# ABSTRACT

This project explores the development of an IoT-based water level monitoring system that utilizes the ESP32 microcontroller, an ultrasonic sensor, and the Blynk IoT platform. The system is designed to provide real-time data on water levels in a storage tank, offering both local and remote alerts for critical levels. When the water level falls below a specified threshold, an LED indicator lights up, and an audible alarm is triggered. Additionally, a push notification is sent to the user’s smartphone via the Blynk app, ensuring that the user is informed promptly, regardless of their location. A similar alert system is activated when the tank reaches its full capacity, preventing potential overflow. The inclusion of a push button allows the user to manually stop the alarm, enhancing user interaction and control.

The integration of the Blynk IoT platform with the ESP32 microcontroller not only facilitates remote monitoring but also demonstrates the practical application of IoT in everyday situations. This system offers a cost-effective and scalable solution for water management, which can be easily implemented in both residential and industrial environments. Through a detailed explanation of the system's design, components, and operation, this project serves as a guide for those interested in applying IoT technology to real-world problems, enabling more efficient resource management and enhancing user conven

# TABLE OF CONTENTS

**PAGE NO**

|  |  |  |
| --- | --- | --- |
|  | Abstract | iv |
|  | List of figures | vii |
| **CHAPTER 1** | **INTRODUCTION** | **1** |
|  | 1.1 Introduction | 2 |
| **CHAPTER 2** | **LITERATURE SURVEY** | **3** |
|  | 2.1 Literature Survey | 4 |
| 2.2 Problem Statement | 5 |
| 2.3 Objective | 6 |
| 2.4 System Overview | 6 |
| **CHAPTER 3** | **METHODOLOGY** | **7** |
|  | 3.1 Methodology | 8 |
| **CHAPTER 4** | **HARDWARE & SOFTWARE**  **DESCRIPTION** | **10** |
|  | 4.1 Hardware Description | 11 |
|  | 4.2 Software Description | 16 |
| **CHAPTER 5** | **IMPLEMENTATION** | **18** |
|  | 5.1 Flow chart | 19 |
|  | 5.2 Circuit & Block diagram | 20 |
| **CHAPTER 6** | **RESULT AND DISCUSSIONS** | **21** |
|  | 6.1 Result and discussions | 22 |
| **CHAPTER 7** | **CONCLSION AND FUTURE SCOPE** | **24** |
|  | 7.1 Conclusion | 24 |
|  | 7.2 Future scope | 25 |
| **REFERENCES** |  | **27** |
| **APPENDIX** |  | **29** |

**LIST OF FIGURES**

|  |  |
| --- | --- |
| Fig 4.1 Node MCU | 12 |
| Fig 4.2 HC-SR04 Ultrasonic Sensor | 12 |
| Fig 4.3 Buzzer | 13 |
| Fig 4.4 Push Button | 13 |
| Fig 4.5 220-ohm Resistor | 14 |
| Fig 4.6 LED | 14 |
| Fig 4.7 Breadboard | 15 |
| Fig 4.8 Jumper Wires | 15 |
| Fig 4.9 Power Supply | 16 |
| Fig 4.10 Arduino IDE Fig 4.11 Blynk App Fig 5.1 Flow chart  Fig 5.2 Circuit Diagram  Fig 5.3 Block Diagram | 17  17  19  20  20 |
| Fig 6.1 Prototype | 23 |
| Fig 7.1 Output | 26 |

**CHAPTER-1 INTRODUCTION**

**1.1 Introduction**

# CHAPTER-1

**INTRODUCTION**

Water is a vital resource, and efficient water management is crucial for ensuring its sustainability. Traditional methods of monitoring water levels in tanks are often manual, time- consuming, and prone to human error. With the advent of Internet of Things (IoT) technology, it is now possible to automate and enhance the accuracy of water level monitoring systems. This project explores the development of a smart water level monitoring system using an ultrasonic sensor, ESP32 microcontroller, and the Blynk IoT platform.

The proposed system leverages the capabilities of the ultrasonic sensor to provide precise measurements of water levels. The ESP32 microcontroller processes these measurements and communicates with the Blynk server over a Wi-Fi connection to ensure real-time updates are available to the user. The Blynk app serves as the interface through which users can monitor water levels, receive alerts, and interact with the system. By using IoT technology, this project aims to reduce the labour, and errors associated with traditional water level monitoring methods.

