Smart Village Air Quality Monitoring and Industrial Pollution Tracking System Using IoT, MERN, and ESP32

Masrath Begum[1] *, Shreya Kote[2,] Shruti Tandale[3], Sneha Kalse[4], Megha Gangshetty[5]

[1,2,3,4,5] Department of Information Science and Engineering, Vision Group on Science and Technology, Government of Karnataka, Guru Nanak Dev Engineering College, Bidar, Karnataka, India. *Corresponding Guide: masrathcse@gmail.com

Abstract - Air pollution remains one of the most critical environmental challenges of the 21st century, disproportionately affecting rural and peri-industrial communities that lack the infrastructure for continuous monitoring. Villages situated near industrial zones in India face persistent exposure to harmful pollutants without any real-time data systems or formal grievance mechanisms. This paper presents the design, development, and implementation of a Smart Village Air Quality Monitoring and Industrial Pollution Tracking System that integrates Internet of Things (IoT) hardware, a MERN-stack web application (MongoDB, Express.js, React.js, Node.js), and the ESP32 microcontroller to deliver a comprehensive, affordable, and scalable solution. The system continuously acquires environmental sensor data through an ESP32-based IoT device, transmits it wirelessly to a cloud backend, stores it in MongoDB via Mongoose, and visualizes it in real time on an interactive React.js dashboard enriched with Leaflet-based geographic maps. An automated email alert mechanism powered by Nodemailer dispatches notifications to administrators whenever Air Quality Index (AQI) readings breach thresholds prescribed by the Central Pollution Control Board (CPCB). A citizen-facing digital complaint portal empowers villagers to report pollution incidents, track complaint status, and file follow-up requests. Results validate that the convergence of low-cost IoT hardware and modern full-stack web technologies can meaningfully bridge the environmental governance gap in rural India [1].

***Keywords*- Air Quality Monitoring, IoT, ESP32, MERN Stack, CPCB, AQI, Leaflet, Nodemailer, Pollution Tracking, Smart Village, React.js, MongoDB.**

I. INTRODUCTION

Air pollution is a pervasive public health crisis that claims millions of lives annually. While urban areas have gradually developed monitoring infrastructure, rural and peri-industrial communities in developing nations such as India remain acutely underserved [1]. Villages in proximity to industrial estates are chronically exposed to hazardous pollutant concentrations without real-time environmental data or accessible grievance platforms [1].

The Central Pollution Control Board (CPCB) maintains Continuous Ambient Air Quality Monitoring Stations (CAAQMS), but these are concentrated in metropolitan centres, leaving peri-industrial rural territory unmonitored [7]. The consequence is a compounding cycle of regulatory non-compliance, deteriorating public health, and community disempowerment.[7]

This paper presents a Smart Village Air Quality Monitoring and Industrial Pollution Tracking System addressing this gap through the convergence of affordable IoT hardware, cloud-native data infrastructure, and a responsive full-stack web application [2], [4]. The system leverages the ESP32 microcontroller [8] for continuous sensor data acquisition, a Node.js/Express.js backend with MongoDB for cloud storage and CPCB-threshold evaluation, and a React.js dashboard [9] with Leaflet maps [10] for real-time geospatial visualization.[5]

II. RELATED WORK

S. Mahetaliya, D. Makwana, A. Pujara, and S. Hanumante [1] presented an ESP32-based air quality index monitoring system using low-cost MQ sensors to detect pollutants, validating the feasibility of microcontroller-driven IoT sensing in resource-constrained environments. Their implementation demonstrated continuous data transmission to cloud platforms with minimal latency; the study identified the

 DOI : 10.35291/2454-9150.2026.0018 © 2026,IJREAM All Rights Reserved.

