

SAE S3.01 – Développement d'Application

Compétence 3 – Système

Compte Rendu

Documentation partie système

Sommaire

Structure du code	2
Architecture	2
Structure	2
Explications	3
Étapes d'installation	4
Étapes de lancement	5
Initialisation	5
Exécution du MQTT	5
Cas de tests	6

Structure du code

Architecture

```
solution IOT
├── application_iot
│   ├── application
│   │   ├── ressource
│   │   ├── data # Fichiers de données générés par l'application
│   │   │   ├── alert.csv      # Alerte de seuils dépassés
│   │   │   ├── data.csv      # Données agrégées des capteurs
│   │   │   ├── B002.csv      # Données salle B002
│   │   │   ├── B106.csv      # Données salle B106
│   │   │   ├── C004.csv      # Données salle C004
│   │   └── mqtt.py           # Configuration MQTT
```

Structure

Importations des bibliothèques

Les bibliothèques Python nécessaires au fonctionnement du projet sont :

- **json**, pour interpréter les messages JSON reçus via MQTT.
- **configparser**, pour lire les paramètres dynamiques stockés dans *configuration.ini*.
- **csv**, pour créer et manipuler les fichiers CSV contenant les données des capteurs.
- **paho.mqtt.client**, pour gérer la communication avec le broker MQTT.
- **logging**, pour consigner les erreurs et informations utiles dans le terminal.

Configuration

On commence par charger un fichier de configuration INI nommé *configuration.ini* en utilisant **configparser.ConfigParser()**.

Puis on récupère les paramètres suivants : le broker MQTT, le port, les topics, et les seuils de capteurs à surveiller.

Initialisation des fichiers CSV

Les fichiers CSV sont initialisés grâce à **clear_csv_files()** pour garantir que les fichiers de données sont vides et prêts à recevoir de nouvelles informations.

Gestion des événements MQTT

Les événements MQTT sont gérés via deux méthodes principales :

- **on_connect**, qui établit une connexion avec le broker et s'abonne aux topics.
- **on_message**, qui traite les messages reçus et met à jour les fichiers CSV en conséquence

Connexion MQTT

Le client MQTT est initialisé, configuré, et mis en boucle infinie pour écouter continuellement les messages entrants.

Explications

La méthode clear_csv_files() :

- **Objectif** : Nettoyer et réinitialiser les fichiers CSV.
- **Étapes** :
 - ◆ Réinitialise le fichier principal data.csv contenant les données agrégées pour toutes les salles.
 - ◆ Réinitialise le fichier alert.csv pour stocker les alertes déclenchées lorsque des seuils sont dépassés.
 - ◆ Réinitialise les fichiers individuels de chaque salle en fonction des sujets définis dans le fichier de configuration.
- **Résultat attendu** : Les fichiers CSV sont vidés et prêts à être remplis avec de nouvelles données.

La méthode on_connect(client, userdata, flags, reason_code, properties=None) :

- **Objectif** : Gérer l'événement de connexion au broker MQTT.
- **Étapes** :
 - ◆ Loggue un message indiquant si la connexion a réussi.
 - ◆ Parcourt tous les topics définis dans la configuration.
 - ◆ S'abonne à chaque topic (ou sous-arbre de topics).
- **Particularité** : La version actuelle est configurée pour s'abonner au topic racine avec # (global).

La méthode on_message(client, userdata, message) :

- **Objectif** : Traiter les messages reçus sur les topics MQTT.
- **Étapes** :
 - Décoder les données JSON du message reçu.
 - Extraire les données spécifiques d'une salle, comme :
 - Température
 - Humidité
 - Taux de CO2
 - TVOC (composés organiques volatils)
 - Infrarouge & Visible (infrarouge et visible)
 - Affiche les données sur la console.
 - Mise à jour des fichiers CSV :
 - Met à jour le fichier data.csv (agrégé).
 - Ajoute une nouvelle entrée dans le fichier spécifique à la salle.
 - Vérification des seuils :

- Si une valeur dépasse un seuil défini, une alerte est ajoutée au fichier alert.csv.

