2026 Volume: 02 Issue: 05 www.ijrpa.com ISSN 2456-9995 Review Article

International Journal Research Publication Analysis

Page: 01-11

AIR POLLUTION MONITORING SYSTEM USING MQTT, FIREBASE AND LORAWAN- BASED SIMULATION FRAMEWORK

***Vaibhav P. Bhosale, Pratik P. Adamuthe**

India.

Article Received: 09April 2026
Article Revised: 29 April 2026
Published on: 19 May 2026

***Corresponding Author: Vaibhav P. Bhosale**
India.

DOI: https://doi-doi.org/101555/ijrpa.6525

ABSTRACT

Air pollution is a huge issue these days, especially with all the factories popping up and cities getting bigger, plus more cars on the road all the time. It leads to stuff like breathing problems, cancer in the lungs, even heart issues from just breathing bad air day after day. And the environment takes a hit too, like everything gets messed up around us. I think thats why we need better ways to track it.

Traditional setups for monitoring pollution, you know those stations, they cost a ton and stay in one spot. They dont really give you up to date info on whats happening right where you are. So that makes it hard to respond quickly or get detailed local data.

This project I worked on is about a smart city system for watching air pollution. Its all simulated using IoT ideas, built with MQTT for communication, Firebase to store stuff in the cloud, and Python to run things. Oh, and it draws from LoRaWAN for the wireless part, but without any real hardware involved. The whole thing is just a model that acts like its collecting data in real time, no actual sensors lets you test ideas without spending money on equipment.

Data comes in from these virtual points around a fake city, things like AQI levels, PM2.5 particles, and PM10 too. It gets sent through the MQTT setup to a central spot in the cloud. From there, its saved in Firebase realtime database. Then for seeing it all, theres a dashboard made with Streamlit. It has graphs that update live, alerts when pollution spikes, ways to compare different city areas, and you can access it from anywhere online.

The system feels scalable I guess, and way cheaper than real ones. Its good for real time checks in smart cities, or just analyzing environmental data, even for research on IoT. Some parts might need tweaking for actual use, but overall it works for what its meant to do. Pollution data keeps flowing in those virtual nodes, repeating the process over and over to simulate ongoing monitoring.

INTRODUCTION

Air pollution monitoring is getting a lot of attention these days because of all the worries about the environment and how we need better setups for smart cities. I think its pretty clear that with more people living in cities, tracking pollution is key for keeping things sustainable. A bunch of researchers have come up with ways to do this using wireless stuff, like IoT devices, cloud computing, and sensor networks. Most of what they focus on is real time checks, saving power in communication, analyzing data, and managing things through the cloud.

One example is what Kennedy Okokpujie did with a system based on wireless sensor networks for urban areas. It collects data on air quality from sensors spread out and sends it to a main station. That setup makes monitoring more efficient and cuts down on people having to collect data by hand, which is a hassle. But it needs actual hardware put in place, so costs go up and maintenance gets complicated.

Then theres Agustin Candias work on using LoRaWAN for smart city monitoring. Its all about long range communication that doesnt use much power. The nodes talk to cloud servers in a way thats scalable and energy saving. LoRaWAN seems good for big areas because it covers distance without a ton of setup costs. Still, the study mostly stuck to how communication works, not so much on deeper analytics or showing data in real time visuals.

Kavi K. Khedos system is another one using wireless sensors to measure pollutants right away. It handles data gathering, keeps energy low, and keeps watching pollution levels. This kind of thing helps manage pollution better in cities, I suppose. But it doesnt really connect well to clouds or have web tools for looking at data from afar, which would be useful for smart cities.

Daudi S. Simbeye looked at industrial emissions with wireless tech too. The focus was on safety, protecting the environment, and checking hazardous gases remotely. Real time

monitoring like that can lower risks from factories. It works okay for watching things live, but relies a lot on physical parts and skips over simulations or cloud based analysis.

Looking at all these, it feels like most systems are heavy on hardware. You have to deploy sensors, maintain them, set up infrastructure, and that gets expensive.
They do communication and getting data fine, but miss out on cloud analytics that scale, dashboards you can see in real time, or cheap ways to simulate for research. Some people might argue thats not a big deal, but it limits how useful they are.

