Airman

Jurnal Teknik dan Keselamatan Transportasi

ISSN: 2622-0105 (print) | 2716-1196 (online)
http://jurnal.poltekbangmakassar.ac.id/

Development of a Visual IoT Air Quality Monitoring System with Web Visualization and Telegram Notification

Abdurrahman Alfaruqi[1*], Magnify Abidullah Mafaza[2], Muhamad Fikri [3], Ikhwanul Kurnia Rahman[4]
[1,2,3,4]Politeknik IDN Bogor

***Correspondence:**
alfaruqiabdurrahman4@gmail.com,

Article info

Received: 08-01-2026
Final Revision: 06-02-2026
Accepted: 30-06-2026
Available online: 30-06-2026

ABSTRACT

Air quality plays a crucial role in maintaining a healthy and comfortable indoor environment; therefore, continuous monitoring is required to reduce potential health risks caused by poor air conditions. This study developed an Internet of Things (IoT)-based air quality monitoring system using a Research and Development (R&D) method with a Waterfall development model consisting of requirement analysis, system design, prototype development, testing, and deployment. The system was built using an ESP32 microcontroller integrated with DHT11 and MQ-135 sensors to measure temperature, humidity, and air quality levels in real time. Sensor data were collected and transmitted every 10 seconds in JSON format through HTTP POST requests to a Node.js backend and stored in a MySQL database. The processed data were displayed on a web-based visualization interface designed in a weather-style format with color-coded air quality indicators, making the information easier to understand for general users. In addition, a Telegram bot was integrated to send automatic warning notifications when air quality values exceeded predefined safety thresholds. Black-box testing was conducted to evaluate sensor readings, data transmission, database storage, web visualization, and Telegram notification functionality. The experimental results showed that the system recorded temperatures ranging from 26.6°C to 28.6°C, humidity levels between 58% and 60%, and MQ-135 analog values between 432 and 519 under normal indoor conditions. The system successfully transmitted sensor data to the backend every 10 seconds without observed data loss or transmission errors. When the MQ-135 value reached 1392, exceeding the predefined threshold, the Telegram bot successfully sent an instant warning notification containing the gas level, temperature, and humidity values. These results indicate that the proposed system can provide real-time air quality monitoring, measurable environmental data, intuitive web visualization, and automated early warning notifications. Therefore, the system offers a low-cost, user-friendly, and scalable solution for indoor air quality monitoring using IoT technology.

Reccomended Citation:

APA Style

INTRODUCTION

Air quality is an important environmental factor that directly affects human health, comfort, and productivity, especially in indoor environments where people spend most of their daily activities. Poor indoor air quality may increase the risk of respiratory problems, discomfort, and reduced concentration if it is not detected and managed properly. Therefore, continuous environmental monitoring is needed to provide early information about air quality conditions and support timely preventive actions.

The Internet of Things (IoT) has become an effective approach for developing real-time environmental monitoring systems. IoT technology enables sensors, microcontrollers, backend services, databases, web applications, and notification platforms to work together in collecting, transmitting, processing, and presenting environmental data. In air quality monitoring, IoT allows temperature, humidity, and gas concentration data to be measured continuously and accessed remotely through internet-based platforms. Previous studies have shown that IoT-based systems are effective for monitoring environmental conditions and providing early warnings (Al-Faris & Elsi, 2024). Other studies have also implemented IoT-based air quality monitoring using rule-based methods and Telegram notifications to classify air quality levels and improve environmental awareness in real-world communities (Haryanto et al., 2024).

IoT-based monitoring has also been applied in other domains, such as security, electrical systems, and household energy monitoring. For example, RFID-based access control systems integrated with Telegram notifications can provide instant alerts for unauthorized access attempts (Ma'muriyah et al., 2023). Web-based IoT applications for electrical system monitoring have been used to visualize real-time data and detect abnormal conditions at an early stage (Alfian et al., 2023). In household energy monitoring, real-time IoT data access helps users understand consumption patterns and make better decisions regarding resource efficiency (Munir & Setiawan Wibisono, 2025). These studies show that IoT is a flexible technology for environmental and operational monitoring because it supports real-time data acquisition, remote access, visualization, and notification features.

However, many existing IoT monitoring systems still focus mainly on technical functionality, such as sensor reading, tabular dashboards, and basic notification delivery. Although these systems are useful, they often present information in a technical format that may be less intuitive for general users. In addition, several previous studies have not clearly emphasized how environmental data can be translated into a user-friendly monitoring model that combines real-time sensing, visual interpretation, historical data access, and instant warning notification in one integrated system. This creates a research gap in the development of an air quality monitoring model that is not only functional, but also easy to understand, visually informative, and practical for non-technical users.

