KGiSL Institute of Technology
(Affiliated to ANNA University, Chennai and Approved by AICTE, New Delhi)
365, KGiSL Campus, Thudiyalur Road, Saravanampatti
Coimbatore – 641035

Department of Artificial Intelligence and Data Science

NAAN MUDHALVAN – IOT

PROBLEM STATEMENT : SMART WATER MANAGEMENT

MATERIAL AND COMPONENTS REQUIRED :

1. Sensors for water quality and quantity.
2. Data loggers to record sensor data.
3. Communication equipment for data transmission.
4. Control systems (SCADA or PLCs).
5. Remote monitoring software.
6. Pumps, valves, and water meters.
7. Distributed Control Units (DCUs).
8. Weather stations for meteorological data.
9. Power supply or batteries.
10. Software and data storage.
11. Security systems for protection.
12. GIS for mapping.
13. Water treatment equipment if needed.
14. Backup systems for reliability.
15. Installation and maintenance tools.
16. Training materials and documentation.

PROCEDURE :

Step 1: Select the Appropriate IoT Sensors

Choose IoT sensors suitable for measuring water consumption. In this case, flow meters are a common choice. Ensure they are compatible with the type of water infrastructure in the public place (e.g., plumbing, irrigation, or industrial water systems).

Step 2: Hardware Installation

Install the selected flow meters at the appropriate locations in the public place. Ensure they are properly connected to the water supply lines. Depending on the type of flow meter, you may need to consult a professional plumber for installation.

Step 3: Sensor Calibration

Calibrate the flow meters to ensure accurate measurements. Calibration involves setting the flow meter to accurately measure the flow rate of water based on specific units (e.g., liters per minute or gallons per minute). Manufacturers often provide calibration instructions.

Step 4: Sensor Data Output

Determine how the flow meters provide data output. Many modern flow meters offer digital outputs such as pulse signals or use communication protocols like Modbus or 4-20mA analog signals. Ensure you understand how to access data from your specific flow meters.

Step 5: IoT Device Selection

Select an IoT device to connect to the flow meters and collect data. A Raspberry Pi, Arduino, or specialized IoT gateway may be suitable. Ensure the IoT device has the necessary communication interfaces to interface with the flow meters.

Step 6: Data Collection Script

Develop a Python script on the IoT device to collect data from the flow meters. This script will depend on the communication protocol used by the flow meters. Here's a simplified example using Python for reading data from a flow meter (replace with your specific flow meter's communication method):

```
import serial

# Create a serial connection to the flow meter

ser = serial.Serial('/dev/ttyUSB0', baudrate=9600, timeout=1)

while True:

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

    print(f"Water consumption: {data} liters per minute")
```

Step 7: Data Storage and Transmission

Incorporate code within your script to store and transmit the data. You can store the data locally or send it to a remote server or cloud platform for further analysis and monitoring. Ensure that you have the necessary credentials and endpoints for data transmission.

Step 8: Monitoring and Analysis

Set up monitoring and analysis tools to process the data collected by the IoT sensors. This could include developing data analysis algorithms or using third-party software platforms to interpret the data.

Step 9: Maintenance and Calibration

Regularly maintain the sensors and IoT devices to ensure they continue to function correctly. Recalibrate the sensors periodically to maintain measurement accuracy.

Step 10: Data Presentation and User Interface

Create a user interface or dashboard to visualize the water consumption data for monitoring and reporting. You can use web development technologies to present data in a user-friendly manner.

