**PHASE-3**

SMART WATER

MANAGEMENT

**TEAM MEMBERS:**

G Bharathi 412721106007

S Kalpana 412721106022

T Raghul 412721106036

M Vignesh 412721106051

**OBJECTIVE:**

Our aim is to create a system, for monitoring the consumption and controlling water level in tank used in public bathroom, school/ colleges and most of the public places. This system is designed to keep track of the water levels in a tank in time with options for both automated and manual control. To achieve this we will be using the Blynk platform, which allows users to control and monitor the system remotely through an app.

The system consists of components. We will be using a distance sensor to measure the water level, an OLED display to provide visual feedback and various elements within the Blynk app for user interaction. The system has two operating modes; "AUTO" and "MANUAL." In "AUTO" mode it constantly monitors the water level. Activates a pump via a relay when it falls below a threshold. In "MANUAL" mode users have control over the pump with a simple button press.

This project has applications in scenarios such as managing water tanks in homes, public places (schools, colleges, public toilets, agricultural settings etc). It provides a solution by ensuring a water supply automating the process of filling up tanks and notifying users when the water level becomes critically low. Overall this water monitoring and control system is a contribution to IoT based solutions, for managing the consumption of water resources.

**CODING:**

#define BLYNK\_TEMPLATE\_ID "TMPLlcLQu4bQ"

#define BLYNK\_TEMPLATE\_NAME "water monitor"

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

// 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;

  }

}

**CODE EXPLANATION:**

To begin, the code sets up multiple configurations, such as Blynk template details, Wi-Fi credentials, water level thresholds, pin assignments, and virtual pins for Blynk. It also initializes the OLED display, buttons.

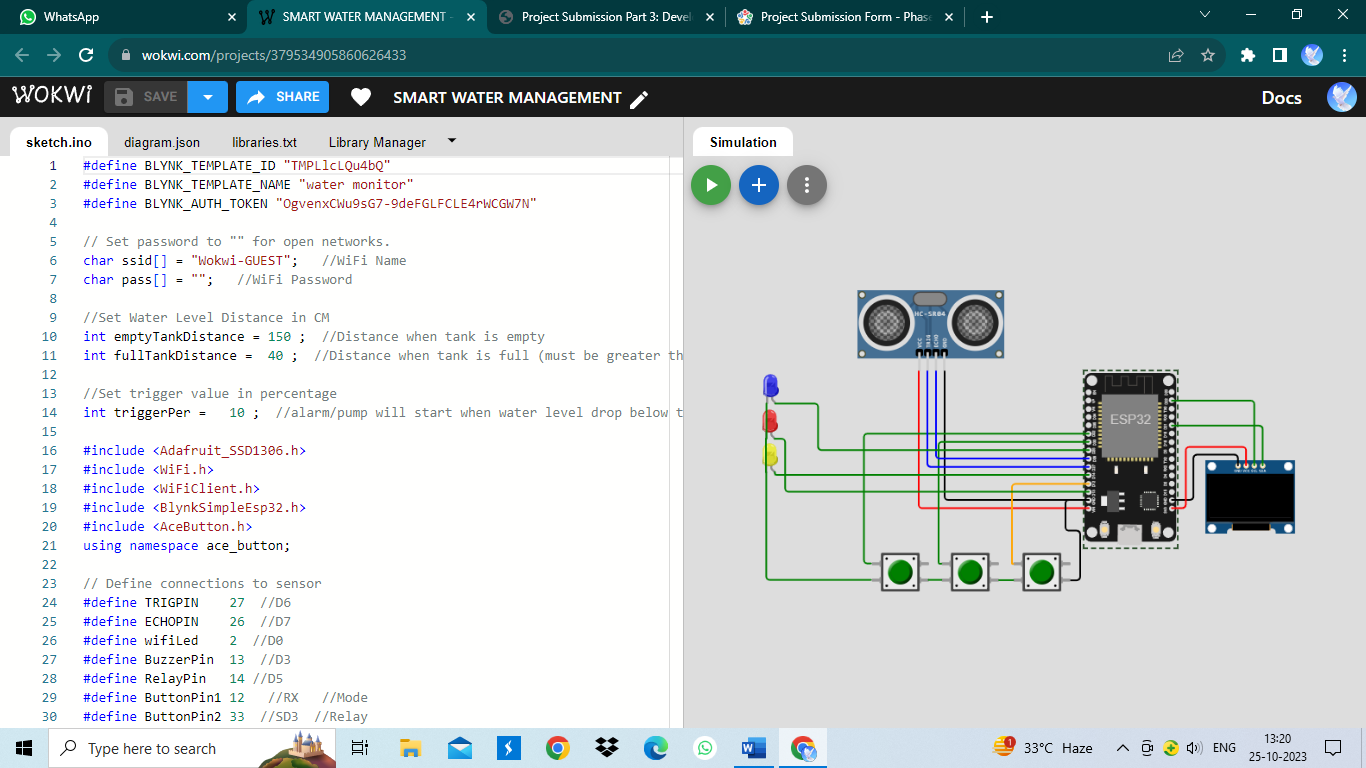
For measuring the water level in the tank, the system utilizes an ultrasonic distance sensor. By sending ultrasonic pulses triggered by TRIGPIN and measuring the return time of the pulses received by ECHOPIN, the system calculates the distance from the sensor to the water surface and determines the water level percentage. This measurement is continuously updated.

