SMART WATER MANAGEMENT

Date	**26-10-2023**
Team ID	**666**
Team Name	**Proj_223434_Team_2**
Project Name	**Smart Water Management**

INTRODUCTION:

Water supply is the most important thing in daily home activity. We commonly supply the water by pumping the groundwater to fill a water tank. However, the utilization of non-automated switch used to turn on and turn off a pumping machine sometimes causes either the water spills or a wasteful electrical consumption. In this work, an automated water tank filling system will be proposed. By applying an ultrasonic sensor, an ultrasonic transmitter is mounted on the top of the tank and transmits an ultrasonic pulse down into the tank. The transmitter is programmed to automatically determine the liquid level and switch the pumping machine. The dynamics of water flow and liquid level during filling and draining the water tank will be reported. We hope to this system, people will enjoy supplying water without their worries related to water spills and a wasteful electrical consumption.

SOURCE CODE:

```
#define BLYNK_TEMPLATE_ID "TMPLlcLQu4bQ"

#define BLYNK_TEMPLATE_NAME "water monitor"

#define BLYNK_AUTH_TOKEN "OgvenxCWu9sG7-9deFGLFCLE4rWCGW7N"


char ssid[] = "Wokwi-GUEST";

char pass[] = "";

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     -1
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);
    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);
        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);

        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);
}
```

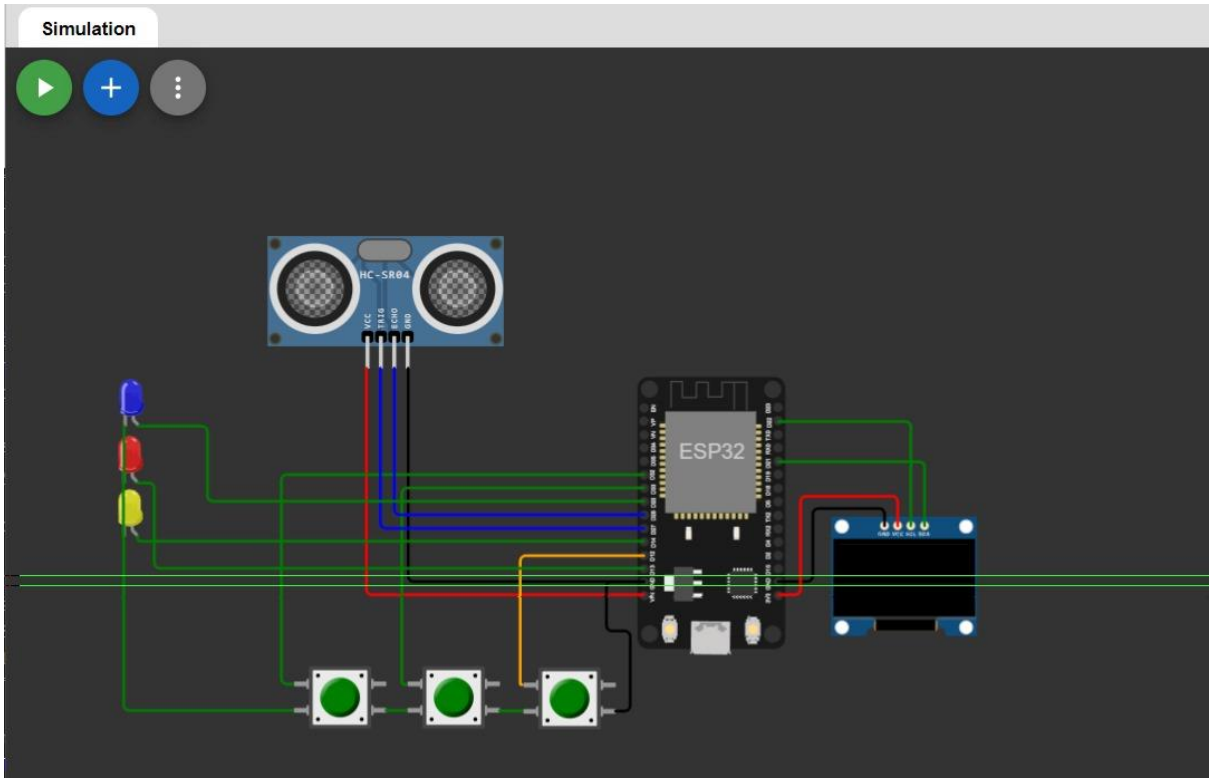
```
void controlBuzzer(int duration){
  digitalWrite(BuzzerPin, HIGH);
  Serial.println(" BuzzerPin HIT");
  delay(duration);
  digitalWrite(BuzzerPin, LOW);
}

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:
      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:
      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;
  }
}
```