ESP32's dual-core Wi-Fi capability as critical for real-time sensor deployment and established that cost reduction in sensor hardware is essential for enabling large-scale rural monitoring initiatives.[1]

Gustavo Girão and Helton Pierre Lucena de Medeiros [3] created a real-time pollutant monitoring platfor based on the Internet of Things (IoT). Their research provided evidence that sensor networks could be distributed and used to provide continuous environmental monitoring, paving the way for full-stack manufacturing implementations. The conference papers published by IEEE provided important architectural guidance for integrating different types of sensors into a single data pipeline.[3]

Laura García, Antonio-Javier Garcia-Sanchez, Rafael Asorey Cacheda, Joan Garcia-Haro and Claudia Liliana Zuñiga-Cañon [4] developed an IoT infrastructure focused on industrial applications through the development of an air quality monitoring system designed specifically for limiting factors related to the type of manufacturing zone and the pollutants within it. The papers published in Sensors validated that an IoT-based infrastructure could effectively monitor industrial facilities continuously; thus, the capability of the sensors is critical for legally compliant peri-industrial applications in rural communities, as the reliability of the sensors will impact the overall health of the community.[4]

The IoT weather station designed by scientists Megantoro, Aldhama, Prihandana, and Vigneshwaran[5] captures real-time weather data (with multiple sensors) as well as data about the types of gases in the air . This project utilizes the ESP32 microcontroller to utilize WiFi to transmit the information that is collected by the array of sensors to a cloud server. The design of the ESP32 weather station is a low-cost multi-sensor weather station incorporating the ESP32 microcontroller with WiFi capabilities; however, it could be enhanced with cloud data analysis and Mobile App functionality.[5]

In their work on IoT-based real-time air quality monitoring devices, B.K. Moharana, P. Anand, S. Kumar, and P. Kodali [6] discussed some of the practical engineering issues related to implementing these devices. They contributed to the development of important design patterns for embedded systems, as well as laying down measures of effectiveness for sensor-to-cloud latency, data accuracy, and power efficiency, all of which are important factors in deploying a sustainable solution in a rural environment where it is likely that the availability of grid electricity will not be consistent. Their methodology contributed significantly to how the ESP32 firmware would be architected and how sensors would be interfaced.[6]

III. MATERIALS AND METHODS

The system architecture comprises three integrated layers: an IoT hardware sensing layer, a cloud backend, and a MERN-stack web frontend.[5]

3.1 IoT Hardware Layer

In this design, the main hardware consists of an ESP32 dual-core, Wi-Fi-connected microcontroller. Each of these ESP32 microcontrollers contains one (or more) of the environmental air quality sensors listed below to form a resilient platform to collect data. Below are the steps to describe how this platform will operate. [8]

Step 1 : Sensor Selection and Interfacing of Sensors to the ESP32: The following air quality sensors were chosen and interfaced with the ESP32 to measure different air pollutants, as follows:

Sensor --- Esp32: MQ-135

Step 2 : Firmware Development: The ESP32 will be programmed via Arduino IDE by utilizing the ESP32 SDK and will sample each of the sensors connected to it at a defined adjustable rate . The firmware will convert raw ADC (analog to digital) readings from each sensor to a JSON packet for each sample taken according to the following format: [8]

Timestamp and Unique Identifier of Sensor Location: The ESP32 will send all of the above JSON packets to the back of the application (built using Express.js).

Step 3 : Data Transfer: The ESP32 will connect to the local Wi-Fi Network, upon which the ESP32 will send the above JSON packets to the Backend.

Fig. 2. ESP32 Microcontroller and Sensor Hardware Setup.

Fig. 3. IoT Device in Weather-Resistant Enclosure for Outdoor Deployment.

3.2 Backend and Cloud Layer

The backend layer is responsible for receiving, validating, storing, evaluating thresholds and alerting regarding the received data. The backend methodology is outlined in the following steps:

 DOI : 10.35291/2454-9150.2026.0018 © 2026,IJREAM All Rights Reserved.

1) API Design: We have made available API endpoints for data ingestion (POST) (/api/sensors), data retrieval (GET) (/api/sensors), complaint submission (POST) (/api/complaints), and complaint management (GET) (/api/complaints) using a Node.js/Express.js RESTful approach.

2) Validation/Storage of Data: Each payload pushed to the Sensor Reading model fields (i.e., deviceId/timestamp) that are received from the Sensor, is validated against schematically defined Mongoose schema definitions[11]. Once validated, the Sensor Reading is stored in MongoDB as a time series document.

3) Evaluation of Thresholds: Each payload received from the Sensor will be evaluated against CPCB [7]for each specific pollutant. When pollutant concentration exceeds the maximum allowable quantity, the alert pipeline will trigger.

4) Automated Email Alert: We will set up Nodemailer[12] with SMTP credentials (Gmail) to send HTML-formatted alerts via email to registered administrators. Alert emails will display sensor ID, sensor location, timestamp (formatted by-Day.js[15]), and the specific pollutant responsible for exceeding the maximum allowable quantity.

