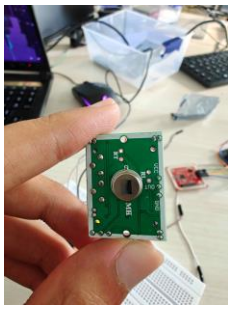
Projet IoT : Système de Contrôle RFID et Surveillance Température/Humidité via MQTT

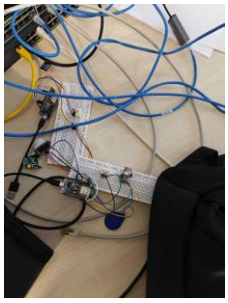
Introduction

Ce projet vise à développer un système de contrôle d'accès basé sur la technologie RFID (Identification par Radiofréquence) et à surveiller l'environnement d'une salle en termes de température, humidité et présence de fumée à travers l'utilisation de capteurs IoT. L'objectif est d'intégrer ces capteurs à un serveur central via le protocole MQTT pour suivre en temps réel les données sur une interface web.

Matériel et Logiciel Utilisés



```
grp6@raspberrypi:~/hospital-iot-backend $ sudo node server.js
Serveur démarré sur le port 80
Connecté au broker MQTT
Client connecté: 9BtU-OEZ9pDYsH79AAAB
Message reçu sur le topic hospital/temperature: 23.5
Température : 23.5°C
Client déconnecté: 9BtU-OEZ9pDYsH79AAAB
Client connecté: XPgv8cXdeMKD-DxMAAAD
```



Le matériel utilisé pour ce projet comprend :
- Module RFID PN532 pour lire les badges d'accès
- Capteurs DHT22 pour mesurer la température et l'humidité
- Capteur MQ135 pour mesurer la fumée/qualité de l'air
- ESP8266 pour gérer la communication Wi-Fi et l'interaction avec les capteurs
- Un serveur MQTT pour la communication avec les capteurs

Logiciel :
- Arduino IDE pour programmer l'ESP8266
- Node.js pour le backend serveur
- Socket.io pour la communication en temps réel
- Chart.js pour afficher graphiquement les données sur le front-end

Description du Projet

Le projet se décompose en plusieurs parties :
- Lecture des badges RFID : Les badges permettent d'identifier l'accès à une zone sécurisée. Chaque badge possède un UID unique qui est comparé avec les UIDs stockés sur le serveur. Si le badge est valide, l'accès est accordé.
- Surveillance environnementale : À l'aide des capteurs, le système surveille en temps réel les niveaux de température, d'humidité et de fumée dans une pièce. Ces données sont envoyées au serveur via MQTT et sont ensuite affichées sur une interface web.
- Interface utilisateur : Les données collectées sont envoyées à un serveur Node.js qui publie les données à l'interface utilisateur via Socket.io. L'interface est construite avec Chart.js pour afficher les courbes de température, humidité, et niveaux de fumée.

Declaration des information Wi-Fi et MQTT

```
const char* ssid = "WIFI-H6-IOT";

const char* password = "iotWIFI25!";

const char* mqtt_server = "192.168.80.1";

const int mqtt_port = 1883;

const char* mqtt_user = "iot6";

const char* mqtt_password = "tpRT9025";
```

-SSID et password : Les identifiants Wi-Fi pour connecter l'ESP8266 au réseau.

-MQTT server et port : L'IP du broker MQTT et le port utilisé (par défaut 1883).

-MQTT user et password : Les informations de connexion pour accéder au broker MQTT.

5. Déclaration des UID des badges RFID :

uint8_t adminUID[] = {0xC3, 0x95, 0x09, 0x2A}; // Badge admin

uint8_t personnelUID[] = {0x53, 0xAA, 0x86, 0x13}; // Badge personnel

adminUID et personnelUID : Ce sont les UID (Unique Identifiers) des badges RFID pour l'administrateur et le personnel.

Difficultés Rencontrées

Au cours du projet, plusieurs défis techniques ont été rencontrés :
1. Problèmes de connexion Wi-Fi : L'ESP8266 a parfois eu du mal à se connecter au réseau Wi-Fi, ce qui a retardé les tests.
2. Configuration du broker MQTT : Il a fallu configurer correctement le broker MQTT afin d'assurer que tous les messages soient bien reçus et envoyés aux bons topics.
3. Synchronisation des données : La gestion de la latence entre l'envoi des données par les capteurs et leur réception par le serveur a nécessité plusieurs ajustements dans la configuration de Socket.io.
4. Limitations de la puissance de l'ESP8266 : Le traitement simultané de plusieurs capteurs a parfois causé des problèmes de mémoire et des ralentissements sur l'ESP8266.

Conclusion

Ce projet a permis de mettre en place un système de surveillance d'un environnement hospitalier en temps réel et d'accéder à des zones sécurisées via des badges RFID. Le protocole MQTT s'est avéré efficace pour gérer la communication entre les capteurs et le serveur central. Des optimisations futures pourraient inclure une meilleure gestion des ressources de l'ESP8266 et l'ajout de nouvelles fonctionnalités, telles que l'envoi d'alertes par email en cas de dépassement de seuils critiques.