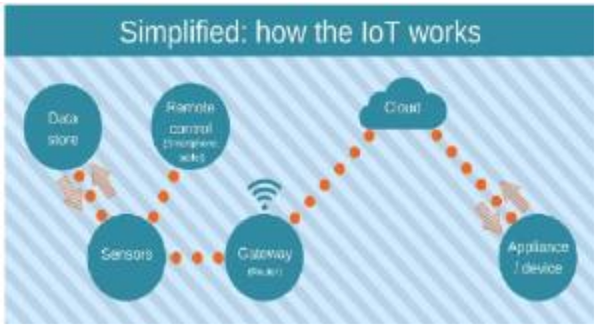
NOISE POLLUTION MONITORING

PROJECT DESCRIPTION:

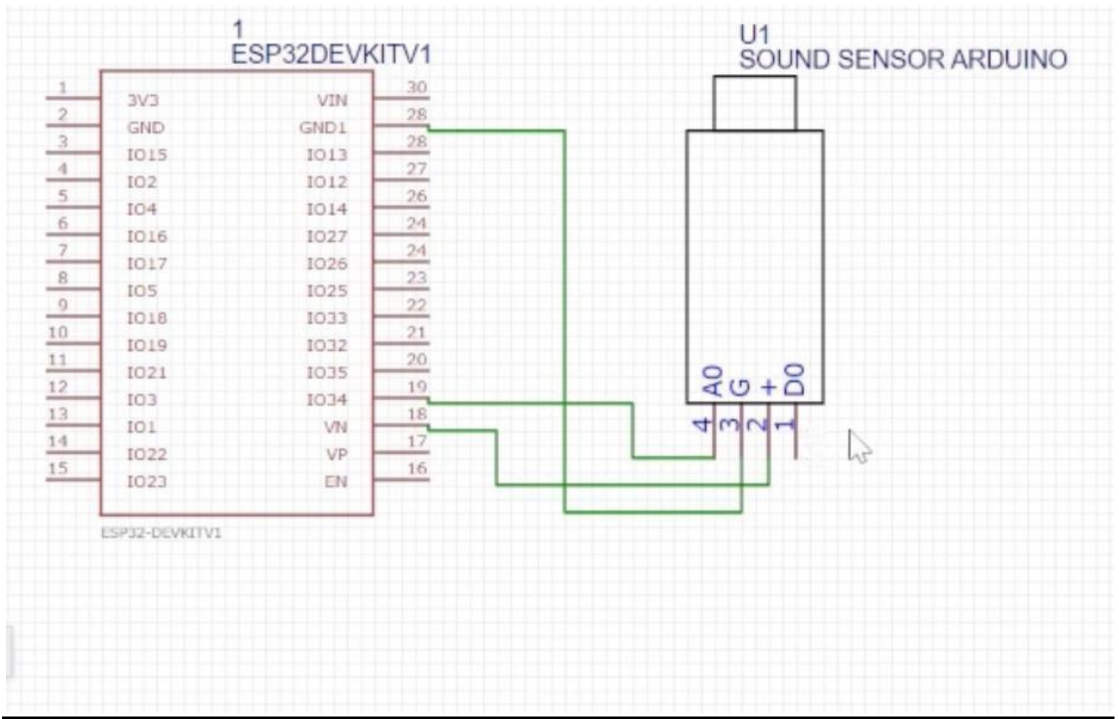
Noise pollution has become a very big issue around the world. The adverse effects of this pollution include hearing impairment, negative social behavior annoyance, sleep disturbance and intelligibility to understand people's speech. In learning context, noise can affect understanding and behavior of people and places with high noise level are not suitable for learning and teaching process. Internet of Things (IoT) technology is one of the best choices to monitor the noise or sound intensity in the environment for the safety of human being.
The aim of this paper is to deliver a development of an IoT based noise monitoring system comprises of a sound sensor, an IoT platform called NodeMCU, LCD and LEDs. The system will provide a real-time alert if the noise exceeds the threshold noise limit set by Environmental Department of Health standard. Equipped with an Android application, the data from the sound sensor will be transferred into the cloud server and subsequently transferred into the app for display and to enable remote monitoring.

