NATIONAL ENGITECH DIGEST

VOL. 3 ISSUE 1

e-ISSN: 3030-5969

Cloud-Based Remote Network Monitoring and Alert System

Siti Salmah Md Kassim[1], Ahmad Faiz Danish Bin Ahmad Yuszairy[2], Muhd Naqib Bin Kahar[3], Muhd Amir Uqail Bin Syamsulariff[4]

[1234] Jabatan Teknologi Maklumat & Komunikasi, Politeknik Ungku Omar, Perak, Malaysia

sitisalmah@puo.edu.my, ahmadfaizdanish00@gmail.com, naqibkaha560@gmail.com, amiruqail324@gmail.com

Abstract

The increasing dependence on network infrastructure requires continuous monitoring of both network devices and server room environmental conditions to maintain service availability and equipment security. This project presents a Cloud-Based Remote Network Monitoring and Alert System that integrates Internet of Things (IoT) hardware, cloud computing, and automated notifications into a centralized monitoring platform. The system employs an ESP32 microcontroller paired with a DHT22 temperature and humidity sensor and a PIR motion sensor to monitor ambient conditions and physical intrusion. Simultaneously, a Python-based monitoring script continuously evaluates the availability of virtual network devices—including a router, switch, Windows Server 2019, and Windows 10 Client hosted on VMware Workstation—via ICMP ping requests. Data is transmitted to the Firebase Realtime Database and displayed through a web dashboard built with HTML, CSS, JavaScript, the Firebase SDK, and Chart.js. The Telegram Bot API delivers automated alerts when network devices become unavailable, environmental thresholds are exceeded, or motion is detected, while EmailJS logs administrator login events. Experimental results confirmed successful data synchronization, accurate device status detection, an average sensor-to-cloud latency of under two seconds, and reliable alert delivery. While functional in a local testbed, scaling to larger enterprise environments requires further evaluation.

Keywords: IoT; Firebase Realtime Database; ESP32; Network Monitoring; Telegram Alert.

Introduction

Modern organizational operations rely heavily on stable network infrastructure, requiring routers, switches, and server nodes to remain continuously operational. Beyond network availability, administrators must monitor environmental factors within server rooms, such as ambient temperature, humidity levels, and physical access, as adverse conditions can impair hardware performance or cause physical damage (Tanenbaum & Wetherall, 2021). Besides monitoring network availability, administrators must also monitor server room environmental conditions such as temperature, humidity, and unauthorized physical access, as these factors may affect hardware performance and security (Wardana, R. W., Wastumirad, A. W., Rusanto, B. H., & Susanto, D., 2026).

Conventional monitoring systems often separate network monitoring from environmental monitoring, requiring administrators to use multiple applications. This fragmented approach delays incident detection and troubleshooting because network failures and environmental abnormalities are monitored independently.

IoT technology enables sensors and microcontrollers to collect environmental information and transmit it through cloud platforms in real time. ESP32 has become a popular IoT controller because it provides built-in Wi-Fi connectivity and low power consumption. Firebase Realtime Database further enables real-time synchronization between hardware devices and web applications, making centralized remote monitoring practical.

Therefore, this project proposes a cloud-based monitoring system that combines environmental sensing, network availability monitoring, dashboard visualization, and automated alerts into one integrated platform.

The objectives of this project are:

1. To design an IoT-based remote monitoring system for network devices and the server room environment.
2. To develop a cloud-based dashboard for real-time monitoring.
3. To test an automated notification system for network failures, environmental hazards, and motion detection.

Methodology

This project adopted the Agile Development Methodology because it supports iterative development and testing of hardware, software, cloud integration, and dashboard components.

The development process consists of five stages:

1. Requirement Analysis – Identify monitoring parameters including network device status, temperature, humidity, and motion detection.
2. Hardware Development – Develop an IoT monitoring node using ESP32, DHT22, and PIR sensor (Aosong Electronics, n.d.).

3. Software Development – Develop Arduino firmware for ESP32 (Espressif Systems, 2023) and a Python monitoring script using the subprocess library to perform continuous ping monitoring.
4. Cloud Integration and Dashboard Development – Store monitoring data in Firebase Realtime Database (Google Firebase, 2024) and display it using HTML, CSS, JavaScript, Firebase SDK, and Chart.js.
5. System Testing – Simulate network failures, environmental hazards, and motion events to verify dashboard updates and notification delivery.

Results and Discussion

The developed system successfully achieved the three project objectives through hardware integration, real-time cloud monitoring, and automated notifications.

Objective 1: IoT-Based Monitoring System

The ESP32 successfully connected to the Wi-Fi network and transmitted temperature, humidity, and motion data to Firebase Realtime Database. Environmental updates appeared in Firebase with an average latency below 2 seconds during testing.



Figure 1. Firebase Realtime Database Receiving Live Sensor Data

Latency Measurement Methodology: To quantify data transfer latency, time-stamped log messages were generated at the ESP32 edge node upon reading sensor values and compared against the server-side timestamp recorded upon writing to the Firebase Realtime Database (Latency = $T_{Firebase} - T_{ESP32}$). Across 50 consecutive telemetry update cycles, the mean round-trip transmission latency remained consistently below 2 seconds (mean = 1.42 seconds) under stable network conditions.

The Python monitoring script successfully tracked the virtualized network devices hosted in VMware Workstation. When the Windows Server or Windows Client virtual machine was powered off, its status changed immediately to "Offline" in Firebase.

Objective 2: Cloud-Based Dashboard

The dashboard displayed live monitoring information from Firebase without requiring page refreshes. Refer to Figure 2 to see the dashboard interface design.

