NOISE POLLUTION MONITORING

Abstract:

Noise or sound level monitoring or measurement is a process to measure the magnitude of Noise in industries and residental area. Data collected from Noise level monitoring Testng helps us to understand trends and acton can be taken to reduce noise polluton. It helps identff work locatons where there are noise problems, emplofees who maf be exposed to noise levels that can cause hearing loss, and where additonal noise measurements need to be made. This informaton also helps determine appropriate noise control measures that need to be put in place.Noise polluton is Low or Highfrequencf sound that can cause/harm the actvitf of human life. It can be caused bf various industrial Machines, Motor Vehicles and Craf etc. Noise Polluton Monitoring process is a part of Environmental Monitoring Testng as noise polluton is also increasing exponentallf in recent fears.

The increasing air and sound pollution is one of the significant issue now days. As the pollution increasing It is giving rise number of diseases so, it has become essential to control the pollution for better future And healthy life .here we propose an air quality as well as sound pollution monitoring system that allows Us to monitor and check live air quality as well as sound pollution monitoring in particular areas through IOT. System uses air sensor to detect or sense presence of harmful gases, compounds in the air and Constantly transmit data to microcontroller. Also system keeps measure soundlevel and report it to the Online server over IOT. The user friendly and easy handling of the system technology is such that it can be Installed in houses, schools and in smallplaces.

INTRODUCTION:

An IoT-based noise monitoring system is a system that uses the Internet of Things (IoT) technology to monitor the noise or sound intensity in the environment for the safety of human beings . The system comprises a sound sensor, an IoT platform called NodeMCU, LCD, and LEDs . The sound sensor module is a small board that mixes a microphone (50Hz-10kHz) and some processing circuitry to convert sound waves into electrical signals . This electrical signal is fed to an on-board LM393 High Precision Comparator to digitize it and is made available at the OUT pin . The module features a built-in potentiometer for sensitivity adjustment of the OUT signal . The IoT decibel meter measures sound in decibels (dB) using a sound sensor and displays it on an LCD display . It also pushes the readings to the Blynk IoT platform, making it accessible from across the world . A device like this will be useful in places like hospitals and schools to track and monitor the sound levels and take action accordingly .

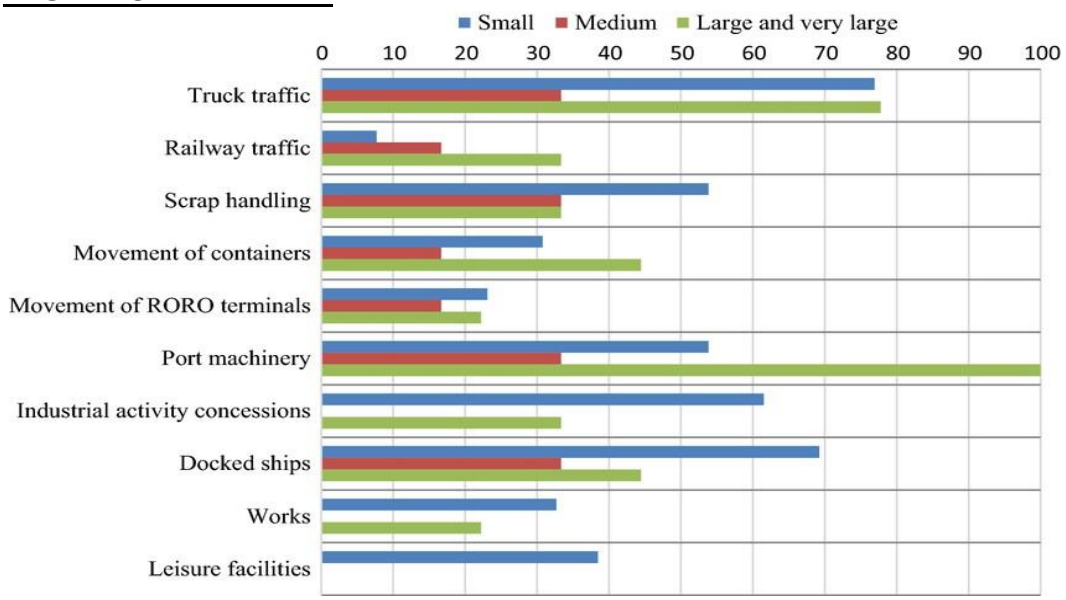
ANALYSIS:

NOISE POLLUTIONS PATTERNS:

I found some information on noise pollution patterns and data analytics. One study published in the International Journal of Environmental Research and Public Health analyzed the sampling methodologies for noise pollution assessment and their impact on the population. The study concluded that the selection of a suitable method for performing noise maps through measurements is essential to achieve an accurate assessment of the impact of noise pollution on the population.

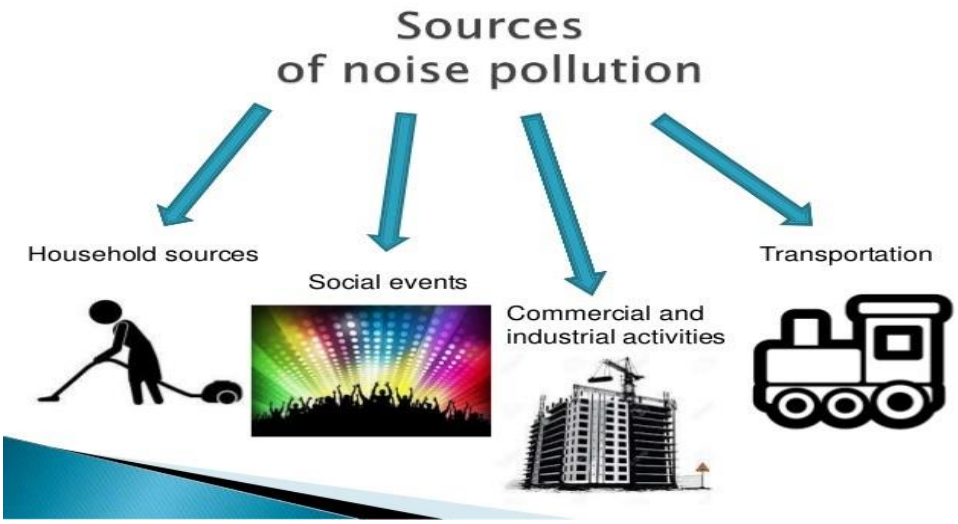
Another source suggests that data analysis is a great tool for noise mapping. This procedure involves analyzing large amounts of data on noise levels, studying trends and patterns, identifying high noise level areas and potential sources, and prioritizing areas for action.

HIGH NOISE AREAS



Noise pollution is the propagation of noise or sound with ranging impacts on the activity of human or animal life, most of which are harmful to a degree . The source of outdoor noise worldwide is mainly caused by machines, transport and propagation systems . Poor urban planning may give rise to noise disintegration or pollution, side-by-side industrial and residential buildings can result in noise pollution in the residential areas . Some of the main sources of noise in residential areas include loud music, transportation (traffic, rail, airplanes, etc.), lawn care maintenance, construction, electrical generators, wind turbines, explosions and people .

SOURCES:



Noise pollution is a type of environmental pollution that can cause health problems for people and wildlife, both on land and in the sea . Noise can come from many sources, including household gadgets like food mixers, grinders, vacuum cleaners, washing machines, dryers, coolers, and air conditioners . Other sources include loudspeakers of sound systems and TVs, iPods, and earphones . Social events such as places of worship, discos and gigs, parties, and other social events also create a lot of noise for the people living in that area . In many market areas, people sell with loudspeakers. Others shout out offers and try to get customers to buy their goods . Commercial and industrial activities such as printing presses, manufacturing industries, and construction sites contribute to noise pollution in large cities . In many industries, it is a requirement that people always wear earplugs to minimize their exposure to heavy noise. People who work with lawnmowers, tractors, and noisy equipment are also required to wear noise-proof gadgets . Transportation such as airplanes flying over houses close to busy airports like Heathrow (London) or O'hare (Chicago), overground and underground trains, vehicles on roads are always making a lot of noise .