IMPLEMENTATION:

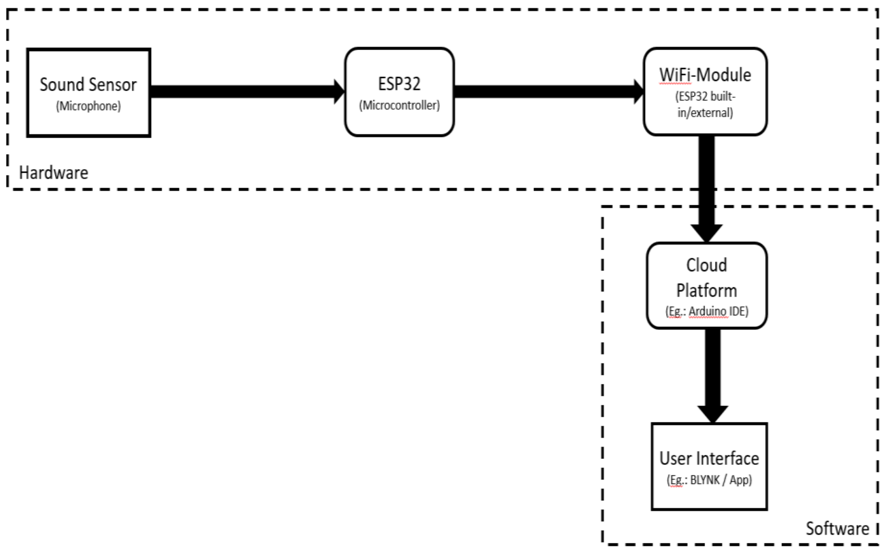

Simplified: how the IoT works
Data store
Remote control (smartphone, tablet)
Cloud
Sensors
Gateway (Router)
Appliance / device

Sound sensor or mic sensor provides digital output and it detects sound from atmosphere. A WiFi module is also connected to Arduino and it is used to transfer data from the sensors to cloud server. ESP8266 WiFi module is used to store the data to online server. The data from sensor are basically analog signal so analog to digital converter (ADC) is used to convert the data. 16 x 2 Liquid crystal display (LCD) is used to display the measured value from the sensors. It can display two lines and each line has 16 characters.

Circuit diagram:



Block diagram:



Source code:

4.1 Front End code:

```
import SwiftUI

struct ContentView: View {
    @State private var isMonitoring = false
    @State private var noiseLevel = 0.0

    var body: some View {
        VStack {
            Text("Current Noise Level: \(noiseLevel) dB")
                .font(.largeTitle)

            Button(action: {
                // Start/stop monitoring code here
                self.isMonitoring.toggle()
            }) {
                Text(isMonitoring ? "Stop Monitoring" : "Start Monitoring")
                    .padding()
                    .background(Color.blue)
                    .foregroundColor(.white)
                    .cornerRadius(10)
            }
        }
    }
}

@main
```

```
struct NoiseApp: App {
    var body: some Scene {
        WindowGroup {
            ContentView()
        }
    }
}
```

xml

```
<!-- activity_main.xml layout file -->
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:orientation="vertical"
    android:gravity="center">

    <TextView
        android:id="@+id/noiseLevelTextView"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Current Noise Level: 0 dB"
        android:textSize="24sp" />

    <Button
        android:id="@+id/startStopButton"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Start Monitoring" />

</LinearLayout>
```

