SAÉ 3.01 2023-2024 - Documentation Python - Équipe 5

Table des matières

1. Introduction

Notre application python permet de récupérer les données envoyées par les différents capteurs présents dans les entrepôts de stockage de MalyArt qui les utilise pour s'assurer que les conditions de stockage sont optimales pour les oeuves d'art qui y sont entreposées.

L'objectif est de pouvoir afficher ces données sur une application JavaFX pour que les employés de MalyArt puissent les consulter facilement.

2. Prérequis

L'application nécessite plusieurs choses qu'il vous faudra installer au préalable :

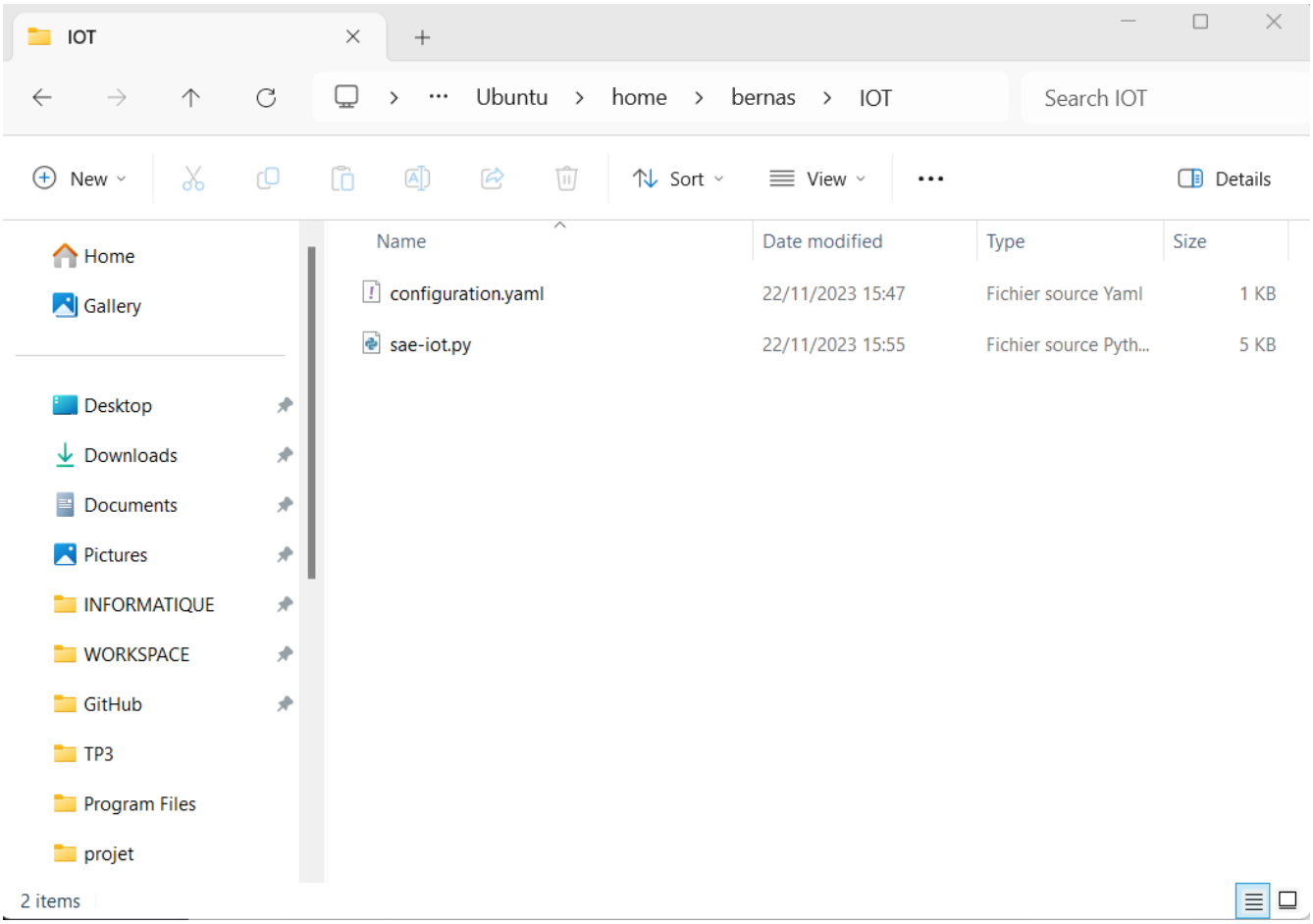
- `Python 3` ou une version ultérieure. (https://www.python.org/downloads/)
- `pip` pour installer les dépendances. (Inclus dans Python 3.4 et supérieur)
- `csv` pour lire les fichiers csv. (Bibliothèque Python)
- `yaml` pour lire les fichiers yaml. (`pip install pyyaml`)
- `paho.mqtt.client` pour communiquer avec le broker MQTT. (`pip install paho-mqtt`)
- `json` pour lire les fichiers json. (Bibliothèque Python)
- `time` pour gérer le temps. (Bibliothèque Python)

