# SMART WATER MANAGEMENT

PHASE 5


Team members:
Vijayavarman K
Vimalarasan T
Karthick P
Saran S
Vigneshwaran KAD
Vimal R

## PROJECT DESCRIPTION

Water plays a big role in supporting our communities. Without water there would be no local business or industry. Fire-fighting, municipal parks, and public swimming pools all need lots of water. An array of pipes, canals, and pumping stations managed by our public water systems are needed to bring a reliable supply of water to our taps each day

Implementing an IoT solution that can help us manage the water we use is critical to improving our quality of living by reducing the amount of water we consume. This includes: ·

Monitoring and controlling your tap's flow to ensure it is running at optimal levels. This will also allow you to monitor the flow of water from your home or office. This will also help you determine if there is any water shortage in your area.


*Block diagram*

## OBJECTIVE:

The objectives of smart water management encompass a range of goals and strategies aimed at optimizing the use, conservation, and quality of water resources in the parks and garden. Many people uses tap waters, bathrooms and pipes. It also helps in developing society while enhancing overall efficiency and sustainability.

## PROJECT CONSTRUCTION:

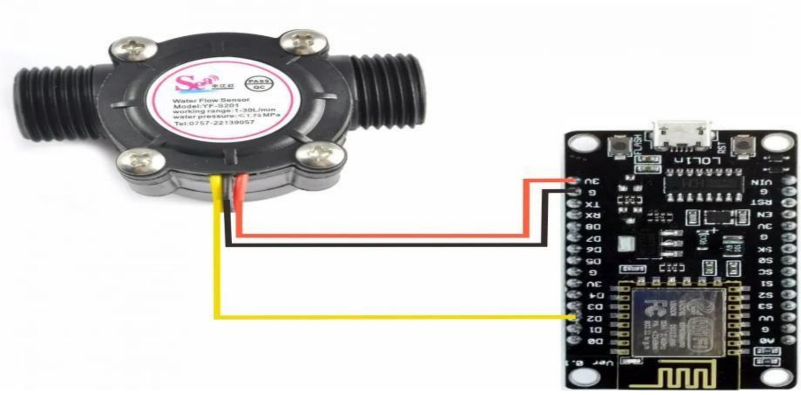
**ESP32 development board**

It is a versatile microcontroller with built in Wi-Fi and Bluetooth capabilities, making it an ideal tool for developing new applications. This system provides a high level of security and privacy protection, as well as a low cost solution to the installation of sensors and monitoring devices.

*ESP32 processor*

**Sensors**

The YF-S201 is the water flow sensor works on Hall Effect principle that can be used to monitor the flow of water help you determine if your water supply is running at optimal levels. It is also a great way to track water usage, which has working range of 1-30L/min. The size is 1/2 BSP which is enough for water usage in parks and garden. We can also use different based on the size of the water supply. This will save us time and money by saving energy.



