NOISE POLLUTION MONITORING

PHASE 4 : DEVELOPMENT PART 2

How does Microphone Module Work?

- The microphone based sound sensor is used to detect sound. It gives a measurement of how loud a sound .
- The sound sensor module is a small board that mixes a microphone (50Hz-10kHz) and a few 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 the OUT pin.
- The module features a built-in potentiometer for sensitivity adjustment of the OUT signal. We will set a threshold by employing a potentiometer.
- So that when the amplitude of the sound exceeds the edge value, the module will output LOW, otherwise, HIGH. Apart from this, the module has two LEDs.
- The facility LED will illuminate when the module is powered. The Status LED will illuminate 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.

Working of the project :

- Now that you have understood the code, you can simply upload it to your NodeMCU board and the project should start working.
- To make sure the values are correct, I compared them to an android application on my phone that could measure sound. As you can see from the pictures, the results were quite close.

Program for IoT Decible Meter :

- IoT-based Noise Pollution Monitor implemented using Arduino, designed to address the growing concern of noise pollution in urban areas.
- The project leverages the power of IoT to create a real-time monitoring system capable of measuring noise levels, collecting data, and providing insights for noise pollution management.
- The system is built around an Arduino microcontroller, which interfaces with a noise sensor (e.g., a sound level sensor or a microphone) to continuously monitor ambient noise levels.

- The collected data is then transmitted to a cloud-based platform using Wi-Fi or other connectivity options, allowing for remote access and analysis.

Python code

```
import serial
 import requests
 arduino = serial.Serial('COM3', 9600)
iot_endpoint = 'https://www.tinkercad.com/things/noise-pollution-monitoring-system-'
 try:
 while True:
 data = arduino.readline().decode('utf-8').strip()
 noise_level = float(data)
 payload = {'noise_level': noise_level}
response = requests.post(iot_endpoint, json=payload)
 print(f"Data sent: {data}")
except KeyboardInterrupt:
arduino.close()
```

ARDUINO UNO R3 :

```
 const int pingPin = 7;
 const int red=11;
 const int blue=10;
 int green=9;
 void setup()
 {
 Serial.begin(9600);
 pinMode(red,OUTPUT);
 pinMode(blue,OUTPUT);
 pinMode(green,OUTPUT);
 pinMode(3, OUTPUT);
}
 void loop()
 {
 digitalWrite(3, HIGH);
 delay(1000); // Wait for 1000 millisecond(s)
 digitalWrite(3, LOW);
 delay(1000); // Wait for 1000 millisecond(s)
 long duration, inches, cm;
 pinMode(pingPin, OUTPUT);
 digitalWrite(pingPin, LOW);
 delayMicroseconds(2);
 digitalWrite(pingPin, HIGH);
 delayMicroseconds(5);
 digitalWrite(pingPin, LOW);
 pinMode(pingPin, INPUT);
```

```
 duration = pulseIn(pingPin, HIGH);
 inches = microsecondsToInches(duration);
 cm = microsecondsToCentimeters(duration);
 Serial.print(inches);
Serial.print("in, ");
 Serial.print(cm);
Serial.print("cm");
Serial.println();
 if(cm<256)
{
analogWrite(red,cm);
analogWrite(blue,255-cm);
analogWrite(green,inches);
}
Else
{
analogWrite(red,0);
analogWrite(blue,0);
analogWrite(green,0);
delay(100);
}
Long microseconds To inches(long microseconds )
{
Return microseconds/74/2;
}
Long microseconds ToCentimeters(long microseconds)
{
return microseconds/29/2;
}
```

MIT APP INVENTOR :

- MIT App Inventor is a user-friendly platform for creating mobile apps, including one to combat noise pollution.
- Users can design an app to measure and record noise levels in their surroundings using a smartphone's microphone.
- The app can display real-time noise data, track historical trends, and provide alerts when noise exceeds set limits.
- It may also incorporate mapping features to pinpoint noise sources.
- By empowering individuals to monitor and address noise pollution, this MIT App Inventor application contributes to raising awareness and fostering a quieter, healthier environment.

MOBILE APP :

1. Design Your App: Start by planning the user interface and features of your app. Consider what information you want to collect and display, such as noise levels and timestamps.

2. Set Up MIT App Inventor: Create an account on the MIT App Inventor platform and begin designing your app. Remember that this is a visual, blocks-based programming tool.

