ENVIRONMENTAL MONITORING

Phase 5 Project Submission Document

Objective:

The primary goal of environmental monitoring in parks is to assess and track the condition of the park's natural resources and ecosystems.It aims to detect changes in environmental factors, such as air and water quality, wildlife populations, vegetation health, and climate patterns.

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:

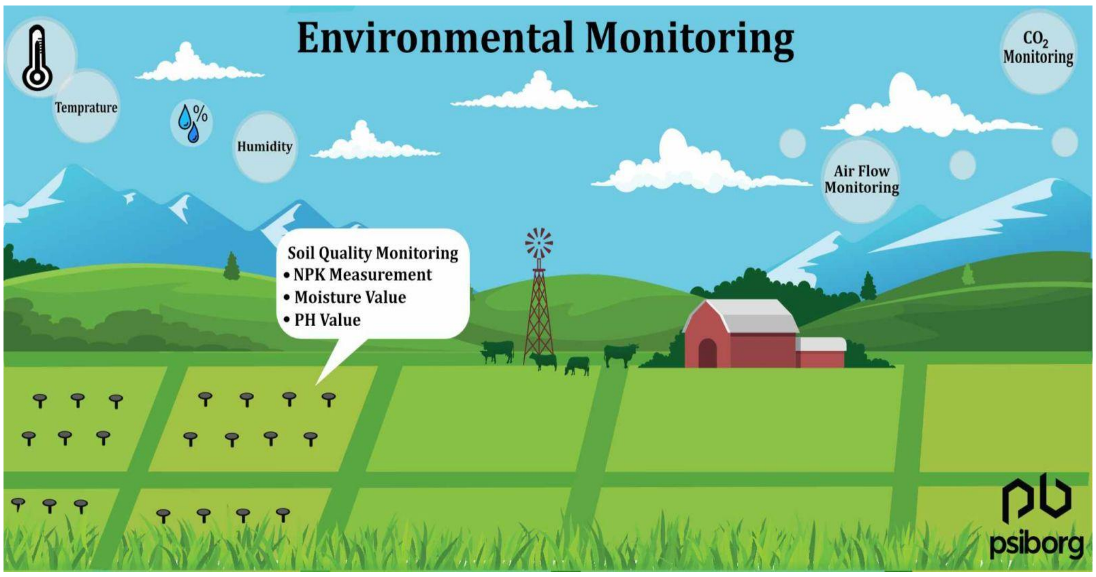
Extreme Weather Conditions:
Parks often experience a wide range of temperatures and humidity levels due to seasonal variations. Extreme heat or cold can make park visits uncomfortable or even unsafe for visitors.

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:



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.

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.

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.

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.

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

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

Data Visualization:

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

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.

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.

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.

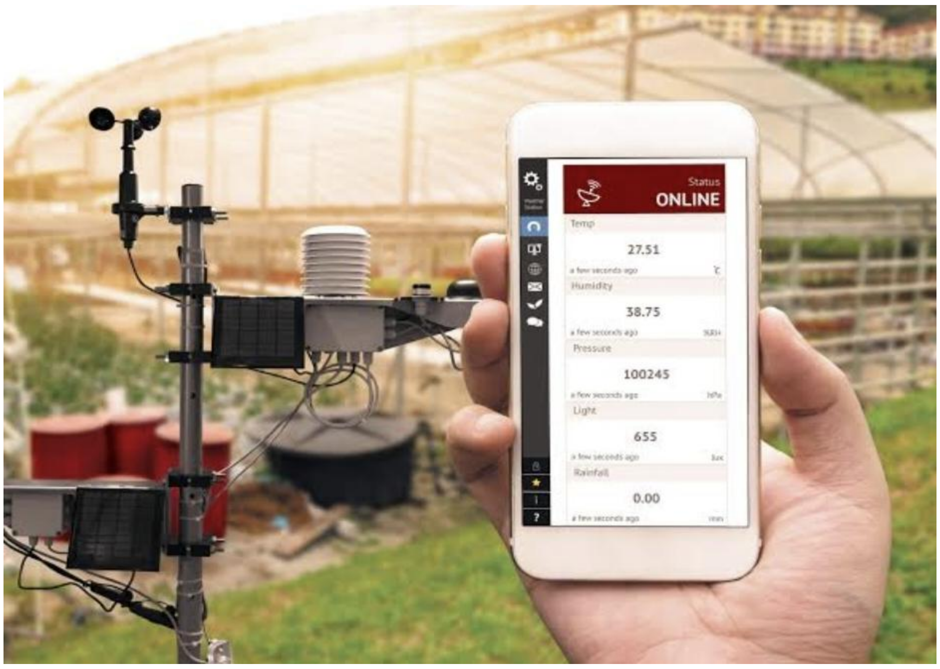
Data Security and Privacy:

Prioritize data security and privacy to protect sensitive information collected from park visitors.