kotlin

```
// MainActivity.kt
```

```
import android.os.Bundle
import android.view.View
import android.widget.Button
import android.widget.TextView
import androidx.appcompat.app.AppCompatActivity

class MainActivity : AppCompatActivity() {
    private lateinit var noiseLevelTextView: TextView
    private lateinit var startStopButton: Button
    private var isMonitoring = false

    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        setContentView(R.layout.activity_main)

        noiseLevelTextView = findViewById(R.id.noiseLevelTextView)
        startStopButton = findViewById(R.id.startStopButton)

        startStopButton.setOnClickListener {
            // Start/stop monitoring code here
            isMonitoring = !isMonitoring
            updateUI()
        }
    }

    private fun updateUI() {
        if (isMonitoring) {
            startStopButton.text = "Stop Monitoring"
        } else {
            startStopButton.text = "Start Monitoring"
        }
    }
}
```

4.2.Back End code:

```
    import sounddevice as sd
import numpy as np
import paho.mqtt.client as mqtt
import time

# MQTT settings
mqtt_broker_address = "your_broker_address"
mqtt_port = 1883
mqtt_topic = "noise_level"

# Sampling parameters
sample_rate = 44100  # Samples per second
duration = 10  # Recording duration in seconds

# Function to calculate dB from audio data
def calculate_db(audio_data):
    rms = np.sqrt(np.mean(np.square(audio_data)))  # Root Mean Square
    db = 20 * np.log10(rms / 0.0002)  # Reference sound pressure level
    return db

# MQTT callback functions
def on_connect(client, userdata, flags, rc):
    if rc == 0:
        print("Connected to MQTT broker")
    else:
        print("Connection to MQTT broker failed")

def on_publish(client, userdata, mid):
    print("Data published to MQTT")

# Create MQTT client
client = mqtt.Client()
```

```
client.on_connect = on_connect
client.on_publish = on_publish

# Connect to MQTT broker
client.connect(mqtt_broker_address, mqtt_port)

# Start recording and publishing
with sd.InputStream(callback=calculate_db, channels=1, samplerate=sample_rate):
    print("Monitoring noise level...")
    while True:
        db = calculate_db(np.random.random_sample(44100))  # Simulated audio data for testing
        client.publish(mqtt_topic, str(db))
        time.sleep(60)  # Publish data every minute (adjust as needed)
```

Sample output screenshot:

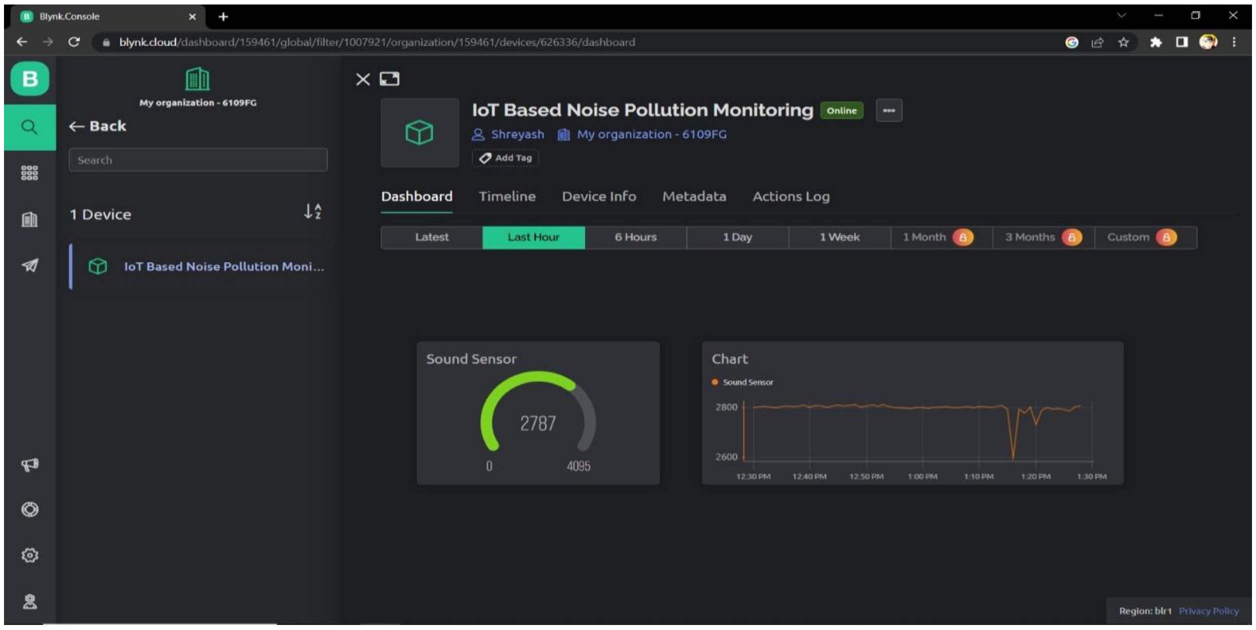


Fig4.1.1initial state

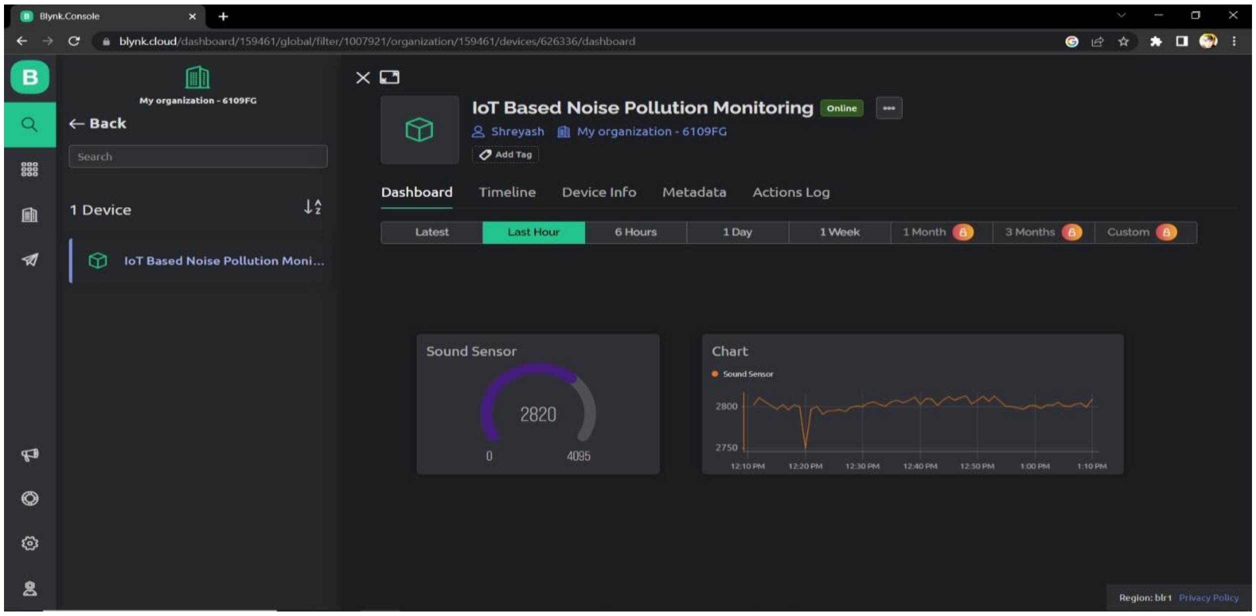


