Phase 5 Submission

SMART WATER MANAGEMENT

Project Objectives:

Water Conservation: The primary goal is to promote responsible water usage by providing real-time insights about the water consumption.

Sustainability: Raise awareness about water conservation and encourage sustainable practices.

Data Collection: Collect real-time water usage data from IoT sensors.

Data Alerting: Send notifications to users when water usage exceeds defined thresholds.

Visualization: Display water consumption data to users through a mobile app.

Raspberry Pi Integration: Use a Raspberry Pi as a data hub and control center for the IoT sensors.

Code Implementation: Develop code to handle data transmission, processing, and communication with the mobile app.

IoT Sensor Setup:

IoT sensors (e.g., flow meters, water pressure sensors) will be installed at various points in a water supply system.

These sensors will collect data on water flow rates, pressure, and usage.

Data from the sensors will be transmitted wirelessly to a central hub, which is a Raspberry Pi.

Mobile App Development:

The mobile app will be developed for iOS and Android platforms.

Users can download and install the app on their smartphones.

The app will provide a user-friendly interface for real-time water consumption monitoring.

Users can view their water usage history, set consumption thresholds, and receive alerts.

The app will support account creation and secure login.

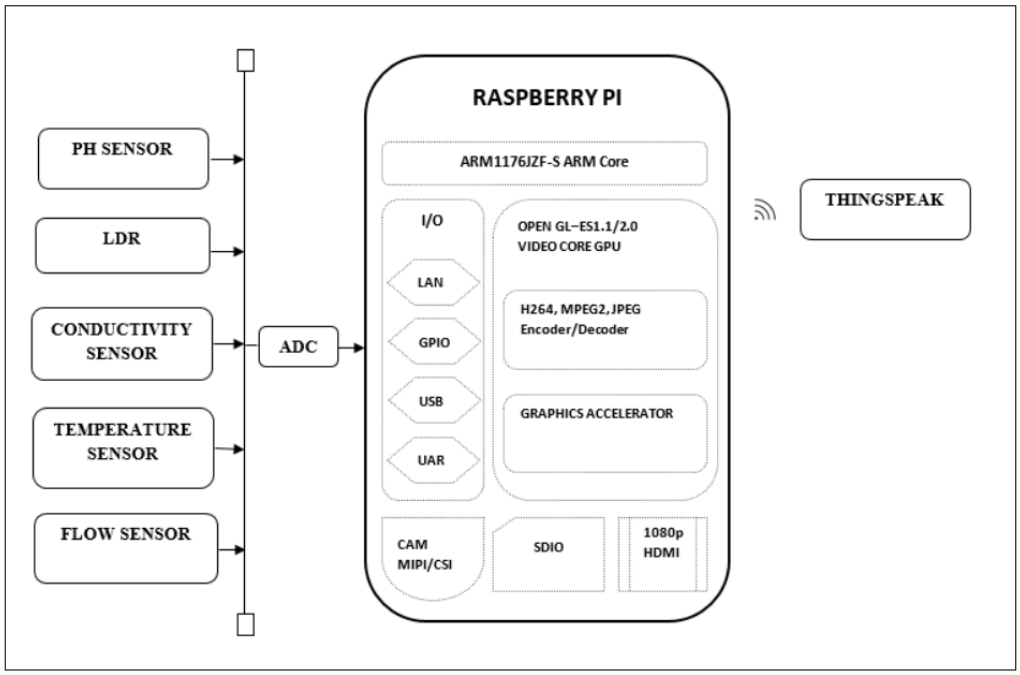
Raspberry Pi Integration:

The Raspberry Pi will act as a central data hub, receiving data from the IoT sensors.

It will store and process incoming data, perform real-time analysis, and generate alerts if water usage exceeds defined limits.

The Raspberry Pi will host a web-based interface for configuration and system status monitoring.

It will communicate with the mobile app through secure API endpoints.



Code Implementation:

Code implementation involves programming the IoT sensors, Raspberry Pi, and the mobile app. Here's a breakdown.

