SAE DevApp : Documentation technique et utilisateur du script Python

Groupe G2A09 – 18/01/2023

Sommaire

Oryann Prochaska
Thomas Demeyere
Anton Xu
Louis Yvelin

script_mqtt.py

script_mqtt.py est un script Python permettant de lire des données envoyées par des capteurs et d'en extraire les données désirées.

Structure

Initialisation variables

Le script Python récupère les données voulues et les exporte au format `.json`.
Pour cela on s'abonne au flux MQTT du réseau LoRaWAN de l'IUT de Blagnac avec les codes suivants :

```
# Port et server initialisé pour se connecter au réseau LoRa de l'IUT
mqttserver = "chirpstack.iut-blagnac.fr"
mqttport = 1883
```

Nous avons besoin du nom du serveur mqtt ainsi que le numéro du port.

Nous initialisons une variable de type booléen afin de savoir lors des appels de fonction si des données nouvelles ont pu être récupérées.
Nous initialisons aussi un dictionnaire de données vide afin de récupérer les valeurs.

```
data_waiting = False
output = {}
```

Définition fonctions

Nous définissons premièrement une fonction de chargement du fichier de configuration :

PY

```
def load_config():
    global config
    # le fichier config.json sera donné par le programme java
    try:
        f = os.open(sys.path[0]+'/config.json', os.O_RDONLY) # ouverture du fichier config.json en lecture
        config_raw = os.read(f, 1024) # lecture de 1024 bytes du fichier pour le stocker en string
        config = json.loads(config_raw) # retourne un dictionnaire python par un string en format json
        os.close(f)
        return True # fichier chargé correctement
    except:
        return False # pas de fichier de configuration
```

Nous l'ouvrons en lecture avec `os.open` afin de récupérer les données qu'il contient. Nous indiquons au programme que nous souhaitons lire les 1024 premiers bytes avec la fonction `os.read`.
Celle-ci retourne True ou False selon si le fichier config.json a été chargé ou non.

Ensuite, nous définissons une fonction appelée à chaque nouveau message du flux MQTT, permettant de stocker ses données dans le dictionnaire initialisé au début, en rechargeant au préalable le fichier de configuration :

PY

```
def get_data(mqtt, obj, msg):
    print('data') # affichage pour informer l'utilisateur
    config_found = load_config() # relecture du fichier de configuration
    if config_found: # faire seulement si le fichier de configuration a pu être chargé
        global jsonMsg # creation d'une variable globale
        jsonMsg = json.loads(msg.payload) # recuperation de tout ce qui est envoyé par les capteurs en un
format lisible par python
        for data in jsonMsg['object']:
            if data in config['data']:
                # tuple (valeur, True/False selon si le seuil configuré a été dépassé ou non)
                output[data] = (jsonMsg["object"][data], jsonMsg["object"][data] > config['data'][data] if
config['data'][data] != None else False)
        global data_waiting # reinitialiser la variable pour que la modification soit effective en dehors de
la fonction
        data_waiting = True # changement de la variable globale data_waiting depuis la fonction
    else:
        print('pas de fichier de configuration, données ignorées')
```

Cette fonction changera par la même occasion la variable globale data_waiting initialisée plus tôt. Cette modification nous servira à indiquer à la fonction d'écriture qu'il y a effectivement de nouvelles données en attente d'écriture.

La dernière fonction est celle qui s'occupera d'écrire les données dans un fichier json :

PY

```
def ecriture(numero, frame):
    global data_waiting
    signal.alarm(30) # la fonction sera rappelée dans 30 secondes
    try:
        if data_waiting: # si le boolean global data_waiting est True
            msg = json.dumps(output)
            print("écriture de :\n" + msg)
            fd = os.open(sys.path[0]+'/'+config['filename'], os.O_WRONLY|os.O_CREAT|os.O_TRUNC) # crée ou ouvre
le fichier d'ecriture des données
            os.write(fd, msg.encode()) # encode en UTF-8 par defaut
            os.close(fd)
            data_waiting = False
    except Exception as e: # leve une exception si l'écriture du fichier est impossible
        print(e)
```

Dans les première lignes on indique que la fonction se rappelera elle-même toutes les 30 secondes.
On effectue un test sur la variable data_waiting afin de savoir si il y a des données en attente d'écriture. Si c'est le cas, on ouvre un fichier avec `os.open` et on les écrit dedans avec `os.write`.
La variable data_waiting est remise à False.

Initialisation du script

Il reste à initialiser le script en se connectant au flux de données MQTT à travers un client mqtt.

Nous mettons en place premièrement la redirection du signal `SIGALRM` vers la fonction ecriture, ainsi que le premier signal d'alarme :

PY

```
signal.signal(signal.SIGALRM, ecriture) # va rediriger le signal d'alarme vers la fonction ecriture
signal.alarm(10) # va lancer une seule alarme 10 sec après le lancement du programme
```

Premier signal d'alarme 10 secondes après le lancement du script

Pour finir, nous initialisons le client MQTT :

PY

```
client = mqtt.Client()
client.connect(mqttserver, mqttport, 600)

client.subscribe("application/1/device/+/event/up") # s'abonner au topic d'upload des capteurs

client.on_message = get_data # chaque message reçu appelera la fonction get_data()

print("En attente de données...") # affichage pour informer l'utilisateur

client.loop_forever() # boucle tant que l'on ne stoppe pas le programme de force
```

Installation

Prérequis

- Être sur un système d'exploitation UNIX/Linux
- Avoir Python 3+

Installation de paho-mqtt

Dans la ligne de commande exécuter la commande
`pip install paho-mqtt`

Exécution du script

Fichier `config.json` de configuration du script

Le fichier de configuration est normalement généré par l'application Java associée.
Il devrait avoir une structure semblable à celle-ci :

JSON

```
{
    "filename": "data.json",
    "data": {
        "co2": null,
        "temperature": 30,
        "humidity": 20
    }
}
```

Exécution du script Python

Dans la ligne de commande exécuter le script :
`python script_mqtt.py`