

Real-Time Air Quality Monitoring and Pollution-Aware Route Optimization Using Vehicle-Mounted IoT Sensors

Aarav Mody

Dept. of Computer Engineering

Dwarkadas J. Sanghvi College of Engg.

Mumbai, India

aarav.mody041

@svkmmumbai.onmicrosoft.com

Aarav Lodha

Dept. of Computer Engineering

Dwarkadas J. Sanghvi College of Engg.

Mumbai, India

aarav.lodha147

@svkmmumbai.onmicrosoft.com

Aryan Shah

Dept. of Computer Engineering

Dwarkadas J. Sanghvi College of Engg.

Mumbai, India

aryan.shah129

@svkmmumbai.onmicrosoft.com

Devraj Jain

Dept. of Computer Engineering

Dwarkadas J. Sanghvi College of Engg.

Mumbai, India

devraj.jain227

@svkmmumbai.onmicrosoft.com

Dr. Kranti Ghag

Dept. of Computer Engineering

Dwarkadas J. Sanghvi College of Engg.

Mumbai, India

kranti.ghag@djsce.ac.in

Dr. Meera Narvekar

Dept. of Computer Engineering

Dwarkadas J. Sanghvi College of Engg.

Mumbai, India

meera.narvekar@djsce.ac.in

Abstract—Urban air pollution represents a critical public health crisis affecting billions of people worldwide, with the World Health Organization attributing approximately 4.2 million premature deaths annually to ambient air pollution exposure. Traditional air quality monitoring infrastructure relies on sparse networks of expensive stationary monitoring stations that fail to capture the fine-grained, route-specific pollution variations experienced by urban commuters. This paper presents the design, implementation and the experimental evaluation of the “Green Corridor” which is a comprehensive real-time air quality monitoring and pollution-aware route optimization system utilizing vehicle-mounted Internet of Things (IoT) sensors. The proposed system integrates an ESP32-based hardware sensing platform equipped with GP2Y1010AU0F optical particulate matter sensors, MQ-135 multi-gas sensors, DHT11 temperature-humidity sensors and NEO-6M GPS modules for continuous environmental data acquisition during vehicle travel. The backend architecture employs a containerized microservices approach orchestrated through Docker Compose, featuring a high-throughput Go-based ingestion service achieving sub-50ms latency using the Fiber framework, a Python FastAPI routing service implementing weighted A* pathfinding with OSMnx graph operations, and Uber’s H3 hexagonal hierarchical spatial indexing system (Resolution 9, approximately 0.1 km² per cell) for efficient geospatial data management. A novel “One Vehicle, One Vote” debouncing mechanism implemented through Redis Hash structures with time-to-live expiration prevents stationary vehicles from biasing aggregated pollution statistics. The routing algorithm implements a parameterized cost function $Cost(e) = T(e)(1 - \beta) + AQI(e)\alpha\beta$, enabling users to dynamically balance travel time against pollution exposure through a continuous balance factor $\beta \in [0, 1]$. Experimental evaluation conducted over a two-month deployment period in Mumbai, India with three instrumented vehicles demonstrates the system’s capability to identify pollution hotspots with sub-second data processing latency, suggest cleaner alternative routes achieving 21-26% average AQI reduction with less than 13% travel time increase, and maintain reliable

operation under varying network conditions. The frontend provides an interactive React-based web interface with Leaflet.js map visualization featuring hexagonal pollution heatmaps, color-coded route comparisons, and quantitative exposure metrics. This work contributes a practical, scalable, and cost-effective solution for citizen-led participatory air quality monitoring and health-conscious navigation aligned with smart city sustainability objectives.

Index Terms—Air Quality Index, Vehicle-Mounted IoT Sensors, Pollution-Aware Routing, H3 Spatial Indexing, Weighted A*, Microservices.