This system can be beneficial in various settings, including residential buildings, agricultural fields, and industrial sites. It helps homeowners manage water usage more effectively, assists farmers in monitoring irrigation tanks, and aids industrial sites in preventing operational disruptions due to insufficient water levels. The integration of Wi-Fi connectivity ensures that users can access the water level data remotely through the Blynk app, providing flexibility and convenience.

Furthermore, the system includes safety features such as alarms for low and high-water levels, ensuring immediate action can be taken to prevent potential issues. This innovative approach to water management highlights the potential of IoT technology in addressing everyday challenges and promoting efficient resource utilization. The project serves as a practical example of how modern technology can enhance traditional processes, making them more reliable and user-friendly.

**CHAPTER-2**

**LITERATURE SURVEY**

**CHAPTER-2**

**LITERATURE SURVEY**

* 1. **Literature survey**

**Priya J, Sailusha Chekuri (2017):** The paper presents an IoT-based water level monitoring system that uses sensors to measure water levels and transmits the data over the internet for real-time monitoring.

**Madhurima Santra et al. (2017)**: This study discusses a smart wireless system for monitoring water levels and controlling pumps automatically, aimed at preventing overflow and ensuring efficient water usage.

**Beza Negash Getu and Hussain A. Attia (2016)**: The paper introduces an automatic water level sensor and controller system, focusing on maintaining water levels in reservoirs by automatically activating pumps.

**Asaad Ahmed Mohammed ahmed Eltaieb and Zhang Jian Min (2015)**: This research explores an automatic water level control system that utilizes sensors to maintain desired water levels in a tank, minimizing human intervention.

**K. Santhosh Kumar et al. (2015)**: The authors propose a microcontroller-based system for automatic water level control, which operates pumps based on sensor data to maintain water levels without manual oversight.

**Sanam Pudasaini et al. (2014)**: The paper presents an automatic water level controller with SMS notification, which alerts users via text message when water levels reach certain thresholds, ensuring timely intervention.

**Md. Moyeed Abrar & Rajendra R. Patil (2014):** This paper presents a logic gate-based system for automatically controlling water levels, utilizing basic digital electronics for automation.

**Sanam Pudasaini et al. (2014):** The study proposes an automatic water level controller with SMS notifications, enabling remote monitoring and control.

**Muktha Shankari et al. (2013):** This research discusses a wireless water level control system using radio frequency communication to automate the process without physical connections.

**S. B. Jagadal & S. V. Halse (2013):** The paper describes a control system based on the 8051 microcontrollers for managing multiple water tanks.

**S. M. Khaled Reza et al. (2010):** The authors focus on designing and implementing an automated water level sensing and controlling system using a microcontroller.

**Pan Lingfeng et al. (2011):** The study deals with the design and realization of an intelligent mixed water valve control system for precise water temperature regulation.

**Tingjun Wang & Li Gu (2009):** This paper introduces an electronic water-saving system for toilet bowls using electromagnetic valve control for efficient water usage.

**Jiang Wei (2010):** The research discusses the use of fuzzy logic for intelligent control of water tanks in buildings, focusing on optimizing water usage and system efficiency.

### Problem Statement

Effective water management is essential for various applications, including residential, agricultural, and industrial settings. Traditional methods of monitoring water levels in tanks rely heavily on manual checks, which are time-consuming, prone to human error, and often fail to provide real-time data. This lack of timely information can lead to water wastage due to overflow or inadequate water supply during critical times, impacting resource efficiency and operational effectiveness. The challenge lies in developing a reliable, automated system that can accurately monitor water levels and provide real-time alerts and notifications to users, ensuring optimal water usage and preventing wastage. This project aims to address these issues by leveraging IoT technology to create a smart water level monitoring system that is efficient, accurate, and user-friendly.

### Objectives

The primary objectives of the Automatic Water Tank Overflow Prevention and Alert System are:

* + - **Develop an Accurate Water Level Monitoring System:** Utilize an ultrasonic sensor to provide precise measurements of water levels in the tank.
    - **Implement Real-Time Data Communication:** Use the ESP32 microcontroller to process sensor data and communicate with the Blynk server over a Wi-Fi connection for real-time monitoring.
    - **Provide User-Friendly Interface:** Enable users to monitor water levels, receive alerts, and interact with the system through the Blynk app on their smartphones.
    - **Automate Alerts and Notifications:** Ensure that users are notified immediately when water levels are too low or too high to prevent overflow and ensure an adequate water supply.
    - **Include Safety Features:** Integrate an LED indicator that glows when the water level is low and an alarm system that triggers in both low and high-water level conditions.
    - **Enhance Remote Accessibility:** Allow users to access water level data remotely via the Blynk app, providing flexibility and convenience.
    - **Ensure Energy Efficiency:** Design the system to operate continuously with minimal power consumption.