→ **Gestion des erreurs :**

- ◆ Gère les exceptions si les données JSON sont mal formées ou si des clés attendues manquent.

Connexion MQTT et boucle principale :

Le client MQTT est **initialisé** via `paho.mqtt.client` puis **configuré** avec les paramètres fournis dans *configuration.ini*. Les fonctions `on_connect` et `on_message` sont **associées** aux événements du client. Une fois la connexion établie, le programme reste actif et maintient une écoute continue des messages avec le **lancement** de `loop_forever()`

Détails des fichiers manipulés :

data.csv contient les données agrégées pour toutes les salles. Les colonnes principales sont : room, temperature, humidity, co2, tvoc, infrared_and_visible.

alert.csv contient les alertes déclenchées lorsque des seuils sont dépassés. Les colonnes sont : room, dataType, threshold, measuredValue.

Les fichiers par salle ({sujet}.csv) contiennent les données spécifiques à chaque salle.

Résultat global attendu

- Flux des données :
 - Les capteurs envoient des messages au broker MQTT.
 - Le programme récupère ces messages, les traite, et les sauvegarde dans des fichiers CSV.
 - Détecte et consigne toute anomalie dépassant les seuils.
- Structure extensible :
 - Peut facilement s'adapter à de nouveaux seuils ou types de capteurs en mettant à jour la configuration.

Étapes d'installation

1. Récupérer le code

Clonez le projet depuis le repository GitHub

2 .Installer les bibliothèques nécessaires :

```
pip install paho-mqtt
```

```
pip install configparser
```

```
pip install logging
```

Étapes de lancement

Initialisation

Vérification de l'os:

- On récupère l'OS de l'utilisateur pour adapter les paramètres suivant :
 - config_file_path : Récupère le chemin pour avoir accès au fichier de configuration
 - data_files_path : Récupère le chemin pour avoir accès aux fichiers de data

Nettoyage des anciennes données :

- Si des fichier csv de précédentes utilisations sont présent dans les dossiers :
 - Supprime les données présent dans data.csv, alerte.csv ainsi que tous les csv associées au salles sélectionné dans le fichier configuration.ini
 -

Lecture du fichier de configuration

- On récupères les information suivantes dans le fichier configuration.ini
 - Le broker MQTT
 - Le port de connexion MQTT
 - Les topics du MQTT ciblés
 - Les seuils d'alertes à ne pas dépasser.

Exécution du MQTT

Connexion au broker :

- Le client MQTT utilise les paramètre récupéré du fichier configuration
- Affichage d'un message de confirmation dans le terminale

Abonnement au topics : Le client MQTT s'abonne au topics sélectionné dans le fichier de configuration.

Initialisation des fichiers CSV

- Les fichiers CSV suivant sont créés au lancement du code afin de stockées les données demandé par le fichier de configuration
 - data.csv : Contient les données de toutes les salles en temps réel
 - alert.csv : Contient toute les alertes déclenchés en cas de dépassement de seuils
 - "Nom de salle demander".csv : contient l'historique de données de chaque salle
- Affiche du débogage
 - Logging.info() : Print dans la console

Traitement du Json :

- Analyse et traite les données reçus
- Affiches les données de façons plus lisibles dans le terminal
- Met à jour les données des différents CSV :
 - data.csv : Ajoute / Remplace les données pour la salle en question
 - [room].csv : Ajout dans le fichier spécifique à la salle une nouvelles lignes avec les données reçus (une sorte de log par salle)
- Lors du dépassement d'un seuil pour une donnée, une nouvelle ligne s'ajoute dans le fichier alert.csv

Répétitions : Le programme Loop sur la partie exécutions afin de toujours récupérer les dernières données du broker et les traités.

