Journal of Computation Physics and Earth Science
ISSN: 2776-2521 (online)
Volume 5, Number 2, February 2026, Page 145 - 151
https://journal.physan.org/index.php/jocpes/index

Design and Implementation of a React.js-Based Monitoring Dashboard for Tangerang Weather Radar and Server Room Air Conditioning Control

Muhammad Fadil Pradana Andi Alief[1], Nadila R Muhiddin[2], Bintoro Puspo Adi[3], Kadek Ari Sudama[4]
[1,2]Undergraduate Program in Applied Instrumentation Meteorology, Climatology, and Geophysics (STMKG)
[3,4]Indonesian Agency for Meteorology, Climatology, and Geophysics (BMKG)

Article Info

Article history:

Received December 30, 2025
Revised March 7, 2026
Accepted March 8, 2026

Keywords:

Weather Radar
Server Room Monitoring
React.js
Node.js
System Integration

A B S T R A C T

The Tangerang Weather Radar Station plays a critical role in aviation safety, yet its maintenance workflow is currently fragmented. Technicians must monitor radar telemetry and control server room cooling systems using separate, non-integrated platforms. This study addresses this inefficiency by developing a centralized web dashboard based on the React.js and Node.js framework. The system integrates real-time BITE (Built-In Test Equipment) data from the radar and provides remote control for the air conditioning units via an existing IoT infrastructure. Testing results demonstrate that the dashboard accurately renders telemetry parameters (High Voltage Power Supply, Transmitter Power, and VSWR) and executes AC control commands with minimal latency within the local network environment. The responsive design ensures accessibility across desktop and mobile devices, providing a unified "Single Pane of Glass" solution for maintaining optimal radar operation and server environmental conditions.

Corresponden Author:

Muhammad Fadil Pradana Andi Alief,
Undergraduate Program in Applied Instrumentation Meteorology, Climatology, and Geophysics (STMKG)
Tangerang, Indonesia
Email: fadilpradana.student@stmkg.ac.id

1. INTRODUCTION

Indonesia's Agency for Meteorology, Climatology, and Geophysics (BMKG) plays a pivotal mandate in ensuring aviation safety in Indonesia by providing fast, precise, and accurate weather information [1]. To support this mission, the Directorate of Aviation Meteorology operates a network of advanced observation infrastructure, including weather radars. The Soekarno-Hatta Class I Meteorological Station relies heavily on the Tangerang Weather Radar to provide real-time weather data, which is crucial for early detection of significant weather phenomena [2]. Any disruption in radar operations can significantly impact vital decision-making processes regarding flight safety.

The continuity of radar operations (uptime) is strictly dependent on the critical supporting infrastructure in the server room, particularly the Air Conditioning (AC) system. Radar servers generate significant heat and operate within tight temperature tolerances. Failure of the cooling system can lead to overheating and subsequent server failure, making environmental monitoring and control a necessity to ensure device reliability [3] [4].

Currently, the operational workflow for technicians at the Tangerang Weather Radar is fragmented and inefficient. Radar operational data is monitored through a proprietary instrument system (EdgeBite) [5], while the control of vital infrastructure, such as the server room AC, is performed through a completely different platform: a Telegram Bot connected to an existing Internet of Things (IoT) device based on the NodeMCU microcontroller. This separation of instrument monitoring and infrastructure control into two distinct platforms creates operational inefficiencies and slows down response times in the event of temperature anomalies.

Furthermore, relying on fragmented systems introduces critical delays in mitigating thermal anomalies. Prolonged radar downtime due to server overheating directly results in observational "data gaps." In aviation meteorology, these data gaps can critically obscure the continuous detection of hazardous weather phenomena, such as severe wind shear or microbursts. Therefore, bridging the research gap between proprietary radar software and independent IoT tools into a unified platform is not merely an operational convenience, but a vital necessity to ensure zero-downtime meteorological surveillance.

To address this issue, this study proposes the design and implementation of a centralized, integrated web dashboard. Leveraging modern web technologies such as React.js and Node.js, this research aims to build a unified interface that retrieves real-time data from the radar database and integrates it with the existing AC control functionality. Unlike previous studies that focus on hardware construction, this research emphasizes system integration—bridging the gap between radar telemetry and the pre-installed IoT infrastructure to create a "single source of truth" for station technicians. Therefore, this study aims to design and implement an integrated web-based dashboard that unifies radar telemetry monitoring and server room AC control into a single operational interface.

