VELAMMAL
INSTITUTE OF TECHNOLOGY

Approved by AICTE - New Delhi
Affiliated to Anna University - Chennai
Accredited by NBA & NAAC

Department of computer science and engineering

Project name : NOISE POLLUTION MONITORING

Team name : Proj_224788_Team_4

Team members :

Sijayini M (1133 21104093)

Siva Nandhini V (113321104094)

Sneka K (113321104096)

Subashree S (113321104098)

PROJECT

Creating a Noise pollution Monitoring (Internet of Things) involves a combination of hardware, software, and connectivity. While web development technologies may not be the only requirement, they can play a crucial role in creating a user interface for monitoring and controlling the air conditions processing.

PLATFORM REQUIRED:

1. Hardware Components:

-IoT Hardware: Choose an IoT development board or microcontroller, such as Arduino, Raspberry Pi, or specialized IoT platforms like ESP8266 or ESP32.

- Power Supply: Ensure a stable power source for your IoT device, which may include batteries or a direct power connection.

2. User Interface :

- If you want to provide real-time monitoring and data visualization, you'll need to create a user interface. This can be a web application or a mobile app. Consider using frameworks like React, Angular, or Vue.js for web development or languages like Swift for mobile app development.

3. Alerting and Reporting :

- Implement a system to generate alerts or reports based on noise pollution thresholds. This can be integrated into the cloud platform or handled separately.

4. Security:

- Ensure that the data transmitted from the IoT device to the cloud and stored in the database is secure. Use encryption, authentication, and access control measures to protect the data.

5. Regulatory Considerations:

- Depending on your location and the purpose of the noise monitoring system, you may need to comply with local regulations regarding data privacy and noise monitoring. Research and ensure that your system adheres to any applicable laws and regulations.

6. Maintenance and Updates:

- Plan for regular maintenance and updates of both the hardware and software components to keep the system running smoothly.

WEB DEVELOPMENT TECHNOLOGIES:

- **Front-End**: You can use HTML, CSS, and JavaScript for creating a web-based dashboard or user interface. Frameworks like React, Angular, or Vue.js can simplify the development process.
- **Back-End**: You might need a server to handle data processing, user authentication, and other backend functionalities. You can use Node.js, Python, Ruby, or any other server-side technology.
- **Databases**: Use databases (e.g., MySQL, PostgreSQL, MongoDB) to store and retrieve data.
- **APIs**: Create APIs to connect the front-end and back-end. RESTful or GraphQL APIs are common choices.

FEATURES :

1. **Security:** Ensure that the data transmitted from the IoT device to the cloud and stored in the database is secure. Use encryption, authentication, and access control measures to protect the data.

2. **Mobile Application**: Consider developing a mobile app for users to access the smart monitoring systematic features and receive real-time updates.

3. **Data Analytics and Visualization**: Use tools and libraries for data analytics and visualization to gain insights from the data collected by the IoT devices.

4. **Remote Monitoring and Control**: Implement the ability to remotely monitor and control the air purity information through the web interface or mobile app.

5. **Notifications and Alerts**: Set up notifications and alerts to notify maintenance personnel or administrators of any issues or anomalies in the device.

6. **Support**: Establish a system for regular updates and troubleshooting.