To address this gap, this study proposes an IoT-based air quality monitoring system that integrates environmental sensors, an ESP32 microcontroller, a Node.js backend, a database, a web-based visualization interface, and Telegram notification. The novelty of this study lies in the development of a weather-style web visualization model for indoor air quality monitoring. Instead of presenting data only in conventional tables or technical dashboards, the proposed system displays environmental conditions using visual indicators, dynamic status information, and color-coded air quality levels. This approach is designed to help users quickly understand whether the air condition is safe or dangerous without requiring technical interpretation.

The main contribution of this research is the design and implementation of a low-cost and user-friendly environmental monitoring model that combines real-time sensor data acquisition, backend data processing, web visualization, and automatic Telegram alerts. The system measures temperature, humidity, and air quality levels, sends the data periodically to the backend, stores the data in a database, and triggers warning notifications when predefined thresholds are exceeded. By combining visual web presentation and instant notification, the system is expected to support early detection of poor air quality and improve user awareness of indoor environmental conditions.

This research also evaluates the system through sensor reading observation, data transmission testing, database storage verification, web visualization testing, and Telegram notification testing. The evaluation is intended to provide measurable evidence of system performance, including sensor reading ranges, data transmission intervals, successful data storage, and alert delivery when air quality exceeds the threshold. Therefore, this study does not only develop an IoT prototype, but also demonstrates its practical performance as an integrated environmental monitoring and early warning system.

The scope of this study is limited to prototype implementation and testing in an indoor or semi-controlled environment using selected air quality and environmental sensors. Future research can improve the system by

adding sensor calibration with standard reference instruments, expanding the number of monitored pollutants, comparing the system performance with commercial air quality monitoring devices, and applying predictive analytics or machine learning to estimate future air quality conditions.

METHOD

This study applied a Research and Development (R&D) methodology using the Waterfall development model to design, implement, and evaluate an Internet of Things (IoT)-based air quality monitoring system. The Waterfall model was selected because the system development process required a clear and sequential workflow, starting from requirement analysis, system design, prototype construction, testing, and deployment. This approach allowed each stage of development to be completed systematically before proceeding to the next stage, ensuring that both hardware and software components were properly integrated and evaluated.

The research focused on the development of a real-time environmental monitoring model that could measure indoor air quality conditions, transmit sensor data to a backend server, visualize the data through a web interface, and send automatic Telegram notifications when unsafe air conditions were detected. The main parameters monitored in this study were temperature, humidity, and gas concentration values detected by the MQ-135 sensor. The system was designed not only to collect environmental data, but also to present the information in a visually intuitive format so that general users could easily understand the air quality status without requiring technical knowledge.

The development process consisted of five main stages: requirement analysis, system design, prototype development, system testing, and deployment evaluation. In the requirement analysis stage, the functional and non-functional requirements of the system were identified. The functional requirements included sensor data acquisition, real-time data transmission, backend data validation, database storage, web-based visualization, air quality status classification, and Telegram-based alert notification. Meanwhile, the non-functional requirements included system responsiveness, data transmission reliability, database storage accuracy, user-friendly visualization, and notification delivery speed.

In the system design stage, the architecture of the IoT monitoring system was developed by integrating hardware and software components. The hardware components consisted of an ESP32 microcontroller, a DHT11 sensor for measuring temperature and humidity, and an MQ-135 sensor for detecting air quality conditions related to gas concentration. The ESP32 was selected because it supports Wi-Fi connectivity, has sufficient processing capability for sensor-based IoT applications, and can transmit data directly to an internet-based backend server. The DHT11 sensor was used to obtain temperature and humidity values, while the MQ-135 sensor was used to detect changes in air quality based on analog gas concentration readings.

The software components consisted of a Node.js backend using the Express.js framework, a MySQL database, a web-based visualization interface, and a Telegram bot notification system. The backend was responsible for receiving sensor data through RESTful API endpoints, validating incoming data, storing valid records in the database, classifying air quality status based on predefined thresholds, and triggering Telegram alerts when sensor readings exceeded the safe limits. The web interface retrieved processed sensor data from the backend and displayed it in a weather-style visualization using dynamic indicators, color-coded status information, and real-time environmental values. The Telegram bot was integrated to provide automatic warning messages without requiring users to continuously monitor the web interface.

During the prototype development stage, the ESP32 was programmed to read data from the DHT11 and MQ-135 sensors periodically. The sensor readings were converted into JSON format and transmitted to the backend server through HTTP POST requests using a Wi-Fi connection. Each data packet contained temperature, humidity, MQ-135 analog value, voltage value, digital status, timestamp, and device identity. The backend then checked whether the received data were complete, valid, and within a reasonable measurement range before storing them in the MySQL database. If the MQ-135 reading exceeded the predefined gas threshold or other environmental parameters exceeded the safety limits, the backend automatically sent a warning notification through the Telegram bot.