Despite the existence of radar monitoring tools and IoT-based server room monitoring systems, there is still a lack of integrated platforms that unify radar telemetry monitoring with infrastructure control within a single operational dashboard. This research addresses this gap by integrating proprietary radar telemetry data with IoT-based environmental control systems into a unified web-based monitoring interface.

2. RELATED WORK

Research on the utilization of weather radar data and the development of server infrastructure monitoring systems has been extensively conducted in separate domains. In the field of meteorological observation, Wei and Hsu [2] conducted a study on the application of real-time radar reflectivity for early detection and weather forecasting. Their research, which focused on Volume Coverage Pattern (VCP) scanning strategies, demonstrated that rapid and accurate integration of radar data is crucial for risk mitigation decision-making. This finding underscores the necessity of real-time data visualization in operational environments. Furthermore, Hsu et al. [6] explored the concept of system integration by developing a unified surveillance platform that combined weather radar imagery with CCTV and water level sensors for urban flooding monitoring. This approach of integrating meteorological instruments with environmental monitoring devices serves as a foundational concept for the dashboard developed in this study.

Regarding infrastructure maintenance, several studies have focused on server room safety using Internet of Things (IoT) technologies. Putra et al. [3] designed a security and control system for server rooms using NodeMCU ESP32 and DHT22 sensors. Their system enabled web-based monitoring and provided alert notifications via SMS and Telegram gateways in the event of thermal anomalies. Similarly, Sadewa and Beta [7] developed a rack monitoring system using ESP8266 to maintain environmental conditions in accordance with ASHRAE standards, preventing server overheating. Their system also relied on a Telegram Bot for real-time technician alerts.

However, a limitation observed in these previous works, particularly [3] and [7], is the reliance on text-based notification platforms (SMS/Telegram) or fragmented monitoring tools. These methods often require technicians to switch between the radar observation system and the infrastructure control interface. This study addresses this gap by proposing a centralized web dashboard architecture. Unlike previous fragmented approaches, our system integrates real-time radar operational data with the existing AC control infrastructure into a single cohesive interface, offering comprehensive visualization and immediate control capabilities.

3. SYSTEM ARCHITECTURE

This study employs a modern full-stack web architecture designed to ensure high responsiveness and modularity. The system follows a decoupled client-server model, where the frontend interface is separated from the backend logic, communicating via RESTful APIs [8]. The overall system architecture and data flow are illustrated in Fig. 1.

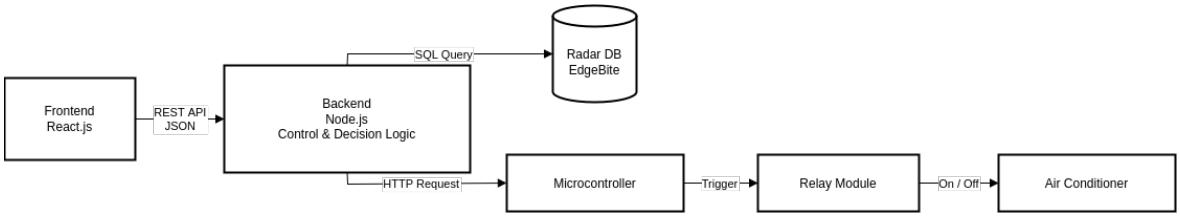

Fig. 1. System architecture diagram illustrating the data flow from the Radar Database (EdgeBite) and Existing IoT Hardware to the React.js Dashboard via Node.js Middleware

3.1. Backend Middleware (Node.js and Express)

The backend functions as an intermediary middleware layer that aggregates data from multiple sources and interfaces with the hardware control unit. Developed using Node.js with the Express framework, the server manages radar data acquisition, infrastructure monitoring, and control logic [9] [10]. Radar telemetry is retrieved from the existing EdgeBite database using the mysql2 library with connection pooling (limit: 10) by executing structured SQL queries to obtain the latest parameter values and a five-minute historical dataset for graphical visualization, without relying on direct API access to the radar system. For environmental monitoring, the backend queries a separate database (radmon), specifically the Air_Conditioner table, to collect real-time current load data (in amperes) from four AC units [11], which serves as a feedback mechanism to verify cooling system operation. Additionally, hardware control commands are implemented through a proxy-based mechanism, where AC toggle requests initiated from the frontend are processed by the Node.js server and forwarded as asynchronous HTTP requests using Axios to the microcontroller's static IP address, ensuring that the frontend does not directly expose hardware network endpoints [12].

