Noise Pollution Monitoring Using Internet of Things

Introduction:

Noise pollution, also known as sound pollution, is a significantenvironmental issue that can have adverse effects on human health, wildlife,and overall quality of life. It is caused by excessive and unwanted sound, oftenoriginating from various sources such as industrial activities, transportation,construction, and urban development. To address this problem, monitoring andmanaging noise pollution have become essential, and the Internet of Things(IoT) offers a powerful and innovative solution.Noise pollution refers to the presence of high levels of noise that disrupt the normal acoustic environment. This can lead to a range of problems, including hearing damage, sleep disturbances, and stress-related health issues.
The IoT is a network of interconnected devices that can collect andexchange data over the internet. In the context of noise pollution monitoring,IoT devices play a crucial role in gathering, transmitting, and analyzing data related to sound levels in various locations. The use of IoT technology for noise pollution monitoring is a promising approach to address this critical environmental issue. By providing accurate and real-time data, IoT systems empower governments, organizations, and individuals to take informed actions to reduce noise pollution and create quieter, healthier living environments.

Problems:

Limited noise monitoring: Traditional noise monitoring methods are often limited in scope and accuracy, making it challenging to comprehensively assess noise pollution levels in urban areas or near industrial sites.

Solution: Deploy IoT-based noise sensors and networks to gather real-time noise data across various locations, enabling more precise monitoring and analysis.
Data Overload: IoT-based noise monitoring systems can generate vast amounts of data, making it difficult to process, analyse, and extract meaningful insights.

Solution: Implement data analytics and machine learning algorithms to process and interpret noise data efficiently, identifying trends, patterns, and potential issues.
Privacy Concerns: Noise monitoring using IoT may inadvertently capture private conversations or activities, raising privacy concerns.

Solution: Ensure that noise monitoring systems are designed and configured to focus on environmental noise rather than specific conversations or activities, and implement data anonymization techniques.
Data Accuracy: IoT sensors used for noise monitoring must be calibrated and maintained regularly to ensure accurate and reliable data.

Solution: Implement a calibration and maintenance schedule, with automatic alerts for sensor issues, and conduct periodic quality assurance checks.
Integration with Urban Planning: Noise pollution data collected through IoT should be integrated into urban planning processes to make informed decisions about land use, transportation, and infrastructure development.

Solution: Establish data-sharing mechanisms between IoT noise monitoring systems and urban planning authorities to inform policy and decision-making.
Noise Mitigation: IoT systems can also be used to implement noise mitigation measures, such as adaptive traffic management or noise barriers. However, these solutions may face challenges in implementation and effectiveness.

Solution: Conduct thorough assessments of the effectiveness of noise mitigation measures and adjust them as needed based on real-time data and feedback.
Community Engagement: Engaging with affected communities and stakeholders to address noise pollution concerns and implement appropriate solutions is crucial.

Solution: Foster community involvement through public awareness campaigns, feedback mechanisms, and collaborative decision-making processes.
Energy Consumption: IoT devices used for noise monitoring should be energy-efficient to reduce their carbon footprint.

Solution: Optimize IoT sensor design for low energy consumption, use renewable energy sources where possible, and employ energy-efficient data transmission protocols.
Regulatory Compliance: Ensure that noise monitoring systems and IoT solutions adhere to local and national regulations regarding noise pollution.

Solution: Collaborate with regulatory bodies and stay up-to-date with noise pollution regulations, ensuring that the IoT system complies with legal requirements.
By addressing these problems and defining noise pollution in the context of IoT, urban planners, environmental agencies, and communities can work together to create healthier and more sustainable urban environments with reduced noise pollution levels.

PROJECT OBJECTIVE

Step 1: Define Project Goals and Scope

Determine the objectives of noise pollution monitoring system. Identify the specific areas want to monitor, such as urban neighborhoods, industrial zones, or transportation hubs.

Step 2: Hardware Selection

Choose appropriate noise sensors or microphones capable of capturing audio data. Consider factors like sensitivity, frequency range, and durability. You may also need microcontrollers like Raspberry Pi or Arduino to collect and process data from the sensors.

In this project we use the ESP32 micro controller as well as Arduino UNO microcontroller as both these suit the best for this project.

Step 3: Components used in Project

Sensor Components

1. Microphone: Select a high-quality microphone capable of capturing a wide frequency range to accurately measure sound levels. Electret condenser microphones are commonly used for this purpose.

2. Microcontroller: Choose a microcontroller like Arduino, Raspberry Pi, or an IoT-specific chip (e.g., ESP8266 or ESP32) to process data from the microphone, manage connectivity, and execute control logic.

3. Connectivity Module: Incorporate a connectivity module (e.g., Wi-Fi, Bluetooth, LoRa, or cellular) to transmit data to a central hub or cloud server. The choice of connectivity depends on the range and data transmission requirements of your application.

4. Power Source: Provide a power source, such as a rechargeable battery or a power adapter. For long-term outdoor installations, consider solar panels and battery storage.

5. Enclosure: Design a weatherproof and durable enclosure to protect the sensor from environmental factors like rain, dust, and extreme temperatures.

Sensor Functionality:

1. Audio Data Capture: The microphone captures audio data, converting sound waves into electrical signals.

2. Analog-to-Digital Conversion (ADC): An ADC within the microcontroller converts the analog microphone signals into digital data for processing.

3. Data Processing: Implement algorithms to preprocess the audio data, remove background noise, and calculate noise level measurements (e.g., dB SPL or A-weighted dB).

4. Data Storage: Store processed data locally (if needed) and transmit it to a central server or cloud platform via the chosen connectivity module.

5. Power Management: Implement power management features to optimize energy usage, such as sleep modes or low-power components.

6. Real-time Monitoring: Develop firmware to monitor noise levels in real-time and trigger alerts when predefined thresholds are exceeded.

7. Time StampingInclude a real-time clock (RTC) to timestamp noise events accurately.

Sensor Deployment:

1. Location: Deploy the sensors in strategic locations to capture representative noise data. Consider urban areas, near highways, industrial zones, residential neighborhoods, and other relevant locations.

2. Mounting: Ensure the sensors are securely mounted to reduce vibration and minimize interference from wind or physical disturbances.

Communication:

1. Data Transmission: Use the chosen connectivity module to transmit data to a central server or cloud platform using standard communication protocols (e.g., MQTT, HTTP, or CoAP).

2. Security: Implement encryption and authentication protocols to secure data transmission and protect against unauthorized access.

Power Supply:

1. Battery: If using batteries, choose a suitable battery capacity to ensure the sensor operates for an extended period without frequent replacements or recharging.

2. Solar Option: For outdoor installations, consider integrating solar panels and rechargeable batteries for long-term, eco-friendly power supply.

Data Visualization and Analysis:

1. Server/Cloud: Set up a central server or cloud platform to receive and store data from multiple sensors.

2. Data Analytics: Implement noise pattern analysis and machine learning algorithms to identify noise pollution patterns, high-noise areas, and potential sources.