### System Overview

The system comprises several key components, including water level sensors, a microcontroller, a relay module, and an alert mechanism. The water level sensors are installed in the tank to measure the water level continuously. These sensors send data to the microcontroller, which processes the information and determines whether the water supply needs to be turned on or off. The relay module, controlled by the microcontroller, regulates the water pump's operation based on the water level data.

When the water level reaches a predefined threshold, indicating that the tank is full, the microcontroller activates the relay to turn off the water pump, preventing overflow. Similarly, when the water level drops below a certain level, the microcontroller can turn on the water pump to refill the tank.

**CHAPTER-3**

**METHODOLOGY**

# CHAPTER-3 METHODOLOGY

### Methodology

The Automatic Water Tank Overflow Prevention and Alert System is designed to address the common issue of water wastage due to overflowing tanks. The system's methodology involves a blend of sensor technology, microcontroller programming, and communication modules to ensure efficient water management and timely alerts. The core component of the system is a set of water level sensors, typically ultrasonic or float-based, installed at different heights within the tank. These sensors continuously monitor the water level and provide real-time data to a central processing unit, usually a microcontroller like Arduino or Raspberry Pi. The sensors detect the precise water level and transmit this information to the microcontroller.

The microcontroller, programmed using languages such as C or Python, processes the data received from the sensors. It compares the current water level against predefined threshold values set to trigger specific actions. When the water level approaches the maximum limit, the microcontroller activates a relay switch connected to the water pump, cutting off the water supply to prevent overflow. This automation ensures that the tank does not exceed its capacity, effectively eliminating the risk of water spillage.

In addition to preventing overflow, the system is designed to alert users about the tank's status. This is achieved through the integration of communication modules like GSM, Wi-Fi, or Bluetooth. When the water level reaches critical points, the microcontroller sends a signal to these modules, which then relay alerts to the users' smartphones or other devices. The alerts can be in the form of SMS, push notifications, or even emails, depending on the communication module and the user’s preference. This feature allows users to stay informed about the water levels remotely and take necessary actions if needed.

Power management is a crucial aspect of the system's methodology. The sensors and microcontroller are powered by a reliable power source, which can be either a direct power supply or a battery with backup. The system is designed to be energy-efficient, ensuring that it

can operate continuously without significant power consumption. In scenarios where a battery is used, the system includes a low-power mode to conserve energy and extend the battery life. The installation process of the system is straightforward and user-friendly. The sensors are securely mounted at appropriate levels within the tank, ensuring they can accurately measure the water levels. The microcontroller and other electronic components are housed in a waterproof enclosure to protect them from moisture and potential damage. Proper calibration of the sensors is essential to ensure precise water level detection and accurate alerts.

Software development for the system involves writing code to handle sensor data, control the relay switch, and manage communication protocols. The code includes algorithms to filter out any noise or false readings from the sensors, ensuring reliable performance. Additionally, the software includes routines to periodically check the system's status and perform self- diagnostics, alerting users in case of any malfunction or maintenance requirement.

The system is also designed to be scalable and customizable, allowing users to modify settings according to their specific needs. For instance, the threshold water levels can be adjusted based on the tank's capacity, and the type of alerts can be customized to suit individual preferences. Moreover, the system can be integrated with other smart home devices, creating a comprehensive home automation network.

The Automatic Water Tank Overflow Prevention and Alert System combines sensor technology, microcontroller programming, and communication modules to create an efficient and reliable solution for water management. By preventing tank overflow and providing timely alerts, the system helps conserve water, reduce wastage, and ensure that users are always aware of their water levels. Its user-friendly installation, energy-efficient design, and customizable features make it a practical and valuable addition to any home or commercial establishment.

