IoT BASED NOISE POLLUTION MONITORING SYSTEM

Name:PRADEESH.M

PROJECT DEFINITION

Noise pollution:

It is considered to be an any unwanted or disturbing sounds that affect the human health and other organisms.

Noise pollution monitoring :

It is a process of measure the magnitude of noise in industries residential areas.

In noise pollution can be monitoring in two different ways like

Instaneous monitoring

Continuous monitoring.

DESIGN THINKING

Objectives of noise pollution monitoring:

1. *Real-time Noise Pollution Monitoring*:

Sensor Network:

Establish a network of noise sensors strategically placed in urban and industrial areas.

- *Data Collection*:

Continuously collect noise data from these sensors and transmit it to a central database in real-time.

- *Data Analysis*:

Develop algorithms to analyze the noise data, identifying trends, hotspots, and potential violations of noise regulations.

- *Alert System*:

Implement an alert system that notifies relevant authorities and the public when noise levels exceed acceptable limits, allowing for immediate action.

2. *Public Awareness*:

- *Educational Campaigns*:

Organize workshops, seminars, and public awareness campaigns to inform residents about the health and social impacts of noise pollution.

- *Noise Reduction Tips*:

Provide practical tips for individuals and businesses on how to reduce noise emissions and be considerate of their neighbors.

- *Interactive Platforms*:

Create interactive online platforms and mobile apps that allow the public to access noise data and report noise complaints.

3. *Noise Regulation Compliance*:

- *Strengthen Regulations*:

Review and update noise regulations to ensure they align with current noise levels and standards.

- *Enforcement Measures*:

Increase the capacity for noise regulation enforcement, including trained personnel and monitoring equipment.

- *Penalties*:

Impose penalties and fines for non-compliance, which can act as a deterrent for noise violations.

4. *Improved Quality of Life*:

- *Noise Mitigation*:

Identify major noise sources through data analysis and work with industries to implement noise-reducing technologies and practices.

- *Urban Planning*:

Integrate noise reduction measures into urban planning and zoning, such as creating buffer zones between noisy industries and residential areas.

- *Community Involvement*:

Involve communities in noise reduction initiatives, seeking their input and collaboration to find practical solutions.

IOT SENSOR DESIGN:

Components required to build noise pollution monitoring :

- ESP32
- SOUND SENSOR ARDUINO
- BLYNK PLATFORM

SOUND SENSOR:

A sound sensor is an electronic device that detects sound waves, converting them into electrical signals, often used for applications like voice recognition, security systems, and noise monitoring.

ESP32:

ESP32 is a series of low-cost, low-power system on a chip microcontrollers with integrated Wi-Fi and dual-mode Bluetooth.

BLYNK PLATFORM:

Blynk is an IoT platform for remote device control, sensor data monitoring, alerts, and custom projects, simplifying IoT development and enabling home automation, data logging, and energy management.

Working of noise pollution moinitoring system:

.

Noise Sensors: The system starts with noise sensors strategically placed in the environment where noise pollution needs to be monitored. These sensors can be microphones or sound level meters that are capable of capturing audio data.

.
.

Data Acquisition: The noise sensors continuously capture audio data, measuring sound levels in decibels (dB). This data may include information about the frequency and intensity of noise.

.
.

Data Processing: The captured audio data is processed locally on the sensor or transmitted to a gateway device for initial processing. This processing might involve filtering, data compression, or feature extraction to reduce the amount of data sent to the cloud.

.
.

Data Transmission: Processed data is then transmitted to a central cloud-based platform using IoT communication protocols, such as Wi-Fi, cellular networks, or LoRaWAN, depending on the deployment location and requirements.

.
.

Cloud-Based Platform: The data is received and stored in a cloud-based platform or server. This platform can be managed by a local authority, environmental agency, or a private organization responsible for noise monitoring.

.
.

Data Analysis and Visualization: The cloud platform processes and analyzes the data in real-time. It can generate noise level statistics, create noise maps, and identify patterns or trends. Visualization tools and dashboards can present the data in a user-friendly format for monitoring and analysis.

Alerts and Notifications: The system can be configured to send alerts and notifications when noise levels exceed predefined thresholds or when unusual noise events occur. These alerts can be sent via email, SMS, or push notifications to relevant stakeholders or authorities.

.
.

