|  |
| --- |
| PREC LONIDifference Between NAAC & NBA Accreditation - Haq Se EngineerJai Shriram Engineering College (@JSREC09) / Twitter**JAI SHRIRAM ENGINEERING COLLEGE**  **TIRUPPUR – 638 660**  Approved by AICTE, New Delhi & Affiliated to Anna University, Chennai  Recognized by UGC & Accredited by NAACandNBA (CSE and ECE) |

**DEPARTMENT OF**

**ELECTRONICS AND COMMUNICATION ENGINEERING**

**IBM - Naan Mudhalvan**

**Internet of Things-Group 3**

**Phase 5 - Project Documentation and Submission**

**NAME: Tamilselvi K**

**NM ID:AU711221106039**

**YEAR : III-ECE**

**OBJECTIVES:**

The primary objective of the project is to develop a noise pollution monitoring system to measure and analyze noise levels in different areas. This system aims to provide real-time data for noise pollution management, identify noise sources, and help in urban planning and policy decisions.Noise or sound level monitoring or measurement is a process to measure the magnitude of Noise in industries and residential area. Data collected from Noise level monitoring & Testing helps us to understand trends and action can be taken to reduce noise pollution.

**IOT DEVICE SETUP:**

**Noise Sensors:**

Deploy noise sensors (microphones) at various locations within the target area. These sensors will capture ambient noise levels.An IoT-based noise pollution monitoring system is implemented using a network of sensors, connectivity technologies, and data analytics platforms.

**Microcontrollers:**

Connect microcontrollers (e.g., Arduino or Raspberry Pi) to the noise sensors. These microcontrollers will process the data and send it to a central server.

**Communication:**

Use wireless communication protocols like Wi-Fi, LoRa, or cellular networks to transmit data from the microcontrollers to the central server.

**PLATFORM DEVELOPMENT:**

**Central Server:**

Develop a central server that will collect and store data from the IoT devices. This server can be hosted on the cloud or a local data center.

**Database:**

Create a database to store noise data, including timestamps and location information.

**Web Application:**

Develop a web-based application for users to access and visualize noise data. Users can view real-time noise levels, historical data, and various visualizations like heatmaps.

**MOBILE APP INTERFACE:**

Developing a mobile app for a noise pollution monitoring system project involves several key steps:

**Define Project Scope:**

Clearly define the objectives and features of your noise pollution monitoring app. Determine what kind of data you want to collect, such as noise levels, location, and time.

**Choose a Platform:**

Decide whether you want to develop the app for iOS, Android, or both. You can use native development (Swift for iOS, Java or Kotlin for Android) or cross-platform tools like React Native or Flutter.

**User Interface (UI) and User Experience (UX) Design:**

Design an intuitive and user-friendly interface to display noise data and allow users to interact with the app.

**Data Collection:**

Integrate features to collect noise data, which might involve using the device's microphone or external sensors. Implement location tracking and timestamp functionality.

**Data Analysis:**

Develop algorithms to analyze and process the collected data, allowing users to visualize noise levels and patterns.

**Mapping and Visualization:**

Implement map features to display noise data on a map and create charts or graphs to present the data clearly.

**Notifications:**

Set up notifications to alert users when noise levels exceed certain thresholds or when they are in noise-prone areas.

**Data Storage:**

Choose a database solution to store and manage the collected noise data securely.

**User Authentication and Security:**

Implement user authentication and ensure data security and privacy.

**Testing:**

Thoroughly test the app to identify and fix any bugs, and ensure it performs well on different devices.

**Compliance:**

Ensure your app complies with relevant privacy regulations and data protection laws.

**Deployment:**

Publish the app to the respective app stores (Google Play Store and Apple App Store).

**Maintenance and Updates:**

Continuously update the app to improve performance and add new features based on user feedback.

**CODE IMPLEMENTATION:**

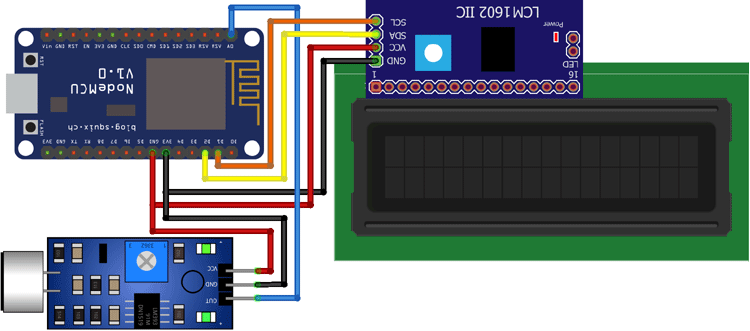
**The code for this project will involve**:

Writing firmware for the microcontrollers to read data from the sensors and transmit it.Developing the server-side code to receive, store, and process the incoming data.Creating a web application with a user-friendly interface to display noise data.