Sensor Data Collection: Write code on the Raspberry Pi to collect data from the IoT sensors and store it in a local database. You can use Python libraries provided by the sensor manufacturers.

Database Integration: Set up a local database (e.g., SQLite or MySQL) to store the sensor data and retrieve it when needed.

Mobile App Development: Create the mobile app using a development framework such as React Native or Flutter. Implement features for user authentication, dashboard, alerts, remote control, and historical data visualization.

Integration with Raspberry Pi: Develop APIs on the Raspberry Pi to allow the mobile app to communicate with it. Use libraries like Flask for Python to create RESTful APIs.

User Interface (UI): Design a user-friendly interface for the mobile app with screens for the dashboard, settings, and user profiles.

Security: Implement secure communication protocols and authentication mechanisms to protect user data and system integrity.

Promoting Water Conservation and Sustainable Practices:

This real-time water consumption monitoring system promotes water conservation and sustainability through several mechanisms:

Immediate Awareness: Users are immediately informed about their water consumption and alerted to leaks, encouraging prompt action.

Behavioral Change: Real-time data encourages users to adopt water-saving habits as they can see the impact of their actions.

Community Engagement: Comparing water usage with others in the community fosters competition and collective efforts in water conservation.

Analytics and Goal Setting: Users can set water-saving goals and track their progress, motivating them to reduce water consumption.

Leak Detection: Early detection of leaks prevents water wastage and potential property damage.

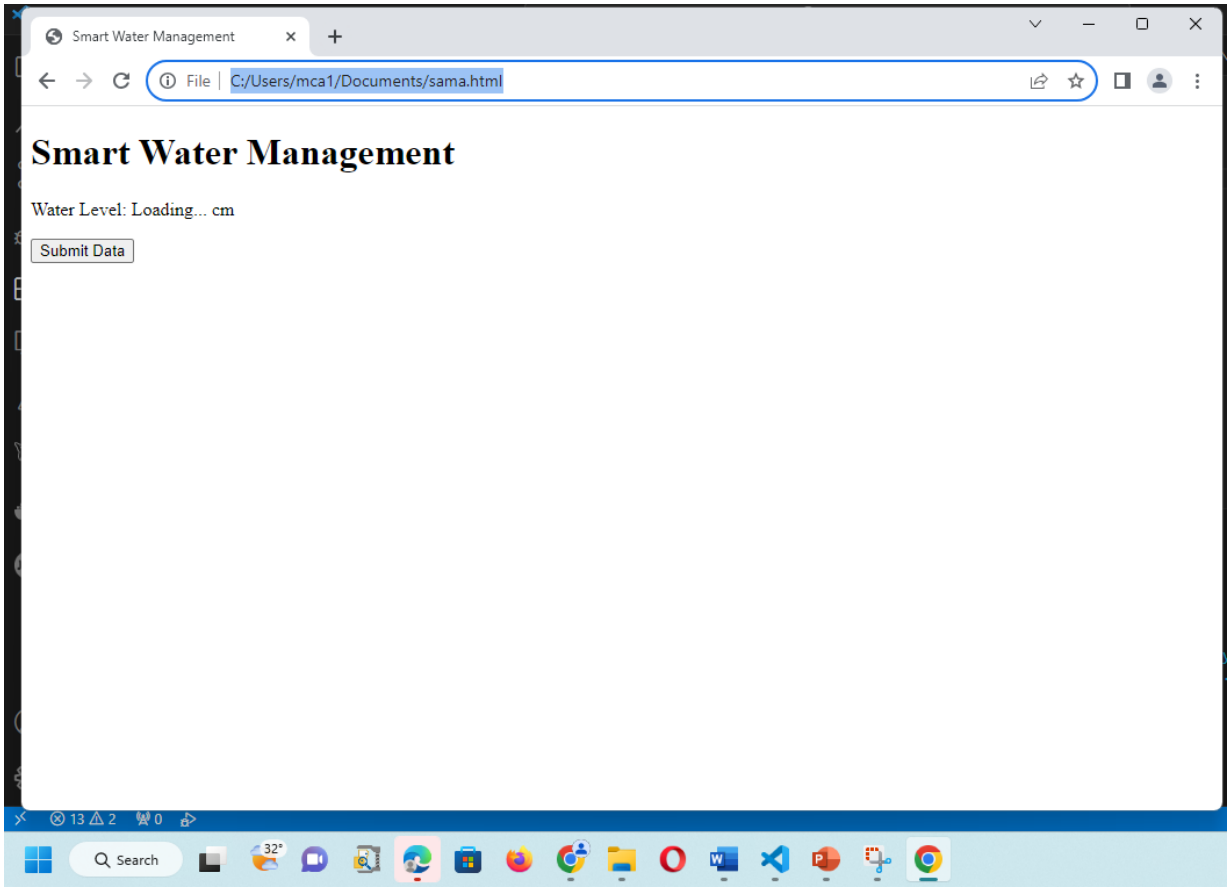
Efficient Resource Management: Water utilities can optimize water distribution and resource allocation based on real-time data.

