Chapter 1

INTRODUCTION

Water is an essential resource for life and economic development, making its effective management critical in addressing water scarcity and wastage.

Water scarcity is a growing threat on a global scale, demanding efficient management strategies. The United Nations estimates that by 2050, nearly 5 billion people will face water scarcity (UN-Water, 2023). This project presents a practical solution - an IOT-based water level monitoring system designed for diverse applications. This project presents the development of an IOT-based water level monitoring system utilizing an ESP32 microcontroller and the Blynk application platform. This system provides real-time water level data remotely, allowing for improved water management strategies. Imagine being able to check the water level in your tank from anywhere using your smartphone, or even receiving alerts when the water level reaches critically low or high points.

At the heart of the system lies the ESP32 microcontroller, a versatile unit programmed to collect sensor data and facilitate communication between components. It is equipped with dual-core processing power, built-in Wi-Fi, and Bluetooth connectivity, making it suitable for Internet of Things (IoT) applications. By integrating the ESP32 with ultrasonic sensors, a smart water level monitoring system can be developed, capable of sending real-time data to remote devices, generating alerts, and providing users with insights into water usage patterns.

Ultrasonic sensors are widely used in water level monitoring systems due to their simplicity, accuracy, and non-contact measurement capability. These sensors measure distance by emitting ultrasonic waves and calculating the time taken for the waves to return after hitting a surface, such as the water level. This makes them ideal for real-time monitoring of water levels in various applications, including household water tanks, agricultural irrigation systems, flood monitoring, and industrial reservoirs.

An OLED display, commonly the SSD1306 model, offers a user-friendly visual representation of the current water level within the tank. Additionally, LEDs can be integrated to provide status indications (e.g., green for normal water level, red for critically low level) while a buzzer can be used for audible alarms in case of critical water levels.

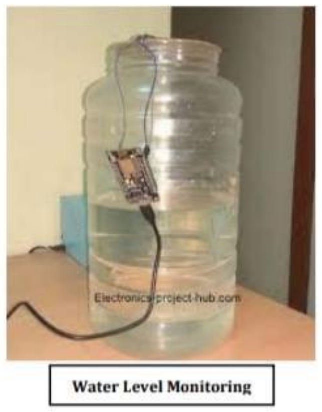


Figure 1: TRADITIONAL SYSTEM

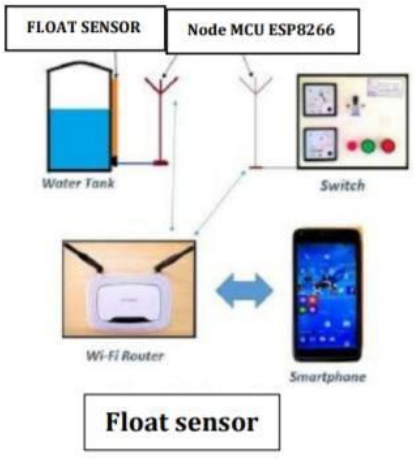


Figure 2: PROPOSED SYSTEM

1.1 Objectives:

Efficient water resource management is essential to address challenges such as water scarcity, wastage, and flooding. Traditional water level monitoring methods are often manual, time-consuming, and prone to inaccuracies. To overcome these limitations, automated systems using ultrasonic sensors and ESP32 microcontrollers have emerged as reliable, cost-effective solutions. This project explores the design and implementation of a water level monitoring system based on these technologies.

The system utilizes an ultrasonic sensor, such as the HC-SR04, to measure water levels through non-contact distance measurements. Ultrasonic waves emitted by the sensor are reflected back from the water surface, and the time taken for the echo to return is used to calculate the distance. The ESP32 microcontroller processes this data and converts it into meaningful water level readings. The ESP32's integrated Wi-Fi and Bluetooth capabilities enable real-time data transmission to cloud platforms, mobile applications, or web interfaces, facilitating remote monitoring and control.

Chapter 2

LITERATURE SURVEY

I. **Development a prototype of river water level monitoring system using ESP32 based on internet of things for flood mitigation.**

N A Pramono*, B A Purwandani , O Ghaisyani, F P P Mallisa, and F I Sofyan
Department of Physics, Universities Negeri Malang, Indonesia

Abstract. Flood is one of the disasters which often strike Indonesia. It is one of the disaster which can cause plenty of losses for the community that are not limited to material losses only, but casualties too. Along with the development of science and technology, flood mitigation can also be performed through the latest technologies which are more effective. One of the technologies which can be applied to help monitoring flood is river level monitoring system that is based on the Internet of Things. The purpose of this study is to create a river level monitoring system by using ESP32 based on the Internet of Things for flood mitigation. This system uses Ultrasonic Sensor and the monitoring results will be displayed on Blynk and Thing speak which are connected to the ESP32. The result is a prototype of a river level monitoring system, which es expected to be used for flood mitigation.

