Design of Weather Tracking System Using MQTT Protocol

[1]K. Srilekha, [2]N. Sneha, [3]B. Sree Harshitha, [4]Ganesh Reddy

[1,2,3] UG Student, [4]Assistant Professor

[1,2,3,4] Department of Electronics and Communication Engineering

[1,2,3,4] Vignan's Institute of Management and Technology for Women, Kondapur (V), Ghatkesar (M), Medchal-Malkajigiri (D) – 501301

ABSRACT:

The MQTT (message queuing telemetry transport) protocol for efficient real-time data communication. The system integrates weather sensors like temperature, humidity, and pressure, transmitting data to a central server or cloud platform via MQTT. This lightweight protocol ensures low-latency, reliable communication and is ideal for remote or resource-constrained environments. The collected data is visualized on a dashboard, offering valuable insights for applications in agriculture, disaster management, and urban planning. The use of MQTT enables scalability and reliability, making the system an effective solution for modern weather monitoring. In the modern era of smart systems and connected devices, real-time weather monitoring has become increasingly important for applications in agriculture, disaster preparedness, urban planning, and environmental research.

Keywords: Raspberry Pi, MQTT, Weather Monitoring.

I. INTRODUCTION

A Weather Monitoring System is essential for collecting and analyzing environmental data such as temperature, humidity, atmospheric pressure, and rainfall. Traditional weather stations often rely on manual data collection or proprietary communication protocols, which can be limited in scalability and efficiency. To address these challenges, the MQTT (Message Queuing Telemetry Transport) protocol offers a lightweight and efficient solution for real-time data transmission in Internet of Things (IOT)-based weather monitoring systems.

The system typically includes IOT-enabled weather sensors (such as DHT11/DHT22 for temperature and humidity), an MQTT broker (e.g., Mosquitos), and a client-side application for data visualization and alerting. By using MQTT, the weather monitoring system ensures reliable, scalable, and energy-efficient communication, making it ideal for smart cities, agriculture, and remote environmental monitoring applications.

II. PROBLEM STATEMENT

With the growing need for real-time environmental monitoring, traditional weather tracking systems often face limitations such as high latency, inefficient data transmission, and poor scalability. Many existing systems rely on HTTP-based communication, which can be inefficient for applications requiring continuous, lightweight, and low-power data exchange.

To address these challenges, there is a need for an efficient, reliable, and scalable Weather Tracking System that can collect, transmit, and analyze environmental data (such as temperature, humidity, pressure, and rainfall) in real time.

III. LITERATURE SURVEY

1.Title: *IOT Based Smart Weather Monitoring System*

Link:IEEEXplore7464823

Abstract: This paper presents a smart weather monitoring system using IoT architecture, where temperature and humidity data are collected through DHT11 sensors and transmitted via the ESP8266 microcontroller. The MQTT protocol is used for lightweight, real-time data transmission to a cloud server. The system is designed for low-power and cost-efficient deployment in remote areas.

2.Title: *Smart Environment Monitoring System Using IOT and MQTT Protocol*
Link: ResearchGate
Abstract: The authors propose an environmental monitoring solution that leverages the MQTT protocol for real-time transmission of sensor data such as air quality, temperature, and humidity. Using a Raspberry Pi as the central controller, the system ensures energy-efficient and reliable data communication over constrained networks. The MQTT protocols publish-subscribe mechanism ensures scalable and asynchronous data handling. With the growing need for real-time environmental monitoring, traditional weather tracking systems often face limitations such as high latency, inefficient data transmission, and poor scalability. Many existing systems rely on HTTP-based communication, which can be inefficient for applications requiring continuous, lightweight, and low-power data exchange.