3. User Interface: Create a user-friendly web dashboard or mobile app for users to access real-time noise data, receive alerts, and view historical trends.

Step 4: Data Collection

Set up the sensors to collect audio data continuously or at predetermined intervals. Design a data collection protocol that ensures data integrity and timestamping for each recording.

Step 5: Data Transmission

If the sensors are distributed over a wide area, use a communication protocol like Wi-Fi, cellular, LoRa, or Zigbeeto transmit data to a central server or cloud platform for storage and analysis.

All transmission tools are used to connectivity purpose.

Step 6: Data Storage

Store the collected audio data in a secure and scalable database or cloud storage system. Ensure data is organized efficiently for easy retrieval and analysis.

Step 7: Data Preprocessing

Preprocess the audio data to remove background noise, normalize audio levels, and convert it into a suitable format for analysis (e.g., WAV or MP3). This step enhances the quality of the data.

Step 8: Feature Extraction

Extract relevant features from the preprocessed audio data, such as sound intensity (dB levels), frequency spectrum, duration, and temporal patterns. These features will serve as inputs for your analytics algorithms.

Step 9: Data Analytics and Machine Learning

Develop data analytics and machine learning models to identify noise pollution patterns, high-noise areas, and potential sources. These models may include clustering algorithms, spectral analysis, or deep learning techniques. Train your models using labeled data if possible.

Step 10: Noise Pattern Identification

Apply the trained analytics models to the feature-extracted data to identify noise pollution patterns and anomalies. This step may involve clustering similar noise events or detecting trends over time.

Step 11: High-Noise Area Mapping

Create visualizations or maps that display high-noise areas based on the analytics results. Use geographical information systems (GIS) tools to overlay noise data onto maps for better visualization.

Step 12: Potential Source Identification

Utilize source localization techniques to pinpoint potential noise sources. This could involve triangulation using multiple sensors or other sound source localization methods.

Step 13: Real-time Monitoring and Alerts

Implement a real-time monitoring system that continuously analyzes incoming audio data. Set up alert mechanisms to notify relevant authorities or individuals when noise levels exceed predefined thresholds.

Step 14: Historical Data Analysis

Analyze historical noise data to identify long-term trends, seasonal variations, and the effectiveness of any noise control measures implemented.

Step 15: User Interface and Reporting

Develop a user-friendly interface like web dashboard or mobile app for users to access noise pollution data, maps, and analytics insights. Provide reports and visualizations for stakeholders.

To develop the Node-RED tool for programming for wiring together hadware device,APIs and online sevices in new and intresing way.IT provides a browser based edito that makes it easy to wire together flows using the wide range of node in the palette that can be deployed its runtime in a single click

Creat a mobile app for user interface using **MIT App inventor.**

Step 16: Deployment and Maintenance

Deploy the IoT noise monitoring system in the target area. Ensure it operates reliably and perform regular maintenance to ensure sensor accuracy and system integrity.

Step 17: Public Engagement

Engage with the community and relevant stakeholders to raise awareness about noise pollution issues and the efforts being made to mitigate them.

Step 18: Smart Home Automation

Clearly define the objectives of your smart home automation system. Determine what you want to achieve with noise pollution monitoring and automation. This could include noise alerts, adjusting home settings based on noise levels, or tracking noise patterns.

By following these steps, you can create a comprehensive IoT-based noise pollution monitoring system that incorporates data analytics to identify patterns, high-noise areas, and potential sources, ultimately contributing to effective noise pollution management and control.

Components used in this project

- ESP8266 NodeMCU Board
- Microphone sensor
- 16*2 LCD Module
- Breadboard
- Connecting wires

ESP8266 NodeMCU Board

The ESP8266 is a **low-cost Wi-Fi microchip, with built-in TCP/IP networking software, and micro controller capability.**

In our project we use this micro controller for network conection.

ESP8266 NodeMCU Board

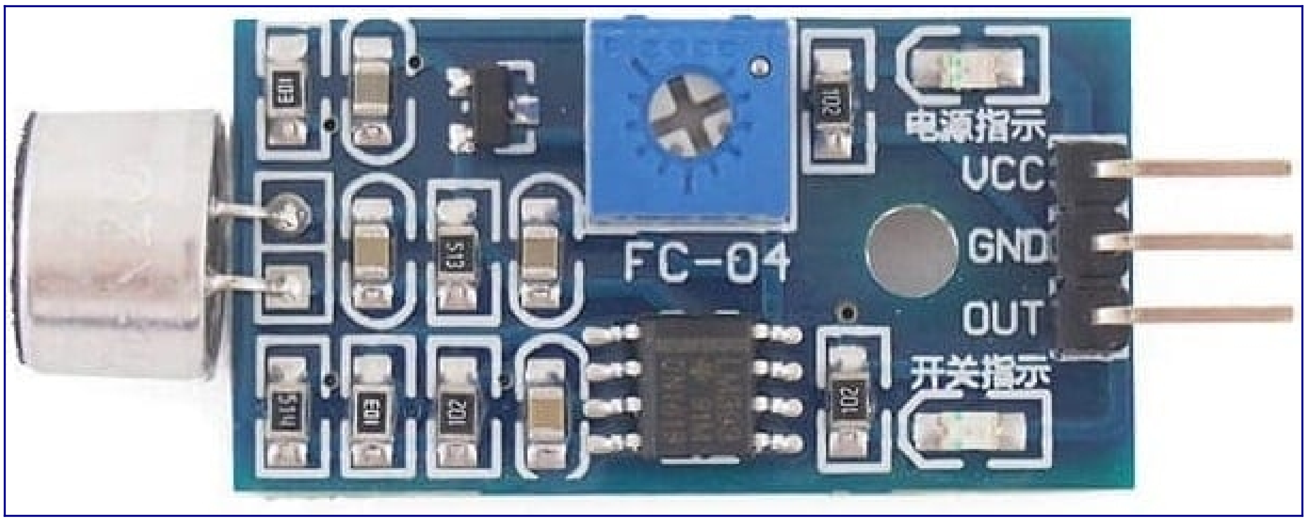
Microphone sensor

The microphone sound sensor, as the name says, detects sound. It gives a measurement of how loud a sound is. The sound sensor is a small board that combines a microphone (50Hz-10kHz) and some processing circuitry to convert sound waves into electrical signals. This electrical signal is fed to on-board LM393 High Precision Comparator to digitize it and is made available at OUT pin.

The module has a built-in potentiometer for sensitivity adjustment of the OUT signal. We can set a threshold by using a potentiometer. So that when the amplitude of the sound exceeds the threshold value, the module will output LOW otherwise HIGH.