*YS-S201 sensor with ESP32 processor*

**Power supply**

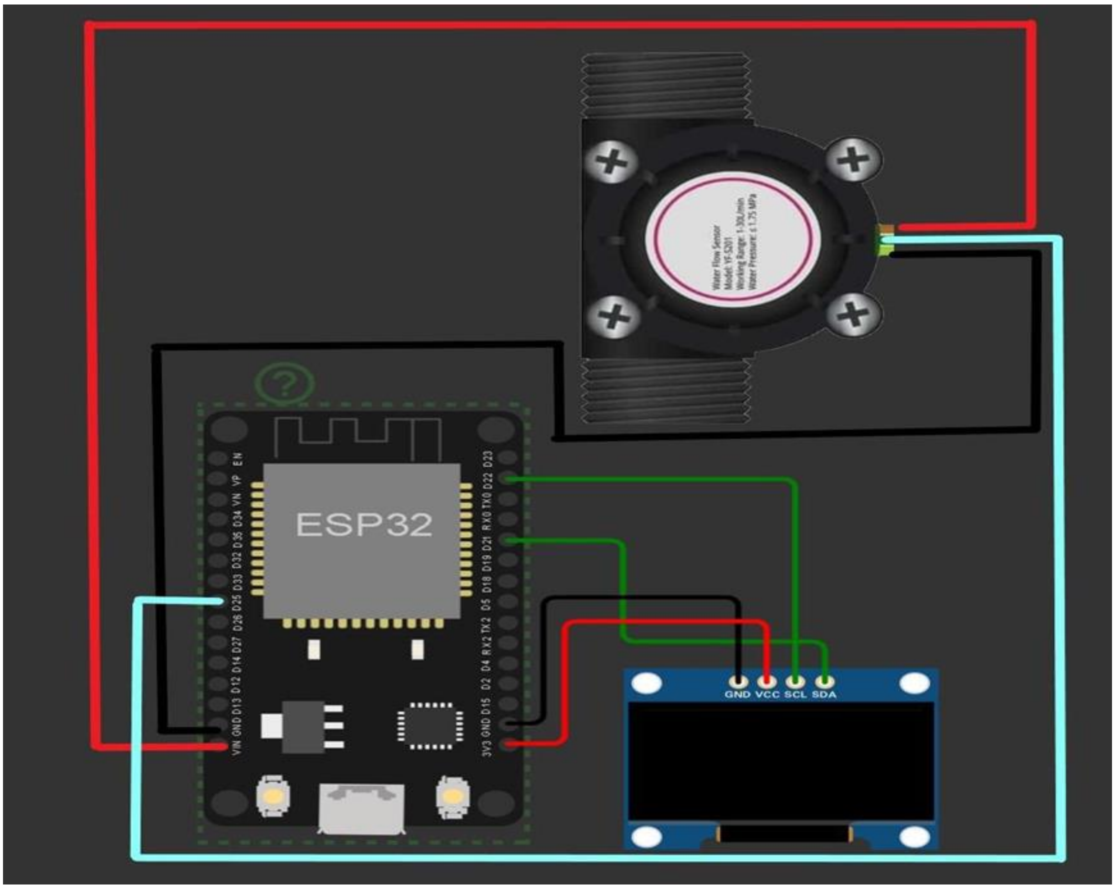
The stable power supply should ensured for ESP32 and sensors. Battery powered solution are possible, but they needed to replaced periodically. Considering power efficiency, we can use ac supply from the power lines or we can use power from solar panel placed in lights in the parks and garden.

**0.96 OLED Display**

0.96 inch OLED displays are small, lightweight, and consume very little power. They are ideal for portable devices that require a low-power display. OLED displays have a high contrast ratio and wide viewing angles, making them easy to read in different lighting conditions. They also have fast response times and can display high-quality images and videos.

## DESIGNING:

Now let us interface YF-S201 Hall-Effect Water Flow Sensor with Nodemcu ESP32 & OLED Display. The OLED Display will show Water Flow Rate & Total Volume of Water passed through the pipe. The same Flow Rate & Volume data can be sent to Thingspeak Server after an interval of 15 seconds regularly. You can switch to Blynk Application if you want immediate data. Similarly using MQTT Protocol better wireless communication can be achieved. But now let us see the IoT Water Flow Meter Circuit Diagram & Connection.




- Connect the VCC pin of the YF-S201 flow sensor to the 5V pin of the ESP32.
- Connect the GND pin of the YF-S201 flow sensor to the GND pin of the ESP32.
- Connect the SIGNAL pin of the YF-S201 flow sensor to any of the available GPIO pins on the ESP32. For this example, we will use GPIO 25.
- Connect the VCC pin of the OLED display to the 3.3V pin of the ESP32.
- Connect the GND pin of the OLED display to the GND pin of the ESP32.
- Connect the SCL pin of the OLED display to GPIO 22 on the ESP32.
- Connect the SDA pin of the OLED display to GPIO 21 on the ESP32.

