**PHASE 4 – IOT PROJECT**

**SMART WATER MANAGEMENT SYSTEM**

**MICROPROCESSOR CODE:**

#include <Wire.h>

#include <Adafruit\_Sensor.h>

#include <Adafruit\_BMP280.h>

#include <DHT.h>

// Pin configurations for sensors

const int BMP280\_SDA\_PIN = 4; // Connect BMP280 SDA pin to GPIO 4

const int BMP280\_SCL\_PIN = 5; // Connect BMP280 SCL pin to GPIO 5

const int DHT\_PIN = 2; // Connect DHT11 data pin to GPIO 2

const int ULTRASONIC\_TRIGGER\_PIN = 6; // Connect ultrasonic sensor trigger pin to GPIO 6

const int ULTRASONIC\_ECHO\_PIN = 7; // Connect ultrasonic sensor echo pin to GPIO 7

Adafruit\_BMP280 bmp; // BMP280 sensor

DHT dht(DHT\_PIN, DHT11); // DHT11 sensor

long duration; // To store ultrasonic sensor duration

float distance; // To store calculated distance from ultrasonic sensor

void setup() {

Serial.begin(115200);

// Initialize BMP280 sensor

if (!bmp.begin(BMP280\_SDA\_PIN, BMP280\_SCL\_PIN)) {

Serial.println("Could not find a valid BMP280 sensor, check wiring!");

while (1);

}

// Initialize DHT sensor

dht.begin();

// Ultrasonic sensor pin modes

pinMode(ULTRASONIC\_TRIGGER\_PIN, OUTPUT);

pinMode(ULTRASONIC\_ECHO\_PIN, INPUT);

}

void loop() {

// Read temperature and pressure from BMP280 sensor

float temperatureBMP = bmp.readTemperature();

float pressure = bmp.readPressure() / 100.0; // Pressure in hPa

// Read temperature and humidity from DHT11 sensor

float humidity = dht.readHumidity();

float temperatureDHT = dht.readTemperature(); // Read temperature in Celsius

// Read water level from ultrasonic sensor

digitalWrite(ULTRASONIC\_TRIGGER\_PIN, LOW);

delayMicroseconds(2);

digitalWrite(ULTRASONIC\_TRIGGER\_PIN, HIGH);

delayMicroseconds(10);

digitalWrite(ULTRASONIC\_TRIGGER\_PIN, LOW);

duration = pulseIn(ULTRASONIC\_ECHO\_PIN, HIGH);

distance = duration \* 0.034 / 2; // Calculate distance in centimeters

// Print sensor values

Serial.print("Temperature (BMP280): ");

Serial.print(temperatureBMP);

Serial.println(" °C");

Serial.print("Pressure: ");

Serial.print(pressure);

Serial.println(" hPa");

Serial.print("Temperature (DHT11): ");

Serial.print(temperatureDHT);

Serial.println(" °C");

Serial.print("Humidity: ");

Serial.print(humidity);

Serial.println(" %");

Serial.print("Water Level: ");

Serial.print(distance);

Serial.println(" cm");

delay(5000); // Delay for 5 seconds before reading again

}

**AQI.html PROGRAM**

<!DOCTYPE html>

<html>

<head>

<title>AQI Data from Firebase</title>

<script src="https://www.gstatic.com/firebasejs/8.10.0/firebase-app.js"></script>

<script src="https://www.gstatic.com/firebasejs/8.10.0/firebase-database.js"></script>

</head>

<body>

<h1>Air Quality Index (AQI) Data</h1>

<div id="aqi-data">

<!-- AQI data will be displayed here -->

</div>

<script>

// Initialize Firebase with your project's configuration

var firebaseConfig = {

apiKey: "YOUR\_API\_KEY",

authDomain: "YOUR\_AUTH\_DOMAIN",

databaseURL: "YOUR\_DATABASE\_URL",

projectId: "YOUR\_PROJECT\_ID",

storageBucket: "YOUR\_STORAGE\_BUCKET",

messagingSenderId: "YOUR\_MESSAGING\_SENDER\_ID",

appId: "YOUR\_APP\_ID"

};

firebase.initializeApp(firebaseConfig);

// Reference to your AQI data in Firebase

var aqiRef =

firebase.database().ref("aqi");