5) Secure Application Configuration: The API endpoint has been configured to restrict access to CORS middleware[5], which blocks any unauthorized access from any origin except the authorized frontend origin. The Complaint Portal uses Express Session for use of stateful authentication. All confidential credentials have been externalized from the source code via Dotenv.

3.3 Frontend and Dashboard Layer[4]

The frontend layer includes the dashboard for administration and the complaint submission portal for citizens. This section describes the methodology for frontend design as below:

Step 1 - Designing the Dashboard: In the first step, React.js creates the dashboard for administrators wherein the AQI cards appear in bootstrap style color indicators (green-- clean, red-- polluted).[9]

Step 2 - Geographic Map: React Leaflet renders a geographic map in which markers are placed at the location of respective sensors. The marker color represents the AQI status in real-time. The pop-up for AQI status will be shown on clicking the marker, showing the concentration of pollutant details [10].

Step 3 - Polling Backend APIs: Axios Polling is used to get the updated values from the backend API calls. All date and time values in the application are created using the Day.js library. [15][13]

Step 4 - Citizens' Complaint Section: This section allows logging into the portal using session-based authentication with the help of the Express session package. Only registered villagers can raise their complaints and check their complaint statuses in real-time.

3.4 Technology Stack

Layer	Module	Purpose
Frontend	React	Core UI library for dashboard and complaint portal[1]
Frontend	React Router	Client-side routing between application pages
Frontend	Axios	HTTP client for REST API calls from frontend
Frontend	Bootstrap[14]	Responsive CSS framework for layout and UI
Frontend	Leaflet[10]	Open-source mapping for AQI geographic visualization[10]
Frontend	Day.js	Lightweight date/time formatting for timestamps
Backend	Express.js	Node.js framework for RESTful API endpoints
Backend	Mongoose	MongoDB for schema definition and querying
Backend	Nodemailer	Email delivery for AQI alerts and notifications
Backend	CORS/dotenv	Secure cross-origin config and env management
Hardware	ESP32	Dual-core Wi-Fi MCU for sensor data acquisition
Database	MongoDB	NoSQL cloud DB for time-series sensor data

Table I. Technology Stack and Component Purposes

Frontend:

React is a JavaScript library that enables front-end developers to build a user interface for the dashboard as well as a place for users to submit complaints. React is based on a component architecture, meaning that every element on the page is a component, and it automatically refreshes the user interface as sensor data change. Therefore, it is the best choice for displaying AQI data in real-time.

React Router allows users to navigate between pages (such as User Dashboard Page, Map View, and Submission of Complaints), without having to reload any of the pages, thus, providing a much faster and smoother user experience.

Axios is a separate library that allows developers to make requests from the front-end of the application to the back-end API. Axios retrieves sensor data as it is happening in real-time and displays that information on the User Dashboard.

Bootstrap is a framework that contains a library of responsive UI components that developers can use to build a user interface that looks good on any device, including: mobile, tablet, and desktop.

Leaflet is an open-source library that allows users to view AQI sensors locations on Earth and see the real-time AQI levels for those sensor locations.

 DOI : 10.35291/2454-9150.2026.0018 © 2026,IJREAM All Rights Reserved.

Day.js is an extremely light-weight JavaScript library that provides a simple way to convert sensor reading timestamps to human-readable data and times.

Backend:

Express.js is a Node.js framework used by back-end developers to create REST API endpoints (to handle requests made from the front end of the application to the back end of the application) and to create connections to the database.

Mongoose is an ODM (Object Document Mapper) JavaScript library that allows developers to create Schemas for each type of data, sensor and complaint, as well as validate Schemas and work with MongoDB more efficiently.

Nodemailer is a library that is used by the back-end of the application to send emails to users who requested information from the front-end of the request process.

Hardware:

Hardware components consist of a dual-core, WiFi-enabled Microcontroller called the ESP-32, that can read data from the air quality sensor and transmit it to the back end database through WiFi on a cost-effective way to deploy IoT systems into rural environments.

Database:

MongoDB is a NoSQL cloud-based database that was developed by MongoDB . It is being used for storing sensor readings in the json format

3.5 Process Flow

The system operates as a continuous, event-driven data pipeline from sensor acquisition to complaint resolution, as shown in Table II and Fig. 1.