CIRCUIT DIAGRAM:



WEB DEDVELPOMENT PLATFORM:

HTML:

```
   <!DOCTYPE html>
<html lang="en">

<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Water Monitor</title>
    <link rel="stylesheet" href="styles.css">
</head>

<body>
```

```
    <div class="container">
        <h1>Water Monitor</h1>
        <div class="status">
            <h2>Water Level: <span id="waterLevel">0%</span></h2>
            <h2>Status: <span id="status">Offline</span></h2>
        </div>
        <div class="controls">
            <button id="toggleMode">Toggle Mode</button>
            <button id="toggleRelay">Toggle Relay</button>
            <button id="toggleBuzzer">Toggle Buzzer</button>
        </div>
    </div>
    <script src="script.js"></script>
</body>
</html>
```

CSS:

```
body {
    font-family: Arial, sans-serif;
}

.container {
    text-align: center;
    margin-top: 50px;
}

.status {
    margin-bottom: 20px;
}
```

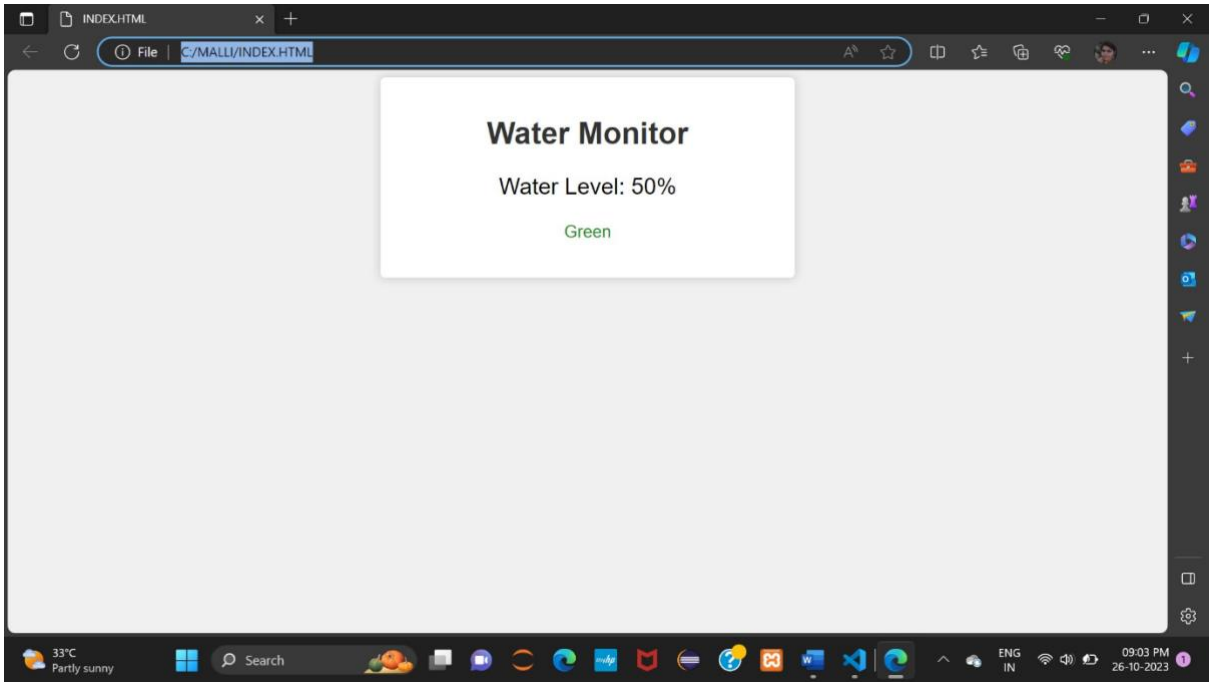
```
.controls {
    display: flex;
    justify-content: center;
}

button {
    margin: 10px;
    padding: 10px 20px;
    font-size: 16px;
}
```

