Remote Garden Monitor IoT Visualizer and Controller Writeup

Team Members

- Tingqi (Ting) Wang (twang356@usc.edu)
- Jeremiah Lim (limjerem@usc.edu)

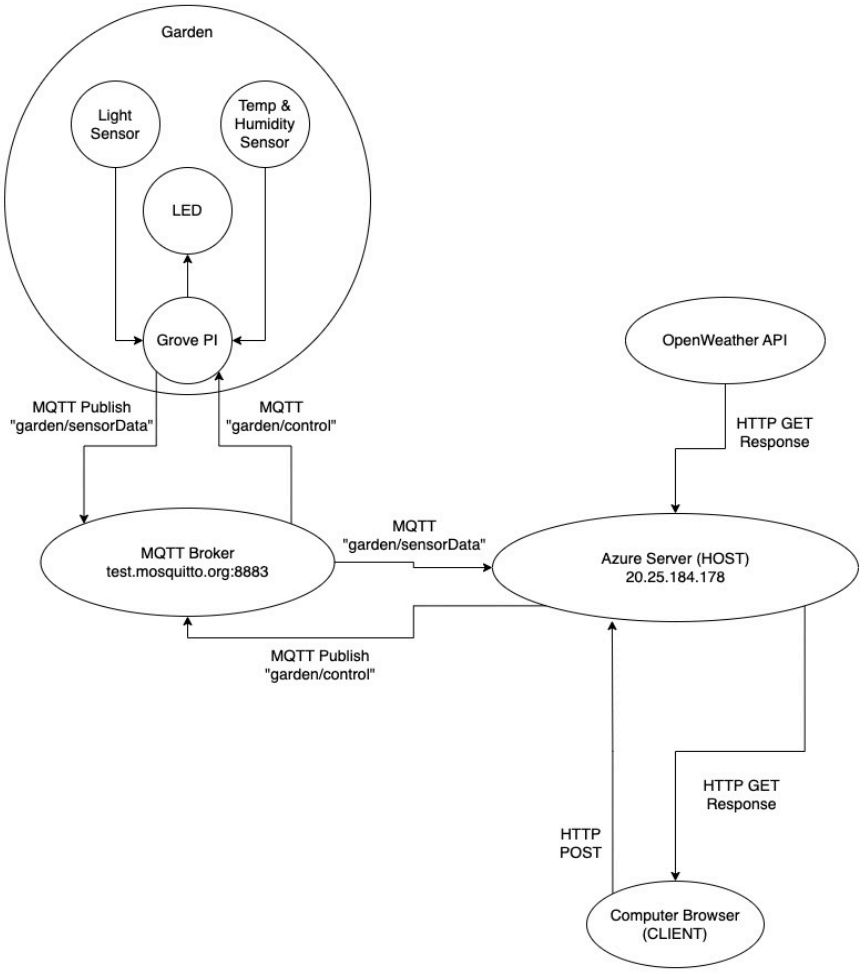
Project Github Repo: https://github.com/TQW0909/EE250_Project.git

Description

Our project utilizes the GrovePI sensors on a RaspberryPi to track the temperature, humidity, and light data regarding a garden, and is complemented by weather data (current weather condition and cloudiness percentage) fetched from the OpenWeather API. The data is communicated between the Raspberry Pi and our server hosted on Azure through encrypted MQTT.

The project provides a user interface accessible through a web browser to visualize all the collected data in real-time. In addition, we have also incorporated a controller feature to allow the user to water the garden with a click of a button on the web page.

Diagram



Component and Implementation Description

- **Flask Web Framework**: Used to create the server for handling requests and serving data.
- **OpenWeatherMap API**: Fetches weather data based on ZIP codes.
- **MQTT Protocol**: RasberryPi publishes sensor data to and receives control messages from an MQTT broker hosted on test.mosquitto.org over port 8883 through encrypted communication.
- **Control Actions**: RaspberryPi responds to received control commands from the server, like turning on an LED.
- **Threading**: Implemented a background thread for periodic weather data updates.
- **HTTP**: Used to fetch weather data from the OpenWeatherMap API, as well as to interact with the Azure server on a web browser.
- **AJAX**: Used on the web page to allow for dynamic updates of collected data.
- **Azure Server (20.25.184.178):** This cloud server receives sensor data and weather information, providing a platform for data processing and hosting the web interface.
- **GrovePi Sensors**: Connects to temperature, humidity, and light sensor for real-time environmental data collection.
- **Modular Architecture**: Each script serves a specific function, making the system easy to maintain and scale.
- **Real-Time Communication**: Uses MQTT, a lightweight protocol ideal for our IoT application.
- **Security**: Incorporates encrypted secure data transmission in MQTT by utilizing the broker's certificate.
- **Data Handling**: Employs JSON for easy data representation and exchange.
- **Python Libraries**: Flask for the web server, requests for HTTP requests, paho-mqtt for MQTT communication, GrovePi for sensor interaction.

Reflection

One limitation that we had in the project was not being able to actually implement a watering actuator using a water pump to water the garden, instead we only had a LED as a virtualization of such a mechanism.

The web front that is used to display the sensor data and send the watering command to the Rpi was not aesthetically pleasing and was very plain and basic. The reason being the limited time we had and the lack of experience with HTML and CSS.

We could have also utilized ML on the server to process the incoming data (temperature, humidity, light intensity, and cloudiness) and automatically issue watering commands based on a trained model to maximize plant growth in the garden. But we did not have enough time to implement this feature.

A lesson that we have learned from this project is how to use threads on the server to periodically fetch data from an API in the background to have real-time data available to display to the user.