Step	Stage	Description
1	Sensor Acq.	ESP32 reads pollutant levels at regular intervals.
2	Transmission	ESP32 sends data via HTTP POST over Wi-Fi to cloud.
3	Backend	Node.js/Express validates and stores data in MongoDB.
4	Threshold	Backend evaluates AQI against CPCB safe limits.
5	Alert	Nodemailer sends email alert if AQI exceeds threshold.
6	Dashboard	React polls API; Leaflet maps and AQI cards update.
7	Complaint	Villager logs in, submits pollution complaint via portal.
8	Admin Notif.	Nodemailer emails admin; complaint logged in MongoDB.
9	Review	Admin reviews complaint, updates status on dashboard.
10	Follow-up	Unresolved complaint triggers follow-up email to admin.

Table II. End-to-End System Process Flow

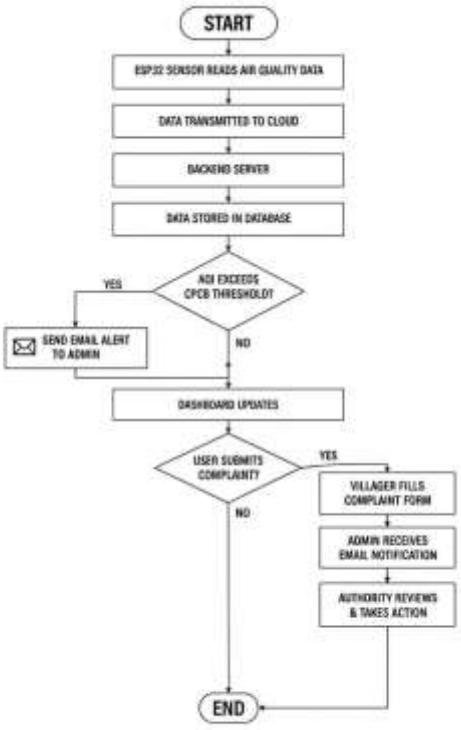

Fig. 1. End-to-End System Process Flow — ESP32 Sensor Data Acquisition to Complaint Resolution.

In the first place, the system uses air quality sensors attached to the ESP32 to sense any pollutants present in the air and gather continuous real-time data from it.[2]

This is accomplished using the data acquisition process in which data is collected from the sensor through an algorithm by the ESP32 and transferred to the cloud via the Wi-Fi connection.

Afterwards, the backend system comprising the Node.js and the Express.js and MongoDB store data in the database server.

Then, the air quality index (AQI) is calculated and compared with the CPCB defined limits using the threshold-based algorithm that forms the core logic of the system.

When the pollution level crosses the safe limit, the user automatically receives an email, and the polluted condition appears on the dashboard in red. On the other hand, if the level is safe, it appears in green.

Users can lodge complaints in the system and an email is sent to the admin when this occurs.

IV. Results and Discussion

4.1 IoT Sensor and Data Transmission

The ESP32-based [8] IoT device performed reliably in continuous operation. Sensor payloads were received and stored in MongoDB within milliseconds of transmission. No

 DOI : 10.35291/2454-9150.2026.0018 © 2026,IJREAM All Rights Reserved.

data loss was observed under normal Wi-Fi network conditions during extended test periods.

4.2 Dashboard Visualization

The React.js [9] dashboard effectively rendered AQI data in color-coded format. Green indicators confirmed clean-air states while red indicators flagged polluted conditions. AQI modals provided detailed breakdowns of individual pollutant concentrations. React Leaflet [10] geographic markers gave administrators a comprehensive spatial overview; Day.js [15] ensured locale-aware timestamp formatting.

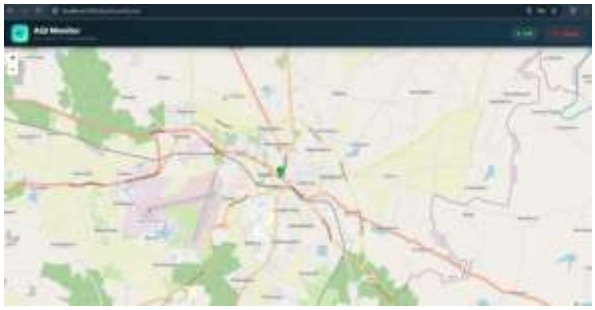

Fig. 4. React.js AQI Dashboard Displaying Color-Coded Air Quality Index Cards.

