ENVIRONMENTAL MONITORING IN PARKS

PHASE 5

TEAM MEMBERS:

UDHAYAKUMAR N (810021106090)
NISHANTHAN T (810021106053)
SURYAKUMAR R (810021106308)
SANTHOSH T (810021106067)
TARUN MR (810021106086)
VIMAL AS (810021106097)

Project Definition:

- The Environmental Monitoring in Parks project aims to implement a comprehensive system for monitoring and managing environmental conditions within parks and natural reserves.
- This project will leverage IoT (Internet of Things) technology to collect and analyze data from various sensors deployed throughout the park.
- The primary goal is to enhanced conservation efforts, ensure the well-being of park ecosystems, and improve the overall visitor experience.

Project Objectives:

- Data Collection and Monitoring: Implement a network of IoT devices and sensors to collect realtime data on weather, air quality, water quality, biodiversity, soil, vegetation, and energy consumption within the park.

- Environmental Conservation: Use collected data to monitor the health of park ecosystems, identify environmental threats, and take proactive measures to protect wildlife, aquatic systems, and vegetation.

- Visitor Safety: Ensure the safety and well-being of park visitors by monitoring air quality, weather conditions, and water quality to provide timely alerts and information.

- Sustainability: Monitor energy consumption and waste management to promote sustainable practices and reduce the park's environmental footprint.

- Data Integration and Analysis: Develop a centralized platform for data integration and analysis, incorporating advanced analytics and machine learning to detect trends, anomalies, and potential environmental issues.

- Reporting and Alerts: Create an automated reporting and alert system for park administrators, stakeholders, and the public, fostering transparency and enabling rapid responses to emerging environmental challenges.

Vision:

Our vision for environmental monitoring in parks is to create a harmonious and sustainable coexistence between nature, recreation, and conservation. We aim to transform parks into living laboratories where cutting-edge technology and community engagement converge to safeguard the environment and enhance the visitor experience.

Problem Statement:

Parks are valuable public spaces for relaxation and outdoor activities, attracting a diverse range of visitors throughout the year. However, fluctuating environmental conditions, particularly temperature and humidity, can significantly impact visitors' comfort and experience. The challenge is to create an IoT-based environmental monitoring system that ensures optimal temperature and humidity levels in park areas, enhancing visitor satisfaction and well-being.

Problem Description:

Extreme Weather Conditions: Parks often experience a wide range of temperatures and humidity levels due to seasonal variations. Extreme heat or cold can make park visits uncomfortable or even unsafe for visitors.

Visitor Experience: Unfavourable environmental conditions can deter visitors from enjoying the park's natural beauty and amenities, affecting their overall experience.

Health Concerns: Extreme temperatures can pose health risks, such as heat exhaustion or hypothermia, especially for vulnerable populations like children and the elderly.

Resource Management: Inefficient use of resources, such as heating and cooling systems in park facilities, can lead to unnecessary energy consumption and environmental impact.

Solution for an IoT-Based Temperature and Humidity Environmental Monitoring Project in Parks:

1. Sensor Deployment: Install a network of temperature and humidity sensors strategically across various park zones, ensuring comprehensive coverage of visitor areas, facilities, and natural spaces.

2. Data Collection and Analysis: Collect data from the deployed sensors in real-time and transmit it to a centralized database or cloud platform. Utilize advanced data analytics techniques, including machine learning algorithms, to process and analyze the data efficiently.

3. Automated Control System: Implement an automated control system that can adjust environmental conditions in park facilities based on the real-time data. For instance, when the temperature rises significantly in a park pavilion, the system can activate cooling mechanisms to maintain a comfortable environment for visitors.

4. Alerts and Notifications: Develop an alert system that can notify park management and staff when extreme conditions are detected, enabling swift response to visitor safety concerns.

5. Visitor Information Access: Create a user-friendly mobile application or integrate data displays in the park's visitor centers to provide real-time temperature and humid

ity information. Include weather forecasts and tips for visitors to prepare accordingly.

