 A blue text on a white background

Description automatically generatedA black and grey logo with a graduation cap and book

Description automatically generated

**GOVERNMENT OF TAMILNADU**

DIRECTORATE OF TECHNICAL EDUCATION, CHENNAI

NAAN MUDHALVAN SCHEME (TNSDC) SPONSORED

STUDENTS DEVELOPMENT PROGRAMME

ON

**IoT AND ITS APPLICATIONS**

**HOST INSTITUTION**

xxxxx

COIMBATORE – 04

**TRAINING PARTNER**

ENTHU TECHNOLOGY SOLUTIONS INDIA PVT LTD

DATE:

|  |  |
| --- | --- |
| Name | Roll No |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |

**Table Of Contents**

|  |  |  |
| --- | --- | --- |
| **S no** | **Title** | **Page no** |
| 1 | Abstract |  |
| 2 | Introduction |  |
| 3 | Hardware and Software Requirements |  |
| 4 | Block Diagram |  |
| 5 | Code |  |
| 6 | Output Results |  |
| 7 | Cloud Output |  |
| 8 | Conclusion |  |

**Abstract**

Water scarcity and inefficient water usage are significant global challenges. A smart water management system can optimize water usage, monitor consumption, and prevent wastage. This project proposes the development of an IoT-based water management system using the ESP32 microcontroller.

Water conservation is a critical challenge in both urban and rural areas. Efficient management of water resources is essential to ensure sustainability. This project presents a smart water management system that utilizes an ESP32 microcontroller in conjunction with an ultrasonic sensor to monitor and control water levels in storage tanks, reservoirs, or other water bodies.

The controller system is further enhanced by integrating it with Wi-Fi connectivity to ThingzMate Cloud, allowing for remote monitoring and control. This connection enables real-time updates on the status of the water management system

The system also features an uplink for remote communication and an authorization mechanism to ensure secure access. Through this project, we aim to improve the efficiency of water management systems.

Overall, this system provides a cost-effective, energy-efficient solution for managing water resources, making it suitable for a wide range of applications, from residential water tanks to large-scale agricultural reservoirs. The use of the ESP32 and ultrasonic sensor combination offers a reliable and scalable approach to addressing the pressing issue of water scarcity.

**Introduction**

Water is a vital resource, essential for life and critical to agriculture, industry, and daily human activities. However, the growing global population, urbanization, and climate change are putting immense pressure on water resources, leading to widespread water scarcity and the need for sustainable water management practices. Inefficient water usage, leaks, and uncontrolled water consumption further exacerbate these challenges.

Traditional water management systems often rely on manual monitoring and control, which can be inefficient, labor-intensive, and prone to human error. These systems lack the ability to provide real-time data or automated responses, making it difficult to optimize water use and prevent wastage.

The advent of the Internet of Things (IoT) has opened new possibilities for enhancing water management through automation, real-time monitoring, and remote control. In this context, the ESP32 microcontroller emerges as a powerful tool for developing smart, connected systems. With its built-in Wi-Fi and Bluetooth capabilities, the ESP32 allows for seamless integration of sensors, actuators, and communication modules, making it ideal for IoT applications.

This project introduces a smart water management system that leverages the capabilities of the ESP32 microcontroller and an ultrasonic sensor to monitor and control water levels in tanks, reservoirs, and other water storage systems. The ultrasonic sensor measures the distance to the water surface, providing accurate real-time data on water levels. The ESP32 processes this data and triggers appropriate actions, such as turning on a pump to refill the tank or sending alerts to prevent overflow.