II. **Instrumentation for Water Level Monitoring System and It's Software Development. SMM Naidu, PS Oza, ON Wagh, Nitin Alzende, Prashant Ahire, ApuravDivekar, Omkar Papade, Virendra Mali, Yuvraj Khelkar, VM Diwate. Journal of Electrical Systems 20 (10s), 53-62, 2024**

The Water Level Tank Monitoring and Prediction System presented in this project harnesses advanced sensor technologies and predictive algorithms to address the critical issue of efficient water management, with depleting water resources and the growing demand for sustainable practices, real time monitoring of water levels in storage tanks becomes imperative. This system integrates ultrasonic sensors, microcontroller (ESP32), and data analysis techniques to monitor water levels in tanks accurately and predict future levels based on historical data. The system begins by continuously collecting real-time data from ultrasonic sensors placed inside water tanks. These sensors measure the water level accurately and transmit the data to a central microcontroller unit. The microcontroller processes the incoming data and sends it to a database for storage and analysis. Using sophisticated algorithms, historical water usage patterns are analyzed to predict future water levels. Machine learning techniques, such as regression analysis and neural networks, are employed to forecast water consumption trends, enabling proactive decision-making. The Water Level Tank Monitoring and Prediction System offers several significant advantages. Firstly, it provides real-time monitoring, ensuring that water levels are constantly observed. This real-time data is crucial for immediate action in case of leakages or sudden increases in demand. Secondly, the predictive analysis facilitates informed planning by anticipating water needs based on historical patterns. This proactive approach helps in optimizing water distribution, minimizing wastage, and ensuring a consistent water supply. In addition to its practical applications for households, industries, and municipalities, the system contributes significantly to water conservation efforts. By promoting efficient use of water resources and reducing unnecessary wastage, it aligns with the global goal of sustainable water management.

Chapter 3

CIRCUIT DIAGRAM

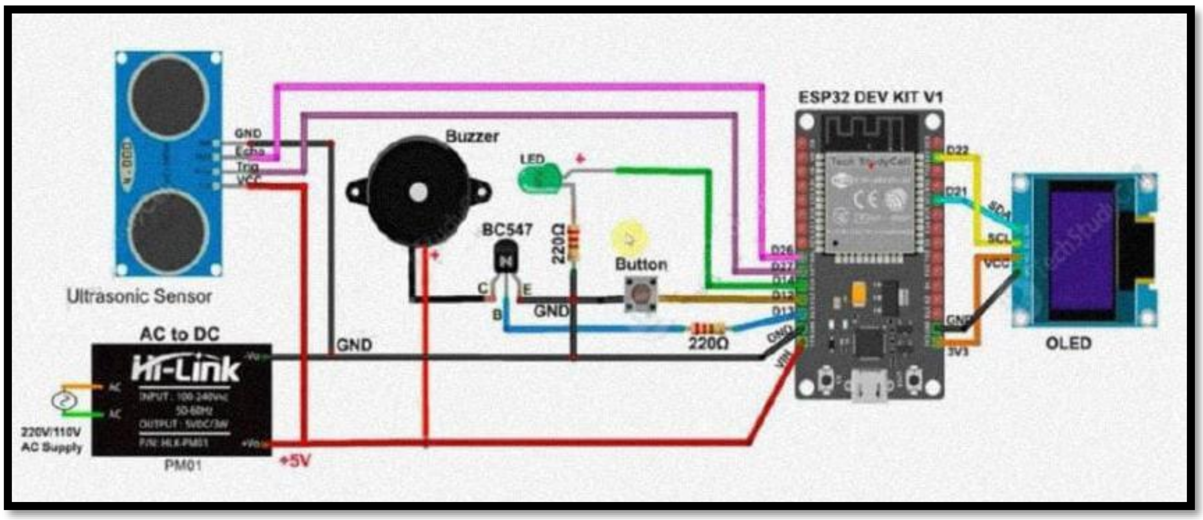


Figure 3.1: circuit diagram

3.1System operation and user interface

The system operates efficiently through a series of steps:

- **Data Collection:** The ESP32 microcontroller triggers the ultrasonic sensor at regular intervals. The sensor transmits a sound wave and measures the time it takes for the echo to return. Based on this time, the distance to the water surface is calculated.
- **Data Processing:** The ESP32 then processes this data and converts it into a water level measurement.
- **Local Monitoring:** This information is subsequently displayed on the OLED display for local monitoring, providing users with immediate visual feedback.
- **Remote Monitoring and Notifications:** The system leverages the power of IoT by utilizing a cloud platform like Blynk. The ESP32 transmits the water level data to the Blynk cloud, enabling remote monitoring and data storage. Users can access the Blynk app on their smartphones or tablets from anywhere with an internet connection to view real-time water level data and historical trends.

The Blynk app also serves as a communication channel for the notification system. If the water level falls below a predefined threshold set by the user, an alert is sent to the user's smartphone, prompting them to take necessary actions to prevent water wastage overflows.

