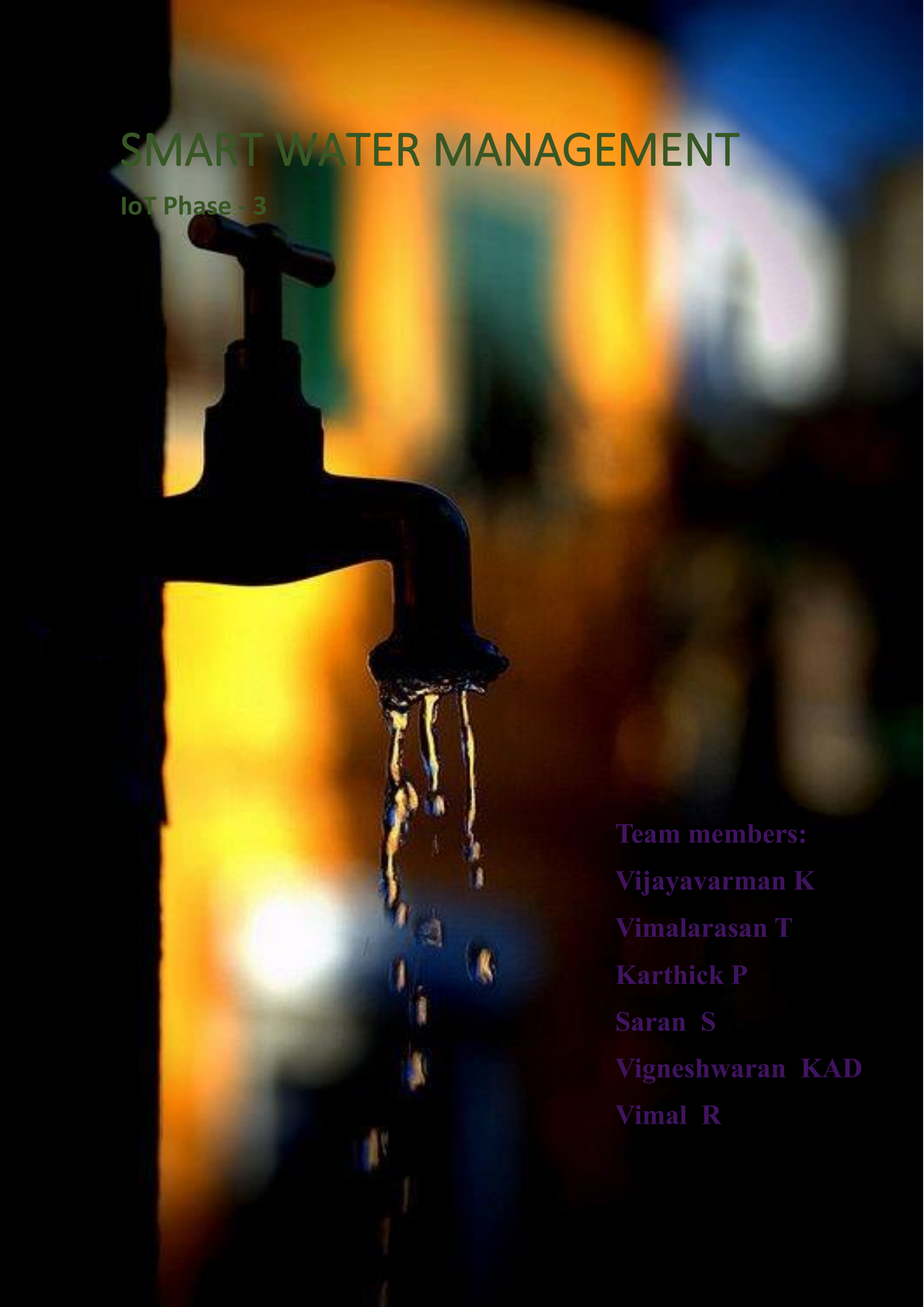


SMART WATER MANAGEMENT

IoT Phase - 3

Team members:

Vijayavarman K

Vimalarasan T

Karthick P

Saran S

Vigneshwaran KAD

Vimal R

PROJECT CONSTRUCTION

INTRODUCTION:

In this phase, we are going to build the project and write a code to get a prototype model for smart water management. Smart water management is an innovative IoT-based solution designed to optimize water consumption in public parks and gardens using ESP32-powered devices.

