Swinburne University of Technology

IoT Programming (SWE 30011) Group Assignment (Practical) Project Report

Leader: Mallika Arachchillage Nidula Sanketh Mallikarachchi
Student ID: 104756611

Member 1: Anh Tran
Student ID: 104204233

Member 2: W Weerakuttige Nethum V Sandadinidu Fernando
Student ID: 104832553

Contents

Introduction 3

IoT System for a Smart Home Environment 3

Project Overview 3

Conceptual Design 4

Project Concept 4

Objectives 4

System Components and Architecture 5

Explanation of Bi-Directional Communication using MQTT 5

Task Breakdown 7

Implementation 8

1. Development and Deployment of the Physical System 8

2. Sending Data to the ThingsBoard Cloud Platform 8

3. Sending Signals from the Main Arduino Board to the Sub Arduino Boards to Control Them Manually 11

4. Implementation of Rules using ThingsBoard 11

User Manual 13

Setting Up Edge Servers 13

Preparing the Code 13

Running MQTT Scripts 13

Important Configuration for Debian Linux Users 13

Setting Up ThingsBoard 14

Running the Application 14

Limitations 14

Resources 15

Conclusion 16

Introduction

IoT System for a Smart Home Environment

In today's technologically advanced world, the integration of Internet of Things (IoT) systems into everyday life has become increasingly prevalent, enhancing convenience, efficiency, and automation in various domains. Our project aims to conceptualize, develop, and implement a comprehensive IoT-based smart home system. This system encompasses multiple subsystems, each designed to automate specific functions within a household, thereby improving overall living conditions and energy management.

Project Overview

Our smart home system integrates three main functionalities:

1. **Smart Lighting System** - Automatically controls lighting based on ambient light levels.
2.
3. **Smart Air Conditioning System** - Regulates the temperature by activating the air conditioning when it becomes too hot.
4. **Smart Garden System** - Manages the watering of plants by monitoring soil moisture levels.

Each of these functionalities is managed by individual Arduino boards, with a main board and two sub-boards, ensuring a modular and scalable design. The main board handles the smart lighting system, while the two sub-boards are responsible for the air conditioning and garden systems, respectively.

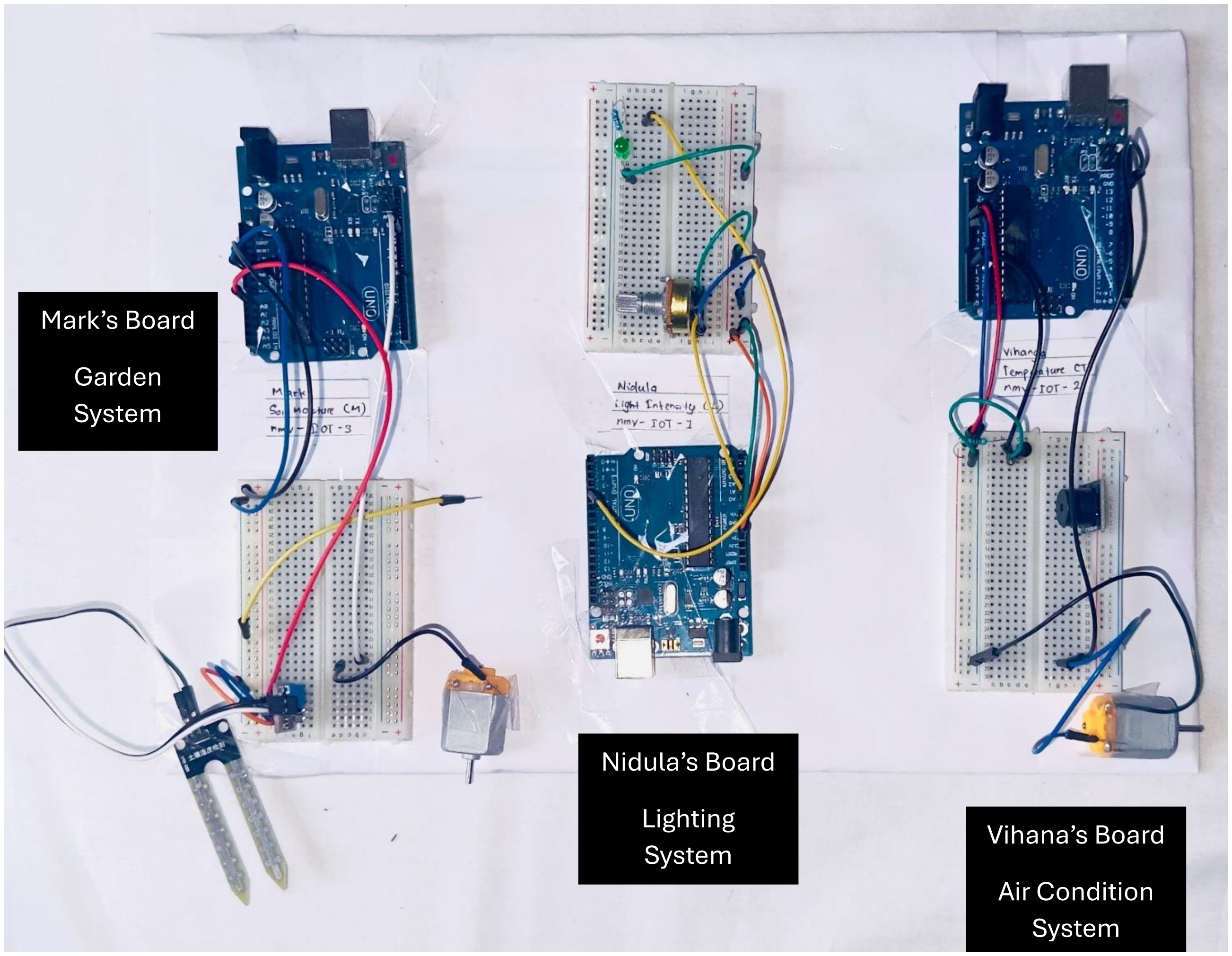


Figure 1. Physical Implementation

Conceptual Design

Project Concept

The concept of our IoT-based smart home system is rooted in enhancing the convenience and efficiency of household management through automation and real-time monitoring. The idea is to develop a system that can autonomously manage lighting, air conditioning, and garden watering, adapting to environmental conditions and user preferences without manual intervention.

