TP les capteurs

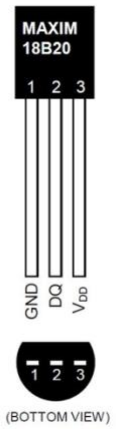
Objectif : prise en main des différents capteurs, affichage des paramètres

Rappel :
Sauvegarder le programme sous le nom ***main.py****.*
Copier le sur la mémoire flash de votre carte (PYBFLASH).
Attendre que la led rouge 1 s'éteigne.
Appuyer sur le bouton RST de la carte.

Nous allons voir ici l'utilisation du capteur de température, du capteur à ultrasons et du capteur barométrique.

Capteur de température

Le capteur DS18B20 est un capteur 1-Wire, cela signifie qu'il communique avec une carte maître au moyen d'un bus 1-Wire (1 fil) contrairement au I2C qui en utilise 2.
Repérez-le sur votre carte.



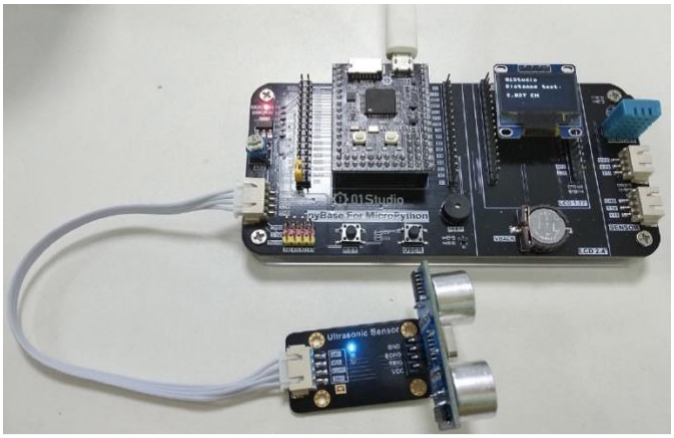
Nous utiliserons le port X11 sur auquel est relié le capteur.
Commandes Python :

Le programme :

```
#importation des modules
from pyb import delay
from machine import Pin,I2C
from ssd1306 import SSD1306_I2C
from onewire import OneWire
from ds18x20 import DS18X20
#Initialisation des éléments
i2c = I2C(sda=Pin("Y8"), scl=Pin("Y6")) # SoftI2C
oled = SSD1306_I2C(128, 64, i2c, addr=0x3c)
#initialisation du DS18B20
ow= OneWire(Pin('X11')) #ouverture du  single bus
ds = DS18X20(ow) # déclaration du capteur DS18B20
rom = ds.scan() #Scan des adresses de capteurs
while True:
    ds.convert_temp() # acquisition and conversion de la température
    #récupération de la 1ère adresse de capteurs pour le DS18B20
    temp = ds.read_temp(rom[0])
    #affichage des données
    oled.fill(0)
    oled.text('Club Prog LAB', 0, 0)
    oled.text('Temp test:', 0,20)
    oled.text(str('%.2f'%temp)+' C',0,40)
    oled.show()
    delay(1000)
```

Conseil : vérifiez bien que les décalages (indentations) soient les mêmes : une indentation = 1 TAB = 4 espaces.

Capteur à ultrasons



Utilisez un câble avec un connecteur à 4 fiches.
Soyez délicat pour le montage !
Une fois alimenté, la Led bleue du capteur doit s'allumer.
Le principe est assez simple, le capteur va emmètre des ultrasons et il va mesurer le temps de retour de l'écho.
La vitesse du son étant de 340m/s, il est aisé de calculer la distance.
Les mesures sont entre 2 et 450 cm avec une précision de 0,5 cm.
Commandes Python :

`HCSR04(trig,echo)`
Déclaration de l'émetteur et du récepteur (Y9 et Y10 pour nous)

`getDistance()` : se passe de commentaire ;)

Le code :

```
#importation des modules
from HCSR04 import HCSR04
from pyb import Pin,delay
from machine import Pin,I2C
from ssd1306 import SSD1306_I2C
#initialisation de l'écran
i2c = I2C(sda=Pin("Y8"), scl=Pin("Y6"))
oled = SSD1306_I2C(128, 64, i2c, addr=0x3c)
#initialisation des ports GPIO trig='Y9',echo='Y10'
trig = Pin('Y9',Pin.OUT_PP)
echo = Pin('Y10',Pin.IN)
HC=HCSR04(trig,echo)
while True:
    #affichage de l'écran
    oled.fill(0)
    oled.text('Club Prog LAB', 0, 0)
    oled.text('Distance test:', 0, 15)
    #Mesure de la distance
    Distance = HC.getDistance()
    # Affichage de la distance
    oled.text(str(Distance) + ' cm', 0, 35)
    oled.show()
    #print
    print(str(Distance)+' cm')
    delay(1000)
```

Conseil : vérifiez bien que les décalages (indentations) soient les mêmes : une indentation = 1 TAB = 4 espaces.