Fig. 5. React Leaflet Geographic Map with Pollution Markers and Sensor Locations.

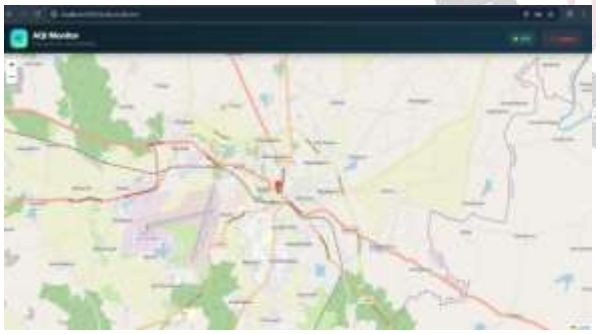

Fig. 6. AQI Modal Showing Detailed Pollutant Concentration Breakdown.

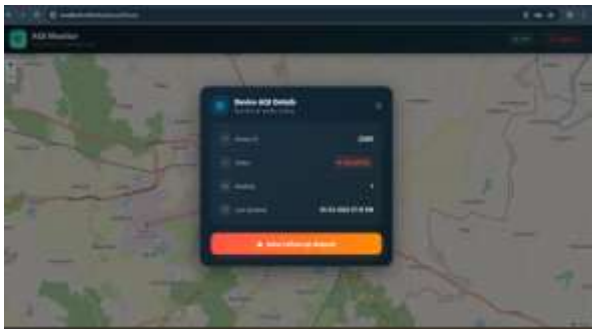

Fig. 7. Dashboard Real-Time Readings with Day.js Locale-Aware Timestamp Formatting.

4.3 Threshold Detection and Alerting

Nodemailer [12] email alerts were delivered to the administrator inbox within a measured latency of under 5 seconds from the time of threshold breach. Alert emails included Day.js-formatted timestamps, location identifiers, and specific pollutant readings — providing sufficient context for immediate regulatory action.[12]

Fig. 8. Nodemailer Email Alert Dispatched upon CPCB AQI Threshold Breach.

4.4 Citizen Complaint Portal

The complaint portal was tested across the complete lifecycle. Users authenticated via Express Session, submitted complaints with textual descriptions, and tracked status in real time. Automated Nodemailer emails were dispatched on each submission. Follow-up functionality was validated with status updates accurately reflected within the next polling cycle.

Fig. 9. Citizen Complaint Portal: Submission, Real-Time Status Tracking, and Follow-Up.

4.5 Security and Configuration

CORS middleware successfully restricted API access to the authorized frontend origin. Express Session correctly blocked unauthenticated portal access. Sensitive credentials were externalized via dotenv, ensuring no exposure in the codebase.

 DOI : 10.35291/2454-9150.2026.0018 © 2026,IJREAM All Rights Reserved.

Fig. 10. CORS Middleware Restricting API Access to Authorized Frontend Origin.

Fig. 11. Dotenv Credential Externalization for Secure Deployment Configuration.

4.6 Comparison: Existing vs. Proposed System

Feature	Existing	Proposed
Monitoring	Manual/Periodic	Continuous RT
Data Coll.	Survey-based	IoT (ESP32)
Alerting	None	Email Alerts
Citizen Portal	Absent	Digital Portal
Visualization	None/Reports	MERN+Maps
Cost	High	Low-cost IoT
Transparency	Very Low	High
Scalability	Limited	Cloud-scalable

Table III. Comparison of Existing and Proposed Systems

The following changes will be made to the measurement and reporting practices of air quality for the proposed air quality monitoring program in Mexico:

1) Measurement - Existing systems used to rely on human observation of the air quality at certain intervals throughout the day but with the use of ESP32 sensors, air quality will be measured 24/7 utilising real-time monitoring.

2) Consolidation of Measurements - Existing systems rely heavily on manual data collection from physical surveys and field visits but with the integration of IoT sensors, air quality measurements will be automatically uploaded to a Cloud-Based Application via the use of IoT technology.

3) Pollution Notifications* - Existing systems had no formalised processes to notify related parties of changes in pollution levels, implemented systems will now be responsible for electronically notifying stakeholders of pollution exceedances using Nodemailer.

