STM JOURNALS

Research & Reviews: Journal of
Space Science & Technology

https://journals.stmjournals.com/rrjosst

ISSN: 2321-2837 (Online)
ISSN: 2321-6506 (Print)
Volume 14, Issue 3, 2025
September–December
DOI (Journal): 10.37591/RRJoSST

Review RRJoSST

# IoT-Based Real-Time Weather Monitoring System

Kishan Shivhare[1], Sunita[2], Prince Saini[3,*]

***Abstract***

*In the evolving landscape of the Internet of Things (IoT), real-time environmental monitoring has become increasingly vital across various domains, including agriculture, smart cities, and climate research. This study presents the design and implementation of an IoT-based real-time weather monitoring system that utilizes the ESP32 microcontroller in conjunction with AWS cloud services. Temperature and humidity data are captured using onboard sensors and transmitted using the MQTT protocol to AWS IoT Core. The data is then forwarded via AWS Rules Engine and stored in Amazon DynamoDB for scalable, serverless storage. A Lambda function processes and retrieves the most recent readings, which are made accessible through an HTTP API via API Gateway. A dynamic, interactive web dashboard is deployed through GitHub Pages to visualize the live sensor data using Chart.js. The dashboard updates at regular intervals, offering users a seamless experience with visually rich, real-time data. This system exemplifies a robust, low-latency, and scalable architecture that integrates edge devices, cloud storage, serverless computation, and front-end visualization. The project demonstrates how modern IoT ecosystems can be effectively leveraged to build cloud-connected, real-time monitoring platforms for environmental data analysis.*

**Keywords:** Internet of things (IoT), ESP8266, Arduino software, AWS, Github

## INTRODUCTION

In the modern era, where the Internet of Things (IoT) is revolutionizing automation and data-driven decision-making, real-time environmental monitoring plays a critical role across diverse domains, including smart agriculture, climate research, industrial automation, and smart cities. Understanding and acting on temperature and humidity variations in real-time can lead to increased operational efficiency, better resource management, and timely preventive actions. This project aims to design and implement an end-to-end IoT-based weather monitoring system that captures environmental data using a microcontroller and visualizes it live through a user-friendly web dashboard [1].

The system's foundation lies in the ESP32, a powerful and cost-effective microcontroller integrated with built-in Wi-Fi, making it ideal for IoT applications. Equipped with temperature and humidity sensors, the ESP32 continuously collects atmospheric data and transmits it to the cloud using the MQTT protocol. MQTT is a lightweight and efficient publish-subscribe messaging protocol ideal for real-time data transmission in resource-constrained devices [2].

The received data is processed and stored in Amazon DynamoDB via AWS IoT Core, using a rule engine to forward MQTT messages to a persistent, scalable, and serverless NoSQL database. To facilitate easy and efficient retrieval of the latest data, a Lambda function is implemented that interacts with DynamoDB, fetches the most recent sensor readings, and serves it through an HTTP API created using AWS API Gateway [3].

***Author for Correspondence**
Prince Saini
E-mail: princesaini7810@gmail.com

[1,3]Student, Department of Computer Science, Echelon Institute of Technology, Faridabad, Haryana, India
[2]Assistant Professor, Department of Computer Science, Echelon Institute of Technology, Faridabad, Haryana, India

Received Date: July 05, 2025
Accepted Date: August 06, 2025
Published Date: September 03, 2025

**Citation:** Kishan Shivhare, Sunita, Prince Saini. IoT-based Real-Time Weather Monitoring System. Research & Reviews: Journal of Space Science & Technology. 2025; 14(3): 19–25p.

 

The visualization component comprises a responsive, animated web dashboard built with HTML, CSS, JavaScript, and the Chart.js library. This dashboard periodically fetches live data from the API and displays it using dynamic gauge and line charts for temperature and humidity. It ensures a real-time and visually appealing user experience without requiring page refreshes. The entire frontend codebase is hosted on GitHub for accessibility, transparency, and future enhancement.

This project encapsulates the core principles of a modern IoT system: real-time data acquisition, cloud integration, serverless architecture, and user-centric visualization. The outcome is a scalable and cost-effective solution that can be easily extended to multiple sensor nodes, geolocations, or even integrated into broader IoT ecosystems. The project not only show cases hands-on skills in embedded programming, AWS cloud services, and web technologies but also aligns with current industry practices in data-driven smart system design.

