NOISE POLLUTION MONITORING USING IoT

1. For a noise pollution monitoring project using IoT, objectives could include realtime data collection, analysis, and providing insights for effective urban planning or public awareness.

2.Additionally, ensuring scalability, accuracy, and low power consumption in the IoT devices would be key goals.

To setup an IoT device ,for noise pollution monitoring, consider using

• a sound sensor (like a microphone),

• a microcontroller (such as Arduino or Raspberry Pi),

• a communication module (like Wi-Fi or GSM).

Connect the sensor to the microcontroller, program it to gather and process sound data, then transmit the data to a central server for analysis.

Ensure power efficiency and robustness for continuous monitoring.

SPECIFICATIONS:

Hardware:

1. Sound Sensor: Captures environmental noise.
2. Microcontroller (e.g., Arduino or Raspberry Pi): Processes and manages data.
3. Communication Module (e.g., Wi-Fi, GSM): Transmits data to a central server.
4. Power Supply: Ensure a reliable and efficient power source.
5. Enclosure: Protects the components from environmental factors.

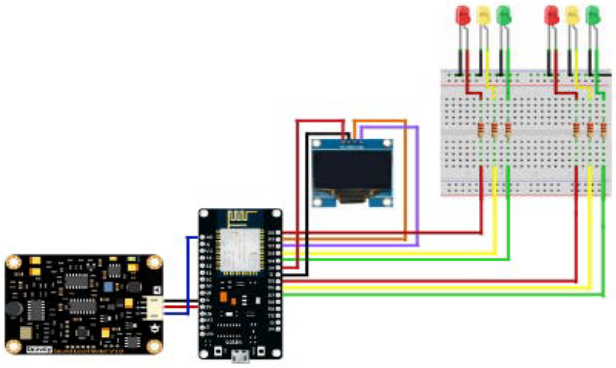
Software:

1. Embedded Software for Microcontroller: Programs the microcontroller for data collection and transmission.
2. Server-side Software: Manages and analyzes the received data.
3. Database: Stores historical noise data for analysis.

4. Web/App Interface: Allows users to access and visualize noise pollution data.

5. Data Analytics Tools: Process and interpret noise data for meaningful insights.

Circuit diagram

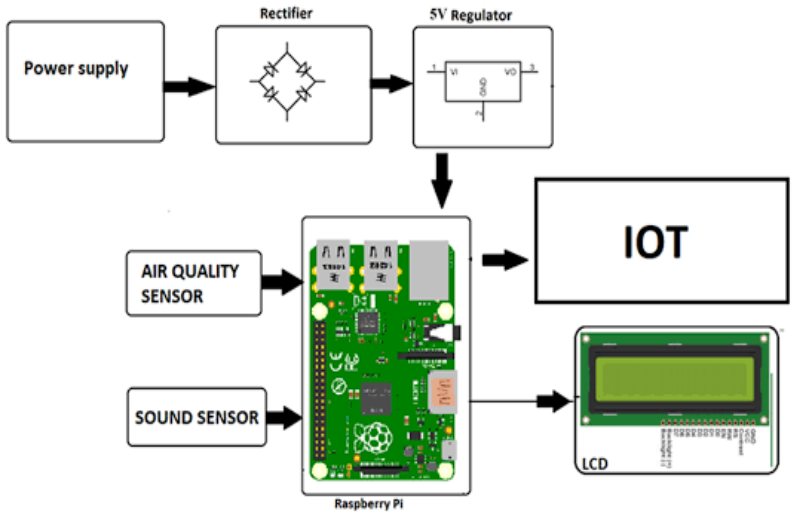


Monitoring noise pollution using IoT (Internet of Things) involves deploying sensors and data collection devices to gather information about noise levels in different locations. Here's how it can be done:

- **Sensor Deployment:** Install noise sensors at various locations you want to monitor. These sensors can be placed in urban areas, near highways, industrial zones, or residential neighborhoods.
- **IoT Connectivity:** Connect the noise sensors to the internet using wireless technologies like Wi-Fi, cellular, or LoRaWAN. This allows the sensors to transmit data in real-time to a central server or cloud platform.
- **Data Collection:** Collect data from the noise sensors continuously. These sensors should measure sound levels in decibels (dB) and record timestamps.
- **Data Storage:** Store the collected data in a secure and scalable cloud database. Cloud platforms like AWS, Azure, or Google Cloud can be used for this purpose.
- **Data Analysis:** Analyze the noise data to identify patterns and trends. You can use machine learning algorithms to detect anomalies or specific noise events that may indicate pollution.
- **Visualization**: Create dashboards or apps to visualize the noise pollution data. This can help both the authorities and the public understand noise levels in real-time or historically.

- **Alerts and Notifications:** Set up automated alerts and notifications for noise levels that exceed predefined thresholds. This allows for timely responses to noise pollution incidents.
- **Data Sharing:** Share the collected data with relevant authorities, environmental agencies, or the public through websites, apps, or APIs to increase awareness and promote transparency.
- **Historical Analysis:** Use historical data to identify noise pollution hotspots, track changes over time, and make informed decisions on urban planning or noise control measures.
- **Regulatory Compliance:** Ensure that the monitoring system complies with local and national regulations regarding noise pollution monitoring and reporting.