The testing stage was conducted using a black-box testing approach. This method was used to evaluate the functionality and performance of the system from the user and system-output perspectives without examining the internal program code. The testing covered five main aspects: sensor validation, data transmission reliability, backend and database performance, web visualization accuracy, and Telegram notification responsiveness. Each test was designed with specific scenarios and quantitative evaluation parameters to provide measurable evidence of system performance.

Sensor validation was conducted to evaluate the consistency and accuracy of the DHT11 and MQ-135 sensor readings. The DHT11 sensor readings were compared with reference temperature and humidity measurements, while the MQ-135 sensor readings were observed under normal air conditions and gas-triggered conditions. The evaluation parameters included minimum value, maximum value, average value, measurement deviation, and percentage error. The percentage error was calculated using the following formula:

Percentage Error = |Reference Value − Sensor Value| / Reference Value × 100%

This validation was intended to determine how closely the sensor readings matched the reference measurements and whether the sensor output remained stable during repeated observations. The DHT11 sensor was evaluated based on temperature and humidity readings, while the MQ-135 sensor was evaluated based on changes in analog values when air quality conditions changed.

Data transmission testing was conducted to evaluate the reliability of communication between the ESP32 and the Node.js backend. The ESP32 transmitted sensor data to the backend at a fixed interval of 10 seconds. The testing measured the number of data packets sent, number of data packets successfully received, number of failed transmissions, transmission success rate, and average transmission delay. The transmission success rate was calculated using the following formula:

Transmission Success Rate = Successfully Received Data / Total Sent Data × 100%

This test was important to ensure that the system could support continuous real-time monitoring without significant data loss. A successful transmission was defined as a condition where the backend received a complete JSON data packet, validated the data, and returned a successful response to the ESP32.

Backend and database testing was conducted to evaluate whether the server could properly receive, validate, process, and store sensor data. The backend was tested by sending valid data, incomplete data, invalid data types, and threshold-exceeding data. The evaluation parameters included API response status, backend response time, database insertion success rate, validation accuracy, and error handling capability. The database insertion success rate was calculated using the following formula:

Database Success Rate = Successfully Stored Records / Successfully Received Records × 100%

This test ensured that only valid sensor data were stored in the database and that the backend could reject incomplete or invalid data. The backend response time was measured to determine whether the system could process incoming sensor data fast enough to support real-time monitoring.

Web visualization testing was conducted to evaluate whether the web interface accurately displayed the latest sensor data stored and processed by the backend. The test scenarios included normal air condition display, warning condition display, real-time data update, color-coded air quality status display, and historical data visualization. The evaluation parameters included data display accuracy, interface update delay, status classification accuracy, and visual consistency. Data display accuracy was evaluated by comparing the values shown on the web interface with the values stored in the database. Interface update delay was measured as the time difference between data reception by the backend and data appearance on the web interface.

Telegram notification testing was conducted to evaluate the responsiveness and reliability of the warning system. The test scenarios included normal condition without notification, gas threshold exceeded, repeated high gas readings, and recovery from dangerous to safe conditions. The evaluation parameters included notification delivery success rate, notification delay, message accuracy, and threshold detection accuracy. Notification delay was measured as the time difference between the backend detecting a threshold violation and the Telegram message being received by the user. Notification delivery success rate was calculated using the following formula:

Notification Success Rate = Successfully Delivered Notifications / Total Triggered Alerts × 100%

A notification was considered successful if the Telegram bot sent a warning message containing the MQ-135 value, temperature, humidity, and warning status after the measured value exceeded the predefined threshold. This test was important to verify that the system could provide early warnings without requiring users to constantly access the web interface.

The deployment evaluation was conducted in an indoor or semi-controlled environment to observe the system's practical performance. During this phase, the system was operated continuously to monitor temperature, humidity, and air quality conditions. The ESP32 collected sensor readings, transmitted the data to the backend, and allowed the web interface and Telegram bot to respond based on the processed values. The

evaluation focused on whether the system could maintain stable data acquisition, reliable data transmission, accurate visualization, and timely notification delivery during continuous operation.

The quantitative evaluation in this study used several measurable indicators, including sensor reading range, percentage error, data transmission success rate, database insertion success rate, backend response time, web interface update delay, Telegram notification delay, and notification success rate. These parameters were used to provide objective evidence of system performance rather than relying only on descriptive observations. The collected data were analyzed by calculating minimum, maximum, average, and percentage-based performance values for each testing component.

This methodology provides a clearer evaluation framework for validating the proposed IoT-based air quality monitoring system. By combining sensor validation, communication reliability testing, backend and database evaluation, web visualization testing, and Telegram notification testing, the study can demonstrate not only that the system works functionally, but also that its performance can be measured quantitatively. This strengthens the reliability of the research findings and supports the contribution of the proposed system as a real-time, user-friendly, and low-cost indoor air quality monitoring model.

