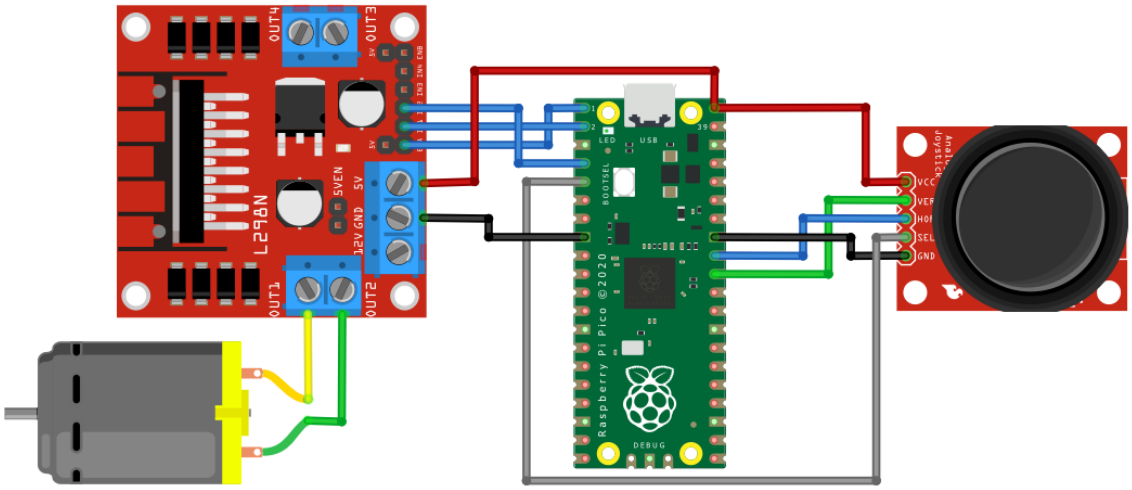
Ce sont des paramètres à prendre en compte si l'on veut intégrer ce capteur dans un projet.

Moteur DC avec carte de puissance ENA L298N et Joystick :

Introduction :

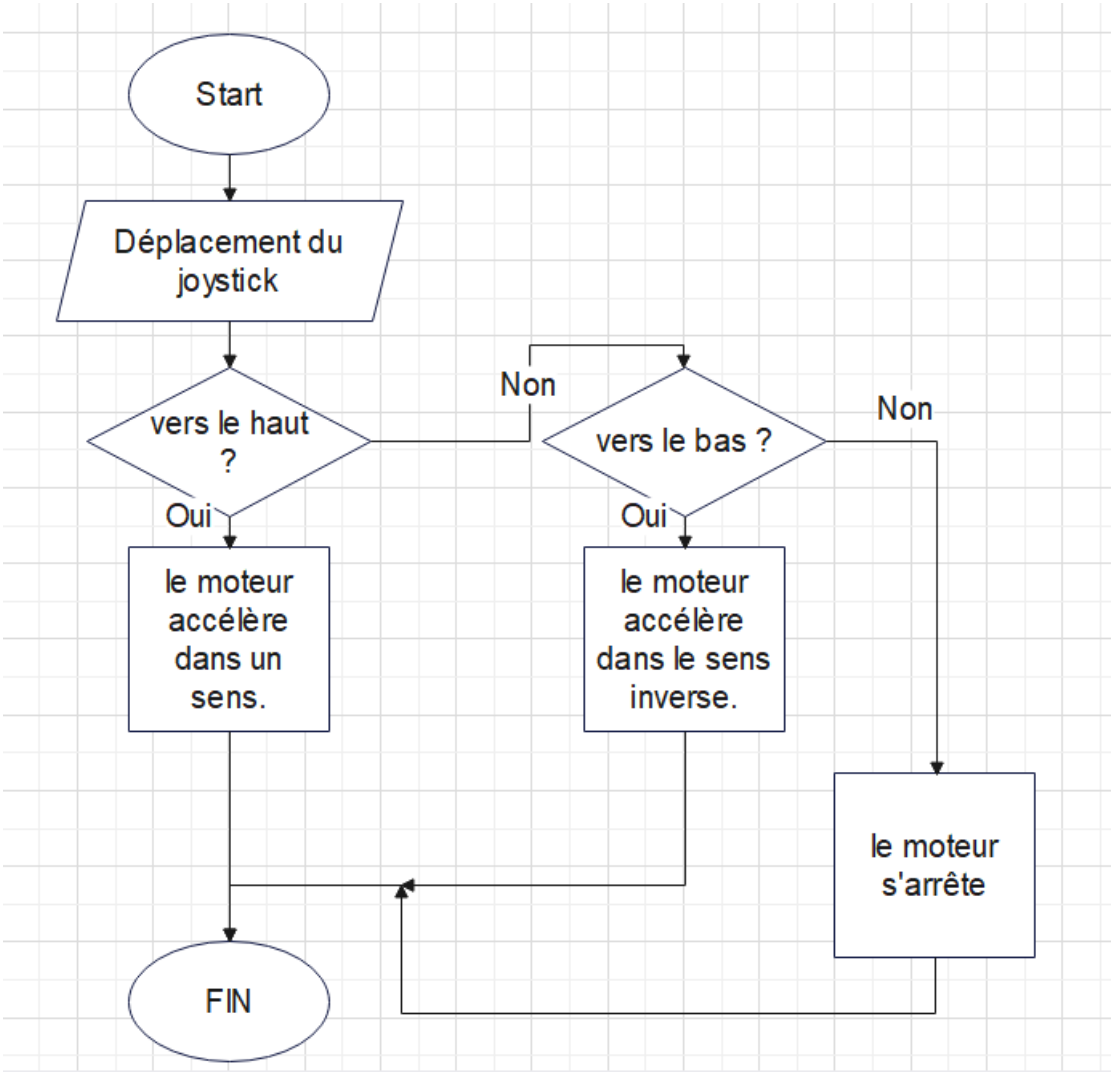
L'un des principaux facteurs dans la robotique est le mouvement/l'actionnement. Celui-ci est généralement engendré par des moteurs. Il est donc important d'étudier le fonctionnement d'un de ces actionneurs qui est le moteur. Mais pas tout seul. Nous allons ici moduler la vitesse d'un moteur DC à l'aide d'un joystick et d'une carte de puissance. Le thème sera la modulation par impulsions (PWM).