En rajoutant quelques lignes, on peut faire allumer certaines Leds en fonction de la distance :

```
#importation des modules
from HCSR04 import HCSR04
from pyb import Pin,delay
from machine import Pin,I2C
from ssd1306 import SSD1306_I2C
from pyb import LED
#initialisation de l'écran
i2c = I2C(sda=Pin("Y8"), scl=Pin("Y6"))
oled = SSD1306_I2C(128, 64, i2c, addr=0x3c)
#initialisation des ports GPIO trig='Y9',echo='Y10'
trig = Pin('Y9',Pin.OUT_PP)
echo = Pin('Y10',Pin.IN)
HC=HCSR04(trig,echo)
while True:
    #affichage de l'écran
    oled.fill(0)
    oled.text('Club Prog LAB', 0, 0)
```

```
    oled.text('Distance test:', 0, 15)
    #Mesure de la distance
    Distance = HC.getDistance()
    # Affichage de la distance
    oled.text(str(Distance) + ' cm', 0, 35)
    oled.show()
    if Distance<10:
        LED(2).on()
        delay(1000)
        LED(2).off()
    else:
        LED(3).on()
        delay(1000)
        LED(3).off()
    #print
    print(str(Distance)+' cm')
    delay(1000)
```

Dans le code ci-dessus, la led 2 s'allume si la distance est inférieur à 10 cm, sinon la led 3 s'allume.

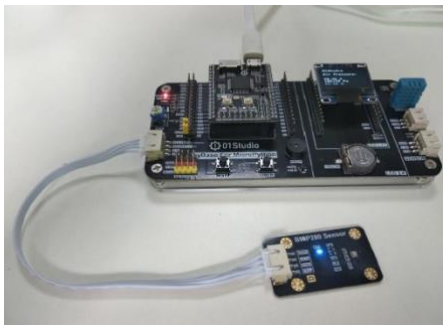
Défi :
À l'aide d'une boucle if… elif…else, modifies le programme afin que la led 2 s'allume si la distance est inférieur à 10cm, la led 3 s'allume si la distance est comprise en 10 et 40 cm, sinon, la led 4 s'allume.

Rappel :
Boucle if…. elif… else :

```
if condition1:
    instruction1
elif condition2:
    instruction2
else:
    intrsuction3
```

Capteur barométrique

Le capteur utilisé est un BMP280. C'est un capteur de pression atmosphérique de haute précision et aussi de température) utilisé sur les portables notamment de par sa petite taille et sa faible consommation.
Repérez-le sur la carte BMP280 Sensor.



Nous utiliserons la encore une nappe à 4 câbles.

La relation entre la pression atmosphérique *P* (Pa) et l'altitude *H* (m) est donnée par la formule suivante :

$$P = P_0 \times (1 - \frac{H}{44300})^{5,256}$$

Ce qui donne :

$$H = 44300 \times (1 - (\frac{P}{P_0})^{\frac{1}{5,256}})$$

P_0 est la pression atmosphérique au niveau de la mer (101325Pa).

Commandes Python :
`bmp280.BMP280(I2C)`: définit l'interface I2C du capteur. Nous mettrons `I2C(2)`.

`getTemp()` : récupère la température.

`getPress()` : récupère la pression atmosphérique.

`getAltitude()` : donne l'altitude.

Le code :

```
#importation des modules
import pyb
```

```
import bmp280
from machine import Pin,I2C
from ssd1306 import SSD1306_I2C

#Initialisation de l ecran
i2c = I2C(sda=Pin("Y8"), scl=Pin("Y6"))
oled = SSD1306_I2C(128, 64, i2c, addr=0x3c)

#Initialisation BMP280, I2C
BMP = bmp280.BMP280(I2C(2))

while True:

    #affichage Oled
    oled.fill(0)
    oled.text('LAB Prog', 0, 0)
    oled.text('Altitude:', 0, 15)

    # affichage temp
    oled.text(str(BMP.getTemp()) + ' C', 0, 35)
    # affichage pression atmo
    oled.text(str(BMP.getPress()) + ' Pa', 0, 45)
    # affichage altitude
    oled.text(str(BMP.getAltitude()) + ' m', 0, 55)
    oled.show()
    pyb.delay(1000)  # Collect every 1 second
```

Conseil : vérifiez bien que les décalages (indentations) soient les mêmes : une indentation = 1 TAB = 4 espaces.