Fig 4.1.2 processing

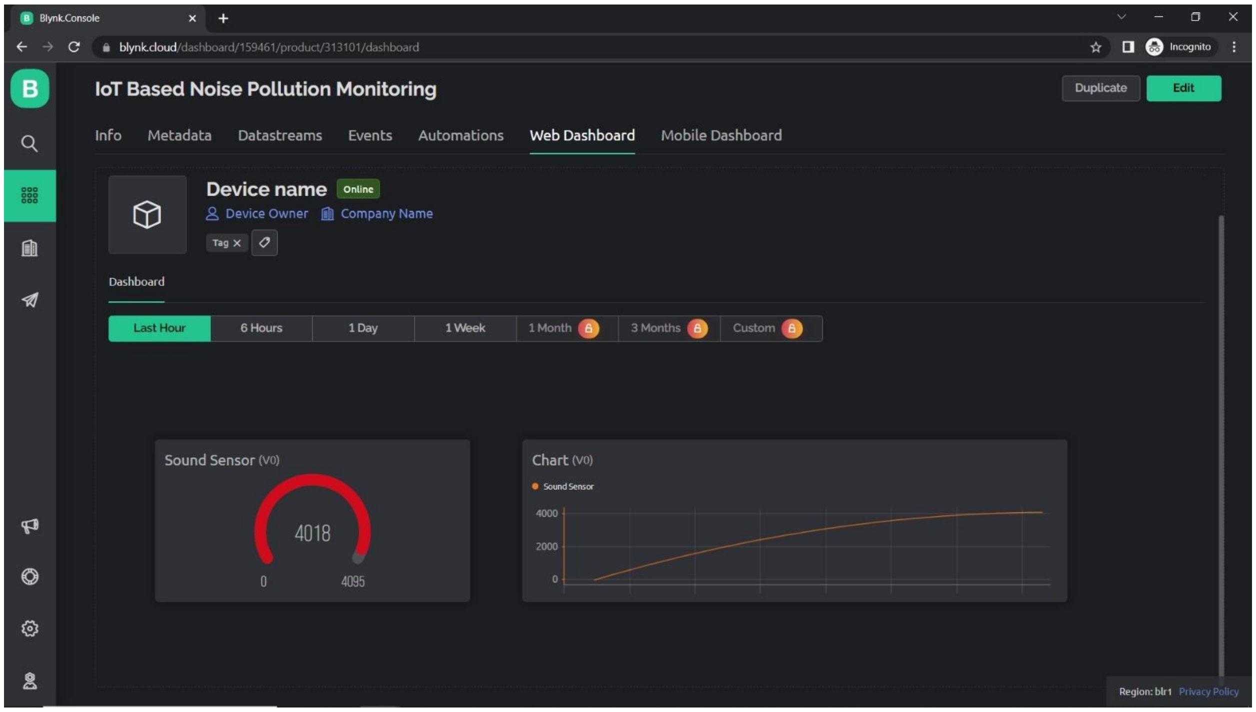


Fig.4.1.3.End of the process

