Compte Rendu de notre Station Météo

Sommaire :

Le Cahier de Charges

Problématique: Comment récupérer les mesures du monde réel grâce a des capteurs pour les rendre accessibles ?

Objectif du projet: Nous voulions créer une station météo portative sur laquelle on peut connaitre différentes mesures de notre environnement (Taux de CO2, Température, Humidité et Pression Atmosphérique). Pour voir ces mesures, on peut utiliser l'écran LCD intégré, une application reliée en Bluetooth, ou une page internet, par Wi-Fi.

Périmètre du projet: Nous nous concentrons sur des personnes désignées.

Description fonctionnelle de besoin: L'application doit être en capacité de renvoyer sur l'écran, les valeurs mesurées par les capteurs tout en renouvelant ces données toutes les 10 secondes pour des valeurs presque en temps réel.

Contraintes: être capable d'avoir des valeurs précises et cohérentes.

Délais: le projet a commencé le 18/12/2024 et on aura terminer vers le 24/01/2025 avec quelques jours de délais pour finaliser le compte rendu.

Le matériel utilisé

Pour réaliser ce projet, nous avions eu besoin de quelques éléments matériels :

- Une carte Raspberry Pi PICO: elle sert a contrôler les différents capteurs et l'écran
- Un capteur SGP30 (Gaz)
- Un capteur BME280 (Température, pression atmoshérique et humidité)
- Afficheur LCD I2C Grove

Le code

```
# Importe tout les modules requis

from machine import Pin, I2C, UART

import uSGP30

import uBME280

from i2c_lcd import lcd
```

```
import time



# Défini l'interface du protocole I2C

i2c = I2C(0, scl = Pin(9), sda = Pin(8), freq = 1024)



# Défini l'interface UART utilisée pour le Bluetooth

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



# Défuini les adresses I2C requis pour les capteurs

BME_addr = 0x76;

SGP30_adrr = 0x58;



# Déclare les paramètres des capteurs

bme280 = uBME280.BME280(uBME280.BME280_OSAMPLE_1, BME_addr, i2c)

# Le BME280 prends comme argument : le mode du capteur, l'adresse i2c du
capteur et l'interface du protocole i2c

sgp30 = uSGP30.SGP30(i2c, SGP30_adrr)

# Le SGP30 prends comme argument : l'interface du protocole i2c et l'adresse
i2c du capteur

display = lcd(i2c)



# Crée les variables contenants les valeurs extérieures

t = str(bme280.read_temperature()/100)

p = str(bme280.read_pressure()/1000)

h = str(bme280.read_humidity()/1000)

# Les valeurs ci dessus doivent être converties en texte car l'écran
requiers des valeurs car sinon le programe renvoie une "TypeError: object
with buffer protocol required"
```

```
# De plus, les valeurs doivent êtres divisés par 100 ou par 1000 car elles ne sont pas dans la bonne unitée (capteur en pascal, valeur attendue en kPascal)

t_CO2, t_COV = sgp30.measure_iaq()

co2 = str(t_CO2)



# Crée la fonction qui permet de récupérer les valeurs

def Update_Valeurs():

t = str(bme280.read_temperature()/100)

p = str(bme280.read_pressure()/1000)

h = str(bme280.read_humidity()/1000)

t_CO2, t_COV = sgp30.measure_iaq()

co2 = str(t_CO2)



# Crée la fonction qui permet d'afficher le "premier écran"

def First_Screen():

# Affiche le premier "écran" (Température + Humidité)

display.home()

display.write("Temp = ")

display.write(t)

display.write(" °C")

display.setCursor(0,1)

display.write("THum = ")

display.write(h)

display.write(" %")



# Crée la fonction qui permet d'afficher le "deuxième écran"
```

```
def Second_Screen():

# Affiche le deuxième "écran" (Pression + CO2)

display.clear()

display.home()

display.write("P = ")

display.write(p)

display.write(" Pa")

display.setCursor(0,1)

display.write("TCO2 = ")

display.write(co2)

display.write(" ppm")



# Crée la fonction qui permet d'afficher les valeurs

def Update_Screen():

# Affiche le premier "écran"

First_Screen()

time.sleep(5)

Second_Screen()

time.sleep(5)

display.clear()



# Crée la fonction qui permet d'envoyer les valeurs en bluetooth

def Send_Datas(a, b, c, d):

BT.write(a)

BT.write(b)

BT.write(c)
```

```
BT.write(d)

# Lance le programme indéfiniment

while True:

Update_Valeurs()

Send_Datas(t, h, p, co2)

Update_Screen()
```