3. Microphone Access: Use MIT App Inventor's "Sound Sensor" component to access the smartphone's microphone. This component can detect sound levels in real-time.

4. Data Collection: Implement a mechanism to continuously collect noise level data from the microphone. You may need to calibrate the microphone to provide meaningful readings.

5. Data Storage: Store the collected data in a list or database within the app. You won't need external hardware, but this will be limited to the device's capabilities.

6. User Interface: Create a user-friendly interface that displays real-time or historical noise data. You can use labels, charts, or other visual elements.

7. Notifications: Include an alert system to notify users when noise levels exceed predefined thresholds.

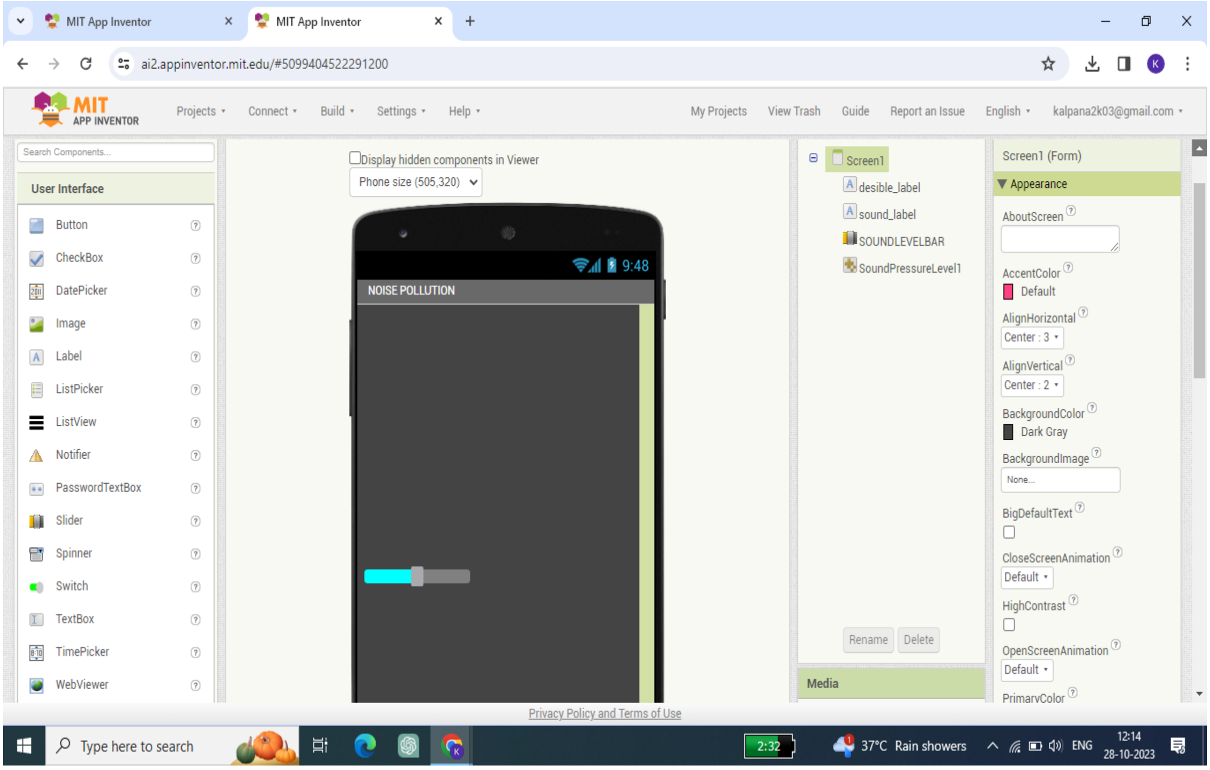
8. Privacy and Permissions: Ensure your app requests necessary permissions for microphone access and address user privacy concerns.