The idea Im working on for a smart city air pollution system tries to fix that. It uses a simulation based IoT setup with MQTT for messaging, Firebase for the database, Python to run things, Streamlit for visuals, and something inspired by LoRaWAN. No need for real sensors at first, which is nice. It handles real time monitoring, lets you access from the cloud, scales to multiple cities, shows graphs, and fits into smart city plans. That part gets a bit messy to explain fully, but it seems promising without the hardware headache.

1. LITERATURE REVIEW

A bunch of researchers out there have come up with ways to monitor air pollution using stuff like wireless sensor networks, IoT devices, and cloud tech for keeping an eye on the environment in real time. It seems like they are trying to make data collection better in different settings. For example, Kennedy Okokpujie worked on a system with distributed sensors in a WSN setup that helps gather environmental info more effectively. Then theres Agustin Candia who built something for smart cities using LoRaWAN, which is good for long distance and low power, so it fits large areas without too much hassle.

Kavi K. Khedo did a framework with wireless sensors aimed at analyzing air quality right away, and Daudi S. Simbeye looked at industrial spots using wireless comms for pollution tracking. Most of these existing setups though, they need pricey hardware and a lot of complicated setup to get going. Many just handle collecting data and sending it around, but they do not really connect well to scalable clouds or do real time analysis, and visualization is often missing or not interactive enough.

The proposed system tries to fix that by going with a simulation based approach for smart city air pollution monitoring. It uses MQTT for the protocol, Firebase as the cloud database, Python to run things, and Streamlit for the dashboard to show visuals. I think this makes it

more straightforward without all the hardware costs. Real time monitoring happens through the cloud, and it feels like it covers the gaps in a way that is practical for testing ideas out. Some parts might still need tweaking, but overall it provides the monitoring and cloud integration that others lack.

2. System Architecture

The system they proposed for monitoring air pollution in smart cities uses this simulation setup with IoT stuff. It pulls together MQTT for talking between parts, Firebase for storing data in the cloud, some Python to handle the processing, and a Streamlit dashboard to show everything visually. They even borrowed ideas from LoRaWAN for how the communication works. Overall, it tries to mimic a real smart city setup where you can watch pollution levels live, save the data online, and look at charts to make sense of it I think the whole thing breaks down into these five main parts, like layers in a stack. Starting with the virtual sensor nodes, which are just Python scripts pretending to be sensors in places like Pune, Mumbai, or Delhi. No real hardware here, which makes it cheaper to test on a big scale. These nodes spit out fake data for things like AQI, PM2.5, PM10, temperature, humidity. The values shift around over time to act like real changes in the air. Each one publishes its info regularly, like independent little devices.

Then there's the MQTT layer for getting that data from nodes to the gateway. MQTT is lightweight, good for IoT because it does not need much power or bandwidth. The nodes publish to topics like air/pune, and the broker, which is Mosquitto, passes it along to subscribers. The gateway subscribes and pulls in the messages. It feels inspired by LoRaWAN, where lots of sensors talk to one central point before sending to the cloud.

The gateway itself is another Python piece that grabs those MQTT messages, decodes the JSON, checks if the data makes sense, adds timestamps, handles any errors, and pushes it to Firebase. It is like the middle guy between the sensors and the cloud, making sure everything is structured right for storage. Without this, the data would just be messy packets floating around.

Firebase handles the cloud storage, syncing everything in real time. You can access it from anywhere with internet, store history, scale up if needed. It is set up for JSON, which fits the pollution info nicely. This turns the simulation into something more like a full cloud platform, maybe even ready for apps or AI later on. Some people might think Firebase is too

