NOISE POLLUTION MONITORING

PHASE 5

**TABLE OF CONTENTS**

1.INRODUCTION AND DESIGN THINKING

2.WORKING

3.APPLICATIONS

3.PYTHON SCRIPT

4.CLOUD APP DEVELOPMENT

TEAM MEMBERS:

KRITHIKA S-71772114121

MONISHA R-71772114125

NITHILA K-71772114127

POORTHIKA M-71772114131

**IoT BASED NOISE POLLUTION MONITORING SYSTEM**

**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.

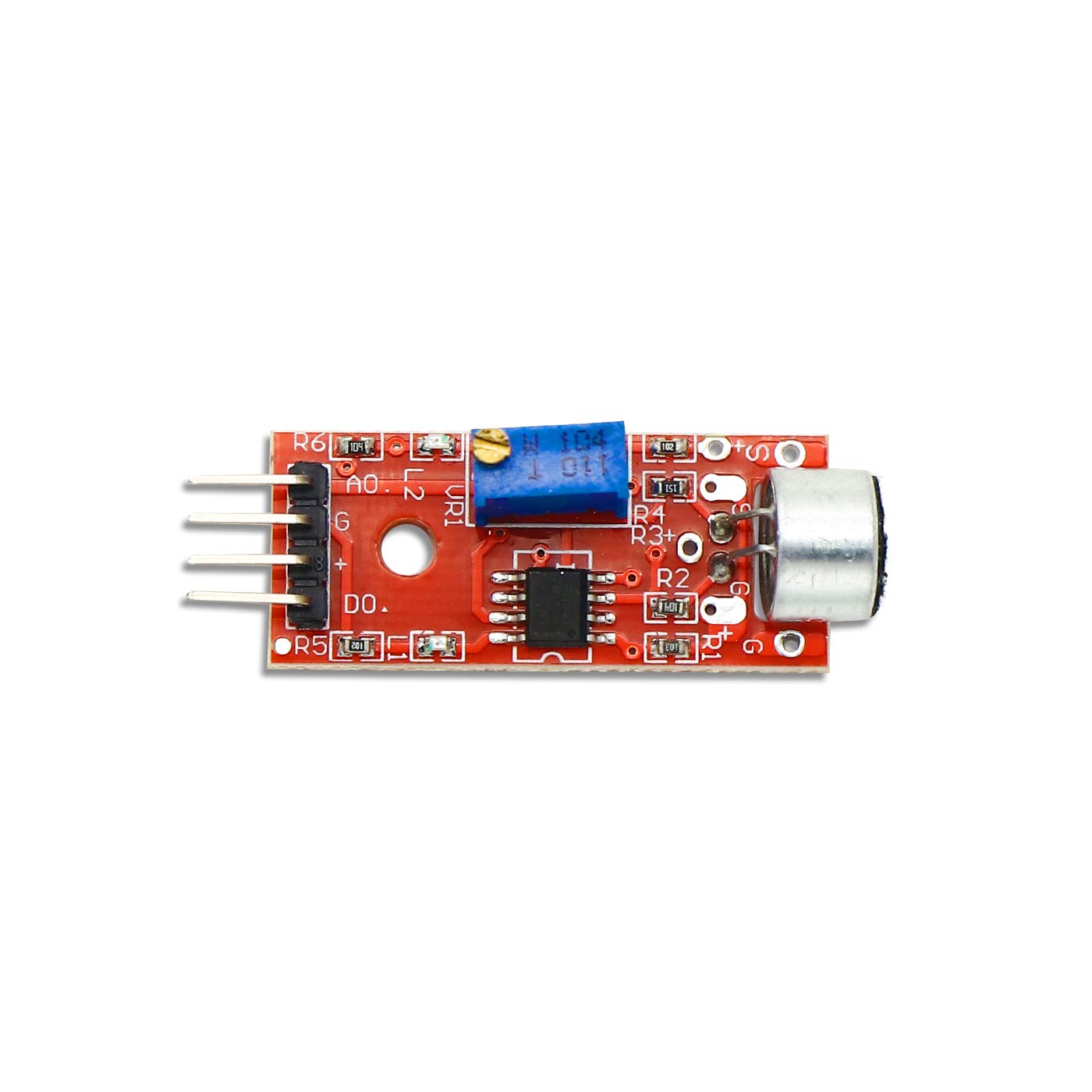
**IOT SENSOR DESIGN:**

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.