9. Documentation: Provide clear instructions on how to use your app.

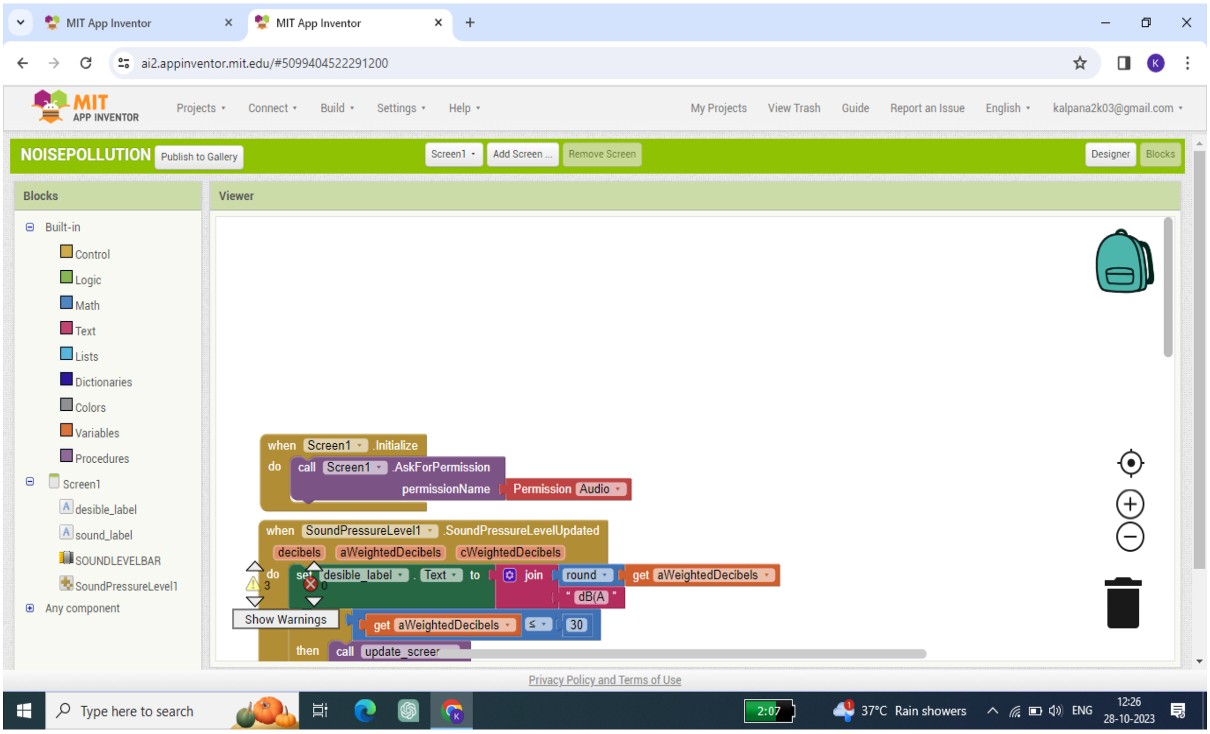
10. Testing: Test your app on various Android devices to ensure it works correctly and collects accurate noise data using the built-in microphone.

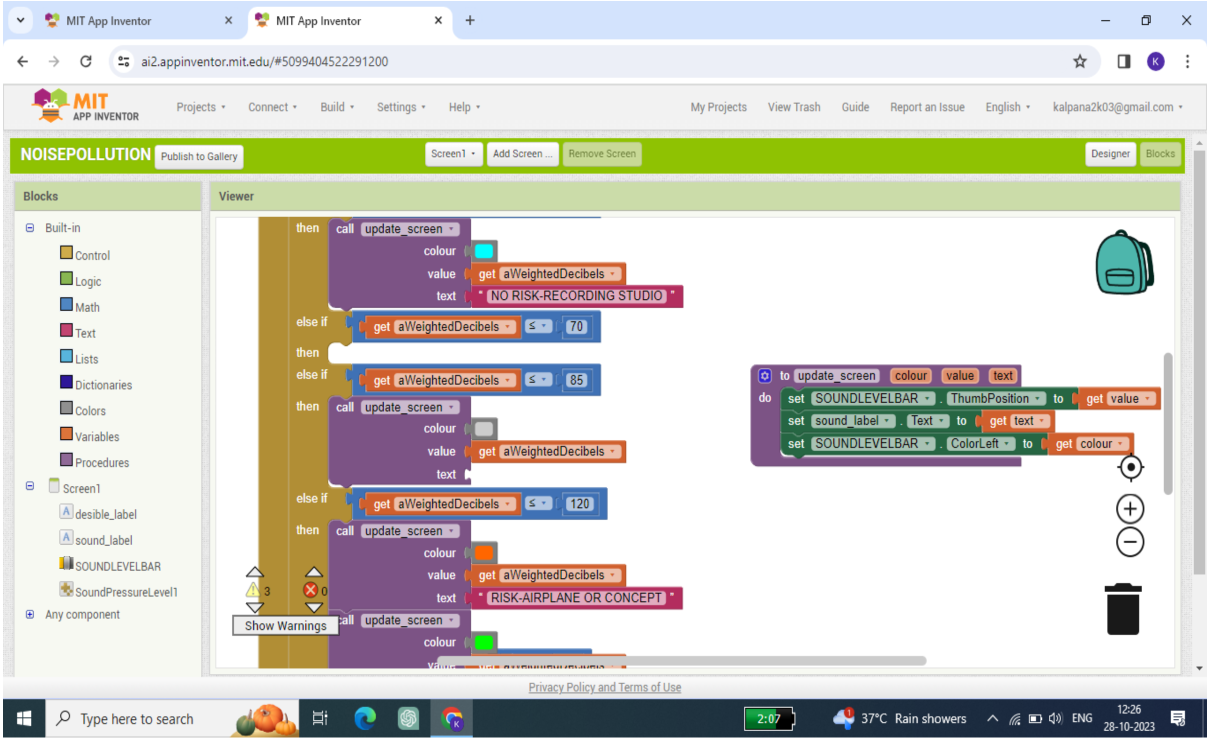
11. Data Analysis: You can develop basic data analysis features within the app, such as averaging noise levels over time or generating simple reports.