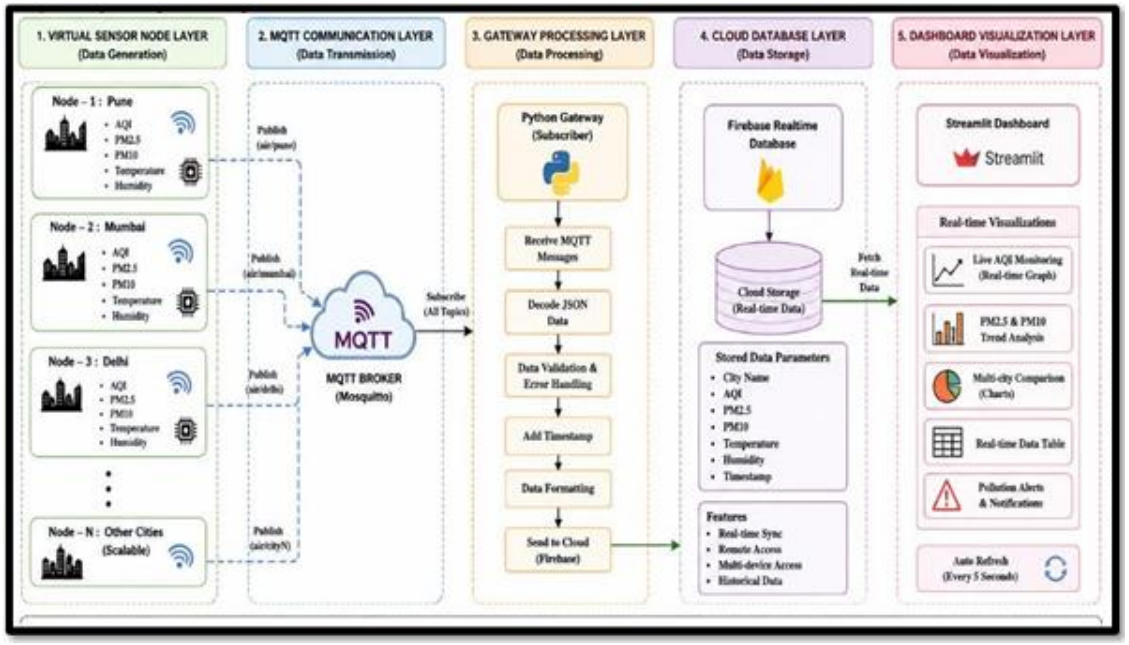
simple for big data, but it works here.

For showing the data, they use Streamlit to build a web dashboard. It pulls live from Firebase and makes graphs with Plotly, like lines for trends in PM2.5 or bars comparing cities. There is a table for raw numbers, alerts if pollution spikes, and it refreshes on its own. You get multi-city views, which seems useful for seeing patterns across places. The interface is pretty straightforward, not too fancy.

How it all flows together is nodes generate data, send via MQTT to the broker, gateway processes and timestamps it, off to Firebase for saving, then dashboard grabs it for visuals. Straightforward, but simulating the whole chain like this avoids buying sensors.

One thing that stands out is how scalable it is, with multiple cities and real-time stuff. MQTT keeps communication light, and the cloud makes it accessible. Plus, since it is simulation-based, costs are low, and you can test big without hassle.

3. Block Diagram



4. WORKING METHODOLOGY

The system they're talking about is this smart city thing for monitoring air pollution, but it's all simulated with IoT stuff instead of real hardware. It uses Python to run everything, along with MQTT for talking between parts, Firebase to store data in the cloud, and Streamlit to show it on a dashboard. Basically, it generates fake pollution data and makes it look like it's

coming from actual sensors in cities, then processes and displays it live. I think that's the main point, to show how it could work without building the whole physical setup.
So, it starts by creating these virtual sensors with Python scripts. There are
nodes for places like Pune, Mumbai, Delhi, each pretending to be a monitoring station. They spit out numbers for AQI, PM2.5, PM10, temperature, humidity, all randomized to change over time, like real weather shifts in a city. Each one acts on its own, sending data as if it's a little IoT gadget connected to a network. It feels a bit artificial, but sort of useful for testing ideas.

Once the data made, these nodes use MQTT to publish it. MQTT's this light protocol for IoT because it doesn't need much bandwidth and handles lots of devices at once. Each city node pushes info to topics like air/pune or whatever, all in JSON format since that's easy and structured for this kind of thing. The Mosquitto broker sits in the middle, grabbing messages from the nodes and passing them along to whoever's listening. That way, you can add more cities without messing up the whole system, which seems scalable.

The gateway part is another Python thing that subscribes to those topics and waits for data to come in. When it gets a JSON packet, it pulls out the values, checks if they're okay, adds a timestamp for tracking over time. Then it sends the cleaned-up data to Firebase in the cloud. Firebase keeps everything updating in real time, so you can pull historical stuff from anywhere with internet. It's handy for not losing data and letting multiple devices see it.