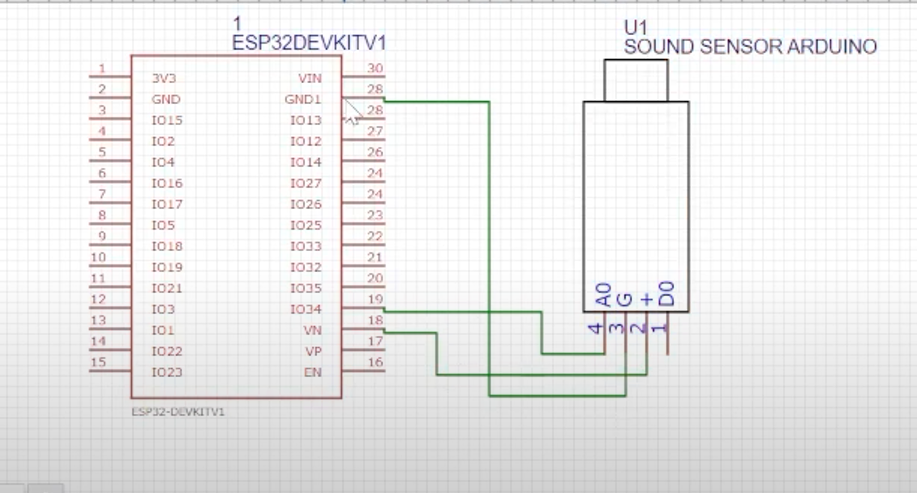
**Data Sharing**: In some cases, data can be shared with the public or integrated into open data initiatives, providing transparency and encouraging community engagement.

