## ABSTRACT

One of the Precious and important resource in the earth maybe water. Now a days people need everything happen smarter other than olden days techniques used. In this paper Internet of Things (IOT) Based on the concept that to define energy conservation in tank level water monitoring system. The Main objective is to have a sensor which detects the level of water and it should notify the user about the water level which is currently available in the tank.

The Ultrasonic Sensor is placed at the top of the tank in which we will measure the level of water and the if the distance of the water from the sensor gets increased, it means that the water in the tank gets low and finally after reaching to an extent the system should notify a warning message to the user. The Major requirement would be Ultrasonic sensor which senses level of water the top of the tank to the bottom of the tank. The sensor is connected to the system using the Wi-Fi of NODE MCU (ESP8266). The Blynk library is installed and connected in the Arduino. The Blynk application is used to get the values and the notification send to the mobile for the user purpose. The user can get notified that the tank is empty and can take further steps to fill the water in the tank.

## PROBLEM STATEMENT

Traditional methods of monitoring water levels in tanks lack real-time feedback capabilities and are often inefficient. To address this issue, there is a need for an Internet of Things (IoT) based solution that offers accurate and timely monitoring of water levels in tanks. The objective of this project is to develop a smart water tank monitoring system using an Ultrasonic Sensor to detect water levels.

The system will utilize NODE MCU (ESP8266) for Wi-Fi connectivity and integrate with the Blynk application for mobile notifications. Challenges include ensuring sensor accuracy in various conditions and establishing robust communication between the sensor system and the Blynk app.

# OBJECTIVES

- Develop a smart water tank monitoring system that employs advanced technology to enhance accuracy and efficiency compared to existing systems.
- Utilize an Ultrasonic Sensor for precise detection of water levels inside the tank, ensuring reliable performance in varying environmental conditions.
- Integrate the Ultrasonic Sensor with NODE MCU (ESP8266) to enable wireless connectivity and transmission of real-time data over Wi-Fi networks.
- Implement Arduino IDE code to facilitate communication between the sensor system and the Blynk application, ensuring seamless integration with the user's smartphone.
- Enable the Blynk application to receive status updates and notifications regarding the water level in the tank, empowering users to take proactive measures as needed.
- Ensure compatibility and user-friendly operation of the monitoring system across different smartphone platforms, maximizing accessibility and usability.
- Conduct thorough testing and validation to verify the accuracy, reliability, and robustness of the smart water tank monitoring system under various conditions.
- Evaluate the energy efficiency of the system to minimize power consumption and maximize battery life, optimizing long-term sustainability and usability.
- Explore potential scalability and expandability options to accommodate future enhancements or additional functionalities, ensuring adaptability to evolving user needs and technological advancements.

## LIBRARIES USED

- **`ESP8266WiFi.h`:** This library provides support for connecting the ESP8266 module to a Wi-Fi network. It includes functions to configure the module to connect to a specific Wi-Fi network, handle authentication, and manage the Wi-Fi connection. This library is essential for enabling Wi-Fi communication in projects using the ESP8266 microcontroller.

- **`BlynkSimpleEsp8266.h`:** This library is specifically designed for integrating the ESP8266 microcontroller with the Blynk IoT platform. It allows seamless communication between the ESP8266 module and the Blynk cloud server, enabling remote monitoring and control of IoT devices via the Blynk mobile application. This library simplifies the process of setting up Blynk projects on ESP8266-based devices.

- **`Wire.h`:** This library provides support for I2C (Inter-Integrated Circuit) communication protocol, commonly used for connecting multiple devices in a master-slave configuration. It includes functions for initializing the I2C bus, transmitting and receiving data between devices, and managing communication errors. This library is essential for projects involving communication between microcontrollers and I2C-compatible sensors or peripherals.

- **`LiquidCrystal_I2C.h`:** This library is used for interfacing with Liquid Crystal Displays (LCDs) that utilize the I2C communication protocol. It simplifies the process of controlling I2C-enabled LCD modules by providing functions for initializing the display, writing characters or strings to specific positions on the screen, and managing backlight brightness. This library is commonly used in projects requiring visual feedback or user interface elements.