To ensure the security of the critical AC infrastructure, the system implements Role-Based Access Control (RBAC) with a secure login mechanism. Users are categorized into general observers, who are restricted to viewing real-time radar telemetry, and administrators, who are granted exclusive privileges to download historical data and actuate the AC relay controls. This authentication layer strictly prevents unauthorized personnel from manipulating the server room's thermal environment. In the current deployment, the AC units retain their last operational state if the network connection is interrupted, ensuring that cooling continues even in the event of temporary communication failures.

3.2. Frontend Interface (React.js)

The user interface is developed using React.js and optimized with Vite to achieve rapid build performance and high responsiveness [13]. The dashboard is designed to provide a "Single Pane of Glass" experience by integrating real-time visualization, interactivity, and responsive layout design. Dynamic area charts rendered using Recharts display critical radar parameters, including HVPS voltage, current, and VSWR ratios, with automatic data refresh every five minutes to support performance trend analysis. Application state and lifecycle management are handled using React's useState and useEffect hooks, while a polling mechanism retrieves radar status at one-second intervals to ensure near real-time monitoring. Visual feedback is enhanced through Framer Motion animations [14], such as pulsing indicators, to represent data freshness and sensor states based on Active High and Active Low logic. Furthermore, the interface adopts a mobile-first responsive design implemented with Tailwind CSS, dynamically adapting from a multi-column grid layout [15] on desktop devices to a simplified vertical layout on mobile platforms, enabling technicians to monitor and control the system effectively within the local network environment.

4. RESULTS AND DISCUSSION

The proposed system was deployed and tested within the local intranet of the Tangerang Weather Radar Station to evaluate its operational feasibility. The experimental results are categorized into three key performance indicators: the usability of the interface design, the integrity of data synchronization between the dashboard and the database, and the latency of the control commands. The following sections detail how the integrated React.js and Node.js architecture successfully functions as a unified monitoring platform, replacing the previously fragmented workflow with a cohesive "Single Pane of Glass" solution.

4.1. Frontend Interface (React.js)

The proposed system has been successfully developed using the React.js framework, featuring a "Dark Mode" interface designed to minimize visual fatigue for operators monitoring the system continuously.

Fig. 2. Technician authentication login interface

Fig. 2 illustrates the authentication login interface designed to restrict system access to authorized technicians. The login mechanism serves as the first security layer of the monitoring platform, ensuring that

only authenticated users can access the operational dashboard and control functions. Through this interface, technicians must provide valid credentials before entering the system. The authentication process is integrated with the backend middleware, which verifies user roles and permissions. This access control mechanism is particularly important because the dashboard provides not only monitoring capabilities but also remote control of the server room air conditioning system, which is considered critical infrastructure for maintaining stable radar operation.

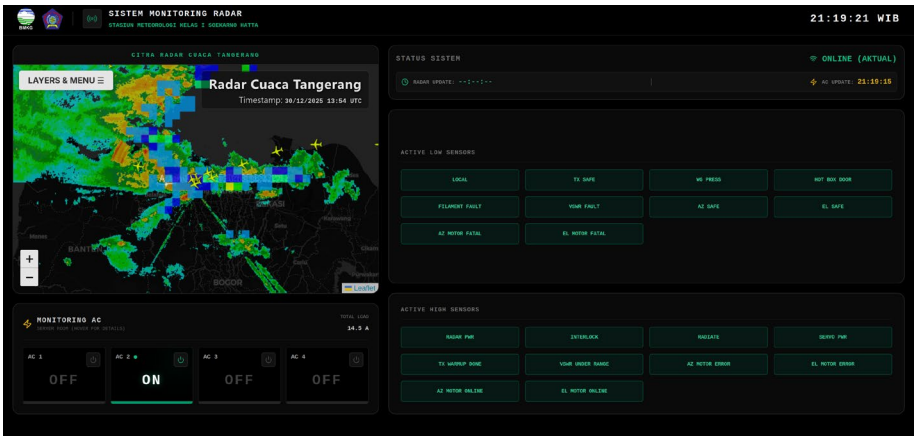

Fig. 3. Main dashboard interface integrating real-time weather radar imagery with server room air conditioning control