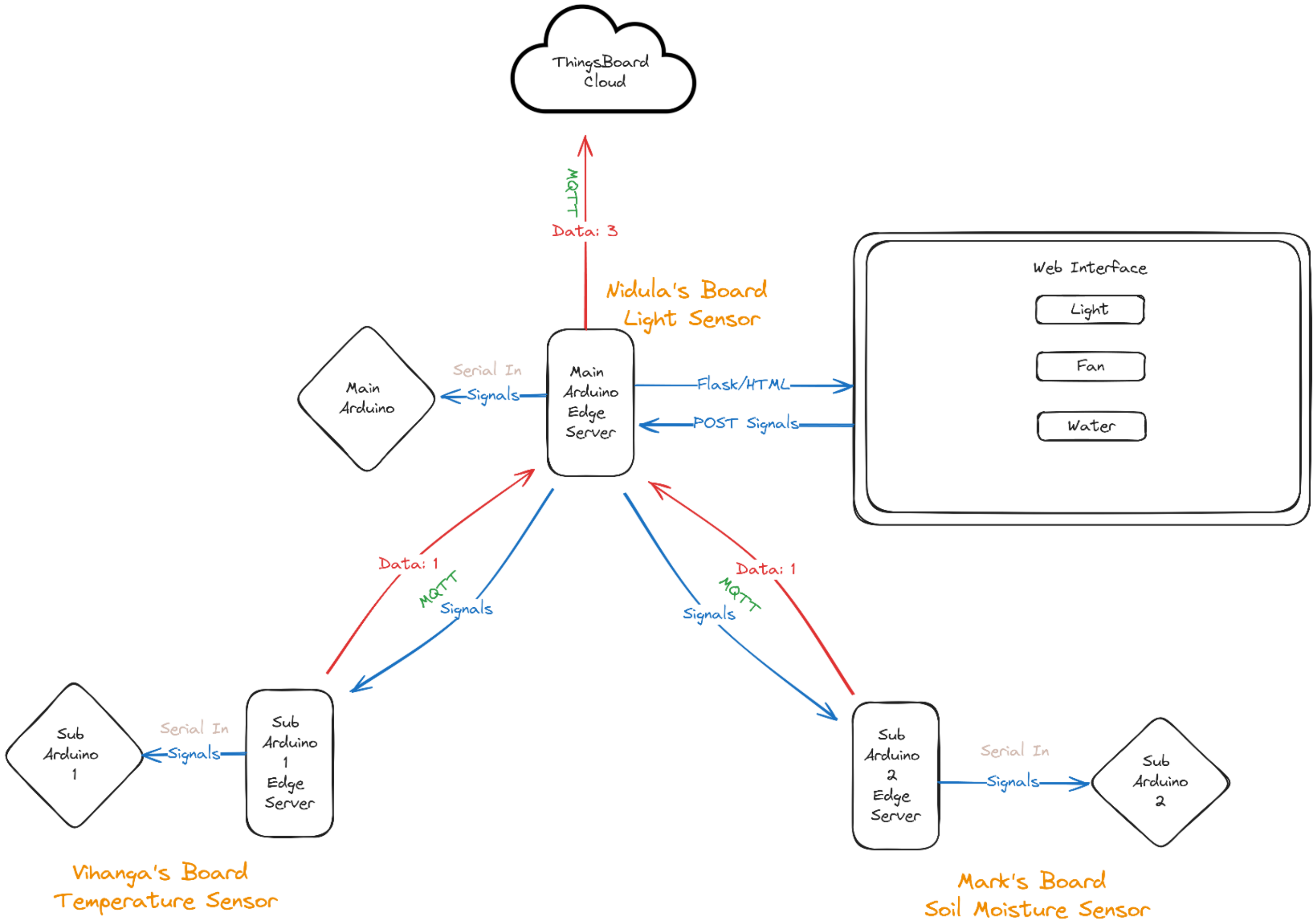


Figure 2. Conceptual Design

Objectives

1. **Automation**: Implement a system that automatically adjusts the lighting, temperature, and plant watering based on sensor data.
2. **Energy Efficiency**: Optimize the use of resources such as electricity and water, reducing wastage and promoting sustainability.
3. **User Convenience**: Provide a user-friendly interface for remote monitoring and control of the system, enhancing the overall user experience.
4. **Scalability**: Design the system in a modular fashion, allowing for easy addition of new functionalities and components.

System Components and Architecture

The smart home system is composed of three primary subsystems, each managed by a dedicated Arduino board. These subsystems are integrated into a cohesive system through a combination of sensors, actuators, communication protocols, and cloud-based data management.

1. **Smart Lighting System**
   - **Components**:
     - LED for lighting.
     - Potentiometer serving as an ambient light sensor.
     - 200-ohm resistor.
   - **Functionality**:
     - The potentiometer measures ambient light levels.
     - When light levels fall below a predefined threshold, the LED is turned on to illuminate the area.
     - The system can also receive remote commands to manually control the LED via a web interface.
2. **Smart Air Conditioning System**
   - **Components**:
     - Temperature sensor.
     - Motor representing the air conditioning unit.
     - 200-ohm resistor.
   - **Functionality**:
     - The temperature sensor monitors the ambient temperature.
     - When the temperature exceeds a set threshold, the motor is activated to simulate the operation of an air conditioning unit.
     - Remote control of the air conditioning system is also possible through the web interface.
3. **Smart Garden System**
   - **Components**:
     - Soil moisture sensor.
     - Motor representing the plant watering system.
   - **Functionality**:
     - The soil moisture sensor measures the moisture level in the soil.
     - When moisture levels drop below a specified threshold, the motor is activated to water the plants.
     - Users can remotely control the watering system via the web interface.

Explanation of Bi-Directional Communication using

The Arduino Boards Communicate with each other using MQTT Protocol. The 2 sub Arduino Boards Sends Signals that are received on their serial ports to the Main Arduino board using MQTT and the Main Arduino board published collected data to the ThingsBoard Cloud Platform for analysis.

A Frontend has been established on the edge server of the main Arduino to control the sub Arduino boards and itself manually. The Interface sends Signals to the sub Arduino boards using MQTT & To it's Arduino board using Serial Communication.

