UNIVERSITATEA TEHNICĂ DIN CLUJ-NAPOCA

Microcontroller based liquid level measuring

Nisioi Mihai-Florin

Third year student e_2031 Applied Electronics

Table of Contents

1 Introduction ... 3
- 1.1 Description ... 3
- 1.2 How the problem was solved? ... 3
- 1.3 New aspects ... 3

2 Theoretical background ... 4
- 2.1 Suitable conversion for the sensor ... 4
- 2.2 How to choose proper resistors for LEDs ... 4
- 2.3 Defining the liquid thresholds ... 4

3 Implementation ... 5
- 3.1 Hardware part-schematic and the total cost of it ... 5
- 3.2 Explanation of the code ... 5

4 Conclusions ... 9
- 4.1 Personal remarks ... 9
- 4.2 Bibliography ... 9

1 Introduction

1.1 Description

This project uses an ESP32 microcontroller, an ultrasonic sensor, and other external components to measure and monitor liquid levels in a container. The system provides real-time feedback on liquid levels using visual (LEDs) and auditory (buzzer) indicators and displays data on a Blynk 2.0 dashboard for remote monitoring.

1.2 How the problem was solved?

The problem of accurately monitoring liquid levels was addressed by leveraging the capabilities of an ultrasonic sensor to measure the distance from the sensor to the liquid's surface. By calculating the height of the liquid and its percentage level in the container, we provide users with actionable data. The ESP32, acting as the main controller, processes the measurements, controls LEDs and a buzzer, and transmits data to the Blynk app for remote visualization.

1.3 New aspects

This project lays the foundation for a versatile liquid monitoring system, but several enhancements can be implemented to improve functionality and efficiency. Potential future improvements include adding a battery backup for uninterrupted operation during power outages, integrating machine learning for predictive analysis of liquid usage, expanding support for multiple sensors to monitor several containers, and adopting alternative communication protocols like LoRa for greater range and reliability. Additionally, the Blynk interface could be enhanced with advanced analytics and real-time alerts for better user engagement.

2 Theoretical background

2.1 Suitable conversion for the sensor

The ultrasonic sensor calculates distance based on the time it takes for a sound wave to travel to the liquid surface and reflect back. The formula used is:

$$Distance\ (cm) = \frac{Time\ (microseconds) \times 0.0344}{2}$$

This ensures accurate measurements of the liquid's height.

2.2 How to choose proper resistors for LEDs

LED resistors were selected to limit the current and protect the LEDs. The resistance is calculated as:

$$R_{LED} = \frac{V_{supply-}V_{LED}}{I_{LED}}$$

For a 3.3V supply and an LED voltage drop of 2V (for the RED led) and 2.3V (for the GREEN led) at 20mA (both) , we should pick resistors close to the values below:

$$R_{RED} = \frac{3.3V - 2V}{20mA} = 65\Omega$$

$$R_{GREEN} = \frac{3.3V - 2.3V}{20mA} = 50\Omega$$

The BLUE led is the internal led of the ESP32 board, so, no resistor needed for limiting the current.

2.3 Defining the liquid thresholds

Full Level: Maximum measurable liquid height (24 cm).

Decent level: Between 7cm and 18cm height

Low Level Warning: Below 7 cm.These thresholds trigger specific LEDs and the buzzer to indicate the liquid state.

3 Implementation

3.1 Hardware part-schematic and the total cost of it

The system consists of:

ESP32 microcontroller for computation and communication. (9€)

Ultrasonic sensor (HC-SR04) to measure liquid level. (3€)

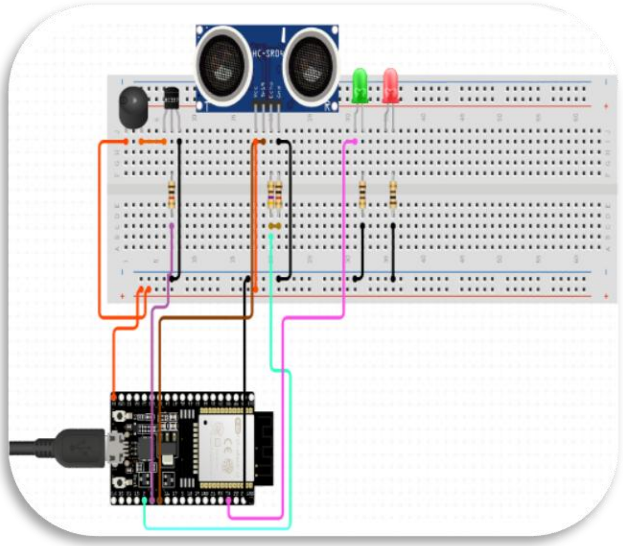
LEDs (Red, Green, and Blue) for visual indicators. (0.30€x2)

Buzzer for low-liquid alerts. (1€)

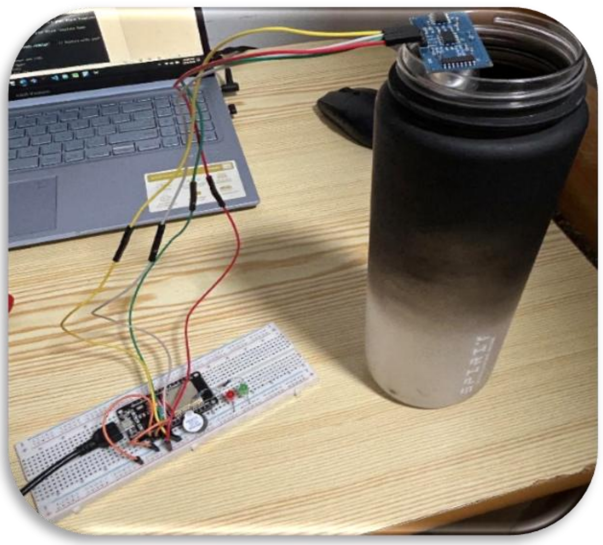
Two 220Ω resistors (0.10€x2)

