

IOT-BASED SMART FARMING SYSTEM



2024/25

Software Engineering for The Internet of Things

Prof. Davide Di Ruscio

Realized by:

Omar Dinari - omar.dinari@student.univaq.it

Contents

1. Introduction	3
2. Objectives	3
2.1. Real-Time Environmental Monitoring.....	3
2.2. Intelligent Alert System	3
2.3. Data Visualization	3
2.4. Scalable & Automated Solution	3
3. System Architecture & Data Flow	3
3.1. System Components.....	3
3.2. Data Flow.....	4
4. Functional and Non-Functional Requirements.....	4
4.1. Functional Requirements (FRs)	4
4.2. Non-Functional Requirements (NFRs)	5
5. Structured MQTT Topic Naming Convention	5
6. Sensor Data Publisher (Python MQTT Client)	5
6.1. Key Features	5
6.2. Sample MQTT Data Payload	5
6.3. Configuration File (<code>config.ini</code>)	5
7. Storing Data in InfluxDB using Telegraf	6
7.1. Telegraf Configuration (<code>telegraf.conf</code>).....	6
9. Node-Red and Telegram:	8
Create the Telegram Bot.....	8
10. Conclusion.....	8

1. Introduction

The **IoT-Based Smart Farming System** is designed to monitor environmental conditions in agricultural fields using IoT sensors. It collects real-time data on **temperature, humidity, soil moisture, and light intensity**, processes it via **MQTT**, stores it in **InfluxDB**, and visualizes it in **Grafana**. Alerts are sent via **Node-RED to Telegram** when values exceed predefined thresholds.

2. Objectives

2.1. Real-Time Environmental Monitoring

- Continuously track **temperature, humidity, soil moisture, and light intensity** across multiple farms.

2.2. Intelligent Alert System

- Notify farmers when conditions exceed safe thresholds to take immediate action.

2.3. Data Visualization

- Provide a **user-friendly Grafana dashboard** with real-time sensor graphs.

2.4. Scalable & Automated Solution

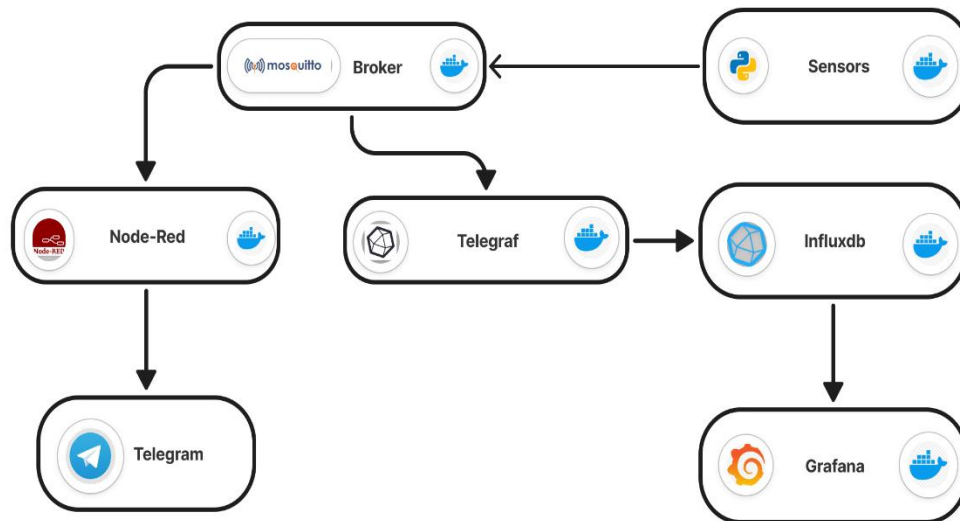
- Support multiple farms with **structured MQTT topics** and **automated data collection**.
-

3. System Architecture & Data Flow

3.1. System Components

1. **MQTT Broker (Mosquitto)** - Manages sensor data exchange between publishers and subscribers.
2. **Sensor Nodes (Python MQTT Publisher)** - Simulated sensors that publish environmental data via MQTT.
3. **Telegraf** - A lightweight agent that subscribes to MQTT topics and forwards data to InfluxDB.
4. **InfluxDB** - A time-series database used to store sensor readings for efficient querying.
5. **Grafana** - A visualization tool that displays real-time sensor data in customizable dashboards.
6. **Node-RED** - A flow-based tool for handling IoT data processing and triggering alerts.
7. **Telegram Bot** - A messaging service that sends notifications for abnormal sensor conditions.

3.2. Data Flow



1. **Sensor Nodes publish data** → MQTT Broker (**Mosquitto**)
2. **Telegraf subscribes to MQTT topics** → Stores data in **InfluxDB**
3. **Grafana queries InfluxDB** → Displays data visually
4. **Node-RED listens to MQTT** → Sends alerts if thresholds are exceeded
5. **Alerts are sent to Telegram** in case of critical conditions

4. Functional and Non-Functional Requirements

4.1. Functional Requirements (FRs)

1. **Real-Time Data Collection** - The system must collect and transmit sensor data in real-time.
2. **MQTT Communication** - Data exchange between sensors and the system must use MQTT.
3. **Data Storage in InfluxDB** - The system must store sensor readings in InfluxDB for historical analysis.
4. **Dashboard Visualization** - The system must provide a real-time dashboard using Grafana.
5. **Alerting Mechanism** - The system must send alerts when sensor values exceed thresholds.
6. **Multiple Farm Support** - The system must handle and differentiate data from multiple farms.

4.2. Non-Functional Requirements (NFRs)

1. **Scalability** - The system must support additional sensors and farms without performance degradation.
 2. **Reliability** - The system must ensure continuous operation with minimal downtime.
 3. **Security** - Data transmitted over MQTT must be secured to prevent unauthorized access.
 4. **Performance** - The system should process and store data efficiently to prevent delays.
 5. **Usability** - The dashboard and alert system should be user-friendly and easy to configure.
-

5. Structured MQTT Topic Naming Convention

To ensure scalability, MQTT topics follow a structured naming convention:

```
farming/farm_X/sensor_type
```

Example Topics

```
farming/farm_1/temperature
farming/farm_2/humidity
farming/farm_1/soil_moisture
farming/farm_2/light_intensity
```

6. Sensor Data Publisher (Python MQTT Client)

6.1. Key Features

- Generates **simulated** temperature, humidity, soil moisture, and light intensity data.
- Uses **multithreading** to simulate multiple farms.
- Publishes data in **JSON format** to structured MQTT topics.
- **Reconnects automatically** if MQTT broker is unavailable.

6.2. Sample MQTT Data Payload

```
{
  "temperature": 30.5,
  "farm": "farm_1"
}
```

6.3. Configuration File (`config.ini`)

```
[data_generation]
farms = 2
time_sleep = 5
sensors = temperature|humidity|soil_moisture|light_intensity
```

7. Storing Data in InfluxDB using Telegraf

7.1. Telegraf Configuration (`telegraf.conf`)

```
[[inputs.mqtt_consumer]]
  servers = ["tcp://mosquitto:1883"]
  topics = ["farming/+/"+]
  data_format = "json"

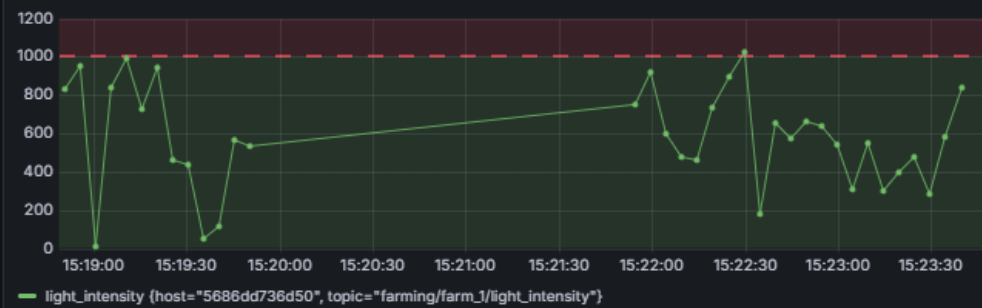
[[outputs.influxdb_v2]]
  urls = ["http://influxdb:8086"]
  token = "my-secret-token"
  organization = "my_org"
  bucket = "farming"
```

8. Explanation of Grafana Results

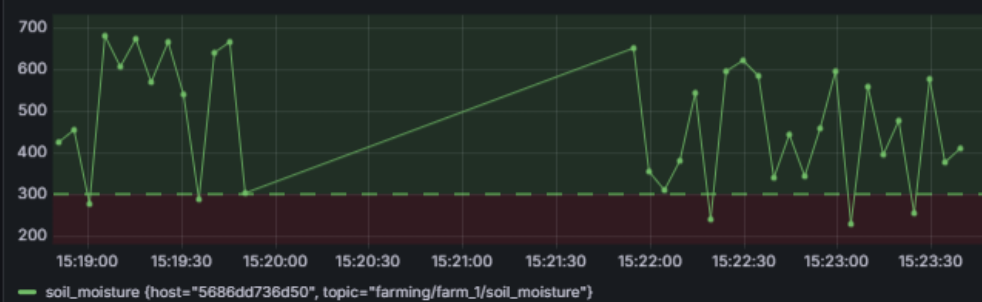
- You should see a **dropdown** in Grafana (e.g., "farm_1", "farm_2").
- When you select a farm, **all panels should update** to show data for that specific farm.
- This is done using **Grafana variables** (e.g., `$farm` for dynamic filtering).

Select farm farm_1 ▾

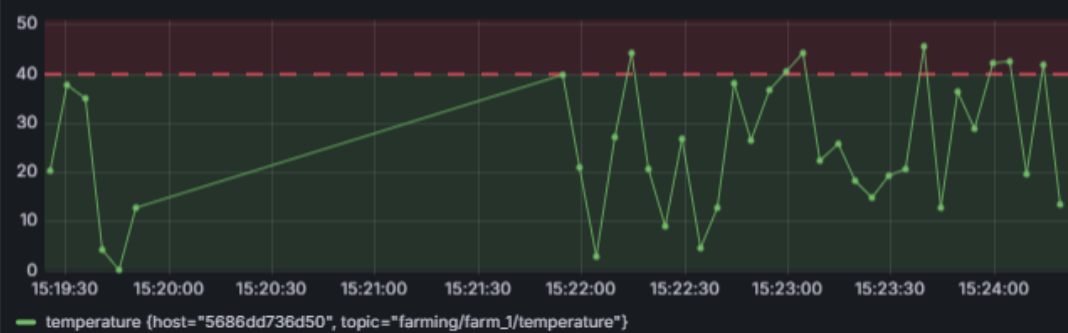
Light Intensity



Soil Moisture



Temperature

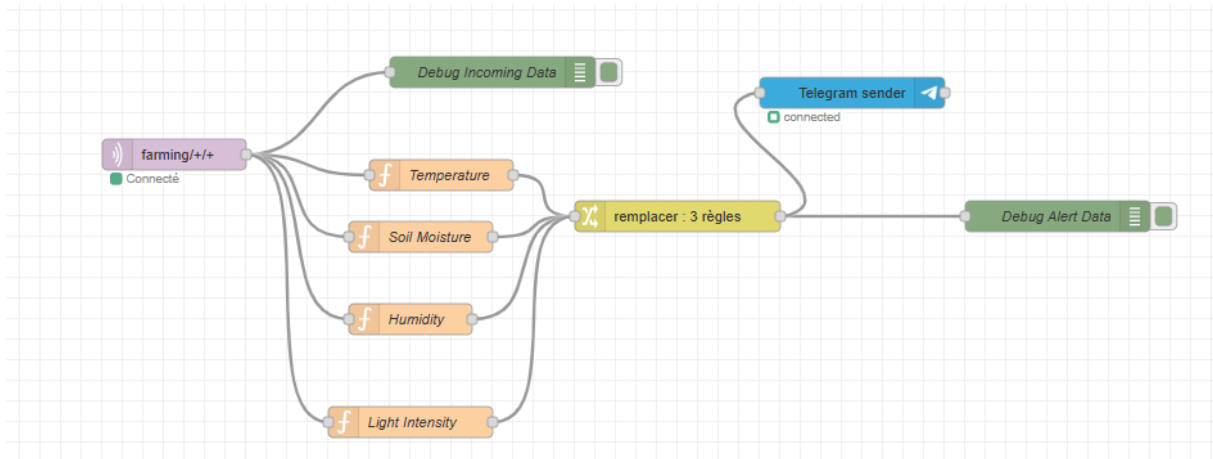


Humidity



9. Node-Red and Telegram:

I employed Node-RED, an IoT visual programming tool, to analyze data retrieved from MQTT Broker. This analysis was used to trigger alerts via a Telegram bot when the system detected some values that are not normal. From the Telegram side, I used “BotFather”, which is a bot created by Telegram that allows you to create and manage your own bots.



Create the Telegram Bot

- Create/open a telegram account
- Search for @botfather in the Telegram search bar
- Click on the Start button to interact with the BotFather
- Type /newbot, and follow the prompts to set up a new bot.
- Save the token the BotFather provides. You will use it to authenticate to the Telegram API and interact with your bot.
- In your newly created bot, send the message /start to activate the bot.

10. Conclusion

This **IoT Smart Farming System** successfully integrates **real-time monitoring, data storage, visualization, and alerts**. Future improvements include:

- **AI-based anomaly detection.**
- **Integration with weather APIs** for predictive insights.
- **Mobile App for real-time alerts & control.**