IV. EXISTING METHOD

In existing weather tracking systems, data collection and transmission are typically done using traditional communication protocols, such as HTTP (Hypertext Transfer Protocol) or FTP (File Transfer Protocol). These systems generally involve sensor nodes that collect environmental parameters like temperature, humidity, pressure, and rainfall, and send this data to a centralized server or cloud platform for processing and visualization.
Limitations of the Existing Method

1. High Overhead and Latency
2. Poor Scalability
3. Energy Inefficiency
4. Unreliable in Low-Bandwidth Networks

V. PROPOSED METHOD

The proposed weather monitoring system using MQTT offers several significant advantages over traditional HTTP-based methods. The MQTT protocol is extremely lightweight, reducing network bandwidth usage and minimizing data packet size, which is crucial for real-time and low-power applications.

Moreover, MQTT’s open-source nature and widespread community support make it highly scalable, customizable, and easy to maintain. It facilitates independent and non-blocking communication between devices, enabling distributed weather stations to operate asynchronously. The system also supports the generation of real-time alerts and notifications for extreme weather conditions, enhancing its applicability for environmental monitoring, disaster management, and smart agriculture. Overall, the MQTT-based weather monitoring system ensures efficient data transmission, scalability, reliability, and low power consumption, making it an ideal solution for modern IoT-based meteorological applications.

VI. SYSTEM ARCHITECTURE

The system architecture consists of multiple environmental sensors connected to a Raspberry Pi for real-time weather monitoring. Sensors measure wind direction, wind speed, rainfall, temperature, humidity, and air pressure.

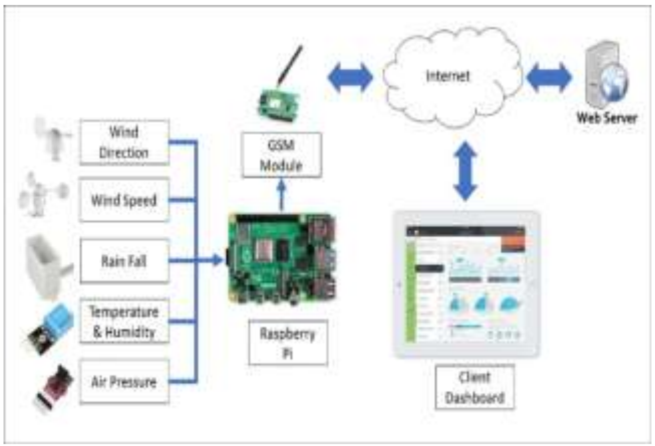


Fig 1: System Architecture

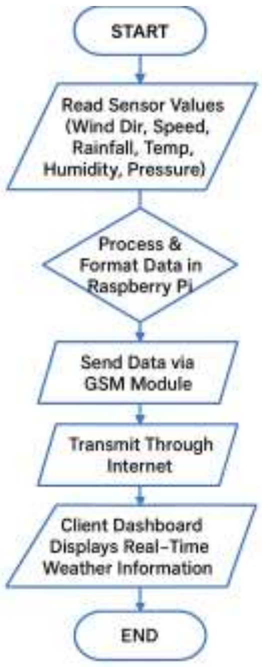
The Raspberry Pi collects and processes this data, then transmits it through a GSM module to a remote web server via the internet. The web server stores and analyzes the incoming sensor data and provides access to authorized users. A client dashboard, accessible on mobile or desktop devices, displays live weather parameters, trends, and analytics. This architecture enables continuous environmental monitoring, remote access, and efficient data visualization for end users.

VII. WORKING METHOD

The Weather Monitoring System using IoT and MQTT protocol integrates sensing, communication, and data processing technologies to provide real-time and efficient monitoring of environmental conditions. The system primarily consists of multiple functional layers, beginning with the sensor layer that measures meteorological parameters such as temperature, humidity, atmospheric pressure, rainfall, wind speed, and light intensity using sensors like DHT11, DHT22, BME280, BMP180, and LDRs. These sensors interface with a microcontroller, such as ESP8266, ESP32, or Raspberry Pi, which acts as the central processing unit. The microcontroller reads sensor data at regular intervals, converts raw readings into human-readable units, and packages the data into structured formats like JSON. It also manages network connectivity through Wi-Fi, GSM, or LoRa, and employs a lightweight MQTT client library (e.g., PubSubClient or Paho) for message handling.

