Table of Contents

Introduction

1.1 Overview

This IoT smart room climate monitoring system is designed to continuously monitor and dynamically control room temperature and humidity. It sends measured values over an MQTT broker, facilitating seamless communication, and visualizes the data on a Node-RED dashboard, and alerts the user via a Telegram bot when the measured values reach the threshold limit.

Key Feature:

1. **Integration of DHT11 Sensor with ESP8266 Microcontroller in Raspberry Pi:**

The DHT11 sensor is a cost-effective digital temperature and humidity sensor renowned for its accurate measurement capabilities in a given environment. This sensor significantly simplifies integration with the ESP8266 microcontroller through a single-wire interface.

The ESP8266 microcontroller connected to Raspberry PI boasts numerous smart features, including built-in Wi-Fi, support for MQTT communication, and low power consumption for efficient operation. This seamless integration of the DHT11 sensor with the ESP8266 microcontroller in Raspberry PI enhances the overall functionality of the system, enabling precise and convenient monitoring of temperature and humidity levels.

2. Message Communication via Mosquitto Broker

The MQTT Mosquitto broker is a lightweight and efficient open-source message broker that implements the MQTT protocol for message communication, following the publish-subscribe model. This model allows devices to publish messages to specific topics and subscribe to receive messages on those topics, resulting in a flexible, loosely coupled, and scalable communication architecture.
Furthermore, Mosquitto offers several features, including:

- **Retain Messages:**
  Ensures that new subscribers receive the last published message immediately upon subscription.
- **Last Will and Testament:**
  Provides a "Last Will and Testament" message, notifying all subscribers in the event of an unexpected disconnect of the publisher.
- **Quality of Service (QoS) Levels:**
  Supports different levels of Quality of Service for message delivery, offering flexibility

in message delivery guarantees.
Importantly, Mosquitto places a strong emphasis on security, offering features such as username/password authentication and SSL/TLS encryption for secure communication between clients and the broker.

3. Node-Red Dashboard:

The purpose of this component is to visualize the messages communicated by the MQTT broker in a more user-friendly manner. Additionally, it provides features to integrate with the broker for receiving published messages, utilizes a Telegram bot to alert the user when room temperature or humidity reaches the threshold limit, and notifies the user in case of an abrupt broker disconnection. Furthermore, it includes a feature to compare measured values with an external weather report API. The dashboard also offers an inline feature for user interaction.

4. Telegram Bot:

Alert the user to take action when the measured values in the room (humidity and temperature) reach the threshold limit for improved climate management. Additionally, it sends a last will message when the broker abruptly disconnects from the client, prompting the user to take corrective action to resolve the issue caused by the broker. This functionality contributes to the efficient operation of the smart system by aiding in the control of room temperature and humidity.

1.2 Project purpose:

This project aims to provide a user-friendly solution for monitoring and controlling room temperature and humidity. By integrating with an MQTT Broker for data communication, Node-RED assists in visualizing the measured values and allows users to interact with the dashboard. Additionally, it alerts the user when the values reach the threshold limit, contributing to effective room climate management.

1.3 Project Scope:

This project scope aims to implement a Smart Room Climate Monitoring System that enhances user control over room temperature and humidity for optimal comfort and energy efficiency.

Scope Elements:

1. **Sensor Integration**:
   Integrate DHT11 sensors to measure and monitor real-time room temperature and humidity.
2. **ESP8266 Microcontroller and MQTT Integration**:
   Implement the ESP8266 microcontroller to gather sensor data in Raspberry pi And Establish communication with an MQTT broker for seamless data transfer.
3. **Node-RED Dashboard**:

Develop a user-friendly and customizable dashboard using Node-RED.

4. **Alerting Mechanism**:
   Implement a notification system to alert users when temperature or humidity values exceed predefined thresholds. Utilize a Telegram bot for timely alerts.
5. **Documentation**:
   Provide comprehensive user documentation for system setup, configuration, and operation.