As illustrated in Fig. 3, the primary desktop interface integrates three core modules: real-time weather radar imagery, the Air Conditioning (AC) control panel, and BITE status monitoring. A key feature of this interface is the intelligent visualization of sensor statuses based on their specific logic characteristics. The system distinguishes between "Active High" and "Active Low" sensors to provide intuitive feedback. For Active High sensors, a logic value of '1' is rendered as a green indicator (Normal), whereas '0' triggers a red alert. Conversely, for Active Low sensors, a logic value of '0' represents a normal state displayed in green, while '1' indicates a fault condition displayed in red. This standardized color-coding mechanism allows technicians to assess the station's overall health and identify anomalies at a glance without needing to interpret raw binary data.

Fig. 4. Real-time BITE graphical monitoring showing HVPS voltage, current, transmission power, and VSWR ratio

To facilitate in-depth technical monitoring, Fig. 4 presents the graphical visualization of critical transmitter parameters. The dashboard renders real-time dynamic area charts for HVPS Voltage, HVPS Current, Forward Power (FWD), and VSWR Ratio. These graphs provide operators with historical trend visibility over a 5-minute window, enabling the early detection of signal fluctuations or power instability before they escalate into critical failures.

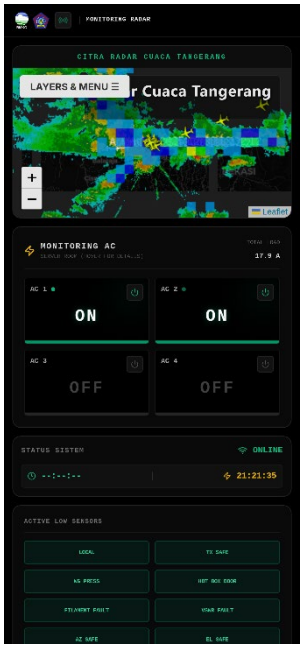

Fig. 5. Responsive mobile implementation featuring a vertical layout for radar surveillance and AC control accessibility

Furthermore, the system incorporates a responsive design approach using the Tailwind CSS utility framework. On mobile devices, as shown in Fig. 5, the application dynamically adapts the multi-column grid into a vertical stack layout. This adaptive capability ensures that essential control functions, particularly the AC power toggle, remain accessible and touch-optimized across various device viewports without compromising data readability.

4.2. Data Integration and Visualization

To validate the system's accuracy, a comparative analysis was conducted between the raw telemetry data transmitted by the backend API and the values rendered on the frontend dashboard. The backend middleware retrieves radar telemetry in JSON format, as exemplified by the raw data payload: {"hvps_v": 668, "hvps_i": 1.13, "forward_pwr": 84.5, "vswr": 1.052}.

Observation confirms that the dashboard successfully parses and renders these floating-point values with exact precision, showing no data discrepancy. The integration logic within the Node.js middleware effectively bridges the disparate database sources (EdgeBite and radmon), delivering synchronized updates to the client interface. Visual indicators for sensor status (e.g., "Active Low" logic) correctly correspond to the binary values received from the microcontroller API, validating the state management logic implemented in the React frontend.

4.3. System Responsiveness

The system's responsiveness relies on the architectural design and network deployment. Since the application, database server, and IoT control units operate within a dedicated Local Area Network (LAN) segment at the station, external network latency is effectively eliminated.

To quantitatively evaluate the responsiveness of the system, latency benchmarking was conducted using a repeated measurement method. Each API endpoint was tested 50 times within the local area network environment. The latency value was calculated as the time difference between the request initiation and the response reception at the client side. The results are summarized in Table 1.