// Listen for changes in the AQI data

aqiRef.on("value", function(snapshot) {

var aqiData = snapshot.val();

// Update the HTML to display the AQI data

if (aqiData) {

document.getElementById("aqi-data").innerHTML = "AQI: " + aqiData;

} else {

document.getElementById("aqi-data").innerHTML = "No data available";

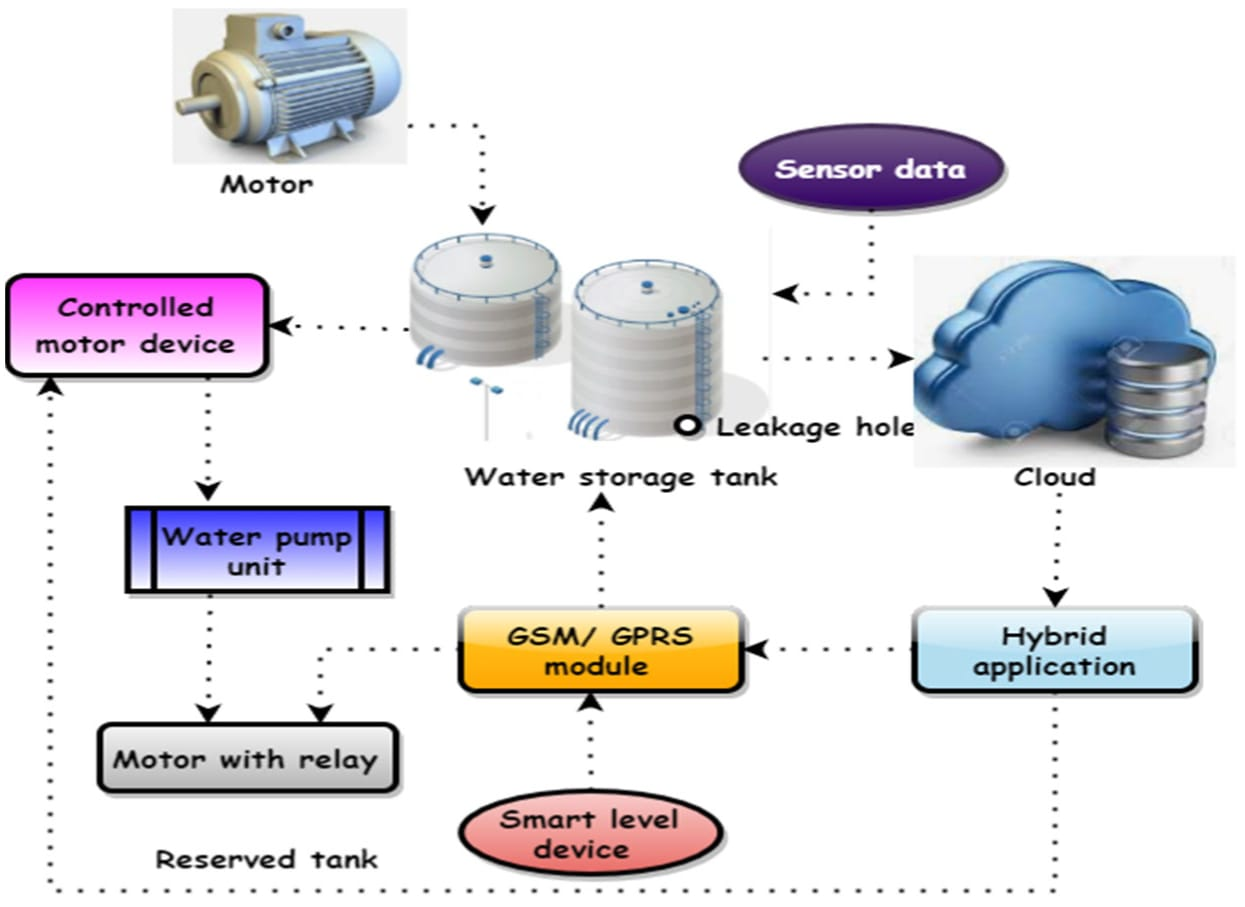
}

});