I. INTRODUCTION

A. Background and Motivation

Air pollution is one of the most serious environmental and public-health challenges in the world. According to the World Health Organization (WHO), about 99% of the global population breathes air which is below the recommended quality standards and outdoor air pollution contributes to at least 4.2 million deaths every year. Impacts also include respiratory and cardiovascular disease, cognitive decline, reduced life expectancy, and major economic loss [2].

Cities are especially affected because the transport, industry construction and the domestic activities are concentrated. In many countries, PM_{2.5} levels frequently exceed the WHO’s limits by 5-10x [3]. The main pollutants of concern are:

- Particulate Matter (PM_{2.5}/PM₁₀)
- Nitrogen Dioxide (NO₂)
- Carbon Monoxide (CO)
- Sulfur Dioxide (SO₂)
- Ground-level Ozone (O₃)

B. Limitations of Traditional Monitoring

Traditional air-quality monitoring depends on the fixed stations which are operated by the government. Although these systems are accurate, they face certain limitations:

High Deployment Costs: Air quality monitoring stations typically cost about INR 1,50,000–6,00,000 to install and INR 50,000–1,20,000 annually for the maintenance, which limits the density of the network [4].

Limited Spatial Coverage: Many cities have only 5-20 stations which are separated by kilometers, thereby missing the neighborhood-level variation that can differ by 50-200% over a very short distance [5].

Area-Level Reporting: AQI is generally reported at the city scale and not at a street or a route scale which limits the commuter-level guidance.

Temporal Lag: Validating and processing the pipelines can also delay the updates by several hours.

C. Opportunities in Mobile Sensing

Low-cost sensors and IoT enable the distributed mobile monitoring. Vehicle-mounted platforms can transform taxis and personal vehicles into moving sensing nodes that can continuously sample the street-level pollution [6].

When compared with the stationary monitoring, mobile sensing provides:

- **Increased Spatial Coverage:** Instrumented fleets that can cover hundreds of kilometers every day.
- **Route-Specific Data:** Measurements that align with the commuters path.
- **Cost Efficiency:** Low cost per node (40\$-50\$) enables scaling.
- **Adaptive Sampling:** Coverage follows traffic flow and activity.

D. Research Questions and Objectives

This paper presents Green Corridor which is a real-time air-quality monitoring and pollution-aware route optimization system. The work addresses:

RQ1: How can low-cost vehicle sensors provide reliability and route-level air-quality data despite the sensor constraints?

RQ2: How can software architecture process high-velocity mobile telemetry at sub-second latency for routing?

RQ3: How should the pollution-aware routing trade travel time, distance, and exposure based on the users preference?

RQ4: Which mechanisms prevent the mobile-sensing quality issues such as the stationary-vehicle bias and the disconnections?

E. Contributions

The main contributions are:

- 1) An end-to-end system spanning the sensing hardware, the backend processing and an interactive pollution-aware navigation.
- 2) A Redis-based “One Vehicle, One Vote” debouncing mechanism which helps in preventing the stationary-vehicle bias.

- 3) H3 geospatial indexing (resolution 9) for efficient and near-constant-time spatial lookup.
- 4) A weighted A* routing algorithm with user-adjustable β to trade the travel time against the pollution exposure.
- 5) A containerized microservices architecture based on Docker Compose with horizontal scaling and deployment to the varied urban settings.
- 6) Comprehensive evaluation showing 21–26% AQI reduction with less than 13% travel-time increase.

II. RELATED WORK

Prior work validates three building blocks: low-cost sensing, mobile monitoring and multiple objective routing. IoT monitoring studies show good feasibility but are fixed deployments [7], [9]. Vehicle-based systems improve the spatial coverage and expose the strong intra-city variability [6], [10]. Pollution-aware routing methods demonstrate the measurable exposure reduction, but often depend on data of the sparse stations rather than the route level data [11], [13].

For low-cost sensing, calibration is very important. A commonly used model is:

$$PM_{calibrated} = \alpha \cdot PM_{raw} + \beta \cdot T + \gamma \cdot RH + \delta \quad (1)$$

where temperature, relative humidity and the coefficients are empirically estimated [8].

