Project

SMART WATER MANAGEMENT

TEAM NAME

PROJECT-212982-TEAM-1

project objectives:

•Automatically control water levels by turning pumps or valves on/off based on predefined thresholds or user commands.

•Optimize energy usage by ensuring pumps operate only when necessary.

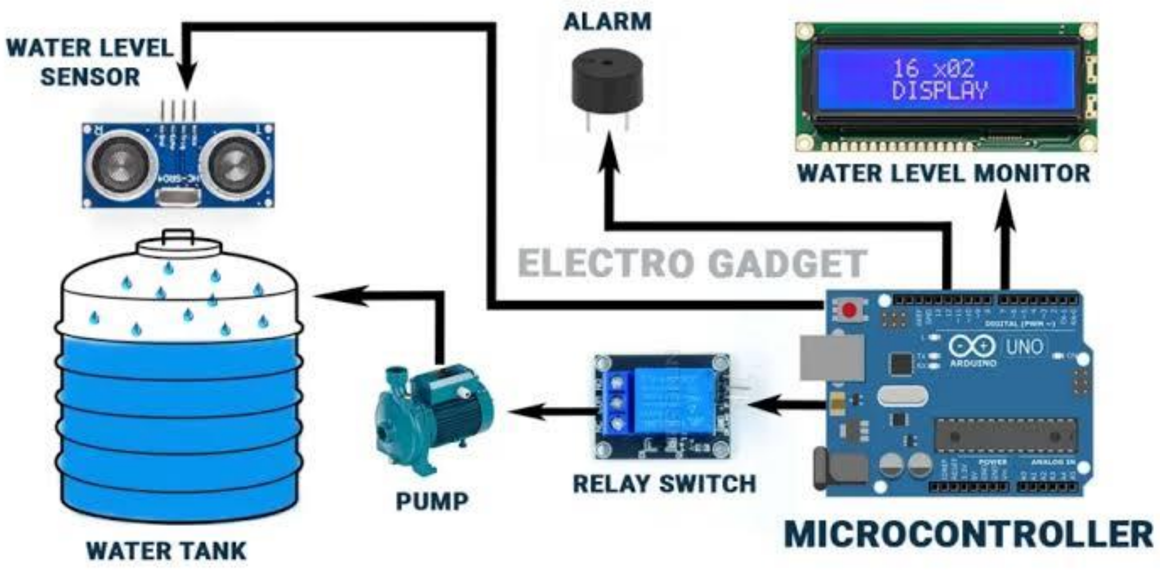
•Implement power-saving features to extend the life of batteries or reduce electricity costs.

•Design the system to be easily expandable to monitor and control multiple locations if needed

•Optimize hardware and software components to keep the system cost-effective.

Project Architecture:

Technical Architecture:



Program:

```
#include <Adafruit_SSD1306.h>

#include <ESP8266WiFi.h>

#include <BlynkSimpleEsp8266.h>

#include <AceButton.h>


#define BLYNK_TEMPLATE_ID “****************”

#define BLYNK_TEMPLATE_NAME “***********************”

#define BLYNK_AUTH_TOKEN “**************************”


Char ssid[] = “*****************”;

Char pass[] = “***************”;


Int emptyTankDistance = 160;

Int fullTankDistance = 20;

Int triggerPer = 20;


Using namespace ace_button;


#define TRIG 12             //D6

#define ECHO 13              //D7

#define Relay 14         //D5

#define BP1 2           //D0

#define BP2 13           //D3

#define BP3 15           //D4


#define V_B_1 V1

#define V_B_3 V3

#define V_B_4 V4
```

