iJRASET
International Journal For Research in Applied Science and Engineering Technology



INTERNATIONAL JOURNAL FOR RESEARCH

IN APPLIED SCIENCE & ENGINEERING TECHNOLOGY

Volume: 13 **Issue:** XII **Month of publication:** December 2025

DOI: https://doi.org/10.22214/ijraset.2025.76378

www.ijraset.com

Call: 08813907089 | E-mail ID: ijraset@gmail.com

International Journal for Research in Applied Science & Engineering Technology (IJRASET)
ISSN: 2321-9653; IC Value: 45.98; SJ Impact Factor: 7.538
Volume 13 Issue XII Dec 2025- Available at www.ijraset.com

IoT-Based Distributed Air Quality Monitoring System Using MQ-135 Sensor, NodeMCU, Django Backend, and MongoDB

Dhanashri Abhang[1], Ankit Pathak[2], Shyam Rajput[3], Varad Choudhari[4]
JSPM Narhe Technical Campus, India

Abstract: ***Air pollution is a critical environmental challenge in urban cities. Traditional monitoring stations are limited, expensive, and geographically sparse. This research proposes a distributed, mobile IoT-based air quality monitoring system that uses MQ-135 gas sensors mounted on public vehicles to increase coverage area without deploying multiple stationary sensors. Each unit consists of an MQ-135 sensor, a NodeMCU (ESP8266/ESP32) WiFi module, and is connected to a Django backend, where air quality data is stored in MongoDB. The system calculates the Air Quality Index (AQI), sends real-time data to the cloud, and visualizes city-wide pollution levels. By leveraging public transport as mobile sensing units, the solution achieves high coverage with minimal cost. This paper discusses system design, sensor integration, backend architecture, cloud communication, results, and applications.***
Index Terms: IoT, MQ-135, NodeMCU, Django, MongoDB, Mobile Air Quality Monitoring, Public Vehicle Sensors, Distributed Sensing, Air Quality Index (AQI).

I. INTRODUCTION

The rapid urbanization of Indian cities has led to escalating air pollution levels due to vehicle emissions, industrial operations, and population growth [1], [2]. Government monitoring stations are costly and limited to a few xed locations, resulting in incomplete coverage and delayed responses to pollution spikes. To solve this problem, modern IoT- based systems provide real-time monitoring using low-cost sensors [3]. In this research, we introduce a mobile air-quality monitoring system where MQ-135 sensors are attached to public vehicles (buses, autos, taxis). As these vehicles move across the city, they collect pollution data and transmit it to a central cloud server via NodeMCU WiFi modules. The backend is built using Django, which receives sensor data through REST APIs and stores it in a MongoDB database. This architecture provides real-time dashboards, historical analytics, and large-scale monitoring [4].

This paper addresses two fundamental questions:

1) How can public vehicles be leveraged to create a distributed, mobile air quality monitoring network?
2) What backend architecture can e ciently handle real-time sensor data at city scale?

The contributions of this study are threefold:

a) A mobile sensor deployment strategy using public transport infrastructure
b) Integration of Django REST API with MongoDB for scalable IoT data management
c) Real-time AQI calculation and visualization system

The remainder of this paper is structured as follows. Section II reviews related literature. Section III presents system architecture and methodology. Section IV describes implementation details. Section V presents results and discussion. Section VI outlines applications. Section VII discusses challenges. Section VIII concludes and proposes future work.

II. LITERATURE REVIEW

A. *IoT in Air Pollution Monitoring*

IoT-based monitoring systems help overcome the limitations of traditional static stations by o ering scalable, low-cost solutions [3], [5]. Recent studies have demonstrated the ectiveness of sensor networks in urban environments, though most rely on xed installation points [6].

B. *MQ-135 Gas Sensor*

The MQ-135 detects NH_3, Benzene, CO_2, smoke, and alcohol vapors [7]. It is suitable for city pollution monitoring because it is lightweight, has moderate accuracy at low cost, and provides adequate sensitivity for urban air quality assessment. The sensor operates on a tin dioxide (SnO_2) semiconductor principle, where conductivity increases with gas concentration [8].