The MQTT communication layer forms the backbone of data transmission, where the device establishes a TCP/IP connection to an MQTT broker and publishes sensor data to specific topics such as “weather/station1/temperature” or “weather/station1/humidity.” Subscriber systems, including dashboards or mobile applications, subscribe to these topics to receive real-time updates. The MQTT protocol is preferred due to its lightweight nature, asynchronous operation, and reliable message delivery supported by Quality of Service (QoS) levels. The broker layer, typically implemented using platforms such as Mosquito, HiveMQ, or EMQX, is responsible for receiving published messages, routing them to subscribed clients, and maintaining session persistence and authentication to ensure low-latency and reliable communication.

VIII. FLOW CHART


START
Read Sensor Values (Wind Dir, Speed, Rainfall, Temp, Humidity, Pressure)
Process & Format Data in Raspberry Pi
Send Data via GSM Module
Transmit Through Internet
Client Dashboard Displays Real-Time Weather Information
END

IX. RESULT

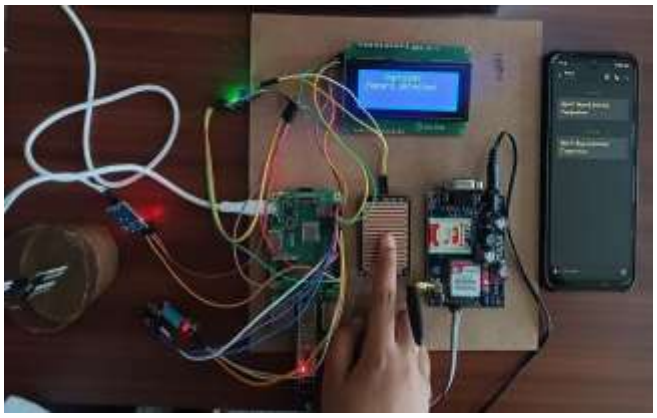


Fig 2: System detects while notifying to mobile.

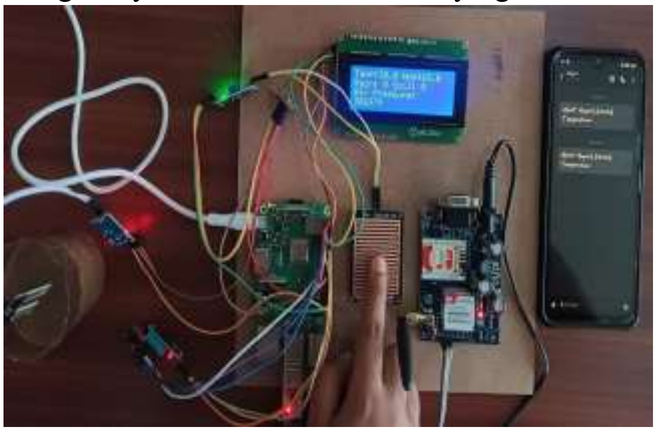


Fig 3: Readings of parameters while detecting.

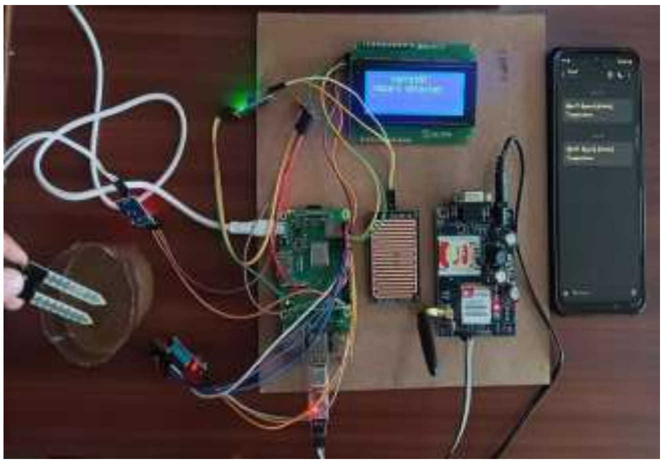


Fig 4: When system is not detecting.