Table 1. Latency benchmark results

System	Function	Mean Latency (ms)	Min (ms)	Max (ms)
Radar Retrieval (EdgeBite DB)	Retrieval of BITE operational parameters	43.58	22.68	208.49
Server Room Monitoring (radmon DB)	Monitoring of AC electrical current and server room temperature	81.20	59.36	365.61
AC Control Communication (ESP32 IoT)	Transmission of control commands from dashboard to AC relay controller	129.34	46.12	396.84

The radar telemetry retrieval subsystem, which accesses operational BITE parameters from the EdgeBite database, achieved the lowest average latency of 43.58 ms, indicating efficient data retrieval for radar operational monitoring. The server room monitoring subsystem, responsible for collecting electrical current data from the air conditioning units and temperature readings from the server environment, exhibited an average latency of 81.20 ms. Meanwhile, communication with the ESP32-based IoT controller used for AC relay actuation showed an average latency of 129.34 ms, reflecting the additional network and microcontroller processing overhead associated with hardware control operations.

Despite these differences, all measured subsystems operate well below the one-second threshold commonly associated with real-time monitoring systems. This demonstrates that the integrated dashboard provides sufficiently responsive performance for operational supervision of radar telemetry and environmental control within the radar station infrastructure.

To further evaluate the system's scalability and address potential high-frequency access scenarios, a stress test analysis was conducted simulating 100 concurrent user requests to the radar telemetry API endpoint. The test measured the system's data throughput and stability under simulated peak loads. The results demonstrated that the Node.js backend successfully processed all 100 concurrent requests with a 0% failure rate (no dropped connections or server crashes). The total execution time for handling this concurrent batch was 1,273.66 milliseconds, yielding an average data throughput of 78.51 requests per second. This quantitative performance analysis confirms that the REST API architecture, coupled with the database connection pooling mechanism, is highly robust. It ensures reliable data delivery and system stability even when accessed by multiple station personnel simultaneously during critical weather monitoring operations.

The use of the React.js Virtual DOM ensures efficient rendering updates, while the Node.js backend handles asynchronous HTTP requests non-blocking. Consequently, the software processing overhead is negligible. Commands initiated from the dashboard are dispatched immediately to the local network, ensuring that the limitation of the control speed is determined solely by the physical switching characteristics of the relay hardware rather than software transmission delays.

5. CONCLUSION AND FUTURE WORKS

This study has successfully achieved its primary objective of designing and implementing a centralized dashboard for the Tangerang Weather Radar Station, effectively resolving the issue of operational fragmentation between instrument surveillance and infrastructure control. By leveraging a robust modern web technology stack comprising React.js for the responsive frontend and Node.js for the backend middleware, the system provides a seamless integration of real-time BITE (Built-In Test Equipment) telemetry from the radar with the remote control mechanism of the server room's Air Conditioning (AC) units.

The empirical results demonstrate that the system maintains high data integrity, with the dashboard rendering critical telemetry parameters (HVPS, Power, and VSWR) that correspond precisely to the raw database values. The "Single Pane of Glass" architecture significantly reduces the cognitive load on technicians, eliminating the need to switch between disparate platforms. Furthermore, the deployment within a dedicated Local Area Network (LAN) ensures minimal latency, allowing for near-instantaneous actuation of the AC relay modules. The responsive design successfully adapts to mobile devices, providing technicians with flexible accessibility to maintain optimal radar operation aligned with international observation standards [16].

For future development, the system functionality can be significantly extended by implementing a customizable data export feature. This would allow operators to download historical monitoring logs based on user-defined time ranges, facilitating deeper retrospective analysis and operational reporting. Additionally,

integrating an automated notification system triggered by specific temperature or voltage anomaly thresholds would further enhance the system's capability in supporting proactive maintenance strategies.

Despite the system's operational benefits, we acknowledge a current limitation regarding automated control logic and hardware fail-safes. Presently, the NodeMCU functions as a direct remote toggle operated manually by the administrator. It does not yet incorporate automated thermal threshold logic, such as hysteresis to prevent compressor short-cycling, nor an autonomous fail-safe to lock the AC relay in a 'Normally Closed' (ON) state during a network outage. Addressing this by embedding independent temperature threshold logic directly into the microcontroller's firmware is a primary focus for our future work. By maintaining stable environmental conditions for radar server infrastructure, the system indirectly contributes to minimizing radar downtime and preventing observational data gaps in meteorological monitoring.