On the visualization side, there's this Streamlit dashboard that grabs the live data from Firebase and turns it into charts. Using Plotly, it makes graphs for trends in AQI, PM levels, comparisons between cities, even alerts if pollution gets bad. The whole thing refreshes automatically, so it's always current. You just open a browser and there it is, no matter where you are.

Overall, putting MQTT, cloud storage, and the dashboard together like this makes for a cheap way to simulate smart city monitoring. It could be good for research or planning real systems later on. Some parts might overlap a bit in how they handle data, but I am not totally sure if that's an issue. The simulation skips real sensors, which keeps it simple, though it might miss some actual complications.

5. Technologies Used

[The whole system for monitoring air pollution in a smart city pulls together a bunch of modern tech like IoT stuff, cloud computing, real-time talking between parts, and ways to show data visually. It all connects to keep track of the environment right as it happens, store things in the cloud, and make graphs that help understand whats going on.

Python ends up being the main language I used for building everything out. Its pretty straightforward, which made it easier to get going, and theres tons of libraries that fit right in with IoT and cloud work. You can develop fast with it too. So yeah, Python handles generating fake pollution data from virtual sensors, talking via MQTT, processing at the gateway, linking up with Firebase, making the dashboard, and even the analytics plus visualizations.

For the communication side, MQTT protocol is what links the virtual sensor nodes to the gateway. Its light weight, meant for IoT setups, and works on a publish subscribe model. That means low bandwidth use, quick data transfer in real time, and it handles multiple devices without much hassle. The architecture scales well for more nodes. In this setup, the city nodes publish their pollution info to specific topics, and the gateway subscribes to grab it all live.

Then theres the Mosquitto broker from Eclipse, which sits in the middle as the server for all that MQTT traffic. It takes messages from the publishers, like the virtual nodes sending pollution data, and routes them over to subscribers, which is the gateway here. It manages topics, routes messages smoothly, keeps real time flow going, and connects everything multi node style. Without it, the whole thing would fall apart I think.

Storing the data happens in Firebase, the realtime database on the cloud. It saves the pollution numbers as they come in, and you can access it from anywhere with internet. Synchronization is instant, storage is all cloud based, it keeps historical stuff too, lets multiple devices pull from it, and supports monitoring remotely. That part feels solid for a system like this.

The dashboard comes together with Streamlit, which makes a web interface that pulls live data straight from Firebase. It shows graphs and analytics for monitoring. Features include watching AQI live, real time graphs, breaking down PM2.5 and PM10 levels, charts comparing across cities, and it refreshes on its own. Some of that gets a bit messy to set up, but it works.

For the actual visuals, Plotly library steps in to create interactive graphs and charts. Its good for professional looking environmental analytics. So it does line graphs, bar charts, trends for AQI over time, and comparisons between multiple cities. That makes the data easier to grasp, kind of.

Pandas helps with processing the data, turning it into tables and handling environmental info before visualizing. You create dataframes, filter out what you need, sort it, process timestamps. Its straightforward for organizing pollution data.

The communication draws from LoRaWAN ideas, even though this is just a simulation. LoRaWAN has that long range thing, multi node setup, low power use, and scales for smart cities. In the future, you could hook it up to real hardware like that. It seems like a good fit conceptually.

Data moves around in JSON format between the nodes, broker, gateway, and Firebase. JSON is light and structured, perfect for IoT exchanges. It keeps things simple without extra bulk.

6. Advantages

1. **Real-Time Monitoring**

   The system continuously monitors pollution parameters such as AQI, PM2.5, and PM10 in real time. Live environmental data helps users observe changing pollution conditions instantly without delay.

2. **Cloud-Based Accessibility**

   The use of Firebase cloud database allows pollution data to be accessed remotely from any internet-connected device. Users can monitor environmental conditions from different locations through the web dashboard.

3. **Cost-Effective System**

   Since the proposed model is simulation- based, it does not require expensive physical sensors or hardware infrastructure during implementation. This reduces development and maintenance cost significantly.

4. **Scalable Architecture**

   The system supports multi-city and multi- node architecture. Additional virtual nodes or real hardware sensors can be integrated easily without major modifications to the existing framework.