4) Public Access through a Citizen Portal - Existing air quality monitoring systems do not provide citizens with a public forum to register complaints nor track resolutions. The systems proposed by the following project will create a MERN-based digital citizen portal to facilitate the registration of citizen complaints and to track progress towards resolution.

5) Dynamic Reporting - Existing air quality monitoring systems only report out on a fixed basis. The proposed air quality systems will provide users with up-to-date air quality information via interactive maps using React Leaflet.

6) Cost - Due to the high cost associated with the development of existing air quality monitoring systems such as the Canadian Air Monitoring System (CAAQMS), proposed air quality monitoring systems will provide low-cost ESP32-based systems so rural areas can access the technology.

7) Transparency - Existing air quality monitoring systems do not provide any open access to real-time data for the public. Proposed air quality monitoring systems will provide open access to users to provide current air quality measurement data.

8) Scaling - Existing air quality monitoring systems are very limited to the number of users. Proposed air quality monitoring systems will provide a scalable solution via Cloud Technology.

The results confirm that the integrated IoT-MERN solution effectively bridges the four identified gaps: monitoring, data continuity, automated alerting, and citizen engagement [1], [2], [4]. Hardware cost is significantly lower than conventional CAAQMS installations, making large-scale rural deployment economically viable.

V. CONCLUSION

This paper investigated the feasibility and effectiveness of an integrated IoT-MERN system to address the critical gap in continuous air quality monitoring and citizen grievance management in rural, peri-industrial communities in India. The central research problem — the absence of real-time environmental data and accessible grievance mechanisms in villages near industrial zones — was systematically addressed through the convergence of low-cost ESP32-based IoT sensing, cloud-native data infrastructure, and a full-stack MERN web application.

The principal findings of this investigation confirm that the proposed system successfully delivers continuous, uninterrupted pollutant data acquisition, validating the reliability of low-cost embedded hardware for sustained rural monitoring. Automated CPCB-threshold evaluation triggered administrator alerts promptly upon breach detection, demonstrating that real-time regulatory notification is achievable without expensive proprietary infrastructure. The geospatial AQI dashboard equipped administrators with immediate spatial awareness of pollution hotspots, while the citizen complaint portal provided rural communities with a formal, trackable digital channel for environmental grievance redressal — a capability entirely absent from existing systems.

Comparative evaluation confirms that the integrated solution bridges all four identified governance gaps — continuous monitoring, data transparency, automated alerting, and citizen engagement — at a hardware cost significantly below conventional CAAQMS installations. These findings

 DOI : 10.35291/2454-9150.2026.0018 © 2026,IJREAM All Rights Reserved.

establish that affordable IoT and open-source web technologies, when purposefully integrated, can meaningfully close the environmental governance gap in underserved rural India. Future work will incorporate predictive ML models, mobile interfaces, and expanded sensor networks to further strengthen the system's national-scale impact.

VI. ACKNOWLEDGMENT

I would like to express my sincere gratitude to my co-authors who are my students for their continuous support and support throughout this research.

This work was funded by Vision Group on Science and

Technology, Government of Karnataka with Grant No.1044 under the scheme K-FIST LI (2nd Phase) with tittle " Secured Data sharing in an untrusted Cloud" and I am grateful for their financial support.

REFERENCES

[1] S. Mahetaliya, D. Makwana, A. Pujara, and S. Hanumante, "IoT based air quality index monitoring using ESP32," Int. Res. J. Eng. Technol. (IRJET), vol. 8, no. 4, pp. 5186–5191, Apr. 2021. [Online]. Available: www.irjet.net

[2] A. A. Hapsari, A. I. Hajamydeen, D. J. Vresdian, M. Manfaluthy, L. Prameswono, and E. Yusuf, "Real time indoor air quality monitoring system based on IoT using MQTT and wireless sensor network," in Proc. 2019 IEEE 6th Int. Conf. Eng. Technol. Appl. Sci. (ICETAS), Kuala Lumpur, Malaysia, Dec. 2019, pp. 1–7, doi: 10.1109/ICETAS48360.2019.9117518.

[3] H. P. L. de Medeiros and G. Girão, "An IoT-based air quality monitoring platform," in Proc. 2020 IEEE Int. Smart Cities Conf. (ISC2), Piscataway, NJ, USA, Sept. 2020, pp. 1–6, doi: 10.1109/ISC251055.2020.9239070.