Requirements

The components required for setting up the IoT smart room climate monitoring system include

2.1 Hardware requirements:

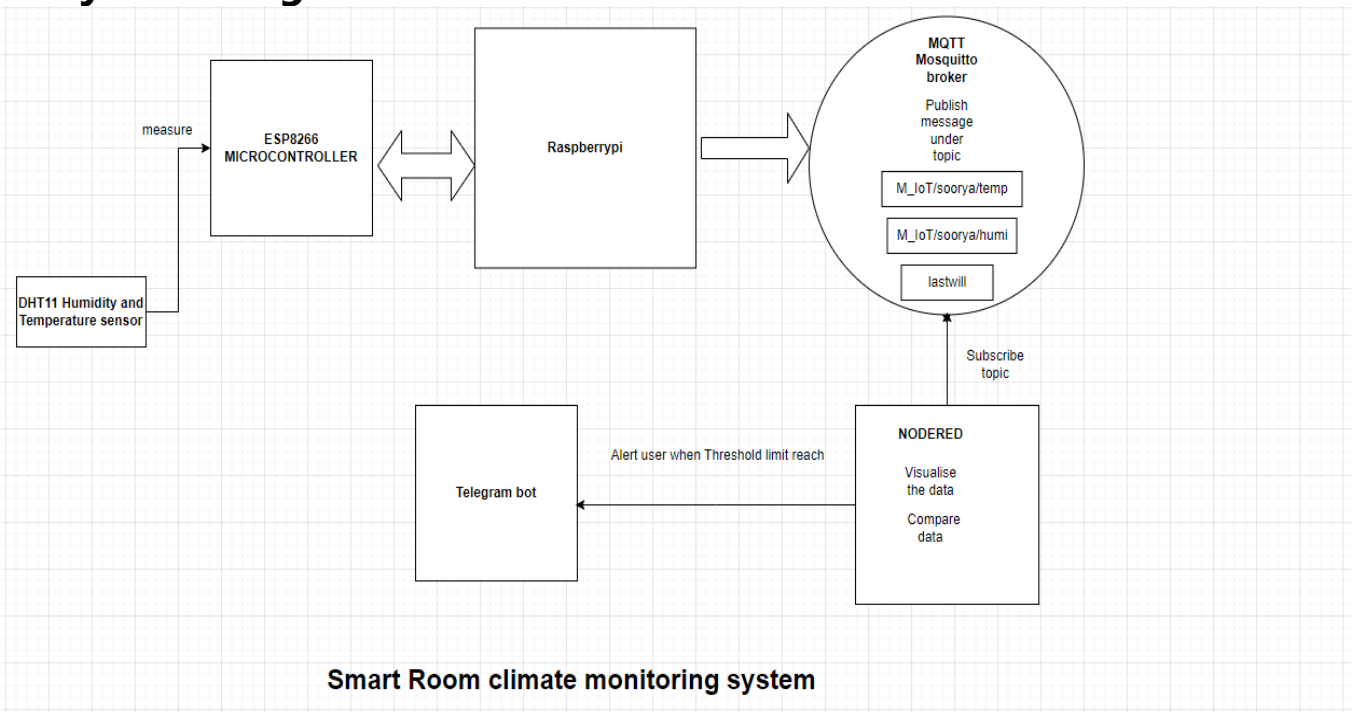
- The Raspberry Pi
- DHT11 humidity and temperature sensor
- ESP8266 Microcontroller

2.2 Software requirements:

- Node-RED
- Thonny
- Telegram

Architecture overview

3.1 System Diagram



Smart Room climate monitoring system

3.2 Component Description:

1. **DHT11 Sensor**:
   The DHT11 sensor is utilized for measuring temperature and humidity in a specific area. It can measure temperatures within the range of 0°C to 50°C with an accuracy of ±2°C and humidity levels from 20% to 90% with an accuracy of ±5% RH.