CHAPTER 4

HARDWARE COMPONENTS

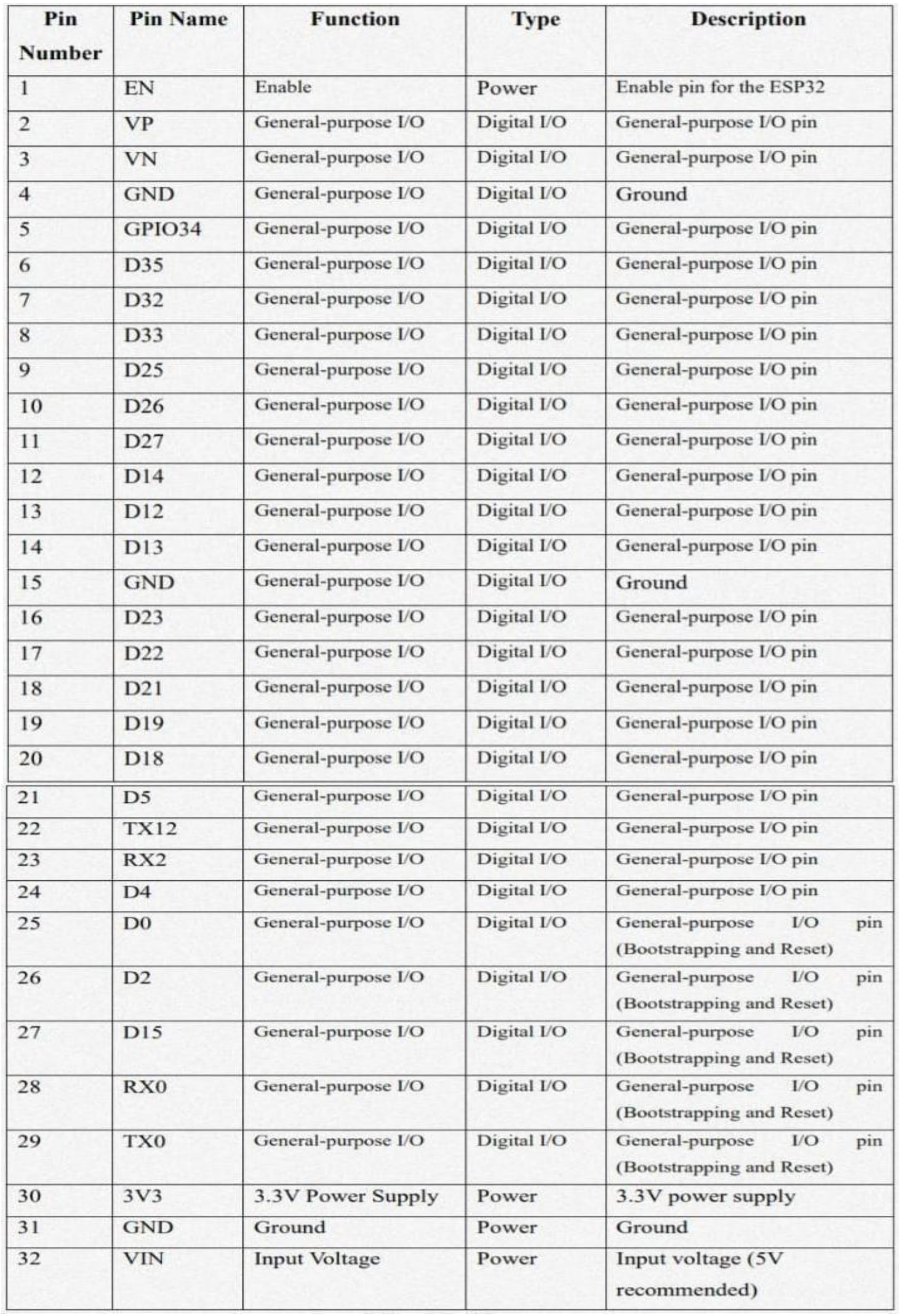
1. ESP32 DEV KIT V1
2. SR04M waterproof ultrasonic sensor OR HC-SR04 sensor
3. 0.96" OLED Display
4. 220-ohm 0.25watt Resistors - 2 no
5. BC547 NPN Transistor
6. LED 5mm - 1no
7. 2-pin Push Button
8. 2-pin Terminal connectors (3 no)
9. 5V DC Buzzer
10. AC to DC converter PM01 5V (Optional)

4.1 Hardware description

I. ESP32 DEVkit

For novice individuals embarking on microcontroller projects, we will be employing the ESP32, which is one of the leading microprocessor boards available in the industry. The ESP32, created by Systems, is well regarded for its adaptability and strong performance, making it an exceptional option for educational and industrial uses. Due to its comprehensive documentation and broad adoption in the last 10 years, the ESP32 provides a dependable framework for acquiring real-time data from sensors and sending it to mobile devices.

The ESP32, being an IoT-enabled device, allows for effortless data interchange with other devices over networks, enabling the generation of data logs, visualization graphs, event triggers, and automation procedures. Therefore, there is no requirement for supplementary components like the Bolt Wi-Fi or Node MCU ESP8266, as the ESP32 is capable of independently managing data transfer with high efficiency. These features make the ESP32 well-suited for a variety of project needs.



Pin Number	Pin Name	Function	Type	Description
1	EN	Enable	Power	Enable pin for the ESP32
2	VP	General-purpose I/O	Digital I/O	General-purpose I/O pin
3	VN	General-purpose I/O	Digital I/O	General-purpose I/O pin
4	GND	General-purpose I/O	Digital I/O	Ground
5	GPIO34	General-purpose I/O	Digital I/O	General-purpose I/O pin
6	D35	General-purpose I/O	Digital I/O	General-purpose I/O pin
7	D32	General-purpose I/O	Digital I/O	General-purpose I/O pin
8	D33	General-purpose I/O	Digital I/O	General-purpose I/O pin
9	D25	General-purpose I/O	Digital I/O	General-purpose I/O pin
10	D26	General-purpose I/O	Digital I/O	General-purpose I/O pin
11	D27	General-purpose I/O	Digital I/O	General-purpose I/O pin
12	D14	General-purpose I/O	Digital I/O	General-purpose I/O pin
13	D12	General-purpose I/O	Digital I/O	General-purpose I/O pin
14	D13	General-purpose I/O	Digital I/O	General-purpose I/O pin
15	GND	General-purpose I/O	Digital I/O	Ground
16	D23	General-purpose I/O	Digital I/O	General-purpose I/O pin
17	D22	General-purpose I/O	Digital I/O	General-purpose I/O pin
18	D21	General-purpose I/O	Digital I/O	General-purpose I/O pin
19	D19	General-purpose I/O	Digital I/O	General-purpose I/O pin
20	D18	General-purpose I/O	Digital I/O	General-purpose I/O pin
21	D5	General-purpose I/O	Digital I/O	General-purpose I/O pin
22	TX12	General-purpose I/O	Digital I/O	General-purpose I/O pin
23	RX2	General-purpose I/O	Digital I/O	General-purpose I/O pin
24	D4	General-purpose I/O	Digital I/O	General-purpose I/O pin
25	D0	General-purpose I/O	Digital I/O	General-purpose I/O pin (Bootstrapping and Reset)
26	D2	General-purpose I/O	Digital I/O	General-purpose I/O pin (Bootstrapping and Reset)
27	D15	General-purpose I/O	Digital I/O	General-purpose I/O pin (Bootstrapping and Reset)
28	RX0	General-purpose I/O	Digital I/O	General-purpose I/O pin (Bootstrapping and Reset)
29	TX0	General-purpose I/O	Digital I/O	General-purpose I/O pin (Bootstrapping and Reset)
30	3V3	3.3V Power Supply	Power	3.3V power supply
31	GND	Ground	Power	Ground
32	VIN	Input Voltage	Power	Input voltage (5V recommended)

