Phase 5

ENVIRONMENTAL MONITORING IN PARKS

Final Documentation

Problem Description:

The description or scope of our problem is to monitor and measure the parameters that decides the weather of the environment to be a pleasant one (i.e) Temperature and Humidity values in parks and to share those values to the people to plan their occasion accordingly for their outdoor activities.

Solution Overview:

The solution is decided and executed in an effective way so that each and every user will use it without any limit. Our project idea is based on the IoT devices and to send the data through the website created by using html, css and js script. This website is exposed by public by means of QR code and a social media platform

IoT Devices:

The following IoT components are used for our project are described below:

- ESP32 Microcontroller.
- DHT22 sensor for temperature and humidity.
- Resistor 10k ohm for accuracy.
- Bread board if needed.
- Power source.

This project-kit is cost effective and management free kit since we used minimal number of components for monitoring the weather data of the park.

Device Setup:

The above components are connected to the corresponding pins as follows:

- Connect the VCC pin of DHT22 sensor to the 3V3 pin of ESP32 controller on the breadboard.

- Connect the ground pin of both the DHT22 and the ESP32 on the bread board.
- Connect the SDA pin of the resistor one pin and connect the other pin of the resistor to the 3V3 pin of the ESP32.
- Give the power supply accordingly to the circuit accordingly.

Circuit Diagram:

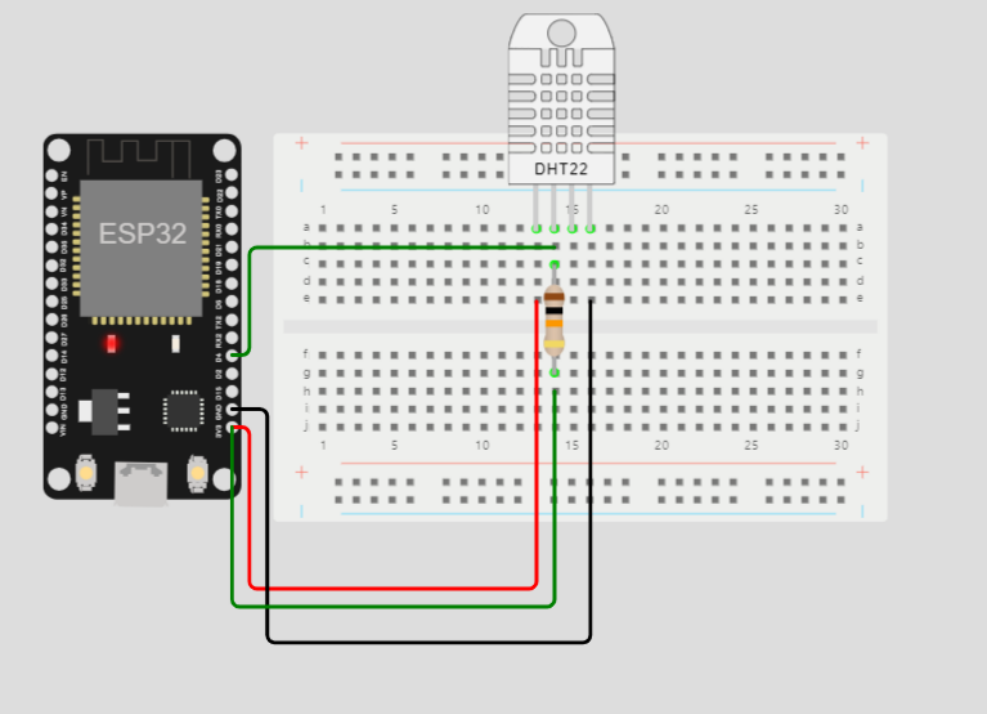


Diagram representation Code:

