NOISE POLLUTION MONITORING

Project objective:

The IoT-based noise pollution project aims to collect real-time data on noise levels in various locations and use that information for analysis and potential mitigation. This project typically involves the deployment of sensors, data processing, and reporting mechanisms.

The objective of an IoT (Internet of Things) based noise pollution project is to monitor, analyze, and potentially mitigate noise pollution in a given environment using connected sensors and devices. Noise pollution can have adverse effects on human health, well-being, and the environment, and an IoT-based solution can provide valuable insights and tools to address this issue.

IOT Device setup:

- **Deployment of sensors:**
  Sound sensors like **SMT32** in Arduino simulator are strategically placed in different areas of the city or region under study. These sensors capture sound levels and can differentiate between various types of noise sources, such as traffic, construction, or industrial activity
- **Integrate data:**
  The noise sensors collect data continuously and transmit it to a central server or cloud platform using wireless communication protocols like Wi-Fi, cellular. This ensures real-time monitoring capabilities. The collected noise data is processed and analyzed to identify noise trends, patterns, and sources of excessive noise pollution.
- **Monitoring :**
  If the sound detected by the sensor exceeds a certain threshold, the system will send out a notification. Implement a system of alerts that can inform the appropriate authorities when noise levels rise. Explore and put into effect techniques for reducing noise pollution, such as

noise barriers, adaptive traffic management, or changing construction methods. IoT data can be used to assess the effectiveness of these tactic.

Wokwi Simulator:

The virtual sensors to run the application by utilizing the Wokwi simulator. Wokwi supports a wide range of popular microcontrollers, such as Arduino, ESP8266, and Raspberry Pi Pico.

Main.py:

```
import machine

import time
import urequests
import ujson
import network
import math

#Define Wi-Fi credentials
wifi_ssid = 'Wokwi-GUEST'
wifi_password = ''

#Connect to Wi-Fi
wifi = network.WLAN(network.STA_IF)
wifi.active(True)
wifi.connect(wifi_ssid, wifi_password)

#Wait for Wi-Fi connection
while not wifi.isconnected():
  pass

#Define ultrasonic sensor pins (Trig and Echo pins)
ultrasonic_trig = machine.Pin(15, machine.Pin.OUT)
ultrasonic_echo = machine.Pin(4, machine.Pin.IN)
#Define microphone pin
microphone = machine.ADC(2)
calibration_constant = 2.0
noise_threshold = 60
#Firebase Realtime Database URL and secret
firebase_url = 'https://noise-pollution-f7094-default-rtdb.firebaseio.com'
firebase_secret = 'AIzaSyAB1DL1O9_Gb6yXW9yVWvnNvwFchlaiFLE'
def measure_distance():
  #Trigger the ultrasonic sensor
  ultrasonic_trig.value(1)
   time.sleep_us(10)
  ultrasonic_trig.value(0)
```