Historical Data Storage: Historical noise data is stored in a database, allowing for trend analysis, compliance reporting, and long-term monitoring of noise pollution.

.
.

User Access: Authorized users, such as environmental agencies, city planners, or the public, can access the real-time noise data and historical records through a web-based or mobile application. This access helps in informed decision-making and public awareness.

.
.

Data Sharing: In some cases, data can be shared with the public or integrated into open data initiatives, providing transparency and encouraging community engagement.

.
.

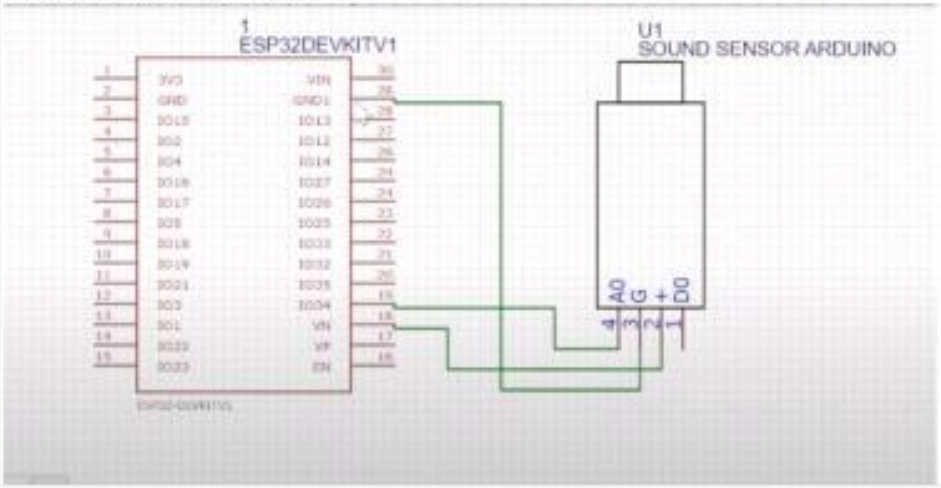
Maintenance and Calibration: Regular maintenance and calibration of the noise sensors are essential to ensure accurate data collection and compliance with standards.

.
.

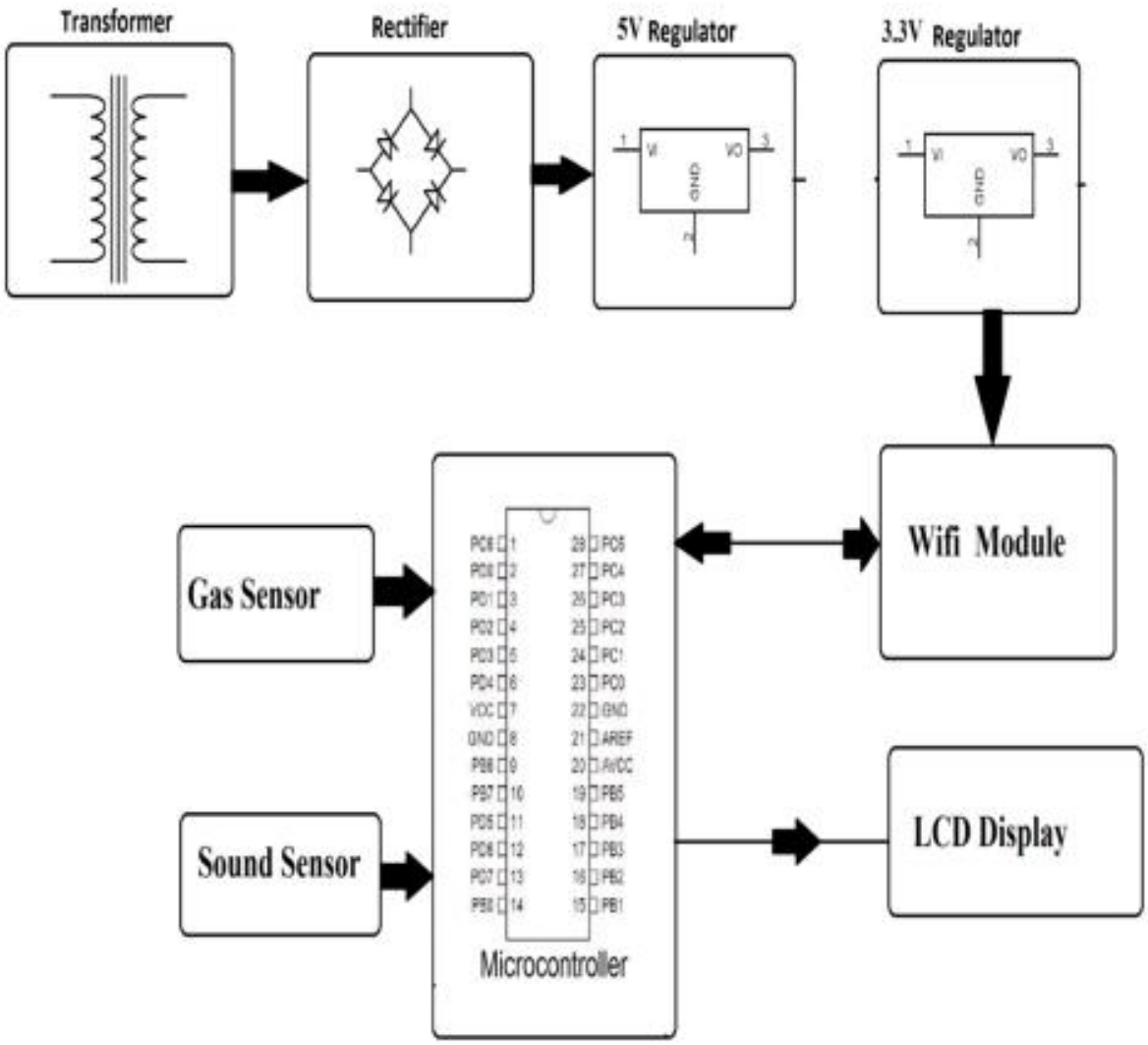
An IoT-based noise pollution monitoring system continuously collects, processes, and analyzes noise data from distributed sensors, enabling real-time monitoring and data-driven decision-making to address and mitigate noise pollution in urban and industrial environments.