Here is the json code for the project on the wokwi platform.

```
{
  "version": 1,
  "author": "Anonymous maker",
  "editor": "wokwi",
  "parts": [
    { "type": "wokwi-breadboard-half", "id": "bb1", "top": -70.2, "left": -
102.8, "attrs": {} },
    { "type": "wokwi-esp32-devkit-v1", "id": "esp", "top": -72.1, "left": -
225.8, "attrs": {} },
    { "type": "wokwi-dht22", "id": "dht1", "top": -134.1, "left": 23.4,
"attrs": {} },
    {
      "type": "wokwi-resistor",
      "id": "r1",
      "top": 24,
```

```
      "left": 18.65,
      "rotate": 90,
      "attrs": { "value": "10000" }
    }
  ],
  "connections": [
    [ "esp:TX0", "$serialMonitor:RX", "", [] ],
    [ "esp:RX0", "$serialMonitor:TX", "", [] ],
    [ "bb1:13t.e", "esp:3V3", "red", [ "v153.6", "h-153.6", "v-86.4" ] ],
    [ "bb1:16t.e", "esp:GND.1", "black", [ "v182.4", "h-172.8", "v-124.8",
"h0" ] ],
    [ "esp:3V3", "bb1:14b.h", "green", [ "v96", "h162.9" ] ],
    [ "bb1:14t.b", "esp:D4", "green", [ "v0", "h-163.2", "v57.6" ] ],
    [ "dht1:VCC", "bb1:13t.a", "", [ "$bb" ] ],
    [ "dht1:SDA", "bb1:14t.a", "", [ "$bb" ] ],
    [ "dht1:NC", "bb1:15t.a", "", [ "$bb" ] ],
    [ "dht1:GND", "bb1:16t.a", "", [ "$bb" ] ],
    [ "r1:1", "bb1:14t.c", "", [ "$bb" ] ],
    [ "r1:2", "bb1:14b.g", "", [ "$bb" ] ]
  ],
  "dependencies": {}
}
```

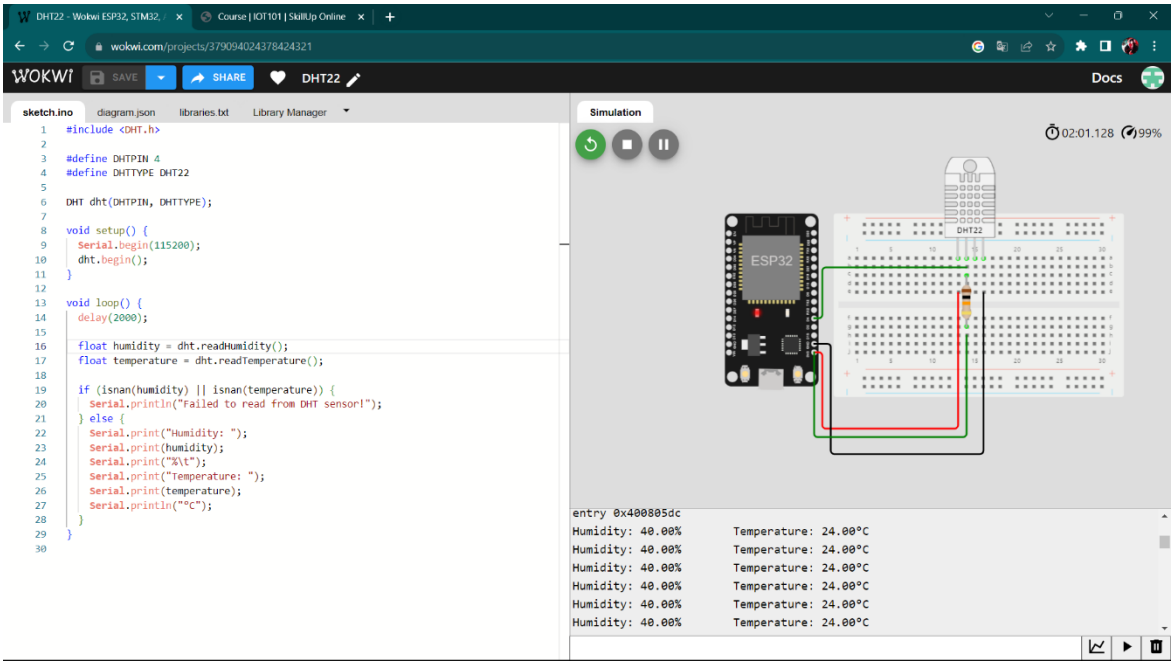
Simulation code:

Here is the simulation code for testing the circuit. The following libraries must be installed in order to simulate the code

✓ DHT sensor library.

```
#include <DHT.h>

#define DHTPIN 4
#define DHTTYPE DHT22

DHT dht(DHTPIN, DHTTYPE);

void setup() {
  Serial.begin(115200);
  dht.begin();
}

void loop() {
  delay(2000);

  float humidity = dht.readHumidity();
  float temperature = dht.readTemperature();

  if (isnan(humidity) || isnan(temperature)) {
    Serial.println("Failed to read from DHT sensor!");
```

```
  } else {
    Serial.print("Humidity: ");
    Serial.print(humidity);
    Serial.print("%\t");
    Serial.print("Temperature: ");
    Serial.print(temperature);
    Serial.println("°C");
  }
}
```



Developing the simulation for data transmission.

