KINGS ENGINEERING COLLEGE

PROJECT TITLE: ENVIRONMENTAL MONITORING

TEAM MEMBERS:

G.DEEPIKA,T.DHANALAKSHMI,D.DHARSHINI,D.S.GEETHA

DEPARTMENT: B.E/BIO MEDICAL ENGINEERING

MENTOR: MARY LALITHA

BUILDING THE IOT ENABLED ENVIRONMENTAL MONITORING IN PARKS SYSTEM:

Building an IoT-enabled environmental monitoring system for parks involves a combination of hardware and software components to collect, transmit, and analyze data from various sensors. This system can help monitor and manage environmental conditions, such as air quality, weather, soil moisture, and more.

1. Define Your Objectives: Identify the specific environmental parameters you want to monitor in the park (e.g., air quality, temperature, humidity, soil moisture, water quality).Determine the purpose of your monitoring system, such as improving park management, protecting ecosystems, or enhancing visitor experiences.

2. Select Sensors: Consider factors like accuracy, range, and power consumption. Common environmental sensors include air quality sensors (for pollutants like CO2, PM2.5, and ozone), temperature and humidity sensors, soil moisture sensors, and water quality sensors.

3. IoT Hardware: Select IoT hardware components such as microcontrollers (e.g., Arduino, Raspberry Pi), communication modules (e.g., Wi-Fi, LoRa, NB-IoT), and power sources (solar panels or batteries).

4. Data Transmission: Choose a data transmission protocol and platform (e.g., MQTT, HTTP, AWS IoT, Azure IoT) to send data from the sensors to a central database or cloud.

5. Data Storage and Management: Set up a cloud-based database to store incoming sensor data. Implement data management and analysis tools to process and visualize the data. Tools like AWS IoT Core, Azure IoT Hub, or custom web applications can be used for this purpose.

6. Power Management: Optimize power usage to extend the lifespan of your IoT devices. This may involve using low-power sensors and sleep modes for the hardware.

7. Visualization and Alerts: Develop a dashboard or application to display real-time and historical data for park administrators and visitors. Implement alerting mechanisms to notify stakeholders when certain environmental parameters exceed predefined thresholds.

8. Connectivity: Ensure the system has a reliable internet connection. This could be through a dedicated network, public Wi-Fi, or cellular networks.

9. Scalability and Redundancy: Design the system to be scalable as your monitoring needs grow. Implement redundancy measures to ensure continuous data collection even in case of sensor or network failures.

10. Security: Implement strong security measures to protect data privacy and prevent unauthorized access to the system.

11. Compliance and Regulations: Ensure that your system complies with environmental regulations and data privacy laws.

12. Testing and Deployment: Thoroughly test the entire system in a controlled environment before deploying it in the park. Deploy the system across the park, taking into consideration sensor placement and power supply.

Deploy IoT devices (e.g., temperature and humidity sensors) in various locations within public parks to measure environmental conditions:

- **Identify Objectives and Requirements**: Define the specific goals of the environmental monitoring project. What environmental conditions do you want to measure, and why? Determine the number of sensors needed, their types, and the data collection frequency.
- **Select Suitable Sensors**: Choose appropriate sensors for measuring temperature and humidity. Make sure they are weather-resistant and accurate. Consider power source options, such as battery-powered sensors or solar panels.
- **Connectivity and Data Storage**: Set up a reliable communication network to transmit data from the sensors to a central location. Options include Wi-Fi, LoRa, or cellular connectivity. Choose a data storage solution to securely store and manage the collected data, such as cloud-based databases or local servers.
- **Power Supply**: Ensure the sensors have a stable power supply. This may involve using batteries, solar panels, or other suitable energy sources.

- **Data Transmission and Reception**: Implement a communication protocol for the sensors to send data to a central server or gateway. Ensure that the central system can receive and process the incoming data.
- **Data Analysis and Visualization**: Develop software for data analysis and visualization. This can involve real-time monitoring, historical data analysis, and the creation of user-friendly dashboards.
- **Maintenance and Calibration**: Regularly check and calibrate the sensors to ensure accurate data collection. Establish a maintenance schedule to replace batteries and perform any necessary repairs.
- **Data Security**: Implement robust security measures to protect the data, as environmental data can be sensitive. Encrypt data during transmission and storage.
- **Compliance and Permissions**: Ensure compliance with local regulations and obtain any necessary permits for data collection and device deployment in public parks.

Develop a Python script on the IoT devices to send real-time environmental data to the monitoring platform:

To develop a Python script for IoT devices to send real-time environmental data to a monitoring platform, you will need to break the task into several components:

- **Data Collection**: Collect environmental data from sensors attached to your IoT device. This could include data such as temperature, humidity, air quality, or any other relevant parameters.
- **Data Processing**: Process the collected data if necessary. This might involve data filtering, aggregation, or any other transformations required before sending it to the monitoring platform.
- **Communication with the Monitoring Platform**: Establish a connection to the monitoring platform and send the processed data to it. You'll likely need to use a network protocol such as HTTP, MQTT, or CoAP, depending on the platform's requirements.
- **Error Handling**: Implement error handling to deal with connectivity issues or platform unavailability.

PYTHON SCRIPT:

```
import paho.mqtt.client as mqtt

import time

import json

import random
```

```
 # Define your IoT device's information
device_id = "your_device_id"
mqtt_broker_address = "mqtt.example.com"
mqtt_port = 1883
topic = "environmental_data"
 # Create an MQTT client
client = mqtt.Client(client_id=device_id)
 # Connect to the MQTT broker
client.connect(mqtt_broker_address, mqtt_port, 60)
while True:
    # Simulate environmental data (replace with real sensor data)
    temperature = random.uniform(20.0, 30.0)
    humidity = random.uniform(40.0, 60.0)
    data = {
        "temperature": temperature,
        "humidity": humidity
    }
 # Convert data to JSON
    payload = json.dumps(data)
   # Publish data to the MQTT topic
    client.publish(topic, payload)
   print(f"Published: {payload}")
    # Adjust the time interval according to your data collection frequency
    time.sleep(10)
```

```
# Close the MQTT connection (not reached in this example)
client.disconnect()
```