6. Energy Efficiency: Ensure the control system is designed for energy efficiency, optimizing heating, cooling, and ventilation systems to minimize resource consumption.

7. Data Visualization: Visualize temperature and humidity data on interactive maps or graphs, allowing park management to identify trends and areas that frequently require adjustments.

8. Visitor Education: Incorporate educational content in the mobile app, explaining the importance of environmental comfort and how responsible resource use benefits both visitors and the environment.

9. Regular Maintenance: Establish a routine maintenance schedule for sensors and control systems to ensure accuracy and reliability. Implement fail-safe mechanisms to address sensor malfunctions promptly.

10. Feedback Loop: Encourage park visitors to provide feedback through the app or on-site kiosks regarding their comfort and experience. This feedback can help in continuous improvement.

11. Data Security and Privacy: Prioritize data security and privacy to protect sensitive information collected from park visitors.

12. Accessibility: Ensure that temperature and humidity information is accessible to all visitors, including those with disabilities, through inclusive design practices.

By implementing these solutions, the IoT-based environmental monitoring system will not only enhance visitor comfort and safety in parks but also contribute to responsible resource management, making parks more enjoyable and sustainable for all.

Design Framework:

In this project, we are going to send Temperature and Humidity sensor data to Thingspeak using DHT11. By this method, we can monitor our DHT11 sensor's temperature and humidity data over the internet using the ThingSpeak IoT server, and we can view the logged data and graph over time on the ThingSpeak dashboard. NodeMCU reads the current temperature and humidity from DHT11 and sends it to the ThingSpeak server for live monitoring from anywhere in the world.

Components Required:

- NodeMCU
- DHT11 Temperature and Humidity Sensor
- Jumper Wires
- Breadboard

Features of the components:

NodeMCU:

- Based on ESP8266 Wi-Fi module.
- Built-in Wi-Fi connectivity. Supports Lua scripting and Arduino IDE. GPIO pins for digital I/O and PWM.
- Analog input pins.
- UART, SPI, and I2C communication support.
- USB-to-Serial interface for programming.

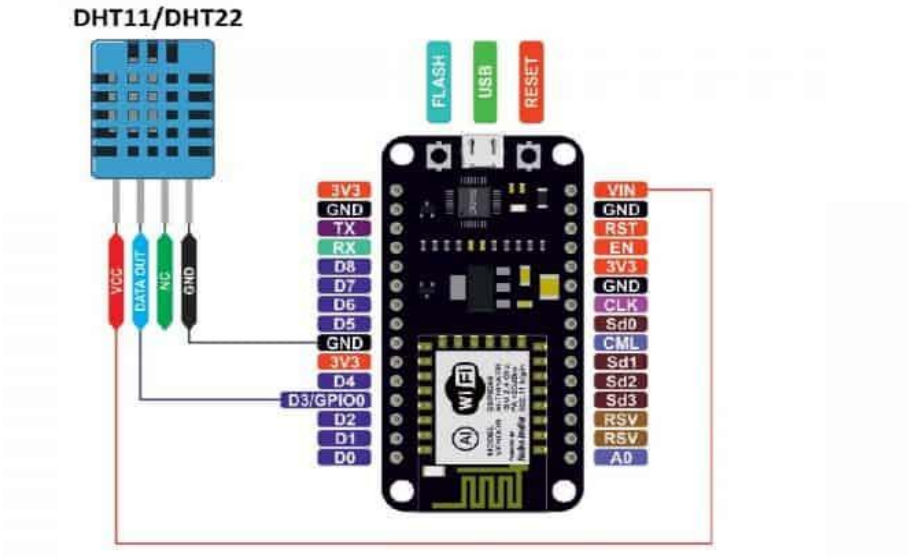
- Low power consumption for battery-powered devices.
- Over-the-Air (OTA) firmware updates.
- Active and supportive community.
- Affordable for hobbyists and developers.

DHT11 Temperature and Humidity Sensor:

- Combined temperature and humidity sensing.
- Digital output for easy interfacing with microcontrollers.
- Low cost and widely available.
- Reliable for basic temperature and humidity monitoring.
- Limited accuracy and range compared to more advanced sensors.
- Single-wire communication protocol.