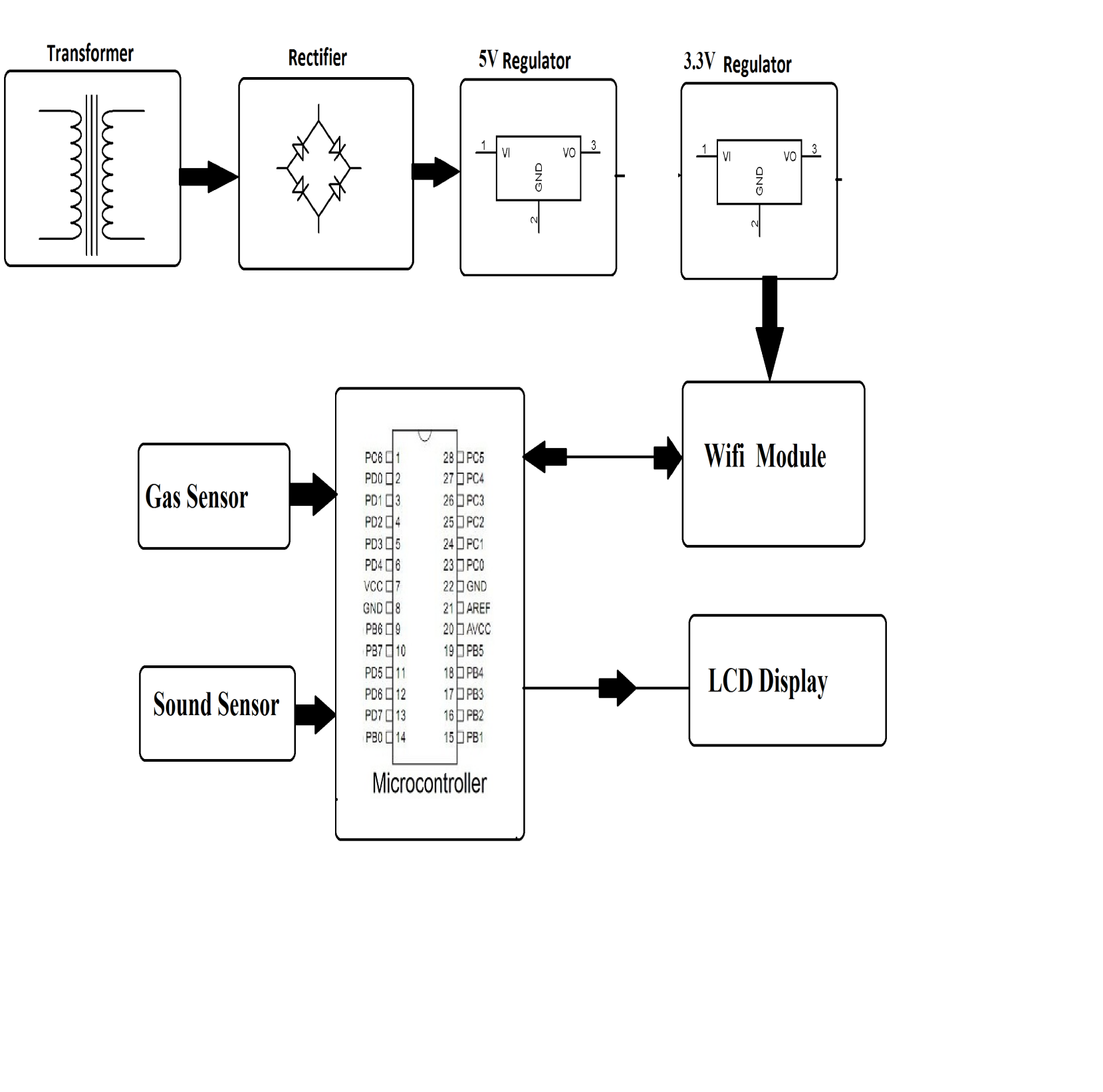
**Maintenance and Calibration**: Regular maintenance and calibration of the noise sensors are essential to ensure accurate data collection and compliance with standards.

**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.

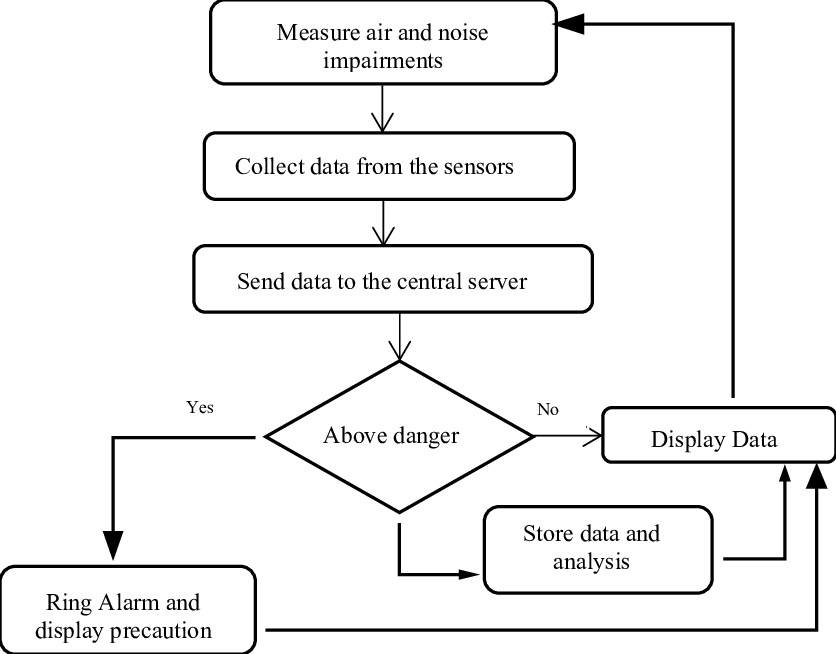
:

**CIRCUIT DIAGRAM:**



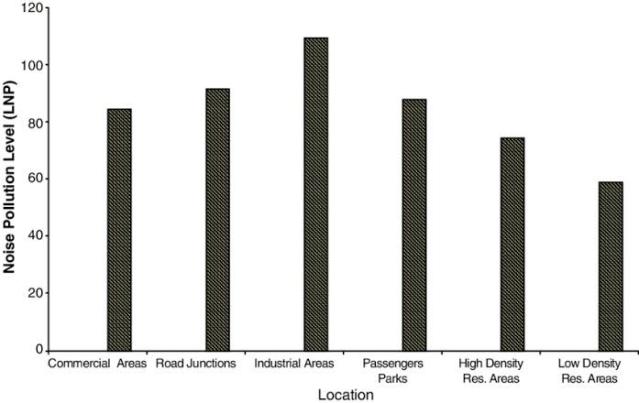
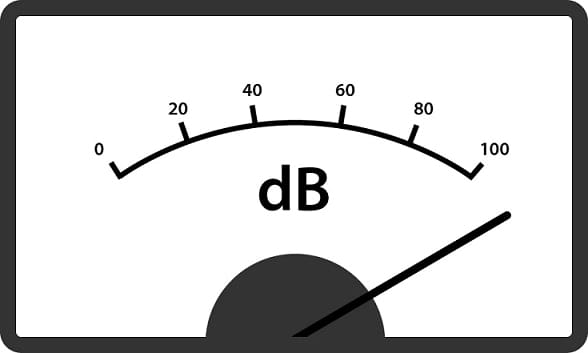
**BLOCK DIAGRAM FOR CIRCUIT DIAGRAM:** 

**FLOW CHART:**



DISPLAY :

窗体顶端

** **

WORKING OF THE PROJECT

* IoT-based noise pollution monitoring uses sensors to measure noise levels, sends data to the cloud, analyzes it, and provides real-time

insights for managing and reducing noise

pollution in urban areas and industrial zones.

* In this diagram we are using ESP32 and sound sensor for IOT based noise pollution

monitoring.

* Sound sensor has 4 pins one is A0,Ground,VCC and D0
* Where A0 is basically analog output and its ground.
* Where VCC is the +5v volt and this one is digital output.
* We can see the analog output of sound sensor is connected to the pin number of 34 inn ESP32 port and ground is connected to the ground of ESP32 port and where the plus VCC is connected to the Vin pin of the ESP32 port.
* We not see the digital output because we want to see analog output given by the sound sensor.
* This is simple circuit diagram for IOT based noise pollution monitoring.

**IDENTIFY NOISE POLLUTION PATTERNS:**

Certainly, incorporating data analytics into your IoT-based noise pollution monitoring system can provide valuable insights and help identify noise pollution patterns. Here's how to integrate data analytics into your project:

1. **Data Collection and Storage**:

Ensure that the noise data collected by your sensors is stored in a structured format, preferably in a database or cloud storage.

1. **Data Preprocessing**:
   * Clean and preprocess the data to handle missing values or outliers.
   * Normalize or standardize the data to make it suitable for analysis.
2. **Data Analytics Tools:**
   * Choose appropriate data analytics tools and libraries such as Python with

libraries like pandas, NumPy, and scikit- learn for analysis.

* + Use specialized noise pollution analysis algorithms or develop custom

algorithms for your specific needs.

1. **Pattern Recognition:**
   * Apply data analytics techniques to recognize patterns and trends in noise pollution data.
   * Identify recurring noise sources, high-noise time periods, or locations with consistent noise problems.
2. **Statistical Analysis:**

Perform statistical analysis to calculate

descriptive statistics, such as mean noise levels, standard deviations, and percentiles, for different time intervals or locations.

1. **Machine Learning**:
   * Consider using machine learning models, such as clustering or time series analysis, to detect patterns or anomalies in the data.
   * Train models to predict noise pollution trends based on historical data.
2. **Visualization:**
   * Create visualizations, such as heatmaps, line charts, or scatter plots, to present noise

pollution patterns effectively.

