UNIVERSITAT
DEGLI STUDI
DELL'AQUILA

Software Engineering for the Internet of Things

SMART CITIES: AIR QUALITY MONITORING

Daniela Fajardo (301128)
Fatima Mabrouk (301138)
Prachi (301124)

2024 | DECEMBER

PROF. DAVIDE DI RUSCIO

TABLE OF CONTENTS

INTRODUCTION

The IoT-Based Air Quality Monitoring System is an advanced solution designed to measure, analyze, and report environmental air quality conditions within urban areas, contributing to the development of smart cities. This system leverages cutting-edge IoT technologies to continuously monitor key environmental factors such as CO2 and NO2 levels, PM2.5, PM10, temperature, and humidity. By integrating sensor power consumption data, the system delivers actionable insights into air quality trends and urban environmental health, empowering city planners, policymakers, and citizens to make data-driven decisions.

FUNCTIONAL REQUIREMENTS

Requirements	Details	Priority
Sensor Integration	The system must integrate sensors to measure CO2, PM2.5, PM10, temperature, humidity, and power consumption data.	High
Communication Protocols	The system must use MQTT for efficient communication between sensors, middleware, and storage.	Medium
Data Processing	The system must be able to aggregate, process, and transform sensor data to perform calculations, including Air Quality Index (AQI).	High
Data Storage	The system must store sensor data for historical data and trend analysis.	High
Visualization	The system must display stored data through intuitive Grafana dashboards, including time series and charts.	Medium
Alerting Mechanisms	The system must trigger alerts based on threshold violations and send notifications via Telegram.	High

NON-FUNCTIONAL REQUIREMENTS

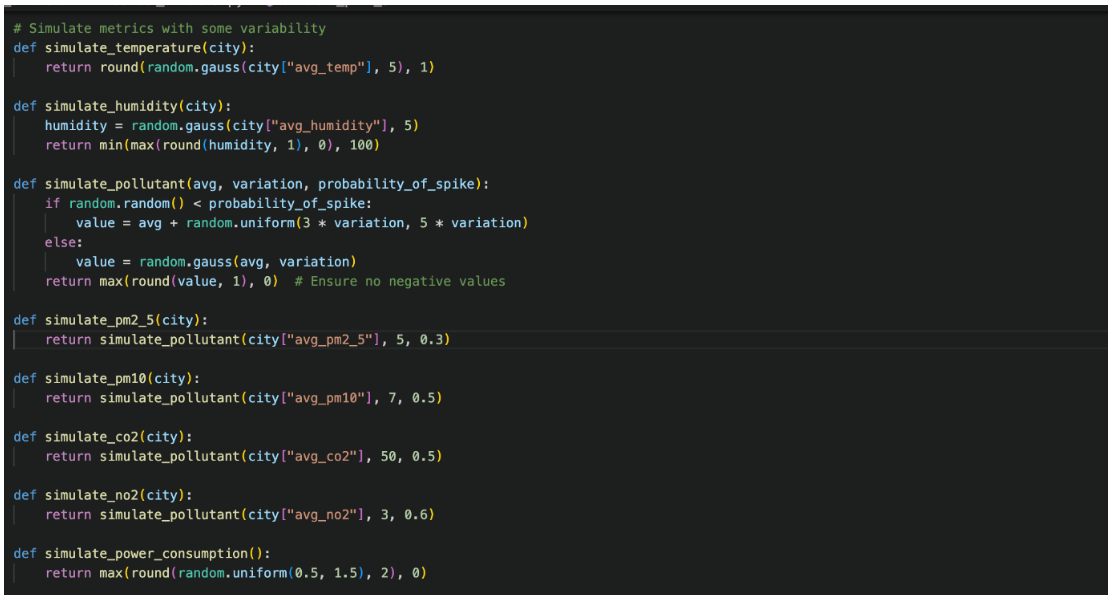
Requirements	Details	Priority
Portability	Fully containerize the system using Docker for deployment across diverse environments.	High
Scalability	The system must support the integration of additional sensors, locations, and data streams easily.	High
Usability	The system must provide user-friendly dashboards and intuitive interfaces for varying expertise levels.	Medium
Security	The system must secure communication channels and prevent unauthorized access to sensor data.	Medium
Resilience	Handle communication failures gracefully and ensure data consistency.	Medium

TECHNICAL IMPLEMENTATION

SENSOR INTEGRATION

1. Simulated Data

A custom Python script with the **paho-mqtt** library was used to simulate sensor data and publish it over MQTT topics. The script leverages Gaussian distributions to model variability in metrics such as power consumption, temperature, humidity, PM2.5, PM10, CO2, NO2, while occasionally introducing spikes to mimic real-world pollution. Each city is pre-configured with realistic environmental averages, ensuring credible and context-specific data.



```python
# Simulate metrics with some variability
def simulate_temperature(city):
    return round(random.gauss(city["avg_temp"], 5), 1)

def simulate_humidity(city):
    humidity = random.gauss(city["avg_humidity"], 5)
    return min(max(round(humidity, 1), 0), 100)

def simulate_pollutant(avg, variation, probability_of_spike):
    if random.random() < probability_of_spike:
        value = avg + random.uniform(3 * variation, 5 * variation)
    else:
        value = random.gauss(avg, variation)
    return max(round(value, 1), 0)  # Ensure no negative values

def simulate_pm2_5(city):
    return simulate_pollutant(city["avg_pm2_5"], 5, 0.3)

def simulate_pm10(city):
    return simulate_pollutant(city["avg_pm10"], 7, 0.5)

def simulate_co2(city):
    return simulate_pollutant(city["avg_co2"], 50, 0.5)

def simulate_no2(city):
    return simulate_pollutant(city["avg_no2"], 3, 0.6)

def simulate_power_consumption():
    return max(round(random.uniform(0.5, 1.5), 2), 0)
```

