IoT-Based

Environmental Monitoring

Environmental Monitoring in park

In this project, we see the development of IoT device to monitor the public park. IoT devices collect the data from the environment of the park and show it on the public platform. The park visitor can see the data and access them. The interactive public platform shows the details of the park, it increases the user experience. Data is published in realtime.

Objectives

- Increase visitor Experience
- Provide quality time for Visitors
- Climate and Weather Monitoring
- Invasive Species Detection
- Biodiversity and Wildlife Monitoring
- Natural Disaster Preparedness
- Visitor Safety and Experience
- Educational and Outreach Programs
- Policy and Management Decisions
- Long-Term Sustainability
- Research and Data Analysis
- Monitoring and Compliance
- Data Accessibility and Transparency
- Adaptive Management

Requirements:

- Python script for simulation
- Html script for website
- Wokwi simulator or Other IoT IDEs
- Firebase realtime database

Procedure

- Open the Wokwi simulator and design the diagram of the circuit. Make the given Connections for the circuit.
- Write the python script to get data from sensor and controller to send the database.
- Next create the python project and save the file format as “main.py”.

- **For connect to Firebase Realtime database**
    - Go to the Firebase Console and sign in. and "Add project" in firebase.
    - After creating the project, click on "Add app" and select the platform of your app (iOS, Android, Web). chose "Web app".
    - To Enable the realtime the database, in database tap click "Create Database" button next "Start in test mode" for development.
    - Finally, create the firebase realtime database path as https//firebaselink.com/data/park/

      >Humidity

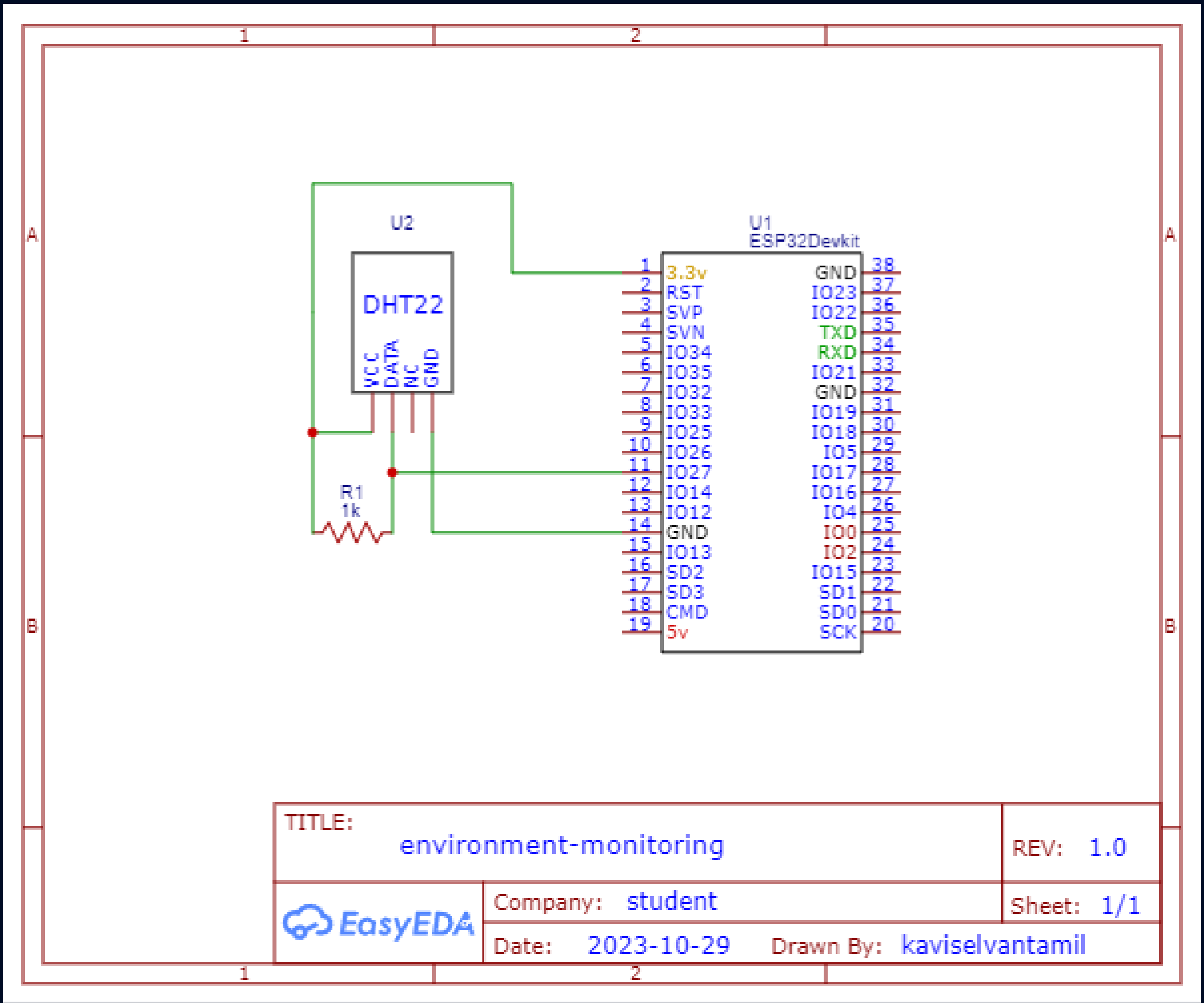
      >Temperature

    - Add the Firebase SDK to your application. For web apps, you'll include the Firebase JavaScript SDK.
    - Copy the database link, database secret key and paste in python script to connect database and park controller