Block Diagram:



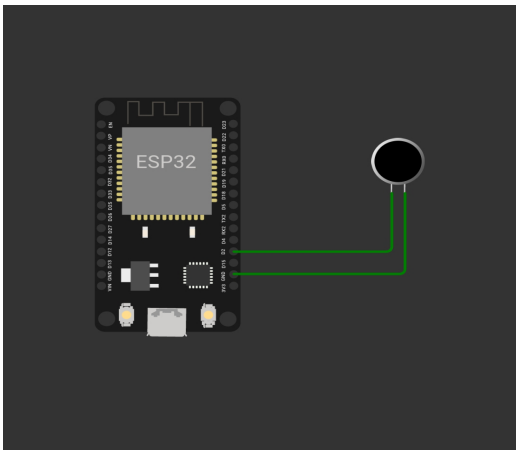
Python code :

Python code for noise pollution monitoring using IoT:

```
from machine import Pin, ADC

from time import sleep
```

```
 pot = ADC(Pin(2))
 pot.atten(ADC.ATTN_11DB)        #Full range: 3.3v
 #ADC.ATTN_0DB: Maximum voltage of 1.2V
#ADC.ATTN_2_5DB: Maximum voltage of 1.5V
#ADC.ATTN_6DB: Maximum voltage of 2.0V
 #ADC.ATTN_11DB: Maximum voltage of 3.3V
 while True:
    pot_value = pot.read()
    print(pot_value)
    sleep(0.1)
'''
import machine, time
 a = machine.ADC(machine.Pin(32))
 while True:
    sample = a.read()     # we want 16 bits, a.read() return 10 bits
    print(sample)
    time.sleep(1/44100)
```



Developing platform:

1.Machine Learning for Anomaly Detection:Implement machine learning algorithms to detect unusual patterns or anomalies in noise data, providing more intelligent insights.

2.Mobile App Integration:Develop a mobile app for users to access real-time noise data, receive alerts, and contribute reports, fostering community engagement.

3.Community Feedback Mechanism:Integrate a feature allowing residents to provide feedback on noise issues, creating a more interactive and community-driven system.

4.Historical Data Trends:Create data visualizations and trends to help users understand how noise levels change over time, enabling better long-term planning.

5.Crowdsourced Noise Mapping:Allow users to contribute noise data through their smartphones, creating a crowdsourced map of noise pollution in different areas.

Creating a noise pollution monitoring system using IoT and web technology involves deploying sound sensors in target areas. These sensors collect data, which is then transmitted to a central server via IoT protocols. A web interface can be developed to visualize and analyze the noise levels in real-time, providing valuable insights for better urban planning and environmental management.

HTML CODE:

```
<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="UTF-8">

<meta name="viewport" content="width=device-width, initial-scale=1.0">
```

```
<title>Noise Pollution Monitoring</title>
<style>
/* Add your styles here */
</style>
</head>
<body>
<h1>Noise Pollution Monitoring System</h1>
<div id="noiseData">
<!-- Display noise data dynamically here -->
</div>
<script>
// Use JavaScript to fetch and display noise data
async function fetchNoiseData() {
try {
const response = await fetch('your_api_endpoint_here');
const data = await response.json();
// Update the noiseData div with the fetched data
document.getElementById('noiseData').innerText = JSON.stringify(data, null, 2);
} catch (error) {
console.error('Error fetching noise data:', error)
  }
}
// Fetch data on page load
fetchNoiseData();
```

```
// Optionally, you can set up periodic data refresh using setInterval
// setInterval(fetchNoiseData, 5000); // Refresh every 5 seconds, for example
</script>
</body>
</html>
```

JAVASCRIPT:

```
// Use JavaScript to fetch and display noise data
async function fetchNoiseData() {
try {
const response = await fetch('your_api_endpoint_here');
const data = await response.json();
// Update the noiseData div with the fetched data
const noiseDataDiv = document.getElementById('noiseData');
noiseDataDiv.innerHTML = `<h2>Noise Data</h2><pre>${JSON.stringify(data, null, 2)}</pre>`;
} catch (error) {
console.error('Error fetching noise data:', error);
  }
}
// Fetch data on page load
fetchNoiseData();
// Optionally, you can set up periodic data refresh using setInterval
// setInterval(fetchNoiseData, 5000); // Refresh every 5 seconds, for example
```

In conclusion, this noise pollution monitoring project utilizes IoT and web technologies to create a system that collects, processes, and displays real-time noise data. By deploying sound sensors in targeted areas, the system communicates with a central server, and a web interface allows users to visualize the noise levels dynamically. The combination of HTML, CSS, and JavaScript provides a user-friendly front end,while the server-side code handles data processing and storage. This project contributes to effective urban planning and environmental management by offering insights into noise pollution levels for informed decision making.