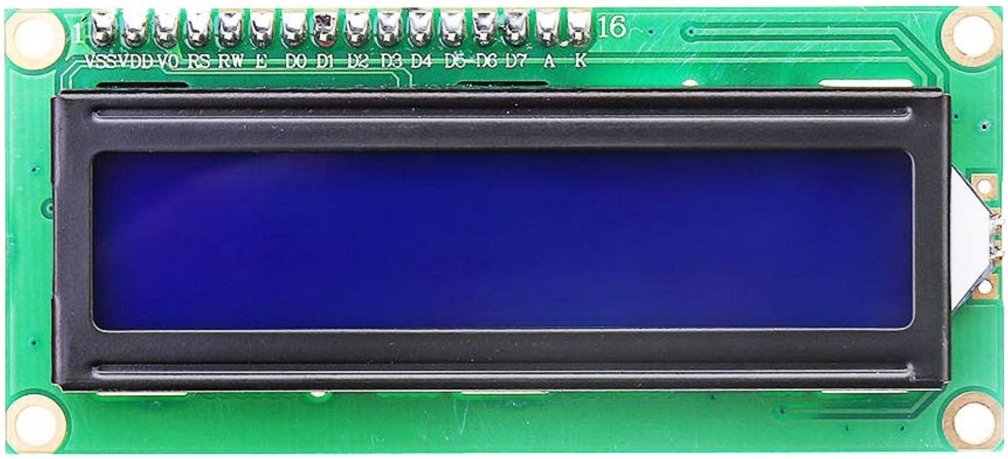
Apart from this, the module has two LEDs. The Power LED will light up when the module is powered. The Status LED will light up when the digital output goes LOW.

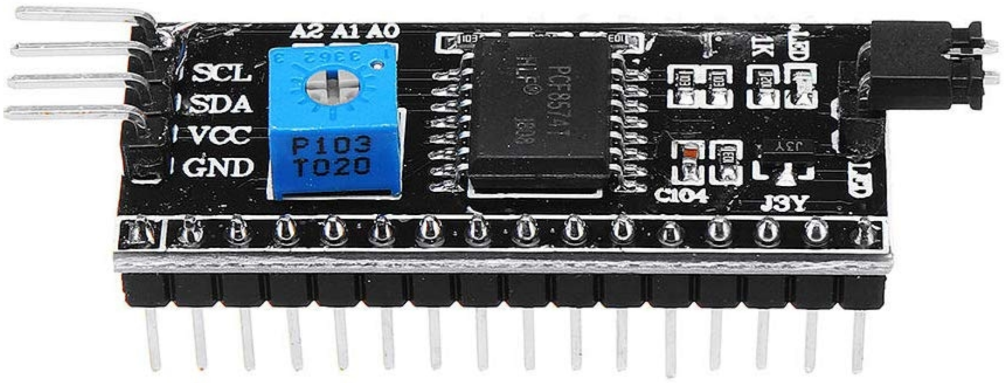
The sound sensor only has three pins: VCC, GND & OUT. VCC pin supplies power for the sensor & works on 3.3V to 5V. OUT pin outputs HIGH when conditions are quiet and goes LOW when sound is detected.



Microphone sensor

16x2 LCD Display with I2C Module with Controller

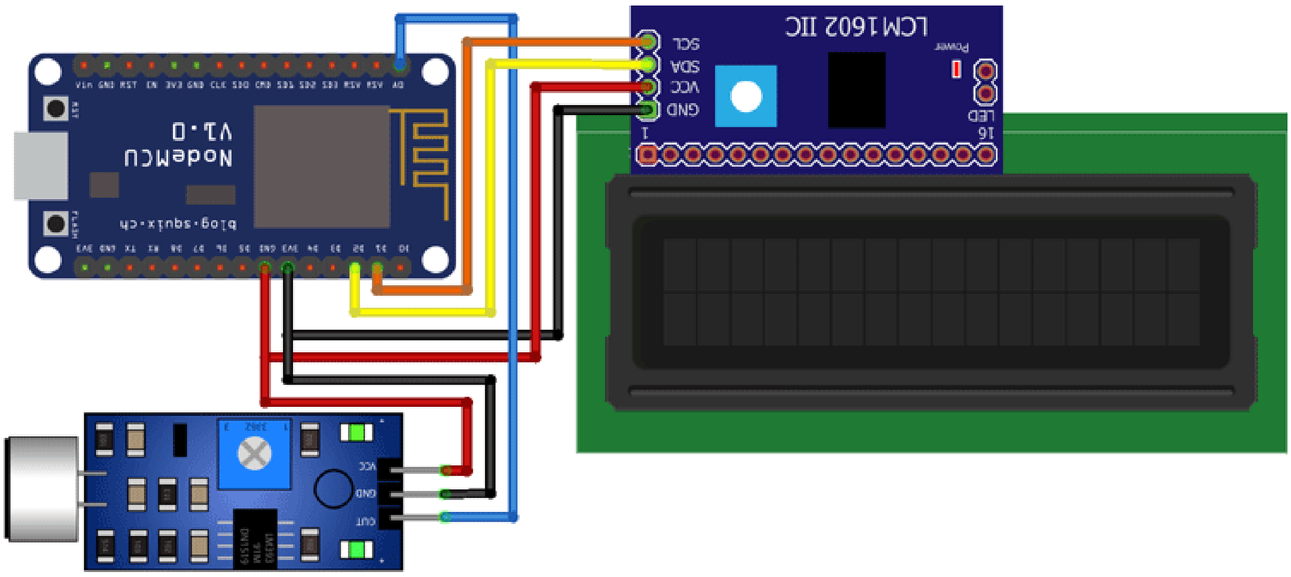




Software and online services:

