Documentation du projet

1. Installation du système

Le projet a pour objectif la surveillance de la température à l'aide d'un capteur connecté à un ESP32, avec transmission des données vers un Raspberry Pi via le protocole MQTT. Les données sont ensuite visualisées et stockées grâce à Node-RED et une base de données SQLite.

1.1 Matériel utilisé

Le système est composé des éléments suivants :

- Un microcontrôleur **ESP32**
- Un **capteur de température LM35**
- Un **Raspberry Pi**
- Un ordinateur pour le développement et la configuration

1.2 Logiciels utilisés

Les logiciels et outils suivants ont été utilisés :

- **Arduino IDE** pour le développement du programme ESP32
- **Mosquitto** comme broker MQTT
- **Node-RED** pour la gestion des flux et l'interface graphique
- **SQLite** pour le stockage des données
- **GitHub** pour la gestion de version et la documentation

2. Configuration du système

2.1 Configuration de l'ESP32

L'ESP32 est configuré pour :

- Se connecter à un réseau Lora
- Lire la température fournie par le capteur LM35
- Convertir la valeur analogique en degrés Celsius
- Envoyer périodiquement les données de température via MQTT

- Recevoir des commandes MQTT (changement de seuil, activation de LED, modification de la période d'envoi)

2.2 Configuration MQTT

Le protocole MQTT est utilisé pour la communication entre l'ESP32 et le Raspberry Pi. Les topics principaux utilisés sont :

- Un topic pour l'envoi des données de température
- Un topic pour la configuration des paramètres
- Un topic pour les alertes

Le broker Mosquitto agit comme intermédiaire entre les différents composants du système.

2.3 Configuration Node-RED

Node-RED est configuré pour :

- S'abonner aux topics MQTT
- Afficher la température en temps réel sur un tableau de bord
- Générer des graphiques d'évolution
- Stocker les données dans une base SQLite
- Déclencher des alertes lorsque la température dépasse un seuil défini

3. Fonctionnement du système

Le fonctionnement global du projet est le suivant :

1. Le capteur LM35 mesure la température ambiante.
2. L'ESP32 récupère la valeur et la traite.
3. Les données sont envoyées via MQTT au Raspberry Pi.
4. Le broker Mosquitto transmet les données aux clients abonnés.
5. Node-RED affiche les données en temps réel et les enregistre dans la base de données.
6. En cas de dépassement du seuil, une alerte est générée et une action peut être déclenchée (LED, message, etc.).

Ce fonctionnement permet une surveillance continue et à distance de la température.

4. Architecture du projet

Le projet repose sur une architecture de type **client-serveur**.

- L'**ESP32** agit comme client MQTT et capteur de données.
- Le **Raspberry Pi** joue le rôle de serveur central avec le broker MQTT.
- **Node-RED** assure la visualisation et la gestion logique.
- **SQLite** stocke les données pour un suivi dans le temps.

Cette architecture permet une séparation claire des rôles et facilite l'évolution du projet.

5. Guide de dépannage

Problème 1 : aucune donnée affichée

Cause possible :

- Problème de connexion Lora
- Mauvais topic MQTT
- Broker MQTT non démarré

Solution :

- Vérifier la connexion réseau
- Vérifier les topics configurés
- Redémarrer le service Mosquitto

Problème 2 : données non enregistrées

Cause possible :

- Base de données inexistante
- Mauvaise configuration SQLite

Solution :

- Vérifier l’existence de la base de données
- Vérifier les permissions et la configuration Node-RED