PYTHON SCRIPT :

```
#Manage.py
    import os
    import sys
    def run_django_app():
        os.environ.setdefault("DJANGO_SETTINGS_MODULE", "alpha.settings")
        try:
            from django.core.management import execute_from_command_line
        except ImportError:
            try:
                import django
            except ImportError:
                raise ImportError("Django is not available. Ensure it's installed and on your
    PYTHONPATH.")
            raise
    execute_from_command_line(sys.argv)
    if __name__ == "__main__":
        run_django_app()
#Using Raspberry Pi
    import os
    import sys
    import RPi.GPIO as GPIO
    import time
```

```
import requests

# Set the DJANGO_SETTINGS_MODULE environment variable
os.environ.setdefault("DJANGO_SETTINGS_MODULE", "alpha.settings")

# Try to import the Django management module
try:
    from django.core.management import execute_from_command_line
except ImportError:
    try:
        import django
    except ImportError:
        raise ImportError(
            "Couldn't import Django. Make sure it's installed and "
            "available on your PYTHONPATH. Did you forget to activate a virtual environment?"
        )
    raise

# Execute the Django management commands
execute_from_command_line(sys.argv)

# Initialize UUGear library and configure it
UUGearDevice.setShowLogs(1)

# Connect to two Arduino devices
device1 = UUGearDevice('UUGear-Arduino-6574-3279')
device2 = UUGearDevice('UUGear-Arduino-1783-1825')

# Set up GPIO pins for Ultrasonic Sensor
GPIO.setmode(GPIO.BCM)
TRIG = 23
```

```
ECHO = 24
RELAY1 = 25
RELAY2 = 27
acc1, acc2 = 0, 0
GPIO.setup(TRIG, GPIO.OUT)
GPIO.setup(ECHO, GPIO.IN)
GPIO.setup(RELAY1, GPIO.OUT)
GPIO.setup(RELAY2, GPIO.OUT)
GPIO.output(TRIG, False)

# Initialize the actuator relays
print("Pump1 off")
acc1 = 0
GPIO.output(RELAY1, True)
print("Pump2 off")
acc2 = 0
GPIO.output(RELAY2, True)

while True:
    temperature = 0
    soil_moisture1 = 0
    soil_moisture2 = 0
    ldr, rain_gauge1 = 0
    rain_gauge2 = 0

    # Read water level using Ultrasonic Sensor
    GPIO.output(TRIG, True)
    time.sleep(0.00001)
    GPIO.output(TRIG, False)

    if GPIO.input(ECHO) == 0:
        pulse_start = time.time()
```

```
while GPIO.input(ECHO) == 1:
    pulse_end = time.time()

pulse_duration = pulse_end - pulse_start
distance = round(pulse_duration * 17150, 2)
print("Distance:", distance, "cm")

# Get data from Arduino device 1
if device1.isValid():
    print('Device 1 is ready')
    temperature = device1.analogRead(1) * 0.40020125
    print("Temperature:", temperature)
    time.sleep(0.1)
else:
    print('Device 1 is not ready.')

# Get data from Arduino device 2
if device2.isValid():
    print('Device 2 is ready')

    soil_moisture1 = device2.analogRead(1)
    print("Soil Moisture 1: ", soil_moisture1)
    time.sleep(0.1)

    soil_moisture2 = device2.analogRead(4)
    print("Soil Moisture 2: ", soil_moisture2)
    time.sleep(0.1)

    # Convert readings from Water Level Sensor to millimeters
    rain_gauge1 = device2.analogRead(5)
    if rain_gauge1 <= 480:
```

```
            rain_gauge1 = 0
        # Handle other rain gauge conditions here
        # ...

        rain_gauge2 = device2.analogRead(3)
        if rain_gauge2 <= 480:
            rain_gauge2 = 0
        # Handle other rain gauge conditions here
        # ...

        ldr = device2.analogRead(2)
        print("LDR (Light Intensity): ", ldr)
        time.sleep(0.1)
    else:
        print('Device 2 is not ready')

    # Perform actions based on sensor readings (adjust as needed)
    if (distance > 10 or soil_moisture1 < 700) or rain_gauge1 > 5:
        print("Pump 1 off")
        acc1 = 0
        GPIO.output(RELAY1, True)
    elif distance < 10 and soil_moisture1 > 700 and rain_gauge1 <= 5:
        print("Pump 1 on")
        acc1 = 1
        GPIO.output(RELAY1, False)
    elif distance < 10 and soil_moisture1 > 500 and ldr > 800 and rain_gauge1 <= 5:
        print("Pump 1 on")
        acc1 = 1
        GPIO.output(RELAY1, False)

    if (distance > 10 or soil_moisture2 < 700) or rain_gauge2 > 5:
        print("Pump 2 off")
```

```
        acc2 = 0
        GPIO.output(RELAY2, False)
    elif distance < 10 and soil_moisture2 > 700 and rain_gauge2 <= 5:
        print("Pump 2 on")
        acc2 = 1
        GPIO.output(RELAY2, True)
    elif distance < 10 and soil_moisture2 > 500 and ldr > 800 and rain_gauge2 <= 5:
        print("Pump 2 on")
        acc2 = 1
        GPIO.output(RELAY2, True)


    # Post data to a Django server or cloud service (adjust URLs)
    if 1:
        post_data = {'sID': "cefccad6-ae42-4fca-baed-7a1cae10dce7', 'value':
float(temperature)}
        response = requests.post('https://jeet007.pythonanywhere.com/sensor/adddata/',
data=post_data)
        content = response.content
        print("Temperature Plant 1: ", temperature, content)
    time.sleep(0.05)


    if 2:
        post_data = {'sID': "6db7e9b1-9d45-4b64-adad-985bba42dfe8', 'value':
float(temperature)}
        response = requests.post('https://jeet007.pythonanywhere.com/sensor/adddata/',
data=post_data)
        content = response.content
        print("Temperature Plant 2: ", temperature, content)
    time.sleep(0.05)


# Add more sensor data posting here
# Clean up and close the devices
device1.detach()
```