Schéma de câblage :



La carte de puissance qu'on utilise peut commander 2 moteurs, mais nous en comandons juste un pendant le tp. Notre moteur est branché aux sorties 1 et 2 de la carte qui est elle-même reliée au raspberry via les entrées ENA IN1 et IN2. L'entrée ENA est celle qui va permettre la modulation d'impulsion et les entrées IN1 et IN2 permettent de contrôler le moteur. On récupère les valeurs des potentiommètres du joystick sur les pin 26 et 27 ADC du raspberry. La carte et le joystick sont alimentés en 5 V sur la Pin Vbus.

Logigramme :



Code :

On va lire les valeurs des potentiomètres du joystick, qu'on va moduler avec ENA.duty_16(). En fonction des états des entrées IN1 et IN2 on peut faire fonctionner le moteur dans les deux sens de marche.

Une fonction scale_value nous permet de convertir la valeur des résistances du joystick en un pourcentage. On peut comme ça avoir une idée du pourcentage de vitesse de notre moteur.

```
from machine import Pin, UART, PWM, ADC
import time

ENA = PWM(Pin(0)) # La modulation par impulsions (PWM) permet de moduler la vitesse en fonction de la valeur val_y.
ENA.freq(2000)
IN1 = Pin(1, Pin.OUT)
IN2 = Pin(2, Pin.OUT)

Vrx = ADC(Pin(26)) # Valeur analogique -> ADC
Vry = ADC(Pin(27))

# Fonction qui convertit la valeur de la résistance de notre joystick en pourcentage. (Elle n'est utilisée que pour voir la différence de vitesse en pourcentage).

def scale_value (value,in_min,in_max,out_min,out_max):
    scaled_value = (value - in_min) * (out_max - out_min) / (in_max - in_min) + out_min
    return scaled_value

while True:

    # Lecture et mise en variables des valeurs x et y du joystick.

    val_y = Vrx.read_u16()
    val_x = Vry.read_u16()

    print(val_y)
    time.sleep_ms(200)

    if val_y <= 45000:
        ENA.duty_u16((-val_y)+65000) # Permet de moduler la vitesse du moteur (Forward).

        IN1.value(1)
        IN2.value(0)

        print(int(scale_value(y_value,0,45000,0,100))) # La vitesse est affichée en pourcentage à l'aide de "scale_value" (Forward).

    elif val_y >= 55000:
        ENA.duty_u16(val_y) # Permet de moduler la vitesse du moteur (marche arrière).
        IN1.value(0)
        IN2.value(1)

        print(int(scale_value(y_value,50000,65535,0,100))) # La vitesse est affichée en pourcentage en utilisant "scale_value" (Backward).

    else:
        IN1.value(0)
        IN2.value(0)
```

Conclusion :

La carte de puissance, associée au joystick, nous a permis de moduler la vitesse du moteur dans les deux sens de fonctionnement. Le résultat est concluant.