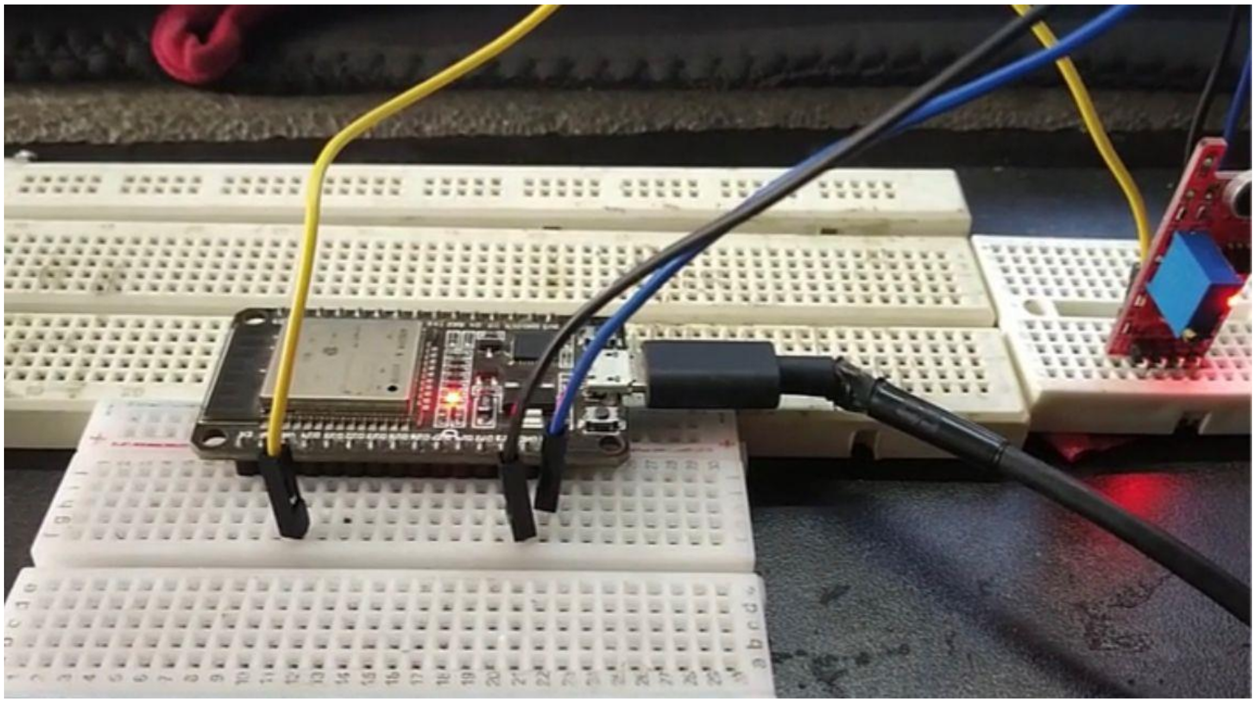


Fig.4.1.4. sensor board

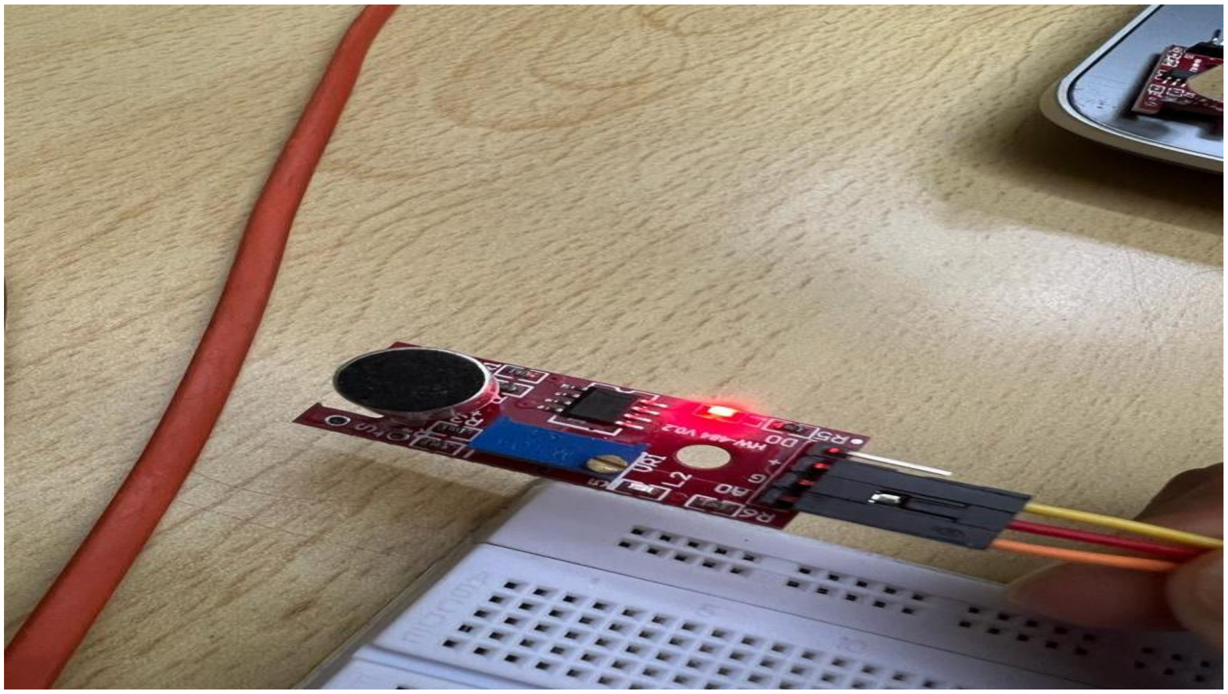


