Smart Greenhouse Monitoring System

Department of Information Engineering, Computer Science and Mathematics

University Of L'Aquila, Italy

Professor Davide Di Ruscio

Team Members

Alaina Faisal
Melissa Puerto Aguayo
Sarosh Krishan

Contents

0.1 Introduction

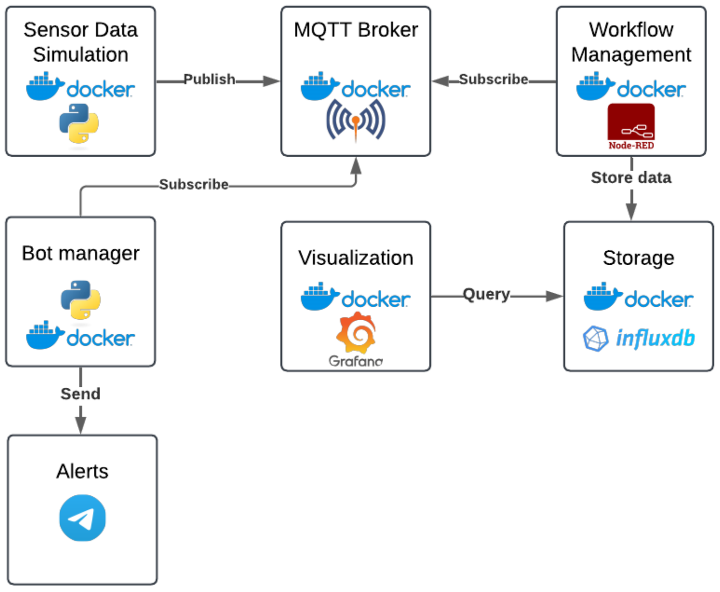
The Smart Greenhouse Monitoring System utilizes Internet of Things (IoT) technology to create an automated and efficient environment for plant growth. By integrating seven key sensors (temperature, pH, humidity, CO2, water level, soil moisture, and light intensity), this system provides real-time insights into environmental conditions, ensuring that each greenhouse maintains the optimal microclimate for its plants. The project leverages advanced data visualization, storage, and alerting systems to make monitoring and decision-making more accessible and proactive.

0.2 Objectives

The objectives of the Smart Greenhouse Monitoring System are outlined below:

- To integrate IoT sensors for real-time monitoring of key environmental parameters such as temperature, humidity, CO_2, pH levels, water level, soil moisture, and light intensity.
- To enable efficient data collection and storage for analysis and decision-making.
- To provide user-friendly visualizations through Grafana dashboards for easy monitoring and trend identification.
- To promote sustainable agriculture by optimizing resource use and improving plant health.

0.3 System Architecture



The system integrates multiple IoT technologies for seamless operation:

- **Node-RED:** Manages data flow between IoT sensors, MQTT broker, and the database.
- **MQTT (Mosquitto):** Facilitates lightweight communication between sensors and processing units.
- **InfluxDB:** Stores high-frequency time-series data collected from the sensors.
- **Grafana:** Visualizes real-time and historical data in interactive dashboards.
- **Telegram:** Provides real time alerts in case a value is above or below the decided threshold.

The system ensures:

- Real-time monitoring across multiple greenhouses.
- Efficient data collection and storage for future analysis.
- Automated alerts for immediate corrective actions.

0.4 System Functionality

The Smart Greenhouse system performs the following key functions:

1. **Data Collection:** Sensors continuously collect data on environmental parameters across greenhouses.
2. **Data Processing:** Node-RED processes incoming MQTT messages and routes them to the database.
3. **Data Storage:** InfluxDB stores data for historical analysis and visualization.
4. **Data Visualization:** Grafana dashboards display real-time readings and historical trends.
5. **Alerting:** Threshold-based alerts are triggered when sensor values exceed safe ranges.

0.5 Technologies Used

The Smart Greenhouse system integrates the following technologies:

0.5.1 Python

Python scripts are used for data preprocessing and simulating sensor input for testing purposes.

0.5.2 Node-RED

Node-RED handles the real-time flow of data between the MQTT broker, sensors, and database. Each sensor publishes its data to a specific topic, and Node-RED processes this data for visualization or alerting.

0.5.3 MQTT (Mosquitto)

Mosquitto serves as the MQTT broker, enabling lightweight and reliable communication between sensors and the backend. The system integrates the following seven sensors to monitor critical environmental parameters:

- **Temperature Sensor:** Measures air temperature to ensure optimal conditions for plant growth.
- **Humidity Sensor:** Tracks ambient moisture levels to maintain balance in both arid and tropical environments.
- **CO_2 Sensor:** Monitors carbon dioxide levels to optimize photosynthesis.
- **Light Intensity Sensor:** Measures sunlight and artificial light levels to ensure plants receive adequate illumination.
- **Soil Moisture Sensor:** Tracks water content in the soil to avoid under- or over-watering.
- **Water Level Sensor:** Monitors water availability in reservoirs for irrigation systems.
- **pH Sensor:** Measures soil and water pH levels to maintain a balanced growing medium.