The Following Diagram Briefly explains the process of communication between the components of the system

Sequence Diagram

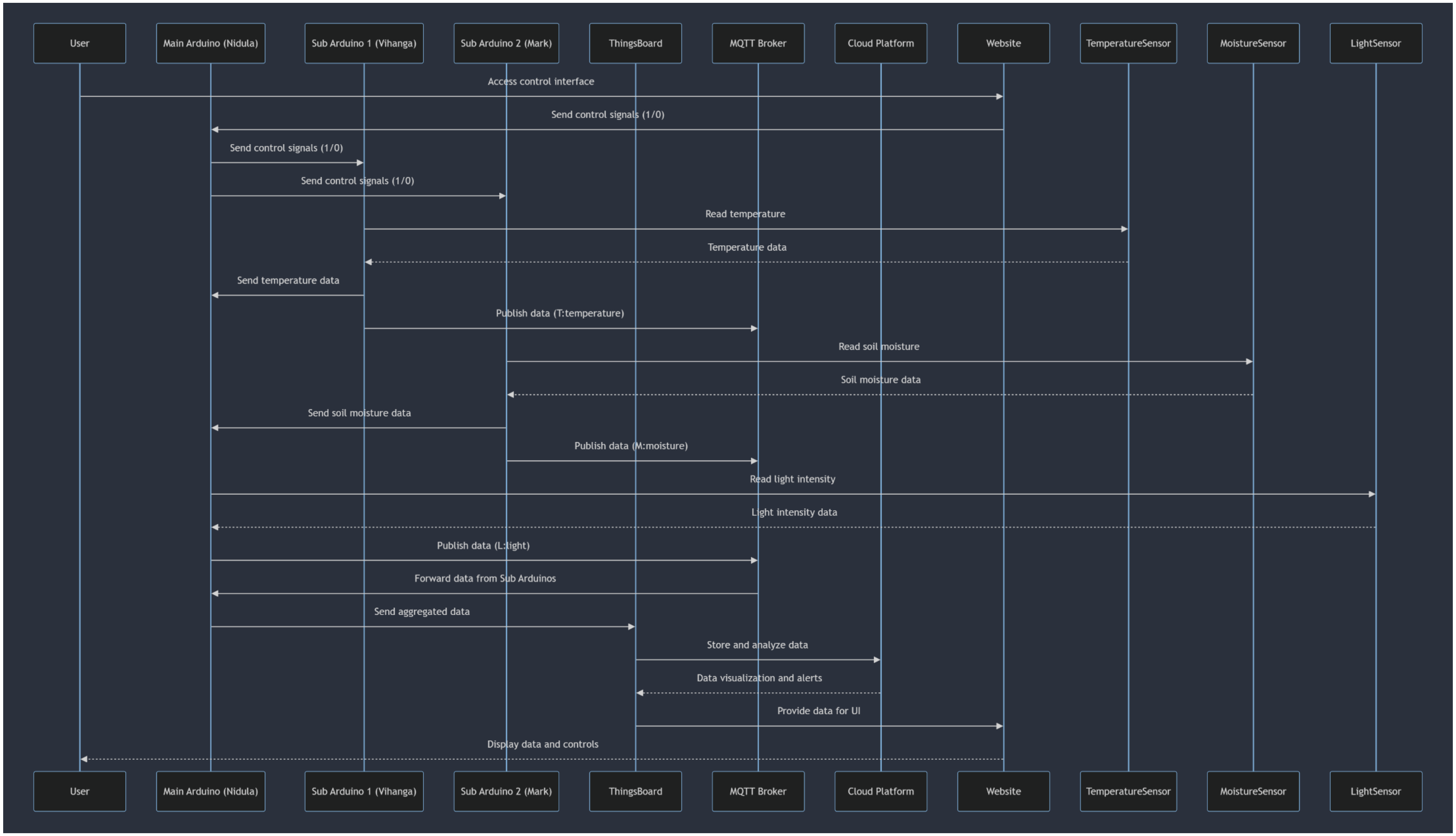


Figure 3. Sequence Diagram

Task Breakdown

Task	Description	Assigned Member	Details
Frontend Development	Designing and implementing the web interface for user interaction. (HTML & Flask)	Nidula Mallikarachchi	Developed HTML and Flask-based web interface, ensuring user-friendly controls and real-time updates.
Bidirectional Communication	Implementing MQTT communication between Arduino boards and the cloud platform. (Python)	Nidula Mallikarachchi	Set up MQTT topics and ensured reliable data transmission and control signals between devices and cloud.
ThingsBoard Connection	Integrating the IoT system with the ThingsBoard cloud platform for data visualization and management.	Nidula Mallikarachchi	Configured ThingsBoard devices, tokens, and telemetry data streams. Created rules for automated responses.
Project Report	Compiling and writing the project report detailing all aspects of the project.	Nidula Mallikarachchi	Documented project objectives, design, implementation, and results.
Arduino Codes Development	Writing and testing the Arduino code for each subsystem.	Mark Tran	Developed and tested Arduino code for smart lighting, air conditioning, and garden watering systems.
Smart Lighting System Code	Writing code for the main board to control the smart lighting system.	Mark Tran	Developed code to read light sensor data and control the LED based on ambient light levels.
Smart Air Conditioning System Code	Writing code for Sub-Arduino 1 to control the air conditioning system.	Mark Tran	Developed code to read temperature sensor data and control the motor representing the air conditioner.
Smart Garden System Code	Writing code for Sub-Arduino 2 to control the garden watering system.	Mark Tran	Developed code to read soil moisture sensor data and control the motor representing the watering system.
Physical Implementation	Assembling the hardware components and wiring the Arduino boards.	Vihanga Sandadinidu	Physically connected sensors, actuators, and ensured proper wiring of the Arduino boards.
Sensors and Actuators Setup	Setting up and calibrating sensors and actuators for each subsystem.	Vihanga Sandadinidu	Installed and calibrated the light sensor, temperature sensor, and soil moisture sensor.

Implementation

The implementation of our IoT-based smart home system can be categorized into four main topics:

1. Development and deployment of the physical system and Arduino codes needed for them.
2. Sending data to the ThingsBoard Cloud Platform for analysis.
3. Sending signals from the Main Arduino Board to the Sub Arduino Boards to control them manually.
4. Implementation of rules using ThingsBoard.

Let's discuss each of these topics one by one.

1. Development and Deployment of the Physical System

We have designated names for each of the Arduino Boards used in the project:

a. Main Arduino Board (Nidula's Board):

- Controls the smart lighting system.
- Collects data from the potentiometer and controls the LED based on light intensity.
- Responsible for sending data to the cloud platform.

b. Sub Arduino 1 (Vihanga's Board):

- Manages the smart air conditioning system.
- Reads temperature data and controls the fan motor based on temperature thresholds.

c. Sub Arduino 2 (Mark's Board):

- Handles the smart garden system.
- Reads soil moisture data and controls the watering motor based on moisture levels.

