A2071 - Robotique partie 1

Portfolio des tp

EPHEC

Bachelier en automatisation
2AU – Année 2022-2023

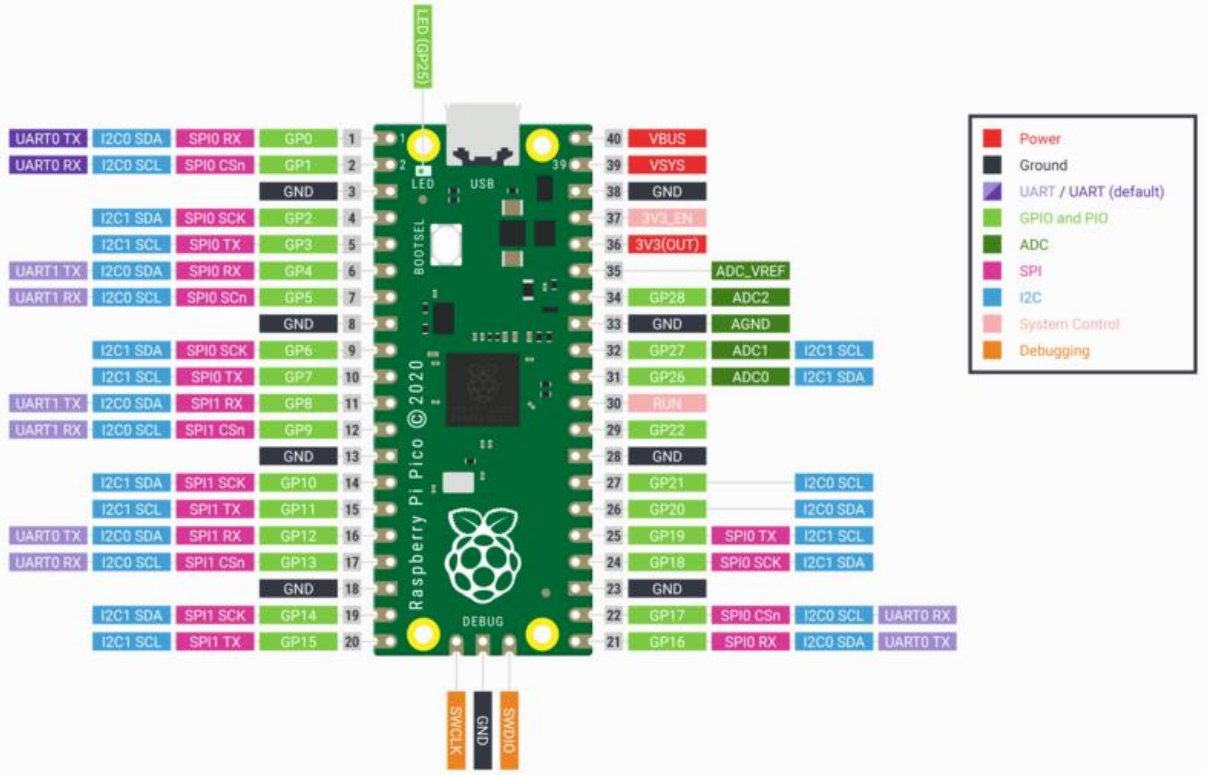
Créé par : MARCHAL Valentin – GOSSART Julien

Table des matières

Introduction générale :

Lors de ce cours de robotique nous avons appris à utiliser différents composant qui nous servirons plus tard lors de notre construction de notre futur robot.

Au début de chaque tp nous avions une explication sur le composant qu'on allait utiliser et son schéma de câblage, le tout contrôler pas un Raspberry pi pico dont voici le schéma.



Comme on peut le voir, chaque pin à une fonction particulière que nous utiliserons en fonction de l'objet du tp.

Pour nos schémas de câblage nous avons utilisé le logiciel Fritzing.

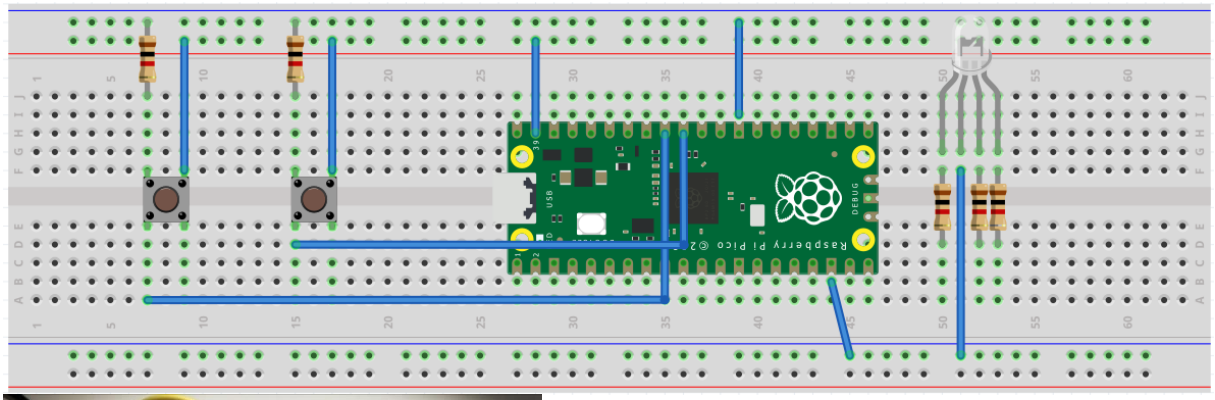
Tp 1 : séquence de couleur

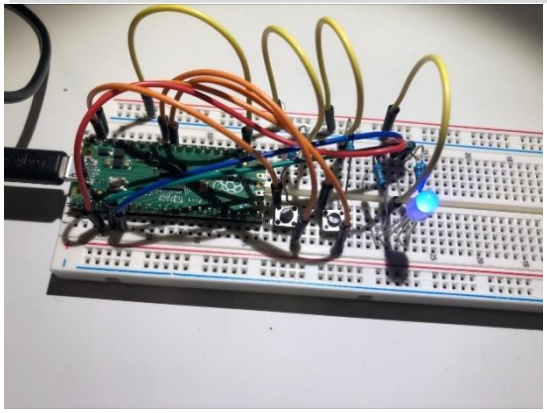
Introduction :