The monitored data is transmitted to a endpoint on the domain called "beeceptor". The name and the url of the endpoint is given below:

`https://smartenvironment.free.beeceptor.com`

The procedures are described as follows:

- Create an account in beeceptor domain to create a endpoint for creatin mock API.
- Define the mocking rules for HTTP request such as POST and GET rules for storing data and used for website.

POST request body:

{

"user-agent": "ESP32HTTPClient",

"content-length": "38",

"accept-encoding": "identity;q=1,chunked;q=0.1,*;q=0",

"content-type": "application/json",

"x-forwarded-for": "136.144.48.177",

"x-forwarded-host": "smartenvironment.free.beeceptor.com",

"x-forwarded-proto": "https"

}

GET request body:

{

"user-agent": "Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/118.0.0.0 Safari/537.36",

"accept": "*/*",

"accept-encoding": "gzip, deflate, br",

"accept-language": "en-US,en;q=0.9",

"origin": "null",

"sec-ch-ua": "\"Chromium\";v=\"118\", \"Google Chrome\";v=\"118\", \"Not=A?Brand\";v=\"99\"",

"sec-ch-ua-mobile": "?0",

"sec-ch-ua-platform": "\"Windows\"",

"sec-fetch-dest": "empty",

"sec-fetch-mode": "cors",

"sec-fetch-site": "cross-site",

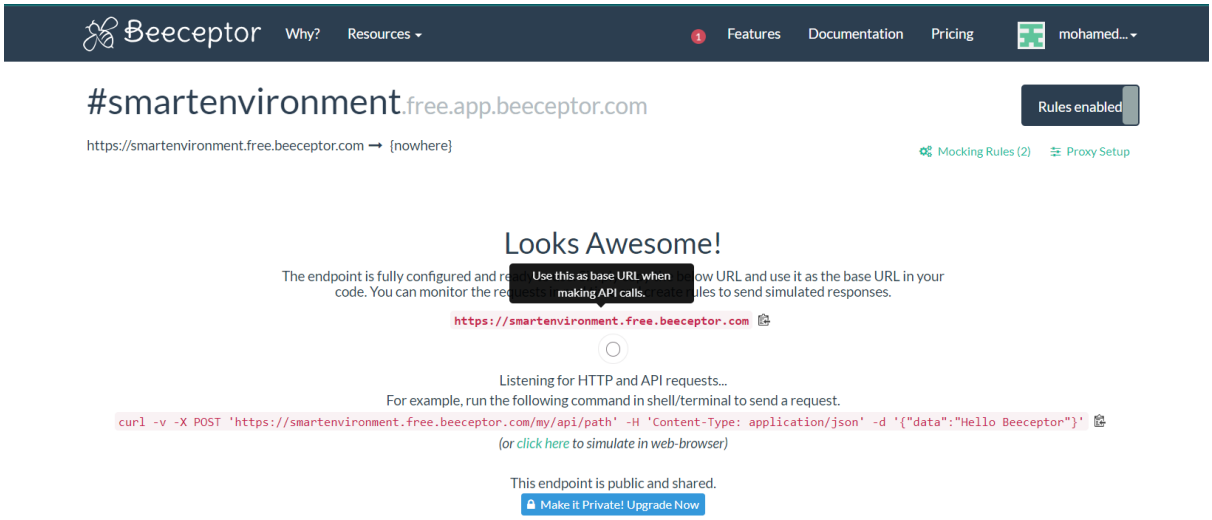
"x-forwarded-for": "157.51.198.81",

"x-forwarded-host": "smartenvironment.free.beeceptor.com",

"x-forwarded-proto": "https"

}

The Image of the platform interface is attached below :



Developed simulation Code:

In order to simulate the code, we must installed the following libraries on wokwi platform.