System Design

The air quality monitoring system is designed to continuously collect environmental data, process it in real time, and provide intuitive visual feedback to users, along with automated notifications through Telegram. The architecture integrates hardware and software components in a coordinated workflow to ensure accurate measurements, timely alerts, and a visually engaging web interface. The hardware layer consists of ESP32 microcontrollers connected to DHT22 sensors for temperature and humidity and MQ-135 sensors for air quality measurements, including CO_2, NH_3, NOx, and smoke. The ESP32 units are programmed to acquire data every five seconds, format readings into JSON objects, and transmit them via Wi-Fi to the centralized backend server. The microcontrollers handle preliminary data processing to ensure that only correctly formatted data are sent to the backend.

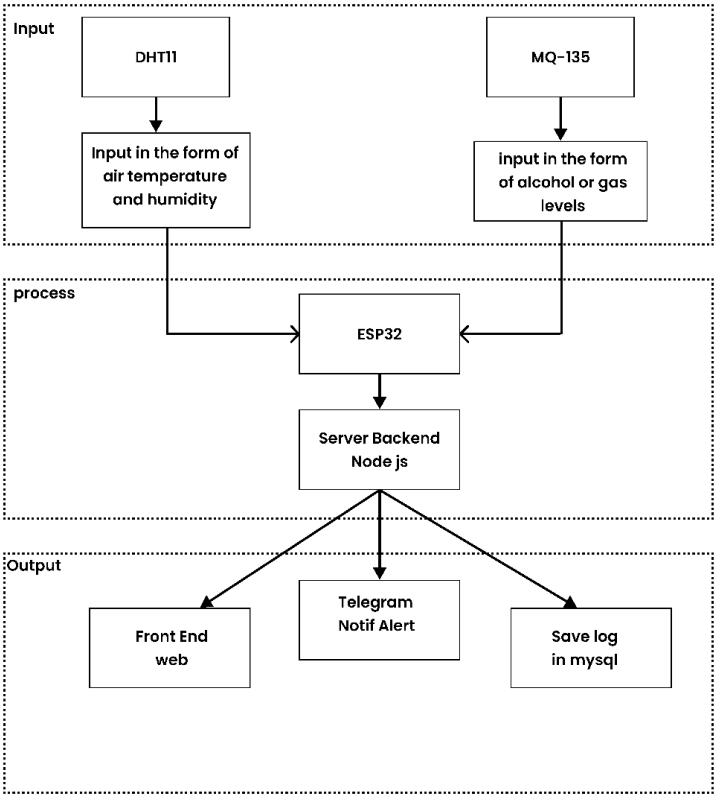

Figure 1 System Design

The backend system is built on Node.js with the Express.js framework. The backend receives sensor data through RESTful API endpoints and performs several critical functions. Incoming data are validated to detect missing values or out-of-range measurements. Validated records are stored in a MySQL database, structured to efficiently handle high-frequency IoT data, supporting both real-time processing and historical queries. The backend continuously monitors sensor readings against predefined thresholds, such as an Air Quality Index (AQI) above 150 or temperature above 35°C. When thresholds are exceeded, the backend triggers the Telegram bot, which sends instant notifications to registered users. This ensures timely alerts and enhances the system's effectiveness for indoor and semi-outdoor monitoring.

The web interface provides a visually rich representation of real-time environmental data. Unlike traditional table-based dashboards, the interface is designed to resemble a weather monitoring website. Temperature and humidity are displayed with animated icons, air quality levels are shown using color-coded gauges and graphs, and warning conditions are highlighted through prominent visual cues. The interface queries the backend periodically to update visualizations dynamically, ensuring that users can instantly see changes in environmental conditions. Historical data and trends are displayed using interactive charts, allowing users to track air quality variations over time. Integration between hardware and software is optimized for reliability and scalability. Multiple ESP32 devices can submit data simultaneously without overloading the Node.js backend, and the MySQL database efficiently manages storage and retrieval of high-frequency measurements. The Telegram bot operates asynchronously, ensuring that alert notifications do not interrupt data collection or backend processing.

In summary, the system design establishes a seamless operational loop: sensors collect environmental data, ESP32 microcontrollers preprocess and send data to the Node.js backend, the backend validates and stores the data in MySQL while evaluating thresholds, the web interface provides real-time visual feedback in a weather-style format, and the Telegram bot delivers immediate alerts when necessary. This design guarantees accurate, real-time monitoring, intuitive visualization, and reliable alerting, making it suitable for practical air quality monitoring applications in indoor and semi-outdoor environments.