Platform UI code for Smart Water Management

```
<!DOCTYPE html>
<html>
<head>
 <title>Smart Water Management</title>
 <style>
 {
 font-size: 24px;
 font-weight: bold;
 }
{
 padding: 10px 20px;
 font-size: 18px;
 }
 </style>
</head>
<body>
 <h1>Smart Water Management</h1>
 <p>Water Level: <span id="water-level">Loading...</span> cm</p>
 <button id="submit-button" onclick="sendDataToServer()">Submit
Data</button>
 <script>
 function updateWaterLevel() {
 fetch('/getWaterLevelData')
 .then(response => response.json())
 .then(data => {
 document.getElementById('water-level').textContent =
data.waterLevel + " cm";
 })
 .catch(error => {
 console.error('Error fetching water level data:', error);
 });
 }
 function sendDataToServer() {
 fetch('/submitData', { method: 'POST' })
 .then(response => {
 if (response.status === 200) {
 console.log('Data submitted successfully');
 } else {
 console.error('Data submission failed with status:',
response.status);
 }
 })
 .catch(error => {
 console.error('Error submitting data:', error);
 });
 }
```

```
 updateWaterLevel();
 setInterval(updateWaterLevel, 10000);
 </script>
</body>
</html>
```

Output for above program:



IoT-based Water Level Indicator using NodeMCU, Ultrasonic Sensor & Blynk with 0.96" OLED

```
BLYNK_TEMPLATE_ID "TMPLlcLQu4bQ"
BLYNK_TEMPLATE_NAME "water monitor"
BLYNK_AUTH_TOKEN "OgvenxCWu9sG7-9deFGLFCLE4rWCGW7N"

char ssid[] = "GUEST";   //WiFi Name
char pass[] = "admin3421";   //WiFi Password

int emptyTankDistance = 150 ;
int fullTankDistance =  40 ;
```

```
int triggerPer =   10 ;

#include <Adafruit_SSD1306.h>
#include <WiFi.h>
#include <WiFiClient.h>
#include <BlynkSimpleEsp32.h>
#include <AceButton.h>
using namespace ace_button;

#define TRIGPIN    27
#define ECHOPIN    26
#define wifiLed    2
#define BuzzerPin  13
#define RelayPin   14
#define ButtonPin1 12
#define ButtonPin2 33
#define ButtonPin3 32
#define fullpin    25

#define VPIN_BUTTON_1    V1
#define VPIN_BUTTON_2    V2
#define VPIN_BUTTON_3    V3
#define VPIN_BUTTON_4    V4
#define VPIN_BUTTON_5    V5

#define SCREEN_WIDTH 128
#define SCREEN_HEIGHT 32

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

float duration;
float distance;
int   waterLevelPer;
bool  toggleBuzzer = HIGH;

bool toggleRelay = false;
bool modeFlag = true;
bool conection = true;
String currMode;

char auth[] = BLYNK_AUTH_TOKEN;

ButtonConfig config1;
AceButton button1(&config1);
ButtonConfig config2;
AceButton button2(&config2);
ButtonConfig config3;
```

```
AceButton button3(&config3);

void handleEvent1(AceButton*, uint8_t, uint8_t);
void handleEvent2(AceButton*, uint8_t, uint8_t);
void handleEvent3(AceButton*, uint8_t, uint8_t);

BlynkTimer timer;

void checkBlynkStatus() {

  bool isconnected = Blynk.connected();
  if (isconnected == false) {
    digitalWrite(wifiLed, LOW);
    conection = true;

  }
  if (isconnected == true) {
    digitalWrite(wifiLed, HIGH);