CONCLUSION:

Noise pollution is a growing concern in many parts of the world. According to a report by NoiseOFF, community noise can have a range of adverse health effects, including hearing impairment, startle and defense reactions, aural pain, ear discomfort speech interference, sleep disturbance, cardiovascular effects, performance reduction, and annoyance responses. The report also highlights that

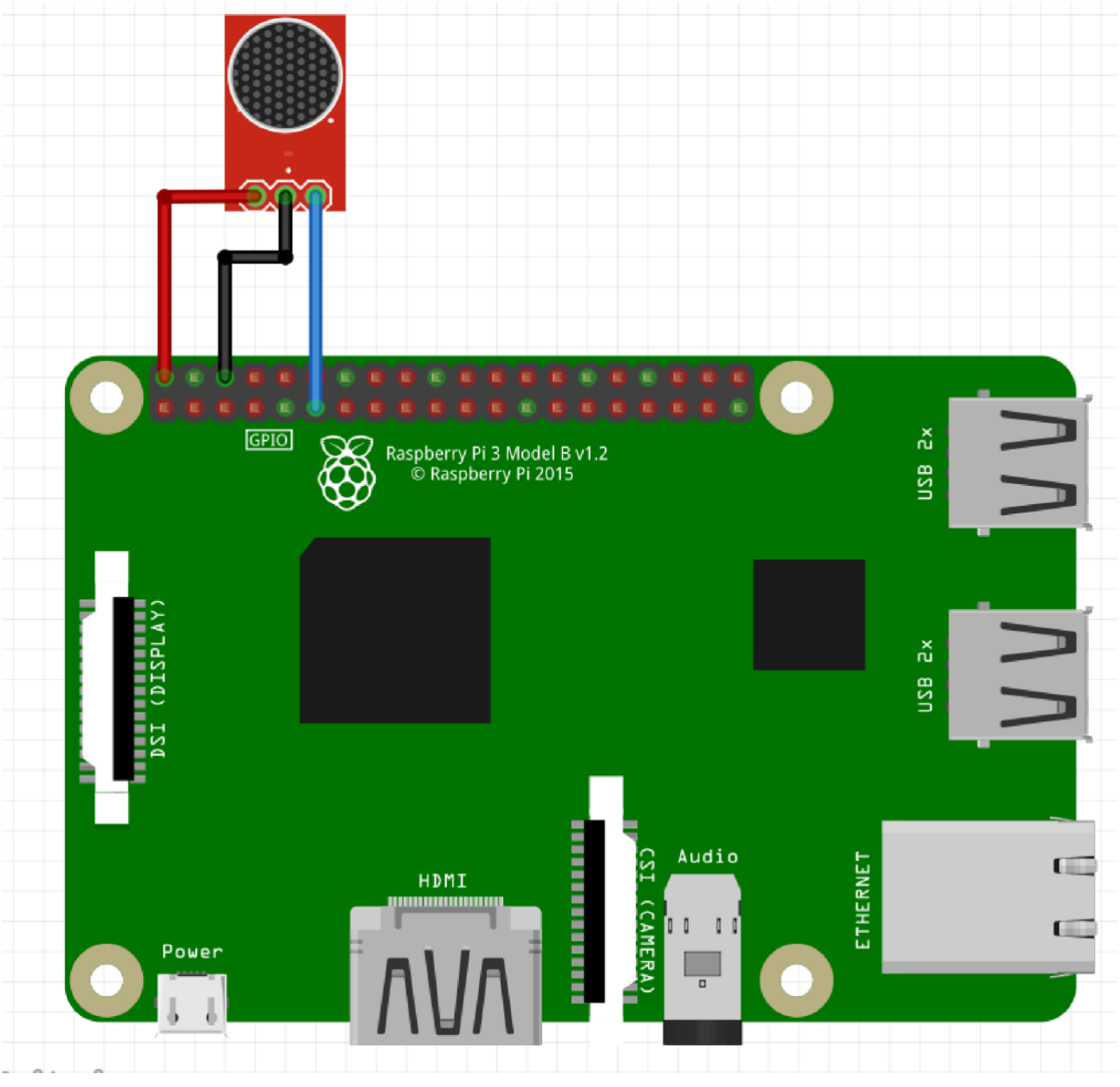
the sources of outdoor noise worldwide are mainly caused by machines, transport, and propagation systems.