- Python IDE
- Arduino IDE
- SQLite
- Vue js
- Think space
-

Circuit Diagram

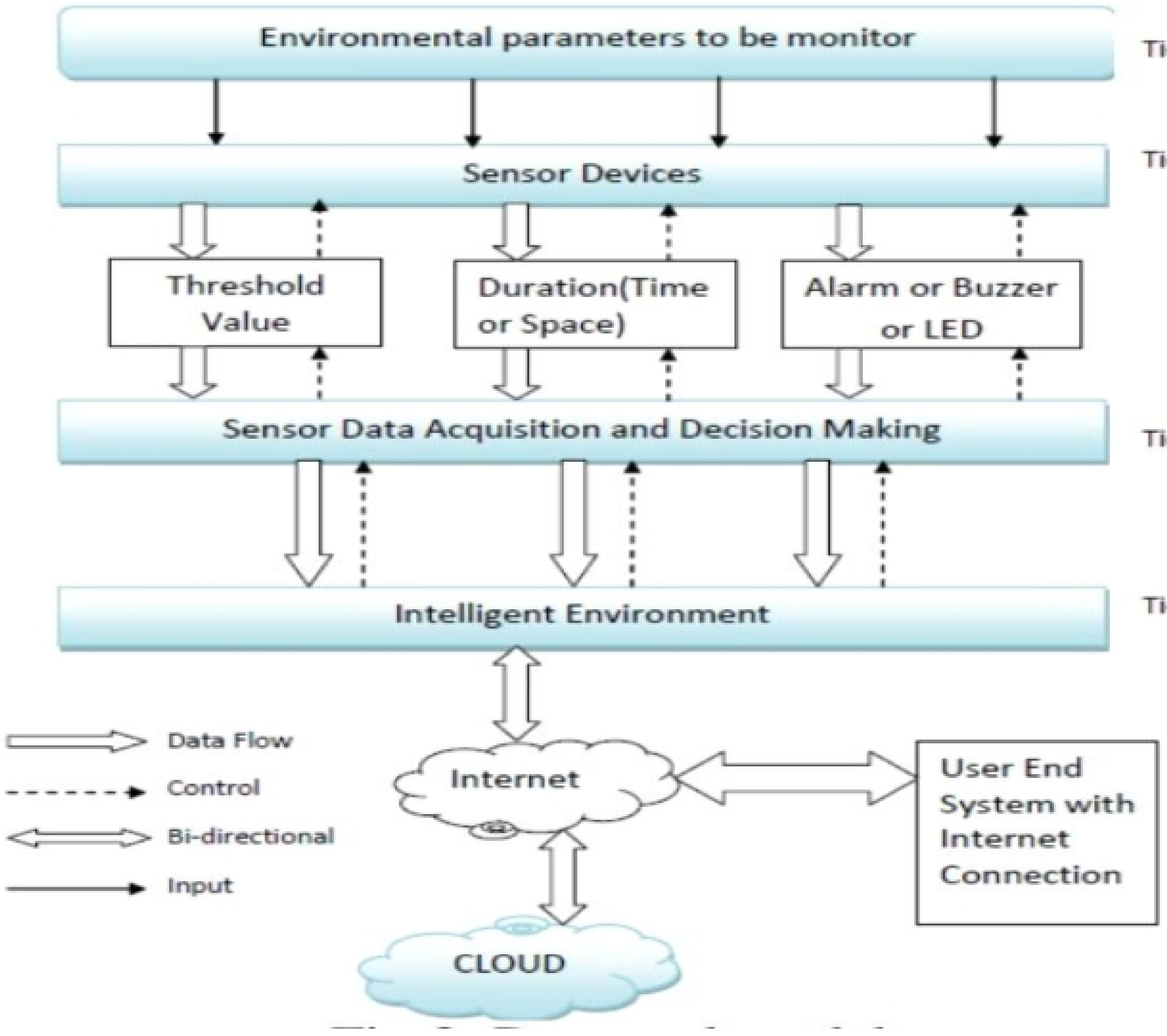


IoT Based Noise Pollution Monitoring System

The connections are pretty simple, we just have to connect the sound sensor to one of the Analog pin and the LCD to the I2C pins.

n the above diagram, we have connected the power pins of the sound sensor and LCD display to 3v3 and GND pin of NodeMCU. Along with that, we have also connected the SCL and SDA pins of the module to D1 and D2 respectively, and the OUT pin of the sound sensor to A0 pin.

Proposed model for this project



Program use in the project

Here, we have to develop a code that takes input from the sound sensor and maps it value to decibels and after comparing the loudness, it should not only print it to the 16*2 LCD display.

In the very first part of the code, we have included all the necessary libraries and definitions. Also, we have defined the necessary variables and objects for further programming.

```
#define BLYNK_PRINT Serial
#include <ESP8266WiFi.h>
#include <BlynkSimpleEsp8266.h>
#include <LiquidCrystal_I2C.h>
#define SENSOR_PIN A0
LiquidCrystal_I2C lcd(0x3F, 2, 1, 0, 4, 5, 6, 7, 3, POSITIVE);
const int sampleWindow = 50;
unsigned int sample;
int db;
char auth[] = "IEu1xT825VDt6hNfrcFgdJ6InJ1QUfsA";
char ssid[] = "YourSSID";
char pass[] = "YourPass";
```

Further ahead, we have created a Blynk function to handle the virtual pin that our gauge is connected to. We are simply sending the values stored in the dB variable to the V0 pin.

```
BLYNK_READ(V0)
{
  Blynk.virtualWrite(V0, db);
}
```

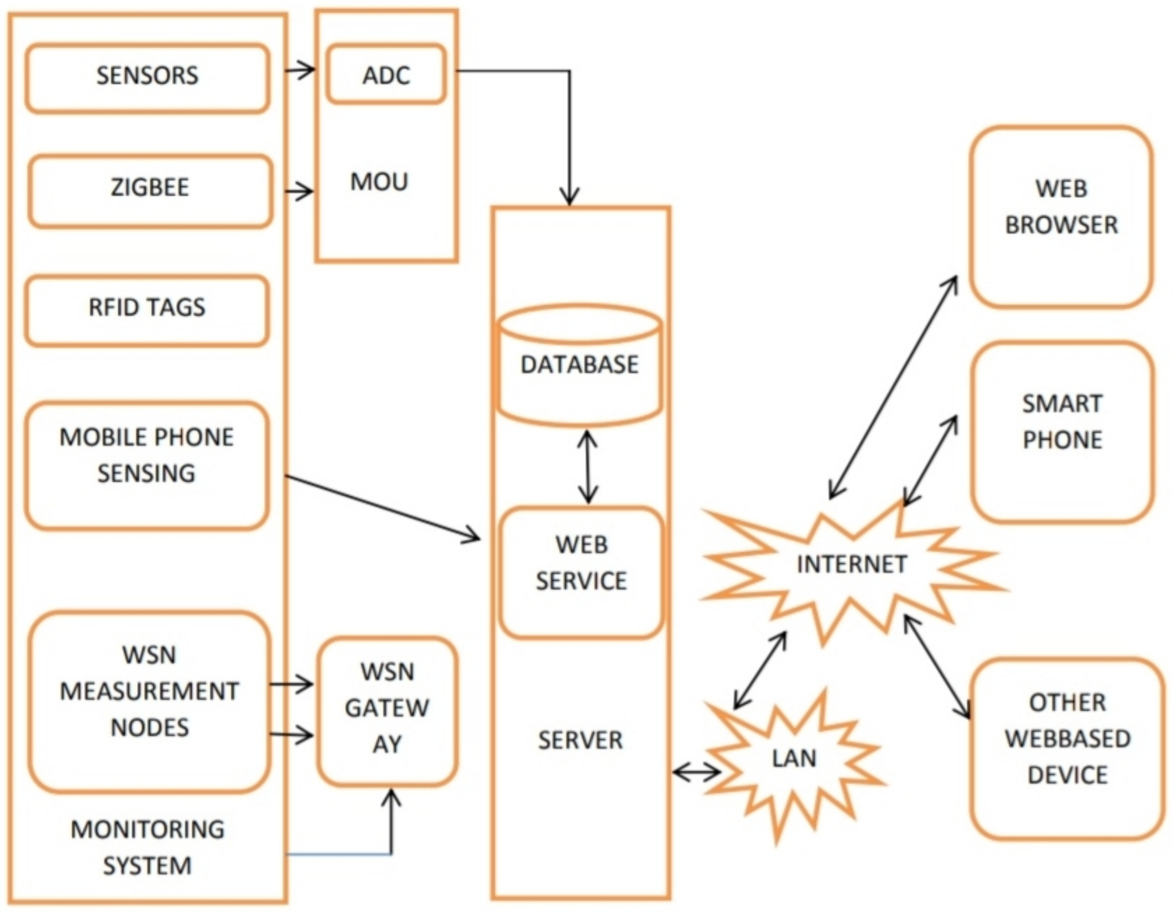
In the setup part of the code, we are defining the pin mode as input and beginning the LCD display as well as the Blynk function.

```
void setup() {
  pinMode (SENSOR_PIN, INPUT);
  lcd.begin(16, 2);
  lcd.backlight();
  lcd.clear();
  Blynk.begin(auth, ssid, pass);
}
```