2. Data Characteristics

The dataset consists of key environmental parameters, operational metadata, and simulated sensor outputs. It can be categorized into metadata and the actual measurements and recorded values.

Metadata

- **Sensor ID**
  - A unique identifier for each sensor in the network.
  - **Format**: CityName_sensor_SensorNumber (e.g., Songdo_sensor_1).
- **City**
  - The smart city where the sensor is installed. The system focuses on monitoring air quality in five urban areas, the predefined cities are:
    - Songdo (South Korea)
    - Amsterdam (Netherlands)
    - San Francisco (USA)
    - Singapore (Singapore)
    - Copenhagen (Denmark)
- **Sensor Type**
  - The type of metric the sensor is measuring.
  - **Values**: temperature, humidity, pm2_5, pm10, co2, no2, power_consumption.
- **Timestamp**
  - The exact time when the data is captured by the sensor.
  - **Format**: Unix timestamp (milliseconds since epoch).

Measured Data

- **Temperature**
  - The ambient temperature at the sensor’s location.
  - **Range**: -10°C to 50°C (varies by city and season).
- **Humidity**
  - The relative humidity of the air at the sensor’s location.
  - **Range**: 30% to 75% (with variations based on city-specific climate).
- **PM2.5**
  - Concentration of fine particulate matter, measured in micrograms per cubic meter (µg/m³).
  - **Range**: 5 µg/m³ to 75 µg/m³.
- **PM10**
  - Concentration of coarse particulate matter, measured in µg/m³.
  - **Range**: 10 µg/m³ to 150 µg/m³.
- **CO2 (Carbon Dioxide)**
  - Concentration of carbon dioxide in the air, measured in parts per million (ppm).
  - **Range**: 400 ppm to 1,000 ppm.
- **NO2 (Nitrogen Dioxide)**
  - Concentration of nitrogen dioxide, measured in micrograms per cubic meter (µg/m³).
  - **Range**: 10 µg/m³ to 100 µg/m³.
- **Power Consumption**
  - The energy consumed by each sensor, measured in watts (W).
  - **Range**: 0.5 W to 1.5 W.
  - **Purpose**: Evaluates sensor efficiency and operational cost.

COMMUNICATION PROTOCOLS

The system uses the **MQTT (Message Queuing Telemetry Transport)** protocol for lightweight, efficient communication between simulated sensors and the data processing backend. MQTT is well-suited for IoT applications due to its low bandwidth requirements and publish/subscribe architecture.

```
# MQTT Server Parameters
MQTT_BROKER = "mosquitto"
MQTT_PORT = 1883
```

MQTT Topics and Format

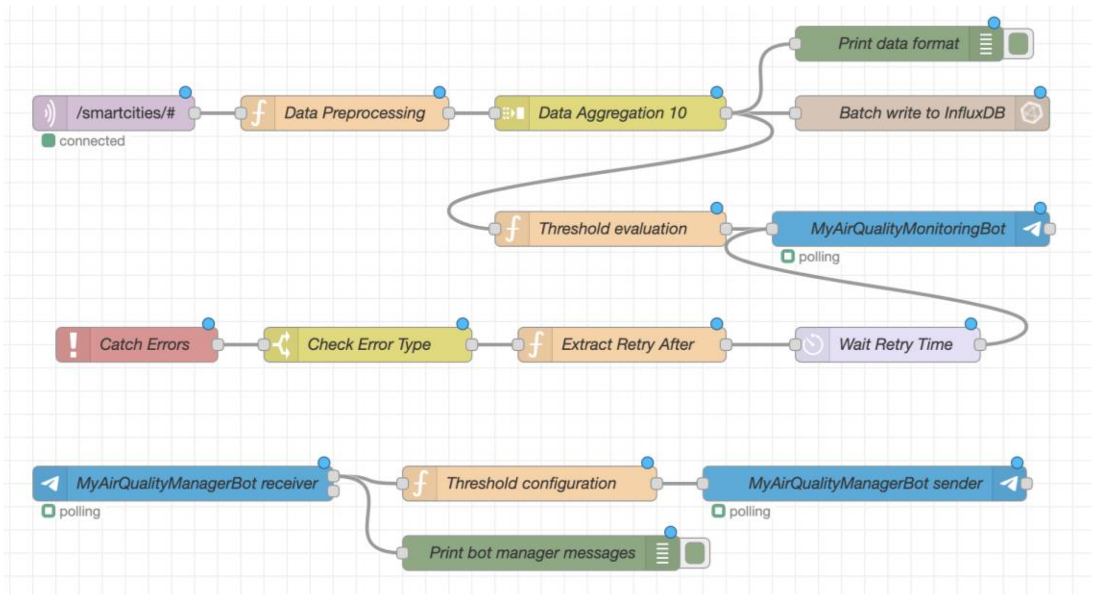
The system employs a structured topic hierarchy to organize and route messages effectively:

- **Topic Format**: /{domain}/{city}/{sensor_type}
  - Example: /smartcities/songdo/temperature
  - **City**: Identifies the location (e.g., Songdo, Amsterdam).
  - **Sensor Type**: Specifies the type of data (e.g., temperature, humidity, etc.).