Device Components

The IoT air quality monitoring system consists of several interdependent components that work together to ensure continuous environmental measurement and real-time alerting. At the heart of the system are ESP32 microcontrollers, which act as the main processing units, reading data from connected sensors, formatting it into JSON, and transmitting it to the backend server via Wi-Fi. The sensors include DHT22 units, which measure temperature and humidity with digital precision, and MQ-135 gas sensors, which detect key air pollutants such as CO_2, NH_3, NOx, and smoke. These readings are crucial for assessing indoor or semi-outdoor air quality conditions. Once transmitted, the data is received by a Node.js backend built with Express.js, where it is validated for integrity, stored in a MySQL database, and analyzed against predefined thresholds. When any parameter exceeds safe limits, the backend automatically triggers a Telegram bot, sending instant notifications to users to ensure timely responses. Simultaneously, a web-based visual interface presents the environmental data in an intuitive, weather-like format, using animated icons, color-coded air quality indicators, and dynamic status displays to provide a clear and immediate understanding of current conditions. Power supply components provide reliable electricity to the ESP32 and sensors, ensuring uninterrupted monitoring. Together, these components form a cohesive and fully integrated system, enabling accurate data acquisition, real-time processing, engaging visual feedback, and reliable alert delivery for practical air quality monitoring applications.

Testing Techniques

The IoT air quality monitoring system was evaluated using a black-box testing approach, which focused on assessing the system's overall functionality, performance, and reliability without examining the internal code of individual components. Testing began with the sensors, ensuring that the DHT22 accurately measured temperature and humidity, and the MQ-135 reliably detected air pollutants such as CO_2, NH_3, NOx, and smoke. Sensor readings were cross-checked against reference instruments to confirm accuracy and consistency. The communication between ESP32 microcontrollers and the Node.js backend was also tested to ensure that JSON-formatted data packets were transmitted correctly and received without loss, while the

backend's ability to validate, store, and process data in the MySQL database was verified, including the detection of readings exceeding predefined thresholds.

The system's alerting and visualization components were further tested to confirm responsiveness and reliability. The Telegram bot was evaluated under multiple scenarios, including simultaneous alerts from different sensors, to ensure timely notifications. The web visual interface was examined to guarantee that real-time data were accurately reflected using dynamic, weather-style visualizations with color-coded air quality indicators. Finally, the system's overall stability was assessed by continuous operation in controlled environments, monitoring data transmission rates, backend processing times, and interface responsiveness. These tests demonstrated that the system could provide accurate real-time monitoring, intuitive visual feedback, and prompt alerts, fulfilling its objectives for indoor and semi-outdoor air quality applications.

RESULTS AND DISCUSSION

The ESP32-based IoT air quality monitoring system successfully captured real-time environmental data, including temperature, humidity, and air pollutant concentrations measured by the MQ-135 sensor. Over multiple measurement intervals, the DHT11 sensor recorded temperatures ranging from 26.6°C to 28.6°C and relative humidity between 58% and 60%, demonstrating stable performance under controlled indoor conditions. Concurrently, the MQ-135 analog sensor readings varied from 432 to 519, corresponding to voltage levels of approximately 0.34 to 0.35 V, while the digital output consistently indicated a logic HIGH (1) state, reflecting the sensor's detection threshold for air pollutants. These values were transmitted to the Node.js backend every 10 seconds via HTTP POST requests, formatted in JSON, and stored in the MySQL database without any observed data loss or transmission errors.

The summarized sensor readings are presented in Table 1, providing a clear overview of environmental conditions captured by the system. The data indicate that the system can reliably measure temperature, humidity, and air quality indicators in near-real-time. Minor fluctuations in the MQ-135 analog values correspond to expected changes in ambient air composition, while the DHT11 readings remained consistent, confirming sensor accuracy. The continuous data transmission to the backend ensured that all measurements were available for real-time visualization on the web interface and for triggering Telegram notifications when thresholds were exceeded. Overall, these results demonstrate the system's ability to monitor indoor air quality effectively, providing accurate environmental readings, reliable data storage, and timely alerting functionality.

Table 1. Testing of DHT11 and MQ-135 output

Temp	Humidity	MQ-135 Analog	Voltage	Digital
26.60 °C	58.00 %	432	0.35 V	1
27.60 °C	60.00 %	500	0.35 V	1
28.60 °C	59.00 %	519	0.34 V	1
27.60 °C	58.00 %	432	0.35 V	1

The real-time IoT air quality monitoring system was implemented using ESP32 microcontrollers with DHT11 and MQ-135 sensors to measure temperature, humidity, and gas concentration. Data were transmitted every 10 seconds via HTTP POST requests to a Node.js backend, which validated and stored the measurements in a MySQL database. This ensured all sensor data could be retained for future retrieval, even if the ESP32 devices temporarily went offline. The system continuously monitored the sensor readings and evaluated them against predefined thresholds for air quality, temperature, and humidity.