</script>

</body>

</html>



**HTML Program**

<!DOCTYPE html>

<html>

<head>

<meta charset="UTF-8">

<title>Smart Water Management IoT Project</title>

<!-- Add your CSS links here if needed -->

</head>

<body>

<header>

<h1>Smart Water Management System</h1>

</header>

<nav>

<!-- Add navigation links if necessary -->

</nav>

<main>

<section id="sensor-data">

<h2>Sensor Data</h2>

<p>Current Water Level: <span id="water-level">N/A</span> cm</p>

<p>Temperature: <span id="temperature">N/A</span> °C</p>

<p>Water Quality: <span id="water-quality">N/A</span></p>

</section>

<section id="control-panel">

<h2>Control Panel</h2>

<button id="pump-control">Start Pump</button>

</section>

</main>

<footer>

<p>&copy; 2023 Your Company Name</p>

</footer>

<!-- Add your JavaScript code here to interact with IoT data and control the system -->

</body>

</html>

**Water management sensor**

#define BLYNK\_TEMPLATE\_ID "TMPLlcLQu4bQ"

#define BLYNK\_TEMPLATE\_NAME "water monitor"

#define BLYNK\_AUTH\_TOKEN "OgvenxCWu9sG7-9deFGLFCLE4rWCGW7N"

// Your WiFi credentials.

// Set password to "" for open networks.

char ssid[] = "Wokwi-GUEST";   //WiFi Name

char pass[] = "";   //WiFi Password

//Set Water Level Distance in CM

int emptyTankDistance = 150 ;  //Distance when tank is empty

int fullTankDistance =  40 ;  //Distance when tank is full (must be greater than 25cm)

//Set trigger value in percentage

int triggerPer =   10 ;  //alarm/pump will start when water level drop below triggerPer

#include <Adafruit\_SSD1306.h>

#include <WiFi.h>

#include <WiFiClient.h>

#include <BlynkSimpleEsp32.h>

#include <AceButton.h>

using namespace ace\_button;

// Define connections to sensor

#define TRIGPIN    27  //D6

#define ECHOPIN    26  //D7

#define wifiLed    2  //D0

#define BuzzerPin  13  //D3

#define RelayPin   14 //D5

#define ButtonPin1 12   //RX   //Mode

#define ButtonPin2 33  //SD3  //Relay

#define ButtonPin3 32  //D4   //STOP Buzzer

#define fullpin    25

//Change the virtual pins according the rooms

#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 // OLED display width, in pixels

#define SCREEN\_HEIGHT 32 // OLED display height, in pixels

// Declaration for an SSD1306 display connected to I2C (SDA, SCL pins)

#define OLED\_RESET     -1 // Reset pin # (or -1 if sharing Arduino reset pin)

Adafruit\_SSD1306 display(SCREEN\_WIDTH, SCREEN\_HEIGHT, &**Wire**, OLED\_RESET);

float duration;

float distance;

int   waterLevelPer;

bool  toggleBuzzer = HIGH; //Define to remember the toggle state

bool toggleRelay = false; //Define the toggle state for relay

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() { // called every 3 seconds by SimpleTimer

  bool isconnected = Blynk.connected();

  if (isconnected == false) {

    //Serial.println("Blynk Not Connected");

    digitalWrite(wifiLed, LOW);

    conection = true;

  }

  if (isconnected == true) {

    digitalWrite(wifiLed, HIGH);

    //Serial.println("Blynk Connected");

    conection = false;

  }

}

// When App button is pushed - switch the state

BLYNK\_WRITE(VPIN\_BUTTON\_3) {

  modeFlag = param.asInt();

  if(!modeFlag && toggleRelay){

      digitalWrite(RelayPin, LOW);  //turn off the pump

      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(){

  // Set the trigger pin LOW for 2uS

  digitalWrite(TRIGPIN, LOW);

  delayMicroseconds(2);

  // Set the trigger pin HIGH for 20us to send pulse

  digitalWrite(TRIGPIN, HIGH);

  delayMicroseconds(20);

  // Return the trigger pin to LOW

  digitalWrite(TRIGPIN, LOW);

  // Measure the width of the incoming pulse

  duration = pulseIn(ECHOPIN, HIGH);

  // Determine distance from duration

  // Use 343 metres per second as speed of sound

  // Divide by 1000 as we want millimeters

  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"));

    // Print result to serial monitor

//    Serial.print("Distance: ");

//    Serial.print(distance);

//    Serial.println(" 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() {

  // Set up serial monitor

**Serial**.begin(9600);

  // Set pinmodes for sensor connections

  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); // check if Blynk server is connected every 2 seconds

  timer.setInterval(1000L,  measureDistance); // measure water level every 1 seconds