Each of these boards is programmed using the Arduino IDE to receive serial input and to output serial data received by the boards.

2. Sending Data to the ThingsBoard Cloud Platform

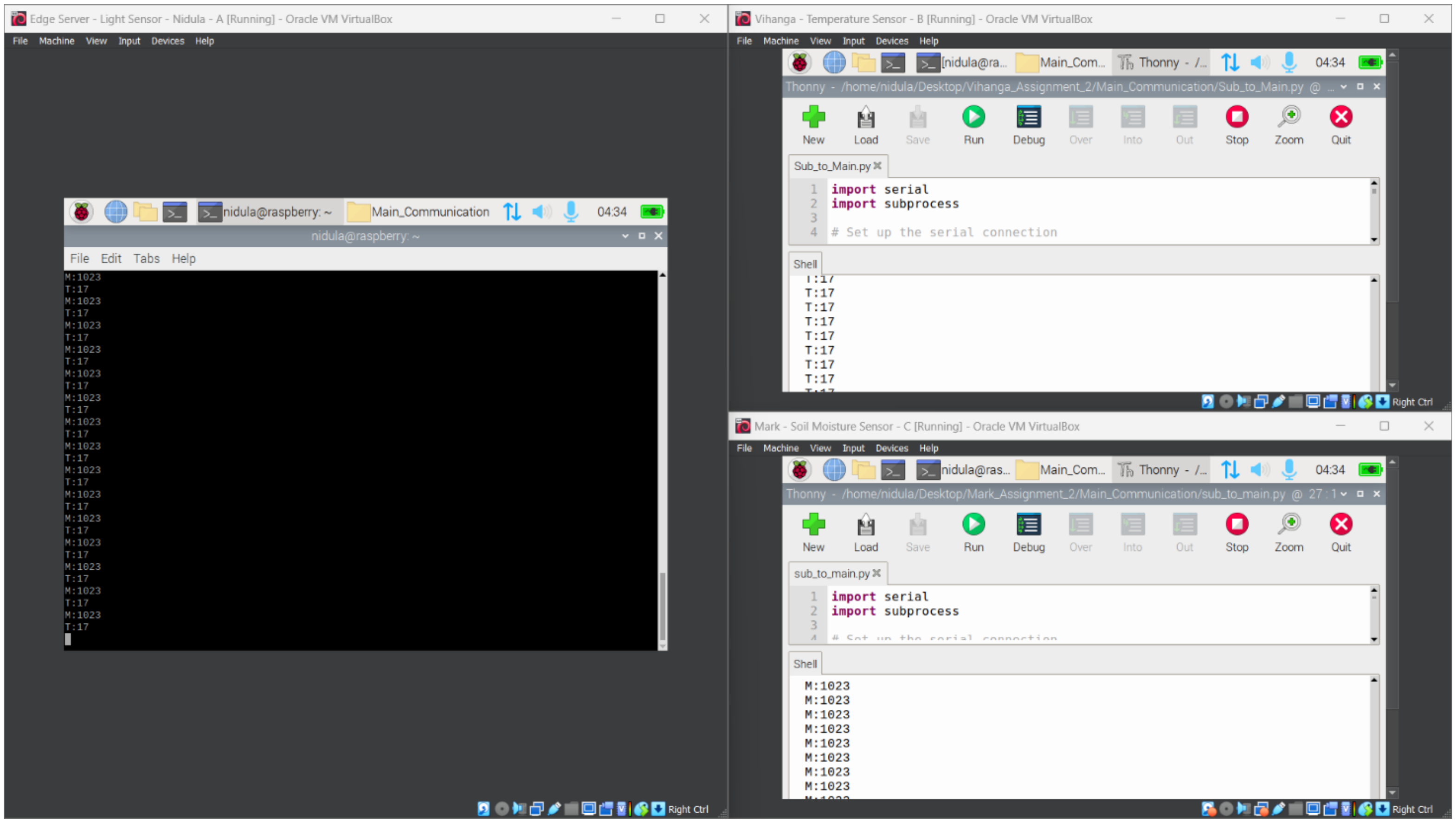
- The Sub Arduino Boards receive data on their serial ports from the sensors.
- This data is captured by the edge server using a Python script named **send_to_main.py**, which sends the data to the main Arduino Board via MQTT protocol under the topic **nvm-main-arduino-topic**.
- The Main Arduino Board captures the data received via MQTT and sends it to the cloud platform using a Python script named **main_to_cloud.py**.
- This Python script contains all the access tokens of the devices:

```
device_tokens = {
    'T': '2hj0DuOWzNI6LqM3yFjl',
    'M': 'HM4E36xjHFWvJ6b09KnV',
    'L': '271FtNXZSU3HjXz8Jra9'
}
```

- Once a message is received, the Python script sends it to the ThingsBoard Platform using the following function:

```python
def publish_to_thingsboard(device, value):
  if device in device_tokens:
    token = device_tokens[device]
    topic = 'v1/devices/me/telemetry'
    message = f'{{"{device}":{value}}}'
    command = [
      'mosquitto_pub', '-d', '-q', '1',
      '-h', 'mqtt.thingsboard.cloud', '-p', '1883',
      '-t', topic, '-u', token, '-m', message
    ]
    subprocess.run(command)
  else:
    print(f"Key '{device}' not found in device_tokens dictionary.")
```

- The sent data is analyzed in the ThingsBoard Cloud Platform in various ways, as shown in the diagrams below.


Edge Server - Light Sensor - Nidula - A [Running] - Oracle VM VirtualBox
Vihanga - Temperature Sensor - B [Running] - Oracle VM VirtualBox
Mark - Soil Moisture Sensor - C [Running] - Oracle VM VirtualBox
Thonny - /home/nidula/Desktop/Vihanga_Assignment_2/Main_Communication/Sub_to_Main.py
Thonny - /home/nidula/Desktop/Mark_Assignment_2/Main_Communication/sub_to_main.py
New
Load
Save
Run
Debug
Over
Into
Out
Stop
Zoom
Quit
Shell

```
M:1023
T:17
M:1023
T:17
M:1023
T:17
M:1023
T:17
M:1023
T:17
M:1023
T:17
M:1023
T:17
M:1023
T:17
M:1023
T:17
M:1023
T:17
M:1023
T:17
M:1023
T:17
M:1023
T:17
M:1023
T:17
M:1023
T:17
```