When the MQ-135 sensor recorded a value of 1392 at 12:00 PM on January 8th, exceeding the set gas concentration threshold, while the temperature and humidity were 27°C and 71%, respectively, the backend immediately triggered the Telegram bot to send a real-time alert: "⚠ Peringatan Gas! MQ-135: 1392 Temperature: 27°C Humidity: 71%." Simultaneously, the web interface displayed live sensor data in a weather-style visualization, including color-coded indicators for air quality status. This integration of accurate sensing, MySQL data logging, immediate Telegram notifications, and an intuitive web visualization demonstrates the system's effectiveness in providing reliable real-time monitoring and actionable feedback for indoor and semi-outdoor environments.

Table 1. Testing of DHT11 and MQ-135 output

Parameter	Minimum	Maximum	Average	Range
Temperature	26.60°C	28.60°C	27.60°C	2.00°C
Humidity	58.00%	60.00%	58.75%	2.00%
MQ-135 Analog	432	519	470.75	87
Voltage	0.34 V	0.35 V	0.35 V	0.01 V

Table 2 presents the quantitative summary of the sensor readings. The temperature data showed a range of 2.00°C, indicating that the indoor test environment was relatively stable during observation. The humidity range was also small, with a difference of 2.00% between the lowest and highest values. The MQ-135 readings showed a wider range of 87 points, which indicates that the gas sensor was more sensitive to changes in ambient air conditions compared with the temperature and humidity sensor. This result is reasonable because gas concentration may fluctuate depending on air circulation, sensor heating condition, and the presence of surrounding pollutants.

The results indicate that the system was able to capture environmental data consistently under normal indoor conditions. The relatively small variation in temperature and humidity shows that the DHT11 sensor produced stable readings during the test period. The MQ-135 analog readings varied between 432 and 519 under normal conditions, which can be used as the baseline range for the tested environment. However, because the test was conducted using a limited number of samples, the result should be interpreted as an initial functional evaluation rather than a complete sensor accuracy validation. To evaluate sensor accuracy more rigorously, future testing should compare the DHT11 and MQ-135 readings with calibrated reference instruments and calculate error percentage, mean absolute error, and standard deviation.

The data transmission test showed that the ESP32 successfully sent sensor readings to the backend every 10 seconds using JSON-formatted HTTP POST requests. All recorded data were received by the Node.js backend and stored in the MySQL database without observed transmission errors during the test. This indicates that the communication between the ESP32, backend server, and database functioned properly in the tested environment. The use of a backend server and database also allowed the system to store historical sensor data, which is important for monitoring changes in air quality over time.

The Telegram notification feature was tested by triggering an unsafe air quality condition. When the MQ-135 sensor recorded a value of 1392 at 12:00 PM on January 8, with a temperature of 27°C and humidity of 71%, the backend detected that the gas concentration exceeded the predefined threshold. The system then automatically sent a warning message through the Telegram bot containing the MQ-135 value, temperature, and humidity information. This result confirms that the alert mechanism was able to detect abnormal air quality conditions and provide instant notification to users. The Telegram notification is useful because users do not need to continuously monitor the web interface to know when the air quality becomes unsafe.

The web visualization interface also successfully displayed real-time sensor data in a weather-style format. Instead of presenting the data only in technical tables, the system used visual elements such as temperature information, humidity values, air quality status, color indicators, and simple descriptive text. This approach helps users understand environmental conditions more easily, especially non-technical users who may not be familiar with raw sensor values. The web display complements the Telegram alert system because the website provides contextual information, while Telegram provides immediate warning when the threshold is exceeded.

Compared with previous IoT-based air quality monitoring studies, the proposed system has similar core functionality in terms of real-time data acquisition, air quality classification, and early warning notification. Al-Faris and Elsi (2024) showed that IoT-based systems can support environmental monitoring and early warning functions. Haryanto et al. (2024) also implemented an IoT-based air quality monitoring system using a rule-based method and Telegram notification to classify air quality levels and improve public awareness. The present study supports these findings by showing that ESP32, DHT11, MQ-135, web visualization, backend processing, database storage, and Telegram notification can be integrated into one functional monitoring system.

However, the contribution of this study is different from previous monitoring systems because it emphasizes a visual weather-style web interface rather than a conventional technical dashboard. Several previous systems mainly focused on sensor readings, classification rules, and notification delivery. In this study, the monitoring result is presented using a more intuitive visual format so that users can understand air

quality conditions more quickly. This design approach makes the system more user-friendly and practical for indoor environments, educational settings, residential spaces, or small workplaces.