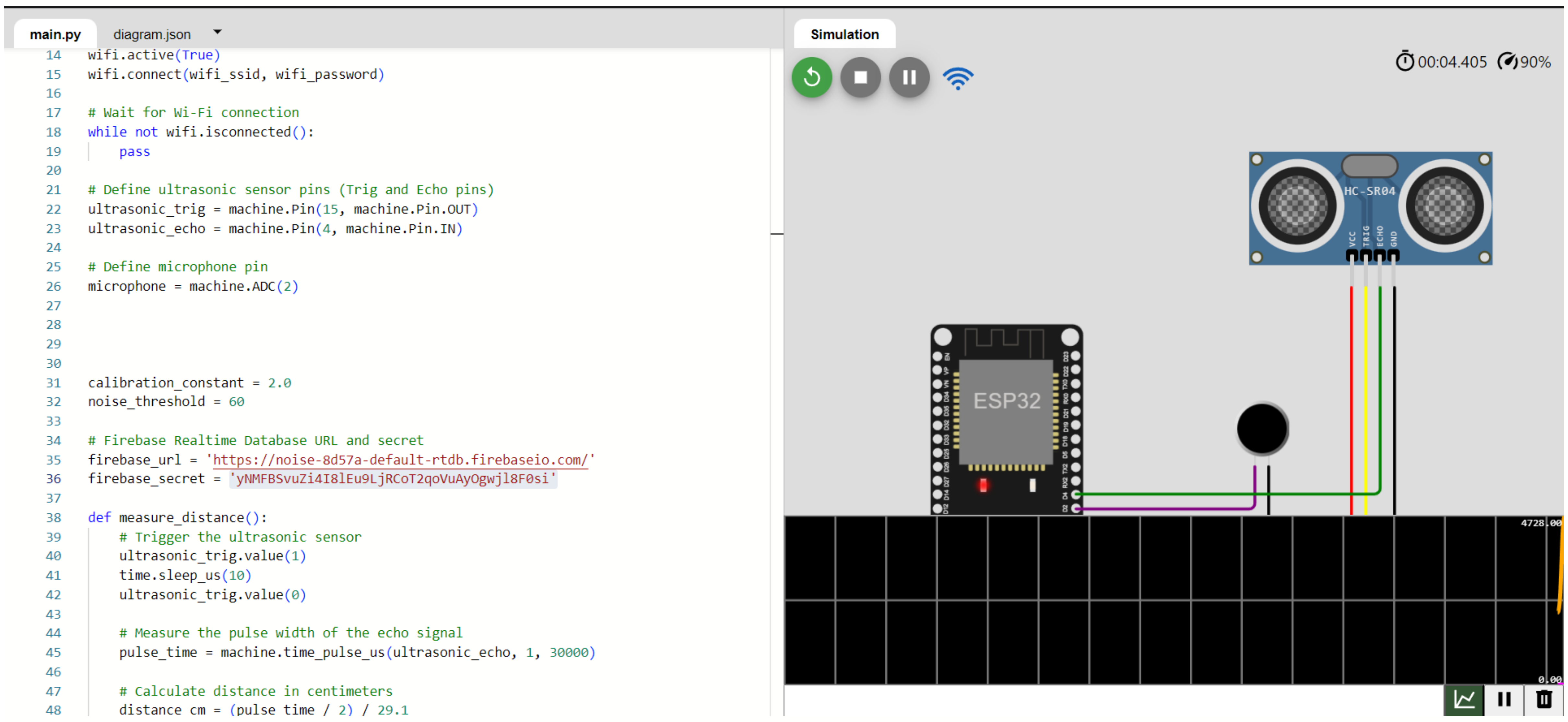
```
    # Measure the pulse width of the echo signal
    pulse_time = machine.time_pulse_us(ultrasonic_echo, 1, 30000)
    # Calculate distance in centimeters
    distance_cm = (pulse_time / 2) / 29.1
    return distance_cm
def measure_noise_level():
    # Read analog value from the microphone
    noise_level = microphone.read()
    noise_level_db = 20 * math.log10(noise_level / calibration_constant)
    return noise_level, noise_level_db
# Function to send data to Firebase
def send_data_to_firebase(distance, noise_level_db):
    data = {
        "Distance": distance,
        "NoiseLevelDB": noise_level_db
    }
    url = f'{firebase_url}/sensor_data.json?auth={firebase_secret}'

    try:
        response = urequests.patch(url, json=data)
        if response.status_code == 200:
            print("Data sent to Firebase")
        else:
            print(f"Failed to send data to Firebase. Status code: {response.status_code}")
    except Exception as e:
        print(f"Error sending data to Firebase: {str(e)}")
try:
    while True:
        distance = measure_distance()
        noise_level, noise_level_db = measure_noise_level()
        print("Distance: {} cm, Noise Level: {:.2f} dB".format(distance, noise_level_db))
        if noise_level_db > noise_threshold:
            print("Warning: Noise pollution exceeds threshold!")
        # Send data to Firebase
        send_data_to_firebase(distance, noise_level_db)
        time.sleep(1)  # Adjust the sleep duration as needed
except KeyboardInterrupt:
    print("Monitoring stopped")
```

diagram.json:

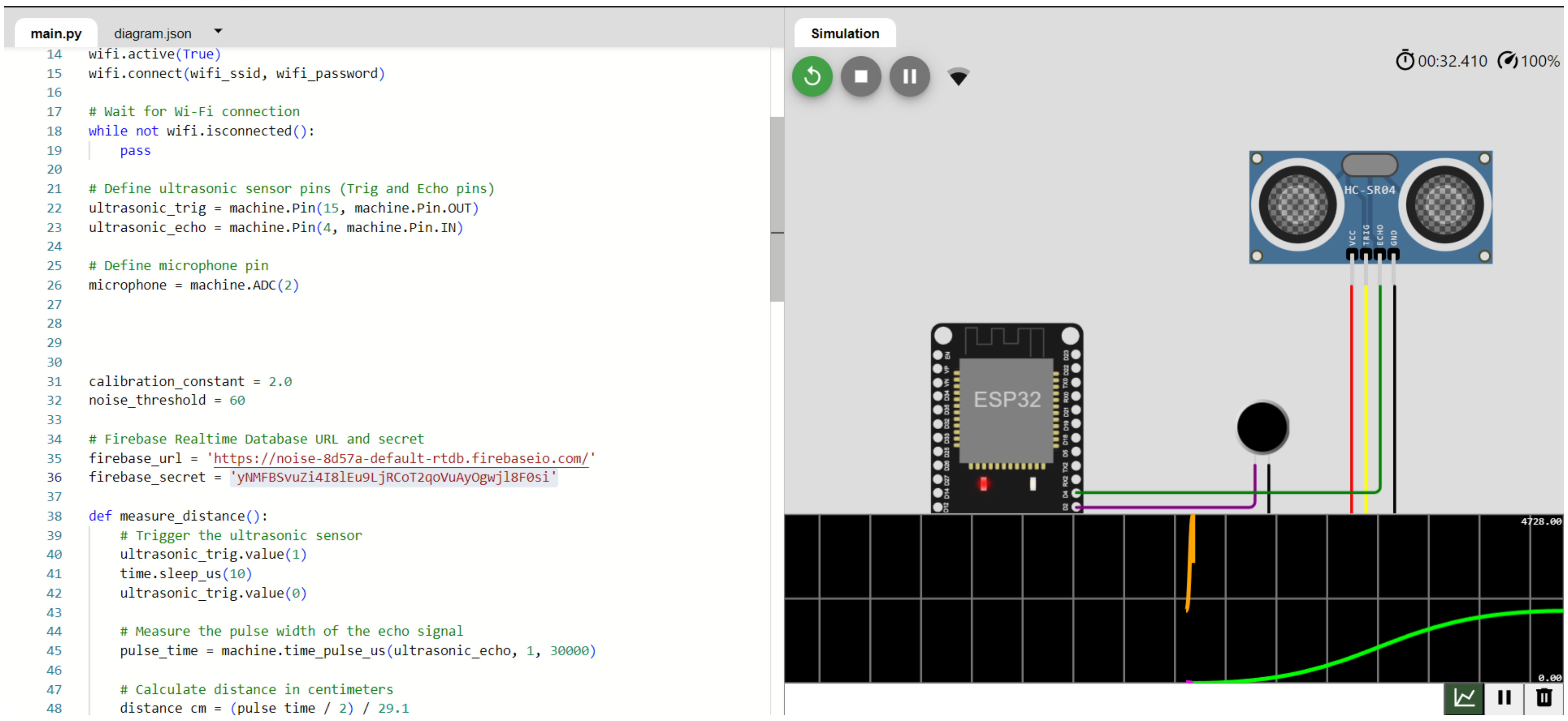

```
{
"version": 1,
"author": "chinnumishel",
"editor": "wokwi",
"parts": [
 {
  "type": "wokwi-esp32-devkit-v1",
```

```
      "id": "esp",
      "top": -72.1,
      "left": 52.6,
      "attrs": { "env": "micropython-20231005-v1.21.0" }
    },
    { "type": "wokwi-microphone", "id": "mic", "top": -16.98, "left": 263.79, "attrs": {} },
    {
      "type": "wokwi-hc-sr04",
      "id": "ultrasonic1",
      "top": -190.5,
      "left": 274.3,
      "attrs": { "distance": "88" }
    }
  ],
  "connections": [
    [ "esp:TX0", "$serialMonitor:RX", "", [] ],
    [ "esp:RX0", "$serialMonitor:TX", "", [] ],
    [ "mic:1", "esp:D2", "purple", [ "v0" ] ],
    [ "mic:2", "esp:GND.1", "black", [ "v0" ] ],
    [ "ultrasonic1:VCC", "esp:3V3", "red", [ "v0" ] ],
    [ "ultrasonic1:TRIG", "esp:D15", "yellow", [ "v0" ] ],
    [ "ultrasonic1:ECHO", "esp:D4", "green", [ "v0" ] ],
    [ "ultrasonic1:GND", "esp:GND.1", "black", [ "v0" ] ]
  ],
  "serialMonitor": { "display": "plotter" },
  "dependencies": {}
}
```