:

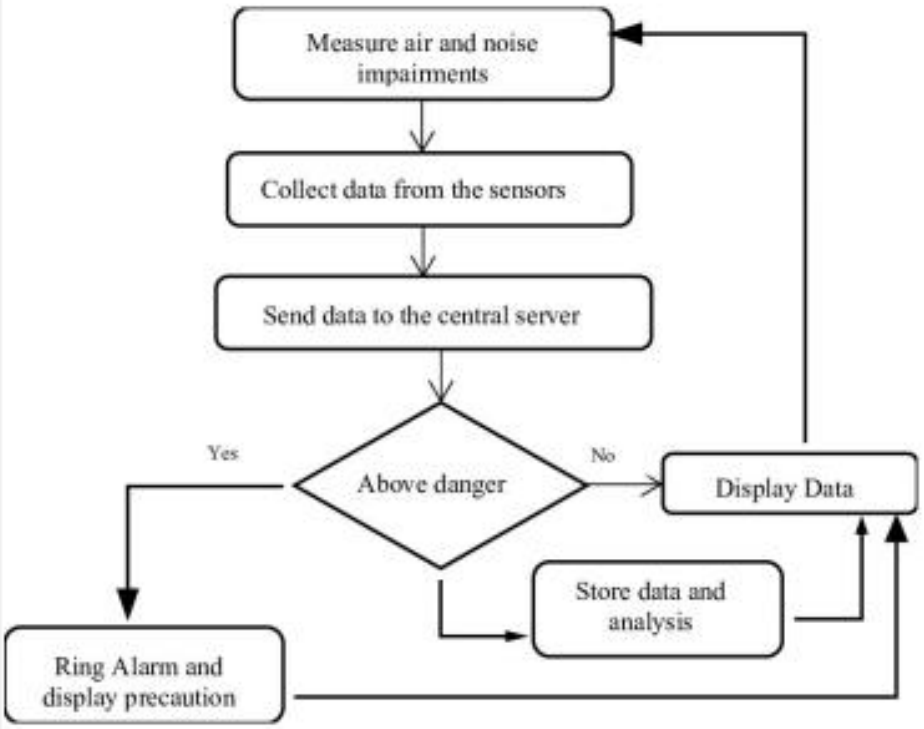
CIRCUIT DIAGRAM:



BLOCK DIAGRAM FOR CIRCUIT DIAGRAM:



FLOW CHART:



DISPLAY :