

Smart Water System

IoT Based Water Level Indicator using Ultrasonic Sensor

Source Codes for Blynk ESP32 Water Level Sensor:

```

/*****
**
*   TITLE: IoT-based Water Level Indicator using ESP32, Ultrasonic Sensor & Blynk
with 0.96" OLED
*   Click on the following links to learn more.
*   YouTube Video: https://youtu.be/9geREeE13jc
*   Related Blog : https://iotcircuithub.com/esp32-projects/
*
*   This code is provided free for project purpose and fair use only.
*   Please do mail us to techstudycell@gmail.com if you want to use it
commercially.
*   Copyrighted © by Tech StudyCell
*
*   Preferences--> Additional boards Manager URLs :
*   https://raw.githubusercontent.com/espressif/arduino-esp32/gh-
pages/package\_esp32\_dev\_index.json,
http://arduino.esp8266.com/stable/package\_esp8266com\_index.json
*
*   Download Board ESP32 (2.0.5) : https://github.com/espressif/arduino-esp32
*
*   Download the libraries
*   Blynk Library (1.1.0): https://github.com/blynkkk/blynk-library
*   Adafruit_SSD1306 Library (2.5.7):
https://github.com/adafruit/Adafruit\_SSD1306
*   AceButton Library (1.9.2): https://github.com/bxparks/AceButton
*****
**/

/* Fill-in your Template ID (only if using Blynk.Cloud) */
#define BLYNK_TEMPLATE_ID ""
#define BLYNK_DEVICE_NAME ""
#define BLYNK_AUTH_TOKEN ""

// Your WiFi credentials.

```

```
// Set password to "" for open networks.
char ssid[] = "";
char pass[] = "";

//Set Water Level Distance in CM
int emptyTankDistance = 70 ; //Distance when tank is empty
int fullTankDistance = 30 ; //Distance when tank is full

//Set trigger value in percentage
int triggerPer = 10 ; //alarm 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 //D27
#define ECHOPIN 26 //D26
#define wifiled 2 //D2
#define ButtonPin1 12 //D12
#define BuzzerPin 13 //D13
#define GreenLed 14 //D14

//Change the virtual pins according the rooms
#define VPIN_BUTTON_1 V1
#define VPIN_BUTTON_2 V2

#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

char auth[] = BLYNK_AUTH_TOKEN;
```

```
ButtonConfig config1;
AceButton button1(&config1);

void handleEvent1(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);
    }
    if (isconnected == true) {
        digitalWrite(wifiLed, HIGH);
        //Serial.println("Blynk Connected");
    }
}

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

void displayData(int value){
    display.clearDisplay();
    display.setTextSize(4);
    display.setCursor(8,2);
    display.print(value);
    display.print(" ");
    display.print("%");
    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);
    displayData(waterLevelPer);
    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){
        digitalWrite(GreenLed, HIGH);
        if (toggleBuzzer == HIGH){
            digitalWrite(BuzzerPin, HIGH);
        }
    }
    if (distance < fullTankDistance){
        digitalWrite(GreenLed, LOW);
        if (toggleBuzzer == HIGH){
            digitalWrite(BuzzerPin, HIGH);
        }
    }
}

if (distance > (fullTankDistance + 5) && waterLevelPer > (triggerPer + 5)){
    toggleBuzzer = HIGH;
    digitalWrite(BuzzerPin, LOW);
}
}

// Delay before repeating measurement
delay(100);
```

```
}

void setup() {
  // Set up serial monitor
  Serial.begin(115200);

  // Set pinmodes for sensor connections
  pinMode(ECHOPIN, INPUT);
  pinMode(TRIGPIN, OUTPUT);
  pinMode(wifiLed, OUTPUT);
  pinMode(GreenLed, OUTPUT);
  pinMode(BuzzerPin, OUTPUT);

  pinMode(ButtonPin1, INPUT_PULLUP);

  digitalWrite(wifiLed, LOW);
  digitalWrite(GreenLed, LOW);
  digitalWrite(BuzzerPin, LOW);

  config1.setEventHandler(button1Handler);

  button1.init(ButtonPin1);

  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
  Blynk.config(auth);
  delay(1000);
}

void loop() {

  measureDistance();

  Blynk.run();
}
```

```
    timer.run(); // Initiates SimpleTimer

    button1.check();

}

void button1Handler(AceButton* button, uint8_t eventType, uint8_t buttonState) {
    Serial.println("EVENT1");
    switch (eventType) {
        case AceButton::kEventReleased:
            //Serial.println("kEventReleased");
            digitalWrite(BuzzerPin, LOW);
            toggleBuzzer = LOW;
            break;
    }
}
```

ESP32 WaterLevel

Info Metadata **Datastreams** Events Automations Web Dashboard Mobile Dashboard

Search datastream

ID	Name	Alias	Color	Pin	Data Type	Units	Is Raw	Idm	Max	Decimals	Default Value
1	WaterLevel	WaterLevel		V1	Integer	%	false	0	100	-	0
2	Distance	Distance		V2	String		false			-	

ESP32 WaterLevel

Info Metadata Datastreams Events **Automations** Web Dashboard Mobile Dashboard

Define which Datastreams will be available in Automation actions and conditions.
Only Virtual Pin, Enumerable and Location Datastreams supported.

Search datastream

2 Datastreams

Name	Pin	Data Type	Type Of Automation	Condition	Action
WaterLevel	V1	Integer	Number	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>
Distance	V2	String	Color	<input type="checkbox"/>	<input type="checkbox"/>