## DESCRIPTION OF THE CODE

- The code begins by including the necessary libraries for WiFi connectivity (`ESP8266WiFi.h`) and interfacing with the Blynk platform (`BlynkSimpleEsp8266.h`). It also defines the Blynk authentication token (`auth[]`) and WiFi credentials (`ssid` and `pass`) required for connecting to the Blynk cloud server and the local WiFi network, respectively.

- The `BLYNK_PRINT Serial` statement configures the Serial output for printing Blynk debug information to aid in troubleshooting.

- Within the `setup()` function, the code initializes the Serial communication, sets the pinMode for the ultrasonic sensor's trigger and echo pins, and initializes the Blynk connection using the provided authentication token and WiFi credentials. It also initializes an LCD widget for displaying distance measurements.

- The `loop()` function contains the main logic of the program. It continuously reads the distance measured by the ultrasonic sensor, sends the distance value to the Blynk server to update the LCD widget, and checks if the water level is below a certain threshold. If the water level is low, it sends an email and a push notification to alert the user.

- In conclusion, the code creates a system that monitors the water level in a tank using an ultrasonic sensor and sends notifications to the user via the Blynk app when the water level is low. It demonstrates the integration of hardware components with IoT platforms for smart monitoring and management of resources.

# CODE

## CODE FOR WATER LEVEL MONITERING:

```
#include <Wire.h>

#include <LiquidCrystal_I2C.h>
#define BLYNK_PRINT Serial
#include <ESP8266WiFi.h>
#include <BlynkSimpleEsp8266.h>

//Initialize the LCD display
LiquidCrystal_I2C lcd(0x27, 16, 2);

#define BLYNK_TEMPLATE_ID "TMPL3mNG6P_u8"
#define BLYNK_TEMPLATE_NAME "water level monitoring system"
#define BLYNK_AUTH_TOKEN "NsOB_lLTQzGWhenkaDNmyhgUHTCtKGvS"

char auth[] = "NsOB_lLTQzGWhenkaDNmyhgUHTCtKGvS";//Enter your Auth token
char ssid[] = "Jimmy";  // Enter your Wifi Username
char pass[] = "6366113723"; //Enter your WIFI password

BlynkTimer timer;

// Define the component pins
#define trig D7
#define echo D8
#define LED1 D0
#define LED2 D3
#define LED3 D4
#define LED4 D5
#define LED5 D6
#define relay 3

//Enter your tank max value(CM)
int MaxLevel = 20;

int Level1 = (MaxLevel * 75) / 100;
int Level2 = (MaxLevel * 65) / 100;
int Level3 = (MaxLevel * 55) / 100;
int Level4 = (MaxLevel * 45) / 100;
int Level5 = (MaxLevel * 35) / 100;
```

```
void setup() {
  Serial.begin(9600);
  lcd.init();
  lcd.backlight();
  pinMode(trig, OUTPUT);
  pinMode(echo, INPUT);
  pinMode(LED1, OUTPUT);
  pinMode(LED2, OUTPUT);
  pinMode(LED3, OUTPUT);
  pinMode(LED4, OUTPUT);
  pinMode(LED5, OUTPUT);
  pinMode(relay, OUTPUT);
  digitalWrite(relay, HIGH);
  Blynk.begin(auth, ssid, pass, "blynk.cloud", 80);

  lcd.setCursor(0, 0);
  lcd.print("Water level");
  lcd.setCursor(4, 1);
  lcd.print("Monitoring");
  delay(4000);
  lcd.clear();

  //Call the functions
  timer.setInterval(100L, ultrasonic);
}