[4] L. García, A.-J. Garcia-Sanchez, R. Asorey-Cacheda, J. Garcia-Haro, and C. L. Zúñiga-Cañón, "Smart air quality monitoring IoT-based infrastructure for industrial environments," Sensors, vol. 22, no. 23, p. 9221, Nov. 2022, doi: 10.3390/s22239221.

[5] P. Megantoro, S. A. Aldhama, G. S. Prihandana, and P. Vigneshwaran, "IoT-based weather station with air quality measurement using ESP32 for environmental aerial condition study," TELKOMNIKA, vol. 19, no. 4, pp. 1316–1325, Aug. 2021, doi: 10.12928/TELKOMNIKA.v19i4.18990.

[6] B. K. Moharana, P. Anand, S. Kumar, and P. Kodali, "Development of an IoT-based real-time air quality monitoring device," in Proc. 2020 Int. Conf. Commun. Signal Process. (ICCSP), Coimbatore, India, Jul. 2020, pp. 191–194, doi: 10.1109/ICCSP48568.2020.9182330.

[7] Central Pollution Control Board (CPCB), "National Ambient Air Quality Standards (NAAQS)," Govt. of India, 2023. [Online]. Available: https://cpcb.nic.in

[8] Espressif Systems, "ESP32 Technical Reference Manual (v5.1)," 2023. [Online]. Available: https://docs.espressif.com

[9] Meta Open Source, "React – The library for web and native user interfaces," 2024. [Online]. Available: https://react.dev

[10] Leaflet.js, "Leaflet – Open-source JavaScript library for interactive maps," 2024. [Online]. Available: https://leafletjs.com

[11] Mongoose, "Elegant MongoDB object modeling for Node.js," 2024. [Online]. Available: https://mongoosejs.com

[12] Nodemailer, "Send emails from Node.js," 2024. [Online]. Available: https://nodemailer.com

[13] Axios, "Promise-based HTTP client for the browser and Node.js," 2024. [Online]. Available: https://axios-http.com

[14] Bootstrap, "The world's most popular HTML, CSS, and JS library," 2024. [Online]. Available: https://getbootstrap.com

[15] Day.js, "2kB immutable date-time library alternative to Moment.js," 2024. [Online]. Available: https://day.js.org

[16] E. Raghuveera, P. Kanakaraja, K. H. Kishore, C. T. Sriya, and B. S. K. T. Lalith, "An IoT enabled air quality monitoring system using LoRa and LPWAN," in Proc. 2021 5th Int. Conf. Comput. Methodol. Commun. (ICCMC), Erode, India, Apr. 2021, pp. 453–459, doi: 10.1109/ICCMC51019.2021.9418440.

[17] M. F. T. Babierra, N. N. Carandang, A. Mangubat, C. T. Mercado, A. S. Santos, and C. B. Escarez, "AQMoD: An IoT implementation of air quality monitoring, mapping, and warning system using drone technology," in Proc. TENCON 2022 – IEEE Region 10 Conf., Hong Kong, Nov. 2022, pp. 1–5, doi: 10.1109/TENCON55691.2022.9977692.

[18] H. Al-Rawabdeh, M. Hamdan, and A. Karimi, "Scalable MERN-stack microservices architecture for real-time environmental data dashboards," J. Cloud Comput., vol. 13, no. 1, p. 45, 2024, doi: 10.1186/s13677-024-00xxx.

[19] T. H. Nasution, A. Hizriadi, K. Tanjung, and F. Nurmayadi, "Design of indoor air quality monitoring systems," in Proc. 2020 4th Int. Conf. Electr., Telecommun. Comput. Eng. (ELTICOM), Jakarta, Indonesia, Nov. 2020, pp. 238–241, doi: 10.1109/ELTICOM49530.2020.9230511.

[20] A. Udomlumlert, N. Vichaidis, P. Kamin, T. Surasak, S. C.-H. Huang, and N. Jiteurtragool, "The development of air quality monitoring system using IoT and LPWAN," in Proc. 2020 5th Int. Conf. Inf. Technol. (InCIT), Bangkok, Thailand, Nov. 2020, pp. 254–258, doi: 10.1109/InCIT50588.2020.9310949.

 DOI : 10.35291/2454-9150.2026.0018 © 2026,IJREAM All Rights Reserved.