This format supports easy filtering and subscription. For example:
- Listen to all temperature data across cities: /smartcities/+/temperature
- Focus on specific metrics for one city: /smartcities/amsterdam/pm10

MIDDELWARE (NODE-RED)

Node-RED serves as the core middleware for integrating data streams, performing calculations, and triggering alerts, ensuring seamless operation of the system.



1. **Data Ingestion**:

- Node-RED subscribes to MQTT topics (/smartcities/#) to receive sensor data streamed from simulated sensors.
- The hierarchical topic structure enables filtering and grouping of data (e.g., /smartcities/songdo/pm2_5 for Songdo's PM2.5 data).

2. **Data Preprocessing**:

- Incoming data is parsed and validated to ensure consistency and reliability.
- Preprocessed data is logged and prepared for further processing, such as calculations and aggregation.

3. **Air Quality Index (AQI) Calculation**:

- Node-RED computes the **AQI** for each city based on the PM2.5 and PM10 pollutant levels.
- **Methodology**:
    - Each pollutant's value is mapped to an AQI score using predefined thresholds (e.g., PM2.5 levels of 35 correspond to an AQI of 100).
    - The final AQI for a city is determined by the highest AQI score among all pollutants.
    - Example: If PM2.5 has an AQI of 120 and PM10 has an AQI of 80, the city's AQI is reported as 120.

4. **Data Aggregation and Storage**:

- Data is aggregated over a 10-second window to smooth out fluctuations.

- Aggregated data is written to InfluxDB in batches, where it can be analyzed and visualized in dashboards like Grafana.

5. **Telegram Bot Integration**:

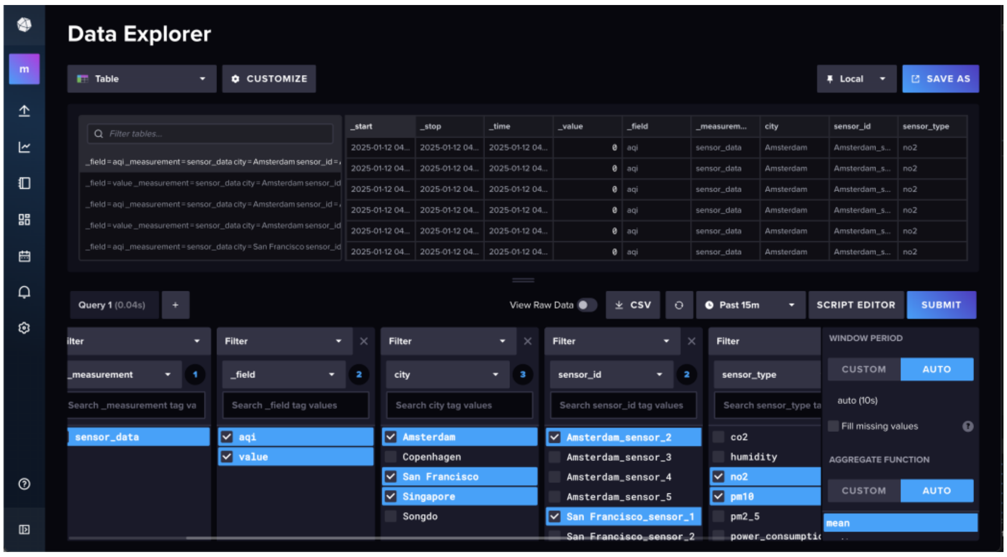
- The bots allow users to:
  - Receive real-time alerts and updates.
  - Query current threshold levels.
  - Update thresholds dynamically for specific metrics.

6. **Error Handling:**

- If a **429: Too Many Requests** error occurs, the system gracefully handles it with a try-catch mechanism, extracting the "Retry-After" value or defaulting to a 10-second wait.
- During this delay, other operations continue uninterrupted, and the alert is retried afterward.

DATA STORAGE

InfluxDB, a time-series database, is used as the storage solution for sensor data in this project. Its focus on time-stamped data and high write/read efficiency makes it ideal for managing IoT datasets such as environmental metrics.



