NOISE POLLUTION MONITORING

PHASE - 4

About Thing Speak:

ThingSpeak is an IoT analytics platform service that allows you to aggregate, visualize, and analyse live data streams in the cloud. The platform's real-time data management tools facilitate swift decision-making based on live information streams. Its popularity in the IoT landscape stems from the simplicity it offers in channel creation and data manipulation. Whether tracking environmental variables, controlling smart devices, or conducting research, ThingSpeak provides a robust framework. The collaborative nature of ThingSpeak encourages a community-driven approach, fostering the sharing of ideas and code snippets. Overall, ThingSpeak stands as a dynamic hub for IoT enthusiasts, amplifying the potential for innovation in the interconnected world of devices and data.

Code to display Real time Transit Information on Thinkspeak:

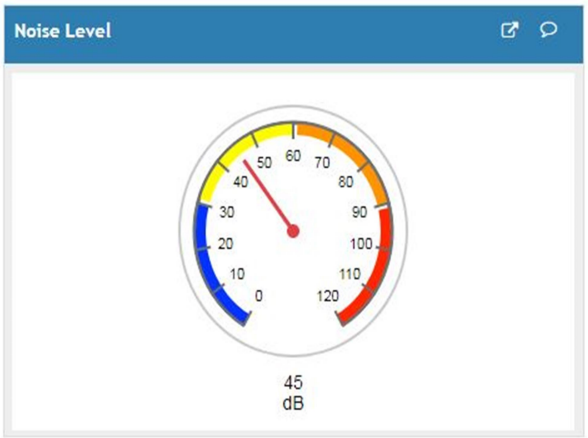
Noise_pollution.ino

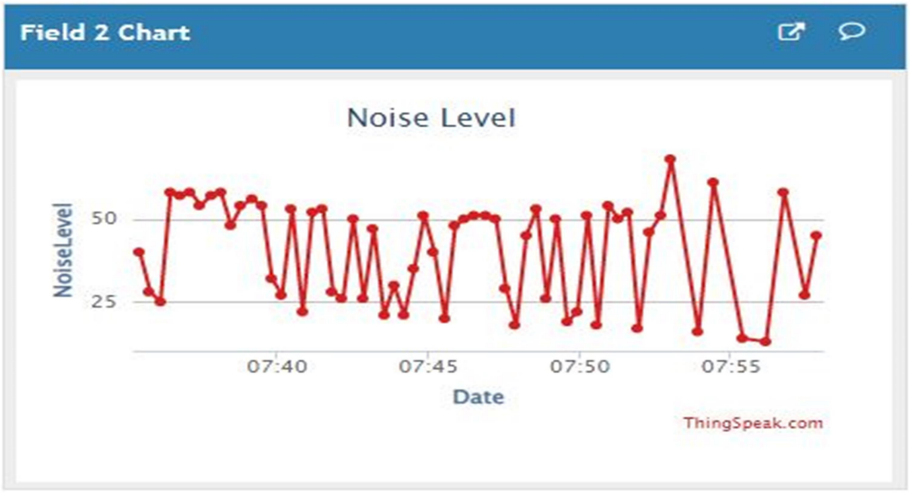
```
#include <ESP8266WiFi.h>
#include <SPI.h>
#include <Wire.h>
#include <Adafruit_GFX.h>
#include <Adafruit_SSD1306.h>

#define SCREEN_WIDTH 128    // OLED display width, in pixels
#define SCREEN_HEIGHT 64    // OLED display height, in pixels
#define OLED_RESET -1       // Reset pin # (or -1 if sharing search Arduino reset pin)
Adafruit_SSD1306 display(SCREEN_WIDTH, SCREEN_HEIGHT, &Wire, OLED_RESET);

String apiKey = "N2WXZ15RHOJED7VC"; // Enter your Write API key from ThingSpeak
const char *ssid = "Alexahome";     // replace with your wifi ssid and wpa2 key
const char *pass = "12345678";
const char* server = "api.thingspeak.com";

const int sampleWindow = 50;                              // Sample window width in mS (50 mS = 20Hz)
unsigned int sample;_

WiFiClient client;

void setup()
{
  Serial.begin(115200);                                    //Serial comms for debugging
  display.begin(SSD1306_SWITCHCAPVCC, 0x3C);              //OLED display start
  display.display();                                      //show buffer
  display.clearDisplay();                                 //clear buffer
  display.setTextSize(1);                                 //Set text size to 1 (1-6)
  display.setTextColor(WHITE);                            //Set text color to WHITE (no choice lol)
  display.setCursor(0,0);                                 //cursor to upper left corner
  display.println("Decibelmeter");                 //write title
  display.display();                                      //show title
  delay(2000);                                            //wait 2 seconds

  WiFi.begin(ssid, pass);
```

```
36
37    while (WiFi.status() != WL_CONNECTED)
38    {
39      delay(500);
40      Serial.print(".");
41    }
42      Serial.println("");
43      Serial.println("WiFi connected");
44
45      display.clearDisplay();
46      display.setCursor(0,0);
47      display.setTextSize(1);
48      display.setTextColor(WHITE);
49      display.print("WiFi connected");
50      display.display();
51      delay(4000);
52      display.clearDisplay();
53  }
54
55  void loop()
56  {
57    unsigned long startMillis= millis();                  // Start of sample window
58    float peakToPeak = 0;                                 // peak-to-peak level
59
60    unsigned int signalMax = 0;                           //minimum value
61    unsigned int signalMin = 1024;                        //maximum value
62
63                                                          // collect data for 50 mS
64    while (millis() - startMillis < sampleWindow)
65    {
66      sample = analogRead(0);                             //get reading from microphone
67      if (sample < 1024)                                  // toss out spurious readings
68      {
69        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
        float db = map(peakToPeak,20,900,49.5,90);      //calibrate for deciBels
        display.setCursor(0,0);                         //cursor to upper left
        display.setTextSize(2);                         //set text size to 2
        display.print(db);                              //write calibrated deciBels
        display.print(" dB");                           //write units


         for(int x =5;x<114;x=x+6){                     //draw scale
           display.drawLine(x, 32, x, 27, WHITE);
         }
        display.drawRoundRect(0, 32, 120, 20, 6, WHITE);  //draw outline of bar graph
        int r = map(db,0,120,1,120);                    //set bar graph for width of screen
        display.fillRoundRect(1, 33, r, 18, 6, WHITE);  //draw bar graph with a width of r
        display.display();                              //show all that we just wrote & drew
        display.clearDisplay();                         //clear the display

        if (client.connect(server, 80)) // "184.106.153.149" or api.thingspeak.com
       {
         String postStr = apiKey;
         postStr += "&field1=";
         postStr += String(db);
         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();
         delay(150);
    }
```