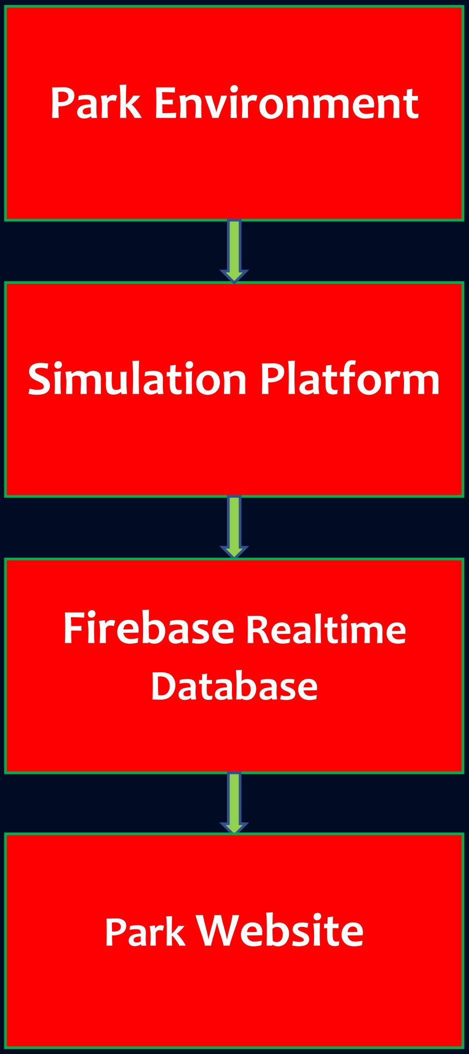
- **HTML Website Connection**
    - Open the HTML file as “my app.html”
    - Copy the SDK script from the firebase database, paste in html script part.
    - Assign the data of temperature and humidity to get from the firebase database.
    - Import the functions and assign the script for the products of firebase SDK.
    - Create the CSS file for the html graphics and save as “my app.css”.
    - Save CSS and html in same folder.

- **Simulation**
    - **Step 1:** Run the wokwi python script. Wait to Wi-Fi connection. Watch the data arrival.
    - **Step 2:** If the data arrived, Open the firebase database here we see the humidity and temperature data.
    - **Step 3:** Open the html file in the browser tap, here we see the realtime data of park environment from sent by firebase database.
    - **Step 4:** Whenever the park environment data changes, it will come to the monitoring website in realtime.