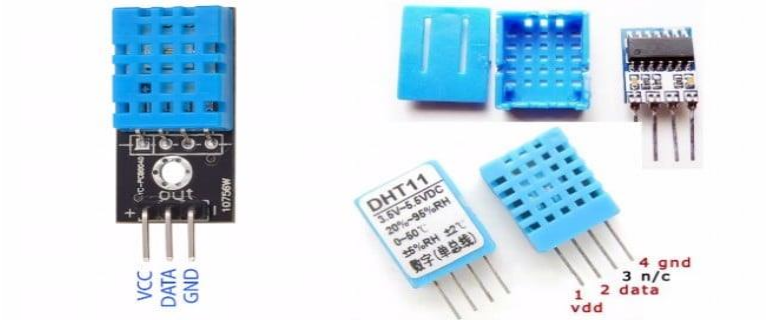
IOT Device Developments:

Circuit Diagram:



DHT11 Humidity & Temperature Sensor:

The DHT11 is a basic, ultra low-cost digital temperature and humidity sensor. It uses a capacitive humidity sensor and a thermistor to measure the surrounding air and spits out a digital signal on the data pin (no analog input pins needed).



It's fairly simple to use but requires careful timing to grab data. The only real downside of this sensor is you can only get new data from it once every 2 seconds, so when using the library, sensor readings can be up to 2 seconds old.

Benefits:

1. Data Collection and Analysis: IoT sensors can monitor various environmental parameters like air quality, temperature, humidity, water quality, and more. This data helps park authorities to understand the ecosystem's health and make informed decisions.

2. Wildlife Conservation: IoT can track the movement of animals within the park, helping researchers gather data on their behavior and migration patterns. This information is vital for wildlife conservation efforts.

3. Weather Monitoring: Real-time weather data collected through IoT sensors can assist park management in predicting and preparing for extreme weather events, ensuring visitor safety.

4. Fire Detection: IoT sensors can detect sudden increases in temperature or the presence of smoke, enabling early wildfire detection and rapid response.

5. Water Management: IoT devices can monitor water sources within the park, helping to manage water levels, quality, and ensure the sustainability of aquatic ecosystems.

6. Visitor Experience: IoT can enhance the visitor experience by providing real-time information on trail conditions, parking availability, and even guided tours using mobile apps.

7. Energy Efficiency: Smart lighting and heating/cooling systems can reduce energy consumption in park facilities, contributing to sustainability goals.

8. Noise Pollution Monitoring: Sensors can measure noise levels and help manage noise pollution in the park, preserving the natural soundscape.

9. Ecosystem Research: IoT data can support scientific research within the park, aiding in ecological studies and climate change research.

10. Remote Monitoring: Park managers can the need for physical inspections and saving resources.

Python Code for the NodeMCU:

```
import
Adafruit_DHT
import requests


apiKey = "Your API
KEY" ssid = "WiFi
Name" password =
"WiFi Password"
server =
"api.thingspeak.com"

DHTPIN = 4


dht = Adafruit_DHT.DHT11
```

```
def setup():
print("Connecting to
")     print(ssid)

   WiFi.begin(ssid, password)     while
WiFi.status() != WL_CONNECTED:
delay(550)        print(".")     print("")
print("WiFi connected")


def loop():
   h, t = Adafruit_DHT.read_retry(dht,
DHTPIN)     if math.isnan(h) or math.isnan(t):

      print("Failed to read from DHT
sensor!")        return     if
client.connect(server, 80):

      postStr = apiKey        postStr += "&field1="        postStr += str(t)
postStr += "&field2="        postStr += str(h)        postStr += "\r\n\r\n"
client.print("POST /update HTTP/1.1\r\n")        client.print("Host:
api.thingspeak.com\r\n")        client.print("Connection: close\r\n")
client.print("X-THINGSPEAKAPIKEY: " + apiKey + "\r\n")
client.print("Content-Type: application/x-www-form-urlencoded\r\n")
client.print("Content-Length: ")


client.print(postStr.length
())
client.print("\r\n\r\n")
client.print(postStr)
client.stop()
print("Data sent!")
```