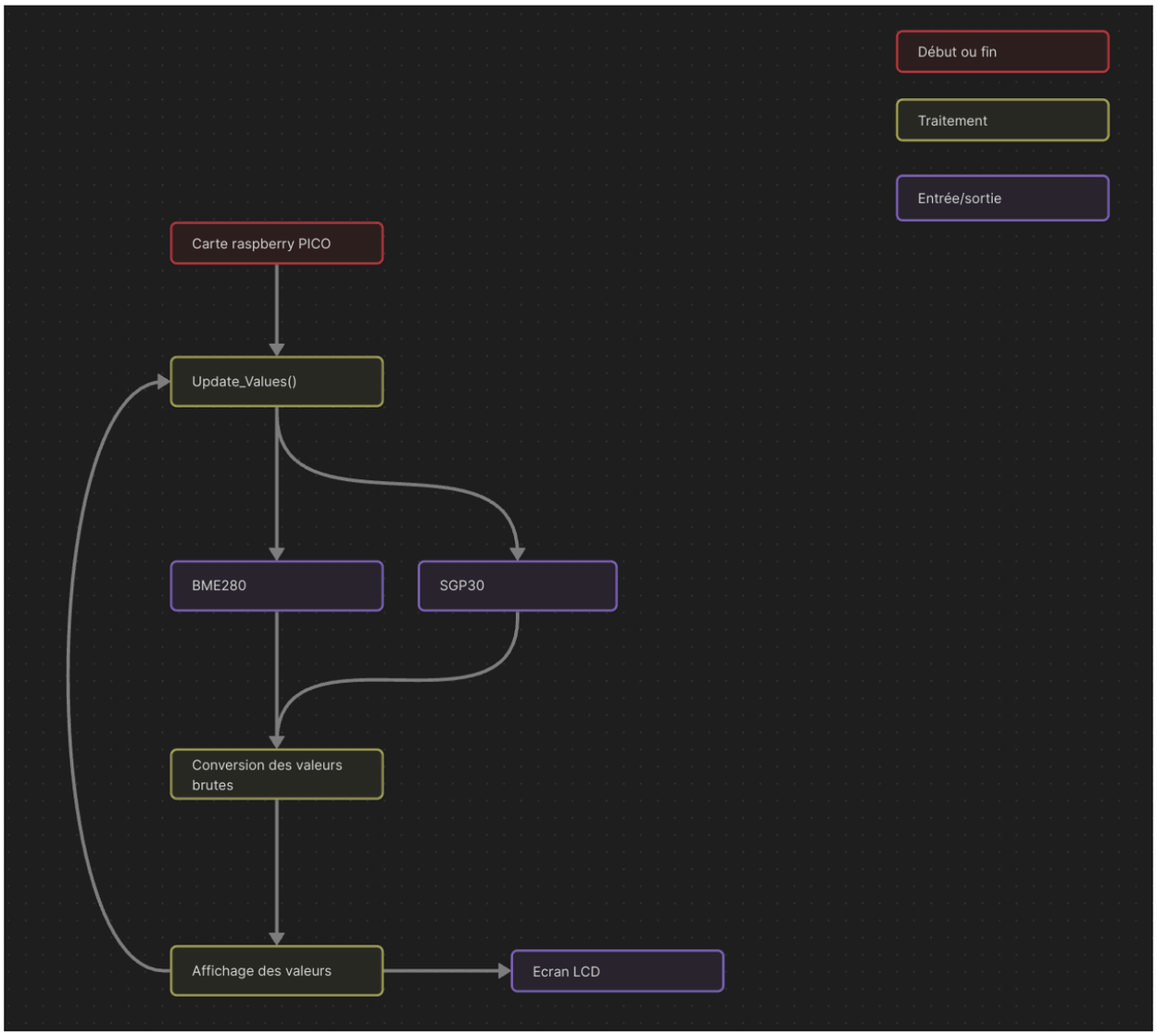
Les parties importantes du code

La fonction Update_Values() qui permet de récupérer les valeurs des différents capteurs

`def Update_Values():` Crée la fonction avec le mot-clé "def"
`t = str(bme280.read_temperature()/100)` Récupère la température depuis le BME280, la transforme en "str" pour que l'écran puisse l'afficher et divise par 100 pour adapter à la bonne unité
`p = str(bme280.read_pressure()/1000)` Récupère la pression atmosphérique depuis le BME280, la transforme en "str" pour que l'écran puisse l'afficher et divise par 1000 pour adapter à la bonne unité
`h = str(bme280.read_humidity()/1000)` Récupère l'humidité depuis le BME280, la transforme en "str" pour que l'écran puisse l'afficher et divise par 1000 pour adapter à la bonne unité
`t_CO2, t_COV = sgp30.measure_iaq()` Récupère le taux de CO2 depuis le SGP30
`co2 = str(t_CO2)` Converti le taux de CO2 en "str" pour pouvoir l'afficher sur l'écran

Schéma de l’algorithme

Schéma de l'algorithme



Captures d'écran

Capture d'écran de la page Web de la station, avec des éléments de template