2. **ESP8266 Microcontroller**:
   The ESP8266, being a low-cost microcontroller, is well-suited for this IoT system as it involves tasks such as reading sensor data, performing simple processing, and transmitting the results. Additionally, its low energy consumption makes it an efficient choice for the IoT system.
3. **Raspberry Pi:**
   Raspberry Pi serves as the central hub in the IoT system, integrating components. An ESP8266 microcontroller reads sensor data from a DHT11 sensor connected to the Raspberry Pi via USB. The message is then transferred to a Mosquitto broker over a WLAN network, publishing data to a specific topic on the broker server.
4. **Node-RED:**
   Node-RED, installed on the Raspberry Pi, visualizes measured data and takes actions to control room climate parameters. It sends alerts to users when threshold limits are reached, enhancing the overall monitoring and control capabilities of the IoT system.
5. **Telegram Bot:**
   The Telegram bot enhances Iot system by alerting users when room humidity and temperature values reach threshold limits. It also sends a broker last will message if the broker abruptly disconnects, prompting users to take corrective actions to address potential issues caused by the broker's disconnection. This proactive approach ensures timely user intervention for maintaining optimal conditions in the monitored environment.

System Configuration

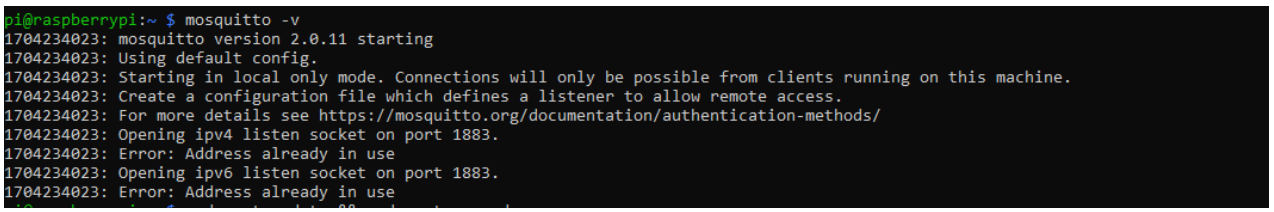
4.1 Raspberry PI setup:

Install Raspberry Pi OS, set up Wi-Fi, enable, and connect with SSH. For detailed instructions, refer to: Getting Started with Raspberry Pi.

4.2 MQTT Broker Configuration:

To install Mosquitto Broker:

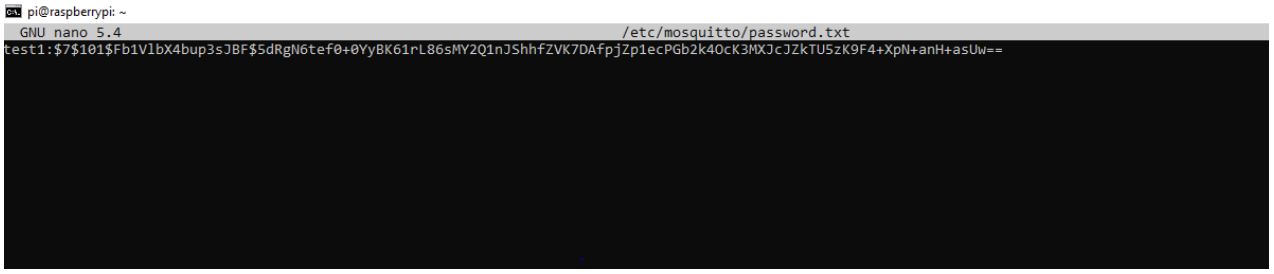
1. Open a new Raspberry Pi terminal window
2. Use Linux command, to install Mosquitto Broker :
   (sudo apt-get install mosquitto) & (apt-get install mosquitto-clients)
3. Check the version of Mosquitto to verify that the installation was successful