In the loop part, we are doing all the processing tasks like comparison and value assignment along with running the Blynk function.

```
void loop() {
  Blynk.run();
  unsigned long startMillis = millis(); // Start of sample window
  float peakToPeak = 0; //peak-to-peak level
  unsigned int signalMax = 0; //minimum value
  unsigned int signalMin = 1024; //maximum value
  // collect data for 50 mS
  while (millis() - startMillis < sampleWindow)
  {
    sample = analogRead(SENSOR_PIN); //get reading from microphone
    if (sample < 1024) // toss out spurious readings
    {
      if (sample > signalMax)
      {
        signalMax = sample; // save just the max levels
      }
      else if (sample < signalMin)
      {
        signalMin = sample; // save just the min levels
      }
    }
  }
  peakToPeak = signalMax - signalMin; // max - min = peak-peak amplitude
  Serial.println(peakToPeak);
  db = map(peakToPeak, 20, 900, 49.5, 90); //calibrate for deciBels
  lcd.setCursor(0, 0);
  lcd.print("Loudness: ");
  lcd.print(db);
  lcd.print("dB");
  if (db <= 50)
  {
    lcd.setCursor(0, 1);
    lcd.print("Level: Quite");
  }
  else if (db > 50 && db < 75)
  {
    lcd.setCursor(0, 1);
    lcd.print("Level: Moderate");
  }
  else if (db >= 75)
  {
    lcd.setCursor(0, 1);
    lcd.print("Level: High");
  }
  delay(600);
  lcd.clear();
}
```

Existing system model



Backup program(other method)

```
 #include <ESP8266WiFi.h>
#include <SPI.h>
#include <Wire.h>
#include <Adafruit_GFX.h>
#include <Adafruit_SSD1306.h>

#define SCREEN_WIDTH 128
#define SCREEN_HEIGHT 64
#define OLED_RESET -1       /
Adafruit_SSD1306 display(SCREEN_WIDTH, SCREEN_HEIGHT, &Wire, OLED_RESET);

int num_Measure = 128 ;
int pinSignal = A0;
long Sound_signal;
long sum = 0 ;
long level = 0 ;
int soundlow = 40;
int soundmedium = 500;
int error = 33;

String apiKey = "14K8UL2QEK8BTHN6";
const char *ssid = "Alexahome";
const char *pass = "12345678";
const char* server = "api.thingspeak.com";

WiFiClient client;

void setup ()
{
  pinMode (pinSignal, INPUT);
  Serial.begin (115200);
  display.begin(SSD1306_SWITCHCAPVCC, 0x3C);
  display.clearDisplay();
  delay(10);

  Serial.println("Connecting to ");
  Serial.println(ssid);

  display.clearDisplay();
  display.setCursor(0,0);
  display.setTextSize(1);
  display.setTextColor(WHITE);
 display.println("Connecting to ");
  display.setTextSize(2);
  display.print(ssid);
  display.display();

  WiFi.begin(ssid, pass);

  while (WiFi.status() != WL_CONNECTED)
  {
    delay(500);
    Serial.print(".");
  }
    Serial.println("");
    Serial.println("WiFi connected");

    display.clearDisplay();
    display.setCursor(0,0);
    display.setTextSize(1);
    display.setTextColor(WHITE);
    display.print("WiFi connected");
    display.display();
    delay(4000);
}


void loop ()
{

  // Performs 128 signal readings
  for ( int i = 0 ; i <num_Measure; i ++)
  {
   Sound_signal = analogRead (pinSignal);
    sum =sum + Sound_signal;
  }

  level = sum / num_Measure;

  Serial.print("Sound Level: ");
  Serial.println (level-error);

    display.clearDisplay();
    display.setCursor(0,0);  //oled display
    display.setTextSize(1);
    display.setTextColor(WHITE);
    display.println("--- Decibelmeter ---");
  if( (level-error) < soundlow)
  {

    Serial.print("Intensity= Low");
    display.setCursor(0,20);
    display.setTextSize(1);
    display.setTextColor(WHITE);
    display.print("Sound Level: ");
    display.println(level-error);

    display.setCursor(0,40);
    display.print("Intensity: LOW");

    display.display();
//    {
//      if( (level-error) < 0)
//      Serial.print("Intensity= Low");
//      display.setCursor(0,20);
//      display.setTextSize(1);
//      display.setTextColor(WHITE);
//      display.print("Sound Level: 0");
//
//      display.setCursor(0,40);
//      display.print("Intensity: LOW");
//
//      display.display();
//    }

  }
  if( ( (level-error) > soundlow ) && ( (level-error) < soundmedium )  )
  {

    Serial.print("Intensity=Medium");
    display.setCursor(0,20);  //oled display
    display.setTextSize(1);
    display.setTextColor(WHITE);
    display.print("Sound Level: ");
    display.println(level-error);

    display.setCursor(0,40);
    display.print("Intensity: MEDIUM");
    display.display();

  }
```

```
if( (level-error) > soundmedium )
{

  Serial.print("Intensity= High");
  display.setCursor(0,20);  //oled display
  display.setTextSize(1);
  display.setTextColor(WHITE);
  display.print("Sound Level: ");
  display.println(level-error);

  display.setCursor(0,40);
  display.print("Intensity: HIGH");
  display.display();

}
sum = 0 ;
delay(200);

if (client.connect(server, 80))
{
  String postStr = apiKey;
  postStr += "&field1=";
  postStr += String(level-error);
  postStr += "r\n";

  client.print("POST /update HTTP/1.1\n");
  client.print("Host: api.thingspeak.com\n");
  client.print("Connection: close\n");
  client.print("X-THINGSPEAKAPIKEY: " + apiKey + "\n");
  client.print("Content-Type: application/x-www-form-urlencoded\n");
  client.print("Content-Length: ");
  client.print(postStr.length());
  client.print("\n\n");
  client.print(postStr);

}
  client.stop();
```

1. ***Noise Level Data:***

Real-time Or historical data on noise levels in specific areas, such as cities, neighborhoods or even individual streets. This data can be collected from various sources like noise monitoring stations, sensors or crowd-sourced report

2. ***Educational Resources:***

   Information on the causes and effects Of noise pollution, including its impact on health, well-being, and the environment. This can include articles, videos, infographics, and interactive content. Noise Mapping: Interactive maps that show noise levels in different areas, allowing users to visualize noise hotspots and trends.

