**Environmental Monitoring**

**Objectives:**

The primary goal of the project is to establish an IoT-based environmental monitoring system within park environments. This involves real-time data collection and analysis to ensure the quality of air, water, noise levels, and temperature for the benefit of park visitors.

**IoT Device Deployment:**

**1. Sensor Selection:**

* Choose appropriate sensors based on the environmental parameters you want to monitor (air quality, temperature, humidity, etc.).
* Ensure the sensors are suitable for outdoor use and can withstand environmental conditions.

**2. Hardware Setup:**

* Connect the sensors to microcontrollers (e.g., Arduino, Raspberry Pi) using appropriate wiring and interfaces.
* Power the devices and ensure they're functioning correctly.

**3. Location Planning:**

* Strategically place sensors in the park, considering factors such as air circulation, exposure to environmental elements, and proximity to potential sources of pollution (roads, waste disposal areas, etc.).

**4. Connectivity:**

* Set up the devices for connectivity. This might involve using Wi-Fi, Bluetooth, or other communication modules.
* Ensure a stable and secure internet connection for data transmission to the IoT platform.

**Platform Development:**

**1. IoT Platform Selection:**

* Choose a suitable IoT platform that aligns with the project's requirements. Options include cloud-based platforms like AWS IoT, Microsoft Azure IoT, Google Cloud IoT, or open-source platforms like ThingsBoard or MQTT.
* Consider factors like scalability, data storage, security features, and analytics capabilities.

**2. Setup and Configuration:**

* Set up the chosen IoT platform and configure the necessary parameters for data ingestion, storage, and visualization.
* Establish connections between the IoT devices and the platform. This may involve registering devices and obtaining credentials for secure data transmission.

**3. Data Collection and Processing:**

* Implement code or scripts to receive and process incoming data from the IoT devices.
* Apply filters, if needed, to clean and organize the data before storing it on the platform.

**4. Database Management:**

* Create a database within the platform to store the collected environmental data securely.
* Design an efficient database schema that accommodates various types of sensor data.

**5. Real-time Data Visualization:**

* Develop a user interface or dashboard to visualize the collected environmental data in real-time.
* Implement charts, graphs, or maps to display air quality, temperature, humidity, noise levels, and other monitored parameters.

**Code Implementation:**

Arduino/Raspberry Pi Code: Developing code to collect data from sensors and transmit it to the IoT platform.

Platform Development: Using Python for back-end processing and data management within the IoT platform.

**Benefits of Environmental Monitoring:**

* Early Detection of Pollution: Real-time monitoring allows for the early detection of environmental pollutants, such as air and water contaminants. This aids in prompt intervention and mitigation strategies.
* Improved Public Health: Identifying high pollution levels enables authorities to take proactive measures, reducing health risks associated with poor air or water quality. Alerts and advisories can be issued to the public in real-time.
* Sustainable Resource Management: Monitoring environmental parameters aids in better resource management. For instance, understanding water quality in a park ecosystem helps in maintaining a balanced ecosystem and preserving natural habitats.
* Data-Driven Decision Making: Collected data facilitates evidence-based decision-making for urban planning, waste management, and policy development to ensure a healthier environment.
* Enhanced Safety Measures: Monitoring systems help in creating safer environments by alerting authorities and the public to hazardous conditions, such as high pollution levels or noise disturbances.
* Educational and Awareness Purposes: Providing real-time data to the public fosters environmental awareness. It educates and engages communities about environmental issues and encourages sustainable practices.
* Economic Benefits: Preserving environmental quality supports industries such as tourism and recreation. A well-maintained natural environment in parks, for instance, attracts visitors and contributes to the local economy.
* Adaptation and Mitigation of Climate Change: Monitoring temperature changes, carbon emissions, and other climate-related factors can help in understanding climate patterns, facilitating planning for adaptation and mitigation strategies.
* Long-term Environmental Planning: Continuous monitoring enables the collection of historical data, which is invaluable for predicting trends, analyzing long-term impacts, and planning for the future.
* Efficient Resource Allocation: By knowing real-time environmental conditions, resources like energy, water, and waste management can be allocated more efficiently, reducing unnecessary consumption and waste.

**Submission Guidelines:**

**Replication Instructions:**

**Hardware Setup:**

Components Required:

* Air Quality Sensor: (e.g., MQ-2, MQ-7, or similar)
* Temperature and Humidity Sensor: (e.g., DHT11, DHT22, or similar)
* Arduino or Raspberry Pi board
* Breadboard
* Jumper wires
* Resistors (if required by specific sensors)

Wiring and Connections for Arduino or Raspberry Pi:

Air Quality Sensor:

* Connect VCC to 5V on Arduino/Raspberry Pi.
* Connect GND to GND on Arduino/Raspberry Pi.
* Connect Analog Output to an Analog Pin on the board (e.g., A0 for Arduino).