_field = aqi _measurement = sensor_data city = Amsterdam sensor_id =
_field = value _measurement = sensor_data city = Amsterdam sensor_id
_field = aqi _measurement = sensor_data city = Amsterdam sensor_id =
_field = value _measurement = sensor_data city = Amsterdam sensor_id
_field = aqi _measurement = sensor_data city = San Francisco sensor_id

_start	_stop	_time	_value	_field	_measurem...	city	sensor_id	sensor_type
2025-01-12 04...	2025-01-12 04...	2025-01-12 04...	0	aqi	sensor_data	Amsterdam	Amsterdam_s...	no2
2025-01-12 04...	2025-01-12 04...	2025-01-12 04...	0	aqi	sensor_data	Amsterdam	Amsterdam_s...	no2
2025-01-12 04...	2025-01-12 04...	2025-01-12 04...	0	aqi	sensor_data	Amsterdam	Amsterdam_s...	no2
2025-01-12 04...	2025-01-12 04...	2025-01-12 04...	0	aqi	sensor_data	Amsterdam	Amsterdam_s...	no2
2025-01-12 04...	2025-01-12 04...	2025-01-12 04...	0	aqi	sensor_data	Amsterdam	Amsterdam_s...	no2
2025-01-12 04...	2025-01-12 04...	2025-01-12 04...	0	aqi	sensor_data	Amsterdam	Amsterdam_s...	no2

ilter	Filter	Filter	Filter	Filter
_measurement (1)	_field (2)	city (3)	sensor_id (2)	sensor_type
Search _measurement tag va	Search _field tag values	Search city tag values	Search sensor_id tag values	Search sensor_type ta
sensor_data	✓ aqi	✓ Amsterdam	✓ Amsterdam_sensor_2	co2
	✓ value	Copenhagen	Amsterdam_sensor_3	humidity
		✓ San Francisco	Amsterdam_sensor_4	✓ no2
		✓ Singapore	Amsterdam_sensor_5	✓ pm10
		Songdo	✓ San Francisco_sensor_1	pm2_5
			San_Francisco_sensor_2	power_consumptic

WINDOW PERIOD: CUSTOM [✓ AUTO]
auto (10s)
Fill missing values
AGGREGATE FUNCTION: CUSTOM [✓ AUTO]
mean

The data is organized into **measurements**, **tags**, and **fields**:

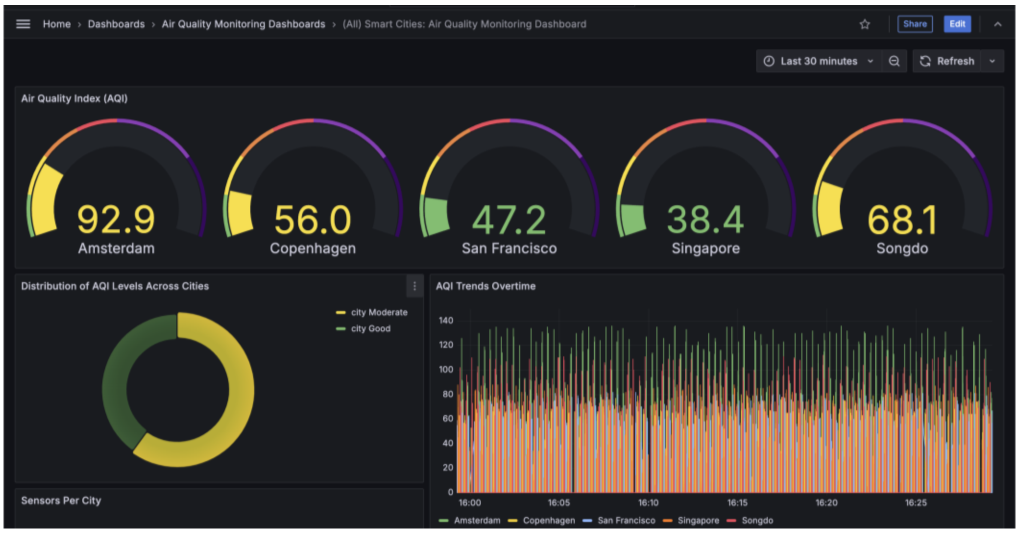
```
msg.payload = {
    fields: {
        value: value,
        aqi: finalAQI,
    },
    tags: {
        sensor_id: msg.payload.sensor_id,
        city: msg.payload.city,
        sensor_type: msg.payload.sensor_type,
    },
    measurement: "sensor_data",
    timestamp: msg.payload.time,
};
```

VISUALIZATION (GRAFANA)

Grafana serves as the tool for visualizing air quality monitoring data. The system includes three well-structured dashboards that provide varying levels of detail. They **highlight threshold violations in red**, enabling users to quickly identify critical issues, and **present summary statistics such as averages**, offering a comprehensive view of air quality trends.