The use of Node.js and RESTful API in this system is also consistent with previous backend implementation studies. Previous research on backend API development has shown that Node.js-based REST APIs can support data processing, database management, and service integration in web applications. In this study, the Node.js backend was used to receive sensor data, validate incoming values, store records in MySQL, process threshold conditions, and trigger Telegram alerts. This confirms that a lightweight backend architecture can support real-time IoT monitoring and notification services.

Although the system functioned successfully during testing, several limitations should be acknowledged. First, the number of sensor data samples was still limited, so the results are not sufficient to fully represent long-term system accuracy and reliability. Second, the sensor readings were not yet compared with calibrated reference instruments, so the measurement error and accuracy level could not be calculated precisely. Third, the MQ-135 sensor provides general gas detection values and requires calibration to estimate specific pollutant concentrations more accurately. Fourth, the system was tested in a controlled indoor environment, so its performance in different environmental conditions, such as high humidity, outdoor air circulation, or long-term continuous operation, still needs further evaluation.

Therefore, future research should include longer testing periods, larger datasets, sensor calibration using reference instruments, and comparison with commercial air quality monitoring devices. Future studies should also measure additional performance indicators such as data transmission success rate, backend response time, web update delay, Telegram notification delay, system uptime, and data loss percentage. In addition, the system can be improved by adding more pollutant sensors, applying machine learning for predictive air quality alerts, and enhancing the web interface with historical trend analysis and user interaction features.

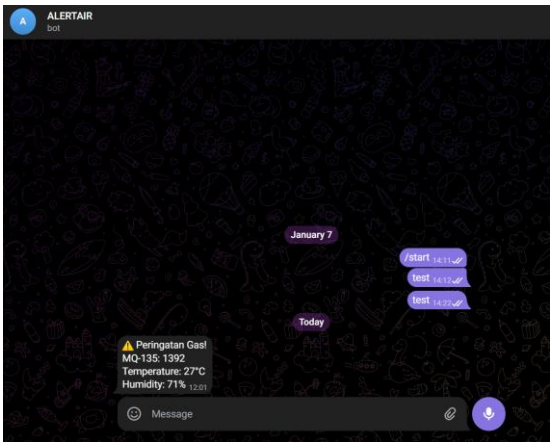

Figure 1 Telegram Alert

The web application serves as a real-time visualization interface for the IoT-based air quality monitoring system, displaying data transmitted from ESP32 microcontrollers connected to DHT22 and MQ-135 sensors. Sensor readings, received via a Node.js backend and stored in a MySQL database, are processed into air quality status, temperature, and humidity, and then presented using dynamic icons, colors, and text in a weather-like display, eliminating the need for conventional tables. This visual interface enables users to quickly interpret environmental changes, facilitates rapid understanding of air quality conditions, and complements the Telegram alert system by providing contextual visual feedback that corresponds with the notifications received.

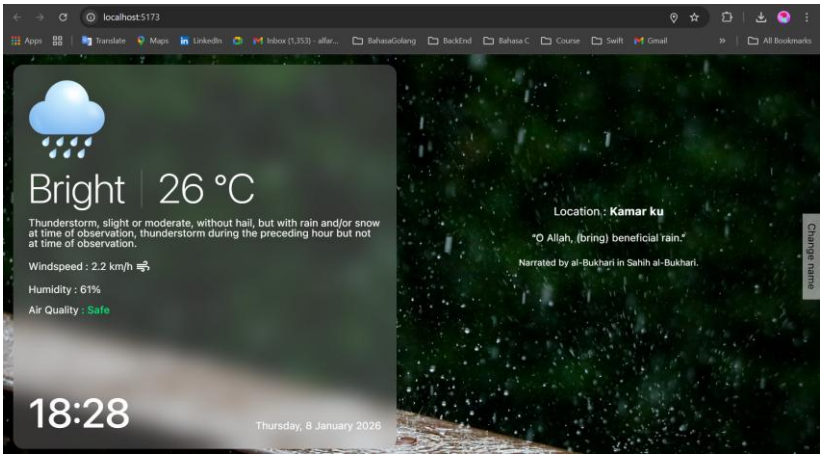

Figure 1 Website

CONCLUSION

This study designed and implemented an IoT-based air quality monitoring system that integrates an ESP32 microcontroller, DHT11 and MQ-135 sensors, a Node.js backend, a MySQL database, a web-based visualization interface, and Telegram notification. The system was developed to support real-time environmental monitoring by collecting temperature, humidity, and air quality sensor readings, transmitting the data to a backend server, storing the data in a database, and presenting the information through a weather-style web interface. The Telegram notification feature was also implemented to provide automatic alerts when the air quality value exceeded the predefined threshold.