Lors de cette séance nous avons approché le hardware de la robotique, c'est-à-dire que nous avons commencé à câbler notre premier système. Ce dernier consiste à l'allumage d'une LED RGB à l'aide d'un Raspberry pi pico et deux bouton poussoir, en fonction du cahier de charge suivant :

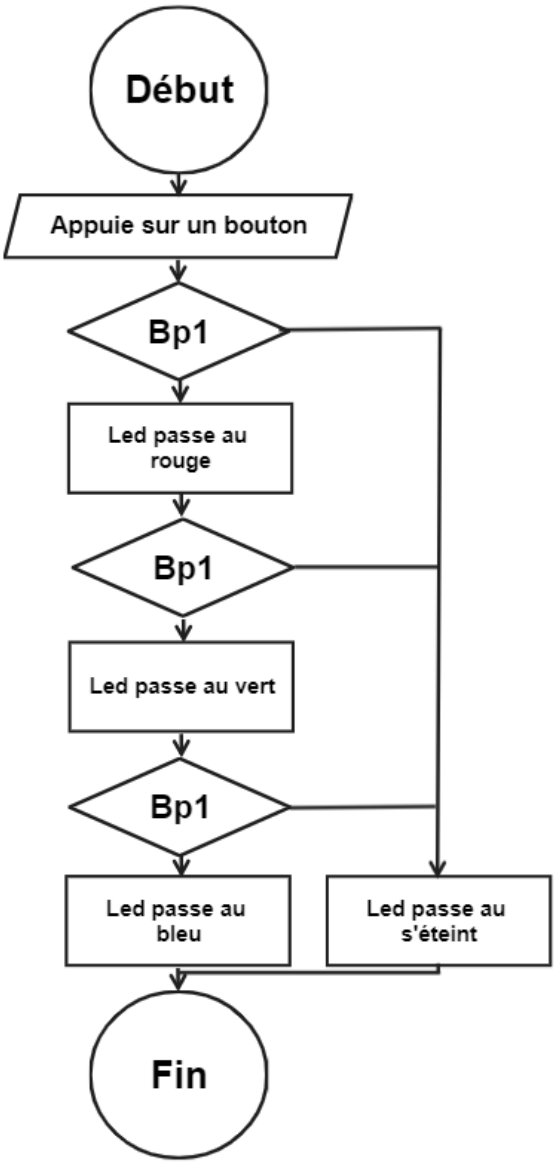
- A la mise sous tension la LED ne s'allume pas.
- A la première pression de BP1 la LED s'allume en rouge.
- Au second appui, la LED s'allume en vert.
- Au troisième appui, la LED s'allume en bleu.
- Au quatrième appui on relance le cycle à partir du premier appui.
- A la pression de BP2, la LED s'éteint peu importe où on est dans le cycle. Si on re-presse BP1, le cycle reprend au rouge.

Schéma de câblage et photo du câblage :





Organigramme :



Code:

```
from machine import Pin,UART
import time

led_rouge = Pin(2, Pin.OUT)
led_vert = Pin(3, Pin.OUT)
led_bleu = Pin(4, Pin.OUT)
B1 = Pin(27, Pin.IN)
B2 = Pin(26, Pin.IN)
decompte = 0        # decompte vas de 0 a 2 pour allumer sucéssivement les leds

def test_button():  #fonction regarde si les bouton son activé
    valeurB1 = B1.value()
    valeurB2 = B2.value()
    if valeurB1:
        time.sleep_ms(500) #temps d'arret pour bouton (pour éviter que ca défile trop vitre)
    return valeurB1, valeurB2

def allume_led(decompte):   #fonction allume lumiere en fonction de décompte 0, 1, 2 crément a chaque chagement de led
    if decompte == 0:
        led_rouge.value(1)  # allume led rouge
        led_vert.value(0)
        led_bleu.value(0)
    if decompte == 1:
        led_rouge.value(0)
        led_vert.value(1)
        led_bleu.value(0)
    if decompte == 2:
        led_rouge.value(0)
        led_vert.value(0)
        led_bleu.value(1)
    decompte += 1
    if decompte > 2:
        decompte = 0
    return decompte

def eteind_led():     #éteint toutes les leds
    led_rouge.value(0)
    led_vert.value(0)
    led_bleu.value(0)
    return

while True:           #boucle sans fin du programe
    time.sleep_ms(50)
    Bb1 = test_button()[0]
    Bb2 = test_button()[1]
    if Bb1 == True:
        decompte = allume_led(decompte)
    if Bb2 == True:
        eteind_led()
        decompte = 0
```

Conclusion :