OUTPUT:





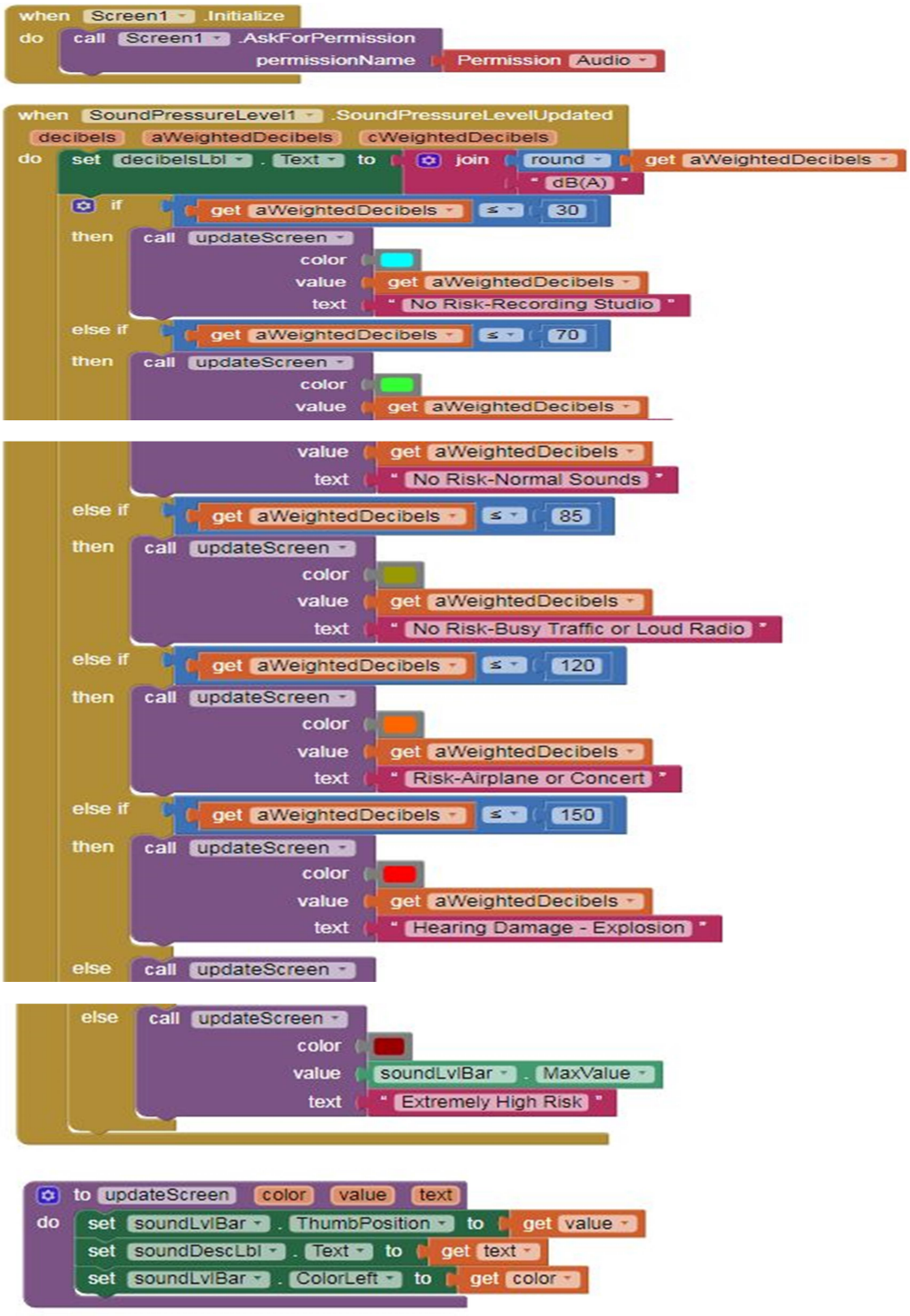
MOBILE APPLICATION:

Additionally, MIT App Inventor provides a simple and intuitive interface for testing and debugging apps in real-time. It emphasizes accessibility by enabling users to create functional apps with minimal effort. The platform supports a variety of features, including sensors, media playback, and connectivity options, allowing users to explore diverse app functionalities. MIT App Inventor serves as a valuable resource for individuals looking to kickstart their journey into mobile app development, fostering creativity and hands-on learning.

For collecting and storing real-time transit data, ThingSpeak channels were employed, capturing information like route details, location, and passenger count for each data point. In the MIT App Inventor, a visual interface facilitated app development, utilizing components like labels, text boxes, and web

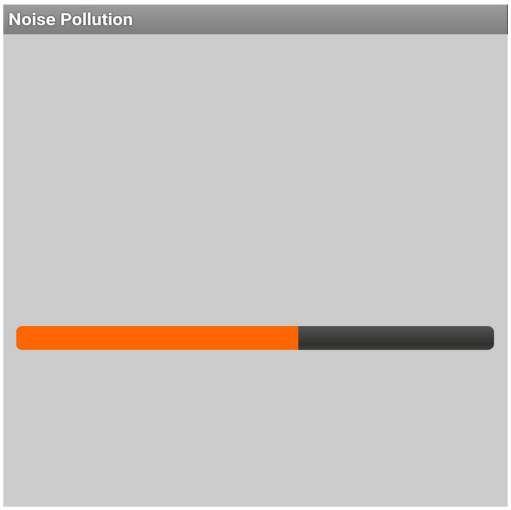
elements to shape the user interface. To retrieve data, the app was configured to interact with ThingSpeak channels through the Web component, making API calls for the latest entries. The app dynamically displayed the retrieved data, presenting route information, location, and passenger count in real-time on its interface.

MIT APP INVENTOR:

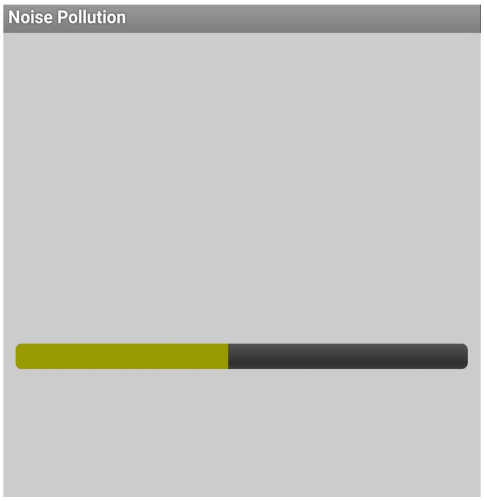


OUTPUT :

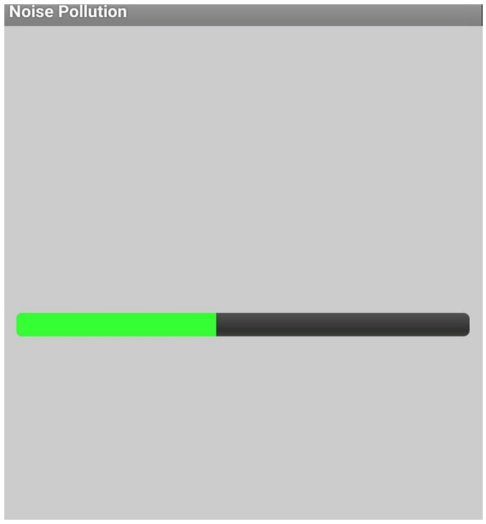
When noise level <= 120 dB



When noise level <= 85dB



When noise level <= 70dB



TEAM MEMBERS:

1. Jayanth N - (2021504523)
2. Lavanya V - (2021504525)
3. Mathana K - (2021504528)
4. Sudhanthira P - (2021504548)
5. Vidhya SS - (2021504557)