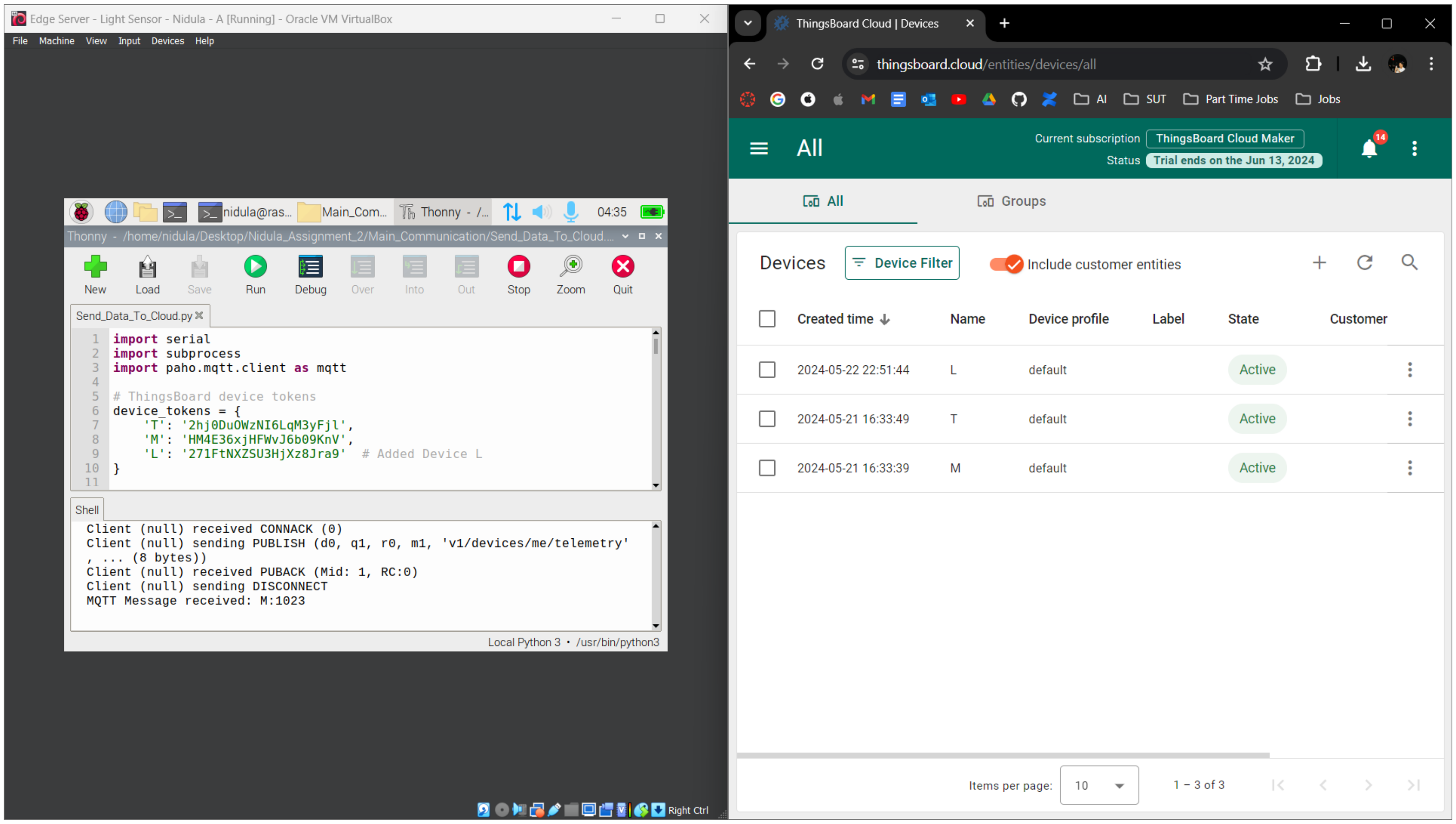
```python
import serial
import subprocess

# Set up the serial connection
```

```
T:17
T:17
T:17
T:17
T:17
T:17
T:17
```

```
M:1023
M:1023
M:1023
M:1023
M:1023
M:1023
M:1023
M:1023
```

Figure 4. Receiving Data on The Main Arduino Board/ Sending Data from the Sub Boards



```
import serial
import subprocess
import paho.mqtt.client as mqtt

# ThingsBoard device tokens
device_tokens = {
    'T': '2hj0DuOWzNI6LqM3yFjl',
    'M': 'HM4E36xjHFWvJ6b09KnV',
    'L': '271FtNXZSU3HjXz8Jra9'  # Added Device L
}
```

```
Client (null) received CONNACK (0)
Client (null) sending PUBLISH (d0, q1, r0, m1, 'v1/devices/me/telemetry'
, ... (8 bytes))
Client (null) received PUBACK (Mid: 1, RC:0)
Client (null) sending DISCONNECT
MQTT Message received: M:1023
```

Created time	Name	Device profile	Label	State	Customer
2024-05-22 22:51:44	L	default		Active	
2024-05-21 16:33:49	T	default		Active	
2024-05-21 16:33:39	M	default		Active	

