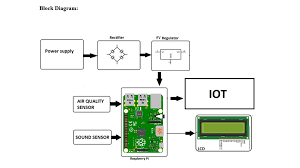
**Noise Pollution Monitoring**

**Project Objective:**

Effects and the control of noise pollution in India and to evaluate the impact on environment of noise pollution. To know an effects of Noise Pollution in India. To Examine ways to control Noise Pollution. To provide preventive and control measures on noise generated within our environment.

System uses air sensors to sense presence of harmful gases/compounds in the air and constantly transmit this data to microcontroller. Also system keeps measuring sound level and reports it to the online server over IOT. The sensors interact with microcontroller which processes this data and transmits it over internet. These AQMNs have multiple design objectives, including characterizing population exposure to pollutants, monitoring source impacts, measuring maximum and background pollutant concentrations, providing data for modeling purposes, and documenting air quality trends over time.

**Circuit Diagram:**

****

**Components used:**

**Power supply:**

A power supply unit is used to provide stable electricity. The device converts and supplies electricity of the required voltage and frequency, excluding noise from the electricity obtained from an electrical outlet. Power supplies are classified by applications for available DC, AC, and output voltage ranges.

**Rectifier:**

Using a rectifier in the power supply helps in converting AC to DC power supply. Bridge rectifiers are widely used for large appliances, which can convert high AC voltage to low DC voltage.

**Regulator:**

This is the basic L7805 voltage regulator, a three-terminal positive regulator with a 5V fixed output voltage. This fixed regulator provides a local regulation, internal current limiting, thermal shut-down control, and safe area protection.

**Air Quality Sensor:**

Air quality sensors monitor gases, such as ozone, and particulate matter, which can harm human health and the environment.

**Raspberry pi:**

The Raspberry Pi is a low cost, credit-card sized computer that plugs into a computer monitor or TV, and uses a standard keyboard and mouse. It is a capable little device that enables people of all ages to explore computing, and to learn how to program in languages like Scratch and Python.

**Sound sensor:**

The sound sensor is one type of module used to notice the sound. Generally, this module is used to detect the intensity of sound. The applications of this module mainly include switch, security, as well as monitoring.

**LCD:**

LCDs are commonly used for portable electronic games, as viewfinders for digital cameras and camcorders, in video projection systems, for electronic billboards, as monitors for computers, and in flat-panel televisions.

**Integration Approach:**

The IoT sensor used in our project will send data to the desired platform through secure and efficient manner. These sensors placed in a public area and it will collect the noise data and transmit to the established communication protocol over the internet or wireless networks. Strong security measures will protect the data during transmission ensuring accuracy and Privacy. This system enables real time monitoring noise pollution and promote noise conservation by providing accessible data to the community.

**Innovation:**

**Data analytics to identify**

* + - Noise pollution patterns
    - High noise areas
    - Potential sources

**Potential Sources:**

Noise pollution can come from outdoor sources, such as road traffic, jet planes, garbage trucks, construction equipment, manufacturing process- es, lawn mowers, leaf blowers, and indoor sources, including: boom boxes, heating and air conditioning units, and metal chairs scraping on floors.

These are the major potential sources of noise pollution.

**High Noise Areas:**

An air quality and noise pollution monitoring system that uses the internet of things (IoT) to monitor and check real-time air quality and noise pollution in a specific location for smart environment.

It detects dangerous and poisonous chemicals such as NH 3 , benzene, smoking, and CO 2 using air sensors.

**Noise Pollution Patterns:**

Sensor Deployment: Install IoT sensors equipped with microphones or sound level meters at various locations within the target area. Ensure that these sensors are capable of collecting real-time audio data.

Data Collection: Collect audio data continuously from these sensors. Ensure that the data is timestamped and geotagged for accurate analysis.

Data Preprocessing: Clean and preprocess the collected data. This may involve removing outliers, filtering noise, and converting raw audio data into a usable format for analysis.

Feature Extraction: Extract relevant features from the audio data. These features might include sound intensity levels, frequency spectra, and temporal patterns.

Data Storage: Store the pre processed data in a secure and scalable database or data warehouse.

Analytics and Pattern Recognition: Apply data analytics techniques such as machine learning algorithms to analyze the data.

Common approaches include clustering to identify noise sources, time-series analysis to detect patterns, and anomaly detection to find unusual noise events.

Visualization: Create visualizations, such as heatmaps or charts, to represent noise pollution patterns over time and space. This helps in understanding the data and conveying insights effectively.

Pattern Identification: Identify patterns in the data, such as daily or seasonal variations in noise levels, noisy hotspots, or specific noise events that require attention.

Alerts and Notifications: Implement a system that can trigger alerts or notifications when noise pollution exceeds predefined thresholds or when unusual patterns are detected.