    //Serial.println("Blynk Connected");
    conection = false;
  }
}
BLYNK_WRITE(VPIN_BUTTON_3) {
  modeFlag = param.asInt();
  if(!modeFlag && toggleRelay){
      digitalWrite(RelayPin, LOW);
      toggleRelay = false;
    }
    controlBuzzer(500);
    currMode = modeFlag ? "AUTO" : "MANUAL";
}

BLYNK_WRITE(VPIN_BUTTON_4) {
  if(!modeFlag){
    toggleRelay = param.asInt();
    digitalWrite(RelayPin, toggleRelay);
    controlBuzzer(500);
  }
  else{
    Blynk.virtualWrite(VPIN_BUTTON_4, toggleRelay);
  }
}

BLYNK_WRITE(VPIN_BUTTON_5) {
  toggleBuzzer = param.asInt();
  digitalWrite(BuzzerPin, toggleBuzzer);
}
```

```
BLYNK_CONNECTED() {
  Blynk.syncVirtual(VPIN_BUTTON_1);
  Blynk.syncVirtual(VPIN_BUTTON_2);

  Blynk.virtualWrite(VPIN_BUTTON_3, modeFlag);
  Blynk.virtualWrite(VPIN_BUTTON_4, toggleRelay);
  Blynk.virtualWrite(VPIN_BUTTON_5, toggleBuzzer);
}

void displayData(){
  display.clearDisplay();
  display.setTextSize(3);
  display.setCursor(30,0);
  display.print(waterLevelPer);
  display.print(" ");
  display.print("%");
  display.setTextSize(1);
  display.setCursor(0,25);
  display.print(conection ? "OFFLINE" : "ONLINE");
  display.setCursor(60,25);
  display.print(currMode);
  display.setCursor(110,25);
  display.print(toggleRelay ? "! ON" : "OFF");
  display.display();
}

void measureDistance(){
  digitalWrite(TRIGPIN, LOW);
  delayMicroseconds(2);

  digitalWrite(TRIGPIN, HIGH);
  delayMicroseconds(20);

  digitalWrite(TRIGPIN, LOW);

  duration = pulseIn(ECHOPIN, HIGH);

  distance = ((duration / 2) * 0.343)/10;

  if (distance > (fullTankDistance - 10)  && distance < emptyTankDistance ){
    waterLevelPer = map((int)distance ,emptyTankDistance, fullTankDistance, 0,
100);
    Blynk.virtualWrite(VPIN_BUTTON_1, waterLevelPer);
    Blynk.virtualWrite(VPIN_BUTTON_2, (String(distance) + " cm"));

    if (waterLevelPer < triggerPer){

      if(modeFlag){
```

```
        if(!toggleRelay){
          controlBuzzer(500);
          digitalWrite(RelayPin, HIGH); //turn on relay
          toggleRelay = true;
          Blynk.virtualWrite(VPIN_BUTTON_4, toggleRelay);
        }
      }
      else{
        if (toggleBuzzer == HIGH){
          digitalWrite(BuzzerPin, HIGH);
          Serial.println(" BuzzerPin high");
        }
      }
    }

    if (distance < fullTankDistance){
      digitalWrite(fullpin, HIGH);
      if(modeFlag){
        if(toggleRelay){
          digitalWrite(RelayPin, LOW); //turn off relay

          toggleRelay = false;
          Blynk.virtualWrite(VPIN_BUTTON_4, toggleRelay);
          controlBuzzer(500);
        }
      }
      else{
        if (toggleBuzzer == HIGH){
        digitalWrite(BuzzerPin, HIGH);
        }
      }
    }
    if (distance > (fullTankDistance + 5) && waterLevelPer > (triggerPer + 5)){
      toggleBuzzer = HIGH;
      Blynk.virtualWrite(VPIN_BUTTON_5, toggleBuzzer);
      digitalWrite(BuzzerPin, LOW);
    }
    if (distance = fullTankDistance){