## **CODE FOR ESP32:**

Creating the complete code for an ESP32 with a YF-S201 water flow detector and a web server to serve the web page would be a comprehensive task. Below is a simplified code using the Arduino IDE and the ESP32 core for Arduino.The required libraries should be installed in the Arduino IDE for the ESP32.

```
#include <WiFi.h>

#include <WiFiClient.h>

#include <WebServer.h>

#include <Wire.h>

#include <Adafruit_GFX.h>

#include <Adafruit_SSD1306.h>


// Replace with your network credentials const
char* ssid = "YourWiFiSSID"; const char* password
= "YourWiFiPassword";


WebServer server(80);


// Initialize the OLED display

#define SCREEN_WIDTH 128

#define SCREEN_HEIGHT 64

#define OLED_RESET    -1

Adafruit_SSD1306 display(SCREEN_WIDTH, SCREEN_HEIGHT, &Wire,
OLED_RESET);


volatile unsigned int pulse;
unsigned int flowRate; unsigned
int flowMilliLitres; unsigned long
totalMilliLitres; unsigned long
oldTime;
```

```
// Water flow sensor input const
int sensorInterrupt = 0; const int
sensorPin = 2;

void ICACHE_RAM_ATTR pulseCounter() {   pulse++;
}

void setup() {
  // Connect to Wi-Fi   WiFi.begin(ssid,
password);   while (WiFi.status() !=
WL_CONNECTED) {     delay(1000);
    Serial.println("Connecting to WiFi...");
   }
  Serial.println("Connected to WiFi");

  // OLED display setup
  if(!display.begin(SSD1306_I2C_ADDRESS, 4, 15)) {
Serial.println(F("SSD1306 allocation failed"));     for(;;);
  }

  display.display();   delay(2000);
display.clearDisplay();
display.setTextSize(1);
  display.setTextColor(SSD1306_WHITE);

  // Initialize the water flow sensor   pinMode(sensorPin,
INPUT);   digitalWrite(sensorPin, HIGH);

  pulse = 0;
flowRate = 0;
flowMilliLitres = 0;
totalMilliLitres = 0;
oldTime = 0;
```

```
  // Attach an interrupt to the sensor input
  attachInterrupt(digitalPinToInterrupt(sensorPin), pulseCounter, FALLING); // Define the route
to serve the HTML page   server.on("/", HTTP_GET, handleRoot);

  // Start the server   server.begin();
}

void loop() {   server.handleClient();

  unsigned long currentTime = millis();

  if (currentTime - oldTime > 1000) {     detachInterrupt(sensorInterrupt);
    flowRate = (pulse / 7.5);  // 7.5 is the number of pulses per liter
flowMilliLitres = (flowRate / 60) * 1000;     totalMilliLitres += flowMilliLitres;
oldTime = currentTime;     pulse = 0;
    attachInterrupt(digitalPinToInterrupt(sensorPin), pulseCounter, FALLING);
  }
}

void handleRoot() {
  // Create the HTML page to display the flow rate   String
html = "<html><body>";   html += "<h1>Water Flow
Detector</h1>";
  html += "<p>Flow Rate: " + String(flowRate) + " L/min</p>";
   html += "</body></html>";

  server.send(200, "text/html", html);
}
```

**HTML CODE:**

```
<!DOCTYPE html>
<html>
<head>
    <title>IoT Smart Water Management</title>
```

```html
    <link rel="stylesheet" href="style.css">
</head>
<body>
  <h1 style="color:rgb(12, 185, 216); ">Smart Water Management</h1>
  <p><b>Rate: <span id="flowRate">0.00</span> L/min</b></p>
  <p><b>Volume: <span id="totalVolume">0.00</span> L</b></p>

  <canvas id="chart" width="400" height="200"></canvas>

  <script src="script.js">

  </script>

</body>
</html>
```

## CSS CODE:

```css
body{
background-image:url(https://wallpapercrafter.com/sizes/2560x1440/291957faucet-fountain-water-dispenser-water-running.jpg);     font-family: Arial, sans-serif;     background-color: #f0f0f0;
margin: 0;     padding: 0;     text-align: center;
}

h1 {
color: #333;
}

p {
  font-size: 21px;
}
```

```
canvas {
   background-color: #fff;
margin: 20px auto;     display:
block;
}
```