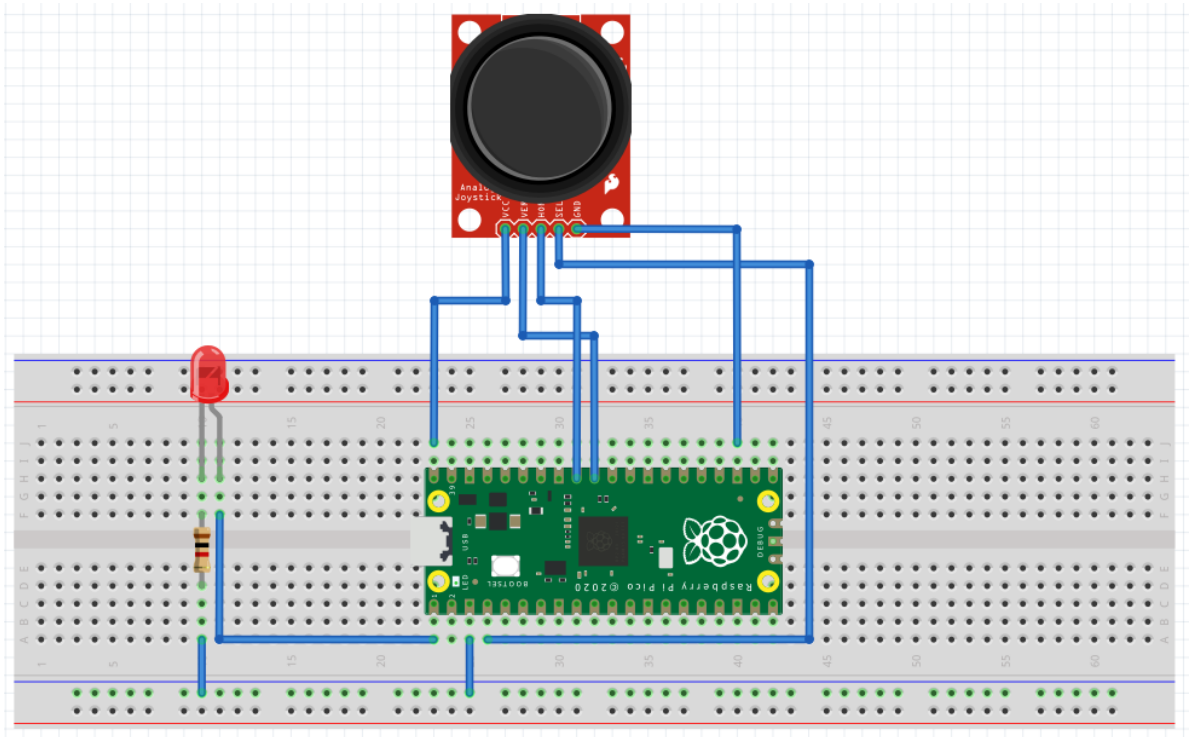
On constate qu'en fonction des signaux qu'on envoi sur les pins RGB de la LED elle émet une couleur différente. Cette LED fonctionne comme une palette de peintre dans lequel on fait le mélange d'une ou plusieurs couleurs pour en créer une autre. C'est alors pratique car du coup une LED peut afficher différents couleurs, on a donc pas besoin de plusieurs de chaque couleur qu'on désire.

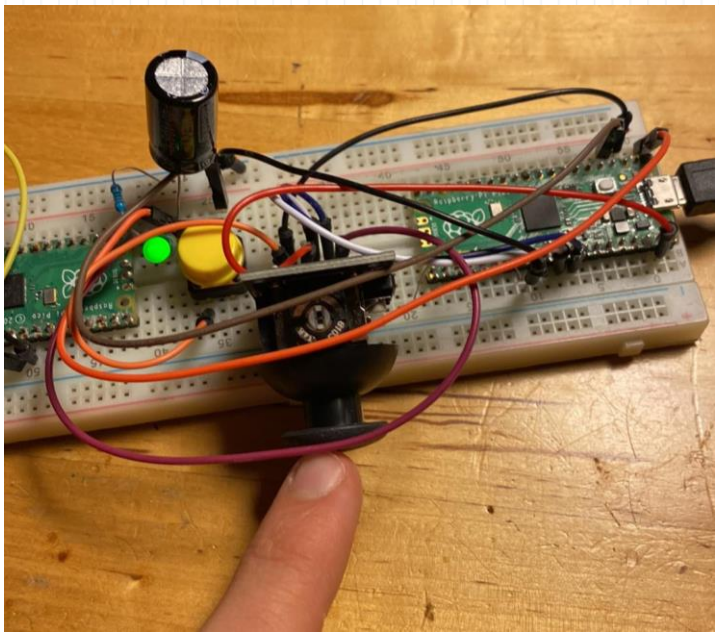
Tp 2 : luminosité via un joystick

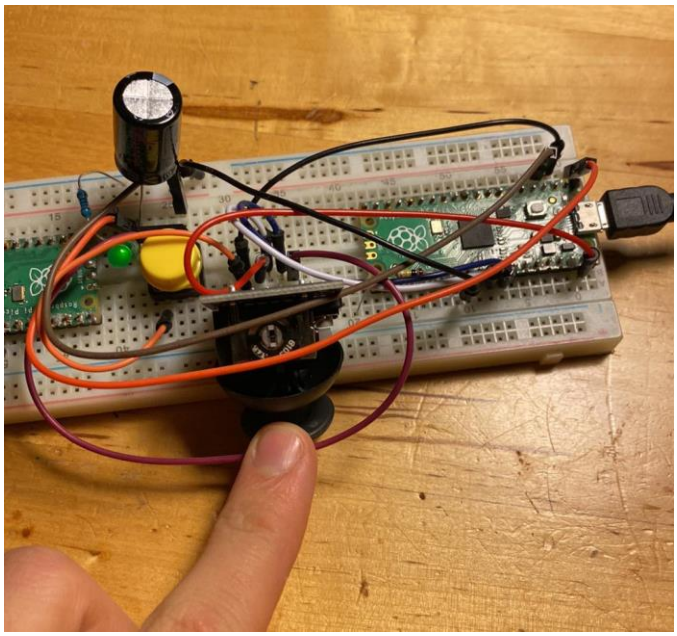
Introduction :

Lors de cette séance nous devions régler la luminosité d'une LED à l'aide d'un joystick. Si on le positionne vers le haut la luminosité augmente et à l'inverse, lorsque le joystick est vers le bas la LED s'éteint.