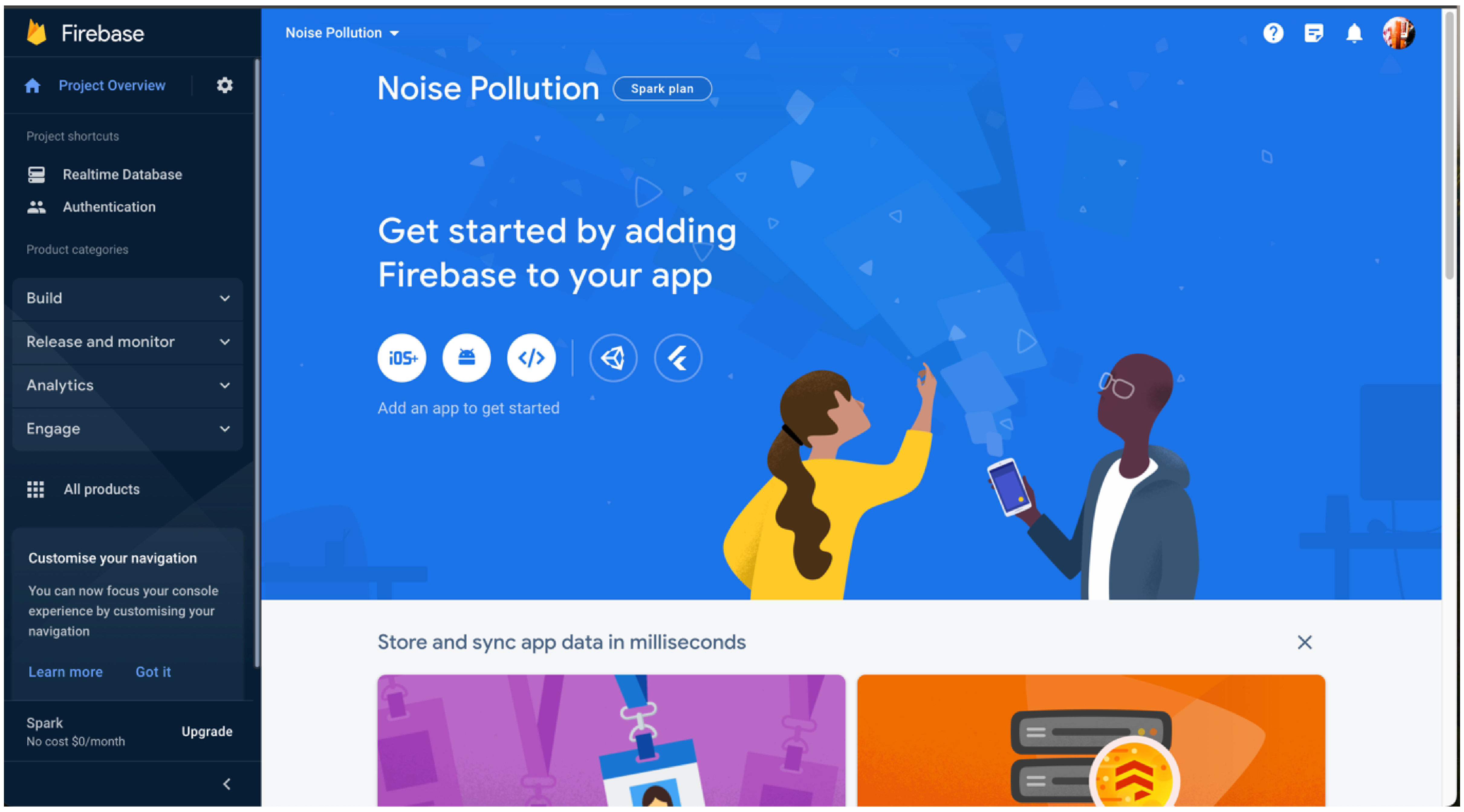
Wokwi platform address:

https://wokwi.com/projects/378838945740627969

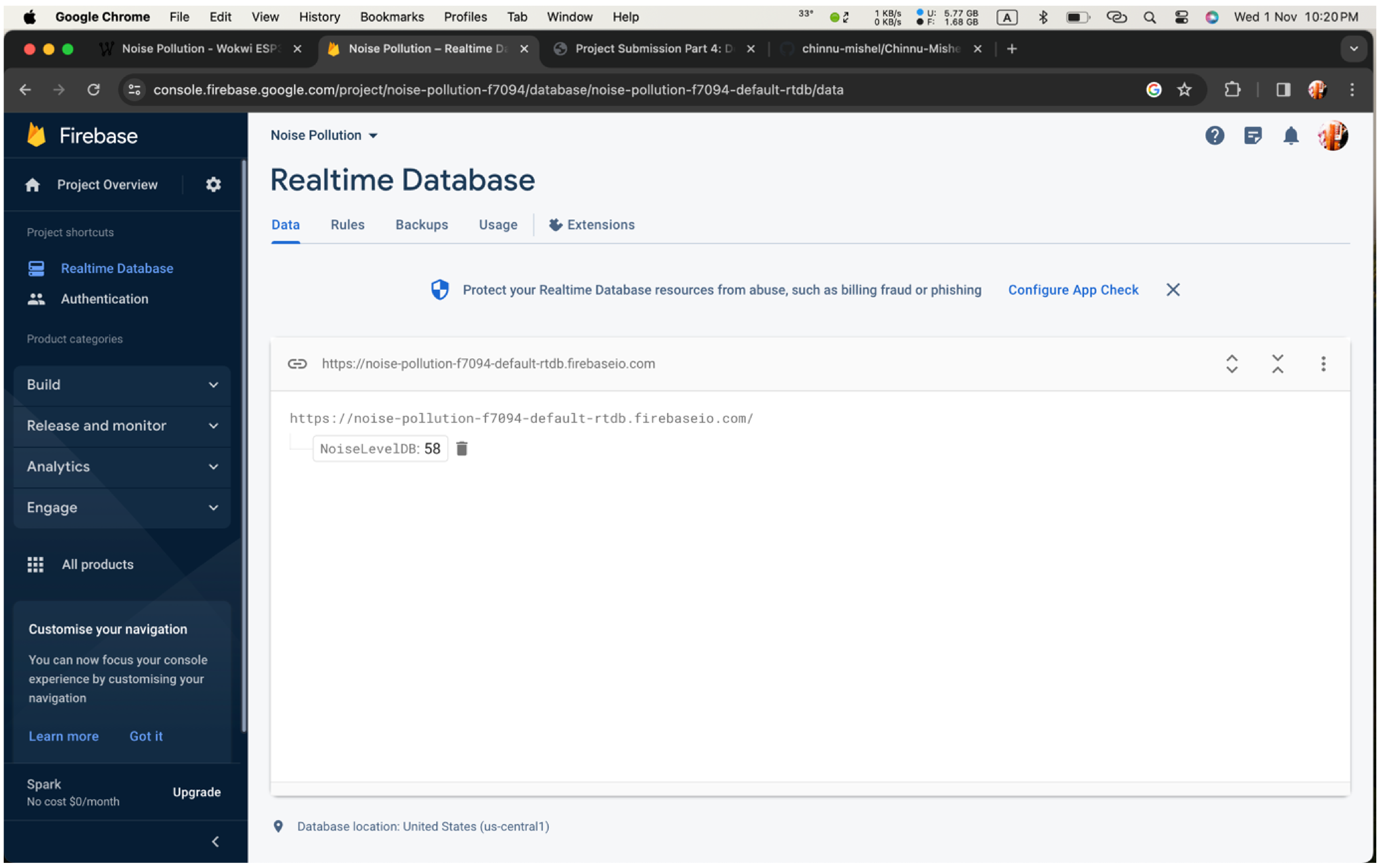


Firebase platform:

- Firebase is a mobile and web application development platform that offers various tools and services like
  Real-time Data Sync to maintain a real-time connection between the server and clients. This means that any changes made to the database are instantly reflected in all connected clients without the need for manual refreshes.



Implement a real-time database and connect to our wokwi simulator to collect data.