Table 4.1: Pin configuration of ESP32

II. **HCSR04**

The HC-SR04 Ultrasonic Sensor employs a pair of ultrasonic transducers to gauge distance. The device functions by generating ultrasonic waves into the surrounding environment and subsequently measuring the duration it takes for these waves to reflect. One of the transducers acts as a transmitter, transforming electrical signals into ultrasonic sound pulses with a frequency of 40KHz. The second transducer acts as a receiver, detecting the reflected pulses. When the TRIGGER pin is activated, the transmitter releases eight sequences of ultrasonic pulses. While these pulses travel through the air, the receiver, which is attached to the ECHO pin, remains in a state of readiness to detect any signals that bounce back. If a signal is not received within a duration of 38 milliseconds, the receiver pin will transition to a low state, indicating the lack of an object in the sensor's vicinity. In contrast, when a signal is detected, the ECHO pin decreases in voltage, resulting in the generation of a pulse with a width that varies between 150µs and 25ms. The length of this pulse is directly proportional to the distance separating the sensor from the observed object. The distance can be determined by multiplying the speed of the signal by the time it takes to reflect, according to the formula: Distance = Speed x Time.

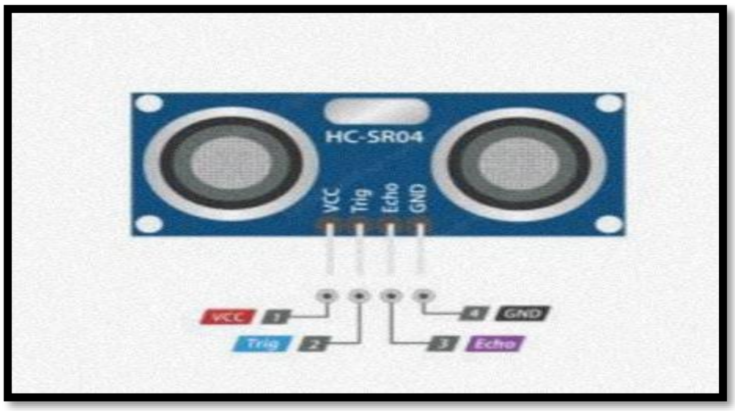


Fig 4.1: HCSR04 Sensor

The sensor can detect distances ranging from 2cm to 400cm, with a precision of 3mm. It will be connected to the ESP-32 using the following pin configuration:

- VCC pin connected to VIN pin of ESP-32
- Trig pin connected to pin 19
- Echo pin connected to pin 18
- GND pin of the sensor to GND pin of the ESP-32

III. BC547 NPN transistor

Connect the emitter of the BC547 transistor (Q1) to ground (GND). The collector of the transistor is connected to one terminal of the LED (LED1). The other terminal of the LED is connected to a resistor (R1, 1kΩ), and the other end of the resistor is connected to +5V power supply 0.96 OLED

- Displays real-time water level information.
- Small, energy-efficient display (commonly 128x64 pixels).
- Communicates with ESP32 using I2C or SPI protocol.

IV. 5V DC Buzzer

Acts as an alarm to notify the user about specific water level conditions (e.g., high, low, or critical water levels).

- The ESP32 controls the buzzer via a GPIO pin.
- The buzzer emits a sound when it receives a HIGH signal from the ESP32.
- The buzzer is triggered when the water level reaches predefined thresholds:
- Low Level Alert: Warns that the water level is too low, indicating the need for a refill.
- High Level Alert: Alerts to prevent overflow when the tank is almost full.
- Critical Level Alert: Sounds if the tank is empty or about to overflow.