REFERENCE

[1] Pemerintah Republik Indonesia, Undang-Undang Republik Indonesia Nomor 31 Tahun 2009 Tentang Meteorologi, Klimatologi, Dan Geofisika. 2009, pp. 1–60. [Online]. Available: https://peraturan.bpk.go.id/Details/38769/uu-no-31-tahun-2009

[2] C. C. Wei and C. C. Hsu, 'Real-time rainfall forecasts based on radar reflectivity during typhoons: Case study in southeastern taiwan', Sensors, vol. 21, no. 4, pp. 1–20, 2021, doi: 10.3390/s21041421.

[3] I. G. Y. Putra, I. G. M. N. Desnanjaya, and I. G. Adnyana, 'Jurnal Galaksi (Global Knowledge, Artificial Inteligence and Information System) Temperature Humidity Control System and Fire Detection for Room Server Based Internet Of Things', Galaksi Glob. Knowl. Artif. Intel. Inf. Syst., vol. 1, no. 2, pp. 81–91, 2024.

[4] Ashrae, '2021 Equipment thermal guidelines for data processing environments', ASHRAE TC 99 Ref. Card, no. 404, pp. 1–8, 2021.

[5] Enterprise Electronics Corporation, 'EDGE TM (Enterprise Doppler Graphics Environment)', 2000.

[6] S. Y. Hsu, T. B. Chen, W. C. Du, J. H. Wu, and S. C. Chen, 'Integrate Weather radar and monitoring devices for urban flooding surveillance', Sens. Switz., vol. 19, no. 4, pp. 1–15, 2019, doi: 10.3390/s19040825.

[7] E. E. Sadewa and S. Beta, 'Temperature, Humidity, and Power Outage Monitoring System of Pamapersada Nusantara's Server Racks', Jaict, vol. 7, no. 2, p. 90, 2022, doi: 10.32497/jaict.v7i2.3269.

[8] M. A. Novianto and S. Munir, 'Analisis dan Implementasi Restful API guna Pengembangan Sistem Informasi Akademik pada Perguruan Tinggi', J. Inform. Terpadu, vol. 8, no. 1, pp. 47–61, Mar. 2022, doi: 10.54914/jit.v8i1.409.

[9] D. Flanagan, JavaScript: The Definitive Guide, 7th edn. O'Reilly Media, 2020.

[10] E. Nurhayati and A. Agussalim, 'Rancang Bangun Back-end API pada Aplikasi Mobile AyamHub Menggunakan Framework Node JS Express', J. Sist. Dan Teknol. Inf. JustIN, vol. 11, no. 3, p. 524, July 2023, doi: 10.26418/justin.v11i3.66823.

[11] W. A. Suteja and A. Surya Antara, 'Analisis Sensor Arus Invasive ACS712 dan Sensor Arus Non Invasive SCT013 Berbasis Arduino', PROtek J. Ilm. Tek. Elektro, vol. 8, no. 1, pp. 13–21, May 2021, doi: 10.33387/protk.v8i1.2116.

[12] Espressif Systems, 'ESP32 Dev Kits Documentation', 2023.

[13] T. A. Rizzi, R. Aziz Abdillah, Y. Efani Yancandra, M. Wijaya, and A. Voutama, 'Implementasi React Vite Dalam Pengembangan Antarmuka Sistem Pemesanan Tiket Pesawat Dengan Metode Scrum', J. Process., vol. 20, no. 1, May 2025, doi: 10.33998/processor.2025.20.1.2231.

[14] J. A. Ramadhanti, L. Isyriyah, and R. Maulidi, 'Penerapan Motion UI untuk Meningkatkan Pengalaman Pengguna dan Penyampaian Informasi pada Website Kampus', SMATIKA J., vol. 15, no. 01, pp. 226–239, July 2025, doi: 10.32664/smatika.v15i01.2039.

[15] S. Azhariyah and Muhammad Mukhlis, 'Framework CSS: Tailwind CSS Untuk Front-End Website Store PT. XYZ', J. Inform., vol. 3, no. 1, pp. 30–36, Apr. 2024, doi: 10.57094/ji.v3i1.1601.

[16] World Meteorological Organization, Guide to Instruments and Methods of Observation (WMO-No. 8). Measurement of Precipitation, vol. III. 2021.