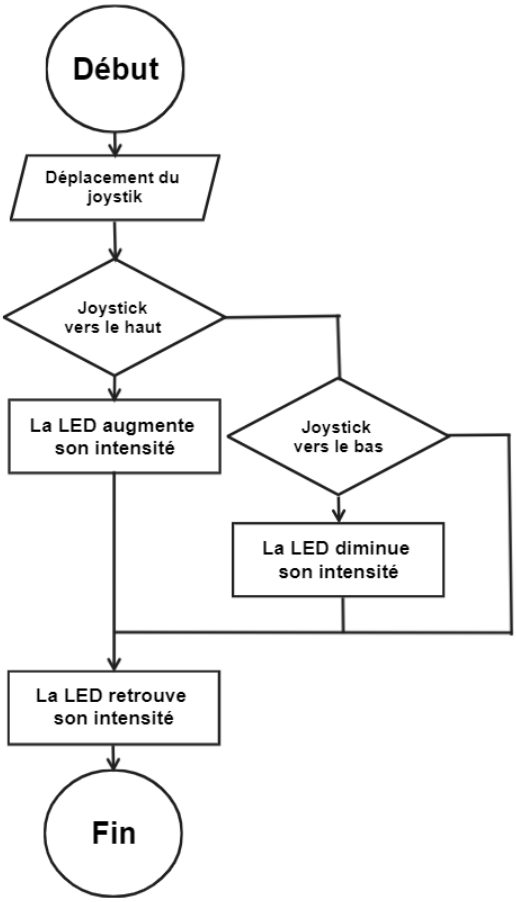
Schéma de câblage et photo du câblage :







Organigramme :



Code:

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

#Définition de nos variables
SW = Pin(2 , Pin.IN, Pin.PULL_UP)
adc = ADC(Pin(26))
adc2 = ADC(Pin(27))
pwm = PWM(Pin(0))
pwm.freq(2000)

