Hochschule für Technik
und Wirtschaft Berlin

University of Applied Sciences

GROW BUDDY

A TELEGRAM BASED SOLUTION FOR MAINTAINING HOUSE PLANTS

REPORT BY

Sneha Surendra

(Matrikelnummer: 595576)
Hochschule für Technik und Wirtschaft (HTW)
Sneha.Surendra@Student.HTW-Berlin.de

1. Introduction

This project aims to develop a personalized Telegram chatbot that provides real-time information about the current temperature, humidity, and soil moisture. It will also send reminders to water the houseplant based on data from an installed sensor. The chatbot will offer command options to fetch real-time data. Additionally, the project addresses all aspects of the Internet of Things (IoT), including nodes, communication, data processing, storage, and visualization.

1.1 Project Purpose

The purpose of this document is to provide a detailed description of the use case that the project is attempting to achieve and also provide an explanation on how various components (Microcontrollers, Sensors, Data Transmission, Data Processing, Data Visualization and Alarm) were used together to implement the project. In the future, this document should act as a reference and foundation on which further improvements can be implemented.

1.2 Project Scope

The objective of this project is to develop an IoT-based system that monitors environmental conditions, such as temperature and humidity, to send timely watering reminders for houseplants. The system integrates sensors to collect real-time data and calculates soil moisture levels to determine when watering is necessary. Notifications will be sent through a Telegram chatbot, providing a user-friendly and accessible platform for users to stay informed. This solution is designed for various applications, including home gardens, nurseries, greenhouses, and small-scale farms, ensuring efficient and precise plant care.

- House Gardening Assistant:

Offering plant-specific care recommendations, such as watering schedules and preferred humidity levels. Monitoring plant growth stages and providing reminders for fertilizing, pruning, and repotting. Connecting with smart devices like automated sprinklers and grow lights.

- Disease and Pest Control:

Detecting conditions conducive to pests or diseases (e.g., high humidity). Sending notifications about potential risks and preventive measures. Integrating pest detection cameras and analyzing data to identify infestations early.

- Nursery and Smart Greenhouse Management:

Automating irrigation schedules for large-scale nurseries based on temperature, humidity, and soil moisture. Monitoring environmental conditions across multiple zones within the nursery. Automating ventilation, shading, and lighting systems.

- Soil Health Monitoring System

Measuring soil pH, nutrient levels, and moisture content. Providing recommendations for soil amendments or treatments. Tracking long-term soil health trends for sustainable farming.

2. Data Processing

2.1 Implementation Of Features

For beginners, simply keeping their houseplants alive can be challenging. A reminder to water the plants can be invaluable in a busy life. A routine tailored to different plant species along with an automated drip irrigation system, would make plant care even easier. Additionally, an alarm to indicate when the temperature is unfavorable for plants could be very helpful. A Telegram chatbot could assist in remotely switching on the water pump. Temperature change alerts delivered via Telegram are critical, as they directly affect the plants' life expectancy. Furthermore, variations in temperature and humidity can signal larger weather patterns, such as the onset of extreme conditions like frost or heat waves. Such notifications are also beneficial for the general population when planning outdoor activities.

Implementation for the Alarm:

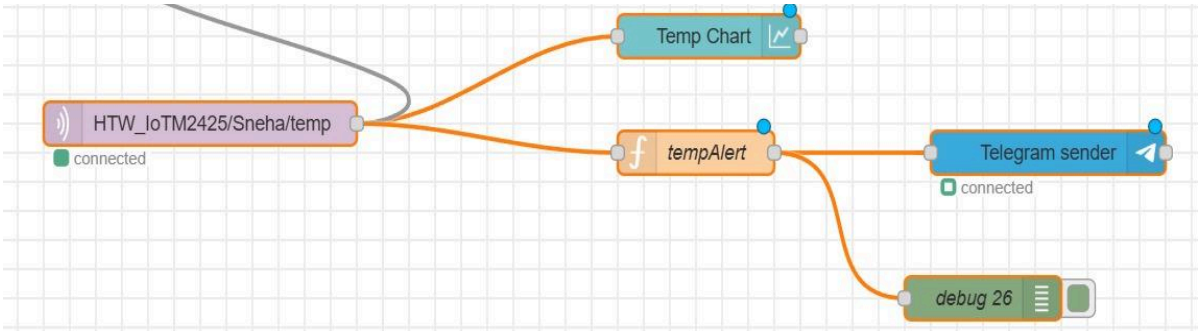

Fig 1.1. Alarm connection in Node Red

Skeleton of the Alarm:

```
if (msg.payload > 15) {
    currentRange = "above 15";
}
else if (msg.payload >= 10 && msg.payload <= 15) {
    currentRange = "10 to 15";
}
else if (msg.payload < 5) {
    currentRange = "below 5";
}
else {
    currentRange = "other";
}

// Check if the current range is different from the previous one
if (currentRange !== flow.get('previousRange')) {
    // Update the previous range
    flow.set('previousRange', currentRange);

    // Trigger the message only if the range changes
    if (currentRange === "above 15") {
        msg.payload = {
            chatId: 1467891015,
            type: 'message',
            content: "The temperature is beyond recommended for your plants " +
msg.payload
        };
        return msg;
    }
     else if (currentRange === "10 to 15") {
        msg.payload = {
            chatId: 1467891015,
            type: 'message',
            content: "The temperature is in a suitable condition" + msg.payload
        }; return msg;}
    else if (currentRange === "below 5") {
        msg.payload = {
            chatId: 1467891015,
             type: 'message',
            content: "The temperature is very low " + msg.payload
        };
        return msg;
    }// Add more conditions if needed for other ranges
}// If there's no change in the range, return null
return null;
```

Fig 1.2. Logic for temperature alarm

Implementation for the Soil Moisture Readings:

Accurate soil moisture readings are important to maintain the growing environment for plants health and yield. The empirical formula is used to calculate soil moisture in this project. Soil's moisture content enables precise irrigation, preventing both overwatering and underwatering. In our project, we see that soil moisture levels control the working of the water pump. Further, it can aid in optimizing fertilizer application, ensuring efficient resource use and promoting sustainable gardening practice. With changing weather conditions, it can aid the owner to strategize the position and planting productivity of new plants.

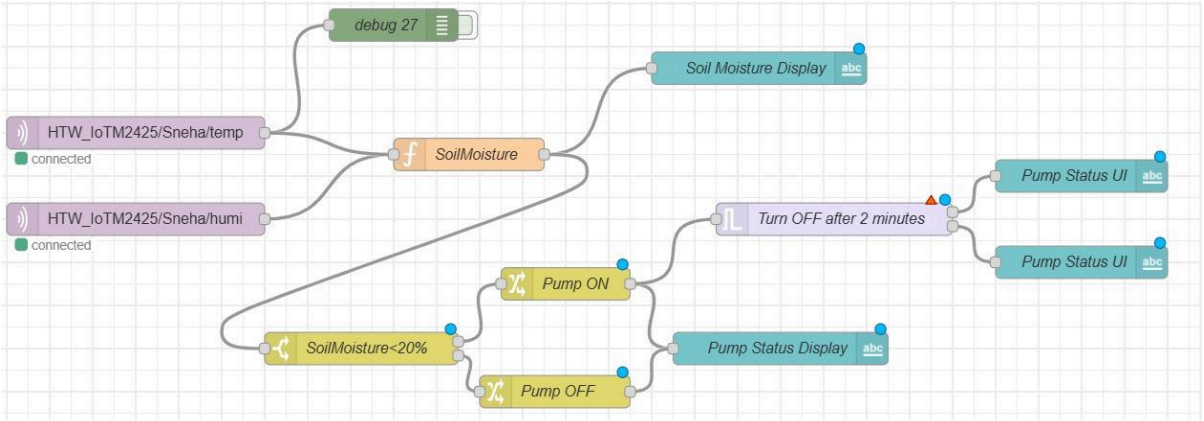

Fig 1.3. Soil Moisture connectivity in Node Red

```
} else if (sensorType === "humi") {
    const rawHumidity = msg.payload;
    global.set("rawHumidity", msg.payload);
    humidity = parseFloat(rawHumidity); // Declare the humidity variable

    // Debug messages
    node.warn("Parsed Humidity: " + humidity);

    if (rawHumidity === null || isNaN(humidity)) {
        // Check if humidity is null or not a valid number and log a warning
        if (rawHumidity === null) {
            node.warn("Humidity is null.");
        } else {
            node.warn("Humidity is not a valid number.");
        }

        // Handle the case when humidity is null or not a valid number
        msg.payload = {
            "content": "Problem with humidity data. Cannot calculate estimated soil moisture."
        };
    } else {
        // If humidity is valid, store it globally
        global.set("humidity", humidity);
    }
}

// Now, calculate soil moisture only when both temperature and humidity values are available
if (global.get("temperature") !== undefined && global.get("humidity") !== undefined) {
    const temp = global.get("temperature");
    const hum = global.get("humidity");

    // Calculate estimated soil moisture using both temperature and humidity
    var estimatedSoilMoisture = (tempCoefficient * temp) + (humidityCoefficient * hum) + constantTerm;

    // Debug messages
    node.warn("Estimated Soil Moisture: " + estimatedSoilMoisture);

    // Create a new message object with the estimated soil moisture, temperature, and humidity
    msg.payload = {
        temperature: temp,
        humidity: hum,
        estimatedSoilMoisture: estimatedSoilMoisture.toFixed(2) // Rounded to 2 decimal places
    };
    global.set("estimatedSoilMoisture", estimatedSoilMoisture);
}
```

Fig 1.3. Logic for Soil Moisture calculation in Node Red

2.2 Integrating Inline Keyboard in Telegram:

The temperature, humidity and the soil moisture data are sent to the end users via a Telegram bot that was created using Botfather, configured, and connected to Node-RED using the token. The Telegram bot was set up to respond to commands and provide end users with real-time temperature and pump control. We added a callback query to listen when the command is triggered and then redirect the request to their respective function for processing using a switch node.

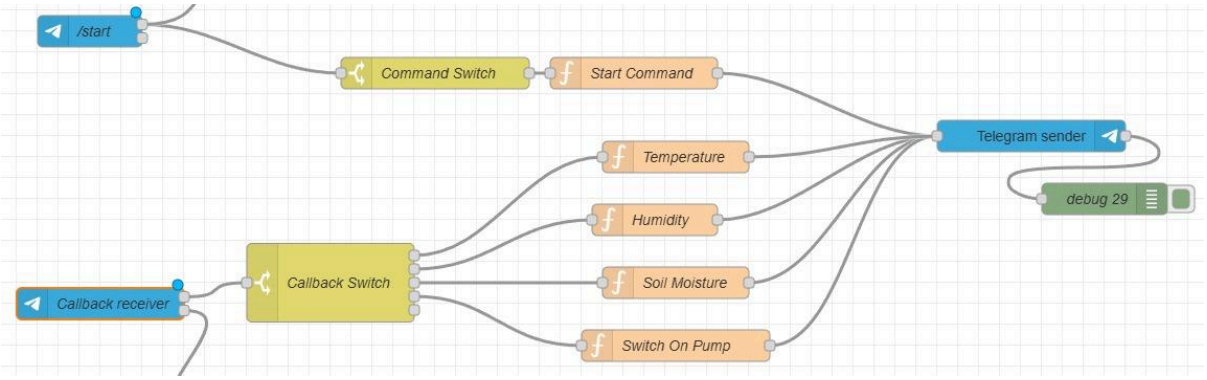

Fig. 1.4. Telegram chat bot flow

Command/ Nodes	Function
/start	It triggers the chatbot start and to display user options
Callback receiver	It triggers the output when a particular event is received from chat.
Start Command function	Sends a welcome message and gives Inline keyboard options to display Temperature, Humidity, Soil moisture values and to control the water pump.
Temperature, Humidity and Soil Moisture, Switch On Pump	It Processes the data and depending upon the value calculates the desired service and controls the water pump

Table 1.5. Components of telegram flow

```
msg.payload = {
    chatId: msg.payload.chatId,
    type: "message",
    content: "Hi Sneha! Choose an option below:",
    options: {
        reply_markup: JSON.stringify({
            inline_keyboard: [
                [{ text: "Check Temperature", callback_data: "temperature" }],
                [{ text: "Check Humidity", callback_data: "humidity" }],
                [{ text: "Check Soil Moisture", callback_data: "soil_moisture" }],
                [{ text: "Switch On Pump", callback_data: "switch_pump" }]
            ]
```

Fig. 1.6. Logic for Inline keyboard in Telegram

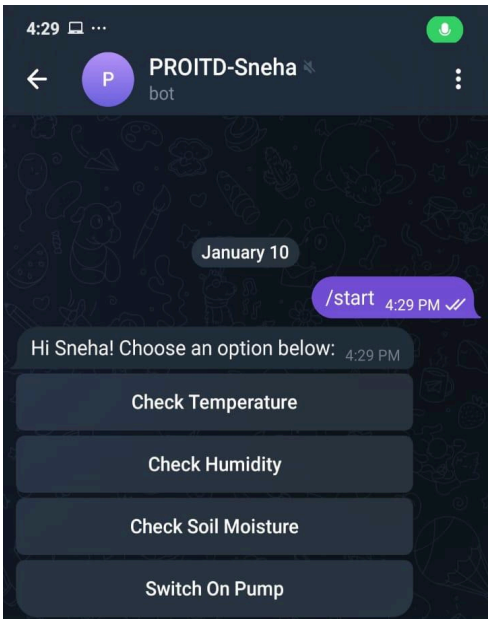

Fig 1.7. Telegram UI Page

3. Data Storage

3.1 Pushing the values in the Influx Database:

We take the published temperature and humidity from our MQTT and store it in Influx Database. The node “InfluxDb Out” helps output data to our database as a particular measurement ie. temperature and humidity. To configure Influx DB nodes, we need to configure the server to where our Influx DB is hosted, its organization name, bucket name, and the measurement name.

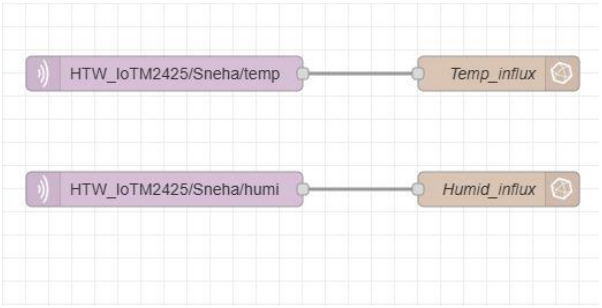

Fig. 1.8.. Node Red flow to push data to Influx DB

Fig. 1.9. Influx Configuration

A **bucket** is a named location where time series data is stored. All buckets have a retention period given in the figure fig. 2.0, here for our case we have a **retention period** of 90 days for the bucket. Influx DB drops all points with timestamps older than the bucket’s retention period. The stored database table show the readings that have been stored over the period of time.

2025-01-12 02:00:14 GMT+1	2025-01-12 03:00:14 GMT+1	2025-01-12 02:24:30 GMT+1	17	value
2025-01-12 02:00:14 GMT+1	2025-01-12 03:00:14 GMT+1	2025-01-12 02:25:30 GMT+1	17	value
2025-01-12 02:00:14 GMT+1	2025-01-12 03:00:14 GMT+1	2025-01-12 02:26:30 GMT+1	17	value
2025-01-12 02:00:14 GMT+1	2025-01-12 03:00:14 GMT+1	2025-01-12 02:27:30 GMT+1	17	value
2025-01-12 02:00:14 GMT+1	2025-01-12 03:00:14 GMT+1	2025-01-12 02:28:30 GMT+1	17	value
2025-01-12 02:00:14 GMT+1	2025-01-12 03:00:14 GMT+1	2025-01-12 02:29:30 GMT+1	17	value

Fig. 2.0. Stored data for the Temperature Readings

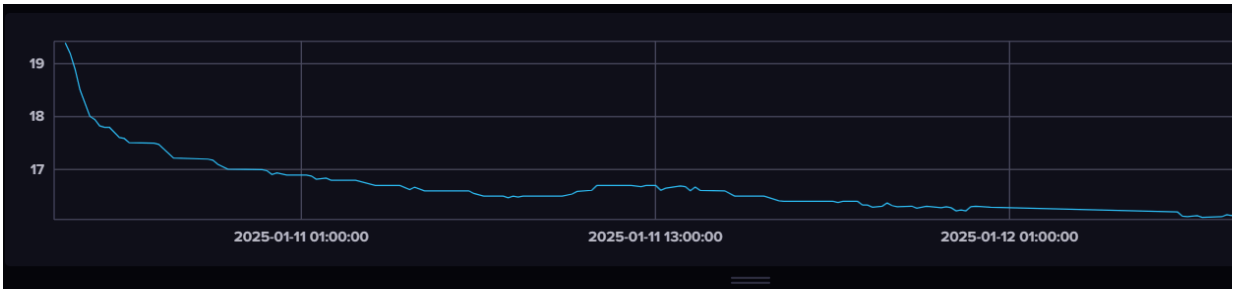

Fig. 5.3. Applying various functions as Interpolation on the stored data

4. UI and Visualization

Node -red Dashboard and Influx db(v2) is used for Visualization

4.1 Data Visualization in Node Red Dashboard

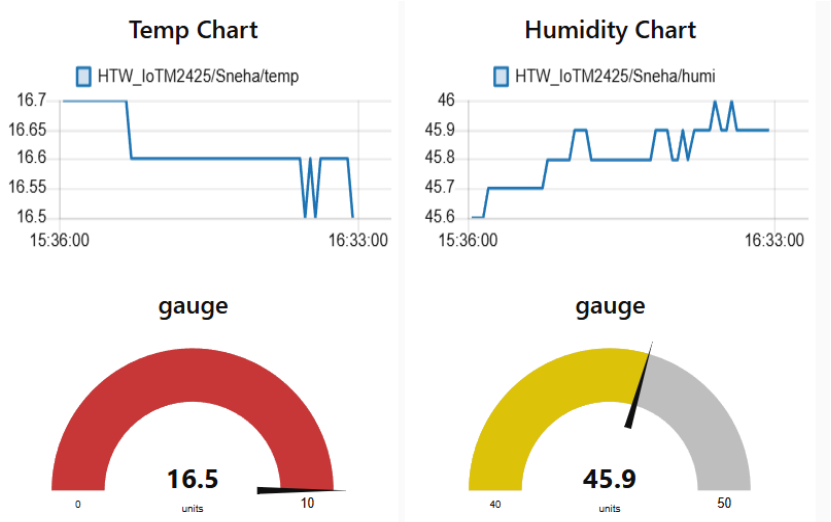

Fig. 6.1. Node Red Dashboard for the individual reading

4.2 Data Visualization in Influx DB

InfluxDB is a time-series database optimized for handling large volumes of time-series data, making it well-suited for IoT applications that generate significant amounts of information. Its second version, InfluxDB v2, offers advanced querying and aggregation tools, including the InfluxQL query language, which enhance its capabilities for analyzing and visualizing IoT data.

Fig. 5.4.Influx DB Graph with average humidity for the recorded data

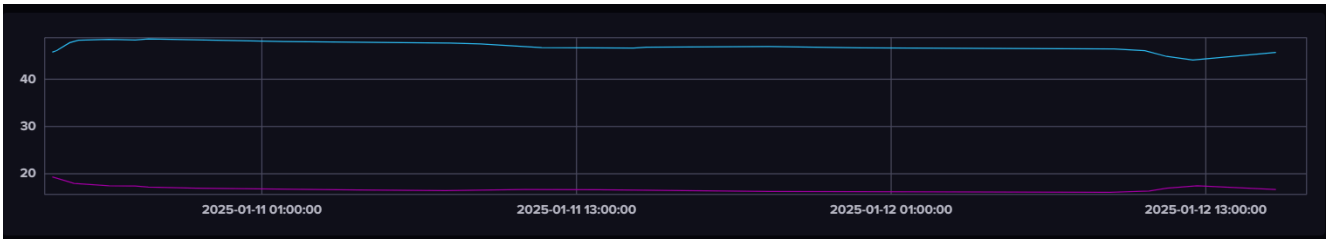

Fig. 5.8. Visualizing the temperature data and humidity data in one graph

Conclusion

The Telegram-based chatbot developed in this project is a convenient and efficient tool for maintaining the house plants. The chatbot is user-friendly and can be easily integrated into various settings such as Nurseries, Green Houses, even the areas where there is pesticide management, crop storage facilities. The chatbot can be further improved by adding more features and integrating with other APIs to provide more information.

On a larger scale, temperature, humidity, soil moisture monitoring is an important aspect in farming or agricultural industries. Accurate and timely data can be used to optimize processes, improve yields, and make informed decisions.

1.3 Future work

- Implementing a machine learning model to make the chatbot more interactive and personalized.
- Integrating the chatbot with other APIs to provide more information such as air quality, UV index, encyclopedia of houseplants maintenance and seasonal pesticide information . Integrating physical sensors such as soil moisture sensors, will only improve the accuracy of the system.
- Adding the feature of historical data and analysing it to predict future conditions.

References

1. Hive-MQTT: https://www.hivemq.com/blog/mqtt-essentials-part-1-introducing-mqtt/
2. Telegram Bot API -https://core.telegram.org/bots/api
3. InfluxDb Documentation: https://docs.influxdata.com/influxdb/v2.1/
4. Telegram Bot documentation: https://flows.nodered.org/node/node-red-contrib-telegrambot
5. Setting up influx DB: https://www.youtube.com/watch?v=CXRXy_PJxiA
6. IOT in automated plant watering system:
7. https://i-cubex.com/pishield/node-red-flow-for-automated-plant-watering-system/
8. https://www.instructables.com/IoT-Irrigation-System/
9. https://community.element14.com/products/raspberry-pi/raspberrypi_projects/b/blog/posts/raspberry-pi-home-irrigation-system-using-node-red
10. Artificial Intelligence in IOT: https://youtu.be/Rf_knQPKKl8?feature=shared

GitLab Repository

https://gitlab.rz.htw-berlin.de/s0595576/IOTW2425