//Get the ultrasonic sensor values
void ultrasonic() {
  digitalWrite(trig, LOW);
  delayMicroseconds(4);
  digitalWrite(trig, HIGH);
  delayMicroseconds(10);
  digitalWrite(trig, LOW);
  long t = pulseIn(echo, HIGH);
  int distance = t / 29 / 2;

  int blynkDistance = (distance - MaxLevel) * -1;
  if (distance <= MaxLevel) {
    Blynk.virtualWrite(V0, blynkDistance);
  } else {
    Blynk.virtualWrite(V0, 0);
  }
  lcd.setCursor(0, 0);
  lcd.print("WLevel:");

  if (Level1 <= distance) {
    lcd.setCursor(8, 0);
    lcd.print("Very Low");
    digitalWrite(LED1, HIGH);
    digitalWrite(LED2, LOW);
    digitalWrite(LED3, LOW);
    digitalWrite(LED4, LOW);
    digitalWrite(LED5, LOW);
  } else if (Level2 <= distance && Level1 > distance) {
    lcd.setCursor(8, 0);
    lcd.print("Low");
    lcd.print("      ");
    digitalWrite(LED1, HIGH);
    digitalWrite(LED2, HIGH);
    digitalWrite(LED3, LOW);
    digitalWrite(LED4, LOW);
    digitalWrite(LED5, LOW);
  } else if (Level3 <= distance && Level2 > distance) {
    lcd.setCursor(8, 0);
    lcd.print("Medium");
    lcd.print("      ");
    digitalWrite(LED1, HIGH);
    digitalWrite(LED2, HIGH);
    digitalWrite(LED3, HIGH);
    digitalWrite(LED4, LOW);
    digitalWrite(LED5, LOW);
  } else if (Level4 <= distance && Level3 > distance) {
    lcd.setCursor(8, 0);
    lcd.print("High");
    lcd.print("      ");
    digitalWrite(LED1, HIGH);
    digitalWrite(LED2, HIGH);
    digitalWrite(LED3, HIGH);
    digitalWrite(LED4, HIGH);
    digitalWrite(LED5, LOW);
  } else if (Level5 >= distance) {
    lcd.setCursor(8, 0);
    lcd.print("Full");
    lcd.print("      ");
    digitalWrite(LED1, HIGH);
    digitalWrite(LED2, HIGH);
    digitalWrite(LED3, HIGH);
    digitalWrite(LED4, HIGH);
    digitalWrite(LED5, HIGH);
  }
}
```

```
//Get the button value
BLYNK_WRITE(V1) {
  bool Relay = param.asInt();
  if (Relay == 1) {
    digitalWrite(relay, LOW);
    lcd.setCursor(0, 1);
    lcd.print("Motor is ON ");
  } else {
    digitalWrite(relay, HIGH);
    lcd.setCursor(0, 1);
    lcd.print("Motor is OFF");
  }
}

void loop() {
  Blynk.run();//Run the Blynk library
  timer.run();//Run the Blynk timer
}
```

## CODE FOR WATER PUMP:

```
#define BLYNK_TEMPLATE_ID "TMPL3C4D_S9Y4"
#define BLYNK_TEMPLATE_NAME "Control Pump"
#define BLYNK_AUTH_TOKEN "rh-VZKF3lj2zr4Xdxr8wMOP_KTdML9bu"

#define BLYNK_PRINT Serial
#include <WiFi.h> // Changed from ESP8266WiFi.h for ESP32
#include <BlynkSimpleEsp32.h> // Changed library for ESP32

char auth[] = "rh-VZKF3lj2zr4Xdxr8wMOP_KTdML9bu";
char ssid[] = "Jimmy";
char pass[] = "6366113723";

int relaypin = 4; // Changed pin assignment from D4 to 4 for ESP32
bool pumpState = false;

void setup() {
  Serial.begin(115200);
  Blynk.begin(auth, ssid, pass);
  pinMode(relaypin, OUTPUT);
}

void loop() {
  Blynk.run();
}

BLYNK_WRITE(V0) {
  int value = param.asInt();

  if (value == 1) {
    digitalWrite(relaypin, HIGH);
    pumpState = true;
  } else {
    digitalWrite(relaypin, LOW);
    pumpState = false;
  }
}

BLYNK_READ(V1) {
  Blynk.virtualWrite(V1, pumpState);
}
```