#Boucle sans fin du programme
while True :
    print(SW.value())
    y_value = adc.read_u16()
    x_value = adc2.read_u16()
    print("la valeur en Y est:", y_value)
    print("la valeur en X est:", x_value)
    pwm.duty_u16(y_value)
    time.sleep_ms(200)
```

Conclusion :

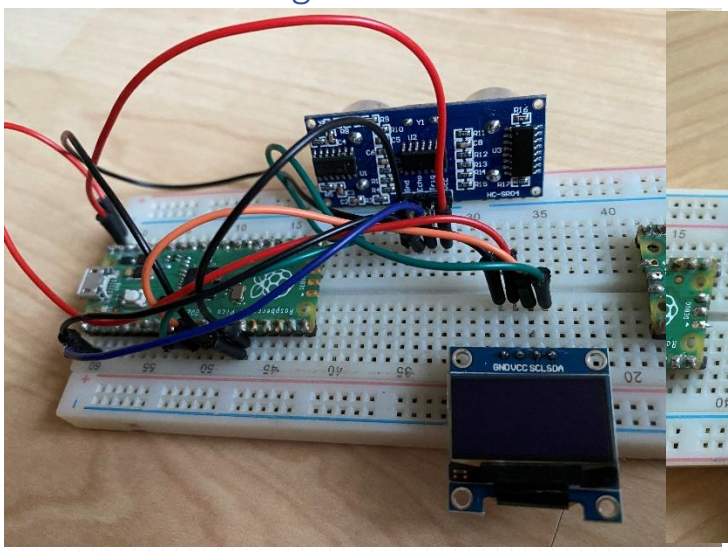
On constate que l'axe x et y du joystick sont des potentiomètres. Les résistances de ces 2 potentiomètres augmentent ou diminuent en fonction de la position du joystick, ce qui renvoie donc une position qui diffère selon l'axe x et y. On utilise ensuite la valeur de y pour régler l'intensité de la LED. De plus, le joystick possède une fonctionnalité supplémentaire, un bouton-poussoir, celui-ci fonctionne comme tout autre bouton poussoir.

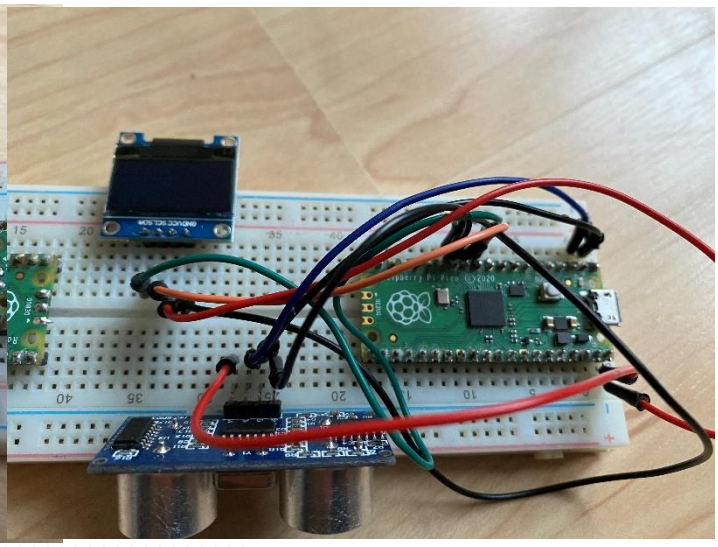
Tp 3 : détection de distance, capteur ultra son et affichage écran oled 0.96 pouce.

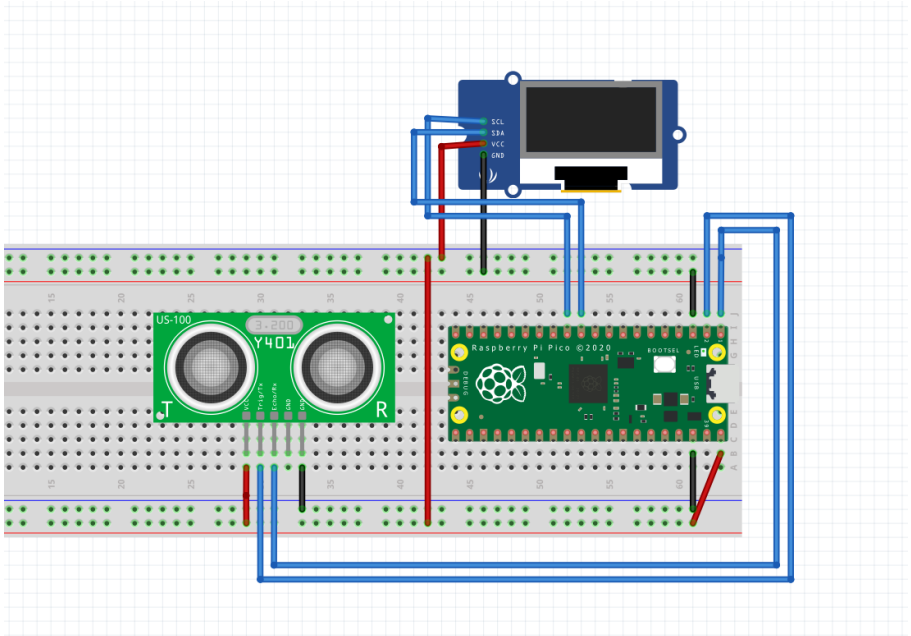
Introduction :

Le but de cette séance est de savoir calculer la distance entre le capteur et une paroi. De reprendre cette valeur et de l'affichée sur un écran oled. Les difficultés sont qu'il fait télécharger une librairie pour faire fonctionner l'écran ainsi que le capteur de distance.

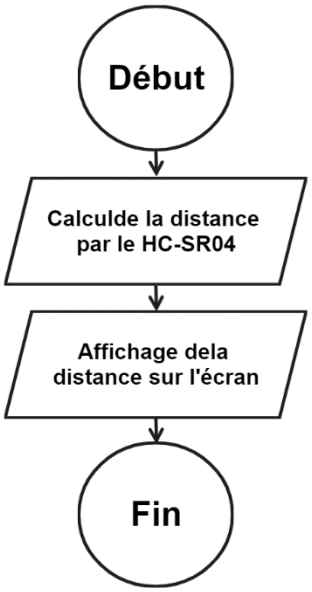
Schéma de câblage :







Organigramme :



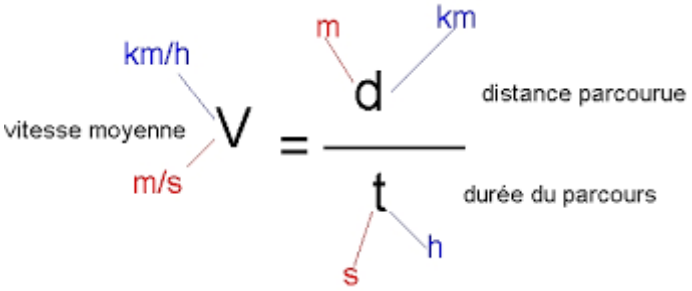
Code :

```
from hcsr04 import HCSR04 # inclu la librairie pour utiliser le capteur de distance à ultrasong
from machine import Pin, I2C # inclu la posibilitée des ce servir des pines I2C
from ssd1306 import SSD1306_I2C #inclu la librairie pour utiliser l'écrant au led
import time #inclu la librairie qui permet de mettre un délais

# réglage de l'écrant oled
pix_res_x  = 128
pix_res_y = 64

# Défini la fréquence de l'écrant oled ainci que les pines qui le relie au rasberry.
i2c_dev = I2C(0,scl=Pin(9),sda=Pin(8),freq=200000)

# Défini les pines auquel est relié le capteur de distance à ultrasong.
sensor = HCSR04(trigger_pin=1, echo_pin=0)

i2c_addr = [hex(ii) for ii in i2c_dev.scan()]