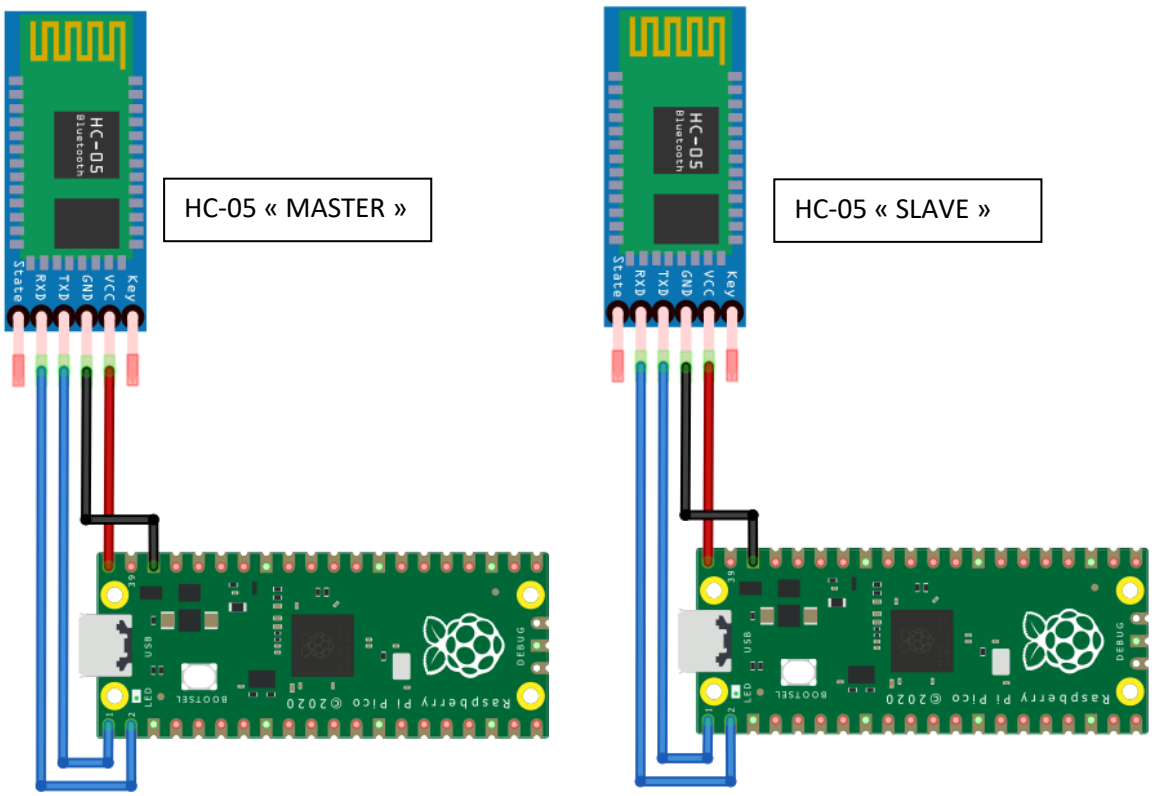
Bluetooth avec HC 04 :

Introduction :

Le but de ce tp est de réaliser un montage électronique nous permettant de créer une communication Bluetooth. Pour cela on utilise deux modules HC-05 associés à deux Raspberry pi pico.