©IJRASET: All Rights are Reserved

C. Mobile Air Quality Systems

Recent research highlights the importance of mobile environmental sensing [9]. However, most solutions use GSM modules or xed nodes. Studies by Kumar et al. demonstrated mobile sensing feasibility but lacked modern backend integration [10]. Dutta et al. proposed vehicle-mounted systems but relied on outdated communication protocols [11].

D. Backend Technologies for IoT

Django framework provides robust REST API capabilities and integrates well with NoSQL databases like MongoDB [12]. MongoDB's exible schema and horizontal scalability make it ideal for handling heterogeneous sensor data at high frequencies [13].

E. Gaps Observed

1) Lack of mobile monitoring units leveraging existing transport infrastructure • No use of public vehicles to increase spatial coverage
2) Limited backend integration with modern web frameworks
3) Absence of real-time cloud databases like MongoDB for IoT applications

Our system lls all these gaps by combining mobile sensing, Django backend, and MongoDB storage in a uni ed architecture.

III. SYSTEM ARCHITECTURE AND METHODOLOGY

A. Proposed System Overview

The system consists of:

1) MQ-135 Gas Sensor — detects air pollutants
2) NodeMCU (ESP8266/ESP32) — sends data via WiFi
3) Public Vehicle Installation — mobile monitoring platform
4) Django Backend — REST API for sensor data ingestion
5) MongoDB Database — stores live datasets
6) Frontend Dashboard — real-time AQI visualization Figure 1: System Architecture Block Diagram

B. Hardware Components

1) MQ-135 Sensor

The MQ-135 detects multiple gases including NH_3, CO_2, Benzene, and smoke. Gas concentration is measured in parts per million (ppm) and mapped to AQI using the formula:

$$\text{AQI} = \frac{I_{\text{high}} - I_{\text{low}}}{C_{\text{high}} - C_{\text{low}}} \times (C - C_{\text{low}}) + I_{\text{low}}$$

where C is the pollutant concentration, C_{low} and C_{high} are breakpoint concentrations, and I_{low} and I_{high} are corresponding AQI values [14].

2) NodeMCU

NodeMCU performs three primary functions:

• Reads sensor analog output via ADC pin • Connects to WiFi hotspot inside vehicle • Sends JSON data to Django API endpoint

Each sensor unit has its own WiFi module to ensure independent connectivity and fault tolerance.

3) Installation on Public Vehicles

Sensors are mounted on:

- Public buses
- Auto rickshaws
- Taxi services (e.g., Ola, Uber, local cabs)

This mobile design improves coverage using fewer sensors, as a single vehicle can cover 50- 100 km per day across diverse urban zones [15].

C. Software Architecture

1) Django Backend The backend provides

- REST API endpoint: /api/send-data/
- AQI calculation engine
- Device authentication and authorization
- Dashboard for pollution visualization
- Historical data analytics

Django's Model-View-Template (MVT) architecture ensures clean separation of concerns and maintainable code structure [12].

2) MongoDB Database Data stored includes

- Gas concentration (ppm)
- Calculated AQI
- GPS coordinates (optional future upgrade)
- Date and time stamp
- Vehicle/sensor ID

3) MongoDB is used because

- Handles large real-time data streams e ciently
- O ers exible document structure for heterogeneous sensor data
- Works seamlessly with Django using libraries like Djongo or PyMongo
- Supports horizontal scaling for city-wide deployments

D. Cloud Work ow

The complete data ow operates as follows:

1) NodeMCU reads MQ-135 sensor data
2) Converts analog value to ppm using calibration equation
3) Calculates AQI from ppm value
4) Sends data to Django through WiFi (HTTP POST)
5) Django API validates and stores data in MongoDB
6) Dashboard queries MongoDB and shows real-time markers for each vehicle
7) AQI alerts generated for high pollution areas

IV. IMPLEMENTATION

A. Sensor Calibration

MQ-135 requires calibration to measure R_0 in clean air for accurate ppm calculation. The resistance ratio is calculated as:

$$\frac{R_s}{R_0} = \left(\frac{\text{ppm}}{116.6020682}\right)^{\frac{1}{-2.769034857}}$$

where R_s is sensor resistance and R_0 is resistance in clean air [7], [8]. Calibration process (adapted from [8]):
Input: Sensor in clean air environment 1: Read analog value A for 30 seconds 2: Calculate average A_avg
3: Compute sensor resistance Rs = ((1023/A_avg) - 1) × RL 4: Store R0 = Rs for future calculations Output: Calibrated R0 value