# Défini les réglages de l'écrant aux led ( largeur, hauteur, en nombre de pixel)
oled = SSD1306_I2C(pix_res_x, pix_res_y, i2c_dev)

while True:
    oled.fill(0) # on rempli l'écrant oled de rien, donc du noir.
    time.sleep_ms(50) # on met un délai de 50 ms
    distance = sensor.distance_cm()  # variable(la distance = valeur du capteur de distance)
    print('Distance:', distance, 'cm')
    oled.text("la distance est:",5,25) # affiche sur l'écrant "la distance est:", position x 5pixels, y 25 pixels
    oled.text(str(int(distance)),5,35) # affiche sur l'écrant "la distance est:", position x 5pixels, y 35 pixels
    oled.show()  # l'écrant oled affiche les variables qu'on luit a donné
```

Conclusion :

On peut voir que le HC-SR04 envoie des ultrasons devant lui, si ces derniers touchent un objet, elles sont renvoyées depuis cet objet vers notre capteur. Celui-ci va ensuite calculer la distance grâce au

temps qu'aura mis l'onde à aller et ensuite revenir grâce à la formule suivante :



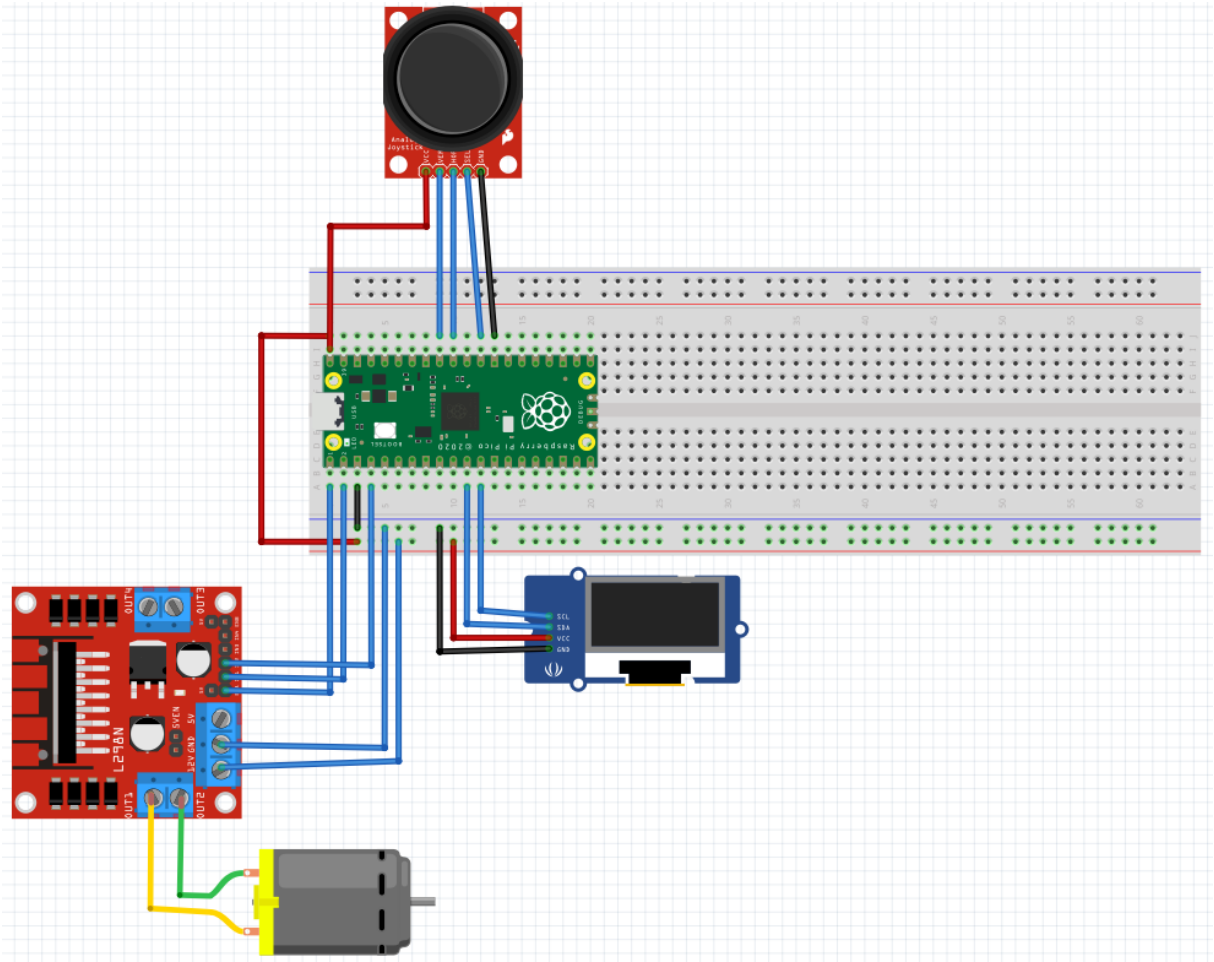
Ensuite nous prenons la distance que nous affichons sur notre écran oled, grâce aux différentes fonctions suivante : oled.fill(), oled.text(), oled.show(). Fonctions qui ont été importées à l'aide de la librairie HCR04.

Tp 4 : contrôle d’un moteur

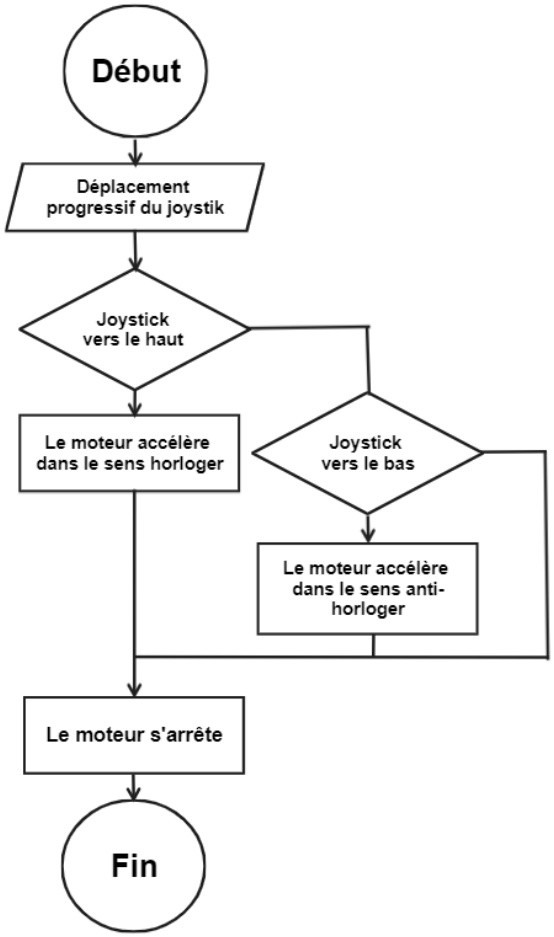
Introduction :

Lors de cette séance nous devions régler la vitesse d’un moteur via un joystick. Si on bouge vers le haut, le moteur tourne à une vitesse proportionnelle à la position du joystick et ce dans le sens horloger. A l’inverse, si on bouge vers le bas, le moteur tourne à une vitesse proportionnelle à la position du joystick mais dans le sens anti-horloger. De plus, on devait afficher le pourcentage de vitesse sur un écran oled 0.96’’. Malheureusement nous n’avons pas d’écran oled et donc n’avons pas afficher le pourcentage.

Schéma de câblage :



Organigramme :



Code:

```
from machine import Pin, UART, PWM, ADC #importe tous les types de fonction qu'on a besoin
from ssd1306 import SSD1306_I2C
import time

ENA = PWM(Pin(0)) # modulation de la largeur d'impulsion
ENA.freq(2000)
IN1 = Pin(1, Pin.OUT)
IN2 = Pin(2, Pin.OUT)

Vrx = ADC(Pin(26)) # valeur analogique de l'axe x
Vry = ADC(Pin(27)) # valeur analogique de l'axe y

def scale_value (value,in_min,in_max,out_min,out_max): # fonction qui traduit les valeurs de sortie du joystick en pourcentage
    scaled_value = (value - in_min) * (out_max - out_min) / (in_max - in_min) + out_min
    return scaled_value

while True:

    val_y = Vry.read_u16() # lecture et écriture de la valeur du joystick dans une variable
    val_x = Vrx.read_u16()

    print(val_y)
    time.sleep_ms(200) # utilisé pour avoir le temps (ms) de lire les valeurs

    if val_y <= 45000:
        ENA.duty_u16((-val_y)+65000) # modifie la vitesse du moteur (sens horloger)
        IN1.value(1)
        IN2.value(0)
        print(int(scale_value(y_value,0,45000,0,100))) # affichage du pourcentage

    elif val_y >= 55000:
        ENA.duty_u16(val_y) # modifie la vitesse du moteur (sens anti-horloger)
        IN1.value(0)
        IN2.value(1)
        print(int(scale_value(y_value,50000,65535,0,100))) # affichage du pourcentage

    else:
        IN1.value(0) #arrêt du moteur
        IN2.value(0)