**CHAPTER 4**

**HARDWARE AND SOFTWARE DESCRIPTION**

# CHAPTER 4

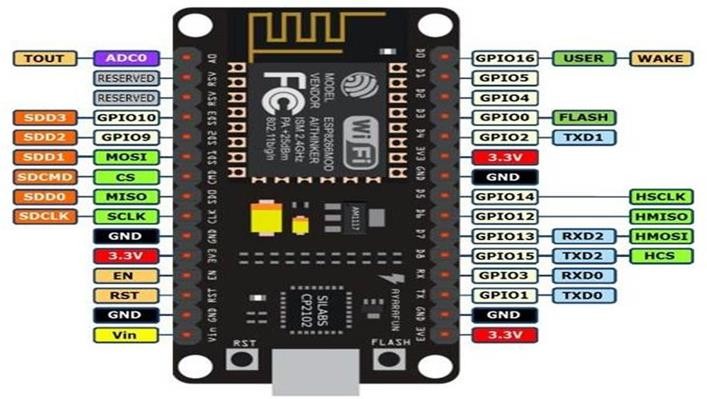
**HARDWARE AND SOFTWARE DESCRIPTION**

## HARDWARE DESCRIPTION

* ESP32 DEV KIT V1
* HC-SR04 Ultrasonic Sensor
* 220-ohm Resistors
* LED
* 2-pin Push Button
* Buzzer
* Breadboard
* Jumper Wires
* Data Communication Cable for ESP32

## ESP32 DEV KIT V1

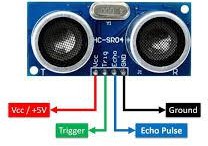
NodeMCU is an open-source development board based on the ESP8266 Wi-Fimodule. It combines the functionality of an Arduino with the ability to connect to the internet, making it ideal for IoT projects. NodeMCU uses the Lua scripting language andfeatures a microcontroller unit (MCU) with integrated Wi-Fi capabilities, allowing for easy programming and wireless communication. With its compact size and low cost, NodeMCU is widely used for prototyping and developing IoT applications, such as home automation, sensor networks, and remote monitoring systems, offering a versatile and affordable solution for building connected device.



**Fig 4.1 NodeMCU**

### HC-SR04 Ultrasonic Sensor

The HC-SR04 ultrasonic sensor measures distance by emitting ultrasonic pulses and timing the echo return. It consists of a Trigger pin to start the measurement and an Echo pin to receive the returned signal. This sensor is pivotal in our project as it continuously monitors the water level by measuring the distance between the sensor and the water surface, enabling accurate level detection and triggering alarms when thresholds are reached.



**Fig 4.2 HC-SR04 Ultrasonic Sensor**

## BUZZER

The 5V DC buzzer is an audible signalling device that produces sound when activated. It operates on a 5V power supply and is controlled by the ESP32 to emit sound alerts. In our project, the buzzer is used to generate audible alarms for low and high-water levels, providing an effective alert system to notify users of critical water conditions that require attention.



**Fig 4.3 Buzzer**

## PUSH BUTTON

The 2-pin push button is a simple mechanical switch used for user interaction with the system. It provides a direct input to the ESP32, allowing users to perform actions such as stopping the alarm. In our project, the push button is used to manually deactivate the alarm, giving users control over the system's audio alerts and ensuring they can easily respond to alarm conditions.



**Fig 4.4 Push Button**

### 220-ohm Resistor

These resistors are designed to limit the current flowing through electronic components, thereby preventing damage. With a resistance of 220 ohms and a power rating of 0.25 watts, they are commonly used in low-current circuits. In our project, these resistors are crucial for protecting LEDs and ensuring they operate safely without excessive current that could potentially harm the components or affect the system’s performance.



**Fig 4.5 220-ohm resistor**

## LED

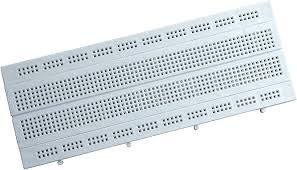
Light Emitting Diodes (LEDs) are used to provide visual indicators for various system states. They emit light when current passes through them, making them useful for signalling conditions such as low or high-water levels. In the context of our project, LEDs offer immediate visual feedback about the water level status and system alerts, helping users quickly assess the situation.



**Fig 4.6 LED**

### Breadboard

A breadboard is a versatile tool for prototyping and testing electronic circuits without the need for soldering. It allows for the temporary and flexible arrangement of components and connections. In this project, the breadboard is used to assemble and test the circuit, making it easier to experiment with different configurations and ensure the components are correctly connected before finalizing the design.