Schematic Diagram



Block Diagram Data transfer


Park Environment

Simulation Platform

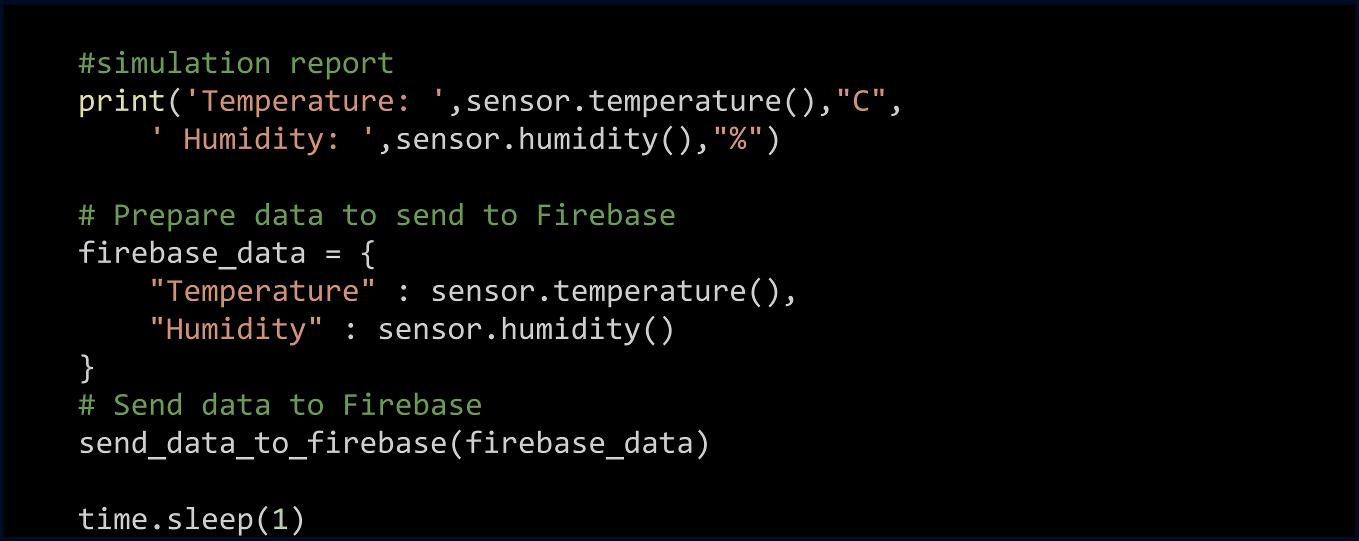
Firebase Realtime Database

Park Website

Wokwi Simulator Python Script

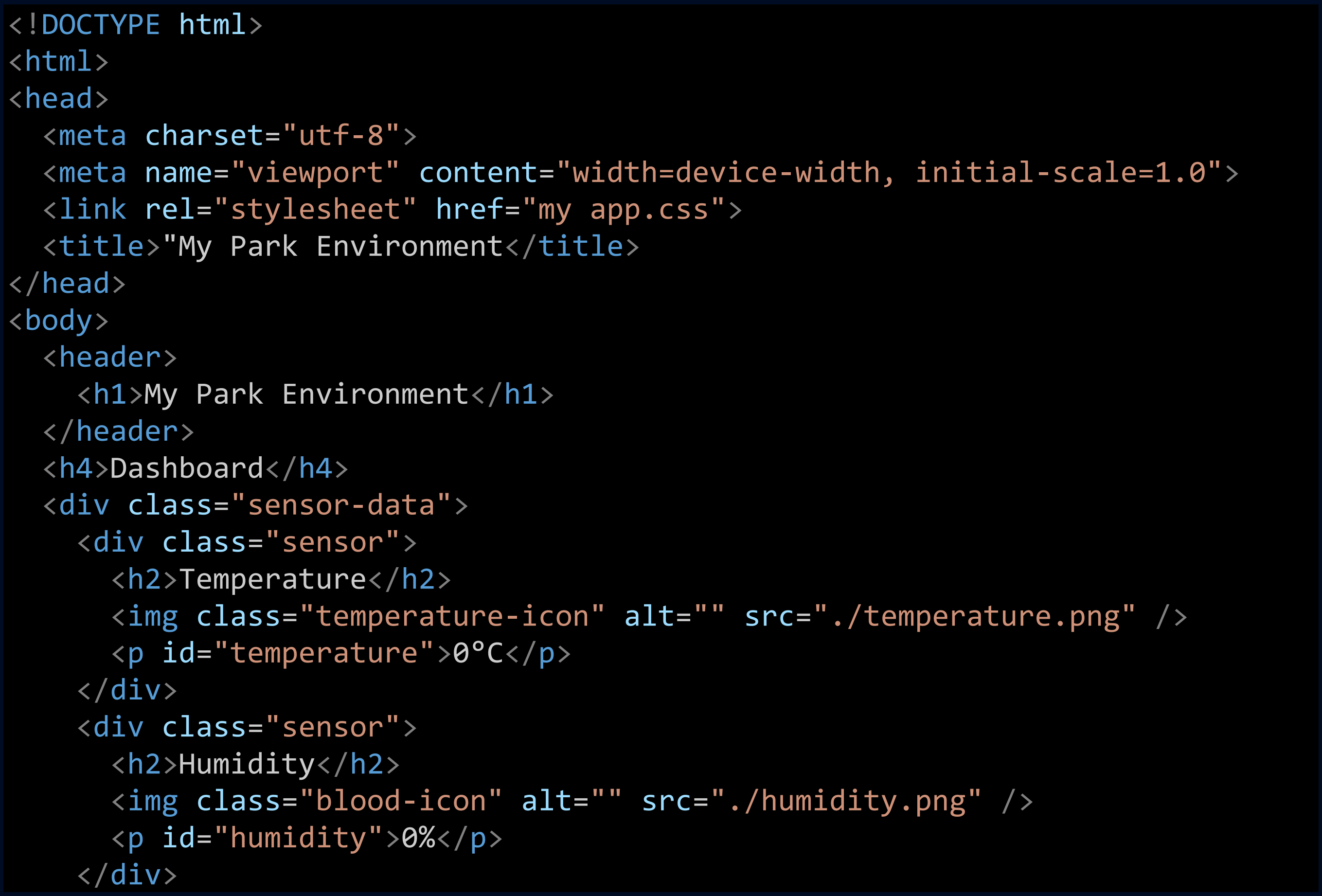
Python script for IoT controller and sensor to send data to firebase realtime database.

```
import network
import time
from machine import Pin
import dht
import ujson
import urequests

#sensor setup in ESP32
sensor = dht.DHT22(Pin(27))

# Firebase configuration
FIREBASE_URL = "https://environment-monitoring-f638b-default-
rtdb.firebaseio.com"
FIREBASE_SECRET = "IouMIfvINfGy5JGOpM4OIuGfjomMuQ0C7slx36KC"

#WiFi connection for ESP32
print("Connecting to WiFi", end="")
sta_if = network.WLAN(network.STA_IF)
sta_if.active(True)
sta_if.connect('Wokwi-GUEST', '')
while not sta_if.isconnected():
  print(".", end="")
  time.sleep(0.1)
print(" Connected!")

#Send data to firebase
def send_data_to_firebase(data):
    print("Sending data to firebase...")
    url = "{}/data/park.json?auth={}".format(FIREBASE_URL, FIREBASE_SECRET)
    headers = {"Content-Type": "application/json"}
    response = urequests.put(url, json=data, headers=headers)
    print("Firebase Response:", response.text)
    response.close()

#Measuring Environment Conditions
print("Measuring Environment Conditions... ")
while True:
    sensor.measure()
```