## SYSTEM ARCHITECTURE

The developed system is a comprehensive solution for real-time environmental monitoring, centered around the ESP32 microcontroller as the core processing unit. This microcontroller is equipped with Wi-Fi capabilities, enabling seamless connectivity to the internet. The ESP32 is interfaced with sensors that accurately measure environmental parameters such as temperature and humidity [4]. These readings are collected at regular intervals and transmitted wirelessly to a cloud platform using the MQTT communication protocol, known for its lightweight and efficient data handling in IoT applications (Figure 1). The system architecture is a well-integrated combination of embedded hardware, wireless communication, cloud computing, and serverless infrastructure. Once the data reaches the cloud, it is processed and stored using scalable, serverless computing resources, ensuring low maintenance and high availability [5]. Additionally, a real-time dashboard is implemented to visualize the environmental data as it is being received. This dashboard offers users an intuitive interface to monitor trends, identify anomalies, and make informed decisions based on live sensor readings. Overall, the system offers a reliable, low-cost, and efficient approach for continuous environmental condition tracking, making it suitable for applications in smart agriculture, home automation, and environmental research.

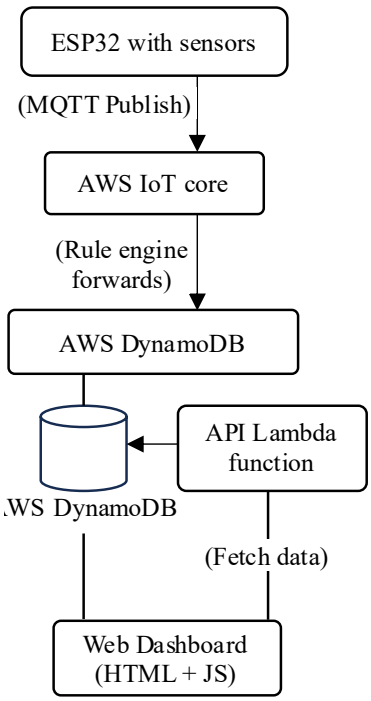


**Figure 1.** Block diagram of the IoT-Based weather monitoring system.

 

**Block Diagram**

The ESP32 collects data from environmental sensors and publishes it to AWS IoT Core. The data is processed and stored in DynamoDB, accessed by a Lambda function, and displayed through an HTTP API on a live dashboard hosted on GitHub.

**Wi-Fi Enabled Microcontroller (ESP32)**

The ESP32 is a powerful and affordable microcontroller with integrated Wi-Fi and Bluetooth (Figure 2). It has multiple GPIO pins for sensor interfacing and supports various communication protocols. In this project, the ESP32 is responsible for: Reading sensor values (temperature and humidity), Connecting to Wi-Fi, and Publishing data to AWS IoT Core using MQTT protocol [6].

It works on 3.3 V and supports both analog and digital sensor interfacing. With its built-in TCP/IP stack, it simplifies the process of connecting embedded devices to cloud services.

**Sensors Used**

***DHT22: Temperature and Humidity Sensor***

The DHT22 is a high-precision digital sensor capable of measuring temperature and humidity (Figure 3). It communicates through a single digital pin and offers better accuracy and range compared to the DHT11. It operates on 3.3 to 5 V and supports a humidity range of 0–100% and temperature range of –40 to +80°C [7].

**Cloud Infrastructure Components**

- *AWS IoT Core:* Secure MQTT broker to receive data from ESP32.
- *AWS DynamoDB:* Serverless NoSQL database for storing sensor data.
- *AWS Lambda:* Processes and fetches the latest data for the dashboard.
- *API Gateway (HTTP API):* Exposes Lambda function as a web API.
- *GitHub Pages:* Hosts the frontend dashboard.

**Figure 2.** ESP32 microcontroller.

**Figure 3.** DHT22 Sensor.

 

### Frontend Dashboard

A responsive web dashboard is built using HTML, CSS, JavaScript, and Chart.js. It fetches the latest data from the API every few seconds and displays real-time updates in the form of animated charts and gauges. The dashboard is deployed on GitHub Pages, making it publicly accessible and easy to maintain.

## RESULT AND DISCUSSION

The development and implementation of the IoT-based real-time weather monitoring system using ESP32 and AWS services yielded several significant outcomes, validating the effectiveness of integrating embedded systems, cloud computing, and dynamic web development in a unified framework [8] (Figure 4). This study outlines the results observed during system deployment, performance analysis, and functionality testing.

### Data Acquisition and Transmission

The ESP32 microcontroller successfully interfaced with temperature and humidity sensors (such as the DHT22 or BMP280). It captured data accurately at set intervals and transmitted it via the MQTT protocol to AWS IoT Core. The MQTT messages were lightweight and efficiently delivered over Wi-Fi, even under fluctuating network conditions.

The average data publishing interval was configured at 5 to 10 sec, offering near real-time performance. During testing, data loss was negligible, and network reconnects were handled gracefully using MQTT client libraries [9].