OVERVIEW :

Introduction:

As urban areas continue to grow and expand, so does the concern over noise pollution. To address this challenge, the development of an IoT-enabled Noise Pollution Monitoring system is crucial. Such a system leverages IoT sensors placed in public areas to measure noise levels in real-time and collect valuable data.This project aims to provide an illustrative example of how such a system can be implemented. A synthetic dataset has been created to simulate noise level data, and a Python script has been developed to demonstrate how an IoT sensor can send this data to a noise pollution information platform. The data collected from these sensors can be used to monitor noise levels, identify trends, and make informed decisions for noise pollution control and management.

Circuit diagram:



Circuit diagram explanation :

In the context of an IoT-enabled Noise Pollution Monitoring system, the use of a sound sensor, specifically a sound level sensor or microphone, is a fundamental component. This sensor is employed in conjunction with a Raspberry Pi to continuously monitor and assess fluctuations in the acoustic environment.The sound sensor is a sensitive component designed to detect and convert sound waves into electrical signals. It captures sound data from its surroundings and transmits it to the Raspberry Pi for analysis and processing. The Raspberry Pi serves as the brain of the system, providing the necessary computational power to receive, interpret, and store the incoming noise data.

Dataset:

```
Timestamp,Noise_Level
2023-10-18T12:00:00,70.2
2023-10-18T12:15:00,68.5
2023-10-18T12:30:00,72.8
2023-10-18T12:45:00,69.3
2023-10-18T13:00:00,75.1
2023-10-18T13:15:00,71.9
2023-10-18T13:30:00,68.7
2023-10-18T13:45:00,74.5
2023-10-18T14:00:00,67.4
2023-10-18T14:15:00,73.2
```

You can save this data to a CSV file and use it as your dataset. To create a larger dataset, you can use the Python code to generate more timestamps and random noise levels as needed.

Code:

```
import random
import time
from datetime import datetime
from paho.mqtt import client as mqtt

# Simulated IoT Sensor
def simulate_noise_sensor():
    return random.uniform(40, 100)  #
Simulating noise level data

# MQTT Broker Configuration (you may need to
modify this for your chosen platform)
mqtt_broker = "broker.example.com"
mqtt_port = 1883
mqtt_topic = "noise_data_topic"
client_id = "noise_sensor_client"

# Initialize MQTT Client
client = mqtt.Client(client_id)

# Connect to the MQTT Broker
client.connect(mqtt_broker, mqtt_port,
keepalive=60)

# Simulate and Send Noise Data
while True:
    noise_level = simulate_noise_sensor()
    timestamp = datetime.now().isoformat()
    payload = f"{{'timestamp': '{timestamp}',
'noise_level': {noise_level}}}"

    # Publish data to the IoT platform
    client.publish(mqtt_topic, payload)

    print(f"Published: {payload}")

    time.sleep(10)  # Simulate sending data
every 10 seconds

# Disconnect from MQTT when done
```

```
client.disconnect()
```

Code explanation:

1. Importing Libraries:

```
import random
import time
from datetime import datetime
from paho.mqtt import client as mqtt
```

- We import necessary libraries for generating random numbers (random), working with time (time), handling timestamps (datetime), and connecting to an MQTT broker (paho.mqtt).

2. Simulated IoT Sensor:

```
def simulate_noise_sensor():
    return random.uniform(40, 100)  #
Simulating noise level data
```

- This function simulate_noise_sensor generates a random noise level data between 40 and 100 (you can adjust this range to simulate realistic noise levels).

3. MQTT Broker Configuration:

```
mqtt_broker = "broker.example.com"
mqtt_port = 1883
mqtt_topic = "noise_data_topic"
client_id = "noise_sensor_client"
```

- Here, we specify the MQTT broker's address, port, the topic to which we will publish the data, and a client ID for the IoT sensor.

4. Initialize MQTT Client:

```
client = mqtt.Client(client_id)
```

- We create an MQTT client with the specified client ID.

