AIR QUALITY MONITORING -- RAMESH T

INNOVATION:

Air pollution has become a common phenomenon everywhere. Specially in the urban areas, air pollution is a real-life problem. A lot of people get sick only due to air pollution. In the urban areas, the increased number of petrol and diesel vehicles and the presence of industrial areas at the outskirts of the major cities are the main causes of air pollution. The problem is seriously intensified in the metropolitan cities. Also, the climate change is now apparent. The governments all around the world are taking every measure in their capacity.

Select Appropriate Sensors: Choose sensors for detecting specific pollutants.

Calibration: Adjust sensor responses for accuracy.

Sensor Placement: Strategically position sensors near pollution sources.

Data Collection: Collect real-time air quality data.

Data Transmission: Send data to a central platform wirelessly.

Data Storage: Securely store collected data for analysis.

Data Analysis: Process and interpret sensor data.

Real-time Monitoring: Continuously track pollutant levels.

Pollutant Identification Algorithms: Develop algorithms to identify pollutants.

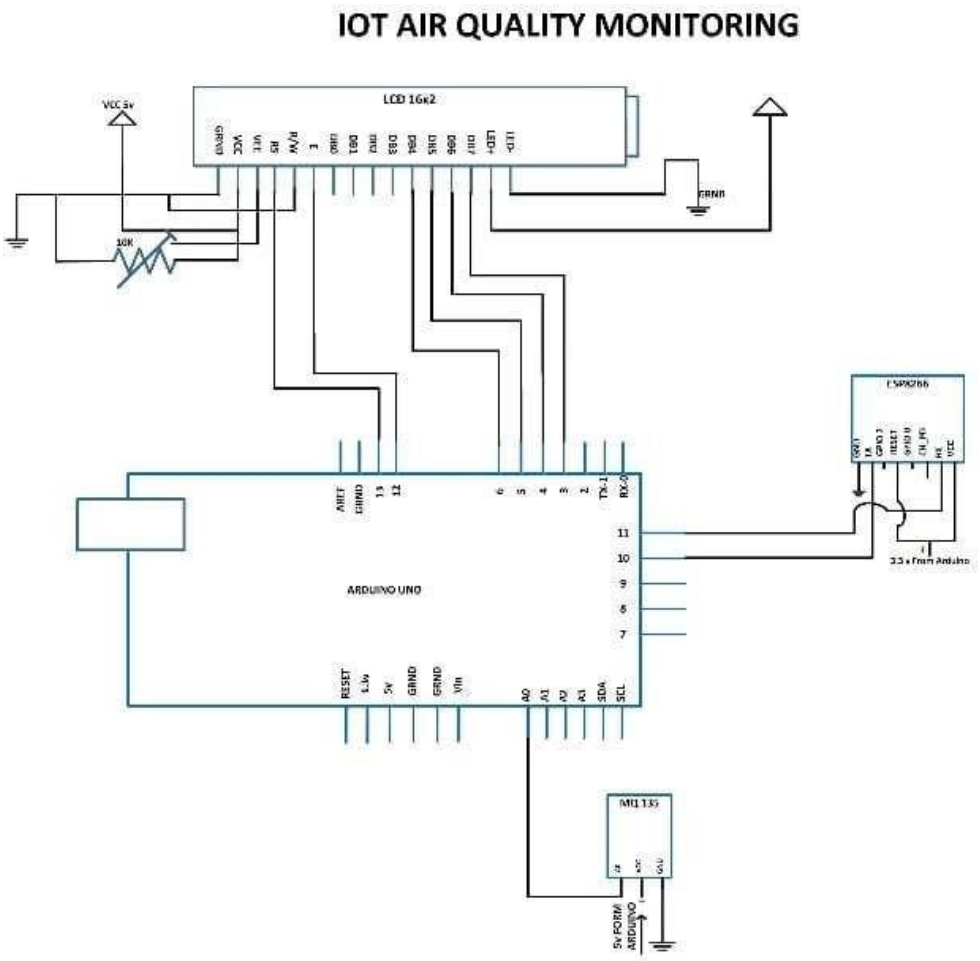
Data Fusion: Combine data from various sources for accuracy.

Alerts and Notifications: Notify authorities and the public of high pollutant levels.

Integration with GIS: Map and visualize pollution data geospatially.

Validation and Quality Assurance: Ensure data accuracy through validation.

Continuous Improvement: Refine algorithms and maintain sensors.

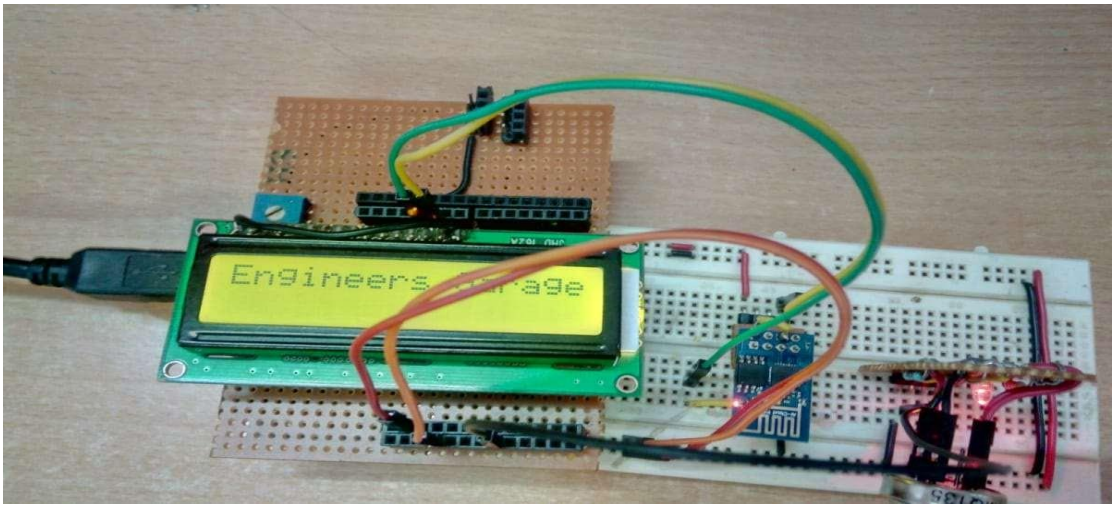


CIRCUIT WORKS AS:

- The device developed in this project can be installed near any Wi-Fi hotspot in a populated urban area.
- As the device is powered, the Arduino board loads the required libraries, flashes some initial messages on the LCD screen and start sensing data from the MQ-135 sensor.
- The sensitivity curve of the sensor for different combustible gases is already mentioned above.

- The sensor can be calibrated so that its analog output voltage is proportional to the concentration of polluting gases in PPM.
- The analog voltage sensed at the pin A0 of the Arduino is converted to a digital value by using the in-built ADC channel of the Arduino.
- The Arduino board has 10-bit ADC channels, so the digitized value ranges from 0 to 1023.
- The digitized value can be assumed proportional to the concentration of gases in PPM.
- The read value is first displayed on LCD screen and passed to the ESP8266 module wrapped in proper string through virtual serial function.
- The Wi-Fi module is configured to connect with the ThingSpeak IOT platform.
- ThingSpeak is an IOT analytics platform service that allows to aggregate, visualize and analyze live data streams in the cloud.
- ThingSpeak provides instant visualizations of data posted by the IOT devices to ThingSpeak server.
- With the ability to execute MATLAB code in ThingSpeak one can perform online analysis and processing of the data as it comes in.

DEVICE MODEL:



CONCLUSION:

In conclusion, the innovation of an IoT-based air quality monitoring device represents a significant step forward in addressing environmental and health concerns. These devices offer real-time data collection and analysis, providing valuable insights into air pollution levels and enabling informed decision-making.

With their ability to connect to the internet and share data seamlessly, they enhance our ability to monitor air quality on a large scale, potentially leading to improved public health outcomes, reduced environmental impact, and more sustainable urban planning. However, it's important to continue refining and expanding these devices to ensure their accuracy, accessibility, and affordability, as well as to foster collaboration between stakeholders to effectively tackle air quality challenges globally. The future of air quality monitoring holds great promise, driven by ongoing innovations in IoT technology and data analytics.

INTRODUCTION

Air quality monitoring using IoT, or the Internet of Things, represents a transformative approach to addressing the pressing global issue of air pollution. With the rapid advancement of technology, IoT has opened up new avenues for efficiently and effectively monitoring the quality of the air we breathe. These IoT-based systems use a network of interconnected sensors and devices to gather, transmit, and analyze real-time data related to air pollutants. This innovative approach has the potential to revolutionize how we understand, manage, and combat air pollution.

In this introduction, we will delve into the significance of air quality monitoring using IoT, the primary objectives it aims to achieve, the role of IoT technology in data collection and dissemination, and the potential benefits it offers to both individuals and the environment. By harnessing the power of IoT, we can gain a deeper understanding of air quality in our communities, identify pollution sources, and take timely actions to improve the air we breathe, ultimately contributing to healthier, more sustainable, and more livable urban environments.

SIMULATION PROCESS

```
from machine import Pin, ADC
from time import sleep
import dht
import network
import urequests
import random

# Initialize Wi-Fi
sta_if = network.WLAN(network.STA_IF)
if not sta_if.isconnected():
    print('connecting to network...')
    sta_if.active(True)
    sta_if.connect('Wokwi-GUEST', '')
```

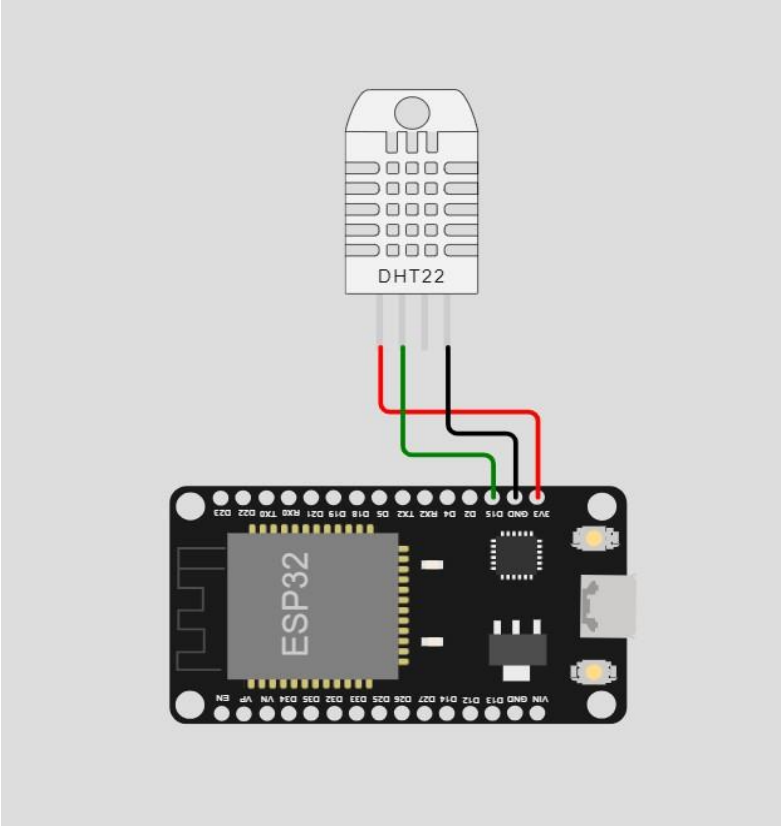
```
    while not sta_if.isconnected():
        pass
    print('network config:', sta_if.ifconfig())

# Initialize DHT22 sensor
sensor = dht.DHT22(Pin(15))

# Initialize ADC pins for gas sensors
# Replace the pin numbers and attenuation settings as needed
# Replace these values with your specific pin and attenuation settings
YOUR_CO_PIN = 34  # Replace with the actual pin number
YOUR_CO_ATTENUATION = ADC.ATTN_11DB  # Replace with the actual attenuation
setting

YOUR_SO2_PIN = 35  # Replace with the actual pin number
YOUR_SO2_ATTENUATION = ADC.ATTN_11DB  # Replace with the actual attenuation
setting

YOUR_NO2_PIN = 36  # Replace with the actual pin number
YOUR_NO2_ATTENUATION = ADC.ATTN_11DB  # Replace with the actual attenuation
setting

firebase_url = 'https://air-quality-monitoring-b1ac7-default-
rtdb.firebaseio.com/'  # Replace with your Firebase URL
firebase_secret = '933awzAPGFzKWjkqSldUBhnFB6IK2zJW6SZZi3g4'

def send_data_to_firebase(temp,hum,CO_level,NO2_level_level,SO2_level):
    data = {
        "Temperature":temp,
        "Humidity":hum,
        "CO": CO_level,
        "NO2": NO2_level,
        "SO2":SO2_level

    }
    url = f'{firebase_url}/Air_data.json?auth={firebase_secret}'

    try:
        response = urequests.patch(url, json=data)  # Use 'patch' instead of
'put'
        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)}")
```

```
while True:
    try:
        sleep(2)
        sensor.measure()
        temp = sensor.temperature()
        hum = sensor.humidity()
        temp_f = temp * (9/5) + 32.0
        print('Temperature: %3.1f C' % temp)
        print('Temperature: %3.1f F' % temp_f)
        print('Humidity: %3.1f %%' % hum)

        # Read gas sensor values (simulated random values)
        CO_level = random.uniform(0, 50)  # Simulated CO level in ppm
        SO2_level = random.uniform(0, 10)  # Simulated SO2 level in ppm
        NO2_level = random.uniform(0, 20)  # Simulated NO2 level in ppm

        print('CO Level: %3.1f ppm' % CO_level)
        print('SO2 Level: %3.1f ppm' % SO2_level)
        print('NO2 Level: %3.1f ppm' % NO2_level)

        # Check if gas levels are in danger as per norms
        if CO_level > 50:
            print('Danger! High CO level detected.')
        if SO2_level > 50:
            print('Danger! High SO2 level detected.')
        if NO2_level > 50:
            print('Danger! High NO2 level detected.')
        else:
            print('you are in good environment')

       # Send data to Firebase
        send_data_to_firebase(temp,hum,CO_level,NO2_level,SO2_level)

       # time.sleep(1)  # Adjust the sleep duration as needed



    except OSError as e:
        print('Failed to read sensor.')
```

STEPS INVOLVED

- Import necessary libraries for hardware control, time, and network communication.

- Check if Wi-Fi is connected; if not, try to connect to the 'Wokwi-GUEST' network.
- Initialize the DHT22 sensor for temperature and humidity measurements.
- Set up placeholders for gas sensor pins and attenuation settings.
- Define Firebase URL and secret token for database access.
- Create a function to send data to Firebase after formatting it.
- Continuously monitor temperature, humidity, and simulated gas sensor data in a loop.
- Print temperature, humidity, and gas sensor values.
- Display a warning if gas levels exceed 50 ppm.
- Call the function to send data to the Firebase database.
- Handle exceptions to capture and print errors during data collection.



WEB DEVELOPMENT

CODE

```
<!DOCTYPE html>
<html>
    <link rel="stylesheet" href="style.css">
<div class="header">
    <img src="bg2.jpg" alt="logo" align="right">
    <h1 style="color:whitesmoke",align="center">AIR QUALITY MONITORING</h1>
</div>

<head>
    <title>AIR-QUALITY-MONITORING</title>
    <style>
        body{
            background-attachment: fixed;
            background-image: url(BG.jpeg);
            background-size: cover;
            background-attachment: fixed;
            opacity: 160;
            background-position: center;

        }
    </style>
    <!-- Include Firebase SDK -->
    <script src="https://www.gstatic.com/firebasejs/10.5.2/firebase-app-
compat.js"></script>
    <script src="https://www.gstatic.com/firebasejs/10.5.2/firebase-database-
compat.js"></script>
</head>


<body>

<div class="header1">
    <h2 style="color:rgb(4, 205, 236)">MAJOR AIR POLLUTANTS</h2>
    <marquee direction="up" behavior="scroll" scrollamount="5"
scrolldelay="100">

    <ol style="color: rgb(159, 131, 131);",class="scroll-list">
        <li style="font-size: 25px";>ground-level ozone particle pollution
(also known as particulate matter, including PM2.5 and PM10)</li>
        <li style="font-size: 25px";>carbon monoxide(CO)</li>
        <li style="font-size: 25px";>Sulfur dioxide(SO2)</li>
        <li style="font-size: 25px";>Nitrogen dioxide (NO2)</li>
        <li style="font-size: 25px";>Volatile Organic Compounds (VOCs)</li>
```

```
         <li style="font-size: 25px";>Lead (Pb)</li>
         <li style="font-size: 25px";>Ammonia (NH3)</li>
         <li style="font-size: 25px";>Hazardous Air Pollutants (HAPs)</li>
         <li style="font-size: 25px";>Particulate Matter (PM)</li>
    </ol>
    </marquee>



</div>
</br>
</br>
<div>
    <table border="1">
        <tr>
            <th>Daily AQI color</th>
            <th>Levels of concern</th>
            <th>Values of Index</th>
            <th>Description of Air Quality</th>
        </tr>
        <tr class="row1">
            <td>Green</td>
            <td>Good</td>
            <td>0 - 50</td>
            <td>Air quality is satisfactory, and air pollution poses
little or no risk</td>
        </tr>
        <tr class="row2">
            <td>Yellow</td>
            <td>Moderate</td>
            <td>51 - 100</td>
            <td>Air quality is acceptable. However, there may be a risk for
some people, particularly those who are unusually sensitive
to air pollution.</td>
        </tr>

        <tr class="row3">
            <td>Orange</td>
            <td>Unhealthty for Sensitive Groups</td>
            <td>101 - 150</td>
            <td>Members of sensitive groups may experience health effects. The
general public is less likely to be affected.</td>
        </tr>
        <tr class="row4">
            <td>Red</td>
            <td>Unhealthy</td>
            <td>151 - 200</td>
```

```
            <td>Some members of the general public may experience health
effects; members of sensitive groups may experience more
serious health effects.</td>
        </tr>
        <tr class="row5">
            <td>Purple</td>
            <td>Very Unhealthy</td>
            <td>201 - 300</td>
            <td>Health alert: The risk of health effects is increased
for everyone.</td>
        </tr>
        <tr class="row6">
            <td>Maroon</td>
            <td>Hazardous</td>
            <td>301 and Higher</td>
            <td>Health warning of emergency conditions: everyone is more
likely to be affected.</td>
        </tr>
    </table>

</div>
</br>
</br>
<div>
    <p style="color:aqua" class="para1">CHECK WHETHER YOU ARE LIVE IN GOOD
ENVIRONMENT OR NOT</p>
</div>
<div class="centered-container">
    <button id="bt1" class="button1"  onclick="getData()">MEASURE</button>
</div>
<div id="dataDisplay"></div>
<div id="safetyStatus"></div>

<script>
   const firebaseConfig = {
  apiKey: "AIzaSyBLnHE4BtKHGcjKKsQtpBdVP0JJ6U28Ylc",
  authDomain: "air-quality-monitoring-9a2b7.firebaseapp.com",
  databaseURL: "https://air-quality-monitoring-9a2b7-default-
rtdb.firebaseio.com",
  projectId: "air-quality-monitoring-9a2b7",
  storageBucket: "air-quality-monitoring-9a2b7.appspot.com",
  messagingSenderId: "473356956937",
  appId: "1:473356956937:web:2ef0a488ec4df8bc190fd8",
  measurementId: "G-XE4550FTRH"
};


    // Initialize Firebase
    firebase.initializeApp(firebaseConfig);
```

```
        // Reference to the Firebase Realtime Database
        var database = firebase.database();

        // Function to retrieve and display data
// Function to retrieve and display data
function getData() {
    console.log("Button clicked");
    var dataDisplay = document.getElementById("dataDisplay");

    // Reference to the data you want to retrieve (e.g., "Air_data" in this
example)
    var dataRef = database.ref("Air_data");

    dataRef.once("value")
        .then(function(snapshot) {
            console.log("Data retrieved successfully");
            var data = snapshot.val();

            // Define safe ranges for your measured values
            var safeRange = {
                Temperature: { min: 0, max: 50 },
                Humidity: { min: 0, max: 50 },
                NO2:{ min: 0, max: 50},
                CO:{min: 0, max:50},
                SO2:{min:0,max:50}

                // Define safe ranges for other parameters
            };

            // Check data against safe ranges
            var isSafe = true;
            for (var key in data) {
                if (data.hasOwnProperty(key)) {
                    for (var param in safeRange) {
                        if (data[key][param] < safeRange[param].min ||
data[key][param] > safeRange[param].max) {
                            isSafe = false;
                            break;
                        }
                    }
                }
            }

            if (isSafe) {
                dataDisplay.innerHTML = "Safe";
                dataDisplay.style.color = "green";
            } else {
```

```
                dataDisplay.innerHTML = "Unsafe";
                dataDisplay.style.color = "red";
            }
            dataDisplay.innerHTML = JSON.stringify(data, null, 2);
            // Display the safety status
           var safetyStatus = document.createElement("div");
           safetyStatus.innerHTML = isSafe ? "Safe" : "Unsafe";
           safetyStatus.className = "safety-status"; // Apply the CSS class
           dataDisplay.appendChild(safetyStatus);

        })
        .catch(function(error) {
            console.error("Error retrieving data: " + error);
            dataDisplay.innerHTML = "An error occurred while retrieving
data.";
        });
}
</script>
</body>
</html>
```

EXPLANATION

HTML Structure:

- The code starts by declaring an HTML document.

Styling:

- An external stylesheet (style.css) is linked for page styling.

Header Section:

- The page header contains a logo and the title "AIR QUALITY MONITORING."

Firebase JavaScript SDK:

- The Firebase JavaScript SDK is included to enable interaction with the Firebase Realtime Database.

"Major Air Pollutants" Section:

- Information about major air pollutants is displayed in a scrolling list.

"Air Quality Index (AQI)" Section:

- A table is provided with different AQI color codes, their associated levels of concern, and descriptions of air quality for each level.

"Check Whether You Are in a Good Environment or Not" Section:

- A call to action encourages users to measure air quality.

"Measure" Button:

- A button with the ID "bt1" is created, and it triggers the getData() JavaScript function when clicked.

JavaScript:

- The firebaseConfig object contains the Firebase configuration with API keys, database URLs, and other details.
- Firebase is initialized using this configuration.
- The getData() function is defined to retrieve data from the Firebase Realtime Database and assess if it falls within safe ranges for various parameters, such as temperature, humidity, NO2, CO, and SO2.
- The safety status (either "Safe" in green or "Unsafe" in red) and the retrieved data are displayed on the page.
- The code uses the Firebase Realtime Database reference to query data under the "Air_data" node.

End of HTML:

- The HTML structure is closed with </html>.

CSS CODE

```
  .header img{
    float:right;
    width:180px;
    height:180px;
    background: #555;

}

.header h1{
    position:relative;
```

```
    top:25px;
    left:25px;
    font-size:70px;
    font-family:'Courier New';


}
.header1 h2{
    position:relative;
    top:30px;

    font-size:40px;
}
table {
    width: 100%;
    border-collapse: collapse;
}

th, td {
    border: 1px solid #000;
    padding: 8px;
    text-align: left;
    font-size: 20px;
}

th {
    background-color: #f2f2f2;
}
.row1
{
    background-color: green;
}
.row2
{
    background-color: yellow;
}
.row3{
    background-color: orange;
}
.row4{
    background-color: red;
}
.row5{
     background-color: purple;
}
.row6{
     background-color: maroon;
}
```

```css
.para1{
    font-size: 30px;
    color: black;
    position: relative;
    left:100px;
}

.centered-container {
    display: flex;
    justify-content: center;
    align-items: center;
    height: 10vh; /* Center vertically on the page */
}
.button1 {
    width: 150px; /* Adjust the width for the size you want */
    height: 75px; /* Half the width for an oval shape */
    background-color: #3498db; /* Blue background color */
    color: #fff;
    border: none;
    border-radius: 50%; /* 50% border-radius creates a circle */
    font-size: 16px; /* Normal font size */
    text-align: center;
    cursor: pointer;
    outline: none;
}

.button1:hover {
    background-color: #2980b9; /* Change color on hover */
}
#dataDisplay {
    font-size: 30px;
    margin-top: 20px;
    text-align: center;
  }
.safety-status {
    font-size: 40px;/* Center the text horizontally and vertically */
  }
```

EXPLANATION

header img:

- Styles the header logo.

- Floats the image to the right.
- Sets a specific width and height.
- Applies a background color.
- Adds margin to the top of the image.

.header h1:

- Styles the header title.
- Adjusts the position, size, and font properties.
- Uses a cursive font family and bold font-weight.

.header1 h2:

- Styles the "Major Air Pollutants" section header.
- Sets the position and color properties.
- Defines font size and color.

table:

- Styles the tables in the document.
- Specifies table width as 100% and collapses borders.

th, td:

- Styles the table headers and data cells.
- Sets border, padding, and text alignment.
- Adjusts font size.

th:

- Styles the table header cells with a background color.

.row1, .row2, .row3, .row4, .row5, .row6:

- Define background colors for specific table rows.

.para1:

- Styles the "CHECK WHETHER YOU ARE IN A GOOD ENVIRONMENT OR NOT" text.
- Sets font size, color, and position properties.

.centered-container:

- Styles the container for centering content.
- Uses flex properties to center content both horizontally and vertically.

.button1:

- Styles the "MEASURE" button.
- Sets width, height, background color, and font properties.
- Defines a circular shape using border-radius.
- Adjusts the cursor and hover effect.

#dataDisplay:

- Styles the data display area.
- Sets font size, margin, and text alignment.

.safety-status:

- Styles the safety status text for "Safe" or "Unsafe."

body:

- Defines background properties for the entire page, including an image as the background with cover scaling.

.scroll-list li:

- Styles the list items inside the scrolling list.

.para1:

- Styles the text within the "para1" class, specifically changing the text color to white.

.center-table:

- Styles a container to center table content vertically (optional)

WEB PAGES

Gmail YouTube Maps

Gmail YouTube Maps

Daily AQI color	Levels of concern	Values of Index	Description of Air Quality
Green	Good	0 - 50	Air quality is satisfactory, and air pollution poses little or no risk
Yellow	Moderate	51 - 100	Air quality is acceptable. However, there may be a risk for some people, particularly those who are unusually sensitive to air pollution.
Orange	Unhealthty for Sensitive Groups	101 - 150	Members of sensitive groups may experience health effects. The general public is less likely to be affected.
Red	Unhealthy	151 - 200	Some members of the general public may experience health effects; members of sensitive groups may experience more serious health effects.
Purple	Very Unhealthy	201 - 300	Health alert: The risk of health effects is increased for everyone.
Maroon	Hazardous	301 and Higher	Health warning of emergency conditions: everyone is more likely to be affected

CHECK WHETHER YOU ARE LIVE IN GOOD ENVIRONMENT OR NOT

MEASURE

URL:

CONCLUSION

In conclusion, Internet of Things (IoT) technology for air quality monitoring is a valuable tool. It provides real-time data, accessibility, and scalability, aiding better decision-making and public health. However, ensuring data accuracy, privacy, and security is essential. Collaboration and public engagement are crucial, and ongoing improvements in technology are needed. IoT-based air quality monitoring has the potential to significantly enhance our understanding and management of air quality concerns, benefiting both the environment and public health.