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:

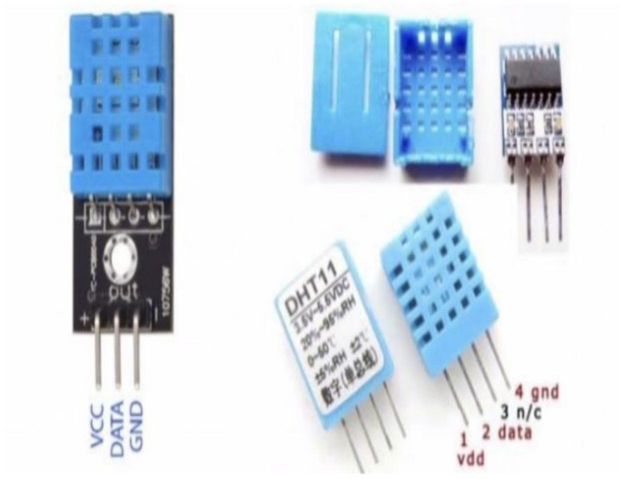
- Based on ESP8266 Wi-Fi module.
- Built-in Wi-Fi connectivity. Supports Lua scripting and Arduino IDE. GPIO pins for digital I/O and PWM.
- Analog input pins.
- UART, SPI, and I2C communication support.
- USB-to-Serial interface for programming.
- Low power consumption for battery-powered devices.
-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

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.

Program:

Python :

```
import network
import time
from machine import Pin,ADC
import dht
import ujson
from umqtt.simple import MQTTClient
# MQTT Server Parameters
MQTT_CLIENT_ID = "micropython-weather-demo"
MQTT_BROKER = "broker.mqttdashboard.com"
MQTT_USER = ""
MQTT_PASSWORD = ""
MQTT_TOPIC = "wokwi-weather"
sensor = dht.DHT22(Pin(15))
MQ7=ADC(Pin(35))
MQ8=ADC(Pin(32))
button=Pin(34,Pin.IN)
led=Pin(33,Pin.OUT)
min_rate=0
max_rate=4095
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!")
print("Connecting to MQTT server... ", end="")
client = MQTTClient(MQTT_CLIENT_ID, MQTT_BROKER, user=MQTT_USER,
password=MQTT_PASSWORD)
client.connect()
print("Connected!")
prev_weather = ""
while True:
CO_sensor=(MQ7.read())*100/(max_rate)
print("CO Sensor value: " + "%.2f" % CO_sensor +"%")
Hydrogen_sensor=(MQ8.read())*100/(max_rate)
print("Soil Sensor value: " + "%.2f" % Hydrogen_sensor +"%")
button_value=button.value()
if button_value == True:
led.value(000)
print("It's Raining")
else:
led.value(0)
print("Measuring weather conditions... ", end="")
sensor.measure()
message = ujson.dumps({
"temp": sensor.temperature(),
"humidity": sensor.humidity(),
})
if message != prev_weather:
print("Updated!")
print("Reporting to MQTT topic {}: {}".format(MQTT_TOPIC, message))
client.publish(MQTT_TOPIC, message)
prev_weather = message
else:
print("No change")
time.sleep(1)
```

C++:

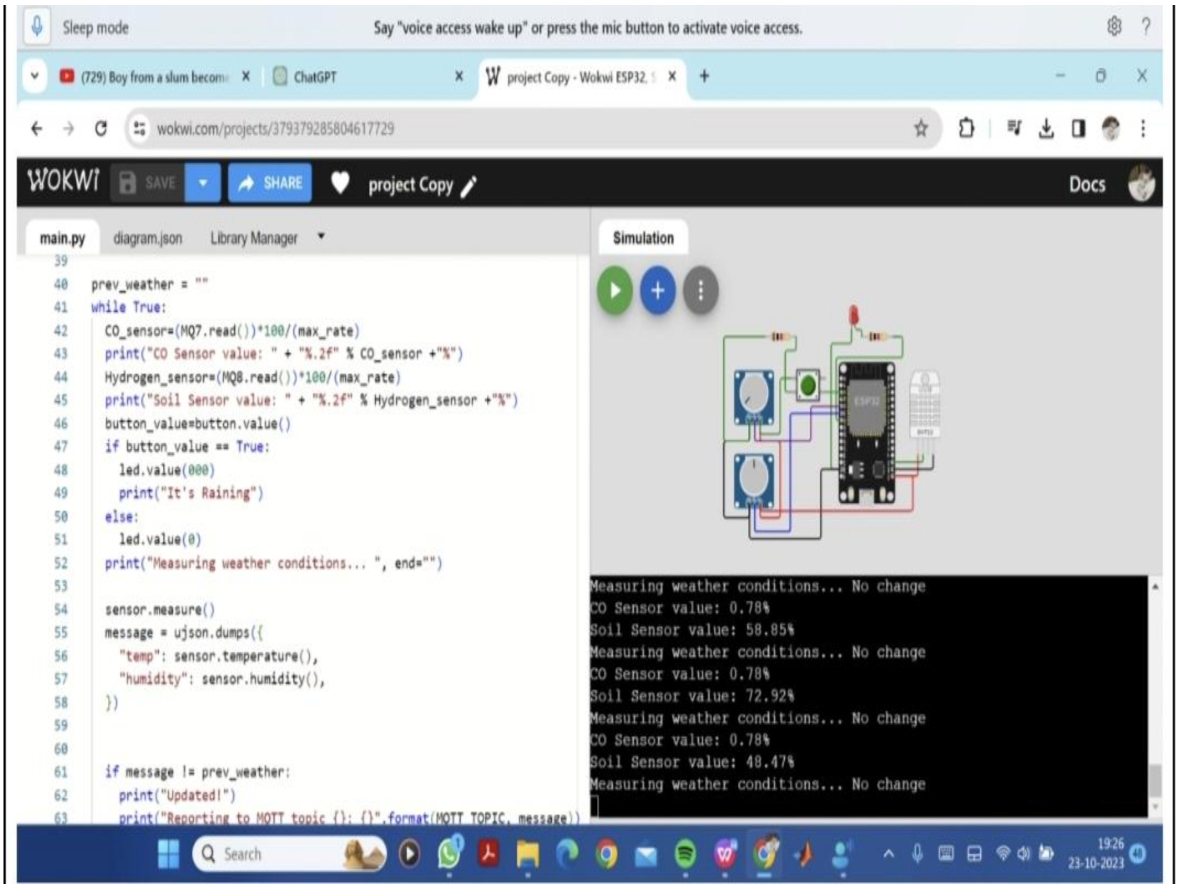
```
#include <Adafruit_Sensor.h>
#include <DHT.h>
#include <ESP8266WiFi.h>
#include <Adafruit_MQTT.h>
#include <Adafruit_MQTT_Client.h>
const char* WIFI_SSID = "YourWiFiSSID";
const char* WIFI_PASS = "YourWiFiPassword";
```

```
// Replace with your Adafruit IO credentials.
#define ADAFRUIT_IO_USERNAME "YourAdafruitUsername"
#define ADAFRUIT_IO_KEY "YourAdafruitAIOKey"
// Define the DHT sensor.
#define DHT_PIN 2 // The pin where your DHT sensor is connected.
#define DHT_TYPE DHT22 // DHT sensor type (DHT11, DHT22, AM2302, etc.)
DHT dht(DHT_PIN, DHT_TYPE);
WiFiClient client;
Adafruit_MQTT_Client mqtt(&client, "io.adafruit.com", 1883, ADAFRUIT_IO_USERNAME,
ADAFRUIT_IO_KEY);
// Define MQTT feeds.
Adafruit_MQTT_Publish temperature = Adafruit_MQTT_Publish(&mqtt,
ADAFRUIT_IO_USERNAME "/feeds/temperature");
Adafruit_MQTT_Publish humidity = Adafruit_MQTT_Publish(&mqtt,
ADAFRUIT_IO_USERNAME "/feeds/humidity");
void setup() {
Serial.begin(115200);
// Connect to Wi-Fi.
WiFi.begin(WIFI_SSID, WIFI_PASS);
while (WiFi.status() != WL_CONNECTED) {
delay(500);
Serial.println("Connecting to WiFi...");
}
Serial.println("Connected to WiFi");
// Connect to Adafruit IO.
mqtt.connect();
Serial.println("Connected to Adafruit IO");
}
void loop() {
// Read temperature and humidity data from the DHT sensor.
float temperatureValue = dht.readTemperature();
float humidityValue = dht.readHumidity();
// Publish data to Adafruit IO.
if (!isnan(temperatureValue)) {
temperature.publish(temperatureValue);
Serial.print("Temperature: ");
Serial.println(temperatureValue);
} else {
Serial.println("Failed to read temperature");
}
if (!isnan(humidityValue)) {
humidity.publish(humidityValue);
Serial.print("Humidity: ");
Serial.println(humidityValue);
} else {
Serial.println("Failed to read humidity");
}
delay(60000); // Delay for 60 seconds (adjust as needed).
}
```