5. Lightweight MQTT Communication

MQTT protocol provides efficient low- bandwidth communication between nodes and gateway systems. This improves communication speed and reduces network overhead in IoT applications.

7. LIMITATIONS

1. Simulation-Based System

The current project is fully simulation-based and does not use physical environmental sensors. Therefore, the pollution values are generated virtually and may not exactly represent real-world environmental conditions.

2. Dependence on Internet Connectivity

The system depends heavily on internet connectivity for MQTT communication, Firebase cloud storage, and dashboard visualization. Poor internet connection may affect real-time monitoring performance.

3. Limited Security Features

The prototype system currently has basic cloud connectivity and does not include advanced security mechanisms such as encrypted communication, authentication layers, or cybersecurity protection for IoT devices.

4. Cloud Service Dependency

The system depends on Firebase cloud services for data storage and synchronization. Any cloud server downtime or service interruption may temporarily affect data accessibility and dashboard operation.

8. APPLICATIONS

1. Smart City Monitoring

The system can be used in smart cities for continuous monitoring of environmental pollution levels and real-time air quality analysis across different urban regions.

2. Industrial Pollution Monitoring

Industries can use the system to monitor pollution generated from factories, manufacturing units, and industrial plants to ensure environmental safety and regulatory compliance.

3. Traffic Pollution Analysis

The system can help analyze pollution levels in high-traffic areas, highways, and urban roads where vehicle emissions significantly affect air quality.

4. **Environmental Research**

   Researchers and academic institutions can use the system for environmental studies, IoT research, pollution analytics, and smart- city technology development.

5. **Government Pollution Control**

   Government environmental agencies can use the system for city-wide pollution monitoring, environmental management, and public safety analysis.

9. FUTURE SCOPE

1. **Integration with Real Sensors** In future implementations, the system can be integrated with real air quality sensors such as MQ135, PMS5003, DHT11, and gas sensors for collecting actual environmental pollution data.
2. **Lorawan Hardware Deployment** The current communication model is inspired by LoRaWAN concepts. Future work can include real LoRaWAN gateways and sensor nodes for long-range low-power environmental monitoring in smart cities.
3. **AI-Based Pollution Prediction** Machine learning and artificial intelligence algorithms can be integrated to predict future AQI levels, pollution trends, and environmental risks based on historical pollution data.
4. **GIS and Heatmap Visualization** The system can be upgraded with GIS mapping and pollution heatmap visualization to display geographical pollution distribution across different city regions.
5. **Mobile Application Development** A mobile application can be developed for Android and iOS platforms to provide live pollution monitoring and alert notifications directly on smartphones.

10. CONCLUSION

This system I worked on for monitoring air pollution in a smart city setup seems to work well for tracking things in real time. It uses stuff like the MQTT protocol to send data around, and then Firebase for storing it all in the cloud. Python handles the main programming part, and Streamlit makes this dashboard that shows graphs and stuff live. What it does is keep an eye on pollution levels, like AQI and those PM2.5 and PM10 particles, coming from these simulated spots in a virtual city. You can see the updates right away on the dashboard, which I think helps visualize how bad the air is getting.

One thing that stands out is how it does not need actual sensors or hardware to run, at least not for this version. That makes it cheaper and easier to scale up for a real smart city, I guess.

The data flows in real time, gets saved, and then you have these charts to look at for analyzing pollution. It feels efficient for that kind of thing, without all the hassle of setting up physical equipment.

For improvements, maybe adding real sensors could make it more practical down the line. Things like LoRaWAN for longer range, or even AI to predict pollution trends. A mobile app might let people check it on their phones too, and fancier cloud tools for deeper analysis. I am not totally sure how all that would fit together yet, but it could lead to better smart infrastructure eventually.

11. REFERENCES

1. MQTT Official Website
2. Firebase Official Website
3. Streamlit Official Website
4. Plotly Python Graphing Library
5. Python Official Website
6. Eclipse Mosquitto MQTT Broker
7. Kennedy Okokpujie, “A Smart Air Pollution Monitoring System,” International Journal of Scientific & Engineering Research, vol. 8, no. 9, pp. 799–809, 2017.