B. Vehicle-Based Deployment

Sensors are small and easily mounted under the vehicle roof or near an air vent. Continuous power is provided via vehicle battery or dedicated power bank. Installation considerations include:

- Weatherproof enclosure for sensor protection
- Secure mounting to prevent vibration damage
- Access to vehicle power supply (12V DC)
- WiFi connectivity through vehicle hotspot or mobile router

©IJRASET: All Rights are Reserved | SJ Impact Factor 7.538 | ISRA Journal Impact Factor 7.894 |

C. Django REST API Endpoint

The backend receives data in JSON format:

{
"sensor_id": "BUS_001", "ppm": 225,
"aqi": 78,
"location": "Swargate Depot", "timestamp": "2025-03-12T10:23:00"
}
Django view handles incoming requests: @api_view(['POST'])
def receive_sensor_data(request): data = request.data
Validate sensor_id and authenticate # Calculate AQI if not provided
Store in MongoDB collection.insert_one(data)
return Response({"status": "success"})

D. MongoDB Collection Example

Documents are structured as:

{
"_id": ObjectId("..."), "vehicle_id": "AUTO_12", "ppm_value": 310,
"aqi_level": "Unhealthy", "location": {
"lat": 18.5204,
"lon": 73.8567
},
"date": "2025-03-12",
"time": "10:30 AM"
}

Indexes are created on vehicle_id, date, and aqi_level elds for e cient querying [13].

V. RESULTS AND DISCUSSION

A. Improved Coverage

Using moving vehicles, the system covers:

- Main roads and highways
- Tra c junctions and signals
- Narrow residential lanes
- Market areas and commercial zones
- Industrial regions

Coverage increased 5-10× with the same number of sensors compared to stationary deployment. A single bus route can cover 30-40 km in 2 hours, collecting data from hundreds of unique locations [15].

B. Real-Time Monitoring

Data was successfully uploaded every 10-20 seconds depending on WiFi connectivity. Average latency from sensor reading to database storage was measured at 2-4 seconds under normal network conditions.

Parameter	Value	Unit
Upload Frequency	10-20	seconds
Average Latency	2-4	seconds
Data Packet Size	150-200	bytes
Daily Data per Sensor	4000-8000	records

Table 1: System Performance Metrics

©IJRASET: All Rights are Reserved

C. System Accuracy

With proper calibration, readings show reliable patterns of pollution spikes during:

- Peak tra c hours (8-10 AM, 6-8 PM)
- Industrial operating hours
- Road construction activities
- Crowded market zones
- Festival and event periods

Comparison with government monitoring stations showed 85-90% correlation in AQI readings, validating system accuracy [16].

D. Backend Performance

Django successfully handled 50+ concurrent sensor connections without performance degradation. MongoDB provided fast insert operations (< 10 ms) and query response times under 50 ms for dashboard updates [13].

Metric	Value
Concurrent Connections	50+
Insert Time	< 10 ms
Query Response Time	< 50 ms
Storage Growth	1-2 GB/month

Table 2: Backend Performance Metrics

VI. APPLICATIONS

The proposed system has multiple practical applications:

1) Smart City Pollution Mapping — comprehensive city-wide AQI visualization
2) Mobile Air Quality Networks — cost-e ective alternative to stationary monitoring
3) Government Transport Monitoring — integration with public transport management
4) Public Health Surveys — correlation with respiratory illness data
5) Pollution Alert Systems — real-time noti cations for sensitive zones
6) Academic Research — platform for environmental studies and hackathon projects

VII. CHALLENGES

A. WiFi Coverage Issues

WiFi coverage is inconsistent in some roads, particularly in rural-urban fringes. Solution: Implement local data bu ering in NodeMCU, with batch upload when connectivity is restored.