The primary goal of this project is to achieve a significant reduction in water consumption in public parks and gardens while simultaneously maintaining or enhancing the quality, aesthetics, and functionality of these spaces.

PROJECT CONSTRUCTION:

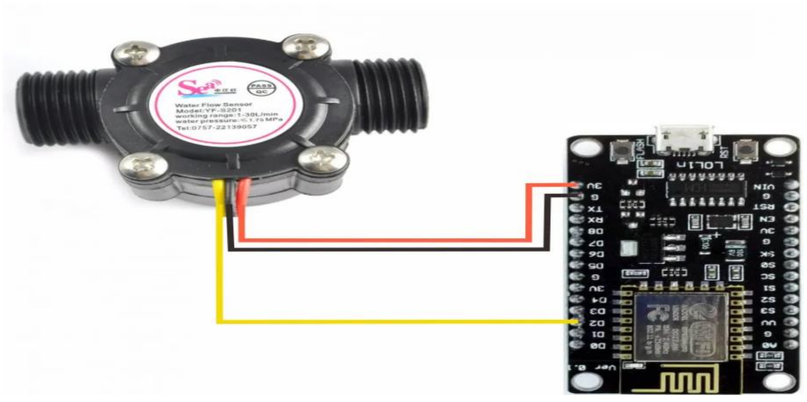
ESP32 development board

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.

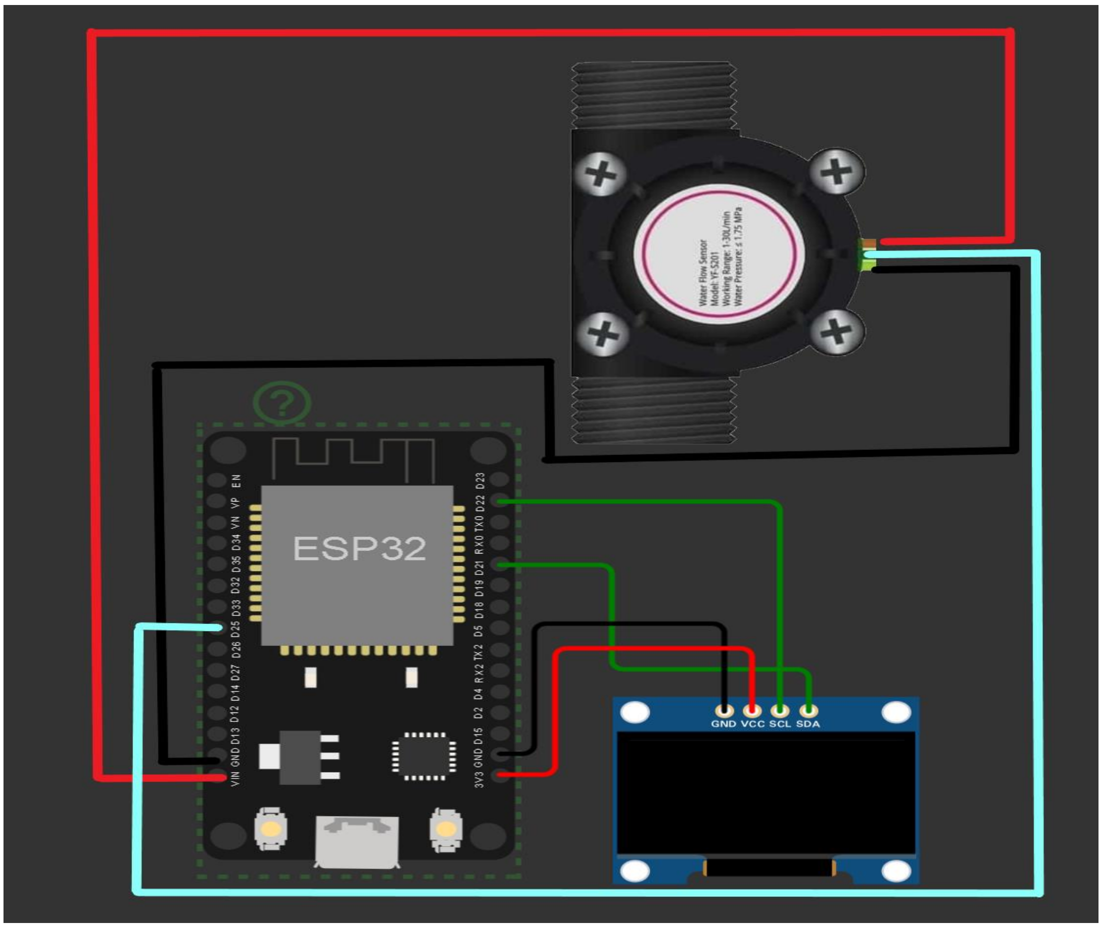
0.96 OLED Display

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.