* + Use tools like Matplotlib or Plotly to generate interactive plots for better understanding.

1. **Alerts and Reporting**:

Implement automated alerts and reporting mechanisms to notify relevant authorities or stakeholders when specific noise pollution patterns exceed acceptable limits.

1. **Continuous Improvement:**
   * Continuously monitor and evaluate the performance of your data analytics

methods.

* + Refine your analytics approach based on new data and feedback to improve pattern recognition accuracy.
  + By incorporating data analytics, your IoT- based noise pollution monitoring system can not only provide real-time information but also offer valuable insights into noise

patterns and trends. This can assist in proactive noise pollution management,

urban planning, and policy-making to create quieter and healthier environments.

**HIGH NOISE AREAS:**

High noise areas are locations or regions where noise levels consistently exceed acceptable limits, causing disturbances and potentially harming the health and well-being of residents or occupants.

These areas are often characterized by elevated noise pollution due to various sources such as:

1**Urban Centers:**

Busy urban areas with heavy traffic,

construction activities, industrial zones, and

entertainment venues like bars and nightclubs tend to have high noise levels.

**Transportation Hubs:**

Places near airports, train stations, bus terminals, and major highways can experience high noise pollution from

vehicular traffic and transportation-related activities.

**Industrial Zones**:

Areas with manufacturing plants, factories, and industrial facilities often produce constant noise from machinery and equipment.

**Commercial Districts:**

Shopping districts and commercial areas with high foot traffic, delivery trucks, and street

vendors can be noisy.

**Construction Sites**:

Construction sites generate significant noise from machinery, drilling, and building activities.

**Recreational Areas:**

Parks, sports stadiums, and concert venues can become high noise areas during events and

gatherings.

**Residential and Commercial Mix:**

Mixed-use areas where residential and commercial properties coexist may experience

noise conflicts, especially if noise regulations are not enforced.

**Transportation Corridors**:

* Roads with heavy traffic and limited noise- reducing infrastructure can create high noise pollution for nearby residents.
* High noise areas can lead to various negative consequences, including sleep disturbances, increased stress, hearing impairment, and

reduced overall quality of life for residents.

Identifying and addressing high noise areas often involves implementing noise-reduction measures, noise barriers, stricter regulations, and urban planning strategies to mitigate

noise pollution and improve the living conditions in affected regions.

**POTENTIAL SOURCES**:

Potential sources of noise pollution in both urban and rural areas can vary widely and may include:

1. **Traffic Noise:**
   * Cars, trucks, motorcycles, and public transportation.
   * Honking horns, engine noise, and tire squeals.
2. **Industrial Activities:**
   * Manufacturing plants, factories, and construction sites.
   * Machinery, equipment, and industrial processes.
3. **Aircraft Noise:**
   * Airports and flight paths.
   * Jet engines, takeoffs, and landings.
4. **Construction and Demolition**:
   * Construction machinery and equipment.
   * Drilling, hammering, and construction- related activities.
5. **Recreational and Entertainment:**
   * Concerts, sports events, and stadiums.
   * Music venues, nightclubs, and loud gatherings.
6. **Public Infrastructure:**
   * Road maintenance, repair work, and street cleaning.
   * Traffic management and public transportation.
7. **Commercial Activities**:
   * Retail stores, restaurants, and bars.
   * Delivery trucks, HVAC systems, and refrigeration units.
8. **Public Announcements:**
   * Public address systems, sirens, and alarms.
   * Emergency vehicle sirens and announcements.
9. **Outdoor Equipment:**
   * Lawnmowers, leaf blowers, and power tools.
   * Agricultural machinery in rural areas.
10. **Trains and Railways:**
    * Train stations and railway tracks.
    * Train whistles, locomotive engines, and track maintenance.
11. **Neighbors and Residential Activities:**
    * Loud music, conversations, and domestic activities.
    * Barking dogs and other pets.
12. **Natural Events:**
    * Thunderstorms, strong winds, and geological activities.
    * These can produce loud noises, although they are not human-generated.
    * Identifying and mitigating noise pollution often involve understanding the specific sources prevalent in a given area and

implementing measures such as noise barriers, soundproofing, urban planning, and noise regulations to reduce the impact on the environment and human health.