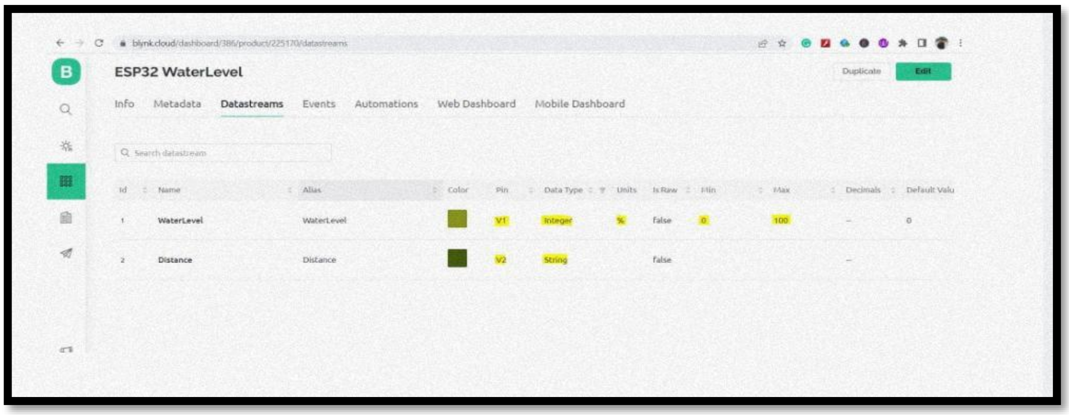
4.2 Create Blynk Template

During creating the template, I selected ESP32 as the hardware and the connection type as WiFi.

4.3 Create Datastreams in Blynk Cloud

In this template, we have created first Datastreams (Pin: V1, Datatype: Integer, Min Value: 0, Max Value: 100) to show the water level in tank in percentage.

Second Datastream (Pin: V2, Datatype: String) will show the distance between sensor and water level in cm.



Id	Name	Alias	Color	Pin	Data Type	Units	Is Raw	Min	Max	Decimals	Default Valu
1	WaterLevel	WaterLevel	🟩	V1	Integer	%	false	0	100	–	0
2	Distance	Distance	🟩	V2	String		false			–	

Fig 4.2: Example of Datastream in BLYNK

4.4 Create Web Dashboard in Blynk Cloud

water level sensor esp32 P18

After that, click and drag 1 Gauge widget, and 1 Level widget, and select the related Datastreams for each widget.

Then click on “Save” to save the template.

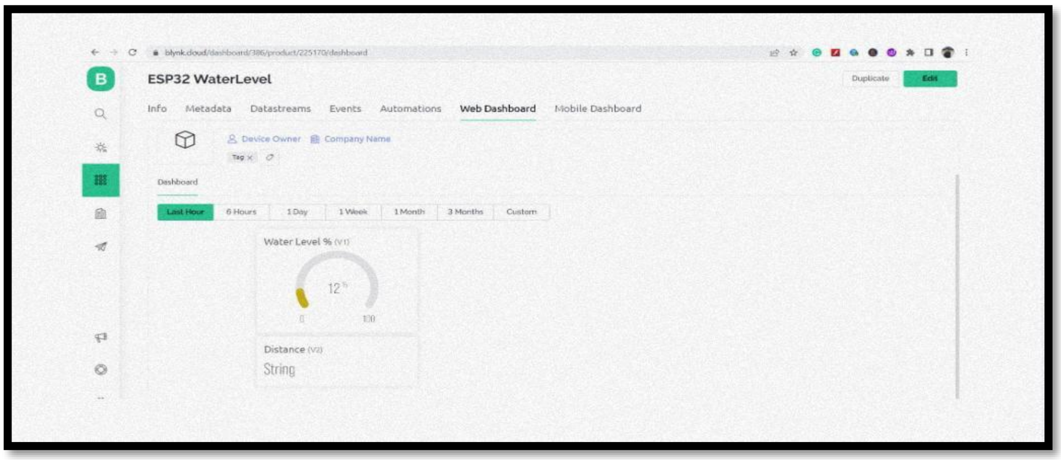


Fig 4.3: Example for web dashboard in BLYNK

4.5 Measurement of Water Level Sensor

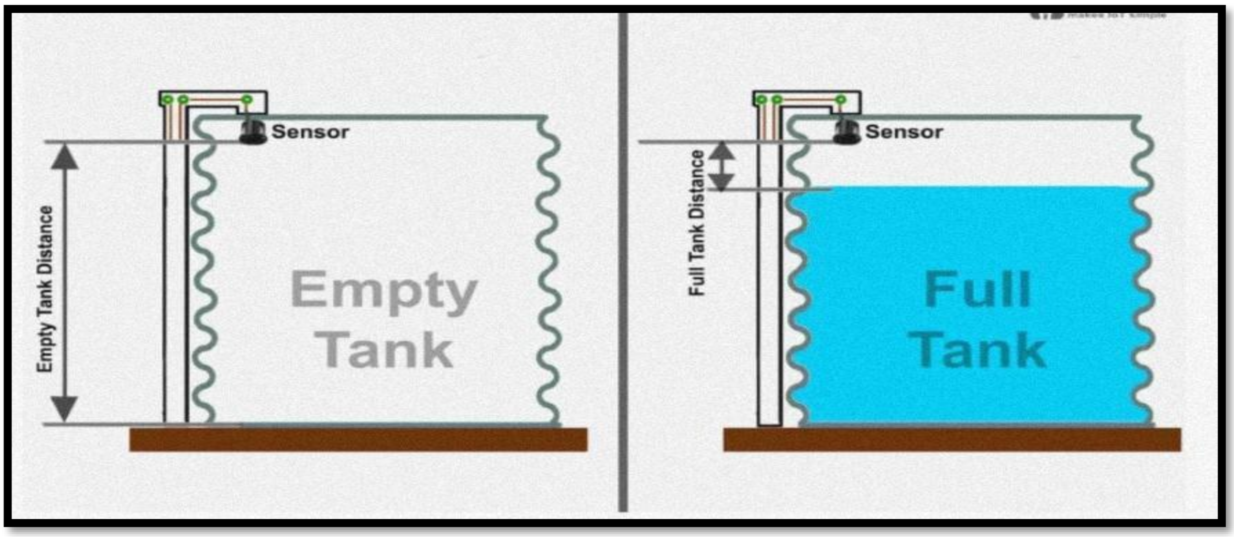


Fig 4.4: Measurement of water tanks for calculation

4.6 Install the Water level Sensor

The ESP32 will only calculate the water level if the measured distance is between empty tank distance and the full tank distance. After doing these changes, please upload the code to ESP32.

