HEADER:

9512-JP COLLEGE OF ENGINEERING, Ayikudi,

Department of electronics and communication engineering

Smart Water foundation

Team members:

R. harish : harishrgul@gmail.com

J. sivalingam : sivalingamj123@gmail.com

L. mariraj : marirajlaksumanan@gmail.com

Mohammed ishok : ishackissz@gmail.com

PROCESS AND DATA:

1. **Defining Project Requirements:**
    - **Begin by clearly defining the objectives of your Smart Water Management project. What specific data are you looking to collect and analyze? What problems are you trying to solve? Understanding your project's goals is essential.**
2. **IoT Device Deployment:**
    - **Choose the appropriate IoT devices for your project. These devices should be capable of collecting, processing, and transmitting data to a central server or cloud platform.**
    - **Deploy these IoT devices strategically in the areas where water management is crucial. Ensure they are properly powered and connected to the internet.**
3. **Developing Python Script:**
    - **Develop a Python script to interface with the sensors and IoT devices. This script should be able to collect data from the sensors, process it, and transmit it to a central database or cloud platform for analysis.**
    - **Ensure the script is robust, capable of handling data from multiple sensors, and includes error-handling mechanisms.**
4. **Data Analysis and Visualization:**

- **Set up a data analysis platform that can receive data from your IoT devices and perform relevant analyses. Tools like Python libraries (e.g., Pandas, Matplotlib, Seaborn) can be used for data analysis and visualization.**

5. **Remote Monitoring and Alerts:**
    - **Implement remote monitoring and alerting systems to notify relevant stakeholders when specific water parameters go beyond predefined thresholds. This can be done through emails, SMS, or mobile apps.**
6. **Documentation and Assessment:**
    - **Create comprehensive documentation of your project, including details about the deployed IoT devices, sensors, the Python script, data analysis, and any findings or insights.**

- **USED SENSORS:**
    - **To monitor water quality, quantity, and related parameters, you can use various sensors, including: a. Water Quality Sensors: These sensors can measure parameters like pH, turbidity, dissolved oxygen, and chemical contaminants. Some common water quality sensors include:**
        - **pH sensors**
        - **Turbidity sensors**
        - **Dissolved oxygen sensors**
        - **Conductivity sensors**
        - **Total organic carbon (TOC) sensors**

b. Water Level Sensors: These sensors are used to measure the water level in reservoirs, tanks, and rivers. They can be:

- **Ultrasonic sensors**
- **Pressure sensors**
- **Float switches**

c. Flow Sensors: Flow sensors are used to measure the rate at which water is flowing through pipes. Some types of flow sensors include:

- **Electromagnetic flowmeters**
- **Ultrasonic flowmeters**
- **Turbine flow sensors**

d. Temperature Sensors: Temperature sensors can be important for monitoring the temperature of water bodies. Common types include:

- Thermocouples
- Resistance Temperature Detectors (RTDs)
- Thermistors

- **PYTHON SCRIPT:**

```
import serial
import time
import requests
import random

# Define your ThingSpeak API key and channel URL
api_key = "G1RSE98QHVUFT626"
url = f"https://api.thingspeak.com/update?api_key={api_key}"

# Initialize the serial connection to Arduino
arduino_port = '/dev/ttyACM0'  # Update with your Arduino's port
ser = serial.Serial(arduino_port, 9600)

# Function to read data from Arduino
def read_arduino_data():
    data = ser.readline().decode().strip()
    return data

# Simulated sensor data (replace with actual sensor readings)
def read_sensor_data():
    temperature = random.uniform(20.0, 30.0)
    water_level = random.uniform(0, 100)
    return temperature, water_level
```

```
while True:
    try:
        # Read data from Arduino
        arduino_data = read_arduino_data()

        # Read sensor data
        temperature, water_level = read_sensor_data()

        # Prepare data to send to ThingSpeak
        data = {
            'field1': temperature,
            'field2': water_level,
            'field3': arduino_data
        }

        # Send data to ThingSpeak
        response = requests.post(url, data=data)
        print("Data Sent to ThingSpeak")

    except Exception as e:
        print(f"Error: {e}")

    # Set the data upload interval
    time.sleep(300)  # Upload data every 5 minutes (adjust as needed)
```