- `signal` pour gérer les signaux. (Bibliothèque Python)

> **WARNING** L'application ne fonctionne que sur **Linux**.

3. Installation

1. Téléchargez le fichier python à partir du lien suivant : sae-iot.py
2. Téléchargez le fichier de configuration à partir du lien suivant : configuration.yaml
3. Regroupez les deux fichiers dans un même dossier.

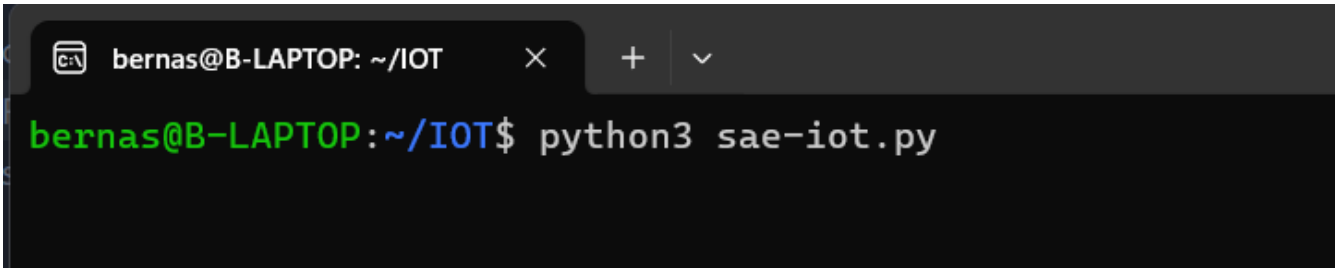


> **WARNING** Il est important que les noms des fichiers soient respectés.

4. Utilisation

Pour utiliser l'application, il vous suffit d'ouvrir un terminal dans le dossier contenant les deux fichiers et de lancer la commande suivante :

```
python3 sae-iot.py
```



5. Exemple d'utilisation

Lorsque l'application est lancée, elle va alors se connecter au broker MQTT et s'abonner aux topics configurés dans le fichier de configuration. Elle va ensuite afficher les données reçues dans la console et les enregistrer dans un fichier csv.

```
bernas@B-LAPTOP:~/IOT$ python3 sae-iot.py
~ Retrieving configuration file
~ Selected data : ['temperature', 'humidity', 'co2', 'activity', 'tvoc', 'illumination', 'infrared', 'infrared_and_visible', 'pressure']
Exécution pendant 600 secondes...
~ Connected with result code 0
~ Subscribed to AM107/by-room/+/data
~ Subscribed to AM107/by-room/E003/data
~ Subscribed to AM107/by-room/E006/data
~ Saving data on : data.csv
~ Waiting for data

[E105] temperature: 17.6, humidity: 42.5, co2: 436, activity: 0, tvoc: 197, illumination: 1, infrared: 1, infrared_and_visible: 1, pressure: 989.1, time: 17
05265921.7014618
Threshold exceeded - temperature: 17.6 (Threshold: 10)
Moyenne des données de la pièce E105 :
temperature : 17.87 humidity : 36.5 co2 : 657.17 activity : 153.33 tvoc : 121.5 illumination : 17.33 infrared : 3.83 infrared_and_visible : 15.0 pressure :
996.05


[hall-entrée-principale] temperature: 18.3, humidity: 43, co2: 453, activity: 0, tvoc: 227, illumination: 1, infrared: 1, infrared_and_visible: 1, pressure:
 990, time: 1705265923.1012342
Threshold exceeded - temperature: 18.3 (Threshold: 10)
Moyenne des données de la pièce hall-entrée-principale :
temperature : 18.93 humidity : 44.62 co2 : 563.83 activity : 6.5 tvoc : 502.42 illumination : 38.67 infrared : 26.0 infrared_and_visible : 67.33 pressure :
993.67
```