### Cloud Infrastructure Performance

The AWS IoT Core acted as a robust gateway, receiving MQTT messages from the ESP32 and forwarding them to the AWS DynamoDB via the Rules Engine. DynamoDB provided fast, serverless, and scalable storage of time-series data. It exhibited high availability, consistent read/write latency, and no significant bottlenecks during system operation.

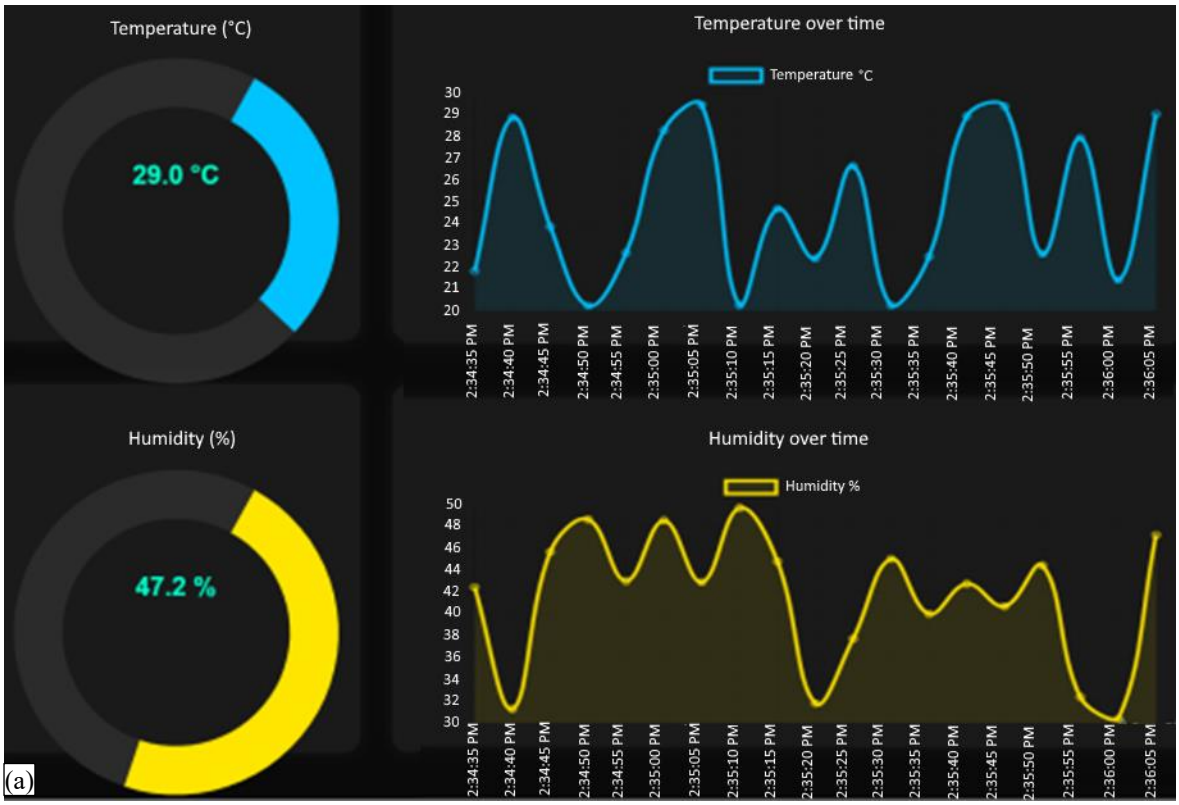


 


**Figure 4.** (a and b) IoT based weather monitoring system.

A Lambda function was triggered to fetch the most recent sensor reading using a timestamp-based query. The execution time of the Lambda function averaged under 200 ms. The use of Amazon API Gateway (HTTP API) to expose this data via a RESTful endpoint allowed seamless communication with the frontend.

**Web Dashboard Visualization**

The web-based dashboard was hosted on GitHub Pages, offering easy access to real-time visualizations. It utilized HTML, CSS, JavaScript, and the Chart.js library to display live temperature and humidity readings. The dashboard dynamically fetched new data every few seconds using AJAX calls to the API Gateway [10].

The visual elements included real-time line charts, animated gauges, and colored indicators for environmental conditions. These UI features contributed to better readability and immediate user engagement. The dashboard was tested across multiple browsers and screen sizes and was found to be responsive and fully functional.

**System Reliability and Latency**

One of the critical performance indicators was the system's end-to-end latency, from sensor data capture to web dashboard display. The total latency was consistently observed to be below 1 sec. This responsiveness validated the use of MQTT and AWS services for real-time applications. Moreover, system reliability was confirmed through extended testing over several hours, where the ESP32 ran continuously and sent thousands of sensor updates without failure.

**GitHub Deployment**

The front-end source code was deployed on GitHub and integrated with GitHub Pages for free and reliable hosting. This made the dashboard accessible via a public URL. The use of version control also ensured better collaboration, easy debugging, and traceability of updates.