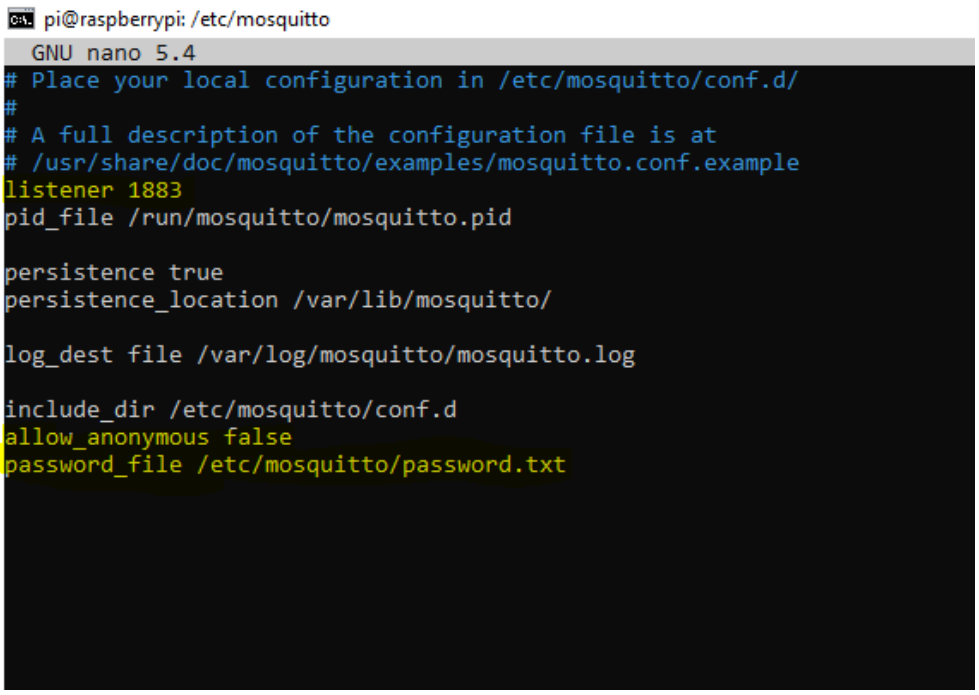
To configure Broker username and password:

1. Create a text file, i.e., password.txt, using the following command:
   sudo nano password.txt
   Enter **username: password** (e.g., test1:12345) and save the text file
2. Use the command to encrypt the password in the file
   mosquitto_passwd -U filename

After executing these steps, reopen the file to view the encrypted password.



Ensure to update mosquitto.conf file with directory of password of text file.



Make sure to restart the broker after making changes in the config file.

To stop and start the Mosquitto service, you can use the following commands:
Stop Service: sudo systemctl stop mosquitto.service
Start Service: sudo systemctl start mosquitto.service

4.3 Telegram bot Configuration:

Step 1: Create a Telegram Bot

- Open the Telegram app and search for the "BotFather" bot.
- Start a chat with BotFather.
- Use the /newbot command to create a new bot.
- Follow the instructions to choose a name and username for your bot.
- Once created, BotFather will provide you with a unique API token. Keep this token secure.

Step 2: Get Your Chat ID

To send messages or notifications to your bot, you need your chat ID.

- Search for the "userinfobot" on Telegram.
- Start a chat with the bot and click on the "Start" button.
- The bot will provide you with information, including your chat ID. Note it down.

4.4 Node Red Configuration:

1. Install node-red in raspberry. Refer https://nodered.org/docs/getting-started/raspberrypi for installation instruction
2. Start the node-red application from raspberry terminal using following command

node-red

To run as service in background:

node-red-start

3. Once node-red starts running, open it in browser and import json flow and install necessary Palette by clicking

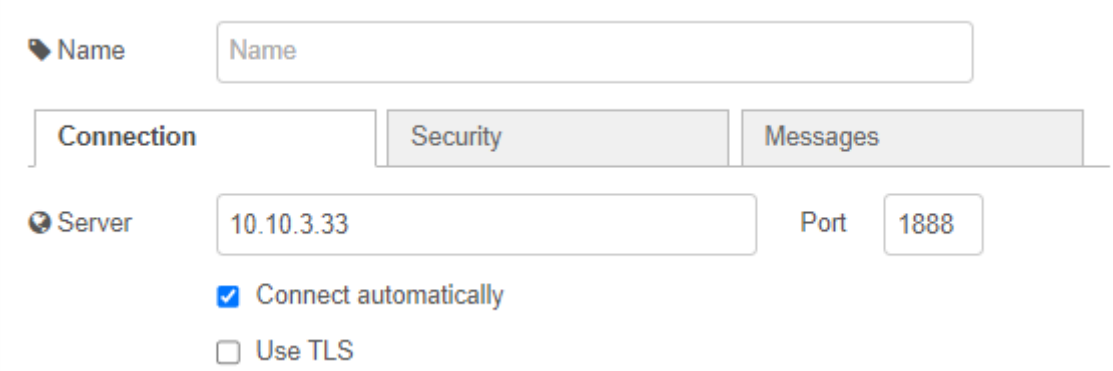
   Manage-palette->palette->install following module

   - node-red-contrib-ui-led
   - Node-red-node-ping
   - Node-red-node-openweathermap
   - Node-red-dashboard
   - Node-red-contrib-telegrambot

4. Configure the node red node:

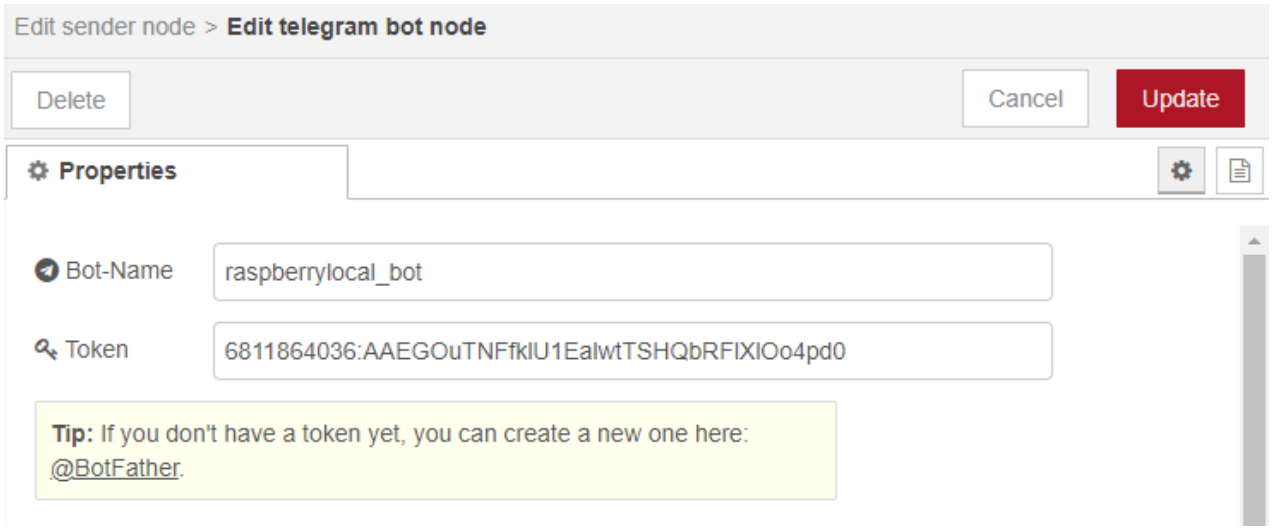
   Mosquitto broker:

   Enter server details & security (username &password)