**CODING:**

#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);  
constintsampleWindow = 50;  
unsigned int sample;  
intdb;  
char auth[] = "IEu1xT825VDt6hNfrcFgdJ6InJ1QUfsA";  
char ssid[] = "realme 6";  
char pass[] = "evil@zeb";  
BLYNK\_READ(V0)  
{  
  Blynk.virtualWrite(V0, db);  
}  
void setup() {  
  pinMode (SENSOR\_PIN, INPUT);  
  lcd.begin(16, 2);  
  lcd.backlight();  
  lcd.clear();  
  Blynk.begin(auth, ssid, pass);  
}  
void loop() {  
  Blynk.run();  
  unsigned long startMillis = millis();  // Start of sample window  
  float peakToPeak = 0;  // peak-to-peak level  
  unsigned intsignalMax = 0;  //minimum value  
  unsigned intsignalMin = 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();  
}

**CIRCUIT DIAGRAM:**



**SCHEMATIC:**

**Microphone Sensor:**Use a high-sensitivity microphone sensor to capture environmental noise levels.

**Microcontroller:**Connect the microphone to a microcontroller (e.g., Arduino, Raspberry Pi) to process the audio input.

**Amplifier (Optional):**You can add an amplifier to boost the microphone's signal if needed.

**Analog-to-Digital Converter (ADC):**Use an ADC to convert the analog audio signal into digital data that the microcontroller can process.

**Data Logger:**Connect the microcontroller to a data logger or storage device to record noise data.

**Power Supply:**Ensure a reliable power source for the microcontroller and other components.

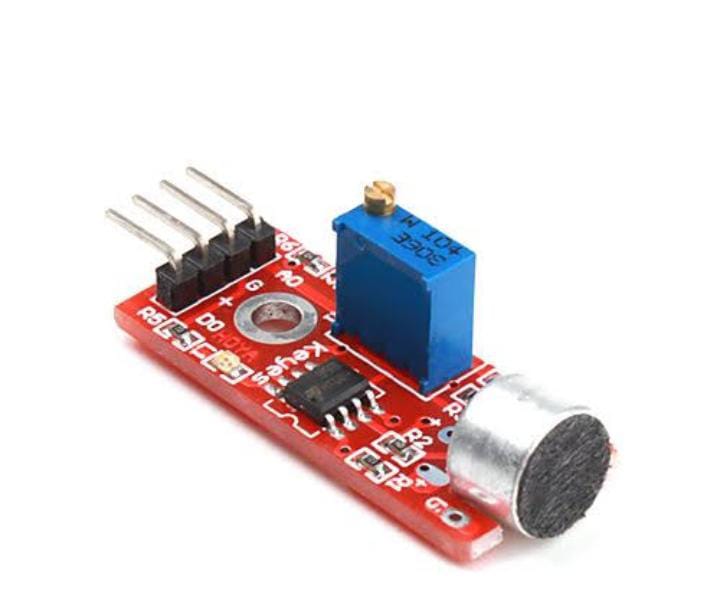
**Display (Optional):**If you want real-time monitoring, connect a display to the microcontroller to show noise levels.

**Connectivity (Optional):**Add Wi-Fi, Ethernet, or other connectivity options for remote data access.

**Housing and Enclosure:**Place all components in a protective housing or enclosure suitable for the monitoring environment.

**Software:**Develop or use software to process and analyze noise data, and potentially create visualizations.

**IOT SENSORS:**



**REAL TIME NOISE LEVEL MONITORING**:

A real-time noise level monitoring system promotes public awareness and contributes to noise pollution mitigation in several ways:

**Immediate Feedback:** Such a system provides real-time information about noise levels in specific areas. This immediate feedback allows individuals and communities to understand the noise pollution in their surroundings.

**Awareness:** By making noise data accessible to the public, individuals become more aware of the noise pollution problem in their area. This awareness can lead to a greater sense of concern and motivation to address the issue.

**Behavioral Change:** When people can see how their actions contribute to noise pollution, they may be more inclined to modify their behavior. For example, they might reduce loud activities during sensitive hours or in noise-sensitive zones.

**Community Engagement:** These systems often involve communities and local authorities in the monitoring process. This engagement can lead to collaborative efforts to reduce noise pollution.

**Policy and Regulation:** Real-time noise data can be used to support the development and enforcement of noise regulations. It provides concrete evidence for policymakers to take action against excessive noise.

**Targeted Interventions:** With data on noise hotspots, local authorities can implement targeted interventions to reduce noise in problem areas.

**Noise Complaint Resolution:** Citizens can use the data to file noise complaints with evidence, making it easier for authorities to address specific issues.

**Research and Analysis:** The data collected by these systems can be analyzed to identify patterns and sources of noise pollution. This information can inform long-term strategies for noise reduction.

**Public Health:** Noise pollution has various health impacts. Real-time monitoring can help communities understand these health risks and take steps to protect their well-being.

**CONCLUSION:**

In conclusion, effective noise pollution monitoring is essential for preserving our well-being and the environment. It enables us to identify sources of noise pollution, implement necessary regulations, and ultimately work towards a quieter and healthier future.