Total cost: 13.8€ (68.53 lei)

Here is the schematic:



The actual implementation of it:



3.2 Explanation of the code

```
#define BLYNK_TEMPLATE_ID "TMPL42GlbKfxM"
#define BLYNK_TEMPLATE_NAME "fuel"
#define BLYNK_AUTH_TOKEN "DfHtZ_A1zihchuVXXhPFF4Ul-rE4KTgt"

#define TRIG_PIN 15
#define ECHO_PIN 4
#define RED_LED 18
```

```
#define GREEN_LED 23
#define BLUE_LED 2
#define BUZZER 5

#define MAX_HEIGHT 23
#define MIN_HEIGHT 2

char ssid[] = "room_329";
char pass[] = "toplita1984";

#include <WiFi.h>
#include <BlynkSimpleEsp32.h>

void setup() {
  Serial.begin(115200);
  pinMode(TRIG_PIN, OUTPUT);
  pinMode(ECHO_PIN, INPUT);
  pinMode(RED_LED, OUTPUT);
  pinMode(GREEN_LED, OUTPUT);
  pinMode(BLUE_LED, OUTPUT);
  pinMode(BUZZER, OUTPUT);

  digitalWrite(RED_LED, LOW);
  digitalWrite(GREEN_LED, LOW);
  digitalWrite(BLUE_LED, LOW);
  digitalWrite(BUZZER, LOW);

  WiFi.begin(ssid, pass);
  Serial.println("Connecting to Wi-Fi...");
  int attemptCount = 0;
  int maxAttempts = 30;

  while (WiFi.status() != WL_CONNECTED && attemptCount < maxAttempts) {
    delay(1000);
    Serial.print(".");
    attemptCount++;
  }

  if (WiFi.status() == WL_CONNECTED) {
    Serial.println("\nWi-Fi connected!");
    Blynk.begin(BLYNK_AUTH_TOKEN, ssid, pass);
  } else {
    Serial.println("\nWi-Fi connection failed. Check credentials.");
    while (true) {
      digitalWrite(RED_LED, HIGH);
      delay(1000);
      digitalWrite(RED_LED, LOW);
      delay(1000);
    }
  }
}

void loop() {
  Blynk.run();
  digitalWrite(TRIG_PIN, LOW);
  delayMicroseconds(2);
  digitalWrite(TRIG_PIN, HIGH);
  delayMicroseconds(10);
  digitalWrite(TRIG_PIN, LOW);

  long duration = pulseIn(ECHO_PIN, HIGH);

  if (duration == 0) {
    delay(500);
    return;
  }

  long distance = duration * 0.0344 / 2;
  float fuel_percentage = (1 - (distance - MIN_HEIGHT) / (float)(MAX_HEIGHT -
MIN_HEIGHT)) * 100;
  fuel_percentage = constrain(fuel_percentage, 0, 100);

  Blynk.virtualWrite(V0, fuel_percentage);
  Serial.print("Fuel Percentage: ");
  Serial.println(fuel_percentage);

  if (fuel_percentage <= 15) {
    digitalWrite(RED_LED, HIGH);
    digitalWrite(BUZZER, HIGH);
    delay(200);
    digitalWrite(BUZZER, LOW);
    delay(200);
    digitalWrite(GREEN_LED, LOW);
    digitalWrite(BLUE_LED, LOW);
  } else if (fuel_percentage <= 35) {
    digitalWrite(RED_LED, HIGH);
    digitalWrite(BUZZER, LOW);
    digitalWrite(GREEN_LED, LOW);
    digitalWrite(BLUE_LED, LOW);
  } else if (fuel_percentage <= 65) {
    digitalWrite(GREEN_LED, HIGH);
    digitalWrite(RED_LED, LOW);
    digitalWrite(BLUE_LED, LOW);
    digitalWrite(BUZZER, LOW);
  } else {
    digitalWrite(BLUE_LED, HIGH);
    digitalWrite(RED_LED, LOW);
    digitalWrite(GREEN_LED, LOW);
    digitalWrite(BUZZER, LOW);
  }

  delay(500);
}
```

The code initializes the ESP32 and establishes a Wi-Fi connection. It periodically triggers the ultrasonic sensor to measure the distance to the liquid surface, calculates the liquid level as a percentage, and sends this data to the Blynk dashboard.

Main Functions:

setup(): Initializes pins and connects to Wi-Fi and Blynk.

loop(): Continuously measures liquid levels, updates LEDs and buzzer states, and sends data to Blynk.

Visual alerts through LEDs based on liquid levels:

Green: good amount of liquid

Blue: decent amount of liquid

Red: low level of the liquid

Red+Buzzer: very low amount of liquid

4 Conclusions

4.1 Personal remarks

The project combines hardware and software elements to solve a real-world problem effectively. The integration of IoT capabilities with basic electronics allows for innovative and efficient monitoring solutions, making the development process both educational and fulfilling. The process was rewarding because it helped me to think for potential changes in the circuit, such that the objective was accomplished.

4.2 Bibliography

Blynk IoT Documentation.

https://docs.blynk.io/en

HC-SR04 Ultrasonic Sensor Datasheet.

https://www.alldatasheet.com/datasheet-pdf/download/1132204/ETC2/HCSR04.html

ESP32 Technical Reference Manual

http://www.ee.ic.ac.uk/pcheung/teaching/DE1_EE/Labs/esp32_technical_reference_manual_en.pdf