These sensors provide comprehensive, real-time data necessary for maintaining optimal greenhouse conditions.

Habitats

The system is designed to operate in two primary habitats:

1. **Arid Habitat:** Simulates a dry environment with controlled water usage and temperature adjustments, suitable for plants adapted to desert-like conditions.
2. **Tropical Habitat:** Mimics a humid and warm climate, ideal for plants that thrive in rainforest-like environments.

The decision to incorporate two habitats allows for flexibility in growing a diverse range of plant species under controlled conditions, catering to specific environmental requirements.

Greenhouses

The system spans **three greenhouses**, each designed to specialize in distinct crops or experiments:

- Allows partitioning for specific plant types, ensuring that different crops with unique needs do not interfere with each other.
- Provides scalability for research and production, enabling simultaneous experiments or production cycles.
- Reduces risk by isolating potential issues like pest infestations or environmental imbalances to one greenhouse.

The inclusion of multiple sensors, habitats, and greenhouses is motivated by the following factors:

- **Comprehensive Monitoring:** Seven sensors ensure all critical environmental parameters are tracked and managed efficiently.
- **Flexibility in Environmental Control:** Two habitats provide the ability to simulate contrasting climates, enabling broader agricultural research and applications.
- **Scalability and Risk Mitigation:** Three greenhouses allow for modular expansion, simultaneous experiments, and containment of localized issues.
- **Sustainability:** By tailoring conditions to specific habitats, resource usage (e.g., water, light) is optimized, minimizing waste and environmental impact.

0.5.4 InfluxDB

influxdb

InfluxDB is used for efficient storage of time-series data, allowing for both real-time and historical analysis.

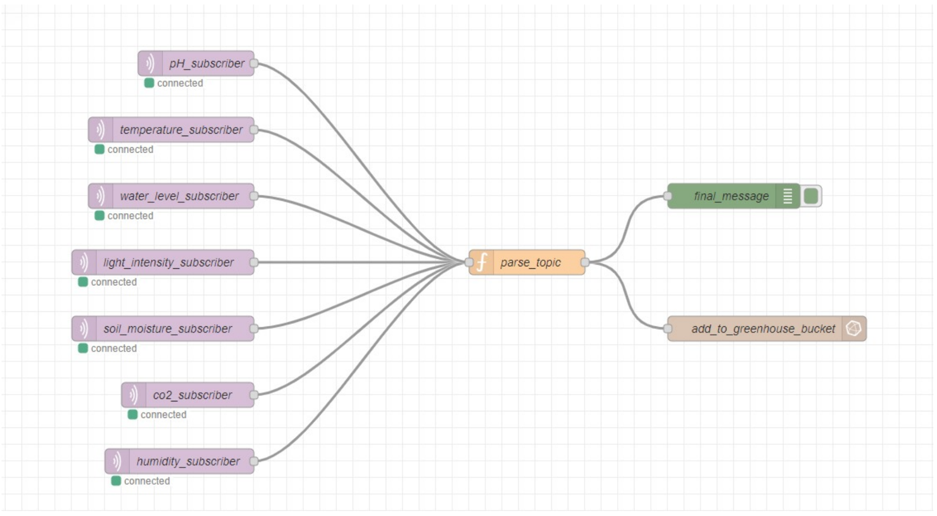
0.5.5 Grafana

Grafana

Grafana provides customizable dashboards for visualizing sensor data. Key features include:

- Real-time gauges for each sensor.
- Time-series graphs for tracking historical trends.
- Configurable alerts with visual thresholds.

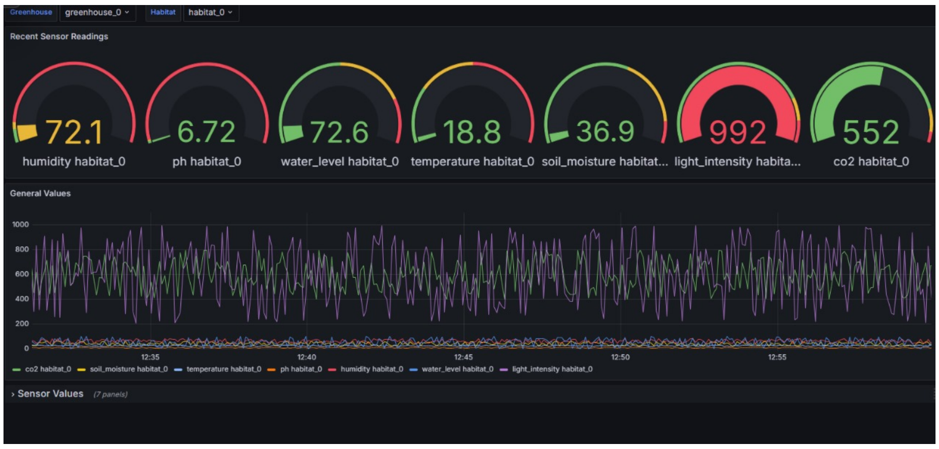
0.6 Node-RED Workflow



The Node-RED workflow processes data streams in real-time:

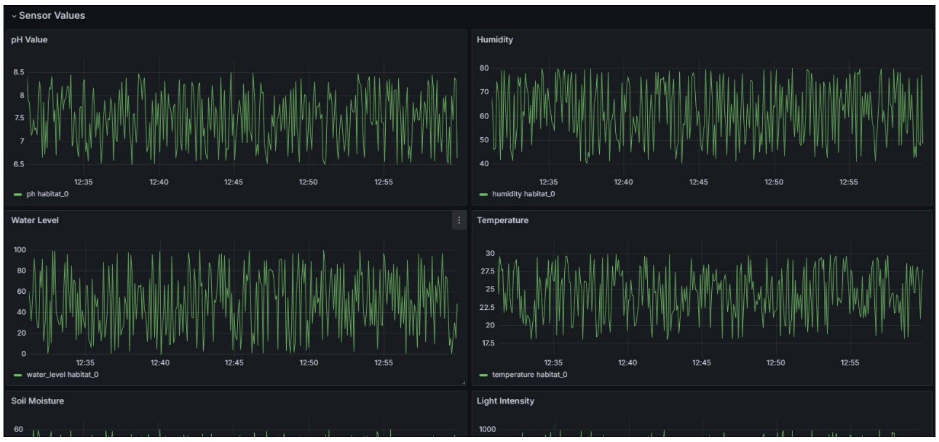
- **Inputs:** Each sensor (e.g., temperature, humidity, CO_2) sends data to specific MQTT topics.
- **Processing:** A ’parse_topic’ function processes the incoming data streams and filters them.
- **Outputs:**
    1. Data is stored in InfluxDB for long-term analysis.

0.7 Grafana Dashboard



The Grafana dashboard provides a comprehensive visualization of greenhouse data:

- **Gauge Panels:** Real-time sensor readings, with thresholds color-coded for clarity.
- **Time-Series Graphs:** Historical trends to identify patterns and anomalies.
- **Multi-Greenhouse Monitoring:** Support for data visualization across different greenhouses and habitats.

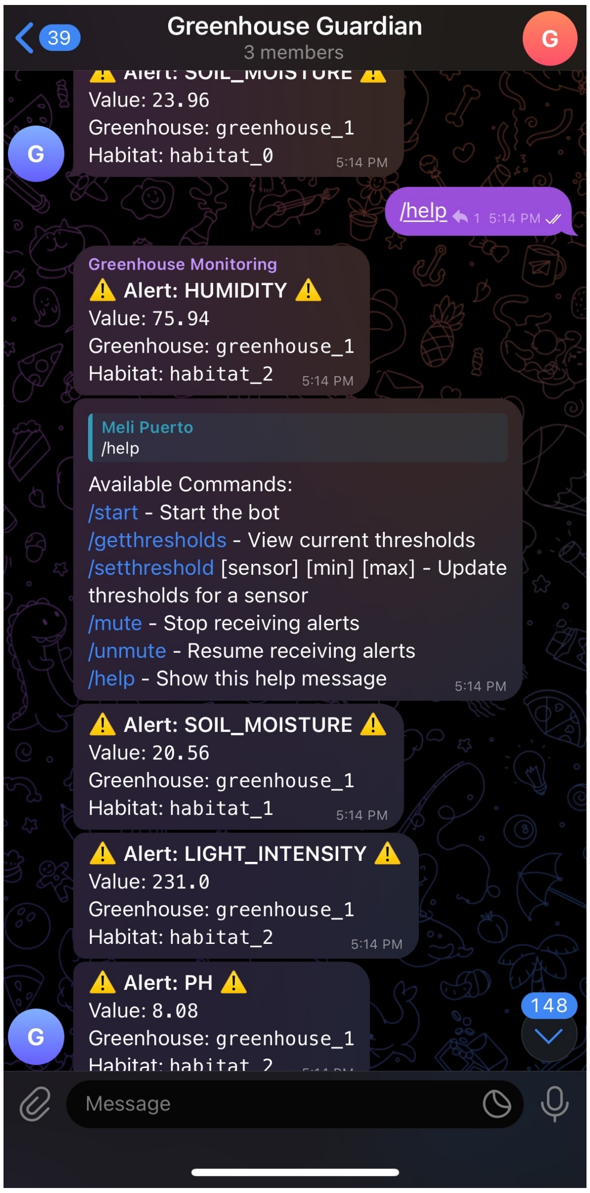


0.8 Telegram Alert System

0.8.1 Data Acquisition:

The bot subscribes to an MQTT broker using the Paho MQTT client. Sensor data from the greenhouse, such as temperature, humidity, and soil moisture, is published to the broker under topic hierarchies (e.g., 'sensor_type/greenhouse_id/habitat_id'). The bot listens to all topics ('#' wildcard) to capture every incoming message, ensuring comprehensive monitoring.

Each message contains key-value pairs in JSON format, specifying sensor readings. This data is decoded and categorized by the bot to identify the sensor type, the greenhouse, and the habitat from which it originated. For instance, a message from the topic 'temperature/greenhouse1/habitatA' with a payload '"temperature": 30' would indicate a reading of 30°C from habitat A in greenhouse 1.



```
⚠️ Alert: SOIL_MOISTURE ⚠️
Value: 23.96
Greenhouse: greenhouse_1
Habitat: habitat_0        5:14 PM

/help  1  5:14 PM

Greenhouse Monitoring
⚠️ Alert: HUMIDITY ⚠️
Value: 75.94
Greenhouse: greenhouse_1
Habitat: habitat_2        5:14 PM

Meli Puerto
/help
Available Commands:
/start - Start the bot
/getthresholds - View current thresholds
/setthreshold [sensor] [min] [max] - Update thresholds for a sensor
/mute - Stop receiving alerts
/unmute - Resume receiving alerts
/help - Show this help message        5:14 PM

⚠️ Alert: SOIL_MOISTURE ⚠️
Value: 20.56
Greenhouse: greenhouse_1
Habitat: habitat_1        5:14 PM

⚠️ Alert: LIGHT_INTENSITY ⚠️
Value: 231.0
Greenhouse: greenhouse_1
Habitat: habitat_2        5:14 PM

⚠️ Alert: PH ⚠️
Value: 8.08
Greenhouse: greenhouse_1
Habitat: habitat_2
```

0.8.2 Threshold Evaluation:

The bot evaluates sensor readings against preconfigured thresholds stored in the 'thresholds.json' file. This file defines acceptable ranges for each monitored parameter, such as temperature ('min: 18°C, max: 28°C') or soil moisture ('min: 30%, max: 55%'). If a value exceeds or falls below these thresholds, an alert is triggered. The system supports dynamic threshold updates via Telegram commands, ensuring adaptability to evolving greenhouse conditions.

0.8.3 User Notification:

When a threshold breach occurs, the bot formats an alert message using Markdown syntax and sends it to a designated Telegram chat. The message includes the sensor name, its value, and its origin (greenhouse and habitat). This notification system ensures that users are promptly informed of critical events. The bot also features a mute function, allowing users to temporarily disable alerts to avoid notification overload during specific periods.

0.8.4 Configuration with Environment Variables

To ensure security and flexibility, sensitive information and configuration details are managed through a '.env' file. The key variables in this file include:

- TOKEN: This variable stores the Telegram bot API token, enabling secure communication between the bot and Telegram servers. For this project, the token is '7799675291:AAHaX7G0SmnlMVTM
jVRM'.

- CHAT_ID: This specifies the target chat ID where alerts are sent. It can be a private chat ID or a group ID. In this case, the bot is configured to send messages to the chat with ID '-1002247342944'.

Using environment variables decouples sensitive data from the source code, enhancing security and making deployment across environments (e.g., development, testing, and production) more manageable.

0.8.5 Implementation Highlights:

The bot leverages the Telegram Bot API and Python's asyncio library for non-blocking operations. This ensures that the bot can process incoming sensor data and send notifications without delays. It is also capable of handling concurrent MQTT messages, making it suitable for larger-scale deployments with multiple greenhouses.

Command handlers are implemented to provide users with control over the bot. For example, '/setthreshold' allows the user to update sensor thresholds dynamically, while '/mute' and '/unmute' control the flow of alerts. These commands ensure that users have full oversight and customization capabilities.

0.9 Conclusion

The Smart Greenhouse Monitoring System demonstrates how IoT technology can revolutionize modern agriculture. By integrating sensors, real-time monitoring, and advanced visualization, this system ensures optimal growing conditions and proactive management of resources. Future enhancements could include predictive analytics and machine learning to further automate greenhouse management.

0.10 Instructions:

0.10.1 Steps to Run the Project

1. **Clone the Repository (If Not Already Cloned)**

   - Open a terminal and run:

     ```
     git clone https://github.com/Alaina358/GreenHouseIoT.git
     ```

2. **Build and Run Docker Services**

   - Run the following command to build the necessary Docker images:

     ```
     docker compose build
     ```

   - Start all the services:

     ```
     docker compose up
     ```