SMART WATER MANAGEMENT

DATE	**26-10-2023**
TEAM ID	**536**
PROJECT NAME	**SMART WATER MANAGEMENT**

PHASE 3: Development Part -1 – SMART WATER MANAGEMENT

<u>PROJECT OVERVIEW:</u>

In this phase, you will see the current water management system in our region suffers from inefficiency, water losses, and a lack of real-time monitoring. The objective is to design, develop, and implement a **"Smart Water Management"** System that addresses these challenges by optimizing water distribution, reducing water wastage, enhancing system resilience, and promoting responsible water use."

REAL TIME SMART WATER MANAGEMENT INFORMATION:

1. Sensor Networks: Sensor networks are deployed throughout a water supply system to monitor various parameters in real-time. These sensors can measure water flow, quality, pressure, and other essential data. They send this information to a central control system.

2. Data Collection: Data from these sensors is collected and processed in real-time. The collected data provides insights into the condition of the water infrastructure and the availability and quality of water.

3. Leak Detection: Real-time data analytics can help detect leaks and other anomalies in the water distribution system. Sudden drops in pressure or flow can indicate a leak or a burst pipe.

4. Demand Forecasting: By analyzing historical data and current usage patterns, water management systems can predict future demand. This helps in planning water distribution and resource allocation.

5. Asset Management: Smart water management systems track the condition of water infrastructure assets such as pipes, pumps, and valves. This data is used to schedule maintenance and replacements, reducing the risk of infrastructure failure.

6. Water Quality Monitoring: Monitoring the quality of water is essential for public health. Real-time sensors can detect changes in water quality, such as contaminants or pH levels, and trigger alerts if water quality falls below acceptable standards.

7. Remote Valve Control: Operators can remotely control valves and pumps to manage water flow and pressure. This capability allows for the immediate response to emergencies and the fine-tuning of water distribution.

8. Customer Engagement: Some systems enable customers to monitor their own water usage in real-time. This can promote water conservation and help customers detect leaks on their property.

9. Data Analytics: Advanced analytics tools are used to process the vast amount of data collected. Machine learning and AI algorithms can provide insights and predictions for system optimization.

10. Water Conservation: Real-time data can be used to encourage water conservation efforts. For example, customers can receive alerts or incentives for reducing usage during peak demand periods.

11.Energy Efficiency: Smart water systems can also optimize energy usage in pumping and treatment processes, reducing operational costs and environmental impact.

12. Emergency Response: In the case of water contamination or other emergencies, real-time data can help authorities respond more rapidly and effectively.

BENEFITS:
Implementing IoT for smart water management offers several benefits:
1. Water Conservation:
- Leak Detection: Early detection and repair of leaks in the water distribution network reduce water loss and waste.

2. **Cost Savings**:

-Reduced Operational Costs: Efficient use of resources, reduced energy consumption, and optimized maintenance schedules lead to cost savings for water utilities.

3. **Improved Service Reliability**:

-Faster Issue Resolution: Real-time monitoring and predictive maintenance help prevent service interruptions, ensuring a more reliable water supply.

4.Enhanced Environmental Sustainability:

-Reduced Energy Consumption: Optimization of water treatment and distribution processes results in lower energy usage and reduced greenhouse gas emissions.

5.Data-Driven Decision-Making:

-Informed Planning: Data analytics and real-time information enable water utilities to make more informed decisions regarding infrastructure investment and expansion.

PROJECT PLANNING:

PROJECT COMPONENTS:

SENSOR SELECTION:

- We have selected a range of sensor, like ultra sonic sensor.
- The ultrasonic sensor is an electronic device used to measure the distance.

DEPLOYMENT OF IOT:

- The deployment of the Internet of Things (IoT) in environmental monitoring in parks is a powerful strategy to enhance the management, conservation, and enjoyment of natural resources and public spaces.

MICROCONTROLLER SELECTION :

- The ESP32 is used in this project. The ESP32 is a single 2.4 GHz Wi-Fi-and-Bluetooth combo chip designed with the TSMC low-power 40 nm technology. It is designed to achieve the best power and RF performance, showing robustness, versatility and reliability in a wide variety of applications and power scenarios.

SENSOR INTEGRATION :

- The ESP32 can interface with various sensors such as flow meters, water quality sensors, pressure sensors, and ultrasonic sensors. These sensors can monitor key parameters like water flow rate, quality, and pressure in real-time.

DATA COLLECTION AND PROCESSING :

- The ESP32 can collect data from connected sensors and process it locally. This processing can include data filtering, calibration, and aggregation. This processed data can be used for decision-making and transmitted to a central server or cloud platform.