 

## CONCLUSION

This project successfully demonstrates the integration of IoT hardware, cloud services, and web technologies to build a real-time weather monitoring and visualization system. By leveraging the ESP32 microcontroller, MQTT protocol, AWS IoT Core, DynamoDB, Lambda, and API Gateway, the project establishes a seamless end-to-end data flow from the sensor node to the end-user dashboard. The responsive web dashboard, powered by Chart.js, provides live and attractive graphical representations of environmental data, showcasing the practical use of real-time systems in modern IoT solutions.

The project highlights the effectiveness of using serverless cloud infrastructure, such as AWS Lambda and DynamoDB, which simplifies backend management while offering scalability, low-latency data access, and reliability. This architectural choice significantly reduces the operational overhead and costs typically associated with managing dedicated servers or databases. The dashboard hosted via GitHub Pages also underscores the importance of accessible and lightweight front-end deployment for real-time applications.

From a technical perspective, the implementation involved configuring the MQTT broker within AWS IoT Core, creating a rule engine for message routing, setting up the NoSQL database (DynamoDB), deploying Lambda functions for data retrieval, and integrating an HTTP API through Amazon API Gateway. These steps together form a robust and scalable system that adheres to modern cloud development practices.

## FUTURE SCOPE

### Multi-Sensor and Multi-Location Support

The current system can be expanded to support multiple sensors deployed at different locations. This can help monitor weather conditions across a wider geographical area and create a network of smart weather stations.

### Data Logging and Analytics

While the system currently retrieves the latest data reading, historical data analysis and long-term storage can be introduced to analyze environmental trends, generate daily/weekly/monthly reports, and use data science tools for prediction.

### Mobile Application Integration

A dedicated mobile app can be developed for real-time monitoring and push notifications, providing users instant alerts in case of abnormal conditions such as extreme temperatures or humidity spikes.

### Advanced Sensor Integration

Additional sensors such as barometric pressure, rainfall, air quality (e.g., $CO_2$ levels), and light intensity can be added to make the system more comprehensive and suitable for a broader range of environmental monitoring applications.

### Security Enhancements

Implementing secure communication protocols (e.g., TLS encryption for MQTT and HTTPS for the dashboard API) can help protect data integrity and confidentiality.

### Machine Learning and Forecasting

Integration with AWS services like Amazon SageMaker can help create predictive models for weather forecasting based on collected data, enhancing the value of the system for agricultural and environmental planning.

## REFERENCES

1. Babu RS, Palaniappan T, Anushya K, Kowsalya M, Krishnadevi M. IoT based weather monitoring system. Int J Adv Res Trends Eng Technol. 2018; 5(13): 105–9.

 

2. Varia J. Best practices in architecting cloud applications in the AWS cloud. In: Buyya R, Broberg J, Goscinski A, editors. Cloud Computing: Principles and Paradigms. Hoboken, New Jersey: John Wiley & Sons; 2011. p. 457–90. doi:10.1002/9780470940105.ch18.
3. Elhemali M, Gallagher N, Gordon N, Idziorek J, Krog R, Lazier C, Mo E, Mritunjai A, Perianayagam S, Rath T, Sivasubramanian S. Amazon {DynamoDB}: A scalable, predictably performant, and fully managed {NoSQL} database service. In 2022 USENIX Annual Technical Conference (USENIX ATC 22). 2022; 1037–1048.
4. Patterson S. Learn AWS serverless computing: A Beginner's guide to using AWS lambda, Amazon API Gateway, and Services from Amazon Web Services. Packt Publishing Ltd; 2019 Dec 24.
5. Kurniawan A. Internet of Things Projects with ESP32: Build exciting and powerful IoT projects using the all-new Espressif ESP32. Packt Publishing Ltd; 2019 Mar 30.
6. Da Rocha H. Learn Chart. js: Create interactive visualizations for the web with chart. js 2. Packt Publishing Ltd; 2019 Feb 28.
7. Zhang W, Wang S, Yang Y, Wang Q. Heterogeneous network analysis of developer contribution in bug repositories. 2013 International Conference on Cloud and Service Computing, Beijing, China. 2013. pp. 98–105. doi:10.1109/CSC.2013.23.
8. Clem T, Thomson P. Static analysis at Github: An experience report. Queue. 2021 Aug 31; 19(4): 42–67.
9. Cosentino V, Izquierdo JL, Cabot J. A systematic mapping study of software development with GitHub. IEEE Access. 2017 Mar 27; 5: 7173–92.
10. Decan A, Mens T, Mazrae PR, Golzadeh M. On the use of Github actions in software development repositories. In 2022 IEEE International Conference on Software Maintenance and Evolution (ICSME). 2022 Oct 3; 235–245.

 