Spatial indexing choices strongly affect the query speed and the aggregation quality. H3 provides the uniform cells, efficient neighborhood operations and a practical resolution control for route-scale analysis [14]. The main gap is not the isolated components but the full integration into a real-time mobile sensing and routing pipeline with a robust bias control.

III. SYSTEM ARCHITECTURE

Green Corridor uses a distributed microservices architecture with four layers: the sensing, the communication, the backend processing and the user application. Fig. 1 shows the high-level topology.

A. Hardware Architecture

The hardware subsystem consists of a vehicle mounted with IoT nodes for continuous environmental monitoring.

1) Microcontroller Platform: The node uses an ESP32-WROOM-32 because it balances the compute capability, the wireless connectivity and the cost. Key specifications include:

- Dual-core Xtensa LX6 processor at 240 MHz
- 520 KB SRAM, 4 MB flash storage
- Integrated 802.11 b/g/n WiFi (2.4 GHz)
- 12-bit SAR ADC with 18 channels
- Low-power operation for vehicle auxiliary supply

The ESP32 supports sensor polling, local aggregation, JSON serialization and HTTP communication.

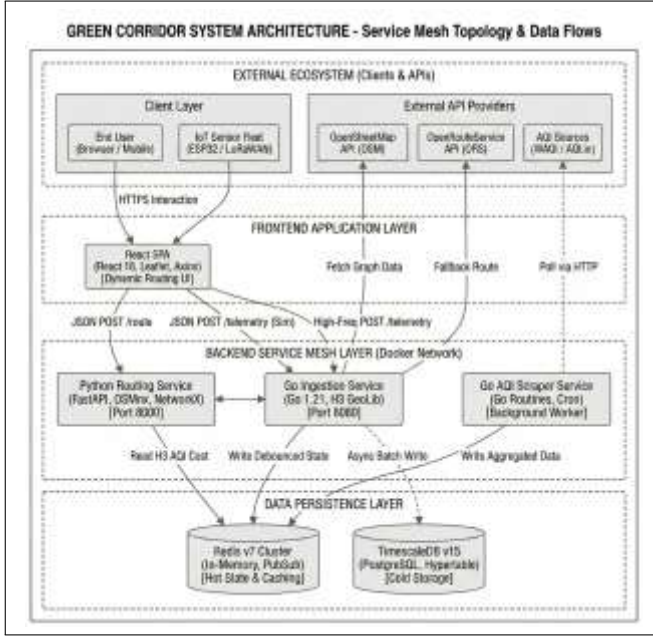


Fig. 1: High-level system architecture showing service mesh topology, data flow paths, and technology stack for each component.

TABLE I: Sensor Component Specifications

Component	Model	Key Specifications
PM Sensor	GP2Y1010AU0F	Detection: 0.8-10 μm Range: 0-500 $\mu\text{g}/\text{m}^3$ Response: 280 μs
Gas Sensor	MQ-135	Gases: NH_3 , NO_x , CO_2 , C_6H_6 Range: 10-1000 ppm Preheat: 24-48 hrs
Temp/Humidity	DHT11	Temp: 0-50°C ($\pm 2^\circ\text{C}$) RH: 20-90% ($\pm 5\%$) Sampling: 1 Hz
GPS Module	NEO-6M	Accuracy: 2.5m CEP Update: 5 Hz Protocol: NMEA 0183

2) *Sensor Array*: Table I summarizes the sensors integrated in each of the node.

Particulate Matter Sensing: The PM sensor uses an infrared light scattering (IR LED + phototransistor at 90° in a shielded chamber). During a 10 ms LED pulse, the ADC readings capture scattered intensity proportional to the particle concentration. Raw voltage V (0–3.3 V) is converted to $\text{PM}_{2.5}$ using:

$$\text{PM}_{2.5} = 170 \times V - 100 \quad [\mu\text{g}/\text{m}^3] \quad (2)$$

Values are clamped between 0-500 $\mu\text{g}/\text{m}^3$.

Gas Sensing: The MQ-135 detects NH_3 , NO_x , CO_2 , C_6H_6 and alcohol vapors. Its tin-dioxide layer changes the resistance with the gas concentration:

$$R_s/R_0 = a \times [\text{Gas}]^b \quad (3)$$

where R_s is sensor resistance in target gas, R_0 is baseline resistance in clean air, and a, b are calibration constants.

Environmental Context: DHT11 provides temperature and humidity for the drift compensation under the changing weather.

Location Tracking: NEO-6M provides GPS via the UART (9600 baud). Firmware parses NMEA 0183 (\$GPGGA, \$GPRMC) to extract the latitude, the longitude, the altitude, the speed and the fix quality.

3) *Firmware Architecture*: Firmware (Arduino IDE with ESP-IDF libraries) follows a cyclic executive schedule:

Algorithm 1 Sensor Node Main Loop

```

1: Initialize WiFi connection with configured SSID
2: Initialize HTTP server on port 80
3: Initialize sensor GPIO and ADC channels
4: loop
5:    $\text{pm}_{25} \leftarrow \text{ReadPMSensor}()$ 
6:    $\text{gas} \leftarrow \text{ReadGasSensor}()$ 
7:    $(\text{temp}, \text{humidity}) \leftarrow \text{ReadDHT11}()$ 
8:    $(\text{lat}, \text{lng}, \text{alt}, \text{speed}) \leftarrow \text{ParseGPSNMEA}()$ 
9:    $\text{aqi} \leftarrow \text{CalculateAQI}(\text{pm}_{25}, \text{gas}, \text{temp}, \text{humidity})$ 
10:   $\text{UpdateJSONPayload}(\text{aqi}, \text{lat}, \text{lng}, \text{temp}, \text{humidity})$ 
11:   $\text{HandleHTTPRequest}()$ 
12:   $\text{delay}(\text{SAMPLE\_INTERVAL})$ 
13: end loop

```

B. Backend Architecture

The backend runs as containerized services in Docker Compose on a dedicated bridge network (green-corridor-network) with DNS-based service discovery.

1) *Ingestion Service*: The ingestion service which is implemented in Go, use a “Fire and Forget” pattern for low latency plus “One Vehicle, One Vote” debouncing for the data quality.

The request-processing pipeline in `telemetry.go` has four stages: **Stage 1 - Request Validation**: Incoming JSON payloads undergo the strict validation:

- Latitude: $-90 \leq \text{lat} \leq 90$
- Longitude: $-180 \leq \text{lng} \leq 180$
- AQI: $0 \leq \text{aqi} \leq 500$, Vehicle ID: non-empty

Invalid payloads return HTTP 400 with explicit errors.

Stage 2 - Geospatial Indexing: Coordinates are converted to the H3 resolution-9 indices using the `h3-go`.

Each hexagon covers about 0.1 km^2 (edge length ~ 174 m) which is suitable for the neighborhood-scale routing analysis.

Stage 3 - Data Debouncing: The “One Vehicle, One Vote” prevents the stationary vehicles from biasing the hex-level AQI through the repeated uploads.

Each vehicle contributes one active reading per hexagon; newer values overwrite the older ones. A 5-minute TTL removes the stale records after the vehicles move away.

Stage 4 - Asynchronous Persistence: Telemetry is written to the TimescaleDB without blocking any HTTP responses in return.

2) *Routing Service*: The FastAPI routing service performs the pollution-aware pathfinding on the road-network graphs.

Graph Engine: `graph_service.py` uses the OSMnx to fetch the drivable OpenStreetMap networks.

A 1-hour TTL cache keyed by the MD5 of the bounding-box coordinates which reduces the Overpass calls while keeping the network data fresh.

Graph Enrichment: Before routing, the edges are annotated with the pollution weights:

Algorithm 2 Graph Pollution Enrichment

```

Require: Road network graph  $G = (V, E)$ 
Require: Redis connection with AQI data
1: for all edge  $e = (u, v) \in E$  do
2:    $points \leftarrow \text{InterpolateEdge}(e, \text{interval}=50\text{m})$ 
3:    $aqiV\ values \leftarrow []$ 
4:   for all point  $p \in points$  do
5:      $hexID \leftarrow \text{H3.geoToH3}(p.lat, p.lng, 9)$ 
6:      $vehicleAQIs \leftarrow \text{Redis.HVALS}("aqi:h3:" + hexID)$ 
7:     if  $vehicleAQIs$  not empty then
8:        $aqiV\ values.append(\text{median}(vehicleAQIs))$ 
9:     end if
10:  end for
11:  if  $aqiV\ values$  not empty then
12:     $e.aqi \leftarrow \text{mean}(aqiV\ values)$ 
13:  else
14:     $e.aqi \leftarrow \text{DEFAULT\_AQI}$ 
15:  end if
16: end for
17: return Enriched graph  $G$ 

```

3) *AQI Scraper Service*: The Go-based scraper service provides the fallback pollution data for the areas without any vehicular coverage:

- **aqi.in**: Scrapes Indian city pollution data via Colly.
- **WAQI API**: Queries the geographic bounding boxes for broader coverage.

Geographic regions are fetched concurrently using the `sync.WaitGroup`.

4) *Data Persistence Layer*: The system uses a hybrid persistence optimized for the different access patterns:

Redis v7 (Hot State): In-memory store for the real-time routing decisions. Table II summarizes the key namespace.

TABLE II: Redis Key Namespace Design

Key Pattern	Type	Purpose
<code>aqi:h3:{hex}</code>	Hash	Per-vehicle AQI readings by hexagon
<code>vehicle:{id}</code>	String	Last known vehicle hexagon
<code>station:h3:{hex}</code>	String	Static station fallback data
<code>stations:all</code>	String	Bulk station list for map display
<code>stats:ingestions</code>	Counter	Ingestion count metric

TimescaleDB v15 (Cold Storage): PostgreSQL extension which is optimized for the time-series storage and the historical analytics.

IV. ROUTING ALGORITHM DESIGN

This section presents the mathematical and the algorithmic design of the routing.

A. Problem Formulation

Given a directed graph $G = (V, E)$ representing the intersections (V) and the road segments (E), the pollution-aware routing takes a path $P = (v_1, v_2, \dots, v_n)$ which minimizes the weighted combination of the travel time and the pollution exposure.

Each edge $e \in E$ has the following attributes:

- $l(e)$: Physical length in meters
- $s(e)$: Speed limit or estimated travel speed
- $T(e) = l(e)/s(e)$: Estimated traversal time
- $AQI(e)$: Aggregated air quality index from the H3 hexagons

B. Cost Function Formulation

The system takes a user-defined balance factor $\beta \in [0, 1]$ controlling the objective trade-off:

- $\beta = 0$: Minimize the travel time (fastest route)
- $\beta = 1$: Minimize the pollution exposure (cleanest route)
- $\beta = 0.5$: Equal weighting (balanced route)

The edge cost function is defined as:

$$\text{Cost}(e) = T(e) \times (1 - \beta) + AQI(e) \times \alpha \times \beta \quad (4)$$

This weighted-additive form preserves the non-negative A^* edge costs, remains interpretable, enables the continuous control through β and yields more stable rankings than the threshold alternatives [11]–[13].

The normalization constant $\alpha = 0.6$ was selected from a Mumbai sensitivity sweep (June–July 2025) over the $\{0.2, 0.4, 0.6, 0.8, 1.0\}$. Lower values lead to the under-weighted AQI, while the higher values caused excessive detours and $\alpha = 0.6$ gave the best trade-off (23.6% mean AQI reduction with 9.0% average travel-time increase).

The total path cost is the sum of the edge costs:

$$\text{Cost}(P) = \sum_{e \in P} \text{Cost}(e) \quad (5)$$

C. A^* Search Algorithm

Routing uses the A^* algorithm with an admissible heuristic based on the Haversine great-circle distance.

1) *Heuristic Function*: The heuristic estimates $h(n)$ from the node n to the goal g as the Haversine distance divided by maximum assumed speed:

$$h(n) = \frac{d_{\text{Haversine}}(n, g)}{V_{\max}} \quad (6)$$

where $V_{\max} = 50$ m/s (180 km/h) ensures that there is no overestimation and preserves the admissibility.

The Haversine equations are as follows:

$$\alpha = \sin^2 \frac{\Delta\phi}{2} + \cos(\phi_1) \cos(\phi_2) \sin^2 \frac{\Delta\lambda}{2} \quad (7)$$

$$d = 2R \cdot \arctan 2 \frac{\sqrt{a}}{a} \sqrt{\frac{1-a}{1+a}} \quad (8)$$

where ϕ represents the latitude, λ represents the longitude and $R = 6,371$ km is the Earth's mean radius.

2) *Algorithm Implementation*: Algorithm 3 shows A^* with the pollution-weighted costs.

Algorithm 3 Pollution-Aware A^* Search

Require: Graph G , source s , target t , balance β

Ensure: Optimal path P from s to t

```

1:  $openSet \leftarrow$  MinHeap with  $(f(s), s)$ 
2:  $gScore[s] \leftarrow 0$ 
3:  $fScore[s] \leftarrow h(s, t)$ 
4:  $cameFrom \leftarrow \{\}$ 
5: while  $openSet$  not empty do
6:    $current \leftarrow openSet.extractMin()$ 
7:   if  $current = t$  then
8:     return ReconstructPath( $cameFrom$ ,  $t$ )
9:   end if
10:  for all neighbor  $n$  of  $current$  do
11:     $e \leftarrow edge(current, n)$ 
12:     $tentative \leftarrow gScore[current] + Cost(e, \beta)$ 
13:    if  $tentative < gScore[n]$  then
14:       $cameFrom[n] \leftarrow current$ 
15:       $gScore[n] \leftarrow tentative$ 
16:       $fScore[n] \leftarrow tentative + h(n, t)$ 
17:       $openSet.insert((fScore[n], n))$ 
18:    end if
19:  end for
20: end while
21: return "No path found"
```

D. Alternative Route Generation

To provide the route choices with an explicit time-pollution trade-offs, we generate the alternatives using an iterative penalty method.

After computing the optimal path P_1 , the edges in the path P_1 receive a cost penalty multiplier which is given by:

$$Cost'(e) = \begin{cases} Cost(e) \times \gamma & \text{if } e \in P_1 \\ Cost(e) & \text{otherwise} \end{cases} \quad (9)$$

where $\gamma = 1.5$ (50% penalty) was empirically selected to balance the diversity and the detour length.

A^* algorithm is then re-run on the penalized graph to obtain the P_2 , and repeated for the P_3 , P_4 , ... with the cumulative penalties.

E. Complexity Analysis

A^* algorithm has a time complexity $O(E \log |V|)$ when using a binary heap. For the typical urban graphs ($|V| = 10,000$ – $100,000$), the route computation completes in about 200–800 ms which is fairly acceptable.

Graph enrichment (Algorithm 2) requires $O(E \times \bar{k})$ Redis operations, where $\bar{k}$ is the average interpolation points for each edge; caching amortizes this cost across the requests.

V. IMPLEMENTATION DETAILS

A. Service Deployment

The software stack is deployed as a Docker microservices architecture: Ingestion service in Go, FastAPI routing, Redis hot-state cache, TimescaleDB historical store and a React frontend. The Health checks and the restart policies provide a great deal of operational resilience, while the service separation enables an independent scaling of the ingestion and the routing paths.

B. API Contract Specification

Three interfaces are central to the operation: telemetry ingestion, route computation and the map retrieval. Validation checks (schema, coordinate bounds, timestamp sanity, and sensor range limits) are also applied before persistence, which reduces the noise in the downstream optimization.

C. Frontend Implementation

The frontend uses a Leaflet-based map which renders with the H3 heatmaps and also does side-by-side route comparison. Users adjust the beta through a slider and receive immediate updates in the route geometry, travel time and the expected AQI exposure on their route.

VI. EXPERIMENTAL EVALUATION

A. Experimental Setup

We tested the system in a two month deployment in Mumbai over different routes. Assessment was based on latency, quality, reliability and the quality of routes.

B. Performance and Route Quality

Throughout the deployment, the ingestion was within the range of the sub-second and the routing was interactive to the queries at a commuter scale. The exposure reduction was in continuous and constant 21-26% with less than 13% increase in the travel-time, which corresponds to the design goal of the viable health-time trade-off.

C. Balance Factor Sensitivity

Fig. 2 shows the route behaviour as the balance factor beta for a 6-km trip.

The curve exhibits the diminishing returns: moderate increases in beta results in substantial AQI improvement, while larger beta yields smaller additional gains with a higher time penalty.

D. Spatial and Temporal Analysis

The H3 resolution 9 provided a good balance between the street-level detail and the data density. The "One Vehicle, One Vote" debouncing mechanism prevented the parked vehicles from inflating the aggregate AQI values, thereby improving the hotspot stability.

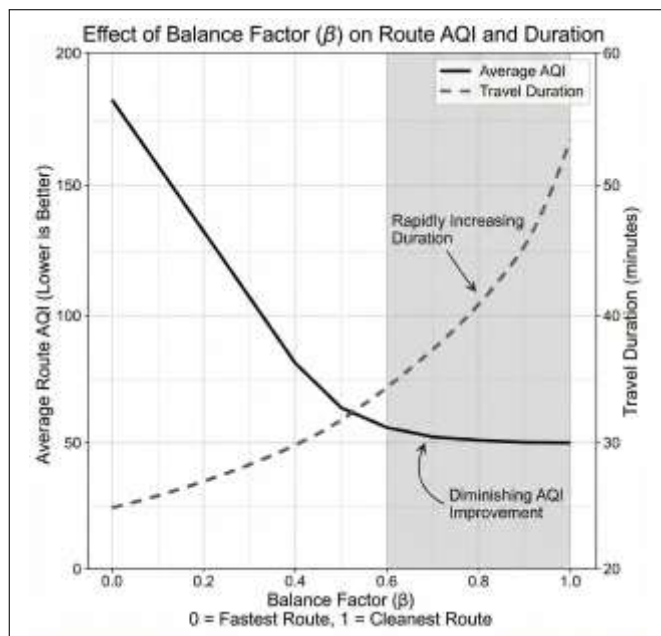


Fig. 2: Effect of the balance factor beta through the route average AQI and the travel duration. The relationship shows the diminishing AQI improvement at higher beta.

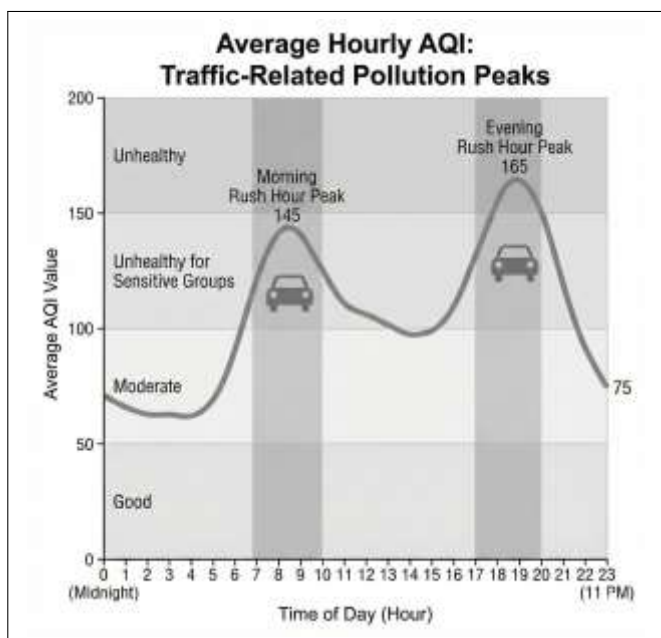


Fig. 3: Average hourly AQI values showing the morning (8-9 AM) and the evening (6-8 PM) traffic-related pollution peaks. Data source: Green Corridor Mumbai pilot telemetry (June–July 2025).

VII. DISCUSSION

The study demonstrates that low-cost mobile sensing can be coupled with practical real-time routing in a dense urban context. Compared with fixed-station-only systems, route-level measurements provide more actionable commuter guidance. Main limitations are fleet size, sensor drift, and single-city evaluation; however, the architecture is modular and supports straightforward horizontal scaling.

VIII. CONCLUSION AND FUTURE WORK

Green Corridor demonstrates that route-aware air quality guidance is feasible with low-cost, vehicle-mounted IoT sensors and a microservices backend. In Mumbai, the system consistently identified cleaner alternatives with moderate time overhead and stable real-time operation.

Future work will focus on larger fleets with adaptive calibration, traffic and weather-aware forecasting and longer deployment across various other cities.

ACKNOWLEDGMENT

The authors thank Dr. Kranti Ghag, the Department of Computer Engineering at DJSCE, and the anonymous reviewers for their guidance and feedback.

REFERENCES

- [1] World Health Organization, "Ambient air pollution: A global assessment of exposure and burden of disease," WHO Press, Geneva, 2024.
- [2] P. J. Landrigan *et al.*, "The Lancet Commission on pollution and health," *The Lancet*, vol. 391, no. 10119, pp. 462–512, 2018.
- [3] J. S. Apte, J. D. Marshall, A. J. Cohen, and M. Brauer, "Addressing global mortality from ambient PM_{2.5}," *Environ. Sci. Technol.*, vol. 49, no. 13, pp. 8057–8066, 2015.
- [4] P. Kumar *et al.*, "The rise of low-cost sensing for managing air pollution in cities," *Environment International*, vol. 75, pp. 199–205, 2015.
- [5] J. S. Apte *et al.*, "High-resolution air pollution mapping with Google Street View cars: Exploiting big data," *Environ. Sci. Technol.*, vol. 51, no. 12, pp. 6999–7008, 2017.
- [6] D. Hasenfratz *et al.*, "Deriving high-resolution urban air pollution maps using mobile sensor nodes," *Pervasive and Mobile Computing*, vol. 16, pp. 268–285, 2015.
- [7] E. Ivanova *et al.*, "Autonomous IoT air monitoring system: 13-month continuous deployment study," *Proc. Univ. of Ruse*, vol. 63, 2024.
- [8] J. Kang *et al.*, "AI-enhanced sensor calibration for humidity and temperature-aware PM_{2.5} monitoring," *Sensors*, vol. 24, no. 8, art. 2456, 2024.
- [9] M. Sinambela, D. Saluza, and R. Pratiwi, "IoT-based air quality monitoring system design using ESP32," *IEEE Access*, vol. 12, pp. 45892–45908, 2024.
- [10] A. R. Whitehill *et al.*, "Mobile vs. fixed air quality monitoring: Temporal and spatial characteristics," *Atmos. Meas. Tech.*, vol. 17, pp. 2991–3014, 2024.
- [11] B. Zou, X. Chen, and Y. Li, "Multi-objective route optimization for pollution exposure minimization," *Transportation Research Part D*, vol. 93, art. 102752, 2021.
- [12] K. K. Meena *et al.*, "Dynamic route planning for minimal air-pollution exposure," *Transp. Res. Rec.*, vol. 2679, no. 3, pp. 456–472, 2025.
- [13] Y. Wang, X. Liu, and H. Zhang, "Route planning for active travel considering air pollution and traveler characteristics," *Transp. Res. Part D*, vol. 103, art. 103456, 2024.
- [14] I. Brodsky, "H3: Uber's hexagonal hierarchical spatial index," Uber Engineering Blog, 2018.