Figure 6. Sending Data to the Cloud using python Script

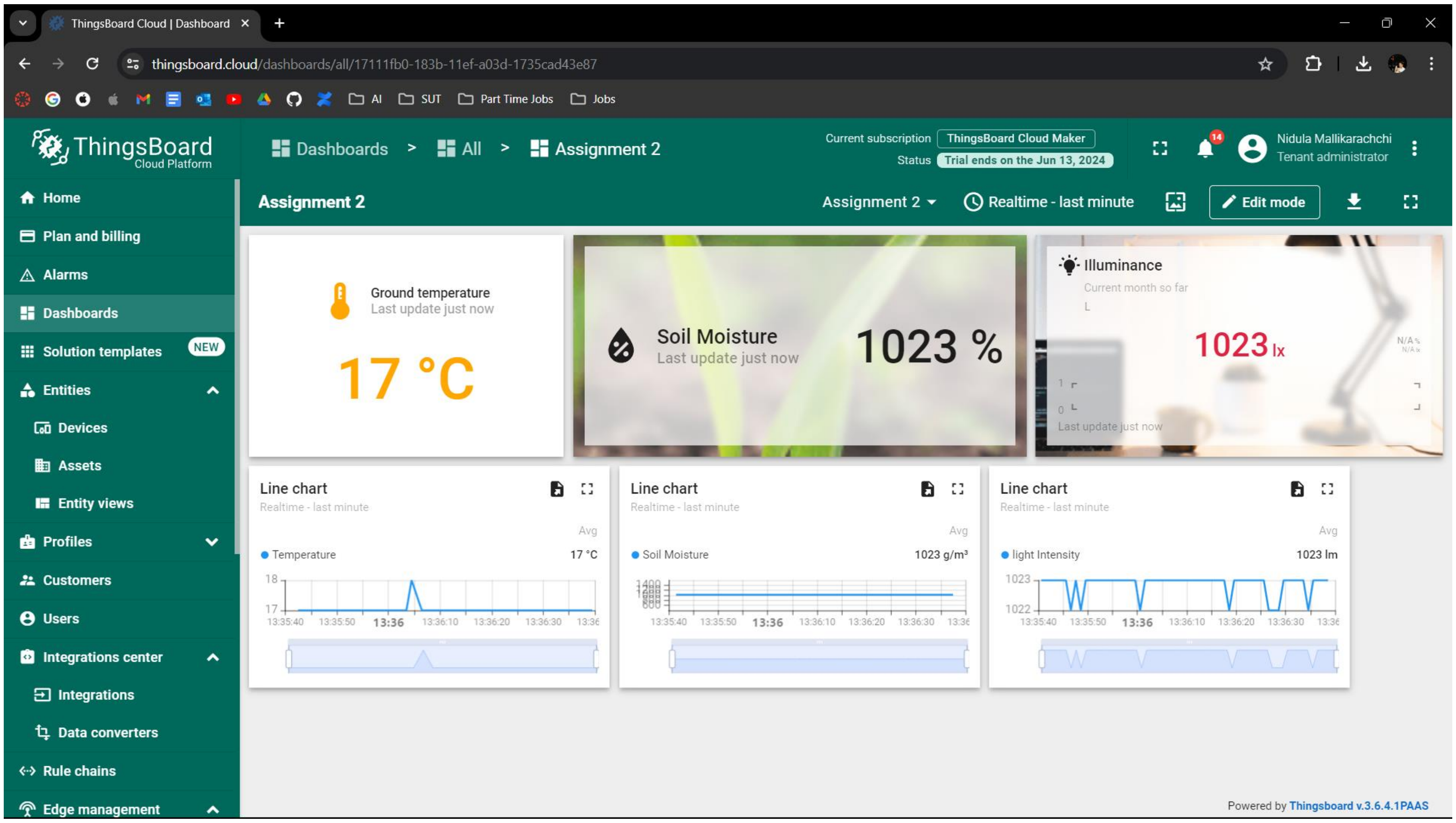


Figure 5. ThingsBoard Dashboard

3. Sending Signals from the Main Arduino Board to the Sub Arduino Boards to Control Them Manually

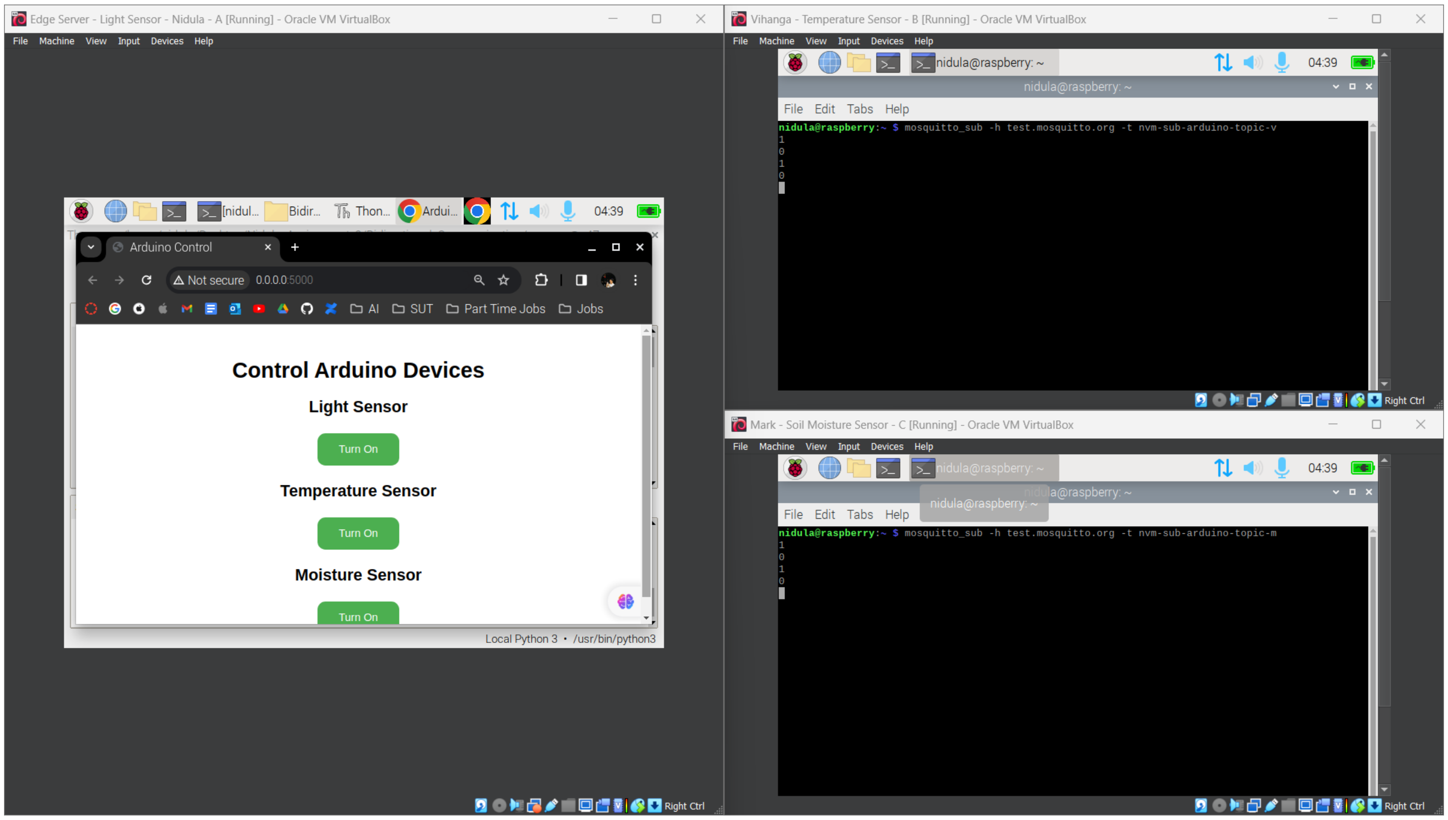
- To send data to the two Sub Arduino Boards, we hosted two unique topics on each of them:
    - For Vihanga's Board – Sub Arduino 1:

```
mosquitto_sub -h test.mosquitto.org -t nvm-sub-arduino-topic-v
```

    - For Mark's Board – Sub Arduino 2:d

```
mosquitto_sub -h test.mosquitto.org -t nvm-sub-arduino-topic-m
```

- After hosting these topics, a Python script named **listener.py** is run on both sub Arduino boards to listen to the MQTT signals received on their terminals.
- This Python script listens to the input received on the terminal and sends a serial input signal to the connected Arduino board. The signals received on the terminal are '1's and '0's, indicating ON and OFF, respectively.


