NOISE POLLUTION MONITORING

Step 1: Setting up the Hardware Components Gather Components:

Collect all the hardware components you need, such as the NodeMCU board, DFRobot Analog Sound Level Meter sensor, OLED display, LEDs, and 220-ohm resistors. Connect Hardware: Follow the wiring instructions provided in your project documentation to connect the components to the NodeMCU board on a breadboard. Ensure all connections are secure and correctly made.

Step 2: Programming the Sensor Node (NodeMCU) Install Arduino IDE:

If not already installed, download and install the Arduino IDE on your computer. Board Configuration:

Configure the Arduino IDE to work with the NodeMCU board by installing the ESP8266 board support. Code Development:

Write the Arduino code for the NodeMCU board. This code should include the following functionality: Reading data from the DFRobot Analog Sound sensor. Converting the analog sensor data to decibels with 'A' weighting (dB(A)). Implementing the traffic signal control logic, including countdown timers.

Adding logic to extend the RED signal time if excessive noise is detected during the RED signal phase.

Displaying countdown timers on the OLED screen. Data Transmission:

Code the logic to transmit the collected data (noise level, signal status, timer values) to an OM2M Eclipse server. You'll need to use libraries and APIs to facilitate this communication.

Step 3: Preparing the Software Environment Install Java 1.8:

Download and install Java 1.8 on your computer if it's not already installed. This version is required to run the OneM2M platform.

Install Eclipse OM2M: Download and install Eclipse OM2M v1.4.1, which is the implementation of the OneM2M standard used in your project.

Step 4: Setting Up the Data Handling and Processing Flow Implement

Django Server: Develop a Django server to handle data received from the OM2M server. This involves creating APIs and routes for data ingestion. Database Setup: Set up a PostgreSQL database to store the data received from the Django server. Design an appropriate database schema to structure the data. Data Transmission Configuration: Configure the OM2M server to send data to the Django server via HTTP requests or another suitable protocol. Ensure the data transmission is secured and reliable.

Step 5: Data Visualization and Analytics NodeJS Installation:

Install NodeJS v14 on your server to run the Django server. Postman Testing: Use Postman to test the APIs and data transmission between the OM2M server and Django server.

Grafana Setup: Install Grafana v9.2 or a later version for data visualization and analytics. Create a dashboard in Grafana to display real-time and historical noise data, traffic signal status, and countdown timer values.

Database Integration: Configure Grafana to connect to the PostgreSQL database to fetch and display data.

Step 6: Testing and Validation System Testing:

Test the complete system to ensure the sensor node accurately measures noise levels and controls the traffic signal as intended. Data Transmission Testing: Verify the data transmission between the sensor node, OM2M server, Django server, and PostgreSQL database. Grafana Dashboard Validation: Check the Grafana dashboard for real-time analytics and historical data trends.

Step 7: Deployment Node Deployment:

Deploy the sensor nodes at high-density traffic junctions, ensuring they are securely installed and properly powered.

Server Deployment: Deploy the OM2M server, Django server, and the PostgreSQL database on reliable and secure servers

. Monitoring: Continuously monitor the system for performance, data accuracy, and reliability in realworld traffic conditions. By following these detailed steps, you can effectively implement your IoT project to monitor noise pollution at traffic junctions and control traffic signals to discourage unnecessary honking by commuters.

Thorough testing and validation are critical to ensure the system's effectiveness and reliability in reducing noise pollution.

Conclusion:

In addressing the problem of noise pollution at high-density traffic junctions, the IoT project outlined here provides a comprehensive solution that leverages hardware components, software platforms, and data processing to not only monitor noise levels but also encourage responsible behavior among commuters. The project has several key components and stages that, when executed successfully, can significantly contribute to reducing the detrimental effects of noise pollution on human health.

Key Achievements and Takeaways:

Hardware Integration:

The project begins with the setup of hardware components, including the NodeMCU board, DFRobot Analog Sound Level Meter sensor, and OLED display. These components work in harmony to capture real-time noise data at traffic junctions.

Software and Platform Implementation:

The integration of software components, including the use of Arduino IDE for sensor node programming and Eclipse OM2M for data transmission, is a critical part of the solution. Java and NodeJS are used to manage servers, while Django and PostgreSQL form the backbone of data handling and processing.

Traffic Signal Control Logic:

The implementation of traffic signal control logic, based on noise levels and timers, aims to create awareness and discourage unnecessary honking. The extension of the RED signal phase when excessive noise is detected is an innovative approach to incentivize quieter behavior

Data Handling and Analytics:

The project sets up a robust data flow, ensuring data is processed efficiently. Data from the sensor nodes is transmitted to the OM2M server, then to the Django server and stored in a PostgreSQL database. Grafana is employed for data visualization and analytics, providing valuable insights into noise patterns and traffic behavior.

Testing and Validation:

Rigorous testing and validation are essential to ensure the system functions as intended. This includes not only hardware and software testing but also checking the real-world effectiveness of the noise reduction strategy.

Deployment and Real-World Impact:

Once validated, the system can be deployed at high-density traffic junctions. By monitoring noise levels and encouraging responsible behavior through traffic signal control, the project can have a positive impact on noise pollution levels in urban areas.

Challenges and Future Developments:

While this project offers a promising solution to noise pollution, it is not without its challenges. It requires careful maintenance and monitoring, as well as addressing potential technical issues that may arise.

Future developments

in this project might include the integration of advanced machine learning algorithms for noise pattern recognition, which could provide insights for better traffic management and enforcement of noise regulations.

In conclusion

the IoT project addressing noise pollution at traffic junctions is a practical and innovative approach to reducing noise pollution and promoting healthier urban environments. Its success depends on the seamless integration of hardware and software components, rigorous testing, and responsible deployment, with the ultimate aim of fostering awareness and behavior change among commuters.

Submitted By:

Mentor : S.Abikayil Aarthi AP/CSE

Team members

AANDAL SA: 821121104002

KEERTHANA J: 821121104026

ANEES PRIYANKA V:821121104004

SHALINI K: 821121104050

GAYATHIRI S: 821121104014