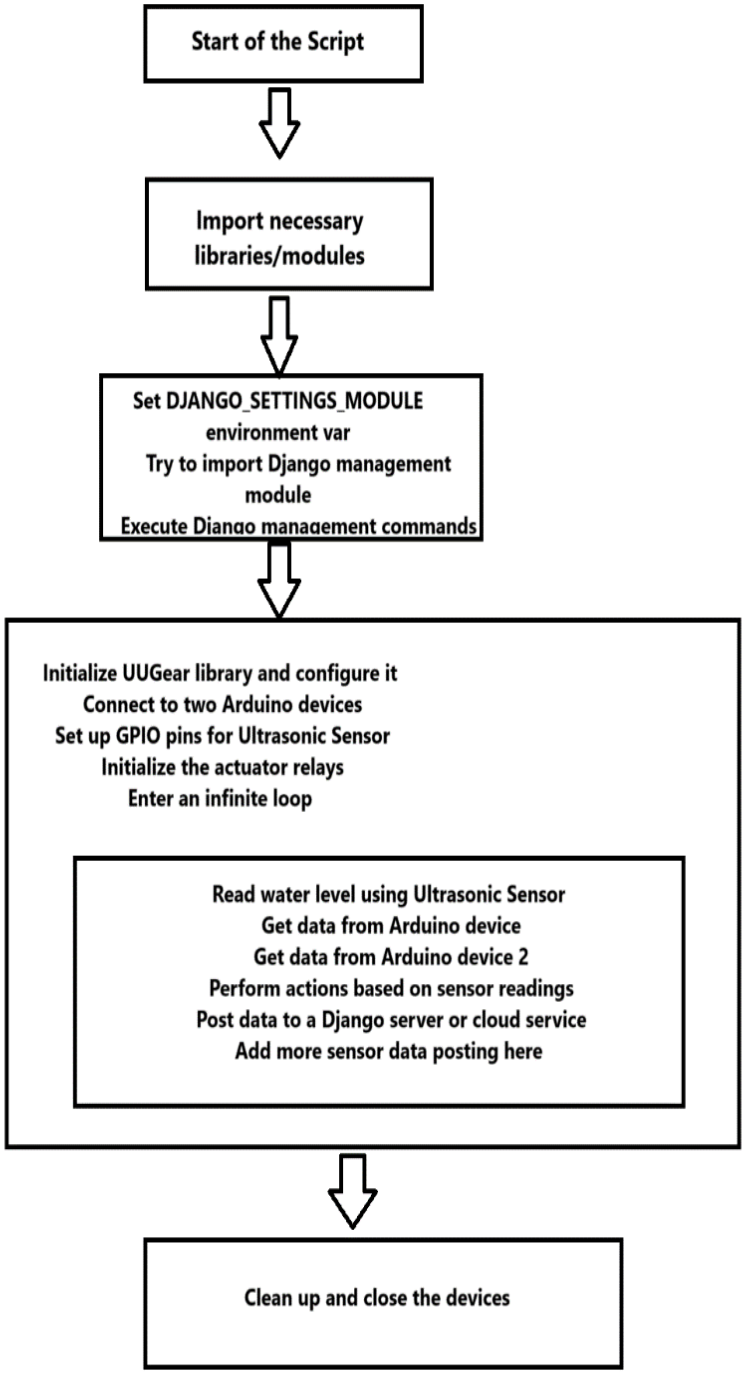
device1.stopDaemon()

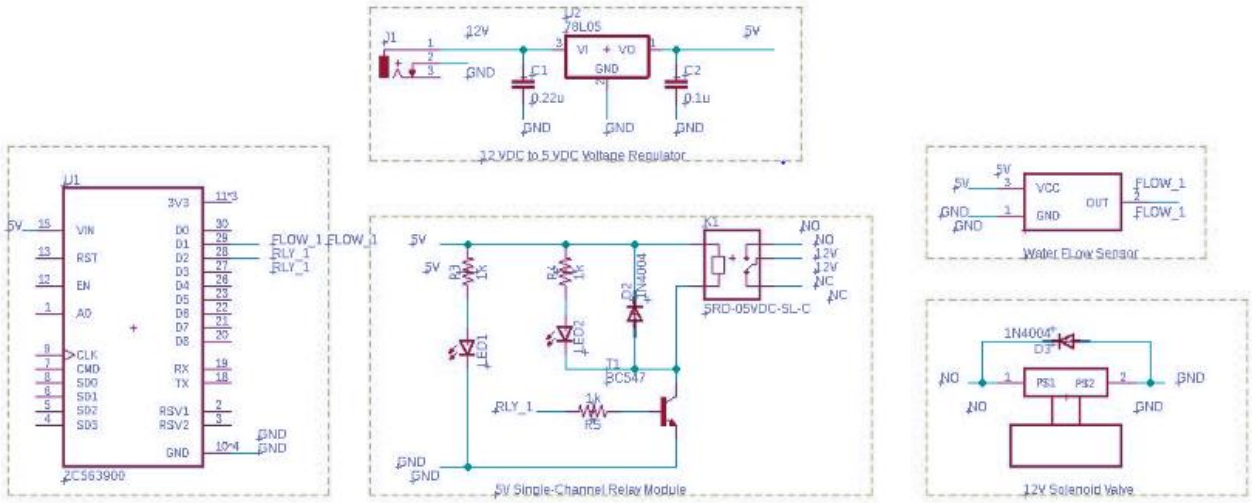
device2.detach()

device2.stopDaemon()

GPIO.cleanup()

FLOWCHART :





Schematic Diagram

CONCLUSION :

In conclusion, smart water management projects leverage technology and data to efficiently manage water resources. These initiatives promote conservation, cost savings, and environmental sustainability. However, success hinges on meticulous planning, expert collaboration, and ongoing maintenance. As water resources become scarcer, these projects are crucial for securing a sustainable water future.