3. ***Noise Regulations:***

   Information about local and national noise regulations and guidelines, helping users understand their rights and responsibilities regarding noise pollution.

4. ***Noise Complaint System:***

   A feature that allows residents to report noise disturbances or violations, which can be sent to relevant authorities for action.

5. Noise Reduction Tips:

Tips and guides on how individuals and communities can reduce noise pollution in their surroundings. This can include advice on soundproofing, choosing quieter appliances, and promoting noise-friendly practices.

WEB DEVELPOMENT USING IN NOISE POLLUTION MONITORING

Monitoring noise pollution using web development involves creating a digital platform that allows users to access real-time or historical noise data, report noise disturbances, and access relevant information about noise pollution. Here's a step-by-step guide on how to develop a noise pollution monitoring system using web development:

1. *Project Planning:*

- ➢ Define the project scope and objectives.
- ➢ Identify the target audience (e.g., residents, local authorities, researchers).
- ➢ Determine the geographic area or region you want to monitor for noise pollution.

2. *Data Collection:*

- ➢ Set up noise monitoring stations or use existing data sources such as sensors and government databases.
- ➢ Determine the data parameters you want to collect, such as noise levels, location, and timestamp.

3. *Web Development:*

➢ Choose the technology stack for web development (e.g., HTML, CSS, JavaScript, and a backend language like Python, Node.js, or Ruby).

➢ Develop a user-friendly web interface that allows users to access noise data, report noise disturbances, and explore the platform's features.

5. *Real-Time Data Integration:*

➢ Integrate the noise monitoring stations or sensors to transmit data in real-time to the platform.

➢ Implement data visualization tools like charts, graphs, and maps to display real-time noise data,

6. *User Registration and Authentication:*

- Implement user registration and login functionality to allow users to create accounts and personalize their experience.
- Ensure data security and privacy by protecting user information and complying with data protection regulations.

7. *Noise Reporting System:*

- Create a noise reporting feature that enables users to submit noise complaints or observations, including the location, time, and description of the noise.

8. *Noise Data Presentation:*

- Display noise data on an interactive map or in a tabular format, allowing users to explore noise levels in different areas.

9. *Noise Alerts:*

- Implement a notification system that can send alerts to users when noise levels exceed predefined thresholds.

10. *Educational Content:*

- Provide information on noise pollution, its effects, and tips on noise reduction.
- Include articles, videos, and infographics to educate users.

11. *Community Engagement:*

- Create discussion forums, comment sections, or social media integration to foster user interaction and community engagement.

12. *Data Analysis and Trends:*

- Offer data analysis tools and reports that highlight noise pollution trends over time.

13. *Mobile Responsiveness:*

➢ Ensure that the platform is accessible on various devices, including smartphones and tablets.

14. *Testing and Quality Assurance:*

➢ Thoroughly test the platform for functionality, usability, and security. • Address any bugs or issues that arise during testing.

15. Deployment:

➢ Host the web platform on a server or cloud service to make it accessible to users.

16. *Maintenance and Updates:*

➢ Regularly update the platform to improve performance, security, and user experience.

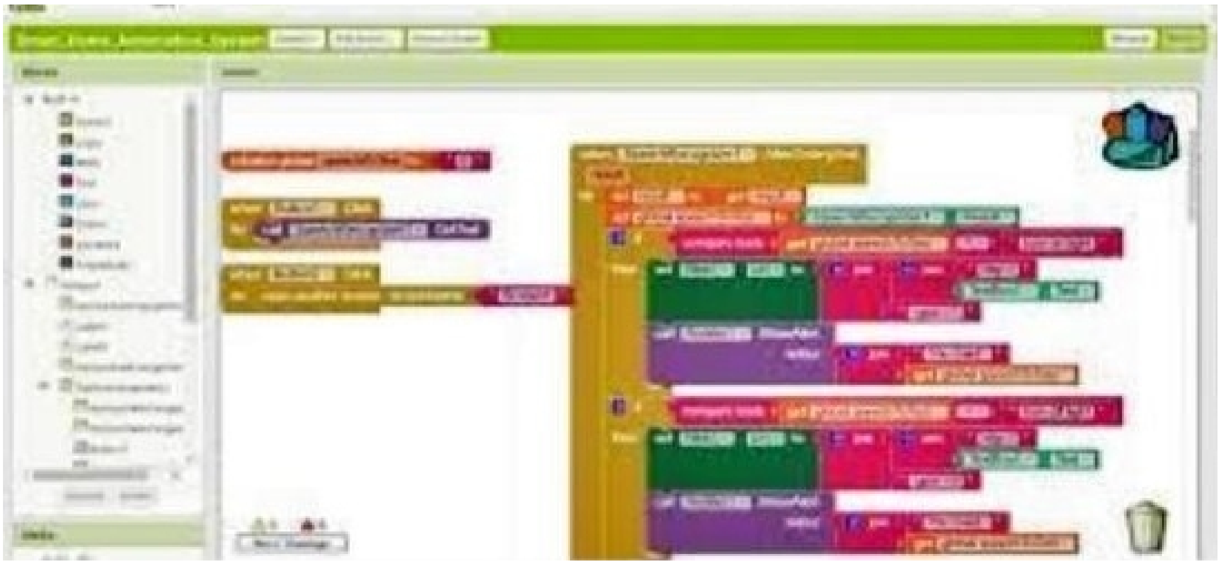
➢ Continuously monitor and maintain noise monitoring equipment to ensure accurate data collection.

17. *Promotion and Outreach:*

➢ Promote the platform through marketing efforts to attract users and increase its visibility within the community.

18. *Compliance:*

➢ Ensure compliance with local, regional, and national regulations related to noise data collection and privacy.



Mothods of noise pollution monitoring system using iot:

- ***Calibration and Maintenance:*** Regular calibration and maintenance of noise monitoring sensors are essential to ensure accurate and reliable measurements. Over time, sensors may drift or degrade, affecting data quality. Proper maintenance procedures help to maintain data integrity.
- G***eospatial Mapping:*** In some applications, noise pollution data can be integrated with geographical information systems (GIS) to create geospatial maps that illustrate noise levels across different areas. This can help in identifying noise hotspots and planning noise-reduction strategies.
- ***Community Engagement:*** Noise pollution monitoring can also involve engaging the community. Citizens can use mobile apps or web platforms to report noise complaints, and this data can be integrated into the monitoring system, allowing for a more comprehensive understanding of noise issues in the area,

- ***Regulatory Compliance:*** For industrial facilities, construction sites, and transportation hubs, noise pollution monitoring is Oftennecessary to comply with local, regional, or national regulations and standards. Data collected can be used for regulatory reporting. Noise Source Identification: Advanced systems may incorporate techniques such as acoustic fingerprinting to identify specific noise sources. This can be valuable for pinpointing the origin of noise pollution and taking appropriate action.