1. (All) Smart Cities: Air Quality Monitoring Dashboard

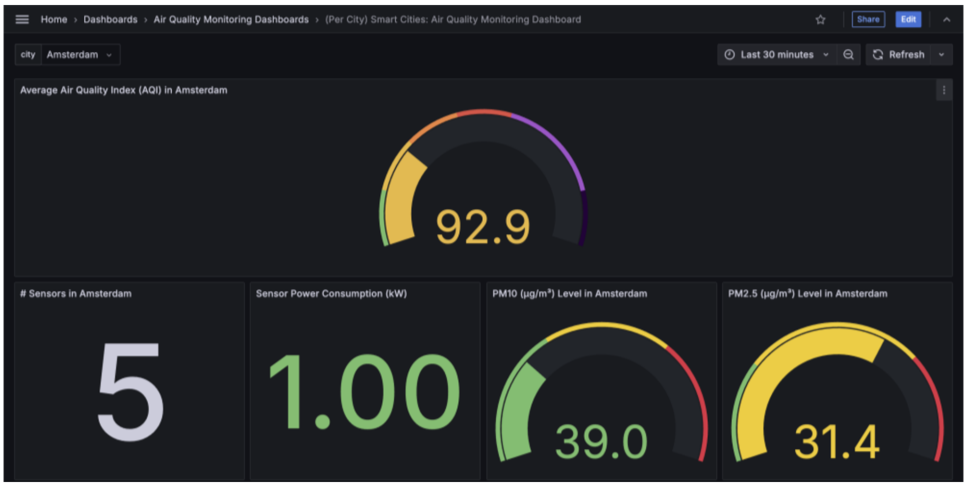
- Provides a high-level overview of all metrics across all cities.
- **Features**:
    - Displays key metrics such as temperature, humidity, PM2.5, PM10, CO2, and NO2 for every city in a consolidated view.
    - Useful for quickly identifying trends and comparing conditions across cities.
- **Use Case**: Ideal for administrators or users who want a summary of air quality conditions across all monitored locations.



2. (Per City) Smart Cities: Air Quality Monitoring Dashboard

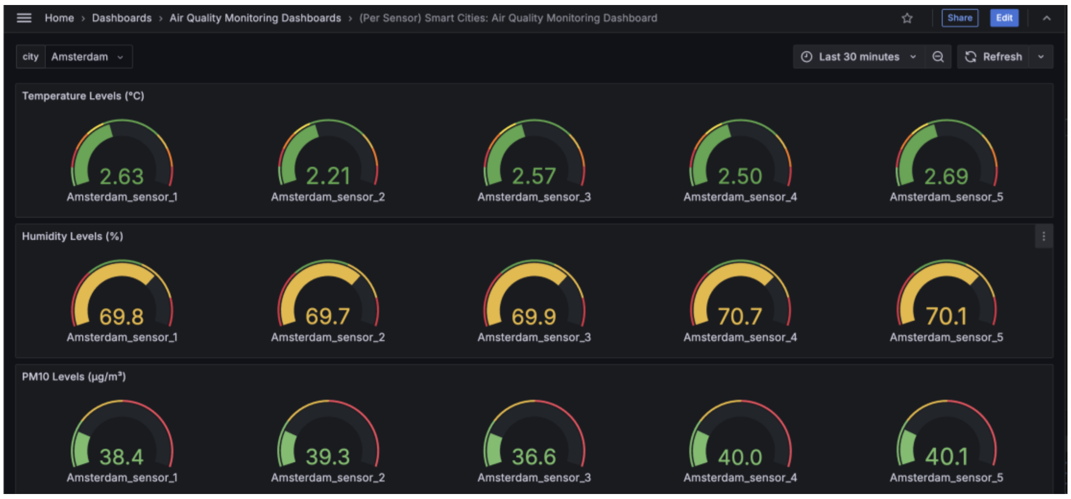
- Focuses on air quality metrics for a specific city.
- **Features**:
    - Dropdown filters enable users to narrow down data to a specific city or time.

- Metrics are displayed for the selected city, providing a detailed view of environmental conditions such as AQI, temperature, and pollutants.
  - Time range filters allow users to analyze historical trends or real-time data.
- **Use Case**: Enables city-level decision-making and in-depth analysis for a particular location.


Home › Dashboards › Air Quality Monitoring Dashboards › (Per City) Smart Cities: Air Quality Monitoring Dashboard
Share
Edit
city Amsterdam
Last 30 minutes
Refresh
Average Air Quality Index (AQI) in Amsterdam
92.9
# Sensors in Amsterdam
5
Sensor Power Consumption (kW)
1.00
PM10 (µg/m³) Level in Amsterdam
39.0
PM2.5 (µg/m³) Level in Amsterdam
31.4

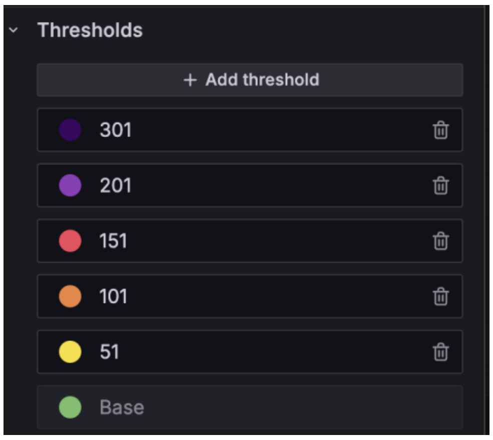
3. (Per Sensor) Smart Cities: Air Quality Monitoring Dashboard

- Offers the most granular view, showing data at the sensor level for each city.
- **Features**:
  - Users can filter by city and time to view individual sensor metrics.
  - Each sensor's data is broken down by type (e.g., temperature, PM2.5) and timestamped for precise tracking.
- **Use Case**: Useful for diagnosing sensor-specific issues or analyzing data from particular locations in detail.