5. Connect to MQTT Broker:

```
client.connect(mqtt_broker, mqtt_port,
keepalive=60)
```

- The script establishes a connection to the MQTT broker using the provided address and port.

6. Simulate and Send Noise Data:

```
while True:
    noise_level = simulate_noise_sensor()
    timestamp = datetime.now().isoformat()
    payload = f"{{'timestamp':
'{timestamp}', 'noise_level':
{noise_level}}}"

    client.publish(mqtt_topic, payload)

    print(f"Published: {payload}")

    time.sleep(10)
```

- In this loop, the script simulates noise data:It generates a noise level using the simulate_noise_sensor function.It creates a timestamp using the current time and formats it as an ISO string.It assembles a JSON payload with the timestamp and noise level.The script publishes this payload to the MQTT topic.It prints the sent data for reference.It then waits for 10 seconds before sending the next data point.

7. **Disconnect from MQTT:**

```
client.disconnect()
```

- When the script is done (which might not happen in this infinite loop), it disconnects from the MQTT broker.

Conclusion:

In conclusion, the integration of a sound sensor with a Raspberry Pi creates a powerful tool for real-time noise pollution monitoring, o ering insights into the acoustic environment, and contributing to better-informed decisions regarding noise management and urban planning.

APPLICATION:

Introduction:

Understanding Noise Pollution: Noise pollution, characterized by excessive, unwanted, or disruptive sounds, has adverse e ects on human health and well-being.

Key Features:

The app o ers real-time noise monitoring through IoT sensors and smartphone microphones.Data visualization tools present noise data via user-friendly charts and graphs.Threshold alerts notify users when noise levels exceed preset limits.Historical data tracking enables the observation of noise patterns over time.Noise maps reveal noise distribution in specific areas.User profiles allow customization of settings and preferences.

IoT Integration:

IoT devices, such as microphones and sound sensors, collect and transmit noise data for analysis.

Data Analysis and Machine Learning:Advanced apps employ data analysis and machine learning to identify noise patterns and sources.

Server-Side Development:A server or cloud infrastructure is established for secure noise data storage.

Mobile App Development: Native or cross-platform mobile apps are developed for iOS and Android platforms, delivering real-time noise updates.

User Experience (UX) Design:

A user-friendly interface ensures accessibility and comprehension of noise data.Notifications and Alerts:Push notifications are used to inform users of noise level breaches.

Regulatory Compliance:

Local regulations and standards are adhered to for noise pollution monitoring.

Privacy and Security:

Data privacy and security measures safeguard user data.

Testing and Quality Assurance:

Rigorous testing ensures accurate noise level measurements.

User Education:

Educational materials inform users about the health impacts of noise pollution and mitigation strategies.

Web based dashboard administration:

Developing a web-based dashboard administration for a smart noise pollution monitoring app using IoT is crucial for managing and visualizing the collected noise data.

Here's an outline of how to create such a dashboard:

Front-End Development:

Choose web development technologies like HTML, CSS, and JavaScript to build the user interface of the dashboard.Select a JavaScript framework or library (e.g., React, Angular, Vue.js) for a responsive and interactive user experience.Design an intuitive and user-friendly interface for administrators to view and manage noise data.

Data Visualization:

Use data visualization libraries (e.g., Chart.js, D3.js) to create charts and graphs displaying real-time and historical noise data.Include features like zooming, panning, and filtering to help administrators analyze noise patterns e ectively.

User Authentication and Authorization:

Implement secure user authentication to ensure that only authorized personnel can access the dashboard.Define di erent user roles and permissions to control what actions users can perform.

Real-Time Data Display:

Integrate real-time data updates from IoT sensors to display current noise levels and trends.Implement WebSocket or server-sent events for live data streaming to the dashboard.

Historical Data Analysis:

Enable administrators to access and analyze historical noise data with various time range selections.Provide data export and download options for further analysis.

Map Integration:

Incorporate mapping tools to visualize noise levels across specific geographic areas.Display noise data on maps and allow administrators to pinpoint noise sources.

Alerts and Notifications:

Create a system for setting noise level thresholds and sending alerts to administrators when thresholds are exceeded.Allow customization of alert preferences, such as email or SMS notifications.