C program:

```
#include <Adafruit_Sensor.h>
#include <DHT.h>
#include <ESP8266WiFi.h>
#include <Adafruit_MQTT.h>
#include <Adafruit_MQTT_Client.h>
// Replace these with your Wi-Fi credentials.
const char* WIFI_SSID = "YourWiFiSSID";
const char* WIFI_PASS = "YourWiFiPassword";
// Replace with your Adafruit IO credentials.
#define ADAFRUIT_IO_USERNAME "YourAdafruitUsername"
#define ADAFRUIT_IO_KEY "YourAdafruitAIOKey"
// Define the DHT sensor.
#define DHT_PIN 2 // The pin where your DHT sensor is connected.
#define DHT_TYPE DHT22 // DHT sensor type (DHT11, DHT22, AM2302, etc.)
DHT dht(DHT_PIN, DHT_TYPE);
WiFiClient client;
Adafruit_MQTT_Client mqtt(&client, "io.adafruit.com", 1883, ADAFRUIT_IO_USERNAME,
ADAFRUIT_IO_KEY);
// Define MQTT feeds.
Adafruit_MQTT_Publish temperature = Adafruit_MQTT_Publish(&mqtt,
ADAFRUIT_IO_USERNAME "/feeds/temperature");
Adafruit_MQTT_Publish humidity = Adafruit_MQTT_Publish(&mqtt,
ADAFRUIT_IO_USERNAME "/feeds/humidity");
void setup() {
Serial.begin(115200);
// Connect to Wi-Fi.
WiFi.begin(WIFI_SSID, WIFI_PASS);
while (WiFi.status() != WL_CONNECTED) {
delay(500);
Serial.println("Connecting to WiFi...");
}
Serial.println("Connected to WiFi");
// Connect to Adafruit IO.
mqtt.connect();
Serial.println("Connected to Adafruit IO");
}
void loop() {
// Read temperature and humidity data from the DHT sensor.
float temperatureValue = dht.readTemperature();
float humidityValue = dht.readHumidity();
// Publish data to Adafruit IO.
if (!isnan(temperatureValue)) {
```

```
temperature.publish(temperatureValue);
Serial.print("Temperature: ");
Serial.println(temperatureValue);
} else {
Serial.println("Failed to read temperature");
}
if (!isnan(humidityValue)) {
humidity.publish(humidityValue);
Serial.print("Humidity: ");
Serial.println(humidityValue);
} else {
Serial.println("Failed to read humidity");
}
delay(60000); // Delay for 60 seconds (adjust as needed).
}
```


Sleep mode
Say "voice access wake up" or press the mic button to activate voice access.
(729) Boy from a slum becom
ChatGPT
project Copy - Wokwi ESP32
wokwi.com/projects/379379285804617729
WOKWI
SAVE
SHARE
project Copy
Docs
main.py
diagram.json
Library Manager
Simulation
Search
19:26
23-10-2023

```

prev_weather = ""
while True:
  CO_sensor=(MQ7.read())*100/(max_rate)
  print("CO Sensor value: " + "%.2f" % CO_sensor +"%")
  Hydrogen_sensor=(MQ8.read())*100/(max_rate)
  print("Soil Sensor value: " + "%.2f" % Hydrogen_sensor +"%")
  button_value=button.value()
  if button_value == True:
    led.value(000)
    print("It's Raining")
  else:
    led.value(0)
  print("Measuring weather conditions... ", end="")

  sensor.measure()
  message = ujson.dumps({
    "temp": sensor.temperature(),
    "humidity": sensor.humidity(),
  })


  if message != prev_weather:
    print("Updated!")
    print("Reporting to MQTT topic {}: {}".format(MQTT_TOPIC, message))
```

```
Measuring weather conditions... No change
CO Sensor value: 0.78%
Soil Sensor value: 58.85%
Measuring weather conditions... No change
CO Sensor value: 0.78%
Soil Sensor value: 72.92%
Measuring weather conditions... No change
CO Sensor value: 0.78%
Soil Sensor value: 48.47%
Measuring weather conditions... No change
```

Conclusion:

The implementation of IoT (Internet of Things) technology for environmental monitoring in parks holds immense potential for advancing our understanding of these vital natural spaces. By leveraging IoT sensors, data analytics, and real-time connectivity, we can transform parks into smarter, more resilient ecosystems.
This innovation not only enhances our ability to protect and conserve these natural wonders but also enriches the park experience for visitors. Through the power of data and technology, we can foster a deeper connection between people and nature.
As we move forward in embracing IoT for environmental monitoring in parks, we look forward to collaboration, investment, and partnerships that will help us turn this vision into a reality. Together, we can create a brighter, more sustainable future for our parks, benefiting both current and future generations.