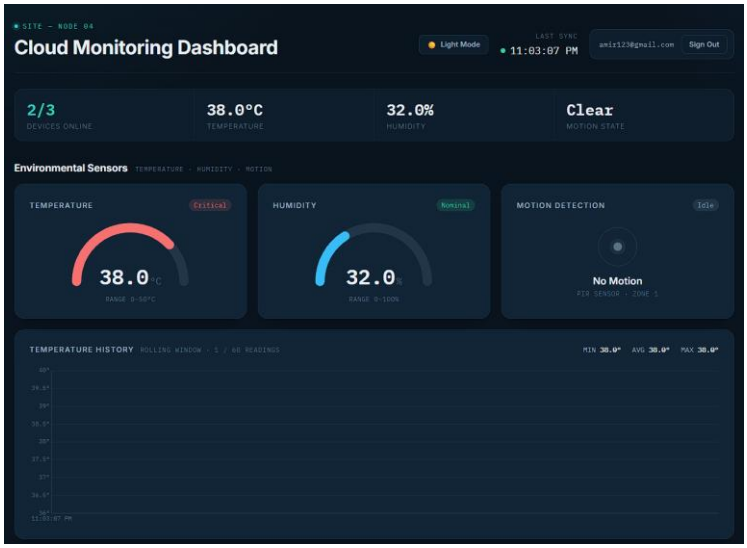


Figure 2. Cloud-Based Monitoring Dashboard

The dashboard includes:

- Temperature gauge.

- Humidity gauge.
- Motion detection status.
- Router, Switch, Server, and Client status indicators.
- Historical temperature graph using Chart.js.

Objective 3: Notification System Testing

Telegram Bot API successfully delivered notifications during simulated abnormal events. Figure 3 shows the interface in telegram bot alert notification. The notification system was evaluated using four test scenarios which depicts in Table 1.

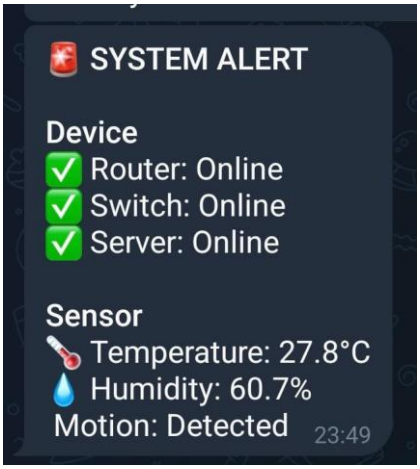


Figure 3. Telegram Alert Notifications

Table 1. Summary of Notification System Testing and Results

Testing Scenario	Expected Output	Result
Windows Server VM Shutdown	SERVER OFFLINE Telegram notification	Successful
Temperature exceeds 35°C	HIGH TEMPERATURE Telegram alert	Successful
PIR Sensor detects movement	MOTION DETECTED Telegram alert	Successful
Dashboard administrator login	Email audit notification	Successful

System Limitations and Testing Constraints

While test results confirmed system functionality, several limitations were observed during evaluation:

- Network Scope and Scalability: System testing was restricted to a single local Wi-Fi network and a limited host environment within VMware Workstation. In enterprise deployments with hundreds of managed endpoints, centralized ICMP ping polling may introduce local host overhead or generate network traffic spikes. Broad claims regarding enterprise scalability require further verification across routed, multi-subnet topologies.
- Sensor Accuracy and Rate Limits: The DHT22 sensor exhibits a sampling interval constraint of approximately 2 seconds. Rapid temperature fluctuations may not be captured instantaneously due to physical sensor response times.
- Cloud Dependency: The current prototype relies fully on active internet connectivity to reach Firebase and Telegram APIs. A local gateway outage halts remote alerting, highlighting the need for local fallback notification channels (such as local buzzer alarms or SMS modules).

Conclusion

This project successfully developed a Cloud-Based Remote Network Monitoring and Alert System that integrates IoT sensors, virtualization-based network monitoring, Firebase cloud services, and automated notifications into a centralized monitoring solution.

The IoT monitoring system successfully collected temperature, humidity, and motion data using ESP32 and transmitted the information to Firebase in real time. The cloud dashboard successfully displayed live environmental readings, historical temperature graphs, and online/offline status for monitored network devices. Telegram notifications and EmailJS audit emails successfully informed administrators about network failures, environmental abnormalities, motion detection, and dashboard login activities.

Overall, the project achieved all objectives and demonstrated an effective and affordable solution for remote monitoring of network infrastructure and server room environments within the tested local testbed. Scalability

to larger enterprise environments has not yet been evaluated. Future work should improve sensor accuracy, network resilience, and monitoring capabilities, and should evaluate the system in larger enterprise environments.

References

Aosong Electronics. (n.d.). *AM2302/DHT22 digital relative humidity & temperature sensor*. https://www.sparkfun.com/datasheets/Sensors/Temperature/DHT22.pdf

Espressif Systems. (2023). ESP32 Technical Reference Manual. https://www.espressif.com/sites/default/files/documentation/esp32_technical_reference_manual_en.pdf

Google Firebase. (2024). Firebase Realtime Database Documentation. https://firebase.google.com/docs/database

Tanenbaum, A. S., & Wetherall, D. J. (2021). Computer Networks (6th ed.). Pearson. https://www.pearson.com/en-us/subject-catalog/p/computernetworks/P200000003188

Wardana, R. W., Wastumirad, A. W., Rusanto, B. H., & Susanto, D. (2026). IoT-based automatic environmental monitoring and network device electrical protection system in a server room. Internet of Things and Artificial Intelligence Journal, 6(3), 670–687. https://pubs.ascee.org/index.php/iota/article/view/1200