Data Management:

Implement features for data management, including adding new sensors, editing sensor configurations, and handling sensor calibration.

Reporting and Exporting:

Develop tools for generating and exporting noise pollution reports.Allow administrators to customize report parameters and formats.

Security and Privacy:

Ensure robust security measures to protect sensitive data and user information.Comply with data privacy regulations and standards.

Scalability:

Design the dashboard to be scalable, so it can handle a growing number of IoT devices and data without performance issues.

Maintenance and Support:

Establish a system for monitoring the dashboard's health and addressing issues promptly.Provide user support and updates to enhance functionality and security.

Documentation:

Create comprehensive documentation for administrators to e ectively use and manage the dashboard.

App development:

Developing a smart noise pollution monitoring app using IoT involves creating a mobile application that allows users to monitor and manage noise levels in real-time. Below are the key steps and considerations for developing such an app:

Project Planning:

Define the app's objectives, target audience, and the specific features you want to include.Determine the IoT devices and sensors you'll use for noise data collection.

Choose Development Platforms:

decide whether you want to develop native apps for both iOS and Android or opt for a crossplatform solution using frameworks like React Native or Flutter.

User Interface Design:

Design an intuitive and user-friendly interface that displays noise data e ectively.Create wireframes and mockups to plan the app's layout.

IoT Sensor Integration:

Connect and configure IoT noise sensors (microphones) to collect real-time noise data.Ensure proper communication between the sensors and the mobile app, using technologies like Bluetooth, Wi-Fi, or cellular data.

Data Processing:

Implement data processing on the mobile app to interpret and display noise data.Apply filters or averaging techniques to present meaningful noise level readings.

Data Storage:

Store historical noise data securely, either locally on the device or on a remote server.

Real-Time Monitoring:

Develop features to display real-time noise levels, often in the form of charts or graphs.Enable users to customize monitoring settings.

Threshold Alerts:

Allow users to set noise level thresholds and receive notifications when these thresholds are exceeded.

Location Services:

Utilize GPS or location services to geotag and map noise data, providing information on where high noise levels occur.

User Profiles and Preferences:

Implement user accounts and profiles to store settings, preferences, and historical data.Allow users to customize notification settings.

Data Visualization:

Use data visualization libraries to create charts, graphs, and maps that provide insights into noise levels over time.

Security and Privacy:

Ensure that user data is handled securely and in compliance with data privacy regulations.Implement secure user authentication and data encryption.

Testing and Quality Assurance:

Thoroughly test the app to identify and resolve any bugs, ensuring accurate noise level measurements.

App Deployment:

Publish the app on the Apple App Store and Google Play Store after testing and ensuring it meets platform-specific guidelines.

Maintenance and Updates:

Continuously monitor the app's performance, fix issues, and provide regular updates to enhance features and security.

User Education:

Include informative content within the app to educate users about the impacts of noise pollution and ways to mitigate it.

Building a smart noise pollution monitoring app using IoT is a multi-faceted project that combines hardware, software, data processing, and user experience design. It has the potential to o er valuable insights for users looking to manage noise pollution in their surroundings.

Real time updates:

Sensors:

Use noise level sensors (e.g., microphones) to measure noise pollution. These sensors should be connected to IoT devices.

IoT Devices:

You'll need IoT devices like Raspberry Pi, Arduino, or dedicated IoT boards. These devices collect data from the sensors and transmit it to a central server or cloud platform.

Connectivity:

Set up a reliable network connection for your IoT devices, such as Wi-Fi, cellular, or LoRaWAN, to ensure real-time data transmission.Data Transmission: Send the noise level data to a cloud platform. MQTT or HTTP APIs are commonly used for this purpose.

Cloud Platform:

Use a cloud platform like AWS, Azure, or Google Cloud to receive, store, and process the data. You can use services like AWS IoT Core or Azure IoT Hub to manage IoT devices and data.

Data Processing:

Analyze and process the incoming data in real-time. Calculate noise levels, generate alerts, and store historical data for future analysis.