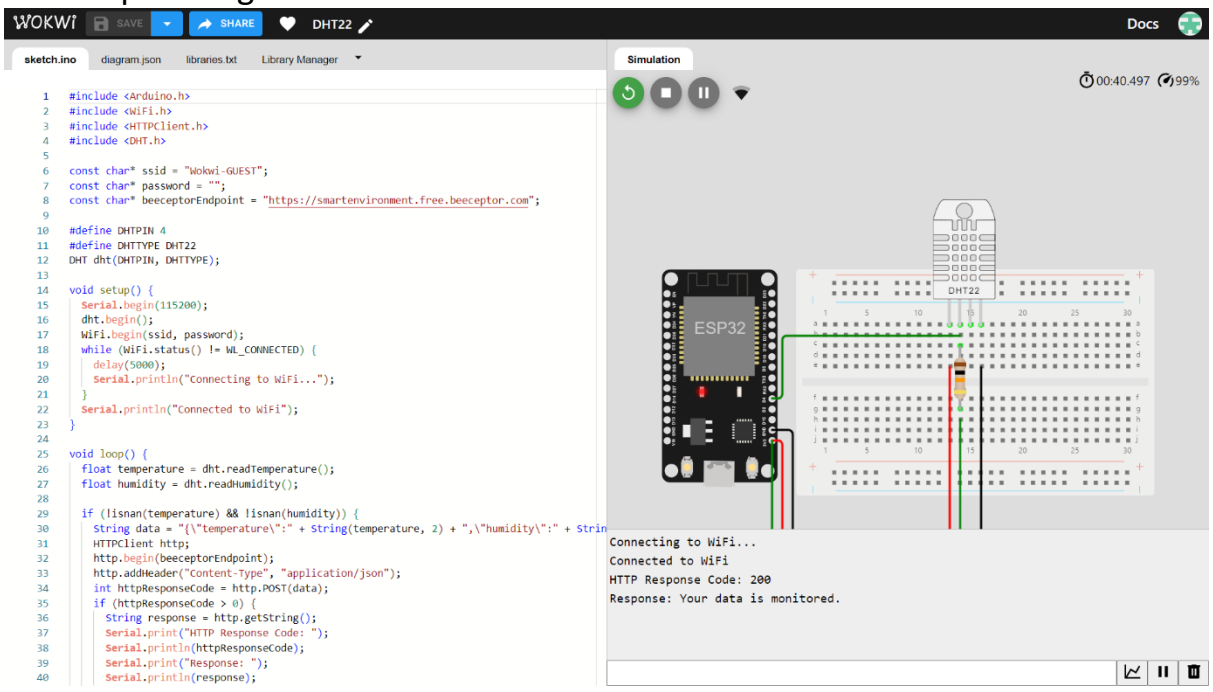
- ✓ Wifi library.
- ✓ DHT22 sensor library.
- ✓ HttpClient library.

The developed code for transmitting data to the beeceptor endpoint using HTTP protocol is given as:

```
#include <Arduino.h>
#include <WiFi.h>
#include <HTTPClient.h>
#include <DHT.h>

const char* ssid = "Wokwi-GUEST";
const char* password = "";
const char* beeceptorEndpoint = "https://smartenvironment.free.beeceptor.com";

#define DHTPIN 4
#define DHTTYPE DHT22
DHT dht(DHTPIN, DHTTYPE);

void setup() {
  Serial.begin(115200);
  dht.begin();
  WiFi.begin(ssid, password);
  while (WiFi.status() != WL_CONNECTED) {
```

```
    delay(5000);
    Serial.println("Connecting to WiFi...");
  }
  Serial.println("Connected to WiFi");
}

void loop() {
  float temperature = dht.readTemperature();
  float humidity = dht.readHumidity();

  if (!isnan(temperature) && !isnan(humidity)) {
    String data = "{\"temperature\":" + String(temperature, 2) +
",\"humidity\":" + String(humidity, 2) + "}";
    HTTPClient http;
    http.begin(beeceptorEndpoint);
    http.addHeader("Content-Type", "application/json");
    int httpResponseCode = http.POST(data);
    if (httpResponseCode > 0) {
      String response = http.getString();
      Serial.print("HTTP Response Code: ");
      Serial.println(httpResponseCode);
      Serial.print("Response: ");
      Serial.println(response);
    }
    http.end();
  }

  delay(60000);  // Send data every 60 seconds (adjust as needed)
}
```