**PYTHON SCRIPT:**

import time

import machine

import network

import urequests

*# Configuration*

WIFI\_SSID = "wifi name"

WIFI\_PASSWORD = "WiFi\_Password"

NOISE\_API\_URL = "https://noise-platform-url.com/api/noise-data"

API\_KEY = "api-key"

*# Initialize Wi-Fi*

sta = network.WLAN(network.STA\_IF)

sta.active(True)

sta.connect(WIFI\_SSID, WIFI\_PASSWORD)

*# Wait for Wi-Fi connection*

while not sta.isconnected():

pass

print("Connected to Wi-Fi")

*# Initialize ADC for the microphone sensor*

adc = machine.ADC(0) *# ADC pin may vary depending on your ESP32 board*

*# Function to measure noise level*

def measure\_noise\_level():

adc\_value = adc.read() *# Read analog value from microphone sensor*

*# Implement calibration and noise level calculation here*

*# For demonstration purposes, we'll use a placeholder value*

noise\_level = adc\_value

return noise\_level

*# Main loop for real-time monitoring and data transmission*

while True:

try:

noise\_level = measure\_noise\_level()

*# Send noise data to the platform*

data = {"noise\_level": noise\_level, "location": "location-info"}

headers = {"Authorization": "Bearer " + API\_KEY}

response = urequests.post(NOISE\_API\_URL, json=data, headers=headers)

if response.status\_code == 200:

print(f"Data sent successfully: {noise\_level}")

else:

print(f"Failed to send data. Status code: {response.status\_code}")

response.close()

*# Adjust the sampling interval as needed*

time.sleep(10)

except KeyboardInterrupt:

break

**OUTPUT OF THE ABOVE PROGRAM:**

* The provided python script for the ESP32 is designed to capture noise level data from a sound sensor and send it to a noise pollution information platform. The program’s output will typically be displayed in the Micro Python REPL(Reas-Eval-Print Loop) or, if you run it as a standalone script, It may not show any output on the device itself.

Here's what you can expect to see in the Micro Python REPL If you have a serial connection to your ESP32:

1.initial message regarding WI-FI connection:

***Connected to Wi-Fi***

2.Real-time output of data being send to the platform:

***Data sent successfully: {noise\_level}***

If the data is successfully sent to the platform, you’ll see this message with the measured noise level value. The actual noise level value will depend on the sensor and calibration in your setup.