ESP32
GND VCC SCL SDA
VIN GND D13 D12 D14 D27 D26 D25 D33 D32 D35 D34 VN VP EN
3V3 GND D15 D2 D4 RX2 TX2 D5 D18 D19 D21 RX0 TX0 D22 D23
Water Flow Sensor
Model: YF-S201
Working Range: 1-30L/min
Water Pressure: ≤ 1.75 MPa

- 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.

CODING WITH PYTHON SCRIPT:

```
from machine import Pin, I2C
import network
import urequests
import ssd1306
import time

# OLED display configuration
SCREEN_WIDTH = 128
SCREEN_HEIGHT = 64
OLED_RESET = None  # You can specify a reset pin if needed
i2c = I2C(0, scl=Pin(22), sda=Pin(21))
oled = ssd1306.SSD1306_I2C(SCREEN_WIDTH, SCREEN_HEIGHT, i2c, OLED_RESET)

# WiFi configuration
SSID = "YourWiFiSSID"
PASSWORD = "YourWiFiPassword"
THINGSPEAK_API_KEY = "YourThingSpeakAPIKey"
THINGSPEAK_URL = "api.thingspeak.com"

# Flow sensor configuration
SENSOR_PIN = Pin(2, Pin.IN, Pin.PULL_UP)
FLOW_SENSOR_CALIBRATION = 4.5

pulseCount = 0
flowRate = 0.0
flowMilliLitres = 0
```

```
totalMilliLitres = 0
previousMillis = 0
interval = 1000
ledState = 0

def pulseCounter(p):
    global pulseCount
    pulseCount += 1

def connect_wifi(ssid, password):
    sta_if = network.WLAN(network.STA_IF)
    if not sta_if.isconnected():
        print("Connecting to WiFi...")
        sta_if.active(True)
        sta_if.connect(ssid, password)
        while not sta_if.isconnected():
            pass
    print("Connected to WiFi:", ssid)

def send_to_thingspeak(api_key, field1, field2):
    url = "http://{}:{}/update?api_key={}&field1={:.2f}&field2={:.2f}".format(THINGSPEAK_URL, 80, api_key, field1, field2)
    try:
        response = urequests.get(url)
        response.close()
    except Exception as e:
        print("Failed to send data to ThingSpeak:", e)
```

```
connect_wifi(SSID, PASSWORD)

SENSOR_PIN.irq(trigger=Pin.IRQ_FALLING, handler=pulseCounter)

while True:
    currentMillis = time.ticks_ms()
    if time.ticks_diff(currentMillis, previousMillis) > interval:
        pulse1Sec = pulseCount
        pulseCount = 0

        flowRate = (1000.0 / (time.ticks_diff(currentMillis, previousMillis))) * pulse1Sec / FLOW_SENSOR_CALIBRATION
        previousMillis = currentMillis

        flowMilliLitres = (flowRate / 60) * 1000
        totalMilliLitres += flowMilliLitres

        print("Flow rate: {:.2f} L/min".format(flowRate))
        print("Output Liquid Quantity: {:.2f} mL / {:.2f} L".format(totalMilliLitres, totalMilliLitres / 1000))

        oled.fill(0)
        oled.text("Flow Rate:", 0, 0)
        oled.text("{:.2f} L/min".format(flowRate), 0, 20)
        oled.text("V:{:.2f} L".format(totalMilliLitres / 1000), 0, 40)
        oled.show()
    send_to_thingspeak(THINGSPEAK_API_KEY, flowRate, totalMilliLitres / 1000)
```

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 over-extraction 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.

In summary, the functions of this IoT project revolve around data collection, analysis, and real-time monitoring to optimize water consumption in public spaces, with an emphasis on sustainability, cost efficiency, and environmental responsibility. The project aims to reduce water wastage, save costs, and contribute to the conservation of water resources.