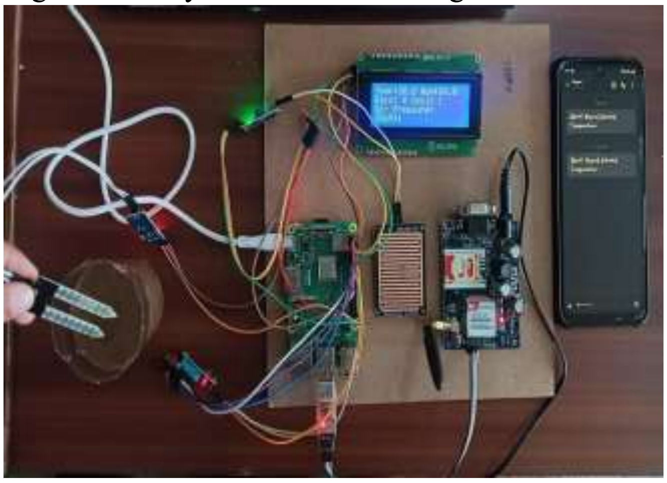


Fig 5: Readings of parameters while not detecting.

X. FUTURE SCOPE

The future of weather tracking systems using MQTT looks promising, driven by advancements in IoT, edge computing, and AI. Here are some key points: Enhanced Real-Time Analytics: Integration with AI and machine learning will enable predictive weather models and smarter decision-making directly at the edge devices. Improved Network Resilience: Development of hybrid communication methods (combining MQTT with LoRaWAN, NB-IoT) will reduce dependence on stable internet connectivity. Greater Security: Implementation of advanced encryption, authentication, and block chain-based data integrity will make MQTT-based weather systems more secure. Energy Efficiency: Low-power MQTT implementations combined with renewable energy sources will support sustainable, long-term deployment in remote areas. Wider Adoption: More smart cities, agriculture, and environmental monitoring projects will leverage MQTT for scalable, cost-effective weather tracking.

XI. CONCULISON

A weather tracking system using MQTT presents a highly efficient, lightweight, and scalable architecture for real-time environmental monitoring. At its core, MQTT (Message Queuing Telemetry Transport) is a publish-subscribe protocol designed specifically for low-power, low bandwidth communication between devices. This makes it ideal for applications such as weather monitoring, where remote sensors and embedded systems like ESP32, Arduino, or Raspberry Pi are used to collect and transmit meteorological data such as temperature, humidity, air pressure, and rainfall.

Lastly, while MQTT supports security features such as TLS encryption and authentication, these are not enforced by default. This can lead to security vulnerabilities, especially if the system is deployed over the public internet without proper safeguards. Potential risks include unauthorized access, data tampering, or denial-of-service attacks.

XII. REFERENCES

[1]. MQTT Protocol Overview o MQTT.org official site: https://mqtt.org/
Explains MQTT basics, specifications, and use cases.
[2]. Building a Weather Station with MQTT (ESP32/Arduino) o Random Nerd Tutorials — Weather Station using MQTT and ESP32: https://randomnerdtutorials.com/esp32-mqtt-publish-subscribe/ Practical step-by-step guide to sending sensor data via MQTT.
[3]. MQTT Broker Setup and Management o Mosquitos MQTT Broker: https://mosquitto.org/ Popular lightweight MQTT broker software, easy to install on Raspberry Pi or servers.
[4]. Cloud-Based MQTT Platforms o ThingsBoard — Open-source IoT Platform with MQTT support:
https://thingsboard.io/docs/user-guide/mqtt/