    Serial.println(" udh bang ");


    }
  }
  displayData();
  delay(100);
}

void controlBuzzer(int duration){
  digitalWrite(BuzzerPin, HIGH);
```

```
  Serial.println(" BuzzerPin HIT");
  delay(duration);
  digitalWrite(BuzzerPin, LOW);
}

void setup() {
  Serial.begin(9600);

  pinMode(ECHOPIN, INPUT);
  pinMode(TRIGPIN, OUTPUT);
  pinMode(wifiLed, OUTPUT);
  pinMode(RelayPin, OUTPUT);
  pinMode(BuzzerPin, OUTPUT);
  pinMode(fullpin, OUTPUT);

  pinMode(ButtonPin1, INPUT_PULLUP);
  pinMode(ButtonPin2, INPUT_PULLUP);
  pinMode(ButtonPin3, INPUT_PULLUP);

  digitalWrite(wifiLed, HIGH);
  digitalWrite(RelayPin, LOW);
  digitalWrite(BuzzerPin, LOW);

  config1.setEventHandler(button1Handler);
  config2.setEventHandler(button2Handler);
  config3.setEventHandler(button3Handler);

  button1.init(ButtonPin1);
  button2.init(ButtonPin2);
  button3.init(ButtonPin3);

  currMode = modeFlag ? "AUTO" : "MANUAL";

  if(!display.begin(SSD1306_SWITCHCAPVCC, 0x3C)) {
    Serial.println(F("SSD1306 allocation failed"));
    for(;;);
  }
  delay(1000);
  display.setTextSize(1);
  display.setTextColor(WHITE);
  display.clearDisplay();

  WiFi.begin(ssid, pass);
  timer.setInterval(2000L, checkBlynkStatus);
  timer.setInterval(1000L,  measureDistance);
  Blynk.config(auth);
  delay(1000);
```

```
  Blynk.virtualWrite(VPIN_BUTTON_3, modeFlag);
  Blynk.virtualWrite(VPIN_BUTTON_4, toggleRelay);
  Blynk.virtualWrite(VPIN_BUTTON_5, toggleBuzzer);

  delay(500);
}
 void loop() {

  Blynk.run();
  timer.run();
  button1.check();
  button3.check();
  if(!modeFlag){
    button2.check();
  }

}
void button1Handler(AceButton* button, uint8_t eventType, uint8_t buttonState) {
  Serial.println("EVENT1");
  switch (eventType) {
    case AceButton::kEventReleased:
      //Serial.println("kEventReleased");
      if(modeFlag && toggleRelay){
        digitalWrite(RelayPin, LOW);
        toggleRelay = false;
        controlBuzzer(500);
      }
      modeFlag = !modeFlag;
      currMode = modeFlag ? "AUTO" : "MANUAL";
      Blynk.virtualWrite(VPIN_BUTTON_3, modeFlag);
      controlBuzzer(200);
      break;
  }
}

void button2Handler(AceButton* button, uint8_t eventType, uint8_t buttonState) {
  Serial.println("EVENT2");
  switch (eventType) {
    case AceButton::kEventReleased:
      //Serial.println("kEventReleased");
      if(toggleRelay){
        digitalWrite(RelayPin, LOW);
        toggleRelay = false;
      }
      else{
        digitalWrite(RelayPin, HIGH);
        toggleRelay = true;
      }
```

```
      Blynk.virtualWrite(VPIN_BUTTON_4, toggleRelay);
      controlBuzzer(500);
      delay(1000);
      break;
  }
}

void button3Handler(AceButton* button, uint8_t eventType, uint8_t buttonState) {
  Serial.println("EVENT3");
  switch (eventType) {
    case AceButton::kEventReleased:
      digitalWrite(BuzzerPin, LOW);
      toggleBuzzer = LOW;
      Blynk.virtualWrite(VPIN_BUTTON_5, toggleBuzzer);
      break;
  }
}
```