Edge Server - Light Sensor - Nidula - A [Running] - Oracle VM VirtualBox
Arduino Control
Not secure 0.0.0.0:5000
Control Arduino Devices
Light Sensor
Turn On
Temperature Sensor
Turn On
Moisture Sensor
Turn On
Local Python 3 • /usr/bin/python3
Vihanga - Temperature Sensor - B [Running] - Oracle VM VirtualBox
Mark - Soil Moisture Sensor - C [Running] - Oracle VM VirtualBox

```
nidula@raspberry:~ $ mosquitto_sub -h test.mosquitto.org -t nvm-sub-arduino-topic-v
1
0
1
0
```

```
nidula@raspberry:~ $ mosquitto_sub -h test.mosquitto.org -t nvm-sub-arduino-topic-m
1
0
1
0
```

Figure 7. Sending Signals through the Interface and Receiving them on the sub–Arduino Boards

- The system interface is hosted on the main Arduino's edge server. This interface, developed using HTML and Flask, allows users to control the subsystems. The HTML file (**index.html**) includes JavaScript code that identifies which button is pressed and sends a POST signal to the Flask application (**app.py**).
- The **app.py** script publishes the signal received from the HTML interface to the relevant Arduino board's topic. When the signals are received on the Sub Arduino boards, they are sent as serial inputs to the connected Arduino boards, which then respond accordingly.

4. Implementation of Rules using ThingsBoard

We have created a chain rule in ThingsBoard to trigger an alarm based on the light intensity data received from the smart lighting system. Following is how the rule is setup

1. **Input**: The chain rule begins by receiving input messages containing light level data.
2. **Message Type Switch**: This component filters the input messages to identify those related to light levels.
3. **Post Telemetry**: The filtered messages are then posted to the telemetry system for further processing.
4. **Script (Light Level Trigger Script)**: A custom script evaluates the light level data against a predefined threshold to determine if an alarm should be triggered.

```
var threshold = 500; // Replace 500 with your threshold value
if (msg['L'] > threshold) {
  // Set metadata to trigger alarm
  metadata['alarm'] = true;
} else {
  metadata['alarm'] = false;
}
return { msg: msg, metadata: metadata };
```

5. **Alarm Activation**:
   - If the script sets **metadata['alarm']** to **true**, the alarm is triggered.
   - If the script sets **metadata['alarm']** to **false**, any existing alarm is cleared.
6. **Integration with Root Rule Chain**: This chain rule is connected to the Root Rule Chain in ThingsBoard, ensuring that it is evaluated as part of the primary data processing workflow.

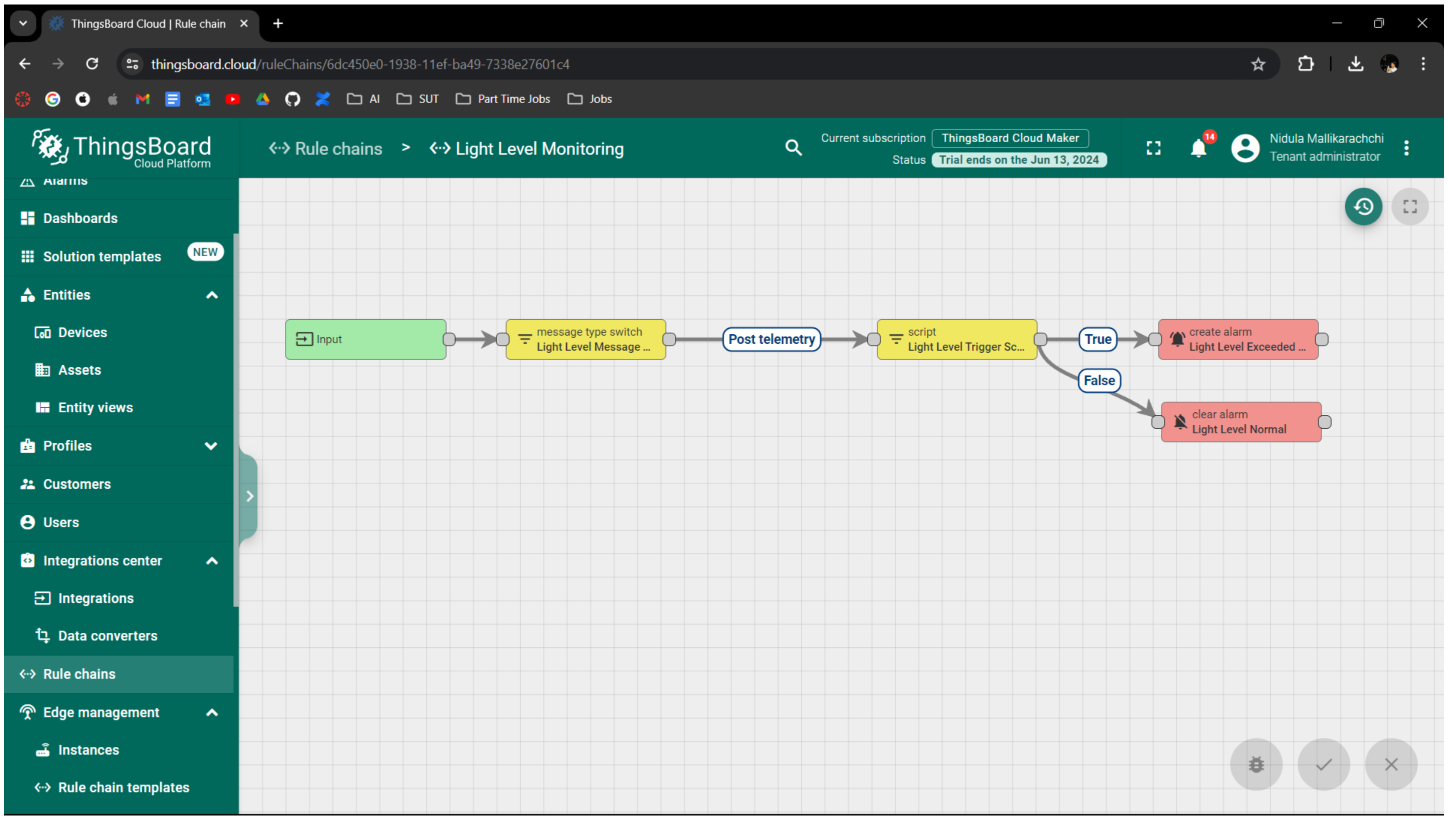


Figure 8. Rule Chain

User Manual

To ensure a smooth implementation, please follow the instructions below carefully.