**Fig 4.7 Breadboard**

### Jumper wires

Jumper wires are flexible electrical wires used to establish connections between different components on a breadboard or between the breadboard and external devices. They come in various lengths and facilitate easy wiring of circuits. In our project, jumper wires are essential for connecting the ESP32 to the ultrasonic sensor, OLED display, and other components, ensuring reliable and convenient connections during assembly and testing.



**Fig 4.8 Jumper wires**

* + 1. **Power supply (lithium ion 18650 battery)**



**Fig. 4.9 Power supply (lithium ion 18650 battery)**

A 7-volt power supply using a lithium-ion 18650 battery provides a reliable and compact energy source for various electronic devices. The 18650 battery is known for its high energy density, long cycle life, and stable voltage output, making it suitable for portable and battery- powered applications. With a nominal voltage of 3.7 volts per cell, two 18650 cells in series can deliver a combined voltage of approximately 7.4 volts, which is close to the 7-volt requirement. This setup ensures a consistent power supply for devices requiring 7 volts. Proper management of charging and discharging is crucial to maximize battery life and maintain performance.

## SOFTWARE DESCRIPTION

* + 1. **ARDUINO IDE**

The open-source Arduino Software (IDE) makes it easy to write code and upload it to the board. This software can be used with any Arduino board. Arduino IDE works in conjunction with the Arduino controller, which helps to write code and upload it to various Arduino boards.

Arduino is an open-source prototype platform that uses simple hardware and software to operate. It comprises of a programmable circuit board (also known as a microcontroller) and ready-made software called Arduino IDE (Integrated Development Environment), which is used to develop and upload computer code to the physical board. A sketch is a program written using the Arduino IDE.

The Arduino IDE supports C and C++ programming languages through the use of unique code structuring guidelines. The Arduino IDE includes a software library from the Wiring project that contains many common input and output processes. User-written code only requires two fundamental functions: beginning the sketch and running the main program loop, which are built and linked with a program stub main () into an executable cyclic executive program using the GNU toolchain, which is also included in the IDE release.



**Fig 4.10 Arduino IDE**

## BLYNK APP

Blynk is an IoT platform enabling users to develop custom apps for remotely controlling hardware. Its intuitive interface allows easy creation of interfaces with drag-and-drop widgets. Compatible with various hardware like Arduino, Raspberry Pi, and ESP8266. Security features ensure safe communication, while APIs facilitate integration with different platforms and languages. Supported by an active community, Blynk provides robust customer support. With Blynk, users can efficiently monitor and control IoT projects, making it a favored choice among hobbyists and professionals alike for its versatility and ease of use.



**Fig 4.11 Blynk App**

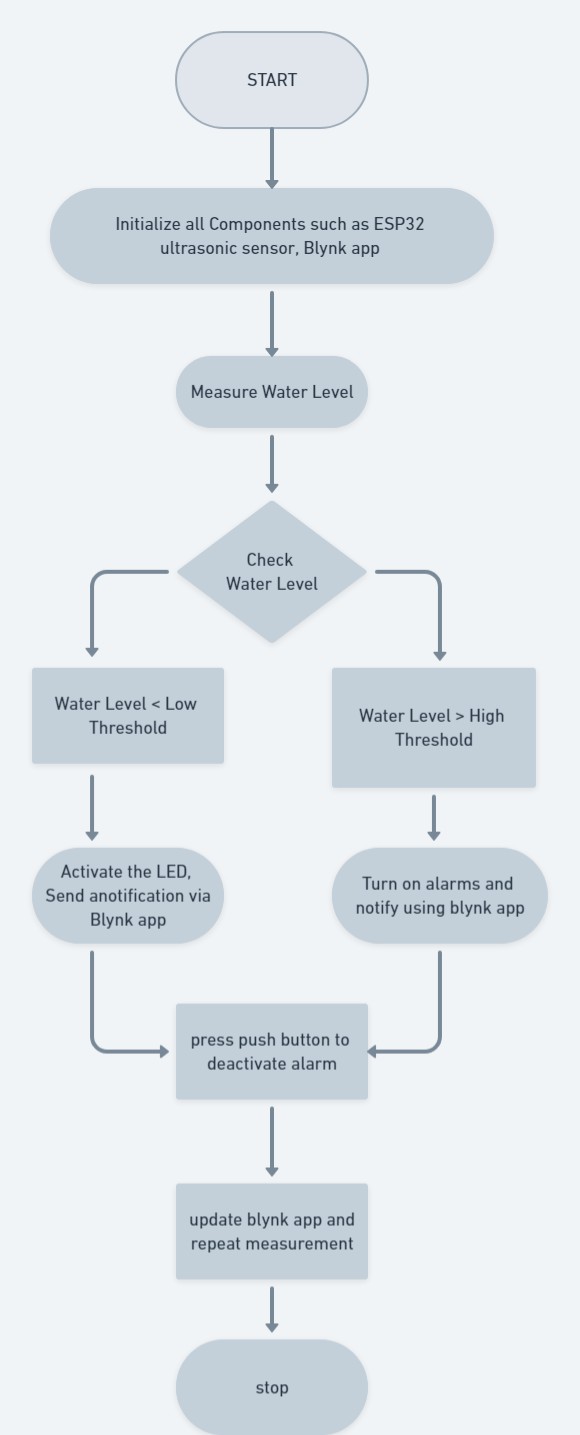
**CHAPTER-5**

**IMPLEMENTATION**

* 1. **Flow Chart**

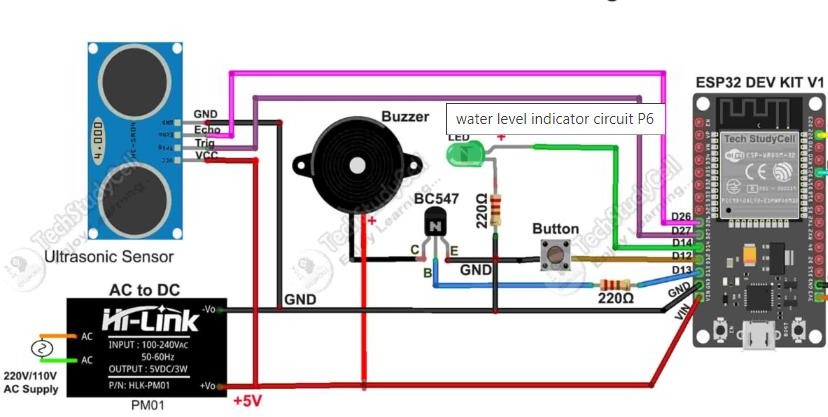
**CHAPTER-5**

**IMPLEMENTATION**



**Fig 5.1 Flow Chart**

* 1. **Circuit & Block Diagram**

 **5.2 Circuit Diagram**

Push Button &

Resistor

ESP32 Node MCU

Transistor

BC547

Battery Supply

Buzzer

LED &

Resistor

Ultra sonic Sensor

**5.3 Block diagram**

Push Button &

Resistor

ESP32 Node MCU

Transistor

BC547

Battery Supply

Buzzer

LED &

Resistor

**CHAPTER-6**

**RESULT AND DISCUSSIONS**

# CHAPTER-6

**RESULT AND DISCUSSIONS**

## 6.1RESULT AND DISCUSSIONS:

Upon completing the assembly and programming of the IoT-based water level indicator, several tests were conducted to evaluate its performance and reliability. The key outcomes and corresponding discussions are as follows:

**Real-Time Water Level Monitoring**

The system successfully measured the water level using the HC-SR04 ultrasonic sensor and Real-time updates provided continuous monitoring, essential for effective water management. Initial calibration was necessary to ensure accuracy, with periodic recalibration required to account for temperature and tank shape variations.

**Alarm Functionality**

The alarm system, including the 5V DC buzzer, effectively alerted users to low and high-water levels. The 2-pin push button reliably deactivated the alarm, enhancing usability by allowing users to manually stop the alert. This demonstrated the successful integration of the buzzer and push button with the ESP32.

**Blynk Notifications**

The ESP32 DEV KIT V1 communicated seamlessly with the Blynk IoT platform, sending timely notifications to a connected smartphone for low and high-water level alerts. The Blynk app accurately displayed water level data, providing real-time updates. While the system's performance depended on reliable Wi-Fi connectivity, overall, it demonstrated consistent notification delivery.

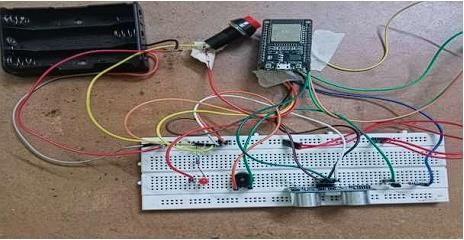
**System Stability and Reliability**

The system operated stably over extended periods, consistently performing monitoring and alerting functions. The breadboard setup facilitated flexible adjustments during testing. For long-term deployment, a more permanent assembly may be preferred to ensure durability.

The project achieved real-time water level monitoring and remote notifications, enhancing user convenience and responsiveness. The push button provided effective user control for alarm deactivation.

**Challenges and Limitations**

Sensor calibration was necessary to maintain measurement accuracy, with periodic recalibration required. The system relied on a stable power supply and reliable Wi-Fi connectivity, which could affect performance during interruptions.



**Fig 6.1 Prototype**

**CHAPTER-7 CONCLUSION AND FUTURE SCOPE**

**CHAPTER-7**

**CONCLUSION AND FUTURE SCOPE**

## CONCLUSION:

The IoT-based water level monitoring system developed in this project effectively demonstrates the integration of modern microcontroller technology with cloud-based IoT platforms to create a practical and user-friendly solution for real-time water level management. By leveraging the capabilities of the ESP32 microcontroller and the Blynk platform, the system provides continuous monitoring, local alerts, and remote notifications, ensuring that users are always informed of critical water levels, whether they are on-site or away. This implementation not only enhances convenience but also significantly reduces the risk of water shortages or overflows, contributing to more efficient water usage.

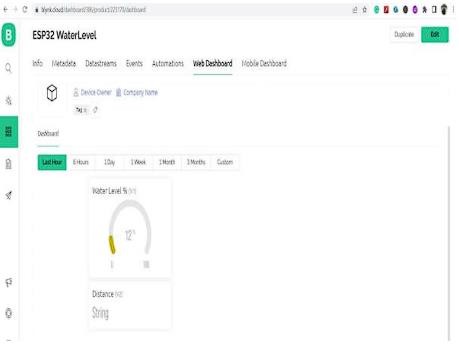
The project not only highlights the potential of IoT in enhancing everyday tasks but also offers a scalable and adaptable solution that can be customized for various applications, from residential water tanks to industrial reservoirs. The use of simple yet reliable components ensures that the system is both cost-effective and easy to replicate, making it accessible to a wide range of users with varying technical expertise. Moreover, the successful execution of this project serves as a foundational step towards further exploration and development of IoT- based systems for resource management, encouraging innovation in this growing field.

## FUTURE SCOPE:

The IoT-based water level monitoring system developed in this project offers a solid foundation with ample opportunities for future enhancements. One key area for advancement is the integration of machine learning algorithms to analyse water usage patterns and optimize resource management. By predicting trends based on historical data, the system could automatically adjust alert thresholds and recommend more efficient water usage strategies. Additionally, expanding the system’s connectivity with other smart home or industrial

automation systems, such as smart irrigation, could lead to more comprehensive and integrated solutions for water management.

Another potential improvement involves scaling the system to monitor multiple tanks or reservoirs, making it suitable for larger applications like agricultural fields or multi-story buildings. Furthermore, adding additional sensors, such as those for water quality, could broaden the system’s functionality, offering users both level and quality monitoring. This would be especially valuable in areas where water quality is a concern, ensuring safe and efficient water use. These future enhancements highlight the system's adaptability and potential to contribute significantly to smarter water management practices.



**Fig 7.1 Output**

# REFERENCES

[1]. Priya J, Sailusha Chekuri, water level monitoring system using IoT, International Research Journal of Engineering and Technology (IRJET) Volume: 04 Issue: 12, Dec-2017.

[2]. Madhurima Santra, Sanjoy Biswas, Sibasis Bandhapadhyay and Kaushik Palit, Smart Wireless water level Monitoring & Pump controlling System, International Journal of Advances in Scientific Research and Engineering (IJASRE), Vol. 03, Issue 4, May -2017.

[3]. Beza Negash Getu and Hussain A. Attia, Automatic Water Level Sensor and Controller System, 2016 IEEE.

[4]. Asaad Ahmed Mohammed ahmed Eltaieb and Zhang Jian Min, Automatic Water Level Control System, International Journal of Science and Research (IJSR), Volume 4 Issue 12, December 2015.

[5]. K. Santhosh Kumar, G. Mukesh, K. Deepti, Microcontroller based Automatic Water level Control System ,International Refereed Journal of Engineering and Science (IRJES) Volume 4, Issue 11, November 2015.

[6]. Sanam Pudasaini, Anuj Pathak, Sukirti Dhakal and Milan Paudel, “Automatic Water Level Controller with Short Messaging Service (SMS) Notification”, International Journal of Scientific and Research Publications, Volume 4, Issue 9, September 2014.

[7]. Md. Moyeed Abrar1, Rajendra.R.Patil2, “Logic Gate Based Automatic Water Level Controller”, IJRET: International Journal of Research in Engineering and Technology, Volume 03, Issue 04, pp. 477-482, 2014.

[8]. Sanam Pudasaini, Anuj Pathak, Sukirti Dhakal and Milan Paudel, “Automatic Water Level Controller with Short Messaging Service (SMS) Notification”, International Journal of Scientific and Research Publications, Volume 4, Issue 9, pp. 1-4, 2014.

[9]. Muktha Shankari , Jyothi , Manu E , Naveen I and Harsha Herle, "Wireless Automatic Water Level Control using Radio Frequency Communication", International Journal of Advanced Research in Electrical, Electronics and Instrumentation Engineering, Vol. 2, Issue 4,

pp. 1320-1324, 2013.

[10]. S. B. Jagadal and s. V. Halse, “8051 Microcontroller Based Multiple Water Tank Control System” Journal of Computer and Mathematical Sciences Vol. 4, Issue 5, pp. 382-401, 2013.

[11]. S. M. Khaled Reza, Shah Ahsanuzzaman Md. Tariq and S.M. Mohsin Reza "Microcontroller Based Automated Water Level Sensing and Controlling: Design and Implementation Issue" Proceedings of the World Congress on Engineering and Computer Science, Vol. I, 2010.

[12]. Pan Lingfeng, Hu Ji and Zhiwang Xu, "Design and realization of a intelligent mixed water valve control system", International Conference on Electrical and Control Engineering (ICECE), pp. 5816-5818, 2011.

[13]. Tingjun Wang and Li Gu, "An Electronic Water-Saving Integrated System for Toilet Bowl Based on Electromagnetic Valve Controlling", Second International Conference on Information and Computing Science ICIC '09, pp. 258 - 261, 2009.

[14]. Jiang Wei, "Intelligent Building Control of Water Tank Based on Fuzzy Theory", International Conference on Intelligent Computation Technology and Automation (ICICTA),

pp. 549 - 552, 2010.

# APPENDIX:

/\* Fill-in your Template ID (only if using Blynk.Cloud) \*/ #define BLYNK\_TEMPLATE\_ID "TMPL3hwr1k8qD" #define BLYNK\_TEMPLATE\_NAME "waterlevel"

#define BLYNK\_AUTH\_TOKEN "ZUCngmnS\_y4qif8lDcEkr6R9t1ZPusEh" #include <Arduino.h>

#include <WiFi.h>

// Your WiFi credentials.

// Set password to "" for open networks. #define WIFI\_SSID "redmi pro 8" #define WIFI\_PASS "prajwal@07"

//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

// OLED SPI pins

#define OLED\_MOSI 23

#define OLED\_CLK 18

#define OLED\_DC 16

#define OLED\_CS 5

#define OLED\_RESET 4

Adafruit\_SSD1306 display(SCREEN\_WIDTH, SCREEN\_HEIGHT, OLED\_MOSI, OLED\_CLK, OLED\_DC, OLED\_RESET, OLED\_CS);

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 setupWiFi()

{

Serial.printf("\r\n[Wifi]: Connecting"); WiFi.begin(WIFI\_SSID, WIFI\_PASS);

while (WiFi.status() != WL\_CONNECTED)

{

Serial.printf("."); delay(250);

}

digitalWrite(wifiLed, HIGH);

Serial.printf("connected!\r\n[WiFi]: IP-Address is %s\r\n", WiFi.localIP().toString().c\_str());

}

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);

// Initialize OLED display if(!display.begin(SSD1306\_SWITCHCAPVCC)) { Serial.println(F("SSD1306 allocation failed")); for(;;);

}

delay(1000); display.setTextSize(1); display.setTextColor(WHITE); display.clearDisplay();

setupWiFi(); Blynk.config(auth);

timer.setInterval(2000L, checkBlynkStatus); // check if Blynk server is connected every 2 seconds

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;

}

}