Continuous Monitoring: Maintain an ongoing monitoring system to track changes in noise pollution patterns and assess the effectiveness of mitigation measures.

By combining IoT technology and data analytics, we can gain valuable insights into noise pollution patterns and take informed actions to address environmental concerns

**Building the project idea:**

**#include <LiquidCrystal.h> // include the LiquidCrystal library**

**const int micPin1 = A0; // define the pin for the first microphone**

**const int micPin2 = A1; // define the pin for the second microphone**

**const int micPin3 = A2; // define the pin for the third microphone**

**const int buzzerPin = 9; // define the pin for the buzzer**

**const int ledPin = 6; // define the pin for the LED**

**const int contrast = 50; // define the LCD contrast**

**LiquidCrystal lcd(12, 11, 5, 4, 3, 2); // initialize the LCD display**

**void setup() {**

**pinMode(buzzerPin, OUTPUT); // set the buzzer pin as output**

**pinMode(ledPin, OUTPUT); // set the LED pin as output**

**lcd.begin(16, 2); // initialize the LCD display**

**analogWrite(6,contrast); // set the LCD contrast**

**Serial.begin(9600); // initialize the serial monitor**

**}**

**void loop() {**

**// read the values from the microphones**

**int micValue1 = analogRead(micPin1);**

**int micValue2 = analogRead(micPin2);**

**int micValue3 = analogRead(micPin3);**

**// calculate the sound levels in dB for each microphone**

**float voltage1 = micValue1 \* 5.0 / 1024.0; // convert the first microphone value to voltage (5V reference)**

**float voltage2 = micValue2 \* 5.0 / 1024.0; // convert the second microphone value to voltage (5V reference)**

**float voltage3 = micValue3 \* 5.0 / 1024.0; // convert the third microphone value to voltage (5V reference)**

**float dB1 = 20 \* log10(voltage1/0.0063); // calculate the sound level in dB for the first microphone**

**float dB2 = 20 \* log10(voltage2/0.0063); // calculate the sound level in dB for the second microphone**

**float dB3 = 20 \* log10(voltage3/0.0063); // calculate the sound level in dB for the third microphone**

**// calculate the average sound level in dB for all microphones**

**float averageDB = (dB1 + dB2 + dB3) / 3;**

**// display the sound level on the LCD display and the serial monitor**

**lcd.setCursor(0, 0); // set the cursor to the first row of the LCD display**

**lcd.print("Sound Level: "); // print the text "Sound Level: " on the LCD display**

**lcd.setCursor(0, 1); // set the cursor to the second row of the LCD display**

**lcd.print(averageDB); // print the average sound level on the LCD display**

**Serial.print("Sound Level: "); // print the text "Sound Level: " on the serial monitor**

**Serial.println(averageDB); // print the average sound level on the serial monitor**

**// control the LED and the buzzer based on the sound level**

**if (averageDB > 70) { // if the sound level is higher than 70 dB**

**digitalWrite(ledPin, HIGH); // turn the LED on**

**tone(buzzerPin, 1000, 500); // turn the buzzer on**

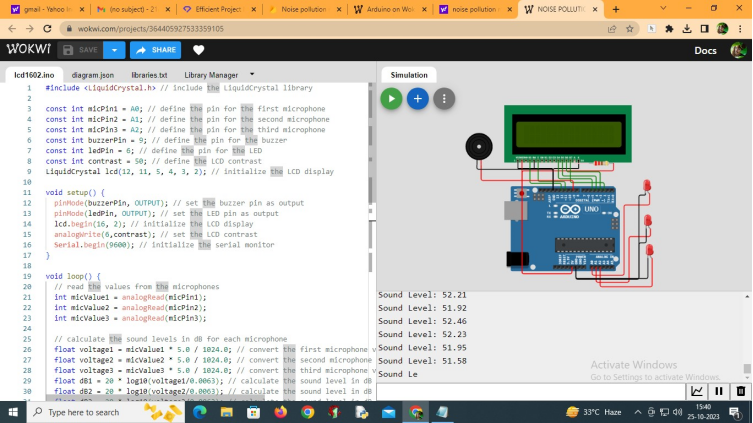
**} else { // if the sound level is lower than 70 dB**