Setting Up Edge Servers

1. **Create 3 Edge Servers:**
    - Ensure all edge devices are connected to the same Wi-Fi network for reliable communication.
    - Use a Bridged Adapter for network connectivity when setting up the EDGE servers.
2. **Software Requirements:**
    - Ensure Python 3, MQTT, Paho, Flask, and Pip3 are installed on all three machines.

Preparing the Code

1. **Unzip the provided "codes.zip" file:**
    - The archive contains three folders:
        1. Main Arduino – Nidula
        2. Sub Arduino 1 – Vihanga
        3. Sub Arduino 2 – Mark
    - Extract each folder to a separate machine.

Running MQTT Scripts

1. **Main Machine:**
    - Open the terminal and run the following command:

```
mosquitto_sub -h test.mosquitto.org -t nvm-main-arduino-topic
```

2. **Sub Arduino 1 Machine:**
    - Open the terminal and run:

```
mosquitto_sub -h test.mosquitto.org -t nvm-sub-arduino-topic-v
```

3. **Sub Arduino 2 Machine:**
    - Open the terminal and run:

```
mosquitto_sub -h test.mosquitto.org -t nvm-sub-arduino-topic-m
```

Important Configuration for Debian Linux Users

1. **Configuring the Python Interpreter:**
    - Open Thonny Python IDE.
    - Click on the button at the bottom left labeled "local python 3 /usr/bin/python3".
    - Click on "Configure Interpreter".

- Go to the “General” tab and uncheck “Allow Only Single Thonny Instance”.
- Press OK and restart the IDE.

Setting Up ThingsBoard

1. **Create 3 New Devices:**
   - On the ThingsBoard platform, create three new devices.
2. **Update Access Tokens and Device Names in “main_to_cloud.py”:**

```
device_tokens = {
  'T': '2hj0DuOWzNI6LqM3yFjl',
  'M': 'HM4E36xjHFWvJ6b09KnV',
  'L': '271FtNXZSU3HjXz8Jra9'
}

data = {'T': None, 'M': None, 'L': None}
```

3. **Run Scripts:**
   - On Sub Arduino 1 and Sub Arduino 2 machines, run: sub_to_main.py
   - On the main Arduino machine, run: main_to_cloud.py
4. **Create Visualizations:**
   - Create three visualizations of your choice on the ThingsBoard platform.
   - At this point, ThingsBoard should be functional.

Running the Application

1. **Run “listener.py” on both Sub Arduino Machines**
2. **Run “app.py” on the main machine on localhost port 5000:**
   - The user interface should now be accessible and operational.

By following these steps, your Smart Home System IoT Project should be fully functional. If you encounter any issues, double-check the configurations and ensure all dependencies are properly installed. Enjoy your smart home system!

Limitations

1. Scalability:

- **Current Limitations:** The existing system architecture supports a fixed number of devices and sensors. This limitation arises from the finite number of ports available on the Arduino boards and the processing power they can handle, which restricts the number of simultaneous operations.
- **Impact:** As the complexity and size of the home increase, adding more devices or sensors might require additional hardware or significant redesign of the system to maintain efficient operations.

2. Power Dependency:

- **Current Limitations:** The system relies entirely on a continuous electrical power supply. There is no integrated battery backup or alternative power source in case of a power outage, which can disrupt the functionality of home automation.
- **Impact:** In areas with frequent power fluctuations or outages, the reliability of the system can be compromised, leading to potential safety and security issues.

3. Sensor Accuracy:

- **Current Limitations:** Sensors used in the system, such as the light sensor, temperature sensor, and soil moisture sensor, can be affected by environmental conditions. Factors such as dirt accumulation, sensor placement, and aging can degrade sensor accuracy over time.
- **Impact:** Inaccurate sensor readings can lead to inappropriate responses from the system, like unnecessary watering of plants or improper indoor temperature management, resulting in inefficiencies and increased operational costs.

4.User Interface and Experience:

- **Current Limitations:** The user interface on the web-based control panel is relatively basic and might not offer the best user experience in terms of aesthetics and navigability.
- **Impact:** A less intuitive user interface can deter users from fully utilizing all the features of the system, thereby not achieving the intended efficiency and convenience.

Resources

1. **Arduino Documentation**
    - Arduino. (n.d.). *Arduino Documentation*. Retrieved from https://www.arduino.cc/reference
2. **MQTT Protocol Tutorial**
    - MQTT.org. (n.d.). *MQTT Protocol Tutorial*. Retrieved from https://mqtt.org/documentation
3. **ThingsBoard Official Guide**
    - ThingsBoard. (n.d.). *Official ThingsBoard Documentation*. Retrieved from https://thingsboard.io/docs
4. **YouTube Videos on IoT Projects**
    - YouTube. (n.d.). *IoT Project Videos*. Retrieved from https://www.youtube.com/results?search_query=IoT+projects
5. **ChatGPT for Troubleshooting**
    - OpenAI. (n.d.). *ChatGPT: Optimizing language models for dialogue*. Retrieved from https://www.openai.com/chatgpt

Conclusion

Completing our IoT Smart Home Automation System has been a rewarding journey, allowing us to bring a touch of automation into everyday life. By integrating smart solutions for lighting, air conditioning, and gardening, our project has showcased the practical benefits of IoT technologies.

While we're proud of what we've achieved, we also recognize areas where our system can grow. Issues like scalability and sensor accuracy present challenges but also opportunities to enhance our system further. We are particularly excited about exploring ways to improve user interaction and integrating more robust security features.

As we look to the future, our team is eager to build on this foundation, refining our design to better meet the needs of smart homes everywhere. This project hasn't just been about building a system; it's been about learning and growing together, and we can't wait to see where these insights take us next.