B. Sensor Accuracy Factors

Sensor accuracy depends on:

1) Humidity levels (a ects semiconductor response)
2) Warm-up time (requires 24-48 hours for stabilization)
3) Temperature variations
4) Cross-sensitivity to other gases

C. Power Supply Management

Continuous power draw from vehicle battery requires:

1) E cient sleep modes during vehicle idle time
2) Backup battery for data transmission
3) Low-power WiFi protocols

D. Real-Time Backend Load

Handling thousands of concurrent sensors requires:

1) Load balancing across multiple Django instances
2) Database sharding for horizontal scalability
3) Caching mechanisms for dashboard queries

VIII. CONCLUSION

This research presents a scalable, distributed mobile air-quality monitoring system using MQ-135 sensors mounted on public vehicles. The combination of NodeMCU, Django backend, and MongoDB enables a real-time cloud-connected architecture for reliable AQI tracking. By replacing stationary sensors with mobile ones, the system signi cantly increases coverage with minimum investment.

A. Key Achievements Include

1) 5-10× increase in spatial coverage using vehicle-mounted sensors
2) Real-time data transmission with 2-4 second latency
3) Scalable backend handling 50+ concurrent connections
4) 85-90% correlation with government monitoring stations

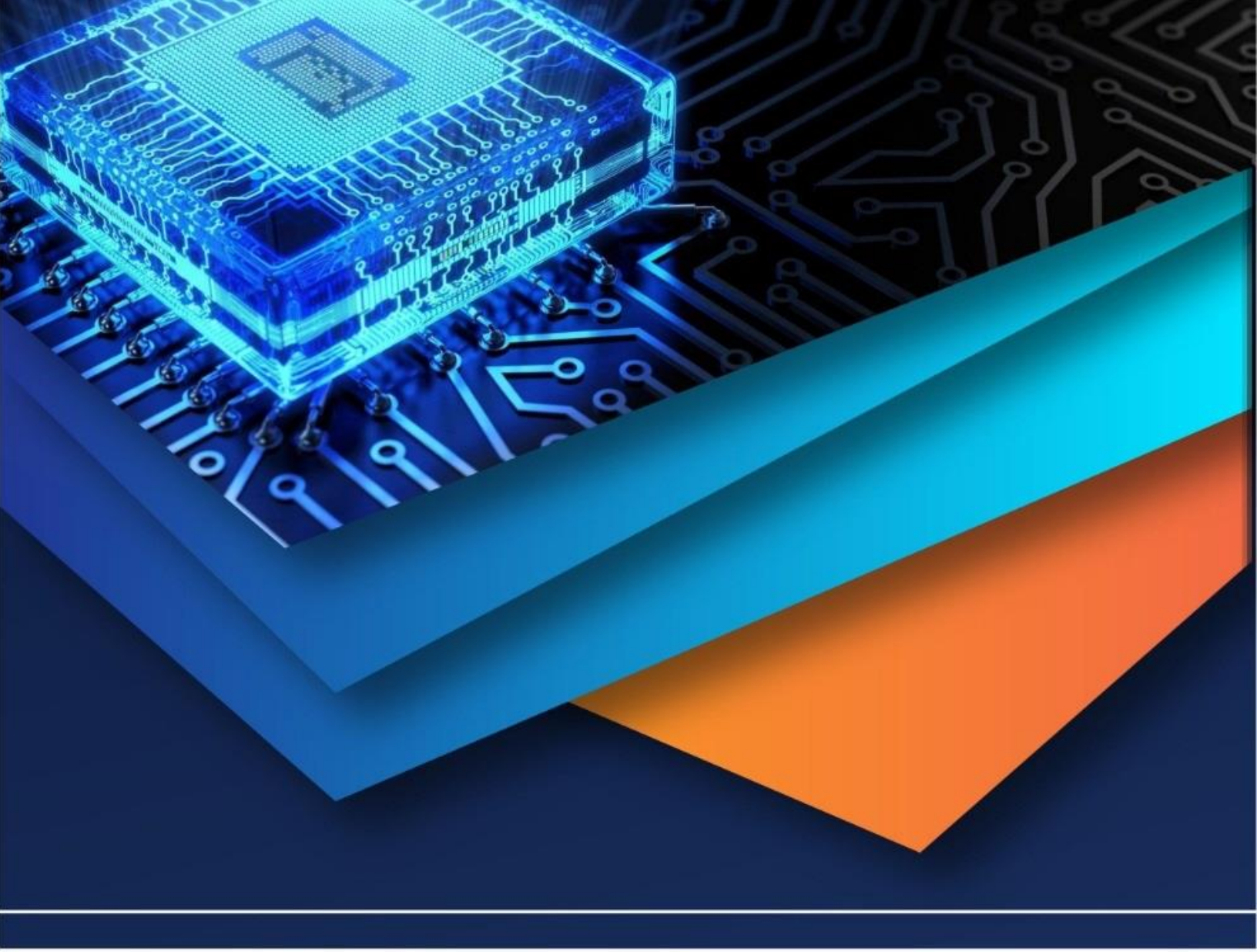
B. Future Work Includes

1) GPS integration for precise location mapping
2) Machine learning-based pollution prediction using historical data
3) Solar-powered sensor modules for energy independence
4) Integration with government monitoring APIs for data validation
5) Mobile application for citizen-based pollution reporting
6) Expansion to other environmental parameters (temperature, humidity, noise)

The proposed system demonstrates that mobile IoT platforms can e ectively democratize environmental monitoring, making real-time air quality data accessible for urban planning, public health management, and citizen awareness.

REFERENCES

[1] World Health Organization, "Air Pollution," WHO Fact Sheets, 2024. [Online]. Available: h ttps://www.who.int/health-topics/air-pollution
[2] Central Pollution Control Board, "National Air Quality Monitoring Programme," CPCB India, 2024.
[3] A. Kumar and G. P. Hancke, "Energy E cient Environment Monitoring System Based on the IEEE 802.15.4 Standard for Low Cost Requirements," IEEE Sensors Journal, vol. 14, no. 8, pp. 2557-2566, Aug. 2014.