6. Explication de la structure du code

Le programme python est divisé en deux parties : - Le fichier de configuration (configuration.yaml) - Le fichier principal (sae-iot.py)

6.1. Le fichier de configuration

Le fichier de configuration est un fichier yaml qui permet de configurer l'application. Il est divisé en plusieurs parties :

6.1.1. Le broker

Cette partie permet de configurer le broker MQTT. Il est possible de modifier l'adresse du broker et le port utilisé.

```
url: "chirpstack.iut-blagnac.fr"
port: 1883
keepalive: 60
```

6.1.2. Les topics

Cette partie permet de configurer les topics MQTT. Il est possible de modifier les topics auxquels l'application va s'abonner.

```
topics: ["AM107/by-room/+/data","AM107/by-room/E003/data","AM107/by-room/E006/data"]
```

6.1.3. Les fichiers

Cette partie permet de configurer les fichiers csv ou se trouvent les données des capteurs. Il est possible de modifier les fichiers csv utilisés par l'application.

```
dataFile: "data.csv"
alertFile: "alert.csv"
```

6.1.4. Les données des capteurs à récupérer

Cette partie permet de configurer les données des capteurs à récupérer. Il est possible de modifier les données des capteurs à récupérer.

```
selectedData:
["temperature","humidity","co2","activity","tvoc","illumination","infrared","infrared_
and_visible","pressure"]
```

6.1.5. Les temps d'attente entre chaque écoute

Cette partie permet de configurer les temps d'attente entre chaque écoute. Il est possible de modifier les temps d'attente entre chaque écoute. (Le temps ou l'application va ecouter les données des capteurs est aussi présent mais il n'est pas possible de le modifier dans l'application)

```
rest_duration  : 30
running_time : 10
```

6.1.6. Les seuils d'alertes

Cette partie permet de configurer les seuils d'alertes. Il est possible de modifier les seuils d'alertes.

```
thresholds:
  temperature : 10
  humidity : 45
  co2 : 10000
  activity : 300
  tvoc : 500
  illumination : 100
```

```
infrared : 100
infrared_and_visible : 100
pressure : 1100
```

6.2. Le fichier principal

Le fichier principal est le fichier python qui permet de récupérer les données des capteurs s'organise autour d'une fonction principale qui est la fonction `on_message(client, userdata, msg)`.

Cette fonction est appelée à chaque fois que l'application reçoit un message du broker MQTT. Celle-ci va alors recupérer les données json du message, les convertir en dictionnaire python et les enregistrer dans un fichier csv puis les afficher dans la console.

La fonction va ensuite vérifier si les données reçues sont supérieures aux seuils d'alertes. Si c'est le cas, elle va enregistrer les données dans un fichier csv d'alerte.

Pour finir la fonction va afficher la moyenne des données reçues pour la pièce dans la console.

En plus de la fonction principale, le fichier python contient aussi une fonction `on_connect(client, userdata, flags, rc)` qui est appelée à chaque fois que l'application se connecte au broker MQTT. Celle-ci va alors s'abonner aux topics configurés dans le fichier de configuration et créer les fichiers csv si ils n'existent pas.

Nous avons également une partie qui permet de gérer les temps d'attente entre chaque écoute à l'aide de la fonction `signal.signal(signal.SIGALRM, handler)` qui permet de gérer les signaux et de la fonction `handle_execution(signum, frame)` qui permet de gérer les signaux reçus par l'application. Nous utilisons les alarmes pour gérer les temps d'attente entre chaque écoute. Lorsque l'application reçoit un signal, elle va alors se mettre en pause pendant le temps d'attente configuré dans le fichier de configuration puis elle va reprendre son écoute.