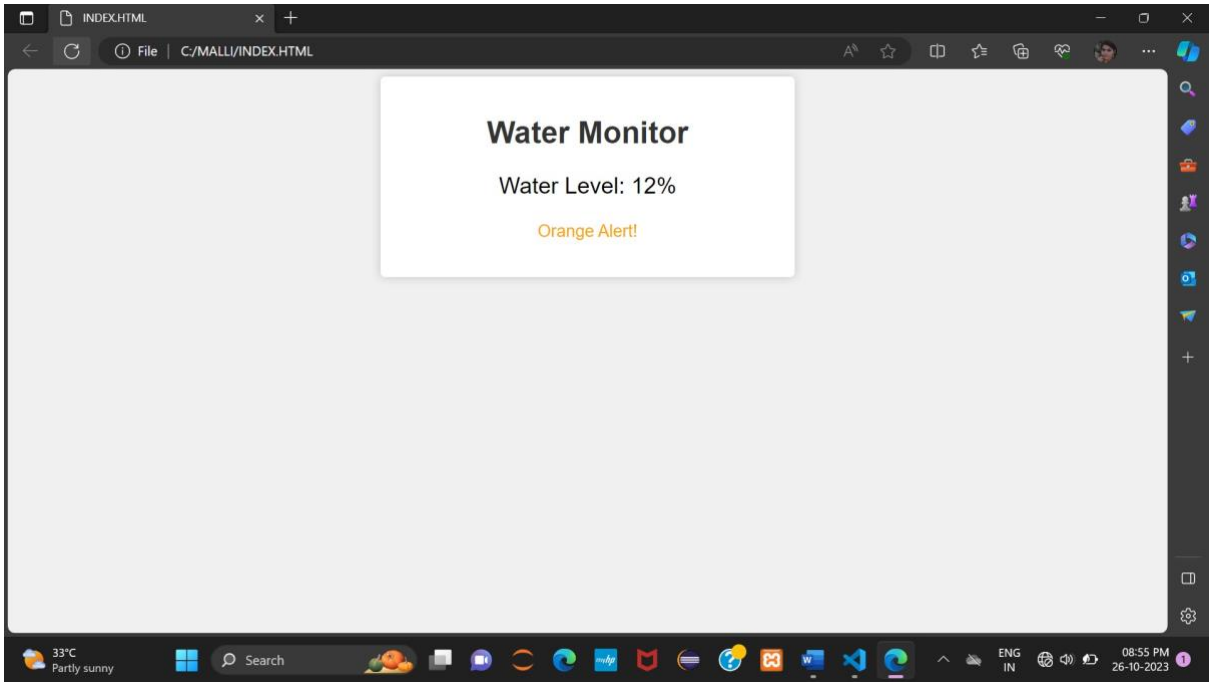
JAVASCRIPT:

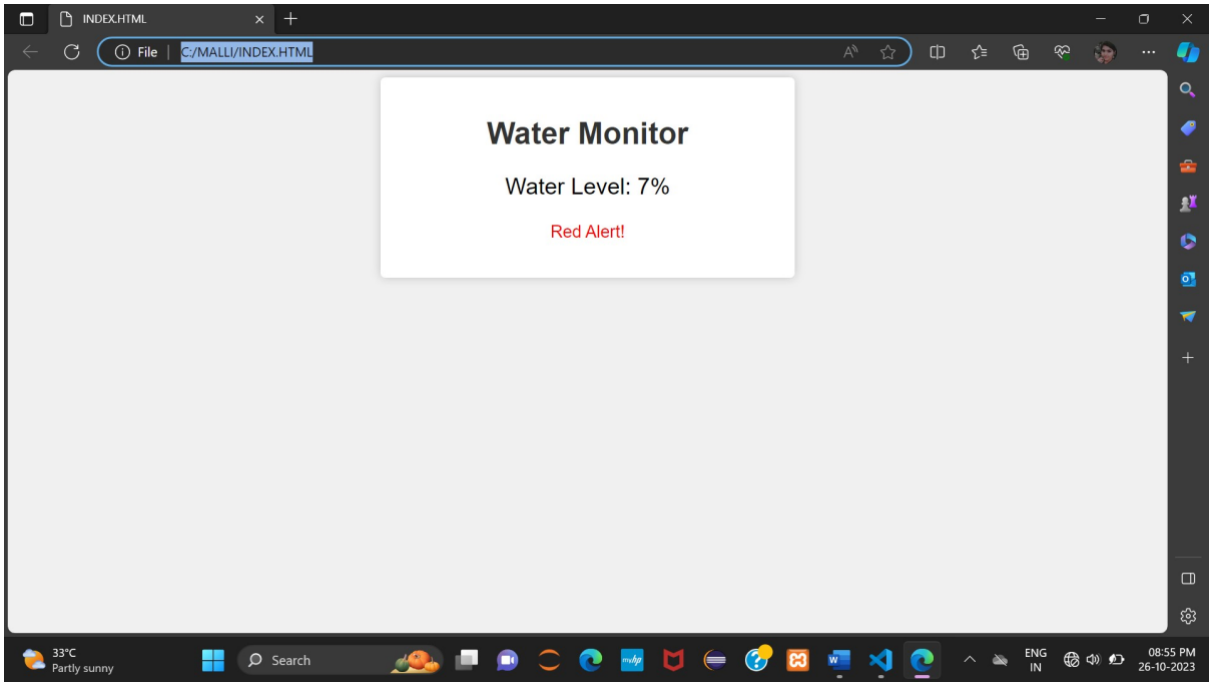
```
document.addEventListener("DOMContentLoaded", function () {
    var waterLevelSpan = document.getElementById("waterLevel");
    var statusSpan = document.getElementById("status");
    var toggleModeBtn = document.getElementById("toggleMode");
    var toggleRelayBtn = document.getElementById("toggleRelay");
    var toggleBuzzerBtn = document.getElementById("toggleBuzzer");

    // Example WebSocket connection to the server (replace 'localhost' with your server address)
    var socket = new WebSocket("ws://localhost:3000");

    socket.onopen = function () {
        statusSpan.textContent = "Online";
        statusSpan.style.color = "green";
    };

    socket.onmessage = function (event) {
        var data = JSON.parse(event.data);
        waterLevelSpan.textContent = data.waterLevel + "%";
```

```
    };

    socket.onclose = function () {
        statusSpan.textContent = "Offline";
        statusSpan.style.color = "red";
    };

    toggleModeBtn.addEventListener("click", function () {
        // Send a message to the server to toggle the mode
        socket.send("toggleMode");
    });

    toggleRelayBtn.addEventListener("click", function () {
        // Send a message to the server to toggle the relay
        socket.send("toggleRelay");
    });

    toggleBuzzerBtn.addEventListener("click", function () {
        // Send a message to the server to toggle the buzzer
        socket.send("toggleBuzzer");
    });
});
```

OUTPUT:


INDEX.HTML
File | C:/MALLI/INDEX.HTML
Water Monitor
Water Level: 50%
Green
33°C
Partly sunny
Search
ENG
IN
09:03 PM
26-10-2023


INDEX.HTML
File | C:/MALLI/INDEX.HTML
Water Monitor
Water Level: 12%
Orange Alert!
33°C
Partly sunny
Search
ENG
IN
08:55 PM
26-10-2023



CONCLUSION:

The source code for Smart Water Management using C and for simulation diagram Json has been successfully created. By using web development technologies like HTML, CSS, JAVASCRIPT to create a platform that displays real-time water level status. We have designed the platform to receive and display real-time water level data.