Install the Water Level Sensor circuit

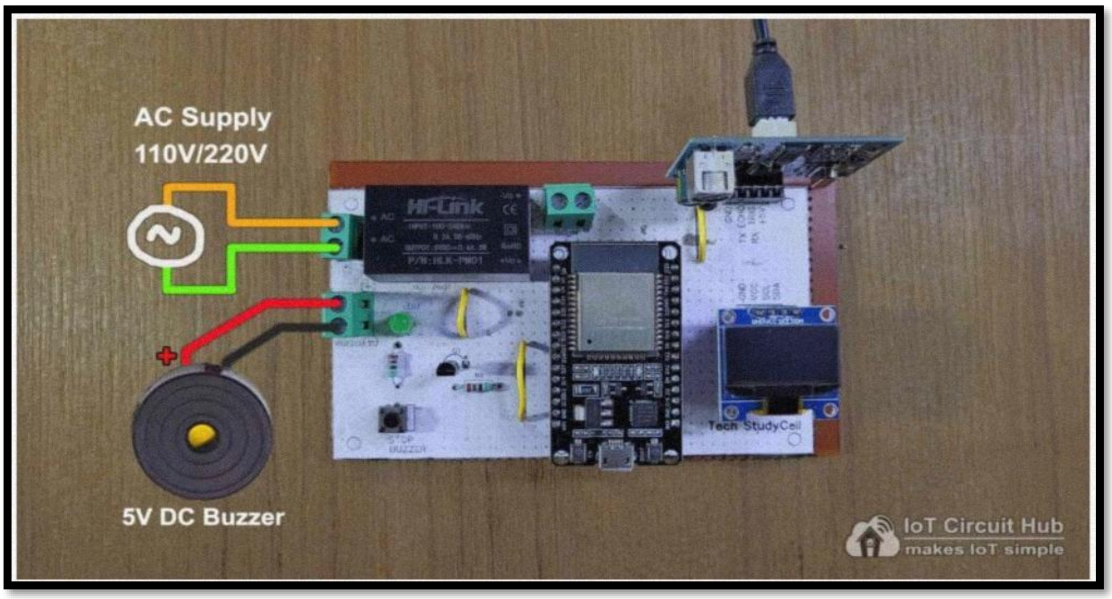


Fig 4.5: Circuit on PCB

CHAPTER 5

EXPERIMENTAL SETUP

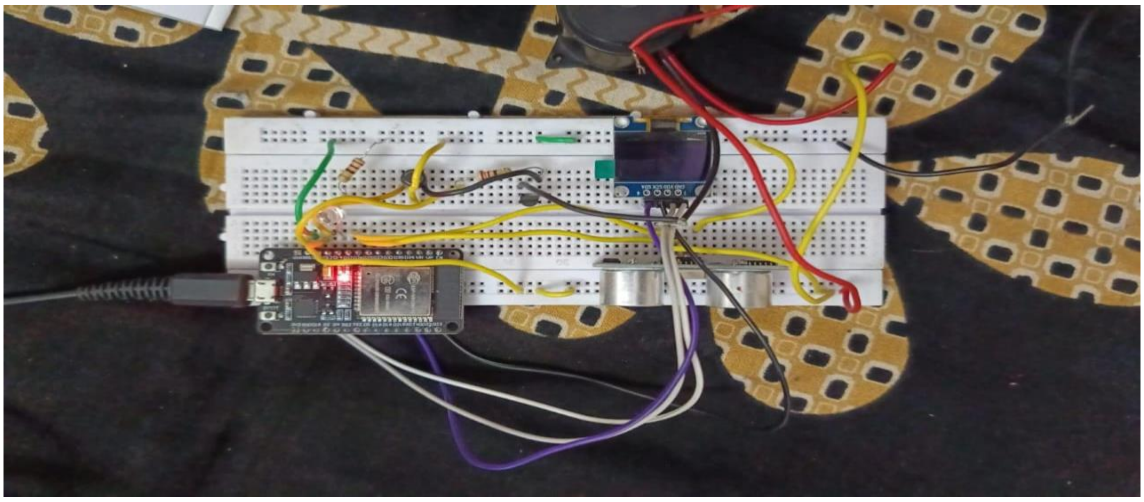


Fig 5.1: Prototype

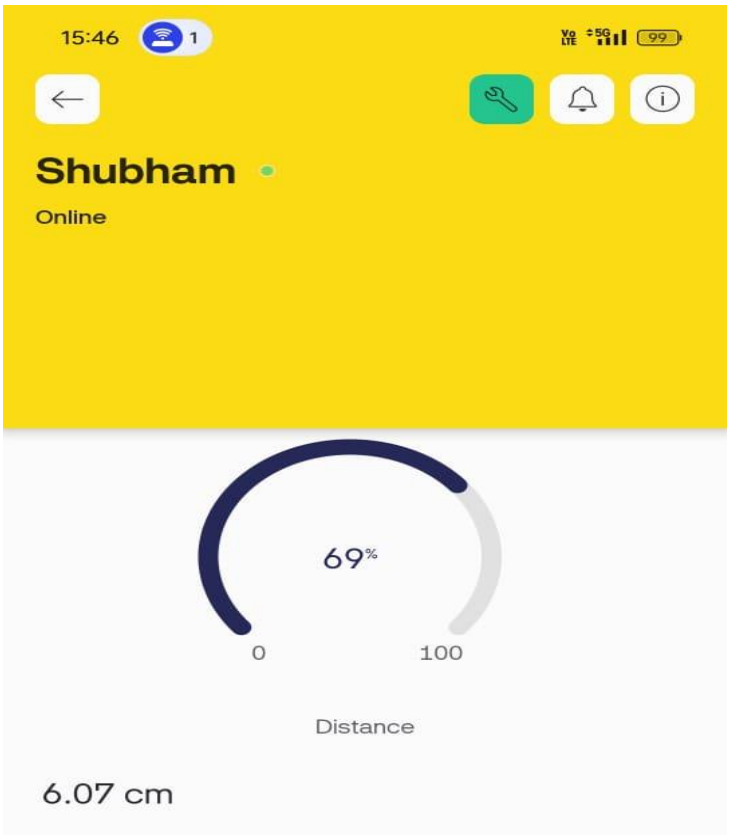


Fig 5.2: Prototype of Result Analysis

CHAPTER 6
RESULT AND ANALYSIS

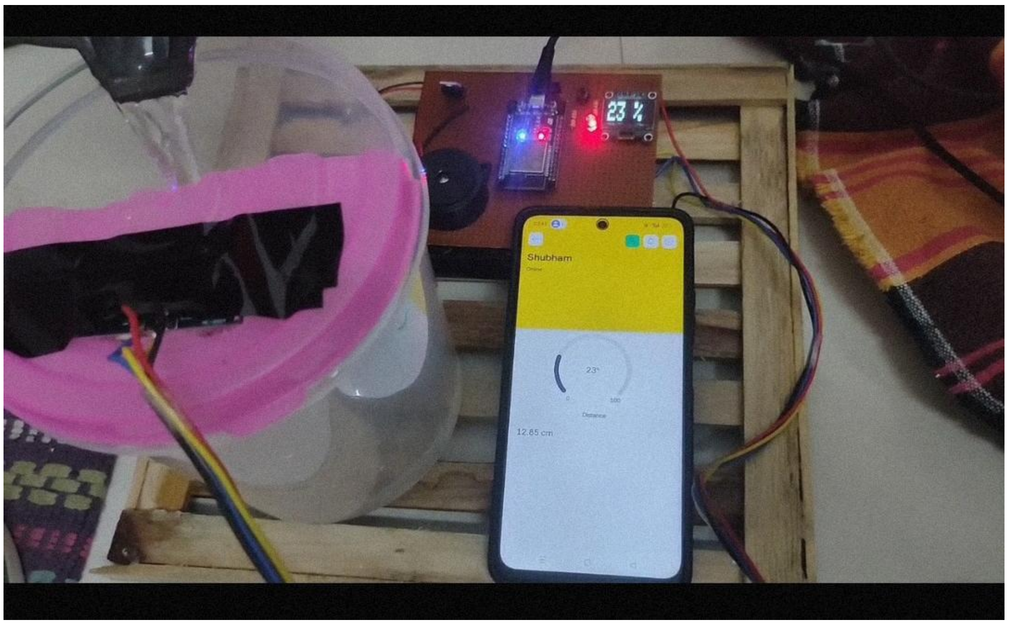


Fig 6.1: Final Outtput

The ultrasonic sensor provides an accurate measurement of the distance to the water surface, with a typical error margin of ±1 cm.
The ESP32 converts this distance into a corresponding water level percentage or depth value.
A buzzer triggers an alert for critical water levels (low or high), ensuring timely action is taken.

The system is compatible with IoT platforms for remote monitoring, utilizing the ESP32's Wi-Fi/Bluetooth capabilities.

6.1 Analysis of system performance

- Efficiency: The system continuously monitors the water level without manual intervention.
  Low power consumption due to efficient components like the ESP32 and OLED display.
- Sensor Performance: The HC-SR04 accurately measures distances within 2 cm to 400 cm. Tanks taller than this range require an alternative sensor.
- Alert Mechanism: Threshold Triggers: Alerts for low and high water levels ensure proactive management of water resources.

CHAPTER 7

ADVANTAGES AND DISADVANTAGES

Advantages

- Real-time Data and Informed Decisions: Users can make informed decisions regarding water usage, minimizing wastage and optimizing water management practices.
- Early Warnings and Prevention: Timely notifications from the system prevent potential overflows or critically low water levels, protecting against water damage and ensuring a consistent water supply.
- Sustainable Water Management: The system promotes responsible water consumption, contributing to environmental conservation by reducing water waste.
- Reduced Costs: By preventing water leaks and overflows, the system can potentially lead to reduced water bills for users.

Disadvantages

- Limited Range: The HC-SR04 has a maximum effective range of 400 cm. Tanks taller than this may require alternative sensors with longer ranges.
- Environmental Sensitivity: Performance can be affected by:
  Humidity: May interfere with the ultrasonic waves.
  Temperature: Speed of sound varies with temperature, slightly affecting accuracy.
  Obstructions: Obstacles inside the tank can distort or block the ultrasonic waves.
- Sensor Orientation: Proper alignment of the HC-SR04 is critical. If misaligned, the sensor may fail to detect echoes correctly.
- Material Limitations: Cannot accurately measure reflective or highly absorbent liquids (e.g., oils, acids).
- Limited Durability: The HC-SR04 is not water-resistant or waterproof, requiring protection from moisture or splashing in humid environments.