Insights with Threshold Indicators

To enhance data clarity, thresholds are applied directly to Grafana dashboards. For example, for the AQI:



●	301
●	201
●	151
●	101
●	51
●	Base

ALERTING MECHANISMS

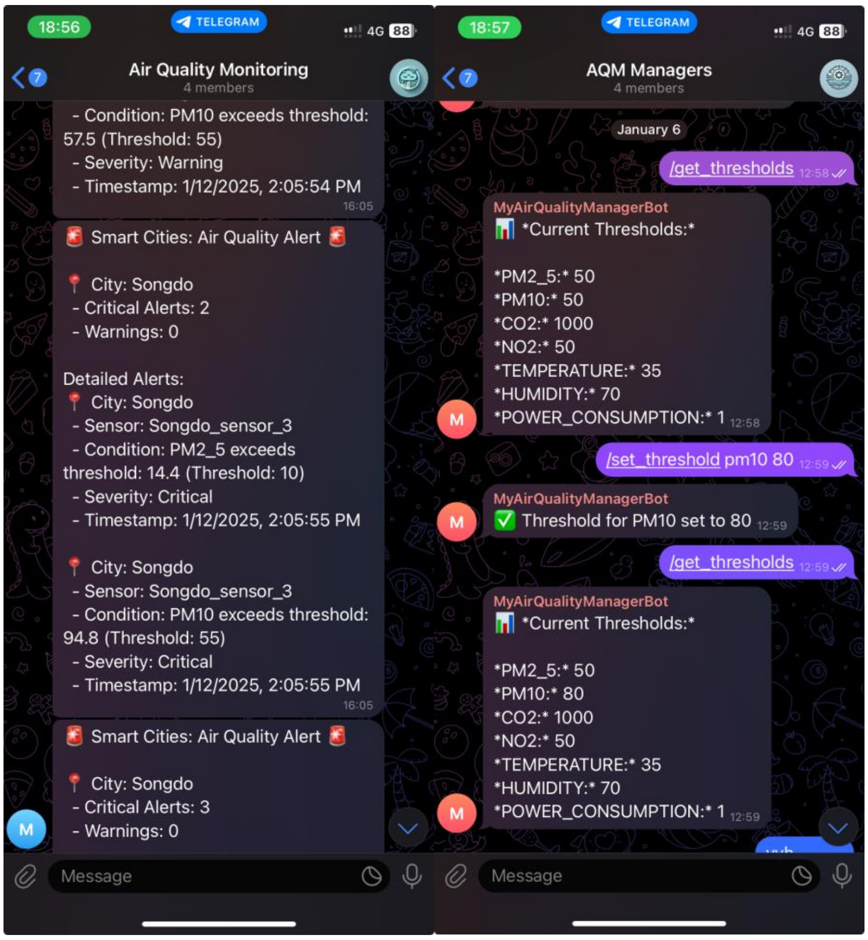
The system employs **Telegram bots** as the means of alerting and user interaction. Two Telegram bots are integrated into the system, each serving a distinct purpose and interacting with dedicated Telegram chats for streamlined communication.

1. Monitoring Alerts

- Provide real-time notifications about air quality threshold violations.
- **How It Works**:
    - **Node-RED** processes incoming data and evaluates it against predefined thresholds (e.g., PM2.5 > 35 µg/m³).
    - If a violation occurs:
        - **Warning Alerts**: Triggered when values exceed the threshold but are within 25% of the threshold limit.
        - **Critical Alerts**: Triggered when values exceed over 25% of the threshold limit.
    - Consolidated alerts are sent to a **Telegram Monitoring Chat** via the **MyAirQualityMonitoringBot**.
    - Aggregated alerts per city are sent in the case where more than one alert message is available.

2. Manager Interaction and Threshold Configuration

- Allow managers to view and update threshold values dynamically via the **MyAirQualityManagerBot**.
- **How It Works**
    - Managers use specific commands in a **Telegram Manager Chat** to interact with the system:
        - **View Current Thresholds**: Using command */get_thresholds*.
    - **Update Thresholds**: Using command */set_threshold <metric> <value>.*
        - The updated thresholds are stored globally in Node-RED, ensuring all future evaluations use the new values.


18:56 TELEGRAM 4G 88

Air Quality Monitoring
4 members

- Condition: PM10 exceeds threshold: 57.5 (Threshold: 55)
- Severity: Warning
- Timestamp: 1/12/2025, 2:05:54 PM
16:05

🚨 Smart Cities: Air Quality Alert 🚨

📍 City: Songdo
- Critical Alerts: 2
- Warnings: 0

Detailed Alerts:
📍 City: Songdo
- Sensor: Songdo_sensor_3
- Condition: PM2_5 exceeds threshold: 14.4 (Threshold: 10)
- Severity: Critical
- Timestamp: 1/12/2025, 2:05:55 PM

📍 City: Songdo
- Sensor: Songdo_sensor_3
- Condition: PM10 exceeds threshold: 94.8 (Threshold: 55)
- Severity: Critical
- Timestamp: 1/12/2025, 2:05:55 PM
16:05