Au début du programme, nous avons une partie qui permet de lire le fichier de configuration et de le convertir en dictionnaire python.

L'application tourne en boucle jusqu'à ce que l'utilisateur appuie sur `CTRL+C` pour l'arrêter. Lorsque l'application est arrêtée, elle va alors se désabonner des topics et se déconnecter du broker MQTT.

7. Tests

Nous allons maintenant tester l'application en utilisant le broker MQTT de l'IUT de Blagnac. Pour cela, nous allons utiliser le topic `AM107/by-room/E003/data` qui contient les données du capteur de la salle E003.

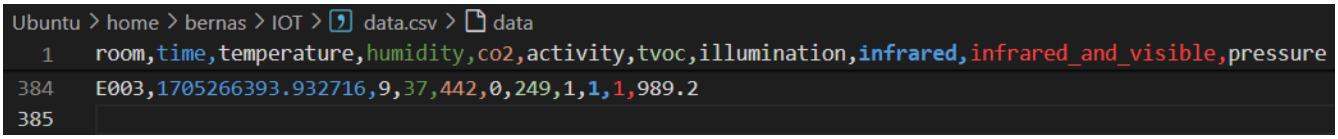
7.1. Test 1 : Lancement de l'application et réception des données normales

Pour ce premier test, nous allons lancer l'application et vérifier que celle-ci reçoit bien les données du capteur de la salle E003.

```
bernas@B-LAPTOP:~/IOT$ python3 sae-iot.py
~ Retrieving configuration file
~ Selected data : ['temperature', 'humidity', 'co2', 'activity', 'tvoc', 'illumination', 'infrared', 'infrared_and_visible', 'pressure']
Exécution pendant 600 secondes...
~ Connected with result code 0
~ Subscribed to AM107/by-room/E003/data
~ Saving data on : data.csv
~ Waiting for data

[E003] temperature: 9, humidity: 37, co2: 442, activity: 0, tvoc: 249, illumination: 1, infrared: 1, infrared_and_visible: 1, pressure: 989.2, time: 1705266393.932716
Moyenne des données de la pièce E003 :
temperature : 16.79 humidity : 48.25 co2 : 580.92 activity : 26.25 tvoc : 458.75 illumination : 11.75 infrared : 4.92 infrared_and_visible : 14.0 pressure : 995.83
```

Comme nous pouvons le voir sur l'image ci-dessus, l'application a bien reçu les données du capteur de la salle E003 et les a affichées dans la console.



Nous pouvons également voir que les données ont bien été enregistrées dans le fichier csv.

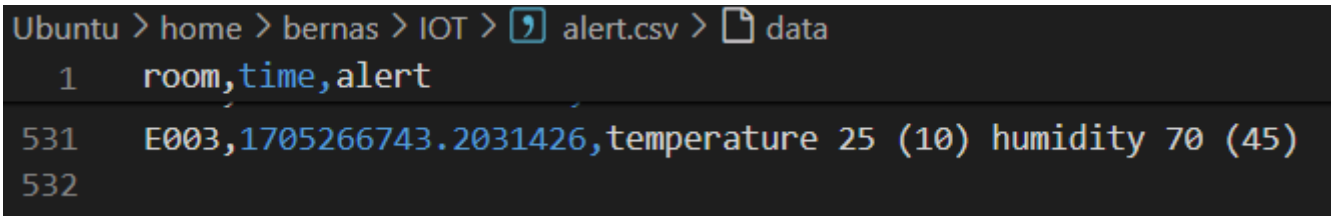
7.2. Test 2 : Réception des données d'alertes

Pour ce deuxième test, nous allons modifier les seuils d'alertes dans le fichier de configuration pour que l'application reçoive des données d'alertes.

```
bernas@B-LAPTOP:~/IOT$ python3 sae-iot.py
~ Retrieving configuration file
~ Selected data : ['temperature', 'humidity', 'co2', 'activity', 'tvoc', 'illumination', 'infrared', 'infrared_and_visible', 'pressure']
Exécution pendant 600 secondes...
~ Connected with result code 0
~ Subscribed to AM107/by-room/E003/data
~ Saving data on : data.csv
~ Waiting for data

[E003] temperature: 25, humidity: 70, co2: 442, activity: 0, tvoc: 249, illumination: 1, infrared: 1, infrared_and_visible: 1, pressure: 989.2, time: 1705266743.2031426
Threshold exceeded - temperature: 25 (Threshold: 10)
Threshold exceeded - humidity: 70 (Threshold: 45)
Moyenne des données de la pièce E003 :
temperature : 17.42 humidity : 49.92 co2 : 570.23 activity : 24.23 tvoc : 442.62 illumination : 10.92 infrared : 4.62 infrared_and_visible : 13.0 pressure : 995.32
```