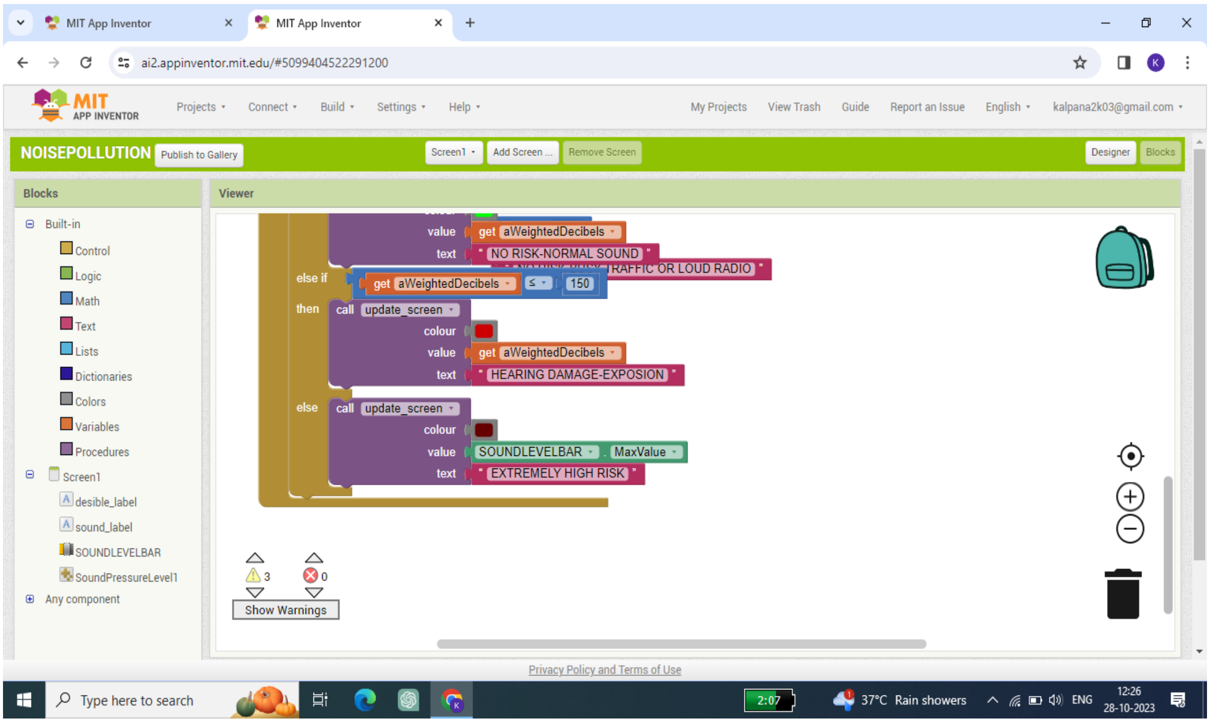
12. Deployment: Package your app as an APK file and distribute it through the Google Play Store or other distribution methods.



BLOCKS :







THESE CODE AND IMAGES ARE INCLUDED IN PHASE 4 : DEVELOPMENT PART 2

BY :

A.EZHILARASI (422621104012)