Project Proposal Submitted

To

IBM NAAN MUDHALVAN

INTERNET OF THINGS

Submitted by

Niraimathi S - 911721106021
Palanibharathi V - 911721106023
Prabakaran R - 911721106024
Prasanth K - 911721106027
Rahul Prasanth A - 911721106028

Noise Pollution Monitoring

Project Tittle: NOISE POLLUTION MONITERING

PHASE 3:loading and preprocessing of dataset

Loading and preprocessing data is a crucial step in preparing data for machine learning tasks. This process involves loading the dataset into memory, cleaning and organizing the data,handling missing values,future engineering,and sometimes splitting the data into training and testing sets.

- **Introduction:**

Noise pollution is defined as any disturbing noise that affect humans or wildlife. Although noise constantly surrounds us, noise pollution generally receives less attention than, for example, water quality and air quality concerns, because it cannot be seen, tasted or smelled. There are two ways of dealing noise pollution in terms of There are two ways of dealing noise pollution in terms of reducing its levels. One is more traditional approach and other a more modern and very popular today with many directions. This are all theintroduction about the project of noise pollution.

1. **Database Selection:**
   - Choose a suitable database system for storing your noise pollution data. You can use relational databases (e.g., MySQL, PostgreSQL) or NoSQL databases (e.g., MongoDB) depending on your data structure and requirements.

2. **Database Schema:**
   - Define the structure of your database by creating tables or collections to store different types of data, such as sensor readings, timestamps, location information, and any other relevant metadata.

3. **IoT Device Setup:**
   - Set up your IoT devices with noise sensors and ensure they can collect data. These devices should be capable of sending data to your chosen database.

4. **Data Collection:**
   - Write code on your IoT devices to collect noise data and transmit it to your database. This code should include data formatting and validation.

5. **Data Transmission:**
   - Establish a secure connection between your IoT devices and the database. You may use IoT protocols like MQTT or HTTP for data transmission.

6. **Data Preprocessing:**
   - Implement preprocessing routines to clean and validate the incoming data. You can filter out noise spikes, calculate averages, and convert data into a consistent format.

7. **Database Integration:**
   - Develop APIs or scripts that allow your IoT devices to interact with the database. Ensure data is stored in the appropriate tables or collections.

8. **Security:**

- Implement security measures to protect your data, including encryption of data in transit and at rest. Use authentication and authorization mechanisms to control access to the database.

9. **Data Storage:**
   - Store the preprocessed data in the designated database tables or collections.

10**. Data Retention:**
    - Define data retention policies to manage how long data is stored in the database. Old data may need to be archived or deleted to save storage space.

11**. Monitoring and Alerting:**
    - Set up monitoring for your IoT devices and the database. Implement alerting mechanisms to detect issues with data collection or database access.

12. Data Analysis and Reporting:
    - Build tools or scripts for analyzing and generating reports from the stored data. This can include identifying noise patterns and trends.

LOADED DATASET:

code for loaded dataset for noise pollution monitoring in IOT:

```
import pandas as pd

# Load the dataset
dataset = pd.read_csv('noise_pollution_dataset.csv')

# Print the first few rows of the dataset
print(dataset.head())
```

EXPLORE DATASET:

```
import pandas as pd

# Load the dataset
dataset = pd.read_csv('noise_pollution_dataset.csv')

# Display the first few rows of the dataset
print(dataset.head())

# Check the dimensions of the dataset
print("Number of rows:", dataset.shape[0])
print("Number of columns:", dataset.shape[1])

# Check the data types of each column
print(dataset.dtypes)

# Check for missing values
print("Missing values:")
print(dataset.isnull().sum())

# Check basic statistics of the dataset
print(dataset.describe())

print("Average noise level:", dataset['noise_level'].mean())
print("Maximum noise level:", dataset['noise_level'].max())

# Group data by a specific column
grouped_data = dataset.groupby('location')
print(grouped_data.mean())

# Visualize the data using plots or charts
import matplotlib.pyplot as plt

# Histogram of noise levels
plt.hist(dataset['noise_level'], bins=10)
plt.xlabel('Noise Level')
plt.ylabel('Frequency')
plt.title('Distribution of Noise Levels')
plt.show()

# Boxplot of noise levels by location
dataset.boxplot(column='noise_level', by='location')
plt.xlabel('Location')
plt.ylabel('Noise Level')
plt.title('Noise Levels by Location')
plt.show()
```

CODE FOR ANALYSIS THE NOISE POLLUTION MONITORING IN IOT:

```
import RPi.GPIO as GPIO
import time
import requests

# Set GPIO pin for noise sensor
NOISE_SENSOR_PIN = 18

# Set API endpoint for sending noise data
API_ENDPOINT = "http://example.com/noise-data"

# Setup GPIO
GPIO.setmode(GPIO.BCM)
GPIO.setup(NOISE_SENSOR_PIN, GPIO.IN)

def send_noise_data(data):
# Prepare data payload
payload = {"noise_level": data}

# Send POST request to API endpoint
response = requests.post(url=API_ENDPOINT, json=payload)

# Check response status
if response.status_code == 200:
print("Noise data sent successfully")
else:
print("Failed to send noise data")
```

CONCLUSION:

In conclusion, the development part 1 of the noise pollution monitoring system in IoT has focused on the design and implementation of the hardware components. This includes the selection and integration of sensors, microcontrollers, and communication modules to collect and transmit noise data. The system has been successfully developed and tested, demonstrating its ability to accurately measure and monitor noise levels in real-time. The next step in the development process will involve the integration of software components to process and analyze the collected data, as well as the implementation of a user interface for data visualization and control. Overall, the development part 1 has laid a solid foundation for the successful implementation of the noise pollution monitoring system in IoT.