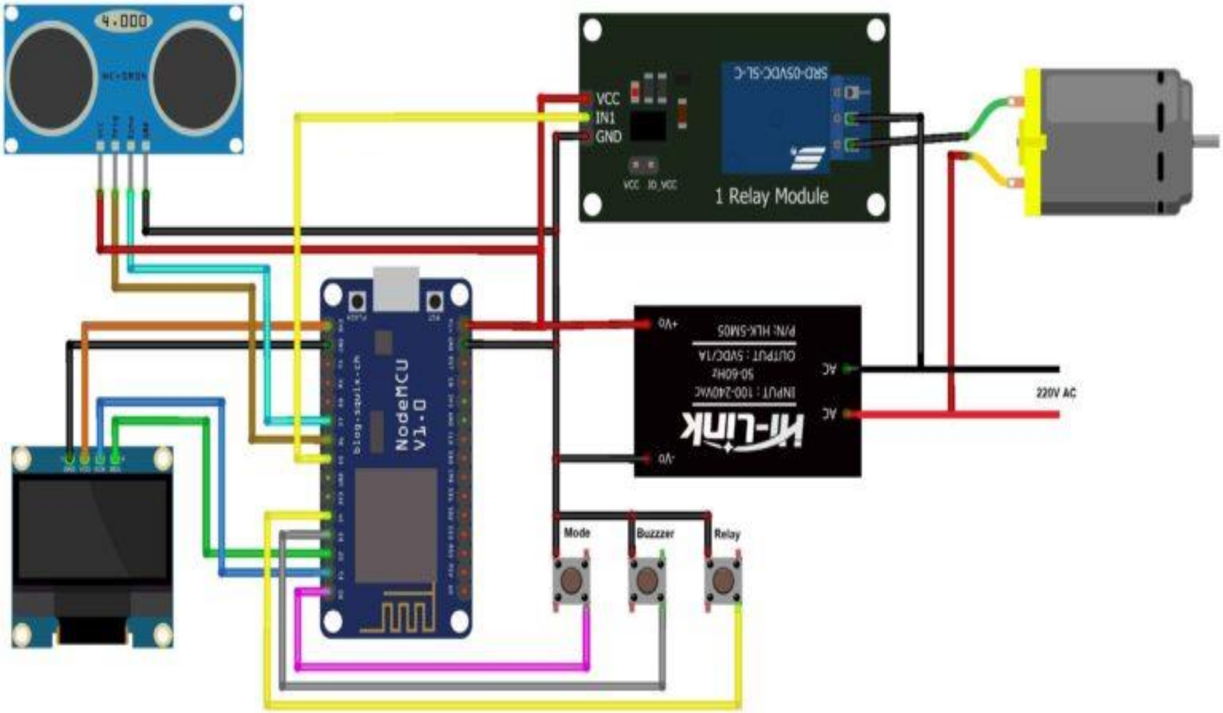
```
#define SCREEN_WIDTH 128
#define SCREEN_HEIGHT 32

#define OLED_RESET -1
Adafruit_SSD1306 display(SCREEN_WIDTH, SCREEN_HEIGHT, &Wire, OLED_RESET);

Float duration;
Float distance;
Int waterLevelPer;

Bool toggleRelay = false;
Bool modeFlag = 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) {
  }
  If (isconnected == true) {
  }
}

BLYNK_WRITE(VPIN_BUTTON_3) {
  modeFlag = param.asInt();
  if (!modeFlag && toggleRelay) {
    digitalWrite(Relay, LOW);
    toggleRelay = false;
  }
  currMode = modeFlag ? “AUTO” : “MANUAL”;
}

BLYNK_WRITE(VPIN_BUTTON_4) {
  If (!modeFlag) {
    toggleRelay = param.asInt();
    digitalWrite(Relay, toggleRelay);
  } else {
    Blynk.virtualWrite(V_B_4, toggleRelay);
  }
}

BLYNK_CONNECTED() {
  Blynk.syncVirtual(V_B_1);
  Blynk.virtualWrite(V_B_3, modeFlag);
  Blynk.virtualWrite(V_B_4, toggleRelay);
}

Void displayData() {
  Display.clearDisplay();
  Display.setTextSize(3);
  Display.setCursor(30, 0);
  Display.print(waterLevelPer);
  Display.print(“ “);
  Display.print(“%”);
  Display.setTextSize(1);
  Display.setCursor(20, 25);
  Display.print(currMode);
  Display.setCursor(95, 25);
  Display.print(toggleRelay ? “ON” : “OFF”);
  Display.display();
}

Void measureDistance() {

  digitalWrite(TRIG, LOW);
  delayMicroseconds(2);
  digitalWrite(TRIG, HIGH);
  delayMicroseconds(20);
  digitalWrite(TRIG, LOW);
  duration = pulseIn(ECHO, HIGH);
  distance = ((duration / 2) * 0.343) / 10;
```

```
 if (distance > (fullTankDistance – 15) && distance < emptyTankDistance) {
  waterLevelPer = map((int)distance, emptyTankDistance, fullTankDistance, 0, 100);
  Blynk.virtualWrite(V_B_1, waterLevelPer);
  If (waterLevelPer < triggerPer) {
   If (modeFlag) {
    If (!toggleRelay) {
     digitalWrite(Relay, HIGH);
     toggleRelay = true;
     Blynk.virtualWrite(V_B_4, toggleRelay);
    }
   }
  }
  If (distance < fullTankDistance) {
   If (modeFlag) {
    If (toggleRelay) {
     digitalWrite(Relay, LOW);
     toggleRelay = false;
     Blynk.virtualWrite(V_B_4, toggleRelay);
    }
   }
  }
 }
 displayData();
 delay(100);
}

Void setup() {
 Serial.begin(9600);
 pinMode(ECHO, INPUT);
 pinMode(TRIG, OUTPUT);
 pinMode(Relay, OUTPUT);

 pinMode(BP1, INPUT_PULLUP);
 pinMode(BP2, INPUT_PULLUP);
 pinMode(BP3, INPUT_PULLUP);

 digitalWrite(Relay, HIGH);

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

 button1.init(BP1);
 button2.init(BP2);
 button3.init(BP3);

 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(V_B_3, modeFlag);
  Blynk.virtualWrite(V_B_4, toggleRelay);

  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(Relay, LOW);
        toggleRelay = false;
      }
      modeFlag = !modeFlag;
      currMode = modeFlag ? “AUTO” : “MANUAL”;
      Blynk.virtualWrite(V_B_3, modeFlag);
      Break;
  }
}

Void button2Handler(AceButton* button, uint8_t eventType, uint8_t buttonState) {
  Serial.println(“EVENT2”);
  Switch (eventType) {
    Case AceButton::kEventReleased:
      If (toggleRelay) {
        digitalWrite(Relay, LOW);
        toggleRelay = false;
      } else {
        digitalWrite(Relay, HIGH);
        toggleRelay = true;
      }
      Blynk.virtualWrite(V_B_4, toggleRelay);
      Delay(1000);
      Break;
  }
}

Void button3Handler(AceButton* button, uint8_t eventType, uint8_t buttonState) {
  Serial.println(“EVENT3”);
  Switch (eventType) {
    Case AceButton::kEventReleased:
      Break;
  }
}
```