To enable remote monitoring and control, the system integrates with the Blynk platform. Users can access and control the system through the Blynk mobile app, utilizing Blynk virtual pins for actions such as changing system modes, manually controlling the relay, and toggling the buzzer.

The system has the ability to control a buzzer in order to provide audible alerts. The buzzer can be activated based on the current mode (either MANUAL or AUTO) and the water level conditions. A relay is utilized to control the water pump. When the system is in "AUTO" mode and the water level drops below a specific threshold, the relay is activated. The system utilizes LED indicators for status indication. These indicators include LEDs for Wi-Fi connectivity, tank status (whether it is full or not), and pump operation. The system continuously monitors the water level, updating the OLED display with the current percentage. It also checks for any alarms or conditions requiring relay control. Additionally, it periodically checks the Blynk connection status to ensure proper communication with the mobile app. Users can interact with the system through the Blynk app, allowing them to switch between different modes, manually control the pump and buzzer, and receive real-time updates on water levels and system status through the OLED display.

The system has the ability to control a buzzer in order to provide audible alerts. The buzzer can be activated based on the current mode (either MANUAL or AUTO) and the water level conditions. A relay is utilized to control the water pump. When the system is in "AUTO" mode and the water level drops below a specific threshold, the relay is activated. The system utilizes LED indicators for status indication. These indicators include LEDs for Wi-Fi connectivity, tank status (whether it is full or not), and pump operation. The system continuously monitors the water level, updating the OLED display with the current percentage. It also checks for any alarms or conditions requiring relay control. Additionally, it periodically checks the Blynk connection status to ensure proper communication with the mobile app. Users can interact with the system through the Blynk app, allowing them to switch between different modes, manually control the pump and buzzer, and receive real-time updates on water levels and system status through the OLED display.

**SIMULATION:**



**CONCLUSION:**

In conclusion, this project helps us to monitoring the consumption of water (usage of water in the tank) by considering its level of water surface in the tank. If the water level is decreases down the threshold level then automatically makes the motor to run and to maintaining the water level in the tank which are used in the public toilets/bathroom, schools/colleges, hospitals and other public places. It also gives the real time data to the public water management team about the water level and its flow meter in the public places where the water places an important role. It indirectly helps to indicate the consumption of water in the public places by the user and allows for remote control and thereby monitoring through the blynk app. To know the output of our project, the below mentioned hyperlink will make you to see our work done in the wokwi website and we apologize for any mistake if happenned in the code and also in the simulation .Also we hope that our simulation is fullfill yours expectation.

**OUR SIMULATION:**

<https://wokwi.com/projects/379534905860626433>

THANK YOU