The output image is attached below:

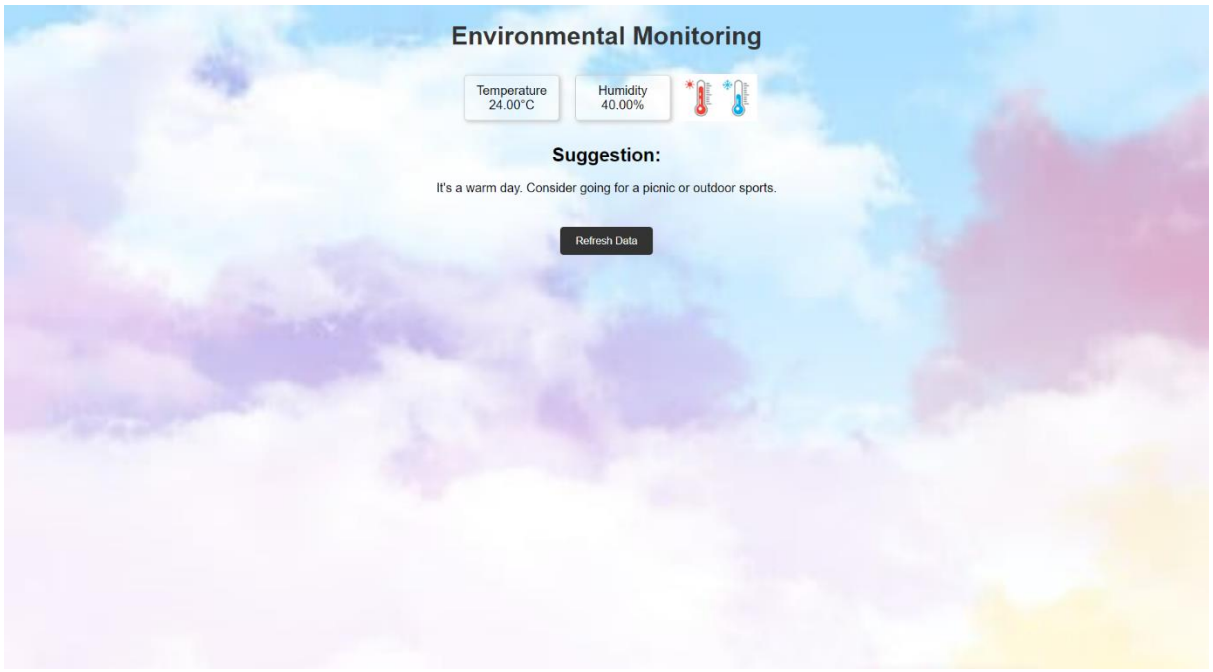


Data Exposure to public:

The data is shared to the public using a website. We chose to use website instead of applications because:

- ✓ Every people owns a 4G mobile but there is a question raised whether their mobile is supported for our application.
- ✓ And also many people did not like to install and use the application rather than some people.
- ✓ So we ended up choosing the website that has been published via social media and QR code that can be available on parks.

The user interface of the website is given below:



The program for creating the website is as follows:

Index.html:

```
<!DOCTYPE html>
<html>
<head>
    <link rel="stylesheet" type="text/css" href="styles.css">
</head>
<body>
    <h1>Environmental Monitoring</h1>
```

```
    <div class="data-container">
        <div class="data-box">
            <p>Temperature</p>
            <span id="temperature">Loading...</span>°C
        </div>
        <div class="data-box">
            <p>Humidity</p>
            <span id="humidity">Loading...</span>%
        </div>
        <div id="image-container">
            <img id="temperature-image" src="" alt="Temperature Image">
        </div>
    </div>
    <div id="suggestion">
        <h2>Suggestion:</h2>
        <p>Loading...</p>
    </div>
    <button id="refresh-button">Refresh Data</button>

    <script src="script.js"></script>
</body>
</html>
```

Styles.css:

```
body {
  font-family: Arial, sans-serif;
  text-align: center;
  background-image: url("images/default-bg.jpg"); /* Default background image
*/
  background-size: cover;
  background-repeat: no-repeat;
  background-attachment: fixed;
}

h1 {
  color: #333;
}

.data-container {
  display: flex;
  justify-content: center;
  align-items: center;
}

.data-box {
  margin: 10px;
```

```
  padding: 10px;
  background-color: rgba(255, 255, 255, 0.1); /* Semi-transparent white
background */
  border: 1px solid #ccc;
  border-radius: 5px;
  box-shadow: 2px 2px 5px #ccc;
  display: inline-block;
  width: 100px; /* Adjust width as needed */
}

.data-box p, .data-box span {
  margin: 0;
}

#temperature-image {
  max-width: 100px;
  display: none;
}

#suggestion {
  margin: 20px;
}

#refresh-button {
  margin: 20px;
  background-color: #333;
  color: #fff;
  border: none;
  padding: 10px 20px;
  border-radius: 5px;
  cursor: pointer;
}

#refresh-button:hover {
  background-color: #555;
}
```

Script.js:

```
document.addEventListener("DOMContentLoaded", function () {
  // Replace with your Beeceptor endpoint URL
  const endpointUrl = "https://smartenvironment.free.beeceptor.com ";

  // Function to fetch data from the endpoint
  function fetchData() {
      fetch(endpointUrl)
          .then((response) => response.json())
```

```
          .then((data) => {
              document.getElementById("temperature").textContent =
data.temperature.toFixed(2);
              document.getElementById("humidity").textContent =
data.humidity.toFixed(2);

              // Customize suggestions based on temperature and humidity
              let suggestion = "Loading...";
              if (data.temperature > 20 && data.humidity < 60) {
                  suggestion = "It's a warm day. Consider going for a picnic
or outdoor sports.";
              } else if (data.temperature < 15 && data.humidity > 70) {
                  suggestion = "It's a cool and humid day. Great for a nature
walk with your loved ones.";
              } else {
                  suggestion = "Enjoy the outdoors responsibly based on the
current weather conditions.";
              }
              document.getElementById("suggestion").lastElementChild.textConte
nt = suggestion;

              // Update the background image based on temperature (dummy
images used here)
              let temperatureImage = "images/default-bg.jpg";
              if (data.temperature > 20) {
                  temperatureImage = "images/warm-bg.jpeg";
              } else if (data.temperature < 15) {
                  temperatureImage = "images/cool-bg.jpeg";
              }
              document.body.style.backgroundImage =
`url(${temperatureImage})`;

              // Update the image based on temperature (dummy images used
here)
              let temperatureImageSrc = "images/default.jpg";
              if (data.temperature > 25) {
                  temperatureImageSrc = "images/hot.jpg";
              } else if (data.temperature < 15) {
                  temperatureImageSrc = "images/cold.jpg";
              }
              document.getElementById("temperature-image").src =
temperatureImageSrc;
              document.getElementById("temperature-image").style.display =
"block";
          })
          .catch((error) => {
              console.error("Error fetching data: " + error);
          });
```

```
  }

  // Initial data fetch
  fetchData();

  // Refresh data on button click
  document.getElementById("refresh-button").addEventListener("click",
fetchData);
});
```

Procedure:

Procedure for creating the website is described as follow:

- ❖ Choose a platform for creating website. We chose Visual studio code.
- ❖ Create a index, styles and script file for the website on the basis of html, css and javascript(js).
- ❖ Upload the necessary images for the backgound images.
- ❖ We used the dynamic transition in our website for various temperature like hot and cold.

Hosting the website:

The created website is hosted by using from various free platform like vercel, google clouds etc., We chose the Netlify platform for hosting our website because it is free of cost and its simplicity is the real deal.

To host the website, follow the instructions as follows:

- Create a account on netlify platform.
- Upload the required files for hosting the website and their required images for background.

Once created and uploaded, the website is hosted successfully. The url of our hosted website is followed by

https://environmentalmonitoringibm.netlify.app/

Publication Idea:

The website can be reached among people by the following innovative ideas:

- We can implement the website using QR code by issuing and exposing it to the public in their corresponding parks.
- By sharing the website URL to the public using social media services like whatsapp group chat, Facebook pages etc,.

Here is the QR code implementation of our website:

Conclusion:

Thus it is concluded that our solution for our problem given is identified in an innovative manner and the solution idea is implemented as a project and it is simulated successfully on the wokwi platform by using the IoT devices and development of the data transmission segment and the data is shared to the public successfully.