ENIRONMENT MONITORING

Building an IoT-enabled Environmental Monitoring in Parks system involves several steps, from setting up the hardware to developing the software for data collection and transmission. Below is a high-level overview of how you can start building the system:

Step 1: Hardware Setup

1. **Select IoT Devices**: Choose IoT devices (temperature and humidity sensors) suitable for outdoor use and with the capability to connect to a network, such as Wi-Fi or LoRa.
2. **Determine Sensor Locations**: Identify suitable locations within the public parks for deploying sensors. Ensure that they are well-distributed to capture a representative view of environmental conditions.
3. **Power Supply**: Ensure the IoT devices have a stable power source. You may need to use batteries, solar panels, or other power sources depending on your specific setup.
4. **Network Connectivity**: Ensure that the IoT devices can connect to the internet or a local network. This might require setting up Wi-Fi access points or LoRa gateways.

Step 2: IoT Device Programming

1. **Choose a Development Board**: Select an IoT development board (e.g., Raspberry Pi, Arduino, ESP8266/ESP32) and write code in Python for it.
2. **Sensor Integration**: Write code to interface with the temperature and humidity sensors. You'll need libraries or drivers specific to your chosen sensors.
3. **Data Collection**: Collect data from the sensors. Ensure that you take into account factors like data accuracy, sampling frequency, and data storage.
4. **Data Transmission**: Develop a Python script that sends the real-time environmental data to your monitoring platform. Use suitable communication protocols (HTTP, MQTT, or other depending on your platform).
5. **Error Handling**: Implement error-handling mechanisms to ensure the robustness of data transmission, especially in challenging outdoor environments.
6. **Security**: Implement basic security measures to protect the data transmitted. You might need to use encryption or authentication mechanisms.

Step 3: Monitoring Platform

1. **Set Up a Monitoring Platform**: Create a server or cloud-based platform to receive and store the data from the IoT devices. This platform could be hosted on a cloud service like AWS, Azure, or a local server.
2. **Data Reception**: Develop the backend of your platform to receive data from IoT devices. This typically involves building RESTful APIs, MQTT brokers, or other data ingestion mechanisms.
3. **Data Storage**: Store the received data in a database or storage solution. Ensure data is timestamped for real-time monitoring and historical analysis.

4. **Data Visualization**: Develop a frontend or web interface for visualizing the environmental data. You can use libraries like D3.js, Plotly, or frontend frameworks like React or Angular for this purpose.

Step 4: Real-Time Monitoring

1. **Dashboard**: Create a dashboard that displays real-time environmental data from the sensors in public parks. Include charts, graphs, and maps to provide an intuitive view.
2. **Alerting**: Implement alerting mechanisms to notify relevant authorities or users when environmental conditions exceed predefined thresholds. This is crucial for emergency situations.

Step 5: Testing and Deployment

1. **Testing**: Thoroughly test your IoT devices, data transmission, and monitoring platform in controlled environments before deploying them in public parks.
2. **Deployment**: Once testing is successful, deploy the IoT devices in the selected locations within public parks.
3. **Maintenance**: Ensure regular maintenance, including replacing batteries, firmware updates, and monitoring system health.

Step 6: Documentation and Submission

1. **Documentation**: Prepare detailed documentation that includes hardware configurations, code, and system architecture.
2. **Submission**: Submit your assignment notebook, which should contain code samples, hardware setup documentation, and an overview of the system's architecture.

To develop a Python script on the IoT devices to send real-time environmental data to a monitoring platform, you'll need to write code that reads data from the environmental sensors and transmits it to your platform using an appropriate communication protocol (e.g., HTTP or MQTT). Below is a basic example using Python that you can adapt to your specific hardware and sensor setup.

In this example, we'll use the `requests` library for HTTP communication. Make sure to install the library if it's not already installed on your IoT device.

```
import requests

import time

import random


# Replace with your monitoring platform's API endpoint

API_ENDPOINT = 'https://your-monitoring-platform.com/api/data'


# Replace with your IoT device's unique identifier

DEVICE_ID = 'your-device-id'
```

```
def read_environmental_data():
    # Replace this with code to read data from your environmental sensors
    temperature = random.uniform(20.0, 30.0)  # Example temperature reading
    humidity = random.uniform(40.0, 60.0)     # Example humidity reading
    return {'temperature': temperature, 'humidity': humidity}

def send_data_to_platform(data):
    headers = {
        'Content-Type': 'application/json'
    }

    payload = {
        'device_id': DEVICE_ID,
        'data': data,
        'timestamp': int(time.time())
    }

    try:
        response = requests.post(API_ENDPOINT, json=payload, headers=headers)
        if response.status_code == 200:
            print('Data sent successfully')
        else:
            print(f'Failed to send data. Status code: {response.status_code}')
    except Exception as e:
        print(f'Error: {str(e)}')

if __name__ == '__main__':
    while True:
        environmental_data = read_environmental_data()
        send_data_to_platform(environmental_data)
        time.sleep(60)  # Adjust the time interval based on your data collection frequency
```

Please note that in this example, we use random values for temperature and humidity to simulate sensor data. You should replace this with actual code to read data from your specific sensors.

Additionally, you'll need to replace `API_ENDPOINT` with the actual URL of your monitoring platform's API, and `DEVICE_ID` with a unique identifier for your IoT device. Make sure your monitoring platform has a compatible API that can accept incoming data.

Remember that building an IoT-enabled environmental monitoring system is a complex project, and each step requires careful planning and execution. You might also need to comply with local regulations and obtain any necessary permissions for deploying IoT devices in public parks.