The realtime database of firebase that linked to wokwi simulator by

firebase_url = 'https://noise-pollution-f7094-default-rtdb.firebaseio.com'

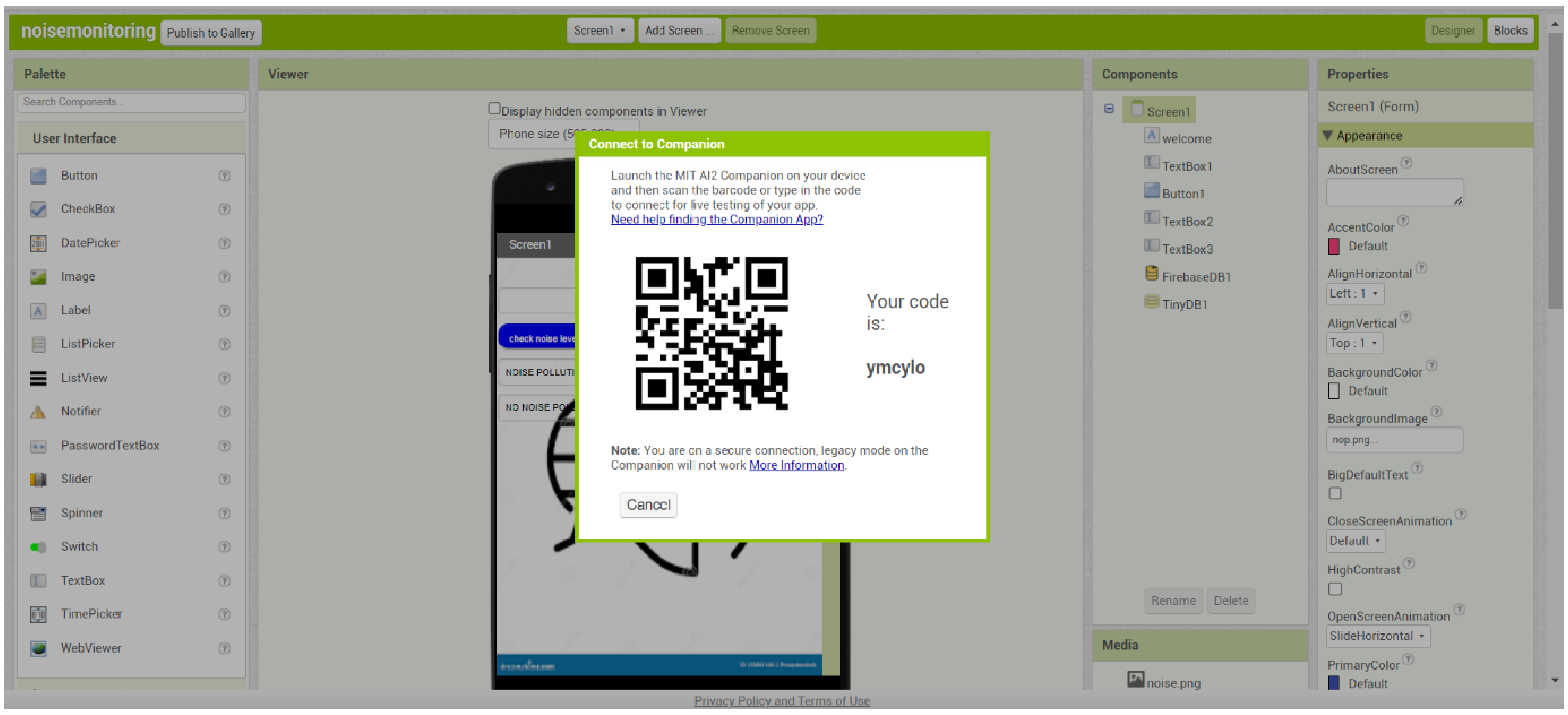
firebase_secret = 'AIzaSyAB1DL1O9_Gb6yXW9yVWvnNvwFchlaiFLE'

APP DEVELOPMENT:

Utilizing "MIT 21 app inventor," create a mobile application.