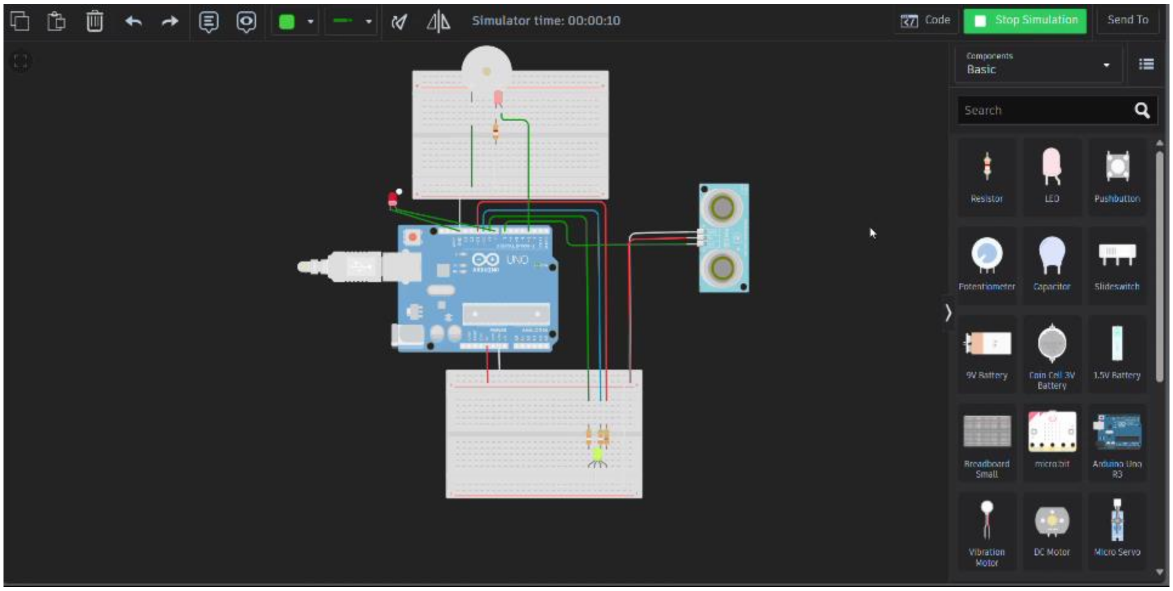
CODE IMPLEMENTATION:

```
•   Const int pingPin = 7;
•   const int red = 11;
•   const int blue = 10;
•   int green = 9;
•
•   // Define pins for the noise LED display
•   const int noiseLED = 8; // You can use any available digital pin
•
•   void setup() {
•     // initialize serial communication:
•     Serial.begin(9600);
•     pinMode(red, OUTPUT);
•     pinMode(blue, OUTPUT);
•     pinMode(green, OUTPUT);
•     pinMode(noiseLED, OUTPUT); // Set noise LED pin as OUTPUT
•     pinMode(3, OUTPUT);
•   }
•
•   void loop() {
•     digitalWrite(3, HIGH);
•     delay(1000)
•     digitalWrite(3, LOW);
```

- Delay(1000);
-
- // establish variables for duration of the ping, and the distance result
- // in inches and centimeters:
- long duration, inches, cm;
- // The PING))) is triggered by a HIGH pulse of 2 or more microseconds.
- // Give a short LOW pulse beforehand to ensure a clean HIGH pulse:
- pinMode(pingPin, OUTPUT);
- digitalWrite(pingPin, LOW);
- delayMicroseconds(2);
- digitalWrite(pingPin, HIGH);
- delayMicroseconds(5);
- digitalWrite(pingPin, LOW);
-
- // The same pin is used to read the signal from the PING))): a HIGH pulse
- // whose duration is the time (in microseconds) from the sending of the ping
- // to the reception of its echo off of an object.
- pinMode(pingPin, INPUT);
- duration = pulseIn(pingPin, HIGH);
-
- // convert the time into a distance
- inches = microsecondsToInches(duration);
- cm = microsecondsToCentimeters(duration);
- Serial.print(inches);
- Serial.print(“in, “);
- Serial.print(cm);
- Serial.print(“cm”);
- Serial.println();

```
•   analogWrite(red, cm);
•       analogWrite(blue, 255 – cm);
•       analogWrite(green, inches);
•     } else {
•       analogWrite(red, 0);
•       analogWrite(blue, 0);
•       analogWrite(green, 0);
•     }
•     int soundLevel = analogRead(A0);
•     if (soundLevel > 200) { // Adjust this threshold as needed
•       digitalWrite(noiseLED, HIGH); // Turn on the noise LED
•     } else {
•       digitalWrite(noiseLED, LOW); // Turn off the noise LED
•     }
•
•     delay(100);
•   }
•
•   long microsecondsToInches(long microseconds) {
•     return microseconds / 74 / 2;
•   }
•
•   long microsecondsToCentimeters(long microseconds) {
•     return microseconds / 29 / 2;
•   }
```

SIMULATION VIDEO:



CONCLUSION:

Building a noise pollution monitoring system using IoT involves a combination of hardware components, software solutions, and careful considerations to ensure its effectiveness and compliance with regulations. Additionally, creating user interfaces, implementing alerting and reporting features, ensuring security are essential aspects of developing a robust and reliable noise monitoring system. This system can contribute to environmental awareness and public health by providing valuable data on noise pollution levels.

THANK YOU