USING WEB DEVELOPMENT CREATE A PLATFORM THAT DISPLAYS ENVIRONMENTAL MONITORING IN PARKS DATA:

HTML CODE:

```
<!DOCTYPE html>
<html>
<head>
   <link rel="stylesheet" type="text/css" href="styles.css">
</head>
<body>
   <div class="container">
      <h1>Environmental Monitoring in Parks</h1>
         <div class="data">
            <h2>Temperature</h2>
            <p id="temperature">28°C(82.4°F)</p>
         </div>
         <div class="data">
            <h2>Humidity</h2>
            <p id="humidity">30%</p>
         </div>
   </div>
   <script src="script.js"></script>
</body>
</html>
```

CSS CODE:

```
 body {     font-family: Arial, sans-
serif;     backgroundcolor:
#f0f0f0;
}
```

```
.container {     text-align: center;     background-
color: #fff;     border-radius: 10px;     padding:
5px;     margin: 10px;     box-shadow: 0 0 10px
rgba(0, 0, 0, 0.1);     height:500px;
width:350px;
}

h1 {     color:
#333;
}


.data {
padding:20px;
margin: 30px;     border:
1px solid #ddd;
borderradius: 10px;
background-color: rgb(204,255,255);
box-shadow: 0 0 5px
rgba(0, 0, 0, 0.1);
height:100px;     width:250px;
}

p {     color:
#555;

}

h2 {     font-size:
24px;     fontweight:
bold;
```

}

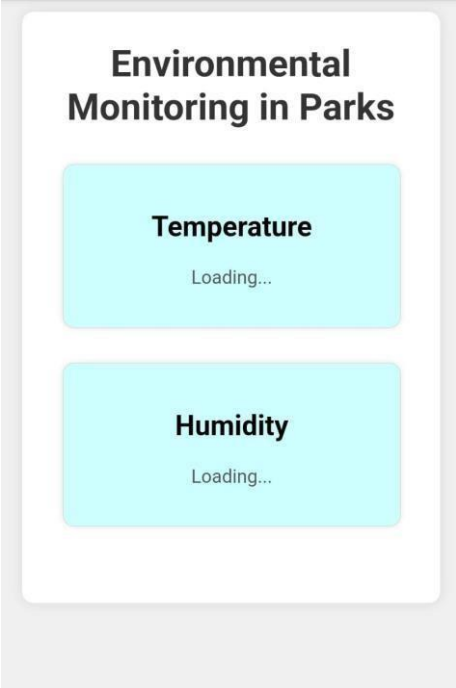
JAVA SCRIPT:

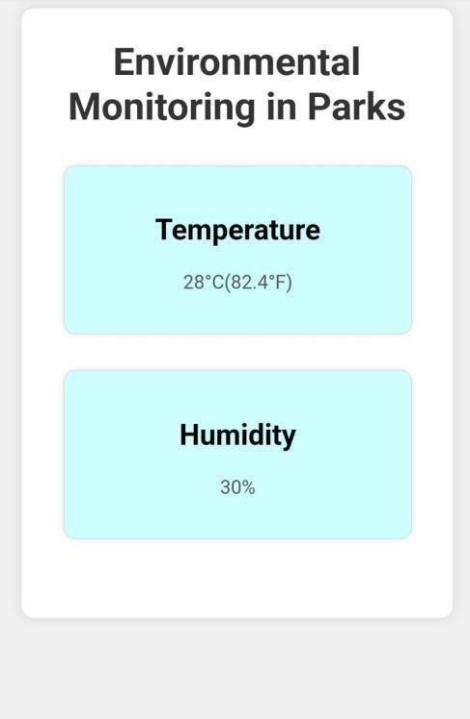
```
// Function to update temperature and humidity data function updateData(temperature,
humidity) {     const temperatureElement = document.getElementById("temperature");     const
humidityElement = document.getElementById("humidity");

   temperatureElement.textContent = ${temperature} °C;     humidityElement.textContent =
${humidity}%;
}

// Function to connect to MQTT broker and subscribe to topics  function
connectToMQTT() {

   // Replace the following with your MQTT broker details     const
brokerUrl = "mqtt://your-mqtt-broker";     const topicTemperature =
"environment/temperature";     const topicHumidity =
"environment/humidity";

   const client = new Paho.MQTT.Client(brokerUrl, "web-client-" + parseInt(Math.random() * 1e9,
10));

   // Set up the callback for a successful connection     client.onConnectionLost =
onConnectionLost;     client.onMessageArrived
= onMessageArrived;

   client.connect({
onSuccess: onConnect,        onFailure:
onFailure,
   });
```

```
  // Callback for successful connection     function
onConnect() {        console.log("Connected to MQTT
broker");        client.subscribe(topicTemperature);
client.subscribe(topicHumidity);
  }

  // Callback for connection failure     function onFailure(responseObject) {        console.log("Failed
to connect to MQTT broker: " + responseObject.errorMessage);
  }

  // Callback for connection loss     function onConnectionLost(responseObject) {
if (responseObject.errorCode !== 0) {            console.log("Connection lost: " +
responseObject.errorMessage);
      }
  }

  // Callback for received MQTT messages     function onMessageArrived(message)
{        if (message.topic === topicTemperature) {            const temperature =
parseFloat(message.payloadString).toFixed(2);
updateData(temperature, parseFloat(humidityElement.textContent));        }
else if (message.topic === topicHumidity) {            const humidity =
parseFloat(message.payloadString).toFixed(2);
updateData(parseFloat(temperatureElement.textContent), humidity);
      }
  }
}

// Call the connectToMQTT function to establish MQTT connection connectToMQTT();
```

DESIGN THE PLATFORM TO RECEIVE AND DISPLAY ENVIRONMENTAL MONITORING IN PARKS DATA

SENT BY THE IOT DEVICES





Sensor Nodes: Deploy sensor nodes throughout the park to measure temperature and humidity. Select appropriate IoT sensors, such as DHT22 or BME280, to capture the data. Ensure power sources (e.g., batteries or solar panels) for the nodes.

Communication: Use a communication protocol, such as MQTT or LoRaWAN, to transmit sensor data to a central server. Ensure network connectivity within the park. Central Server:Set up a central server to collect and store data. Implement data validation and error handling for reliability. Utilize a database (e.g., MySQL or NoSQL) to store historical data.

User Interface: Create a web-based or mobile application for users to access the data. Design an intuitive user interface to display temperature and humidity data. Implement features like real-time updates and historical data retrieval.

Alerts and Notifications: Implement alerting mechanisms for extreme environmental conditions (e.g., heatwaves or high humidity).Send notifications to users or park authorities through email, SMS, or app notifications.

Data Analysis: Incorporate data analytics to identify trends or anomalies in the environmental data. Use tools like Python and data visualization libraries to create graphs and charts.

Security: Ensure data encryption and secure authentication to protect the system from unauthorized access.

Scalability: Design the platform to easily add more sensors or expand to cover larger areas in the park.

Power Management: Optimize power usage in sensor nodes to extend battery life or reduce the need for frequent maintenance.

Maintenance and Support: Establish a maintenance plan for regular checks and sensor calibration. Provide support for users and address issues promptly.

Documentation: Create comprehensive documentation for setup, maintenance, and troubleshooting.

Regulatory Compliance: Ensure compliance with local regulations and data privacy laws when handling sensor data.

Visualization and Reporting: Enable the generation of reports for park management or research purposes.

Community Engagement: Consider involving the local community or park visitors in the project, promoting awareness and engagement.

Conclusion:

This project demonstrates how to utilize the DHT11 Temperature and Humidity Sensor in combination with NodeMCU to send real-time temperature and humidity data to the ThingSpeak IoT server. By doing so, it enables remote monitoring of environmental conditions over the internet, allowing users to access and visualize this data on the ThingSpeak dashboard. This project provides a practical example of how IoT technology can be employed for global environmental data tracking and analysis.