DATA TRANSMISSION:

- Data transmission in environmental monitoring in parks is a critical aspect of the monitoring process.
- It involves the transfer of collected environmental data from sensors and monitoring devices to a central data repository or control center where it can be processed, analyzed, and utilized for various purposes, such as decision-making, research, and public information.

REAL TIME MONITORING :

- The ESP32 can continuously monitor data from sensors and send updates in real-time to a central control system. This allows for immediate responses to critical events, such as leak detection or water quality issues.

CONNECTIVITY :

- The built-in Wi-Fi and Bluetooth capabilities of the ESP32 enable it to connect to the internet and other devices. Wi-Fi can be used to connect to a local network, while Bluetooth can be utilized for short-range communication with nearby devices**.**

INTEGRATION WITH CLOUD PLATFORM:

- The ESP32 can communicate with cloud platforms like AWS IoT, Google Cloud IoT, or Microsoft Azure IoT, where data can be securely stored, analyzed, and visualized.

Open-Source Community:

- The ESP32 is supported by a robust open-source community, providing access to a wide range of libraries, tools, and resources for developing custom applications.

PYTHON SCRIPT DEVELOPMENT :

- Develop a Python script that will run on the IoT for smart water management
- This script should be responsible for the following tasks:
- Initialize the GPIO pin for the water sensor (analog input) and the buzzer (digital output).
- Set a threshold value to determine when the alarm (buzzer) should be activated.
- Create a loop that continuously monitors the water level:

a. Read the water sensor value using the analog input pin.

b. Check if the water level (sensor value) is greater than the threshold:

- If it is, set the buzzer pin to HIGH to activate the alarm.

- Print a message indicating that water has been detected.

c. If the water level is below the threshold:

- Set the buzzer pin to LOW to turn off the alarm.

- Print a message indicating that no water has been detected.

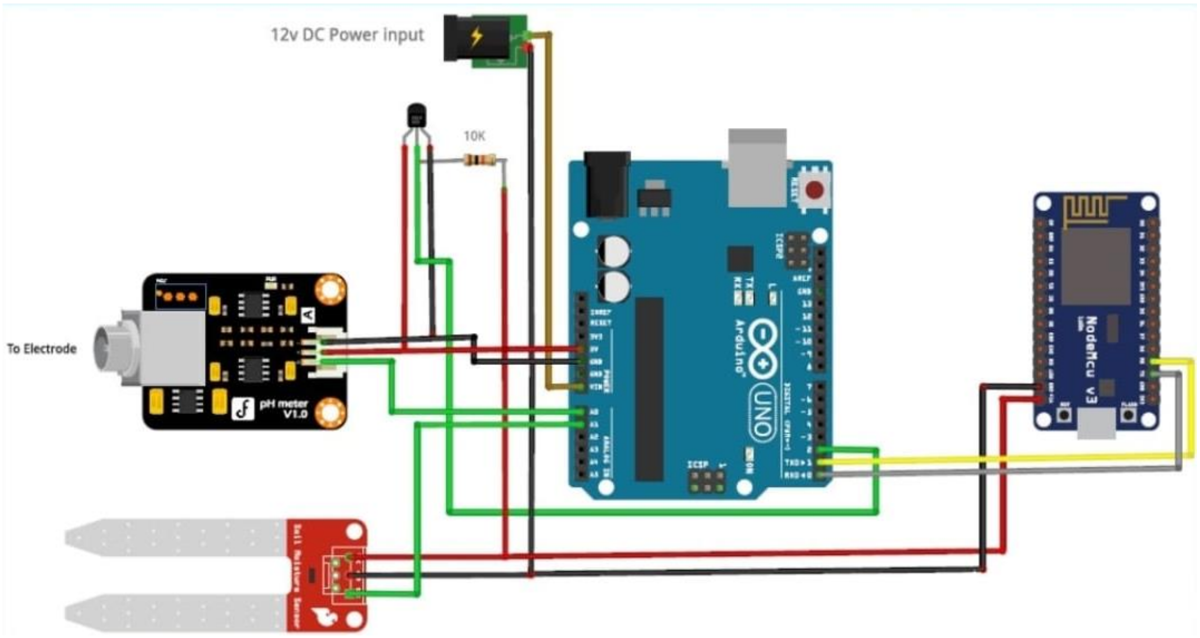
- Add a delay to control how often the water level is checked, typically using the `time.sleep` function.
- Repeat steps 3 and 4 indefinitely, so the system continuously monitors the water sensor.
- End the program.

<u>PYTHON PROGRAM:</u>

```
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)
```

SIMULATION:



12v DC Power input
10K
To Electrode
pH meter V1.0
Arduino
UNO
NodeMcu v3
Soil Moisture Sensor