- ***Noise Trends Analysis:*** Long-term noise data collection enables the analysis of noise trends and patterns over extended periods. This can reveal seasonal variations, trends related to changes in urban development, and other insights for urban planning.

- ***Noise Pollution Mitigation:*** IOT noise monitoring systems can be part of a broader approach to noise pollution mitigation. Data can inform the development of strategies to reduce noise levels, such as optimizing traffic flow, implementing noise barriers, or modifying industrial processes.

- ***Public Awareness and Education:*** Accessible noise pollution data can raise public awareness about the impact of noise on health and well-being. It can also be used for educational campaigns, encouraging responsible behavior in noisy environments.

- ***Integration with Other Data:*** Noise pollution data can be integrated with other environmental data, such as air quality, temperature, and weather conditions. This holistic view can help in understanding the complex interplay of environmental factors.

- ***Customization:*** IOT noise monitoring solutions are often customizable to meet specific needs. Different industries and locations may have unique requirements, and the monitoring system can be tailored accordingly.

- ***Scalability:*** The monitoring system should be scalable to accommodate additional sensors and increased data volume as the need arises. Scalability is crucial for expanding the coverage of the monitoring network.

- ***Technological Advancements:*** Ongoing advancements in sensor technology, data analysis techniques, and IOT connectivity options will continue to enhance the capabilities Of noise pollution monitoring systems.

- ***Data Privacy and Security:*** As with any IOT system, ensuring the privacy and security of noise pollution data is vital. Robust data encryption and access controls must be in place to protect sensitive information.

Overall, IOT-based noise pollution monitoring is a powerful tool for understanding, managing, and mitigating noise pollution in various settings. It contributes to improved public health, urban planning, and the protection Of the environment.

noise pollution monitoring using IOT represents a dynamic and technology-driven approach to addressing the challenges posed by excessive noise in our modern world. This method offers a range of benefits and capabilities that significantly contribute to environmental protection, public health, urban planning, and regulatory compliance. Here are the key takeaways:

Accurate and Real-Time Data: loT-based noise pollution monitoring systems provide accurate, real-time data on noise levels, helping stakeholders make informed decisions and take prompt action when noise pollution exceeds acceptable limits.

Data Insights: The data collected can reveal patterns and trends, aiding in the identification of noise hotspots, seasonal variations, and the sources of noise pollution, which can be used for more effective mitigation strategies.

Regulatory Compliance: Industries and organizations can use 10T noise monitoring to ensure compliance with noise regulations, facilitating responsible practices and reducing the impact on neighboring communities.
Public Awareness and Engagement: These systems can raise public awareness about the effects of noise pollution and engage communities in reporting and addressing noise issues, promoting a collaborative approach to problem-solving.

Integration and Customization: IOT noise monitoring can be integrated with other environmental data and customized to suit the specific needs of different industries and locations.

Example outputs of IoT sensor data transmission, platform UI, and mobile app interfaces.

Step 1: IoT Sensor Setup:

For this example, we'll use a Raspberry Pi with a USB microphone as the IoT sensor to capture noise data.

1. **Hardware Setup:**
    - Procure a Raspberry Pi and a USB microphone.
    - Connect the USB microphone to the Raspberry Pi.
    -
2. **Python Code for Data Transmission:**

    Here's a Python script to capture audio data and send it to a server over HTTP using the `requests` library.

    python

2.
```
import requests
import sounddevice as sd

# Replace with your server's URL
SERVER_URL = "http://your-server.com/api/noise-data"

def audio_callback(indata, frames, time, status):
    if status:
        print(status, file=sys.stderr)
    data = indata.tobytes()
```

```
        response = requests.post(SERVER_URL, data=data)
        print(response.text)

    sd.InputStream(callback=audio_callback).readinto()
```

Step 2: Central Platform (Server Side):

For this example, we'll use Flask, a micro web framework for Python, to create a simple server to receive and store audio data.

1. **Install Flask:**

   bash

-
  ```
  pip install flask
  ```

- **Python Code for Central Server:**

Here's a basic example of a Flask server that receives audio data from sensors and stores it in a CSV file. This is for demonstration purposes, and in a real-world scenario, you would use a proper database.

python

2.
```
from flask import Flask, request
import csv
import os

app = Flask(__name__)

@app.route('/api/noise-data', methods=['POST'])
def receive_noise_data():
    data = request.data
    with open('noise_data.csv', 'a') as f:
        writer = csv.writer(f)
        writer.writerow([data])
    return "Data received and stored."

if __name__ == '__main__':
    app.run(debug=True)
```

Step 3: Noise Pollution Information Platform:

For this example, we'll use Flask again to create a simple platform to view the noise data.

1. **Python Code for Platform:**

   Here's an example of a basic web platform using Flask to display noise data:

   python

- `from flask import Flask, render_template`

```
import csv

app = Flask(__name__)

@app.route('/')
def display_noise_data():
    data = []
    with open('noise_data.csv', 'r') as f:
        reader = csv.reader(f)
        data = [row[0] for row in reader]
    return render_template('index.html', data=data)

if __name__ == '__main__':
    app.run(debug=True)
```

- **HTML Template (index.html):**

Create an HTML template for the platform to display the noise data.

html

2.
```
<!DOCTYPE html>
<html>
<head>
    <title>Noise Pollution Platform</title>
</head>
<body>
    <h1>Noise Pollution Data</h1>
    <ul>
        {% for item in data %}
            <li>{{ item }}</li>
        {% endfor %}
    </ul>
</body>
</html>
```

Step 4: Mobile App Development:

For this example, we'll create a simple Python script to simulate a mobile app interface using the Flask framework.