Fig.4.1.5 circuit board

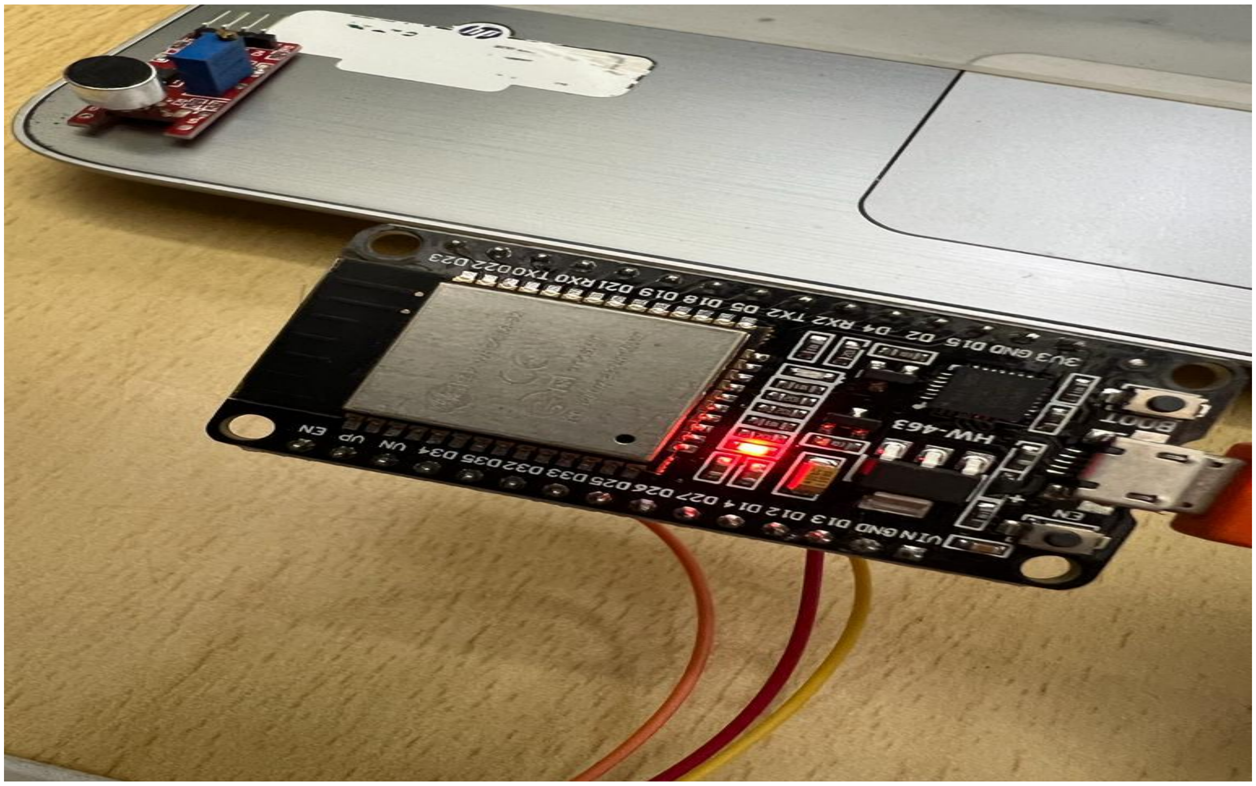


Fig.4.1.6 Over all circuit board