Based on the initial testing, the system was able to record temperature values ranging from 26.60°C to 28.60°C, humidity values between 58.00% and 60.00%, and MQ-135 analog readings between 432 and 519 under normal indoor conditions. The system also successfully transmitted sensor data to the backend every 10 seconds using HTTP POST requests and stored the data in the MySQL database. In the alert testing scenario, when the MQ-135 sensor value reached 1392, the system successfully triggered a Telegram warning notification containing the gas level, temperature, and humidity information. These results indicate that the proposed system can perform basic real-time monitoring, data logging, web visualization, and automatic warning notification functions.

The main contribution of this study is the integration of IoT-based air quality monitoring with a visually intuitive weather-style web interface and Telegram-based early warning notification. Compared with conventional monitoring displays that commonly present data in technical tables or simple dashboards, the proposed interface presents environmental information using visual indicators and status descriptions that are easier to understand for general users. Therefore, the system has potential to be applied as a low-cost and user-friendly prototype for indoor air quality monitoring in residential, educational, or small workplace environments.

However, this study has several limitations. First, the number of testing samples was still limited, so the results cannot yet fully represent long-term system accuracy and reliability. Second, the sensor readings were not compared with calibrated reference instruments, so the measurement accuracy and error percentage could not be determined quantitatively. Third, the MQ-135 sensor provides general gas concentration indicators and requires further calibration to identify specific pollutant concentrations more precisely. Fourth, the system was tested in a limited indoor environment, so its performance under different environmental conditions, longer operating periods, and multiple sensor nodes still needs further evaluation.

Future research should conduct longer-term testing with larger datasets, validate sensor readings using calibrated reference instruments, calculate quantitative performance metrics such as sensor error percentage, data transmission success rate, backend response time, web update delay, Telegram notification delay, and system uptime, and compare the results with previous IoT-based air quality monitoring studies or commercial air quality monitoring devices. Further development may also include additional pollutant sensors, improved calibration methods, historical trend analysis, and machine learning-based prediction to enhance the accuracy and usefulness of the monitoring system

REFERENCES

M. G. Ioannides, A. Stamelos, S. A. Papazis, A. Papoutsidakis, and V. Vikentios, "IoT monitoring system for applications with renewable energy generation and electric drives," Renewable Energy and Power Quality Journal, vol. 19, no. 5, Jan. 2024, doi: 10.24084/repqj19.347..

H. Nguyen, D. Nawara, and R. Kashef, "Connecting the indispensable roles of IoT and artificial intelligence in smart cities: A survey," Journal of Information and Intelligence, vol. 2024, p. 100014, Jan. 2024, doi: 10.1016/j.jiixd.2024.01.003.

Ma'muriyah, N., Yulianto, A., & Tymoden. (2023). Perancangan sistem smart door dengan notifikasi Telegram, 8(2), 64–71. http://e-journal.sari-mutiara.ac.id/index.php/7/article/view/4642.

S. A. Almohaimeed, "Techno-environmental analysis of a microgrid energy system in a university office complex," Sustainability, vol. 15, no. 16, p. 12506, Aug. 2023, doi: 10.3390/su151612506.

Munir, M., & Setiawan Wibisono, I. (2025). Pemantauan daya listrik real-time menggunakan IoT untuk efisiensi energi rumah tangga. Jurnal Algoritma**,** 22(2), 1853–1862. https://doi.org/10.33364/algoritma/v.22-2.2391.

R. Bhavya, P. Manjuladevi, E. S. Santhoshini, M. Ramya, and R. Dhanapal, "Development and improving energy management system in smart grid through integration with renewable energy system with AI and internet of things," in Proc. Int. Conf. Inventive Comput. Technol. (ICICT), 2023, pp. 398–403, doi:10.1109/icict57646.2023.10134409.

M. Rajca, "Privacy risks and regulatory challenges in smart grids and renewable energy systems," Internet Quarterly of Antimonopoly and Regulatory Studies, vol. 13, no. 2, pp. 9–25, 2024, doi: 10.7172/2299-5749.ikar.2.13.1.

Riady, A. M. N., Paniran, P., & Suksmadana, I. B. (2024). Perancangan backend API berbasis REST-API pada aplikasi rekomendasi resep makanan. Mars: Jurnal Teknik Mesin, Industri, Elektro dan Ilmu Komputer, *2*(3), 94–106. https://doi.org/10.61132/mars.v2i3.137

E. Cano-Suñén et al., "Internet of things (IoT) in buildings: A learning factory," Sustainability, vol. 15, no. 16, p. 12219, Aug. 2023, doi:10.3390/su151612219.

Nugroho, M. F., Primajaya, A., & Jajuli, M. (2023). Rancang bangun REST API aplikasi manajemen toko menggunakan NodeJS pada Cantika Paint. *Jurnal Mahasiswa Teknik Informatika (JATI), 7*(6), 3904–3912. https://ejournal.itn.ac.id/jati/article/view/7882