App Development:

Create a mobile or web app for users to access real-time noise pollution data. You can use native app development (e.g., Android Studio for Android) or cross-platform frameworks like Flutter or React Native.

User Interface:

Design a user-friendly interface displaying noise levels, locations, and other relevant information. Include features like maps, charts, and notifications.

Real-Time Updates:

Implement a mechanism for real-time updates in the app. Use technologies like WebSockets or server-sent events (SSE) to push data to users as it becomes available.

Notifications:

Integrate push notifications to alert users when noise pollution exceeds predefined thresholds.Data Visualization: Create visualizations like graphs, heatmaps, or color-coded maps to help users understand noise pollution patterns.

User Management:

Implement user authentication and access control to ensure that only authorized users can access the data.

Scalability and Reliability:

Ensure your system can handle a large number of IoT devices and users while maintaining high availability.

Maintenance and Updates:

Regularly update both the hardware and software components to ensure system reliability and security.

Compliance:

Ensure that your system complies with local regulations and data privacy requirements, especially if you're collecting and sharing sensitive location-based data.

Data analytics and Report:

Data Collection:

Gather data from IoT devices, including noise level measurements, location, and timestamps.

Data Preprocessing:

Clean the data by removing outliers and errors.Aggregate data over time intervals (e.g., hourly, daily) to create a more manageable dataset for analysis.

Data Analysis:

Calculate various statistics, such as average noise levels, peak noise levels, and noise level distributions.Identify trends, patterns, and anomalies in the data.Correlate noise data with other factors like time of day, weather conditions, or tra c patterns.

Visualization:

Create visualizations to make the data more understandable. Common types of visualizations include:Time series graphs showing noise levels over time.Heatmaps to display noise intensity across geographic areas.Bar charts or histograms to represent noise level distributions.Scatterplots to explore relationships with other variables.

Report Generation:

Generate automated reports summarizing the analysis results. These reports can be in PDF, HTML, or other formats.Key Metrics:Include key metrics like daily average noise levels, noise level trends over time, and areas with the highest noise pollution.

Geospatial Analysis:

Use geographic information systems (GIS) to analyze spatial aspects of noise pollution. Identify noise hotspots and patterns in di erent regions.

Customization:

Allow users to customize the type of reports they want, such as daily, weekly, or monthly summaries.Alerts and Recommendations:Include alerts in the reports for noise levels that exceed predefined thresholds.Provide recommendations for noise reduction or mitigation based on the analysis.

Data Export:

Allow users to export raw or processed data for their own analysis if needed.Historical Data:Include historical data in the reports to track changes and improvements in noise pollution over time.

Interactive Dashboards:

Create interactive dashboards that allow users to explore the data and generate custom reports on the fly.

User Access and Sharing:

Implement access control to ensure that only authorized users can view reports.Allow users to share reports with others.

Data Security and Privacy:

Ensure that the data analysis and report generation process adheres to data security and privacy regulations.

Scalability:

Design the system to handle a growing amount of data and users over time.Feedback Mechanism:Incorporate a way for users to provide feedback on the reports to continually improve the analysis and reporting process.

Program:

```
import pyaudio import

numpy as np


CHUNK = 1024

FORMAT = pyaudio.paInt16

CHANNELS = 1

RATE = 44100

audio = pyaudio.PyAudio()


stream = audio.open(format=FORMAT, channels=CHANNELS, rate=RATE, input=True,

                frames_per_bu er=CHUNK)


while True:

   data = np.fromstring(stream.read(CHUNK), dtype=np.int16)

volume = np.abs(data).mean()     print(f'Noise level: {volume}')


stream.stop_stream() stream.close()

audio.terminate()
```

Conclusion:

In conclusion, the development of a smart noise pollution monitoring app using IoT technology is a significant step towards addressing the growing issue of noise pollution in urban areas. This innovative solution o ers real-time data collection, analysis, and visualization, empowering both individuals and authorities to make informed decisions for a quieter and healthier environment. By harnessing IoT, this app provides a cost-e ective and scalable way to monitor noise levels, contributing to better urban planning, improved public health, and enhanced quality of life for residents. The future holds great potential for such applications as they continue to evolve and make our cities more peaceful and sustainable.