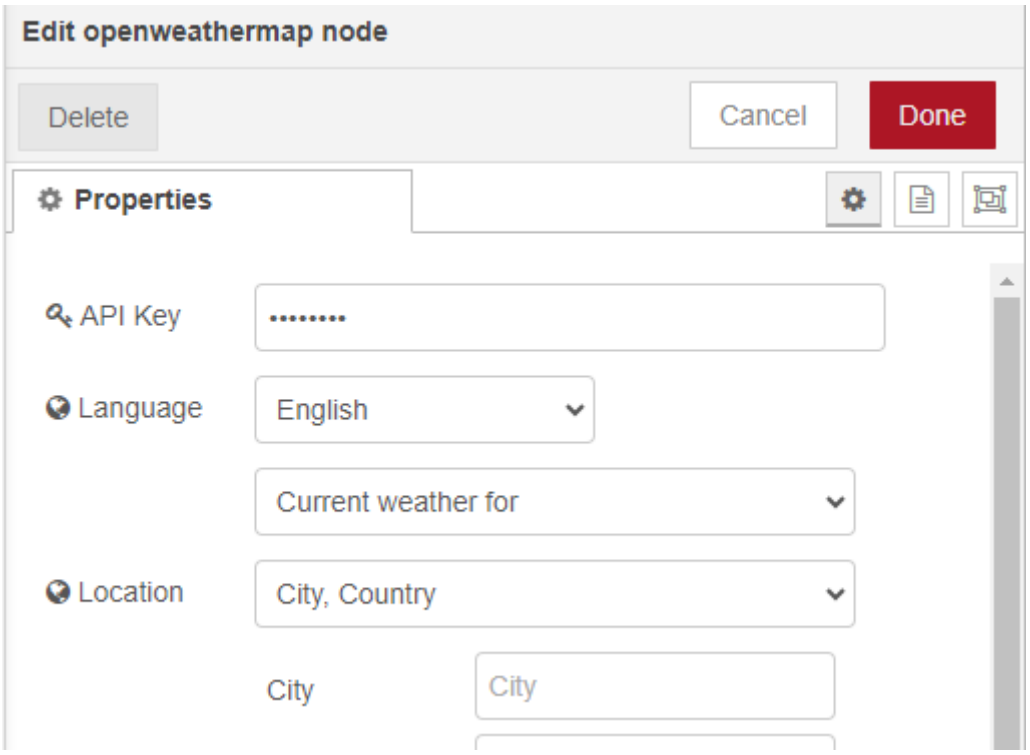
Telegram bot:

- Enter bot-name & token in telegram bot



Open weather api key:

- Create account in https://home.openweathermap.org/api_keys
- Copy the key and paste it in open weather node



4.5 Hardware Configuration:

1. **Raspberry Pi:**
   Connect the Raspberry Pi to a power source.
2. **ESP8266:**
   - Power the ESP8266 using a suitable power source (USB port from Raspberry Pi).
   - Establish communication with the ESP8266 through a USB-to-Serial adapter.
   - GPIO pins on the ESP8266 can be used for sensor connections.
3. **DHT11 (3-pin) sensor:**
   1. Connect the VCC (Power) pin to a 3.3V or 5V power source.
   2. Connect the DHT11 data pin to a GPIO pin(2) on the ESP8266.
   3. Connect the ground (GND) pin to the ground (GND) on the ESP8266.

Implementation:

5.1 Description:

Below is the python code for an MQTT publisher that needs to be stored in an ESP8266 microcontroller. This code enables the microcontroller to measure temperature and humidity and publish messages under the respective topics, utilizing the Mosquitto broker installed on a Raspberry Pi.

Modify the WiFi and broker parameters in the Python code for an MQTT publisher as per your specific requirements