Explanation of connections:



1. **ESP8266 Board (e.g., NodeMCU):** This is the microcontroller that will handle the communication with your Wi-Fi network and the cloud.

2. **Ultrasonic Sensor (e.g., HC-SR04):** This sensor will be used to measure the water level.

3. **Relay Module:** This will control the water pump. When the water level falls below a certain threshold, the ESP8266 will trigger the relay to turn on the pump.

4. **Water Pump:** This is the device that pumps water.

5. **Power Supply:** Ensure you have a power source that can supply the necessary voltage and current for all components.

Here’s a basic explanation of connections:

1. **Ultrasonic Sensor (HC-SR04):**

   - **VCC** to 5V on the ESP8266.
   - **GND** to GND on the ESP8266.
   - **Trig** to a GPIO pin (e.g., D2) on the ESP8266.
   - **Echo** to a GPIO pin (e.g., D3) on the ESP8266.

2. **Relay Module:**

   - **VCC** to 5V on the ESP8266.
   - **GND** to GND on the ESP8266.
   - **IN** to a GPIO pin (e.g., D4) on the ESP8266.

3. **Water Pump:**

   - Connect one wire of the pump to one terminal of the relay.
   - Connect the other wire of the pump to the positive terminal of the power source.
   - Connect the negative terminal of the power source to the common (COM) terminal of the relay.

4. **ESP8266 (NodeMCU):**

   - **VCC** to 5V from the power source.
   - **GND** to the ground of the power source.

- Connect the **CH_PD** and **RST** pins to 3.3V.

- Connect the **TX** and **RX** pins to GPIO pins on the ESP8266 (usually D1 and D2).

- Connect the ESP8266 to your Wi-Fi network.