while True:
    oled.fill(o)
    time.sleep_ms(50)
    oled.text(str(int(scale_value(y_value,0,45000,0,100))),5,25)
    oled.show()
```

Conclusion :

Nous faisons tourner le moteur dans les deux sens de rotations ainsi qu'un mode d'arrêt, les sens de rotations sont définis sur l'angle de l'axe y du joystick. Si l'axe y est positif (y >= 55000) la rotation sera positive, si négative (y <= 45000) rotation inversée (négative). Si y se trouve entre 55000 et 45000 c'est que il n'y a pas d'angle sur le joystick donc on le considère comme étant à l'arrêt le moteur est arrêté.

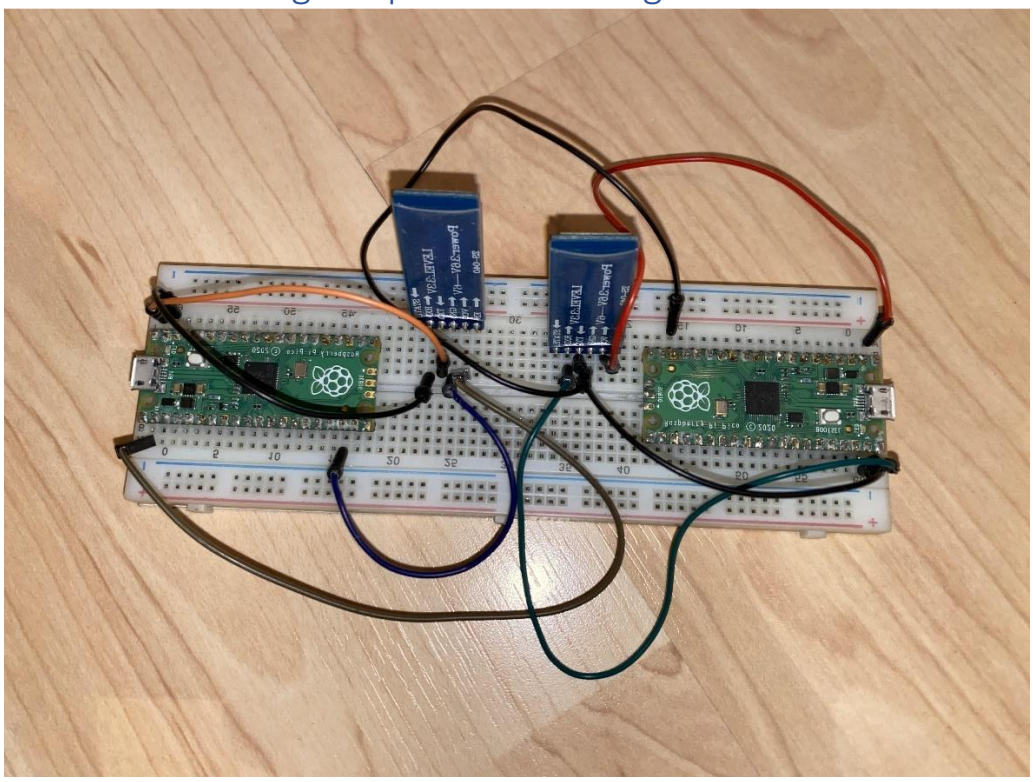
En parallèle de cela nous affichons le pourcentage de l'angle du joystick en Y, qui est calculer à l'aide de la fonction scale_value() sur un écran Oled, angle du joystick en Y qui correspond à la vitesse du moteur.

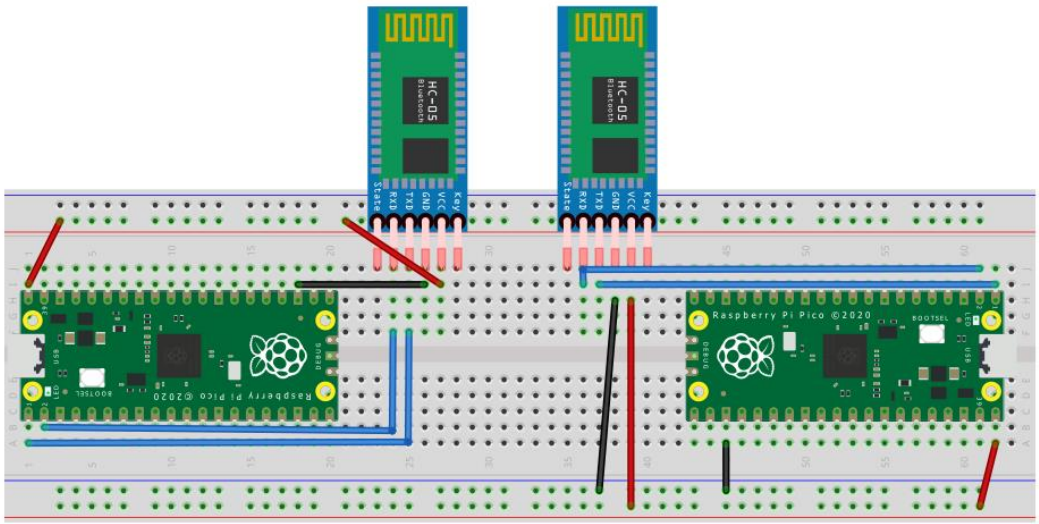
Tp 5 : liaison Bluetooth

Introduction :

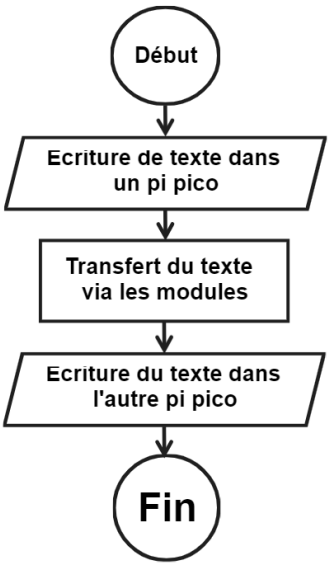
Nous voulons faire communiquer deux modules Bluetooth pour échanger des informations. Le but sera finalement de créer une télécommande qui enverrait les différentes informations des boutons et joystick de celle-ci via un module Bluetooth HC-05 (esclave ou maître) vers un autre module qui recevrais les informations.

Schéma de câblage et photo du câblage :





Organigramme :



Code :

```
from machine import UART, Pin #import l'utilisation des pines UART
import time #import une librairie nous permettant d'utiliser une fonction mettant du délait
import json

# défini (leur emplacements, le temps de rafraichissement des pines UART, la position du TX et RX) .
uart1 = UART(0, baudrate = 38400, tx= Pin(0), rx=Pin(1))

while (True):
    uart1.write('salut')  # on revoie "salut" au module breuthoot raccordé pour qu'il envoie le message
    print(uart1.read(5)) # on écrit l'information envoyée
    if uart1.any() > 0: # on regarde si on a recu une quelconque réponce
        strBT = str(uart1.readline(),"utf-8") # crée une variable str( qui équiveau au message recu)
        print(strBT) # on print la variable
        strSplit = strBT.split(";") # on recoit un string séparé par ;
        print(strSplit[0])
        for x in range(len(strSplit)): # on affiche tout les mots contenu dans la suite
            print(strSplit[x])
    time.sleep(1) # latence de 1 ms
```

Conclusion :

Nous avons dû programmer les modules Bluetooth pour les faire se relier et définir lesquels est l'esclave ou le maître. Pour ce faire, nous avons utilisé Arduino IDE pour leur donner leur adresse et rôle. Via différentes fonctions : AT, AT+NAME (rôle, esclave, maître), AT+RMAAD (sauvegarde).
Nous avons donc eu du mal pour faire cela. Mais finalement tout a fonctionné et somme parvenue à faire communiquer les différents modules. Nous pouvions transmettre des messages écrits ou des variables.

Conclusion générale :

Nous arrivons maintenant à contrôler tour les différents objets qu'on a vus et ce avec plus ou moins de difficultés. Les objets sont les suivants : LED RGB, joystick, HC-SR04, écran oled 0.96'', L298N, moteur CC, HC-05. On peut désormais lier le tout si l'envie nous vient.

Webographie/Bibliographie :

- http://docs.micropython.org/en/latest/rp2/quickref.html
- Fritzing (logiel)