Cas de tests

Connexion MQTT :

- Vérifiez que le programme se connecte avec succès au serveur MQTT.
- Connexion au topic :

```
o : Success
INFO:root:Subscribing to topic: AM107/by-room/B111/#
INFO:root:Subscribing to topic: AM107/by-room/B003/#
INFO:root:Subscribing to topic: AM107/by-room/B201/#
INFO:root:Subscribing to topic: AM107/by-room/B002/#
INFO:root:Subscribing to topic: AM107/by-room/B110/#
INFO:root:Subscribing to topic: AM107/by-room/B001/#
```

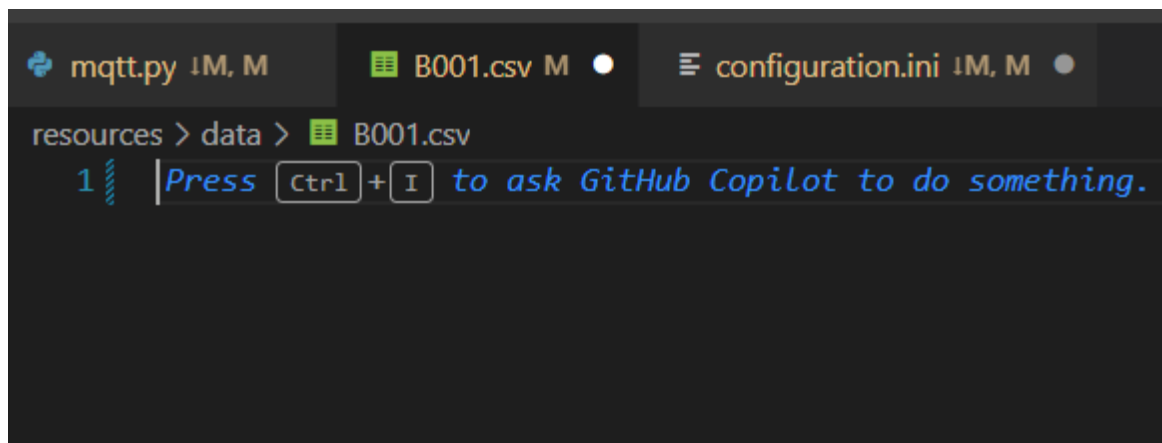
- Réception des données dans le terminale :

```
INFO:root:MQTT script started.  
INFO:root:Connecting to MQTT broker at mqtt.iut-blagnac.fr:1883  
INFO:root:Connected with result code : Success
```

```
INFO:root:Message received on topic: AM107/by-room/B001/data  
Salle : B001  
co2 : 487  
activity : 0  
humidity : 38  
temperature : 21
```

- Vérifiez que les fichiers CSV sont remplis correctement.

Csv de la salle b001 avant réception est vide :



The screenshot shows a code editor with three tabs: 'mqtt.py', 'B001.csv', and 'configuration.ini'. The 'B001.csv' tab is active, showing a file path 'resources > data > B001.csv'. The file content is empty, with a cursor at line 1. A GitHub Copilot suggestion is visible: 'Press Ctrl + I to ask GitHub Copilot to do something.'

Ecriture dans le csv une fois la données reçus :

```
resources > data > B001.csv > data  
1 room;co2;activity;humidity;temperature;  
2 B001;487;0;38;21  
3
```

Alerte de seuil :

Afin de détecter un seuil, je met le seuil de température à 5 degrés :

```
[THRESHOLD]
activity=100.0
co2=1000.0
temperature=5.0
humidity=80.0
```

Fichier CSV avant que la réception de données avec un seuil dépassé :

```
resources > data > alert.csv > data
room;dataType;threshold;measuredValue
```

Réception de la données avec un seuil dépassé :

Dans le terminal

```
Salle : hall-amphi
co2 : 761
activity : 32
humidity : 42.5
temperature : 19.1
```

Dans le csv alert.csv

```
resources > data > alert.csv > data
1 room;dataType;threshold;measuredValue
2 hall-amphi;temperature;5.0;19.1
3
```