If the umqtt.simple package is not already available, either download the corresponding module or run the MicroPython code for umqttsimple

```
import network
import dht
import machine
import time
import ubinascii
from umqttsimple import MQTTClient

# Wifi Parameters
WIFI_USERNAME = "Rechnernetze"
WIFI_Password = "rnFIW625"

# MQTT Server Parameters
```

```
MQTT_CLIENT_ID = ubinascii.hexlify(machine.unique_id())
MQTT_BROKER = "10.10.3.33"
MQTT_USER = "test1"
MQTT_PASSWORD = "12345"
MQTT_PORT = 1888

# Set the last will message
LAST_WILL_TOPIC = "M_IoT/soorya/lastwill"
LAST_WILL_MESSAGE = "The broker disconnected unexpectedly"
LAST_WILL_QOS = 1
LAST_WILL_RETAIN = True


def restart_and_reconnect():
    print('Failed to connect to MQTT broker. Reconnecting...')
    time.sleep(10)
    machine.reset()

def connect_mqtt():
    try:
        client = MQTTClient(MQTT_CLIENT_ID, MQTT_BROKER, user=MQTT_USER,
password=MQTT_PASSWORD, port=MQTT_PORT, keepalive=60)
        client.set_last_will(LAST_WILL_TOPIC, LAST_WILL_MESSAGE, LAST_WILL_QOS,
LAST_WILL_RETAIN)
        client.connect()
        return client
    except OSError:
        print("Error connecting to MQTT broker")
        raise

def connect_to_wifi(ssid, password):
    wifi.active(True)
    wifi.connect(ssid, password)

    while not wifi.isconnected():
        print("Connecting to Wi-Fi...")
        time.sleep(1)

    print("Connected to Wi-Fi")
    print("IP Address:", wifi.ifconfig()[0])

def main():
    print("main!")
    #connecting to wifi
```

```
    wifi = network.WLAN(network.STA_IF)
    connect_to_wifi(WIFI_USERNAME, WIFI_Password)

    # Allow only when wifi connected
    while not wifi.isconnected():
        pass

   #sensor initialiation to pinno
    sensor = dht.DHT22(machine.Pin(2))

    try:
        #connecting to broker
        client = connect_mqtt()
        print("Connected!")
        while True:
            print("inside Mosquitto broker!")
            sensor.measure()
            temp, hum = sensor.temperature(), sensor.humidity()
            print(temp)
            print(hum)
            #checking the reading value
            if isinstance(temp, (float, int)) and isinstance(hum, (float, int)):
                client.publish("M_IoT/soorya/temp", str(temp))
                client.publish("M_IoT/soorya/humi", str(hum))
                time.sleep(10)
            else:
                print("Invalid reading")
    except OSError:
        restart_and_reconnect()

if __name__ == "__main__":
    wifi = network.WLAN(network.STA_IF)
    main()
```

Steps to start the system:

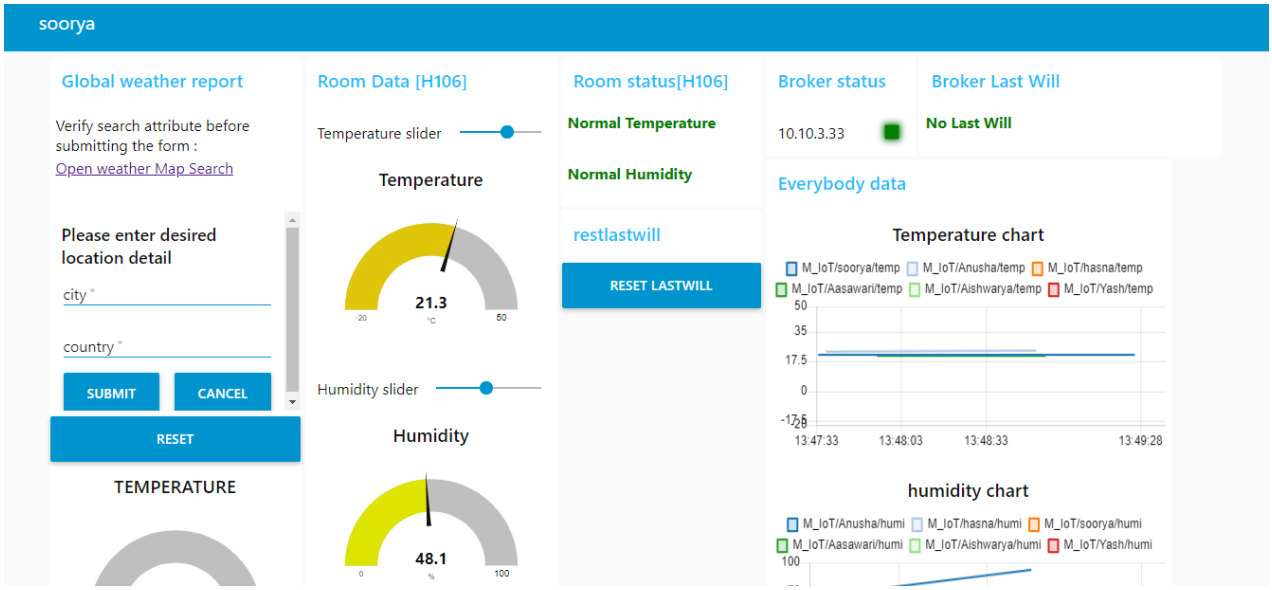
- Power up the Raspberry Pi to supply the microcontroller and execute the Python code for measuring values.
- Initiate Node-RED to automate the message flow.
- View the dashboard by navigating to the Raspberry Pi's IP address, followed by '/ui'."