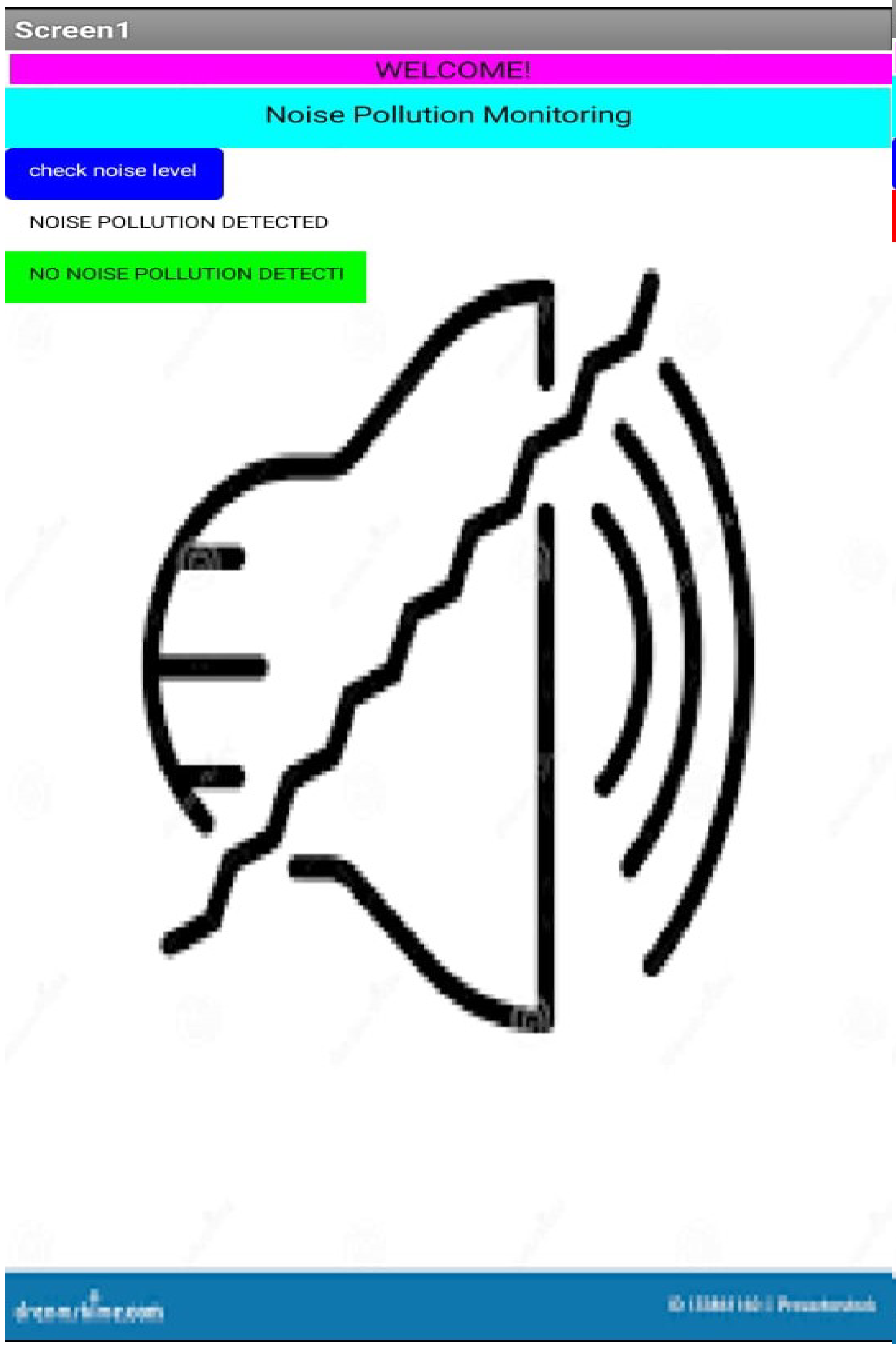
MIT app inventor: MIT App Inventor is a visual programming environment that allows people with little to no programming experience to create mobile apps for Android devices. App Inventor uses a block-based coding approach, where you can visually drag and drop components and program them using a set of predefined blocks, making app development accessible to a wider audience.