Chapter 8

APPLICATIONS

- **Home Applications:** A water level monitoring application tracks the real-time water level in home tanks. It provides notifications for low water levels or when the tank is full, preventing shortages or overflows. The app displays usage trends through detailed analytics. It can integrate with pumps for automated water management. Users can monitor multiple tanks remotely via a smartphone or web interface. The application enhances water conservation and efficient resource management at home.
- **Domestic Applications**

  Overhead Water Tanks: Monitors the water level in residential tanks to prevent overflows or water shortages, can trigger pumps automatically when water levels drop below a certain threshold.
  Underground Water Storage: Ensures optimal water levels in underground tanks for domestic use.
- **Industrial Applications**

  Industrial Water Tanks: Tracks water levels in large storage tanks to maintain a consistent water supply for manufacturing processes.
  Chemical Storage Tanks: Monitors liquid levels in non-reactive chemical tanks (e.g., water-based solutions).
  Boilers and Cooling Systems: Ensures appropriate water levels in boilers or cooling systems to prevent damage due to overheating or dryness.
- **Agricultural Applications**

  Irrigation Systems: Monitors water levels in reservoirs or storage tanks used for irrigation. Ensures timely water supply to crops and prevents wastage.
  Aquaculture: Maintains proper water levels in fish or shrimp farming ponds. Rainwater Harvesting Systems: Tracks the levels in storage tanks used for rainwater harvesting.

Chapter 9

CONCLUSION

This project presented the design of an IoT-based water level monitoring system using an ESP32 microcontroller and the Blynk application. The system offers real-time remote monitoring capabilities through a combination of the ESP32's processing power, Wi-Fi connectivity, and the Blynk app's user interface features. Users can choose between ultrasonic sensors for distance-based water level calculation or pre-built water level sensors for simpler integration. This design provides a versatile and customizable solution for various water management applications, promoting efficient monitoring and potential automation based on water level readings.

REFERENCE

1. Remote Water Level Monitoring System":
   Studies on remote monitoring systems using IoT, sensors, and communication technologies (e.g., GSM, LoRa, or Wi-Fi).
   - Example: IEEE papers on water level monitoring using ultrasonic or pressure sensors.
2. "Automated Water Level Monitoring Using IoT":
   Focuses on IoT-enabled systems for real-time tracking.
   Example: Articles in journals like *Sensors* or *Environmental Monitoring and Assessment*.
3. Hydrological Studies:
   Books and manuals on hydrological monitoring methods, such as *"Handbook of Hydrology"* by David Maidment.

APPENDIX- CODE:

```
#define BLYNK_TEMPLATE_ID "TMPL68xmtqpkP"
#define BLYNK_TEMPLATE_NAME "ESP32 WaterLevel"
#define BLYNK_AUTH_TOKEN "yFR6zDpl1KVUAKVWRI14Xnp77xDVnnjJ"
char ssid[] = "V";
char pass[] = "ZZZZZZ";
int emptyTankDistance = 30;
int fullTankDistance = 5;
int triggerPer = 10;
int thresholdPercentage = 20;
#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 ButtonPin1 12
#define BuzzerPin 13
#define GreenLed 14
#define VPIN_BUTTON_1 V1
#define VPIN_BUTTON_2 V2
#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;
#define TRIGPIN 27
#define ECHOPIN 26
#define wifiLed 2
#define ButtonPin1 12
#define BuzzerPin 13
#define GreenLed 14
#define VPIN BUTTON 1 V1
#define VPIN BUTTON 2 V2
#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;
char auth[] = BLYNK_AUTH_TOKEN;
ButtonConfig config1;
AceButton button1(@config1);
void handleEvent1(AceButton* button1, uint8_t eventType, uint8_t buttonState) {
  // Handle button events here
}
BlynkTimer timer;
void checkBlynkStatus() {
  bool isConnected = Blynk.connected();
  if (!isConnected) {
    digitalWrite(wifiLed, LOW);
  } else {
    digitalWrite(wifiLed, HIGH);
  }
```

```
}
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() {
   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);
      displayData(waterLevelPer);
      Blynk.virtualWrite(VPIN_BUTTON_1, waterLevelPer);
      Blynk.virtualWrite(VPIN_BUTTON_2, (String(distance) + " cm"));
      Serial.print("Distance: ");
   }
}
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(100);
  }
  // ... (some code omitted)
}
void setup() {
  // ... (some code omitted)
  pinMode(ECHOPIN, INPUT);
  pinMode(TRIGPIN, OUTPUT);
  pinMode(wifiLed, OUTPUT);
  pinMode(BuzzerPin, OUTPUT);
  pinMode(GreenLed, OUTPUT);
}
pinMode(ButtonPin1, INPUT_PULLUP);
digitalWrite(wifiLed, LOW);
digitalWrite(GreenLed, LOW);
digitalWrite(BuzzerPin, LOW);
config1.setEventHandler(handleEvent1);
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);
Blynk.config(auth);
```

```
delay(1000);
void loop() {
  measureDistance();
  Blynk.run();
  timer.run();
  button1.check();
}
void handleEvent1(AceButton* button, uint8_t eventType, uint8_t buttonState) {
  switch (eventType) {
    // ... (switch case omitted)
  }
}
case AceButton::kEventReleased:
  digitalWrite(BuzzerPin, LOW);
  toggleBuzzer = LOW;
  break;
```