Comme nous pouvons le voir sur l'image ci-dessus, l'application a bien reçu les données d'alertes du capteur de la salle E003 et les a affichées dans la console et en plus nous avons des données d'alertes affichées dans le terminal et dans le fichier csv d'alertes.



7.3. Test 3 : Réception des données anormales

Pour le troisième test, nous allons modifier le message reçu par l'application pour que celle-ci reçoive des données anormales.

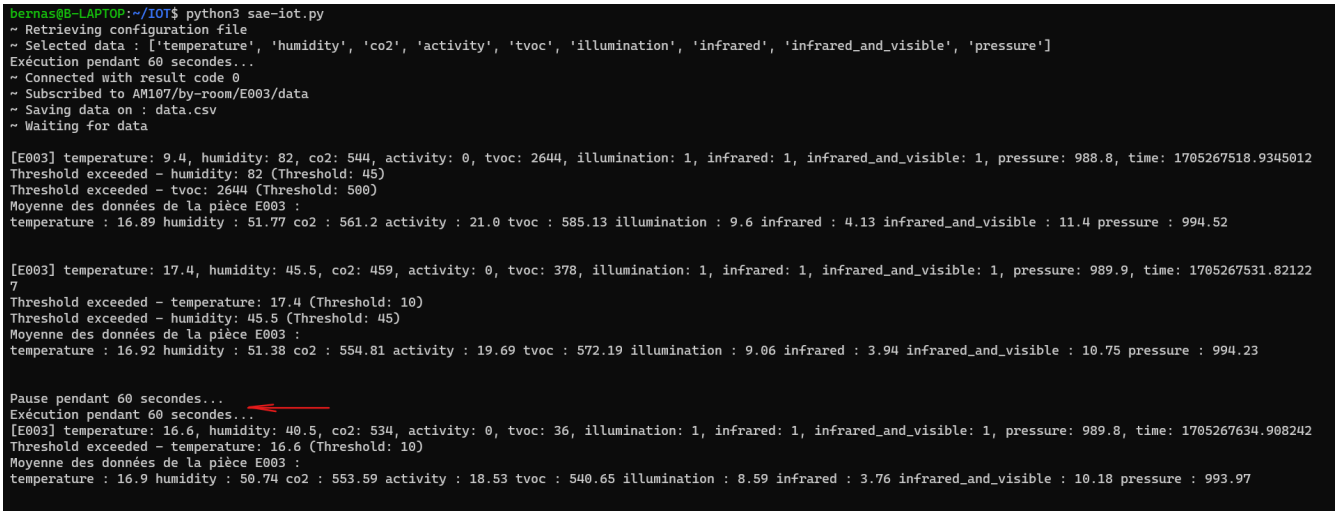
```
bernas@B-LAPTOP:~/IOT$ python3 sae-iot.py
~ Retrieving configuration file
~ Selected data : ['temperature', 'humidity', 'co2', 'activity', 'tvoc', 'illumination', 'infrared', 'infrared_and_visible', 'pressure']
Exécution pendant 600 secondes...
~ Connected with result code 0
~ Subscribed to AM107/by-room/E003/data
~ Saving data on : data.csv
~ Waiting for data

~ Erreur lors de la réception des données
```

Nous pouvons voir sur l'image ci-dessus que l'application a signalé que les données reçues étaient anormales.

7.4. Test 4 : Temps d'attente

Pour le quatrième test, nous allons modifier le temps d'attente entre chaque écoute pour que le temps d'attente soit reduit.



Nous pouvons voir sur l'image ci-dessus que l'application a bien effeccuté les écoutes avec une pause de 60 secondes.