**digitalWrite;**

**}**

**}**

**Output&Simulation:**

****

**Sound Level: 52.14**

**Sound Level: 51.43**

**Sound Level: 51.41**

**Sound Level: 51.50**

**Sound Level: 51.45**

**Sound Level: 51.70**

**Sound Level: 51.80**

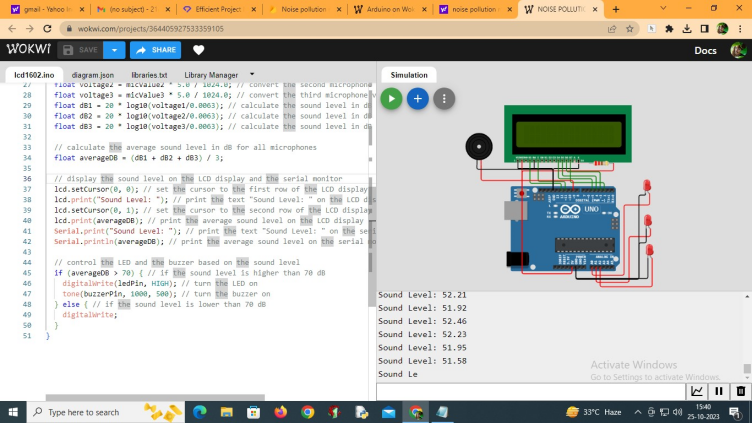
**Sound Level: 51.97**

**Sound Level: 52.36**

**Sound Level: 52.16**

**Sound Level: 52.24**

**Sound Level: 52.27**

****

**RaspberryPi Integration:**

**1. \*\*Hardware Setup\*\*:**

- Choose a compatible microphone or sound sensor for capturing audio data.

- Connect the microphone/sensor to the Raspberry Pi's GPIO pins.

- Ensure proper power supply and connectivity.

**2. \*\*Software Configuration\*\*:**

- Install the necessary operating system (e.g., Raspbian) on the Raspberry Pi.

- Set up Python or another programming language for data processing.

- Install libraries for audio processing, such as PyAudio.

**3. \*\*Data Acquisition\*\*:**

- Continuously record audio data from the microphone/sensor using the Raspberry Pi.

- Store audio files or processed data locally or transmit it to a remote server for analysis.

**4. \*\*Data Processing\*\*:**

- Analyze audio data to detect noise levels, patterns, and frequencies.

- Use software libraries for audio analysis and signal processing to identify noise pollution events.

**5. \*\*Data Storage and Visualization\*\*:**

- Store the processed data in a database or local files.

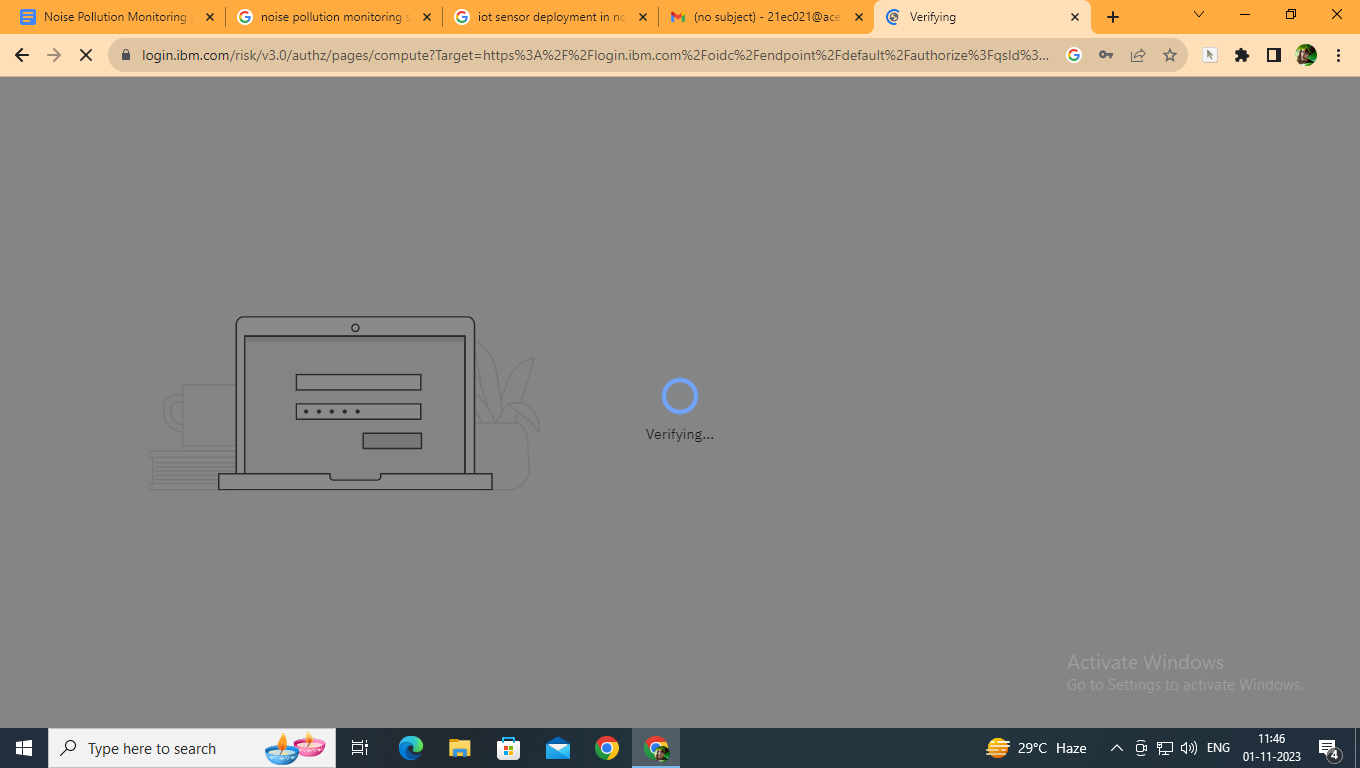
- Create visualizations or graphs to represent noise pollution trends and patterns.