## JAVA SCRIPT:

```
const flowRateDisplay = document.getElementById("flowRate");
const totalVolumeDisplay = document.getElementById("totalVolume"); const chart
= document.getElementById("chart").getContext("2d");

let flowData = [];

function updateData(flowRate, totalVolume) { flowRateDisplay.textContent =
flowRate.toFixed(2); totalVolumeDisplay.textContent = totalVolume.toFixed(2);

flowData.push(flowRate); if
(flowData.length > 10) {
flowData.shift();
}

chart.clearRect(0, 0, 400, 200);

chart.beginPath(); chart.moveTo(0, 200);
flowData.forEach((flow, index) => {
chart.lineTo(index * 40, 200 - flow * 10);
});
chart.stroke();
}

// Function to fetch data from ESP32 server function
fetchData() {
fetch('/data') // Replace with the URL of your ESP32 server
```

```
.then(response => response.json())
.then(data => {
updateData(data.flowRate, data.totalVolume);
})
.catch(error => {
console.error("Error fetching data: " + error);
});
}

// Periodically fetch data from the server setInterval(fetchData, 1000);
```

---

In this setup, the ESP32 measures the flow rate and total volume of water used. It also displays this information on the OLED screen.

This code sets up a web server on your ESP32, displays the flow rate on a connected OLED display, and serves a simple HTML page displaying the flow rate.

Make sure to replace "YourWiFiSSID" and "YourWiFiPassword" with your actual Wi-Fi credentials.

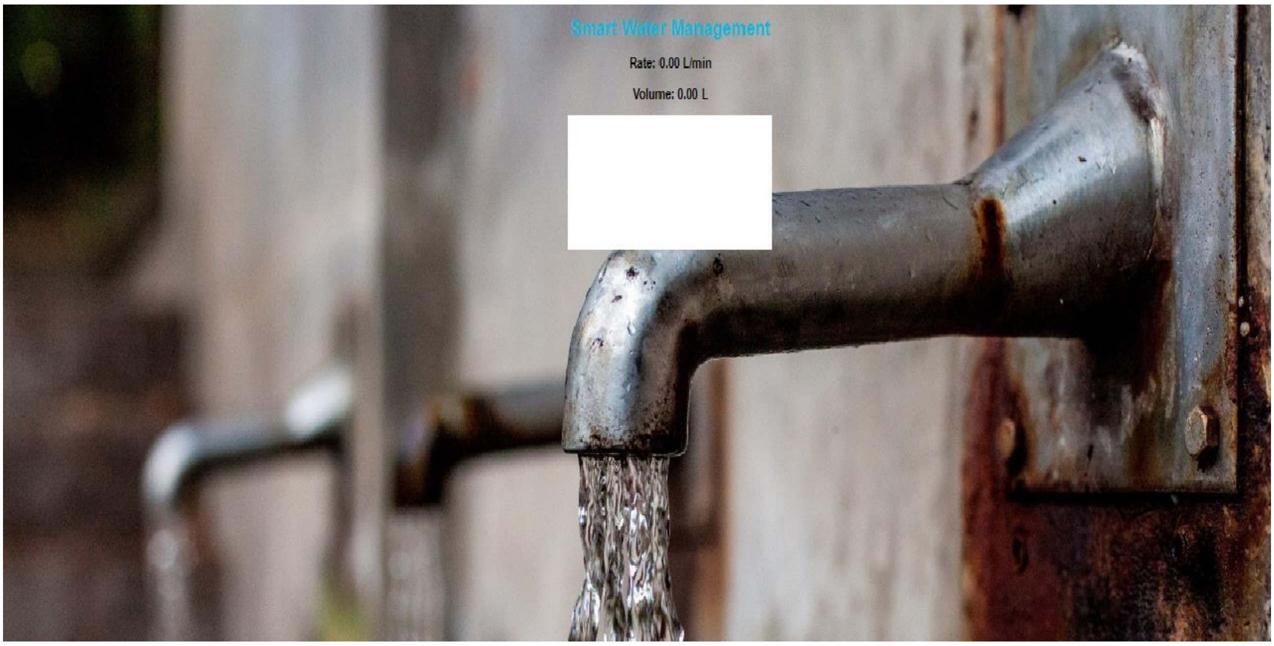
The YF-S201 sensor generates pulses based on the flow rate, and the code counts these pulses to calculate the flow rate.

We have a basic HTML structure with a title, header, and a paragraph to display the flow rate.