Features used in this project:

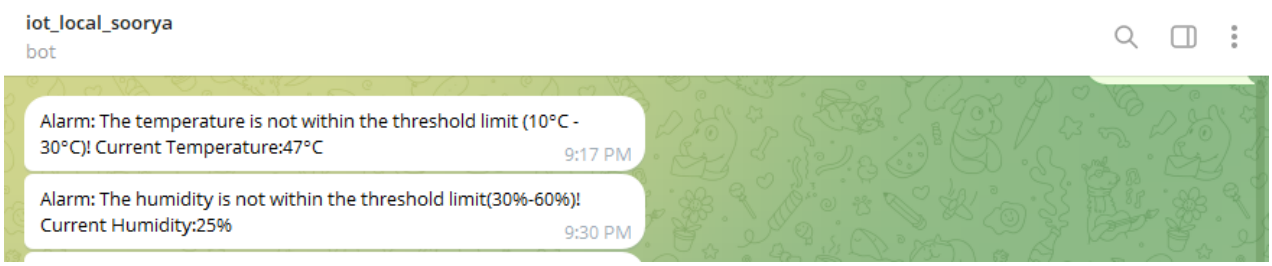
1. Utilized a local Mosquitto broker with authentication, ensuring that only authenticated individuals can access the data.
2. Implemented Last Will or Testament in the broker to publish a message when it disconnects abruptly. This ensures that subscribers are immediately notified.
3. Implemented error handling to manage incorrect sensor readings or any issues in the broker connection

5.2 Screenshot

Node red dashboard



Telegram bot message:



5.3 Limitations:

1. **Dependency on Sensors**:
The accuracy and effectiveness of the system heavily rely on the quality and calibration of the installed sensors.
2. **Continuous Maintenance**:
Regular maintenance is required to ensure the proper functioning of sensors and connected devices.
3. **Node-RED Layout Changes**:
There is a chance that the Node-RED layout may change, necessitating adjustments in the dashboard when it is not right.

Conclusion:

6.1 Summary:

In summary, the Smart Room Climate Monitoring System leverages DHT11 sensors, an ESP8266 microcontroller, and a Raspberry Pi to create an efficient IoT solution. Utilizing MQTT communication through a Mosquitto broker and visualizing data on a Node-RED dashboard enhance user control. The integration of a Telegram bot ensures timely alerts for critical changes.

While the system successfully meets its goals, acknowledging dependencies on sensor quality and the need for regular maintenance is essential. Additionally, occasional adjustments may be required due to potential Node-RED layout changes.

Online reference:

http://www.steves-internet-guide.com/mosquitto-broker/
http://www.steves-internet-guide.com/mqtt-username-password-example/
http://www.steves-internet-guide.com/mqtt-keep-alive-by-example/
http://www.steves-internet-guide.com/mqtt-last-will-example/
https://randomnerdtutorials.com/micropython-mqtt-esp32-esp8266/
https://flows.nodered.org/node/node-red-node-openweathermap