Title: Real-time Water Consumption Monitoring System for Smart Water Management

Objectives:

Develop a system to monitor water consumption in real time.
Promote water conservation and sustainable practices through data-driven insights.
IoT Sensor Setup:
Ultrasonic Distance Sensor:

Purpose: Measure water level in the tank.
Connection: Connect the TRIG and ECHO pins of the sensor to the GPIO pins on the Raspberry Pi.
OLED Display:

Purpose: Display water level, connectivity status, mode, and pump status.
Connection: Connect via I2C protocol to the Raspberry Pi.
Relay:

Purpose: Control the water pump based on water level and mode.
Connection: Connect to a GPIO pin on the Raspberry Pi.
Buttons:

Purpose: Control mode, pump, and reset buzzer.
Connection: Connect to GPIO pins on the Raspberry Pi.
Buzzer:

Purpose: Provide audio alerts.
Connection: Connect to a GPIO pin on the Raspberry Pi.
Mobile App Development:
Blynk Integration:
Objective: Enable remote monitoring and control of the system.
Code Implementation: Utilize Blynk libraries for communication between the Raspberry Pi and the mobile app.
Raspberry Pi Integration:
Software Libraries:

Use RPi.GPIO for GPIO control.
Use Adafruit SSD1306 for OLED display.

Code Implementation:

Write Python code to handle sensor data, control relays, update the OLED display, and interact with buttons.
Diagrams and Schematics:
System Architecture Diagram:

Include a high-level overview of the entire system, showing how the Raspberry Pi, sensors, and mobile app interact.
Circuit Diagram:

Provide a detailed wiring diagram showing how each component is connected to the Raspberry Pi.
Flowchart of Code Logic:

Visualise the flow of the Python code, depicting how sensor data is processed and actions are taken.
Screenshots:

Include screenshots of the mobile app interface, showcasing how users can monitor water levels and control the system remotely.
Promoting Water Conservation:
Data-Driven Insights:

Explain how real-time monitoring allows users to track their water usage patterns and make informed decisions.
Alerts and Automation:

Describe how the system can send alerts when water levels are low and automate the pump to optimize usage.

Raspberry pi code:

```
import time
import Adafruit_GPIO.SPI as SPI
import Adafruit_SSD1306
import RPi.GPIO as GPIO

# Define GPIO pins
TRIG_PIN = 23
ECHO_PIN = 24
```

```
WIFI_LED_PIN = 2
BUZZER_PIN = 13
RELAY_PIN = 14

# Initialize OLED display
disp = Adafruit_SSD1306.SSD1306_128_32(rst=None, i2c_address=0x3D)
disp.begin()

# Initialize GPIO
GPIO.setmode(GPIO.BCM)
GPIO.setup(TRIG_PIN, GPIO.OUT)
GPIO.setup(ECHO_PIN, GPIO.IN)
GPIO.setup(WIFI_LED_PIN, GPIO.OUT)
GPIO.setup(BUZZER_PIN, GPIO.OUT)
GPIO.setup(RELAY_PIN, GPIO.OUT)

# Function to measure distance using ultrasonic sensor
def measure_distance():
    # Implement the logic to measure distance using TRIG_PIN and ECHO_PIN
    pass

# Function to control buzzer
def control_buzzer(duration):
    GPIO.output(BUZZER_PIN, GPIO.HIGH)
    time.sleep(duration)
    GPIO.output(BUZZER_PIN, GPIO.LOW)

# Function to update OLED display
def display_data():
    # Implement logic to display data on the OLED
    pass

# Main loop
try:
    while True:
        # Measure distance
        distance = measure_distance()

        # Implement the rest of the logic similar to the original code
        # Update OLED display
        display_data()

        time.sleep(1)  # Adjust delay as needed

except KeyboardInterrupt:
    GPIO.cleanup()
```

Team members name:
A.Athira – 962821104020
A.P.Bala Bharathy – 962821104023
K.Faish Marshooka – 962821104030
G.T.Harini - 962821104033