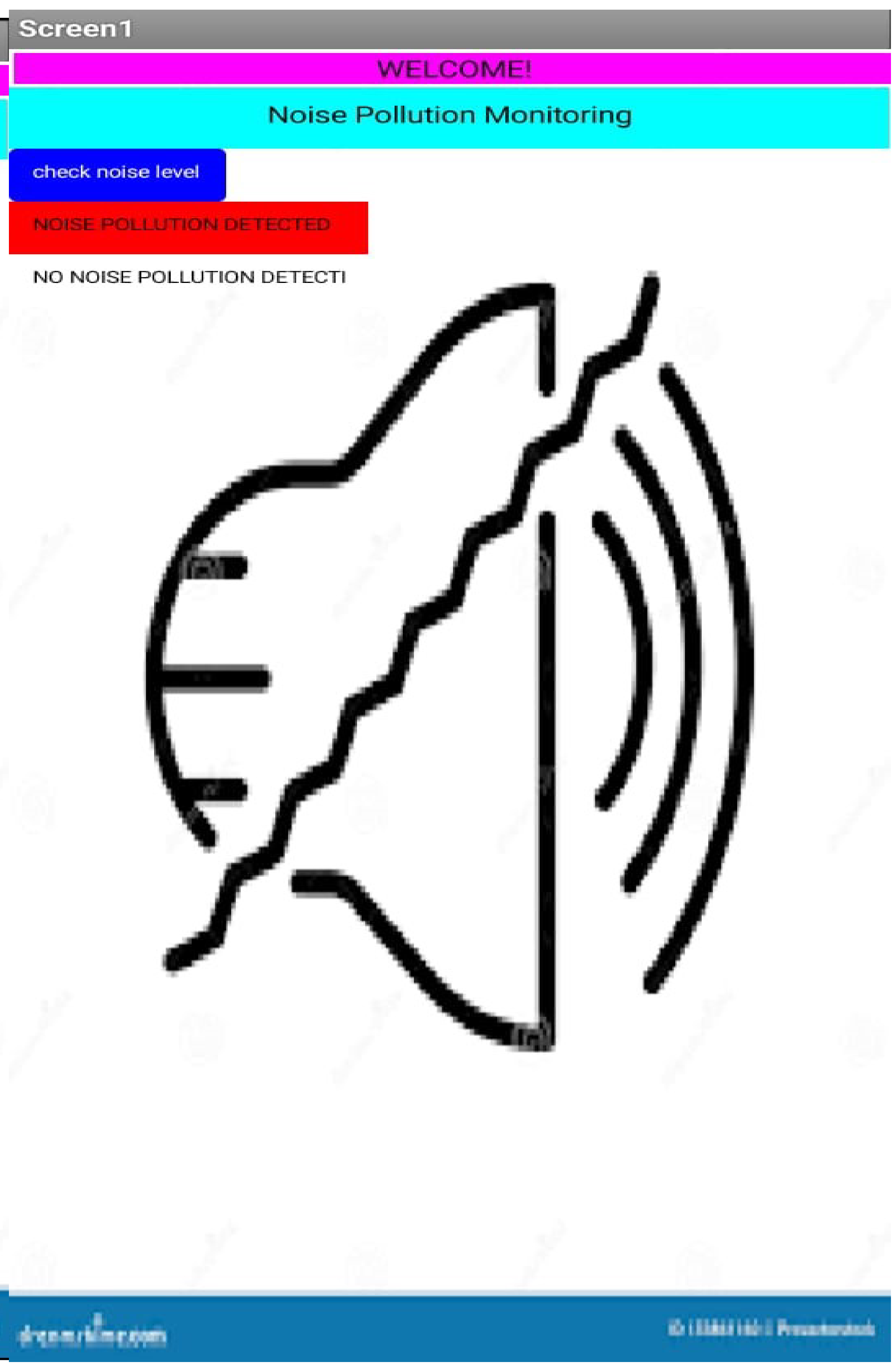
Using "Mit App Inventor," we can use our generated application on our mobile device by either scanning the QR code or entering the six digits.



Download the application as android application and download in mobile phone and check the application. This application is used to monitor the noise level in the place where the sensor deployed.

Output:


Screen1
WELCOME!
Noise Pollution Monitoring
check noise level
NOISE POLLUTION DETECTED
NO NOISE POLLUTION DETECTI


Screen1
WELCOME!
Noise Pollution Monitoring
check noise level
NOISE POLLUTION DETECTED
NO NOISE POLLUTION DETECTI