Temperature and Humidity Sensor:

For DHT11:

* Connect VCC to 5V.
* Connect GND to GND.
* Connect Data pin to a digital GPIO pin (e.g., pin 2 on Arduino).

For DHT22:

* Connect VCC to 3.3V on Raspberry Pi or 5V on Arduino.
* Connect GND to GND.
* Connect Data pin to a digital GPIO pin (e.g., pin 2 on Arduino).

**IoT Platform Setup:**

1. Selecting ThingsBoard IoT Platform:

* Visit the ThingsBoard website: ThingsBoard

2. Account Setup:

* Sign Up and Create an Account:
* Go to the ThingsBoard website and sign up for an account.
* Fill in the required details and confirm your email to activate the account.
* Accessing the Dashboard:
* After signing up and logging in, you will land on the ThingsBoard dashboard.

3. Project Creation:

* Once logged in, navigate to the menu options to create a new project.
* Name your project (e.g., "Environmental Monitoring in Parks") and proceed to create it

**Data Transmission:**

**Data Collection from Arduino/Raspberry Pi:**

Arduino Data Collection (Sample code using Arduino IDE):

void setup() {

Serial.begin(9600);

// Initialize sensors and necessary components

}

void loop() {

// Read sensor data

int sensorValue = analogRead(A0); // Sample sensor reading, replace with your sensor data collection

// Send data via serial

Serial.print("SensorData:");

Serial.println(sensorValue);

delay(1000); // Adjust as needed for data transmission interval

}

**Raspberry Pi Data Collection (Sample code using Python):**

# Python code to read sensor data from Raspberry Pi and format for transmission

import serial

import time

ser = serial.Serial('/dev/ttyUSB0', 9600) # Change port and baud rate as per your setup

while True:

if ser.in\_waiting > 0:

data = ser.readline().decode('utf-8').strip()

if data.startswith('SensorData:'):

sensor\_value = data.split(':')[1]

# Process and structure the sensor data here

# Code to transmit data to IoT platform using MQTT or HTTP

# (See the section below for data transmission examples)

time.sleep(1) # Adjust as needed for data transmission interval

**Data Transmission to IoT Platform (Sample using MQTT):**

Using Paho MQTT library in Python:

import paho.mqtt.client as mqtt

# Callback functions for MQTT connection

def on\_connect(client, userdata, flags, rc):

print("Connected with result code "+str(rc))

def on\_disconnect(client, userdata, rc):

print("Disconnected with result code "+str(rc))

# Define the data to be sent

sensor\_data = "your\_processed\_data\_here"

# Configure MQTT connection details

client = mqtt.Client()

client.on\_connect = on\_connect

client.on\_disconnect = on\_disconnect

# Connect to the MQTT broker

client.connect("your\_broker\_address", your\_port\_number, keepalive=60)

# Publish the sensor data

client.publish("topic", sensor\_data) # Change 'topic' to your specific MQTT topic

# Loop to maintain the connection

client.loop\_forever()

**API Integration with the IoT Platform:**

Example of integrating with an API (generic example):

import requests

# API endpoint details

url = "your\_platform\_api\_endpoint\_here"

headers = {

"Authorization": "your\_api\_token",

"Content-Type": "application/json"

}

# Define the payload/data to be sent

data = {

"sensor\_data\_field": "your\_processed\_data\_here"

}

# Send data to the IoT platform using an HTTP POST request

response = requests.post(url, json=data, headers=headers)

# Check the response status

if response.status\_code == 200:

print("Data sent successfully")

else:

print("Failed to send data")

**Example Outputs:**

**| Timestamp | Sensor\_ID | Parameter | Value | Unit |**

**|------------------------|----------------|-----------------|----------|----------|**

**| 2023-10-15 08:00:00 | AQ001 | PM2.5 | 12.3 | µg/m³ |**

**| 2023-10-15 08:00:00 | AQ001 | PM10 | 28.7 | µg/m³ |**

**| 2023-10-15 08:00:00 | AQ002 | CO | 2.1 | ppm |**

**| 2023-10-15 08:00:00 | AQ002 | NO2 | 18.5 | ppb |**

**| 2023-10-15 08:00:00 | Temp001 | Temperature | 25.6 | °C |**

**| 2023-10-15 08:00:00 | Humidity01 | Humidity | 52.4 | % |**

**| 2023-10-15 08:15:00 | AQ001 | PM2.5 | 12.7 | µg/m³ |**

**| 2023-10-15 08:15:00 | AQ001 | PM10 | 29.1 | µg/m³ |**

**| 2023-10-15 08:15:00 | AQ002 | CO | 2.0 | ppm |**

**| 2023-10-15 08:15:00 | AQ002 | NO2 | 18.3 | ppb |**

**| 2023-10-15 08:15:00 | Temp001 | Temperature | 25.7 | °C |**

**| 2023-10-15 08:15:00 | Humidity01 | Humidity | 52.1 | % |**