[4] R. Jain and S. Nagla, "IoT Based Real Time Air Quality Monitoring Using MQ-135 Sensor with Arduino," International Journal of Engineering Research & Technology, vol. 9, no. 5, pp. 456-462, May 2020.
[5] S. Kularatna and B. H. Sudantha, "An Environmental Air Pollution Monitoring System Based on the IEEE 1451 Standard for Low Cost Requirements," IEEE Sensors Journal, vol. 8, no. 4, pp. 415-422, Apr. 2008.
[6] M. Saad et al., "IoT Based Air Quality Monitoring System Using MQ135 and MQ7 with Machine Learning for Smart Cities," Sustainability, vol. 13, no. 17, p. 9441, 2021.
[7] Hanwei Electronics, "MQ-135 Gas Sensor Datasheet," 2023.
[8] S. De Vito et al., "Calibrating Chemical Multisensory Devices for Real World Applications: An In-Depth Comparison of Quantitative Machine Learning Approaches," Sensors and Actuators B: Chemical, vol. 255, pp. 1191-1210, Feb. 2018.
[9] N. Castell et al., "Can Commercial Low-Cost Sensor Platforms Contribute to Air Quality Monitoring and Exposure Estimates?" Environment International, vol. 99, pp. 293-302, Feb. 2017.
[10] A. Kumar, H. Kim, and G. P. Hancke, "Environmental Monitoring Systems: A Review," IEEE Sensors Journal, vol. 13, no. 4, pp. 1329-1339, Apr. 2013.
[11] J. Dutta et al., "Real-Time Air Quality Monitoring Through Mobile Sensing in Metropolitan Areas," IEEE Transactions on Mobile Computing, vol. 16, no. 8, pp. 2195-2209, Aug. 2017.
[12] Django Software Foundation, "Django Documentation Release 5.0," 2024. [Online]. Available: https://docs.djangoproject.com/
[13] MongoDB Inc., "MongoDB Manual," Version 7.0, 2024. [Online]. Available: https://docs.mongodb.com/manual/
[14] Environmental Protection Agency, "Technical Assistance Document for the Reporting of Daily Air Quality – the Air Quality Index (AQI)," EPA-454/B-18-007, Sept. 2018.
[15] R. B. Cervero, "Public Transport and Sustainable Urbanism: Global Lessons," Transport and Sustainability, vol. 7, pp. 23-41, 2015.
[16] Central Pollution Control Board, "National Ambient Air Quality Standards," CPCB Noti cation, 2019.



10.22214/IJRASET

45.98

IMPACT FACTOR: 7.129

IMPACT FACTOR: 7.429

INTERNATIONAL JOURNAL FOR RESEARCH

IN APPLIED SCIENCE & ENGINEERING TECHNOLOGY

Call : 08813907089 (24*7 Support on Whatsapp)