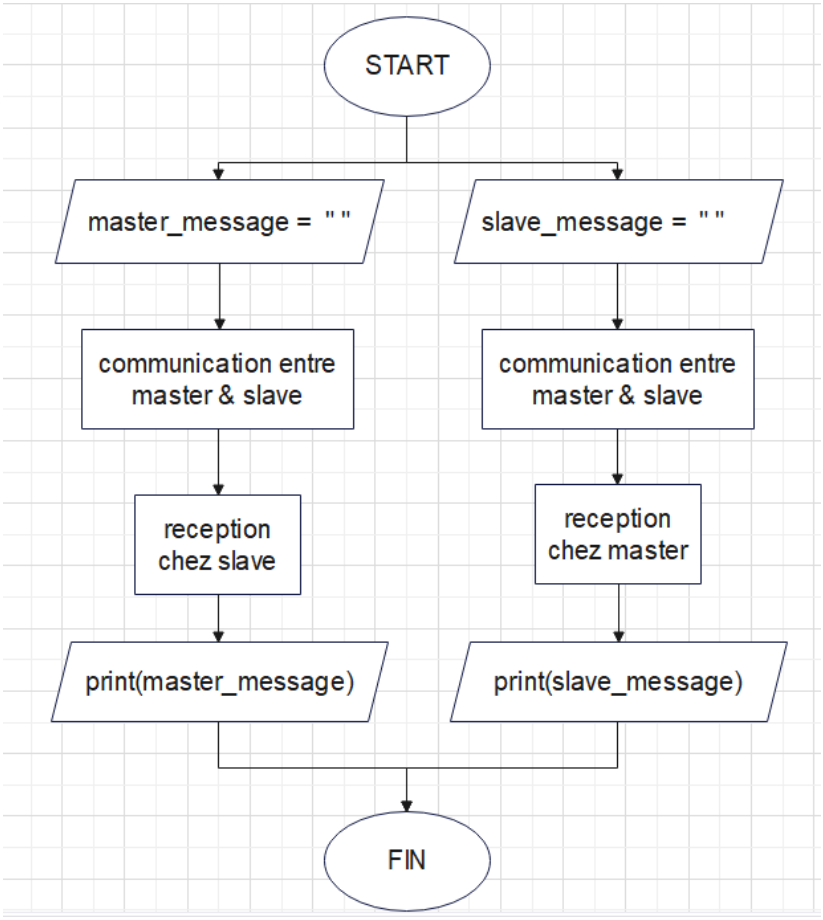
Afin de créer la communication Bluetooth, on doit relier/ binder les deux modules ensemble en utilisant la méthode du « Master » and « Slave ». Le premier module doit être définit comme « MASTER » et l'autre comme « SLAVE ». Pour ce faire nous devons mettre les modules HC-05 en mode « configuration » et utiliser des commandes « AT » qui sont des commandes permettant de modifier l'identité de chaque module.

Schéma de câblage :



Chaque module possède une borne RX et TX qui sont reliées aux bornes UART0 TX et RX du Raspberry. Celles si permettent de recevoir et envoyer les informations de l'un vers l'autre. L'alimentation est reliée au 5V.

Logigramme :



Code :

Comme nous avons deux Raspberry PI PICO, nous avons également deux codes. L'un permet de commander le « MASTER » et l'autre le « SLAVE ».

Master :

```
from machine import UART, Pin
import time
import json

uart1 = UART(0, baudrate = 38400, tx= Pin(0), rx=Pin(1)) # içi on définit la connection du module
str2 = "lo"

while (True): # boucle principale du programme.

    uart1.write('Salut')    # Envoie du mot 'Salut'
    print(uart1.read(5))
    if uart1.any() > 0:     # lorsque la pin UARt 1 reçois une valeur autre que "0" c'est qu'il à reçu une information
        strBT = str(uart1.readline(),"utf-8") #lecture de l'information
        print(strBT)
        strSplit = strBT.split(";") # permet de séparer les mots au niveau des ";"
        print(strSplit[0])
        for x in range(len(strSplit)):
            print(strSplit[x]) # Affichage l'information
    time.sleep(1)
```

Slave :

```
from machine import UART, Pin
import time
import json

uart1 = UART(0, baudrate = 38400, tx= Pin(0), rx=Pin(1)) # içi on définit la connection du module
str2 = "lo"

while (True): # boucle principale du programme.

    uart1.write('Hello')    # Envoie du mot 'Hello'
    print(uart1.read(5))
    if uart1.any() > 0:     # lorsque la pin UARt 1 reçois une valeur autre que "0" c'est qu'il à reçu une information
        strBT = str(uart1.readline(),"utf-8") #lecture de l'information
        print(strBT)
        strSplit = strBT.split(";") # permet de séparer les mots au niveau des ";"
        print(strSplit[0])
        for x in range(len(strSplit)):
            print(strSplit[x]) # Affichage l'information
    time.sleep(1)
```

Conclusion :

Nous avons appris à faire communiquer deux appareils via des modules Bluetooth HC-05. La communication sans fil est l'un des éléments important dans la robotique, qui nous sera utile lors de la réalisation du projet du Q2.

Conclusion générale :

Pour conclure, nous avons vu les principaux composants utiles à la réalisation des premiers systèmes robotisés. Du mouvement en passant par la signalisation jusqu'à la communication sans fil « Bluetooth ». Une belle base de connaissance à déjà été explorée dans le domaine de la robotique. Nous pouvons maintenant prendre notre envol en nous lançant dans un projet de conception robotisée, au cours duquel nous améliorerons nos connaissances en la matière.