diagram.json:

```
"version": 1,
  "author": "Anonymous maker",
  "editor": "wokwi",
  "parts": [
    { "type": "wokwi-esp32-devkit-v1", "id": "esp", "top": 0, "left": 0,
"attrs": {} },
    { "type": "board-ssd1306", "id": "oled1", "top": 102.61, "left": 139.57,
"attrs": {} },
    {
      "type": "wokwi-pushbutton",
      "id": "btn1",
      "top": 210.09,
      "left": -82.64,
      "attrs": { "color": "green" }
    },
    {
      "type": "wokwi-pushbutton",
      "id": "btn2",
      "top": 209.94,
      "left": -165.82,
      "attrs": { "color": "green" }
    },
    {
      "type": "wokwi-pushbutton",
      "id": "btn3",
```

```
      "top": 209.63,
      "left": -247.46,
      "attrs": { "color": "green" }
    },
    {
      "type": "wokwi-hc-sr04",
      "id": "ultrasonic1",
      "top": -98.88,
      "left": -265.69,
      "attrs": { "distance": "137" }
    },
    {
      "type": "wokwi-led",
      "id": "led1",
      "top": 74.72,
      "left": -383.66,
      "attrs": { "color": "yellow" }
    },
    {
      "type": "wokwi-led",
      "id": "led2",
      "top": 36.18,
      "left": -383.97,
      "attrs": { "color": "red" }
    },
    {
      "type": "wokwi-led",
      "id": "led3",
      "top": -4.65,
      "left": -383.35,
      "attrs": { "color": "blue" }
    }
  ],
  "connections": [
    [ "esp:TX0", "$serialMonitor:RX", "", [] ],
    [ "esp: RX0", "$serialMonitor:TX", "", [] ],
    [ "oled1:GND", "esp:GND.1", "black", [ "v-12.55", "h-51.53", "v52.48" ] ],
    [ "oled1:VCC", "esp:3V3", "red", [ "v-21.39", "h-69.82", "v70.82" ] ],
    [ "oled1:SCL", "esp:D22", "green", [ "v0" ] ],
    [ "oled1:SDA", "esp:D21", "green", [ "v0" ] ],
    [ "ultrasonic1:GND", "esp:GND.2", "black", [ "v0" ] ],
    [ "ultrasonic1:VCC", "esp:VIN", "red", [ "v0" ] ],
    [ "btn3:2.r", "btn2:2.l", "green", [ "h0" ] ],
    [ "btn2:2.r", "btn1:2.l", "green", [ "h0" ] ],
    [ "btn1:2.r", "esp:GND.2", "black", [ "h10.68", "v-57.57", "h-17.06", "v-
34.12" ] ],
    [ "btn1:1.l", "esp:D12", "orange", [ "h-1.3", "v-101.73" ] ],
    [ "esp:D33", "btn2:1.l", "green", [ "h-173.93", "v141.24" ] ],
```

```
    [ "btn3:1.l", "esp:D32", "green", [ "h-7.99", "v-158.4" ] ],
    [ "ultrasonic1:ECHO", "esp:D26", "blue", [ "v0" ] ],
    [ "ultrasonic1:TRIG", "esp:D27", "blue", [ "v0" ] ],
    [ "led3:C", "led2:C", "green", [ "v0" ] ],
    [ "led2:C", "led1:C", "green", [ "v0" ] ],
    [ "led1:A", "esp:D14", "green", [ "v0" ] ],
    [ "led2:A", "esp:D13", "green", [ "h12.22", "v61.32" ] ],
    [ "led3:A", "esp:D25", "green", [ "h49.73", "v53.95" ] ],
    [ "led1:C", "btn3:2.l", "green", [ "v0" ] ]
  ],
  "dependencies": {}
}
```

Simulation: