Documentation Python

Table des matières

1 - Analyse des données du capteur AM107 **2**

a) Les données fournies par le capteur AM107 **2**

b) Typage des données pour les mesures **3**

2 - Les valeurs acceptables pour notre projet **4**

3 - Le code python **4**

a) Le nécessaire à l'exécution **4**

b) Création du fichier de configuration **4**

c) Le code **5**

Documentation Python

Objectif du projet :

L'agence CREAWEBUT propose des solutions avec l'Internet des Objets (IoT) pour simplifier la gestion des entrepôts de stockage ou faire des économies d'énergie (réglage de la température, contrôle d'hygrométrie, contrôle de luminosité …).

1 - Analyse des données du capteur AM107

Le JSON est structuré en différents objets (entouré par des accolades) et différents tableaux (entouré par des crochets). Dans ce cas-ci nous trouvons une case "object" contenant différentes données mesurées avec plusieurs capteurs.


```
"object":{
    "activity":37,
    "co2":575,
    "humidity":52.5,
    "illumination":43,
    "infrared":8,
    "infrared_and_visible":36,
    "pressure":1000.7,
    "temperature":20.5,
    "tvoc":210}}
```


a) Les données fournies par le capteur AM107

- Le niveau d'activité ("activity") :

Le capteur peut détecter la présence humaine afin de surveiller le niveau d'activité dans une certaine zone.

- La quantité de CO2 ("CO2"):

Le capteur récupère les données sur la concentration de CO2 afin d'aider les gens à réagir aux problèmes de qualité de l'air intérieur

- Le TVOC (Total Volatile Organic Compounds - "TVOC")

Le TVOC est la totalité des **composés organiques volatils dans l'air et permet de** surveiller le niveau de **composés organiques volatils** dans l'air ambiant afin de s'assurer de garder la meilleure qualité de l'air possible.

- La pression atmosphérique ("pressure")

Le capteur mesure et enregistre les données de la pression atmosphérique de l'endroit où il est situé.

- La température (“temperature”)

Le capteur mesure et affiche en temps réel la température de l’environnement dans lequel il se trouve.

- Le taux d’humidité (“humidity”)

Le capteur mesure et affiche en temps réel le taux d’humidité de l’environnement dans lequel il se trouve.

- La luminosité (“illumination”)

Le capteur récupère une information sur l’intensité lumineuse de l’environnement où celui-ci est installé.

- L’infrarouge et le visible (“infrared_and_visible”)

b) Typage des données pour les mesures

La luminosité :

Light	
Range	60000 lux (Visible + IR, IR)
Accuracy	±30%

La pression atmosphérique :

Barometric Pressure		
Sensor Type	—	MEMS
Range	—	300 - 1100 hPa (-40°C - 85°C)
Accuracy	—	±1 hPa
Resolution	—	0.1 hPa

La température :

Temperature	
Sensor Type	MEMS
Range	-20°C to + 70°C
Accuracy	0°C to + 70°C (+/- 0.3°C), -20°C to 0°C (+/- 0.6°C)
Resolution	0.1°C

L’humidité :

Humidity	
Sensor Type	MEMS
Range	0% to 100% RH
Accuracy	10% to 90% RH (+/- 3%), below 10% and above 90% RH (+/- 5%)
Resolution	0.5% RH

Le TVOC :

La concentration en CO2 :

Carbon Dioxide (CO_2)		
Sensor Type	—	Nondispersive Infrared (NDIR)
Range	—	400 - 5000 ppm
Accuracy (0°C to +50°C)	—	± (30 ppm + 3 % of reading)
Resolution	—	1 ppm
Range	—	0 - 60000 ppb
Accuracy	—	±15 %
Resolution	—	1 ppb

2 - Les valeurs acceptables pour notre projet

La **concentration en CO2** ne doit pas dépasser 1000 ppm.

Le **taux d'humidité** doit se situer entre 30% et 50% pour le stockage des vêtements

L'**activité** ne doit pas dépasser un certain seuil par rapport au nombre de personnes ayant accès à l'entrepôt la nuit / la journée.

La **pression atmosphérique** doit être la plus proche de 1000 hPa.

La **température idéale** doit être entre 18°C et 22°C dans l'entrepôt.

Le **niveau d'éclairage** de l'entrepôt doit être situé entre 100 et 150 lux.

Le **TVOC** doit être inférieur à 600 ppb et le plus proche possible de 200 ppb.

3 - Le code python

a) Le nécessaire à l'exécution

- pip install paho.mqtt

Le module paho doit être installé afin de pouvoir exécuter le programme.

Le programme doit être exécuté sur Linux car on utilise des signaux os (tel que SIGALRM).

b) Création du fichier de configuration

```
[server]

nom : chirpstack.iut-blagnac.fr

time : 1

[settings]

activity : true
```

```
Co2 : true

Humidity : false

illumination : true

pressure : true

Temperature : true

[devices]

AM107-2 : true

AM107-3 : true

AM107-4 : true

AM107-5 : true

AM107-6 : true
```

c) Le code

```
from os import *

from time import sleep

import paho.mqtt.client as mqtt

import json

import signal

from datetime import datetime

from configparser import ConfigParser


StockDico = dict()

parser = ConfigParser()

parser.read('Configuration.ini')


def envoie():

    for i in StockDico:

        fd = open(i+'.txt', O_RDWR | O_CREAT | O_APPEND, 0o644)

        write(fd,bytes(StockDico[i], 'utf-8'))
```

```
def fonctionALRM(signal,frame):
    envoie()

signal.signal(signal.SIGALRM,fonctionALRM)

def get_data(mqttc, obj, msg):

    jsonMsg = json.loads(msg.payload)
    try:
        """
        Test si le nom de l'appareil est dans la liste des paramètres.
        """
        if(parser.get('devices',jsonMsg["deviceName"]) == 'true'):

            today = datetime.strftime(datetime.now(),"%Y %m %d | %H h %M min %S s")
            device = jsonMsg["deviceName"]

            chaine = "| {0} | {1} ".format(today,device)

            if(parser.get('settings','activity') == 'true'):
                donneeAct = jsonMsg["object"]["activity"]
                #chaine3 ='| Activité : '+ str(donneeAct)+''
                chaine = chaine+'| Activité : '+ str(donneeAct)+''
            if(parser.get('settings','Co2') == 'true'):
                donneeCo2 = jsonMsg["object"]["co2"]
                chaine = chaine+'| Activité : '+ str(donneeCo2)+''
                # chaine1 ='| Co2 : '+ str(donneeCo2) +' ppm '
            if(parser.get('settings','Humidity') == 'true'):
```

```
                donneeHum = jsonMsg["object"]["humidity"]
                chaine = chaine+'| Activité : '+ str(donneeHum)+''
                # chaine3 ='| Humidité : '+ str(donneeHum) +' % '
            if(parser.get('settings','illumination') == 'true'):
                donneeIll = jsonMsg["object"]["illumination"]
                chaine = chaine+'| Activité : '+ str(donneeIll)+''
                # chaine3 ='| illumination : '+ str(donneeIll) +' % '
            if(parser.get('settings','pressure') == 'true'):
                donneePre = jsonMsg["object"]["pressure"]
                chaine = chaine+'| Activité : '+ str(donneePre)+''
                # chaine3 ='| Pression : '+ str(donneePre) +' Pa '
            if(parser.get('settings','Temperature') == 'true'):
                donneeTemp = jsonMsg["object"]["temperature"]
                chaine = chaine+'| Activité : '+ str(donneeTemp)+''
                #chaine2 ='| Température : '+ str(donneeTemp) +' °C '
            if(parser.get('settings','tvoc') == 'true'):
                donneeTvoc = jsonMsg["object"]["tvoc"]
                chaine = chaine+'| Activité : '+ str(donneeTvoc)+''
                #chaine3 ='| tvoc : '+ str(donneeTvoc) +' % '
            chaine = chaine+' | \n'
            StockDico[device] = chaine
    except:
        pass


mqttc = mqtt.Client()
mqttc.connect(parser.get('server','nom'), port=1883, keepalive=600)


mqttc.subscribe("application/1/device/+/event/up")
mqttc.subscribe("application/1/device/24e124128c012114/event/up")
```

```
mqttc.on_message = get_data
mqttc.loop_start()


while True:
    signal.alarm(int(parser.get('server','time'))*10)
    sleep((int(parser.get('server','time'))*10)+1)
    StockDico = []
```