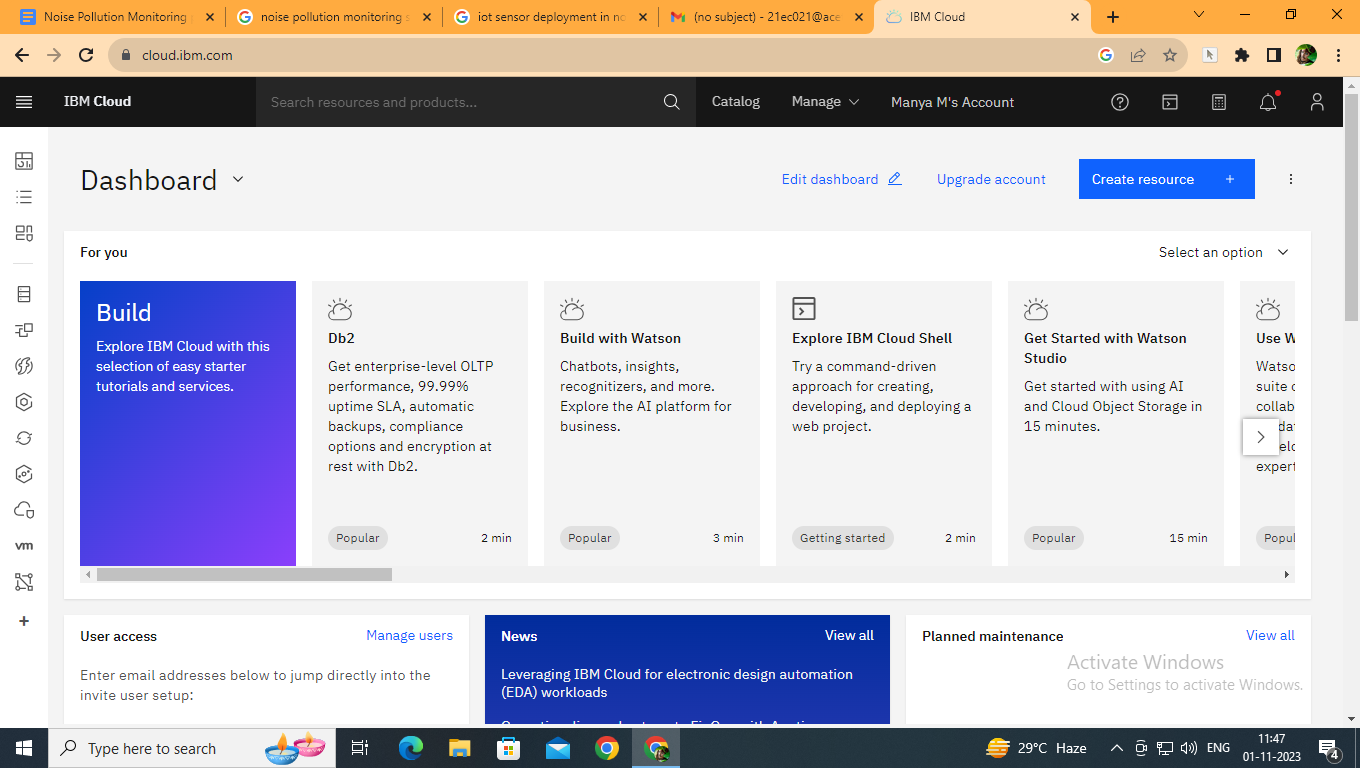
**Web Development:**

**The link for web development to display real time noise level data is**

**https://noisepollutionmonitoring1.mydurable.com**

**IBM Cloud Creation:**

****

****

This is the IBM cloud creation to access the Polluted Noise and viewing the Sound level in the LCD display.

**Conclusion:**

This project present a valuable solution for promoting noise pollution in public places through a use of IoT technology. Higher levels of noise are hazardous and it is also difficult to make them escape in a closed environment. The increased levels of noise pollution in the environment have made it an urgent need to create awareness about the causes, effects, and prevention of noise pollution.