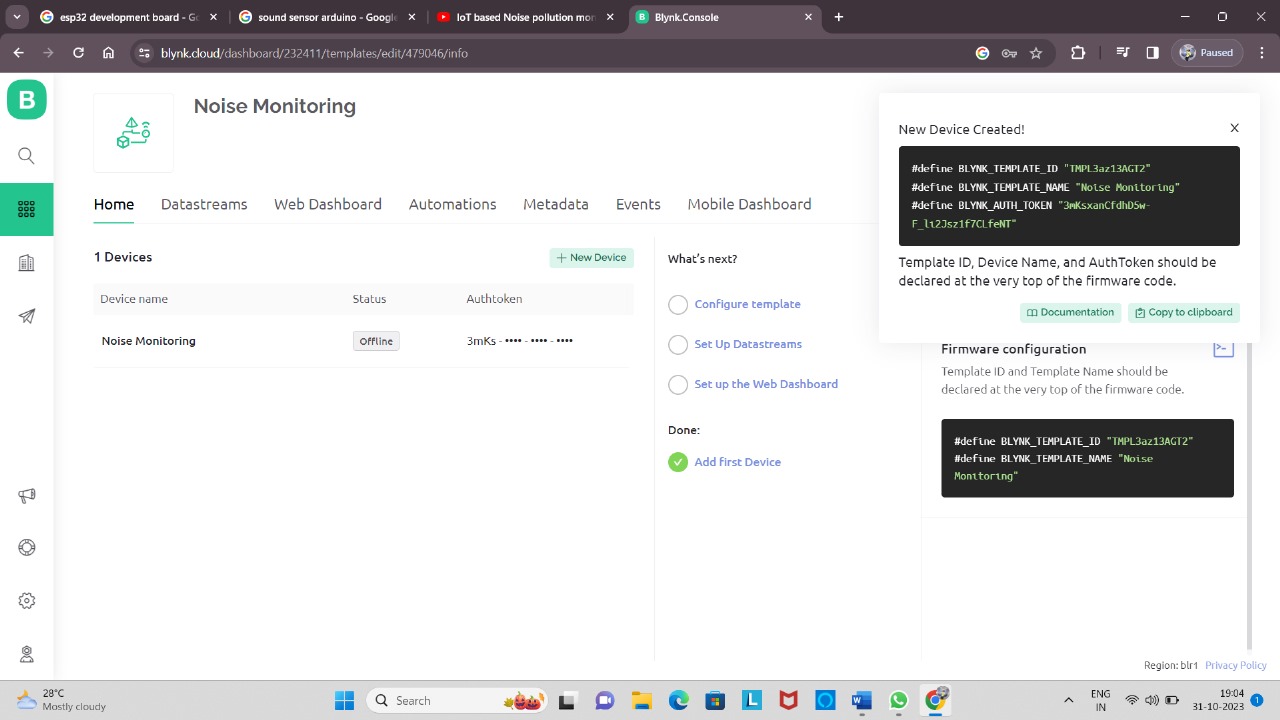
3.Error message (if any) when data transmission fails:

***Failed to send data. Status code: {status\_code}***

If there’s an issue with sending data to the platform(e.g., a network problem or incorrect URL/credentials).you’ll see an error message with the HTTP status code indicating the failure.

The exact output may vary depending on your hardware, network setup, and how you run the script. If you are not seeing any output, you can add print statements for debugging purposes to check the flow of the script or any potential errors.

**1)BLYNK IOT PLATFORM :**

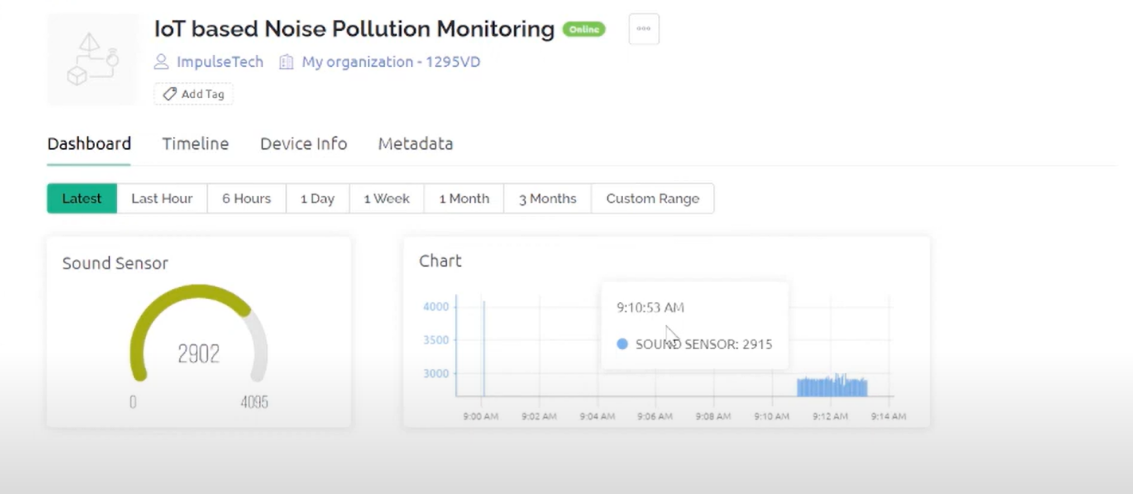


**Link:** <https://blynk.cloud/dashboard/232411/templates/edit/479046/info>

***Steps to create BLYNK:***

* First to sign up the BLYNK cloud.
* Click new template and create the template as well as give the project name, devices and connection.
* Create data streams, web dashboard, automations, metadata, events, mobile dashboard.
* Add the program for the circuit and the see the output.
* This website is used to detect the noise level at different frequencies level.