1. **Python Code for Mobile App Interface:**

   Here's an example of a simple Python script that simulates a mobile app interface using Flask:

   python

- `from flask import Flask, render_template`

```
app = Flask(__name__)

@app.route('/mobile-app')
def mobile_app():
    return render_template('mobile_app.html')

if __name__ == '__main__':
    app.run(debug=True)
```

- **HTML Template (mobile_app.html):**

Create an HTML template for the mobile app interface.

html

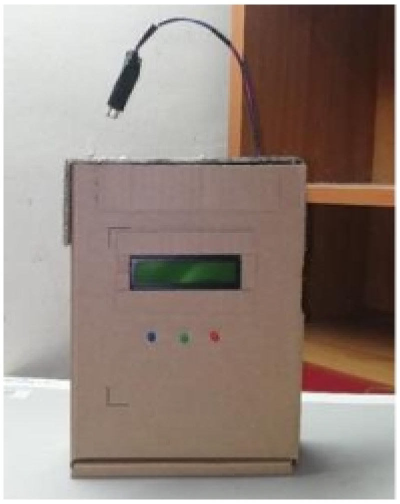
2.
```
<!DOCTYPE html>
<html>
<head>
    <title>Mobile App Interface</title>
</head>
<body>
    <h1>Mobile App Interface</h1>
    <!-- Your mobile app UI components would go here -->
</body>
</html>
```

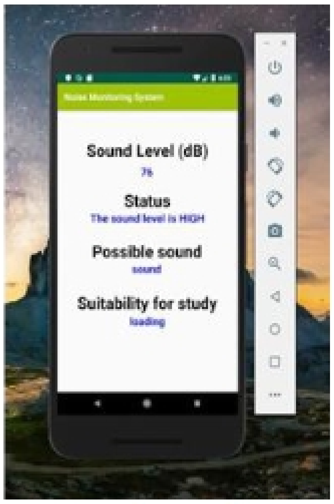
Step 5: Integration using Python:

To integrate all components, you would need to adapt your central server to handle real-time data and create APIs for the mobile app to retrieve noise data from the platform.

Remember that this is a simplified example. In a real-world scenario, you would use proper databases, implement security measures, and design more complex and user-friendly user interfaces for the platform and mobile app.

Prototype of the Project and interface of the app on mobile smartphone



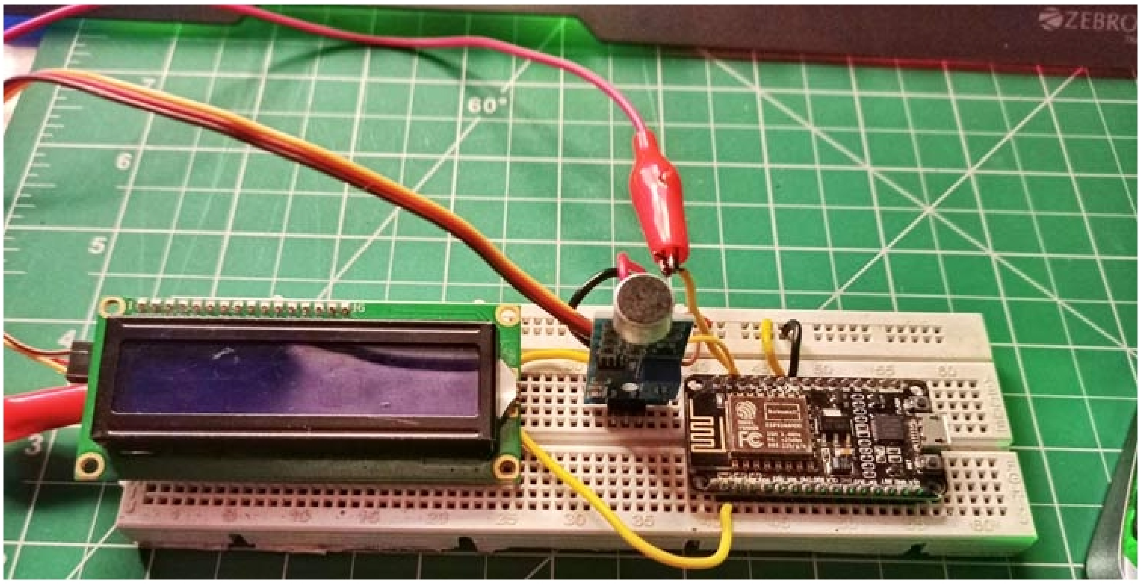

Noise Monitoring System

Sound Level (dB)
76

Status
The sound level is HIGH

Possible sound
sound

Suitability for study
loading



Data Analysis from Prototype

Table 1. Data Analysis from Prototype.

Time	Weekend (Saturday) (dBA)	Weekday (Sunday) (dBA)	Allowable Noise Level according to Environmental Department of Malaysia (dBA)	Permissible Level of Comfort according to CIBSE (dBA)
Morning 7.00am-12.00pm	47 – 60	57 – 71	55	60
Afternoon 12.01pm-14.00pm	43 – 49	54 – 69	55	60
Evening 2.01pm-7.00pm	43 – 49	62 – 69	55	60
Night 7.01pm-12.00am	42 – 59	48 – 63	55 (7.01pm–10.00pm) 45 (10.00pm– 12.00am)	60
Midnight 12.01am-6.59am	34 – 35	34 – 35	45	60

he allowable noise level that is suggested for researched area is below 55dBA for daytime and 45dBA for night. This justification is based on the researched area which is categorized asmedium density residential area. Based on CIBSE, 60dBA is the maximum allowable sound levelto achieve comfort in a building. Any sound that exceeds 60dBA will interrupt the speech intelligibility and the learning process. Based on Table 1, during weekend, the sound level in the morning and night are in between 47dBA to 60dBA and 42dBA to 59dBA. It shows that noise problem does occur during that time because the value of 55dBA is included in the range. As for afternoon, evening and midnight, all readings of sound level are below the allowable noise level standard which are during afternoon (43dBA – 49dBA), evening (43dBA – 49dBA) and midnight (34dBA -35dBA). Thus, during these times, noise problem does not occur. As for weekend, the suitable time for students to study is the whole day starting from morning until midnight because all readings are showing that they are within permissible level of comfort which is 60dBA.. As for weekday, the sound level during morning until night are all above the allowable noise level standard which are; morning (57dBA – 71dBA), afternoon (54dBA – 69dBA), evening (62dBA – 69dBA) and night (48dBA – 63dBA). It can be concluded that during that specific time, the noise problem does occur. As for midnight, the sound level recorded is 34dBA to 35dBA which indicates no noise problem. The reading at midnight also shows that it is less than 60dBA which makes it a comfortable study time for students.

APPLICATIONS OF THE PROJECT

- Roadside pollution Monitoring.
- Industrial Perimeter Monitoring.
- Site selection for reference monitoring stations.
- Indoor Air Quality Monitoring.
- Design server using IoT and upload data on that
- server with date and time.
- To make this data available to the common man.
- To set a danger limit on that server and inform
- authorities to take future actions for wellbeing.

CONCLUSION

To implement this need to deploy the sensor devices in theenvironment for collecting the data and analysis. Bydeploying sensor devices in the environment, we can bring theenvironment into real life i.e. it can interact with other objectsthrough the network. Then the collected data and analysisresults will be available to the end user through the Wi-Fi.

This data will be helpful for future analysis and it can beeasily shared to other end
users. Thismodel can be furtherexpanded to monitor the developing cities and industrial zonesfor pollution monitoring. To protect the public health frompollution, this model provides an efficient and low costsolution for continuous monitoring of environment.