  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(); // Initiates SimpleTimer

  button1.check(); //mode change

  button3.check(); //buzzer reset

  if(!modeFlag){  //if in manual mode

    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);  //turn off the pump

        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);  //turn off the pump

        toggleRelay = false;

      }

      else{

        digitalWrite(RelayPin, HIGH);  //turn on the pump

        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:

      //Serial.println("kEventReleased");

      digitalWrite(BuzzerPin, LOW);

      toggleBuzzer = LOW;

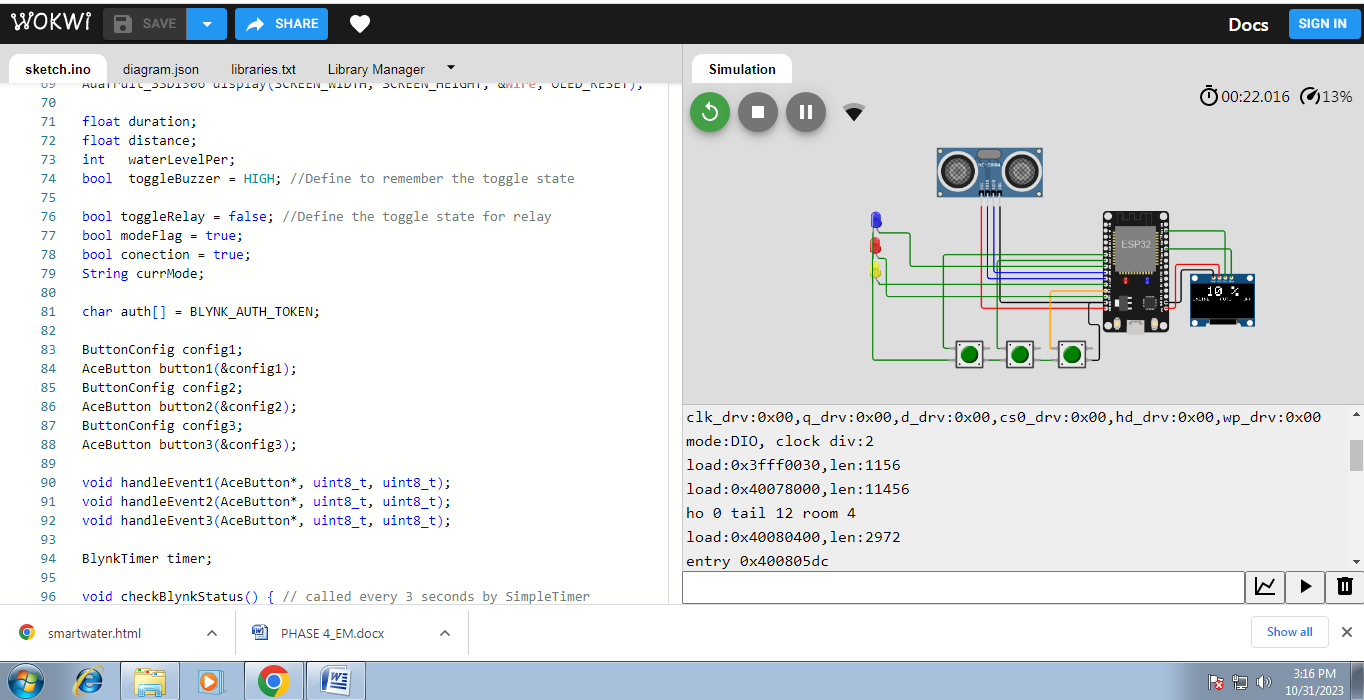
      Blynk.virtualWrite(VPIN\_BUTTON\_5, toggleBuzzer);

      break;

  }

}

**RESULT:**



**Conclusion:**

In conclusion, a smart water management system is a crucial and innovative solution for addressing the growing challenges of water scarcity and sustainability. By leveraging advanced technologies like IoT, data analytics, and automation, such a system can optimize water usage, detect leaks, and improve overall efficiency in water distribution. This not only conserves a precious resource but also reduces costs and enhances the quality of life for communities. As we continue to face water-related issues, investing in smart water management systems is a forward-looking and sustainable choice that can benefit both the environment and society.