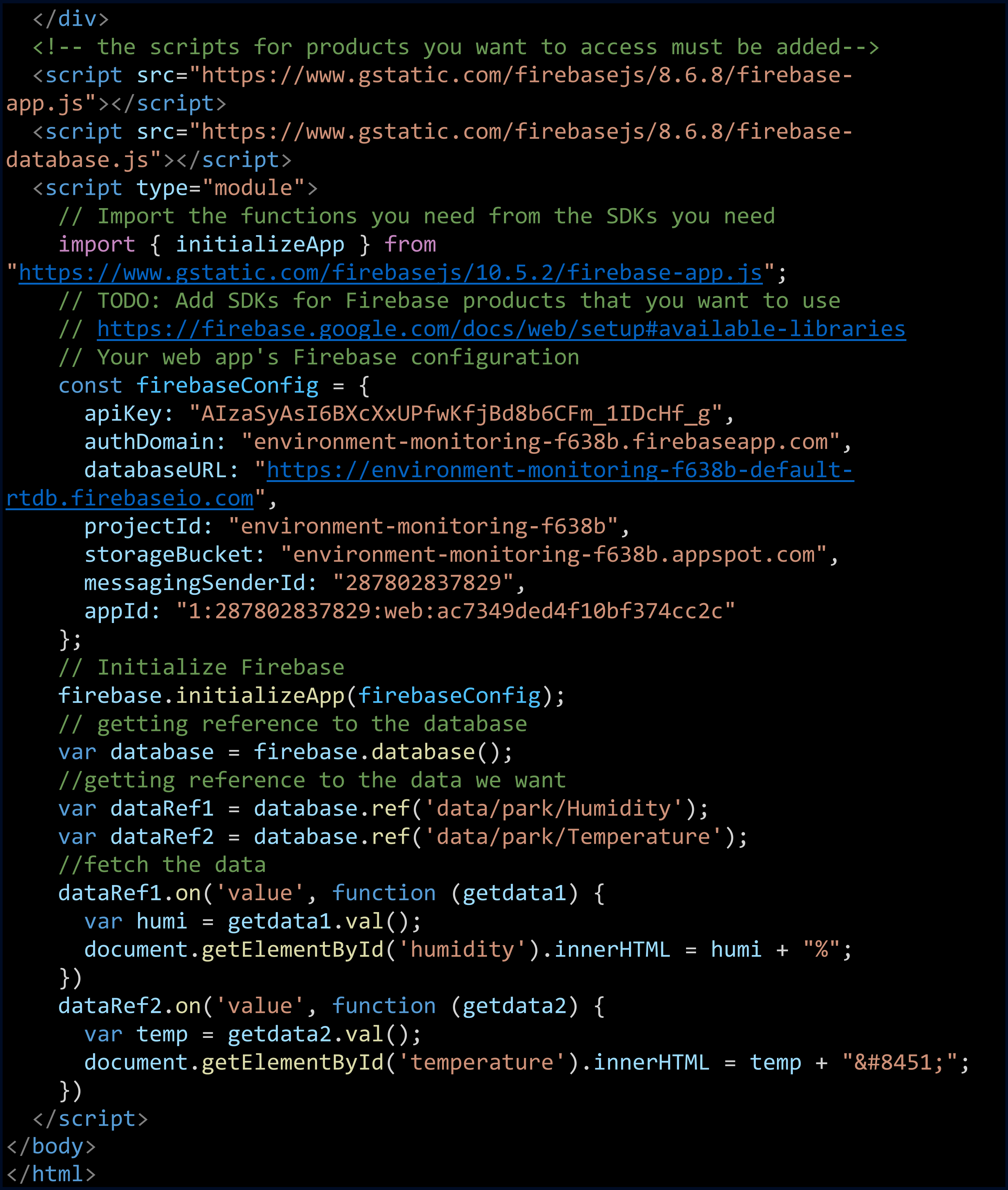


```
    #simulation report
    print('Temperature: ',sensor.temperature(),"C",
        ' Humidity: ',sensor.humidity(),"%")

    # Prepare data to send to Firebase
    firebase_data = {
        "Temperature" : sensor.temperature(),
        "Humidity" : sensor.humidity()
    }
    # Send data to Firebase
    send_data_to_firebase(firebase_data)

    time.sleep(1)
```

HTML Website Script

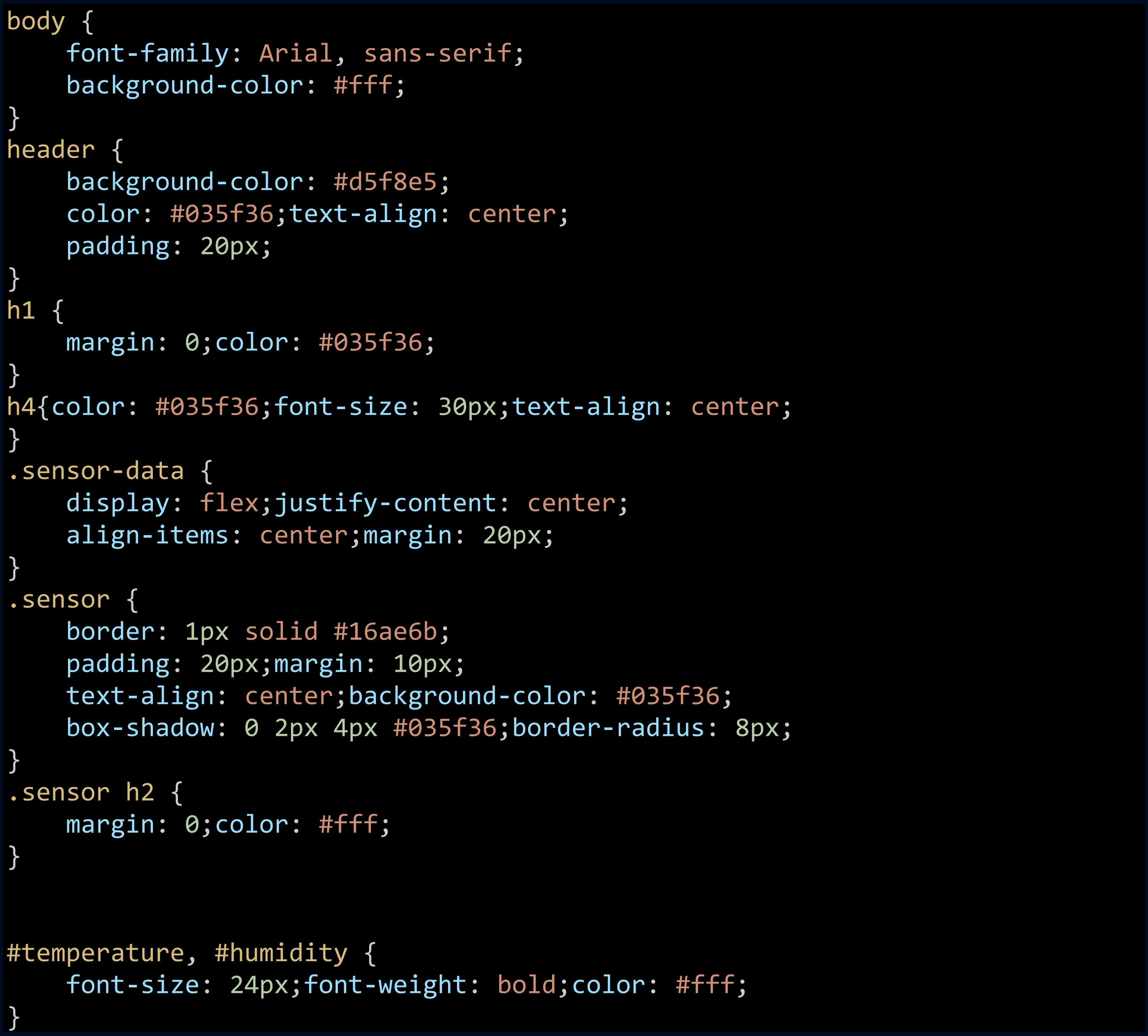


```
<!DOCTYPE html>
<html>
<head>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <link rel="stylesheet" href="my app.css">
  <title>"My Park Environment</title>
</head>
<body>
  <header>
    <h1>My Park Environment</h1>
  </header>
  <h4>Dashboard</h4>
  <div class="sensor-data">
    <div class="sensor">
      <h2>Temperature</h2>
      <img class="temperature-icon" alt="" src="./temperature.png" />
      <p id="temperature">0°C</p>
    </div>
    <div class="sensor">
      <h2>Humidity</h2>
      <img class="blood-icon" alt="" src="./humidity.png" />
      <p id="humidity">0%</p>
    </div>
```



```
  </div>
  <!-- the scripts for products you want to access must be added-->
  <script src="https://www.gstatic.com/firebasejs/8.6.8/firebase-app.js"></script>
  <script src="https://www.gstatic.com/firebasejs/8.6.8/firebase-database.js"></script>
  <script type="module">
    // Import the functions you need from the SDKs you need
    import { initializeApp } from "https://www.gstatic.com/firebasejs/10.5.2/firebase-app.js";
    // TODO: Add SDKs for Firebase products that you want to use
    // https://firebase.google.com/docs/web/setup#available-libraries
    // Your web app's Firebase configuration
    const firebaseConfig = {
      apiKey: "AIzaSyAsI6BXcXxUPfwKfjBd8b6CFm_1IDcHf_g",
      authDomain: "environment-monitoring-f638b.firebaseapp.com",
      databaseURL: "https://environment-monitoring-f638b-default-rtdb.firebaseio.com",
      projectId: "environment-monitoring-f638b",
      storageBucket: "environment-monitoring-f638b.appspot.com",
      messagingSenderId: "287802837829",
      appId: "1:287802837829:web:ac7349ded4f10bf374cc2c"
    };
    // Initialize Firebase
    firebase.initializeApp(firebaseConfig);
    // getting reference to the database
    var database = firebase.database();
    //getting reference to the data we want
    var dataRef1 = database.ref('data/park/Humidity');
    var dataRef2 = database.ref('data/park/Temperature');
    //fetch the data
    dataRef1.on('value', function (getdata1) {
      var humi = getdata1.val();
      document.getElementById('humidity').innerHTML = humi + "%";
    })
    dataRef2.on('value', function (getdata2) {
      var temp = getdata2.val();
      document.getElementById('temperature').innerHTML = temp + "&#8451;";
    })
  </script>
</body>
</html>
```

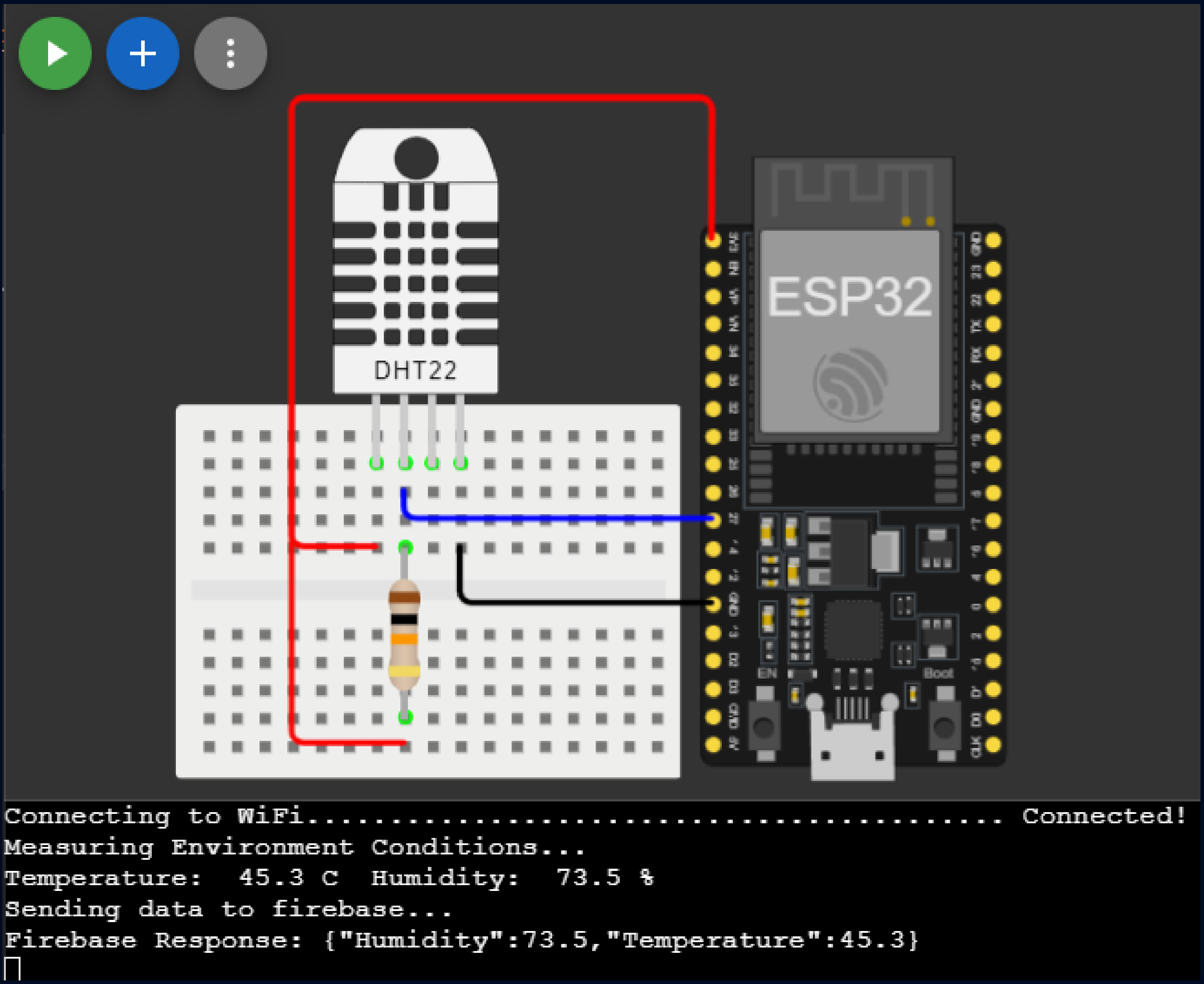
CSS Script For Website



```css
body {
    font-family: Arial, sans-serif;
    background-color: #fff;
}
header {
    background-color: #d5f8e5;
    color: #035f36;text-align: center;
    padding: 20px;
}
h1 {
    margin: 0;color: #035f36;
}
h4{color: #035f36;font-size: 30px;text-align: center;
}
.sensor-data {
    display: flex;justify-content: center;
    align-items: center;margin: 20px;
}
.sensor {
    border: 1px solid #16ae6b;
    padding: 20px;margin: 10px;
    text-align: center;background-color: #035f36;
    box-shadow: 0 2px 4px #035f36;border-radius: 8px;
}
.sensor h2 {
    margin: 0;color: #fff;
}


#temperature, #humidity {
    font-size: 24px;font-weight: bold;color: #fff;
}
```

Wokwi Simulation

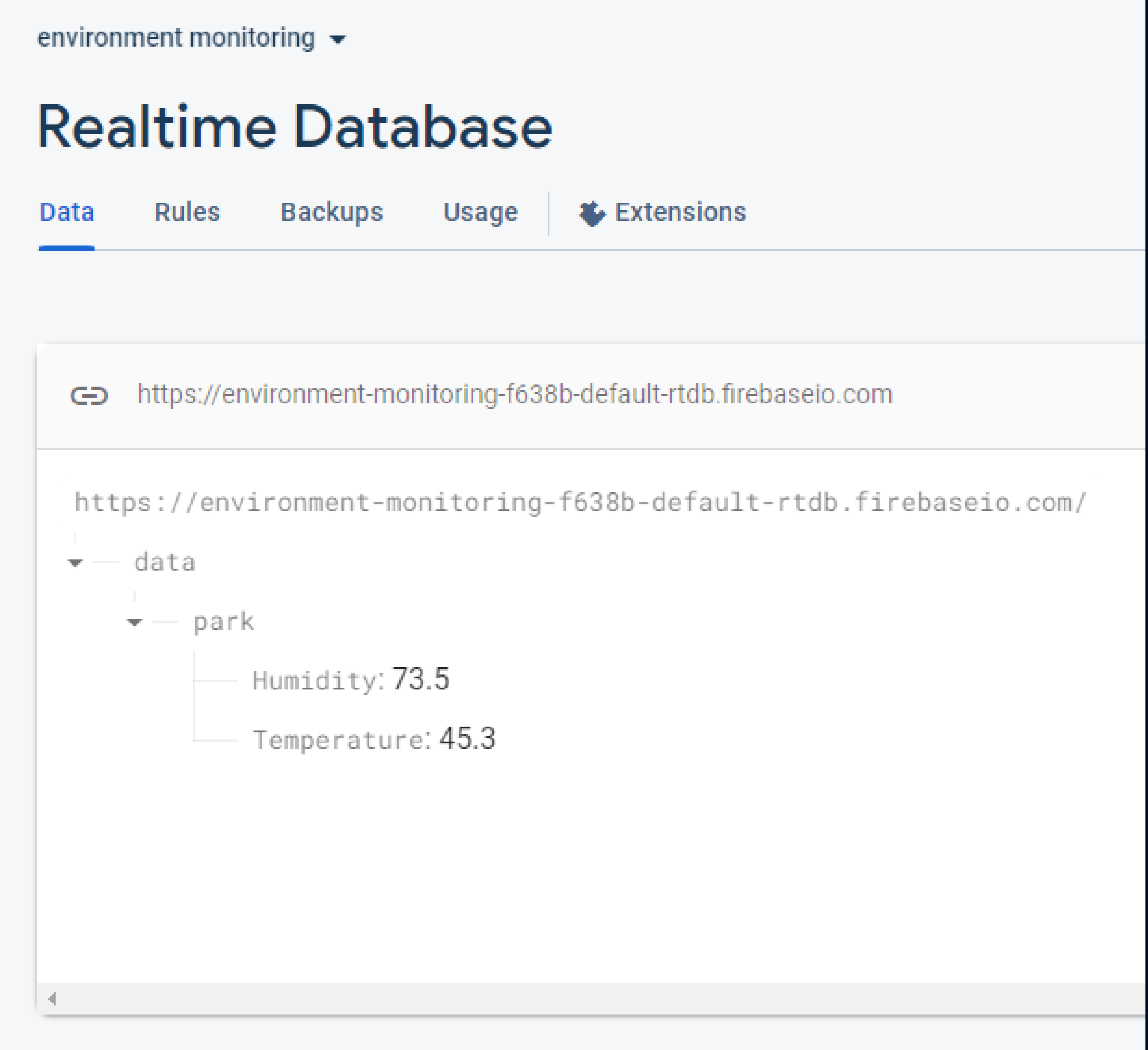
For Temperature and Humidity with ESP32 and DHT22.



```
Connecting to WiFi........................................... Connected!
Measuring Environment Conditions...
Temperature:  45.3 C  Humidity:  73.5 %
Sending data to firebase...
Firebase Response: {"Humidity":73.5,"Temperature":45.3}
```

Firebase Realtime Database

The Temperature and Humidity data received in firebase.

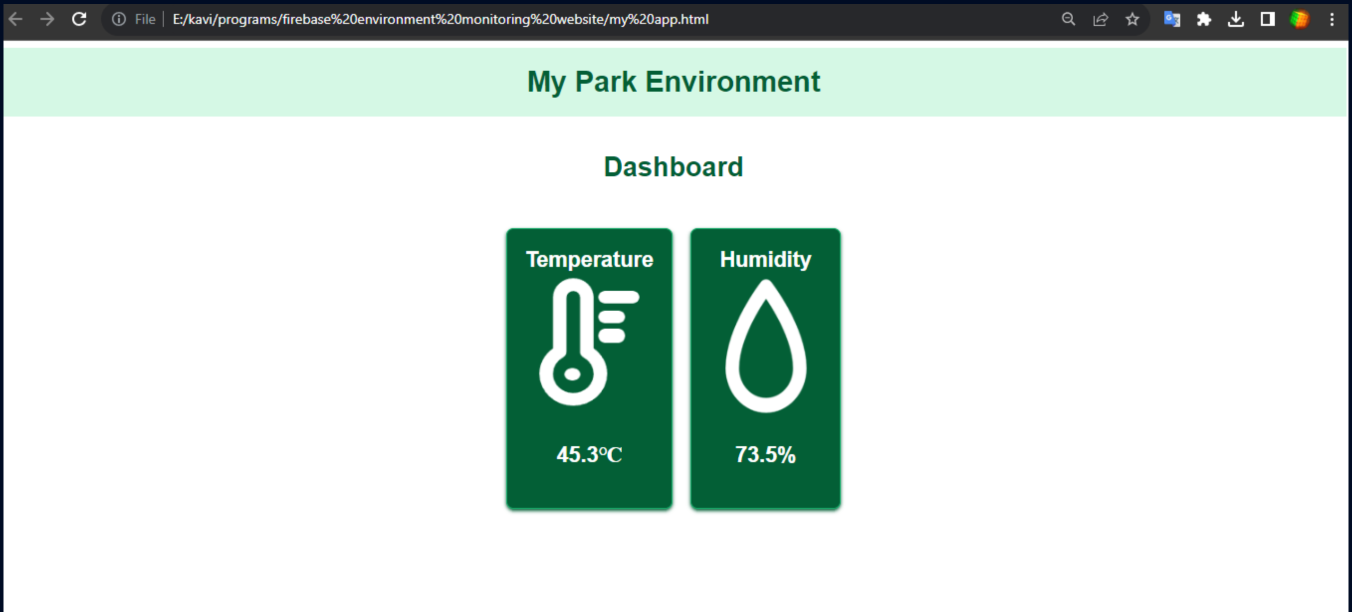


https://environment-monitoring-f638b-default-rtdb.firebaseio.com

```
https://environment-monitoring-f638b-default-rtdb.firebaseio.com/
▾ — data
    ▾ — park
          — Humidity: 73.5
          — Temperature: 45.3
```

Monitoring Website

The Temperature and Humidity data in the html websiten dashboard.



File | E:/kavi/programs/firebase%20environment%20monitoring%20website/my%20app.html

My Park Environment

Dashboard

Temperature

45.3°C

Humidity

73.5%

Real-time Monitoring benefits

Realtime environmental monitoring system benefits park visitors.

- **Weather Awareness:** Park visitors can access real-time temperature and humidity data to plan their activities effectively. This information can help them dress appropriately and choose activities that are suitable for the current weather conditions, preventing discomfort or health risks associated with extreme temperatures or humidity levels.
- **Health and Safety:** High temperatures and humidity can be dangerous, especially in hot and humid climates. Real-time data can help visitors avoid heat-related illnesses and provide warnings when conditions are unsafe for certain activities like hiking or physical exertion.
- **Historical Data:** Over time, the system can collect and store historical data, allowing visitors to access past weather information. This can help them plan future trips and gain a deeper understanding of the park's climate patterns.
- **Better Planning:** Park visitors can use the data to make informed decisions about their trips, such as when to visit and which trails or areas are more comfortable for their activities. This can help prevent overcrowding at certain times, thus improving the overall experience.

- **Alerts and Notifications:** In the event of sudden weather changes or extreme conditions, the IoT system can send alerts to visitors' mobile devices, helping them stay safe and make timely decisions to leave the park if necessary.
- **Energy Efficiency:** For eco-conscious visitors, the system can display information on the park's energy consumption and renewable energy usage, promoting sustainability and environmental awareness.
- **Data for Research and Conservation:** The collected data used for research and conservation efforts within the park. It can help conservationists in monitoring changes in the environment, tracking the impact of climate change, and implementing measures to protect the park's ecosystem.
- **Interactive Engagement:** Park visitors can engage with the system through their smartphones or other devices Historical Data: Over time, the system can collect and store historical data, allowing visitors to access past weather information. This can help them plan future trips and gain a deeper understanding of the park's climate patterns.
- **Environmental Education:** The system can be used to educate visitors about the importance of conservation, wildlife protection, and the park's natural resources through real-time data and educational content.