JavaScript is used to fetch data from the ESP32's API endpoint, which should be set up on the ESP32 to provide the current flow rate.

We update the flow rate on the web page every 1 seconds using setInterval.

**Here is the platform to receive and display water usage data sent by iot device:**




The above image shows the demo of the smart water management. The white space will display the chart after the flow meter gets the readings. The water flow rate and volume rate will be shown and refresh for every 1 second.

## FUNCTIONS:

To clarify the functions of an IoT project focused on optimizing water consumption in public spaces like parks and gardens, here's a breakdown of the key functions and features:

### 1. Data Collection:

The IoT devices, equipped with sensors, collect real-time data on various parameters, including soil moisture, weather conditions, and water flow. These sensors are strategically placed in parks and gardens.

### 2. Data Processing:

The collected data is processed locally on the IoT devices to make sense of the information. Algorithms may be used to interpret sensor readings and assess the need for irrigation.

3. **Water Usage Analysis:**

The system analyzes the data to determine optimal watering schedules based on soil moisture levels, weather forecasts, and other factors. It calculates the precise amount of water required for irrigation.

4. **Real-time Monitoring:**

The IoT devices transmit data in real-time to a central server or cloud platform, allowing for remote monitoring and control. This data is accessible via a user interface.

5. **Remote Control and Alerts:**

The system enables remote control of irrigation systems, allowing users to start or stop watering as needed. It can also send alerts or notifications for critical events, such as leaks or extreme weather conditions.

6. **Water Conservation:**

The primary function is to promote water conservation by ensuring that parks and gardens receive the right amount of water at the right time. This prevents over-irrigation, which can lead to water wastage.

7. **Cost Efficiency:**

The system aims to reduce operational costs by optimizing water usage. By using data-driven decisions, it can lead to significant cost savings in terms of water bills and maintenance expenses.

8. **Environmental Responsibility:**

The project functions with the goal of reducing the environmental impact of water consumption. It helps conserve local ecosystems and prevents overextraction of water resources.

9. **User Interface:**

Users, such as park managers, can access a user-friendly interface, which provides real-time information on water consumption, irrigation schedules, and sensor data. This interface may be web-based or available through a mobile app.

**10. Security and Authentication:**

Implement security measures to ensure that only authorized personnel can access and control the IoT devices and the data.

**11. Scalability:**

The system is designed to scale as more parks and gardens are added to the network, enabling centralized management of multiple locations.

**12. Integration with IoT Platforms:**

The project can integrate with IoT platforms or cloud services to enhance data storage, analytics, and reporting capabilities.

**13. Data Logging and Reporting:**

The system logs historical data for future analysis and generates reports on water consumption, savings, and environmental impact.

**14. Firmware Updates:**

The IoT devices should support firmware updates to ensure they remain up to date with the latest features, improvements, and security patches.

## CLOUD PLATFORM GOVERNANCE AND COMPLIANCE PROTOCOLS:

1. **Choose a Cloud Provider:** Sign up for a free tier cloud service like AWS, free resources for new users.

2. **Set Up an IoT Device:** You'll need an IoT device or simulator to send data to your cloud platform. Raspberry Pi or Arduino are popular choices. You can also simulate IoT devices using software.

3. **Data Ingestion:** Create an IoT Hub or similar service on your chosen cloud platform to ingest data from your devices. This service typically has a free tier with limited usage.

4. **Data Storage:** Store the incoming data in a database or storage service. Most cloud providers offer free or low-cost database services for small projects.

5. **Data Processing:** You can use serverless functions or containers to process the data. AWS Lambda or Google Cloud Functions are examples of serverless options with a free tier.

6. **Data Visualization:** Use free tools like Grafana, Kibana, or cloud-based dashboards to visualize your IoT data.

7. **Set Up Alerts:** Configure alerts and notifications for specific conditions using cloud monitoring services. Some free tier options may be available.

8. **Security:** Ensure the security of your IoT data and devices. Most cloud providers offer basic security features in their free tiers.

9. **Scaling:** As your project grows, you might need to move to a paid tier or optimize your setup for cost-efficiency.

10. **Documentation and Learning:** Learn and document as you go. Most cloud providers offer extensive documentation and tutorials.