🚨 Smart Cities: Air Quality Alert 🚨

📍 City: Songdo
- Critical Alerts: 3
- Warnings: 0

Message

18:57 TELEGRAM 4G 88

AQM Managers
4 members

January 6

/get_thresholds 12:58

MyAirQualityManagerBot
📊 *Current Thresholds:*

PM2_5: 50
PM10: 50
CO2: 1000
NO2: 50
TEMPERATURE: 35
HUMIDITY: 70
POWER_CONSUMPTION: 1 12:58

/set_threshold pm10 80 12:59

MyAirQualityManagerBot
✅ Threshold for PM10 set to 80 12:59

/get_thresholds 12:59

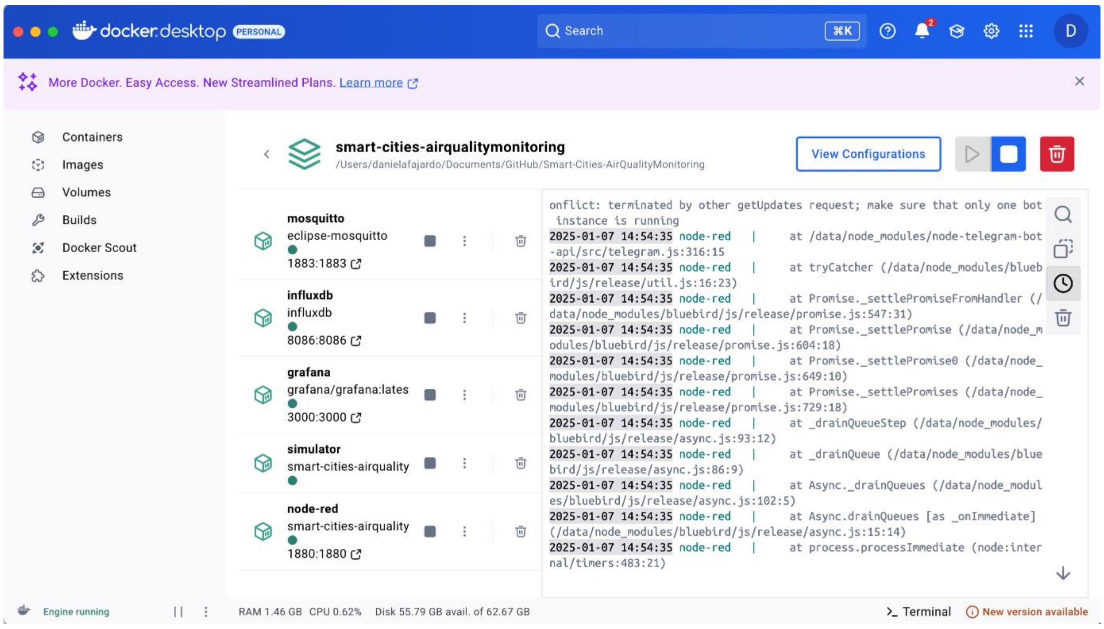
MyAirQualityManagerBot
📊 *Current Thresholds:*

PM2_5: 50
PM10: 80
CO2: 1000
NO2: 50
TEMPERATURE: 35
HUMIDITY: 70
POWER_CONSUMPTION: 1 12:59

Message

DEPLOYMENT (DOCKER)

The system is containerized using **Docker**, ensuring portability, scalability, and ease of deployment. Each major component of the air quality monitoring system runs in its own container, managed via **Docker Compose** for orchestration. This approach simplifies setup and allows seamless integration of all services.



smart-cities-airqualitymonitoring
/Users/danielafajardo/Documents/GitHub/Smart-Cities-AirQualityMonitoring