*Output of the noise level:*



**2)MOBILE PLATFORM:**

HTML (Hypertext Markup Language) is used for structuring web content, CSS (Cascading Style Sheets) for styling, and JavaScript for interactivity, creating dynamic, user-friendly web applications and websites.

HTML CODE:

<!DOCTYPE html>

<html>

<head>

<title>Noise Level Monitoring</title>

<style>

/\* Your CSS styles here \*/

</style>

</head>

<body>

<h1>Noise Level Monitoring</h1>

<div id="noise-level">Loading...</div>

<script>

function updateNoiseLevel(noiseData) {

document.getElementById('noise-level').innerText = `Noise Level: ${noiseData} dB`;

}

// Use AJAX or WebSockets to fetch data and update the UI

// Example using Fetch API

function fetchData() {

fetch('http://your\_server\_url/data\_endpoint')

.then(response => response.json())

.then(data => updateNoiseLevel(data.noise\_level))

.catch(error => console.error(error));

}

// Update data every 10 seconds (adjust as needed)

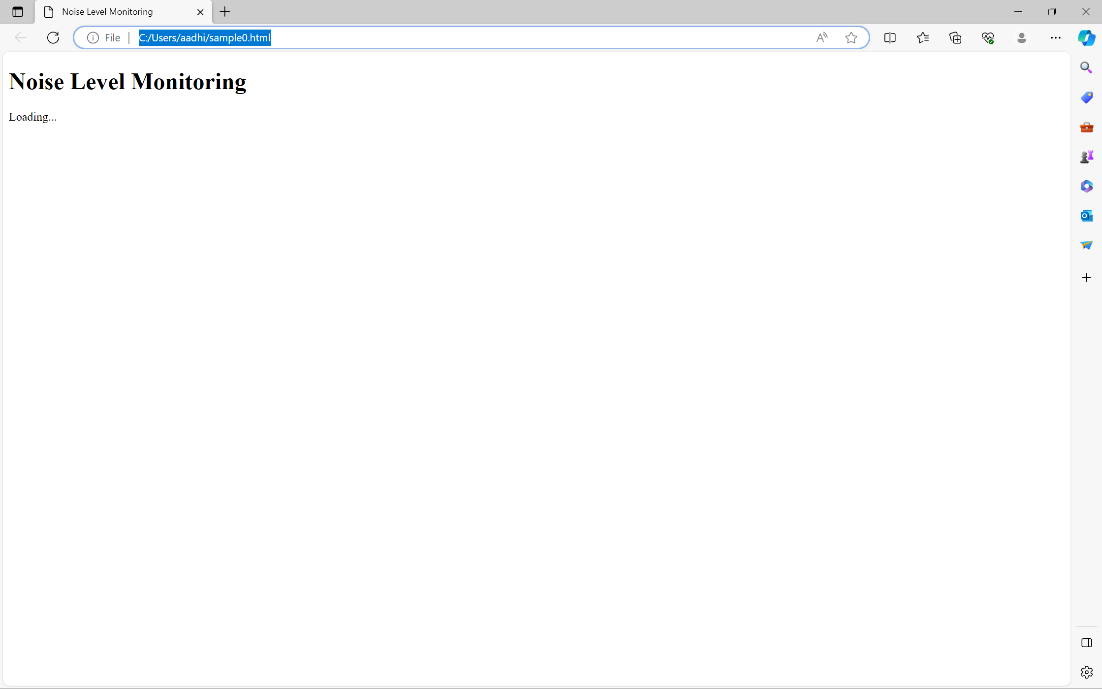
setInterval(fetchData, 10000);

</script>

</body>

</html>

**WEBSITE:**



**DEVICE:**