mosquitto
eclipse-mosquitto
1883:1883

influxdb
influxdb
8086:8086

grafana
grafana/grafana:lates
3000:3000

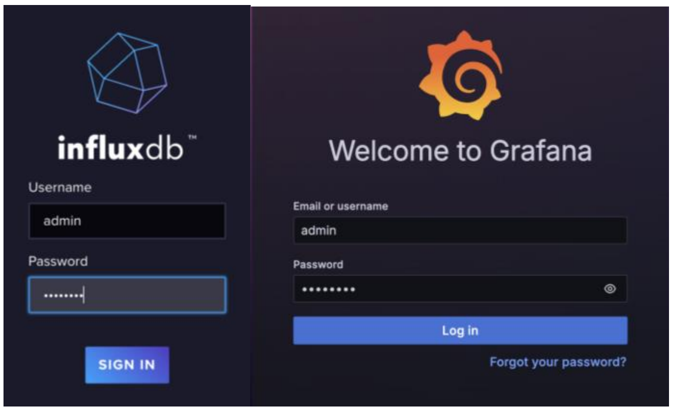
simulator
smart-cities-airquality

node-red
smart-cities-airquality
1880:1880

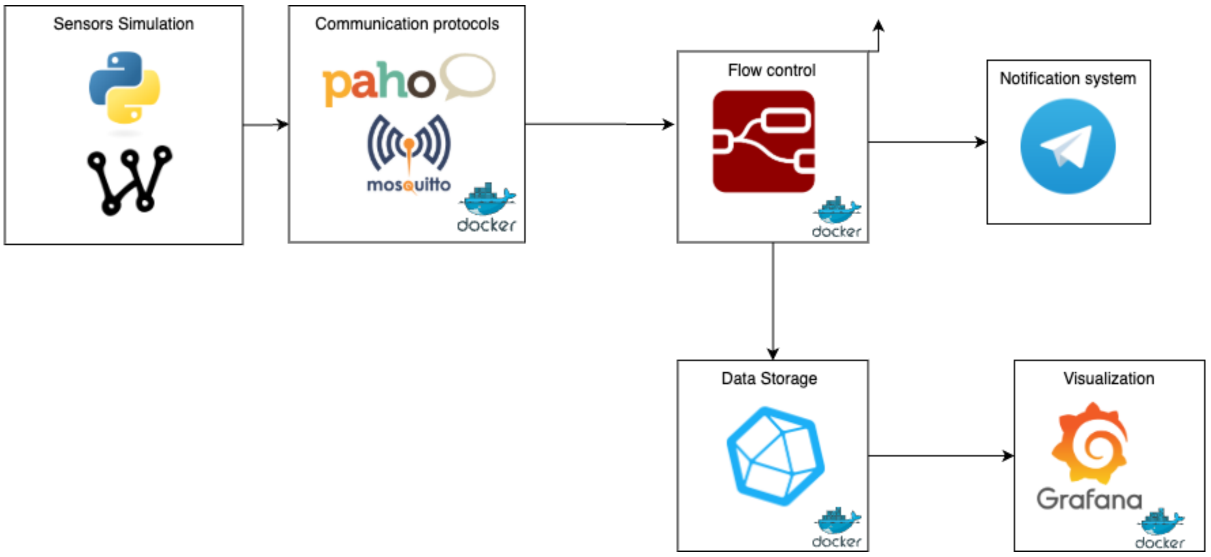
```
onflict: terminated by other getUpdates request; make sure that only one bot
 instance is running
2025-01-07 14:54:35 node-red   |     at /data/node_modules/node-telegram-bot
-api/src/telegram.js:316:15
2025-01-07 14:54:35 node-red   |     at tryCatcher (/data/node_modules/blueb
ird/js/release/util.js:16:23)
2025-01-07 14:54:35 node-red   |     at Promise._settlePromiseFromHandler (/
data/node_modules/bluebird/js/release/promise.js:547:31)
2025-01-07 14:54:35 node-red   |     at Promise._settlePromise (/data/node_m
odules/bluebird/js/release/promise.js:604:18)
2025-01-07 14:54:35 node-red   |     at Promise._settlePromise0 (/data/node_
modules/bluebird/js/release/promise.js:649:10)
2025-01-07 14:54:35 node-red   |     at Promise._settlePromises (/data/node_
modules/bluebird/js/release/promise.js:729:18)
2025-01-07 14:54:35 node-red   |     at _drainQueueStep (/data/node_modules/
bluebird/js/release/async.js:93:12)
2025-01-07 14:54:35 node-red   |     at _drainQueue (/data/node_modules/blue
bird/js/release/async.js:86:9)
2025-01-07 14:54:35 node-red   |     at Async._drainQueues (/data/node_modul
es/bluebird/js/release/async.js:102:5)
2025-01-07 14:54:35 node-red   |     at Async.drainQueues [as _onImmediate]
(/data/node_modules/bluebird/js/release/async.js:15:14)
2025-01-07 14:54:35 node-red   |     at process.processImmediate (node:inter
nal/timers:483:21)
```

RAM 1.46 GB CPU 0.62% Disk 55.79 GB avail. of 62.67 GB

- **Security Configuration**
  - Both InfluxDB and Grafana are configured with API tokens as well as admin usernames and passwords that are passed as environment variables in the docker-compose.yml file for secure access.



SYSTEM ARCHITECTURE



ADVANTAGES & LIMITATIONS

Advantages	Limitations
Real-time air quality monitoring.	Accuracy depends on the realism of simulated data.
Configurable thresholds for actionable alerts, and modular design for scalability and portability.	Limited by the processing capabilities of middleware.
Comprehensive dashboards for decision-making.	Deployment requires network connectivity and reliable MQTT infrastructure.

CONCLUSION

The **IoT-Based Real-Time Air Quality Monitoring** demonstrates how IoT technologies can address smart city air quality challenges through real-time monitoring, dynamic processing, and actionable insights. By simulating realistic environmental data, leveraging Node-RED for processing, and using Grafana for visualization, the system empowers policymakers and citizens to make data-driven decisions for healthier cities. Real-time alerts via Telegram bots and dynamic threshold management enable timely responses to critical air quality risks. This project highlights the potential of scalable, secure IoT systems to integrate real sensors and tackle broader environmental issues, while emphasizing the importance of modular architectures, user-friendly interfaces, and robust error handling in addressing real-world challenges.

FUTURE SCOPE

Looking ahead, the Air Quality Monitoring System can focus on several enhancements to elevate its functionality. The integration of advanced machine learning algorithms could enable **predictive analysis**, providing proactive measures to address air quality issues. Additionally, expanding the system's applications to include **utility monitoring, traffic data correlation, and energy usage** would further support smart city objectives. Finally, developing a **mobile-friendly application** for remote monitoring and control would enhance accessibility and engagement, paving the way for smarter and more connected urban living.

REFERENCES

1. Using Air Quality Index | AirNow.gov. (n.d.). https://www.airnow.gov/aqi/aqi-basics/using-air-quality-index/
2. United Nations Environment Programme. (n.d.). How is air quality measured? UNEP. https://www.unep.org/news-and-stories/story/how-air-quality-measured