**Hardware and Software Requirements**

**Hardware Requirements**

1.ESP32 Microcontroller

2.Ultrasonic sensor

3.BreadBoard

4.Display module

5.Jumper Wires

6.Water pump

7.Relay module

**Software Requirements**

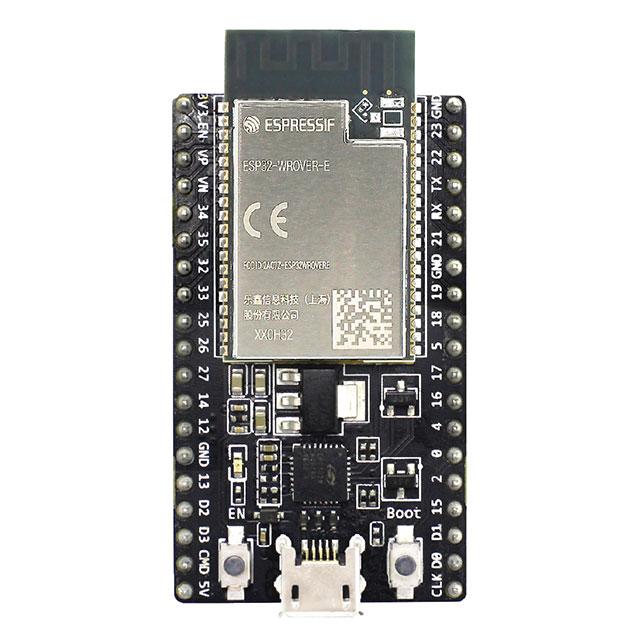
1.Wokwi Simulator

2.Arduino IDE

3.Thingzmate Cloud

****

**ESP32 Microcontroller**

****

The ESP32 microcontroller is a powerful and versatile chip used in this project to control the four-way traffic light system. It features integrated Wi-Fi and Bluetooth capabilities, allowing for seamless communication with the ThingzMate Cloud for remote monitoring and control. With its dual-core processor and multiple GPIO pins, the ESP32 efficiently

**Ultrasonic sensor**

****

An ultrasonic sensor is a type of electronic device that uses sound waves to measure the distance between the sensor and an object. It operates on the principle of echolocation, similar to how bats navigate in the dark. The sensor emits ultrasonic sound waves, which travel through the air and reflect off the surface of an object, such as water.

**BreadBoard**

****

The breadboard is an essential component in this project, used to prototype the circuit connections for the four-way traffic light controller system. It allows for easy placement and rearrangement of the LEDs, resistors, and connections to the ESP32 microcontroller without the need for soldering. The breadboard's flexibility makes it ideal for quickly testing and modifying the circuit design as the project develops.

**Display module**



A display module is an essential component in a smart water management system, providing a visual interface for users to monitor system parameters such as water levels, pump status, and sensor readings. In a system using the ESP32, a display can show real-time data, alerts, and operational statuses, making the system more user-friendly and interactive.

**Jumper Wires**

****

Jumper wires are essential in this project, used to connect the ESP32 microcontroller to the components on the breadboard. These wires provide a flexible and reliable way to link the microcontroller’s GPIO pins to the LEDs, resistors, and other circuit elements, enabling proper signal and power flow. Their ease of use allows for quick modifications and testing during the prototyping stage.

**Water pump**

****

A water pump is a mechanical device used to move water from one location to another. In a water management system, the water pump plays a crucial role in controlling the distribution and flow of water, whether for irrigation, filling storage tanks, or maintaining water pressure in a plumbing system.

**Relay module**

****

A relay is an electrically operated switch that allows a low-power control signal to control a higher-power circuit. In a water management system using an ESP32, the relay plays a crucial role in enabling the microcontroller to control high-power devices, such as water pumps, by isolating the low-power control circuit from the high-power load.

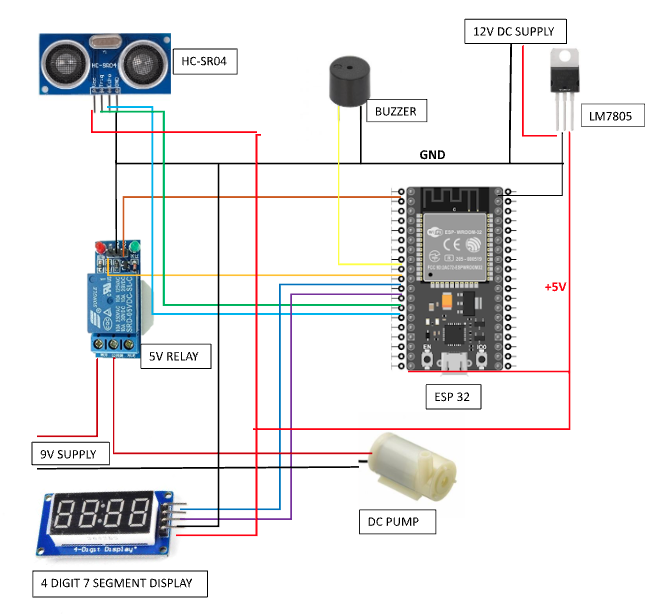
**Arduino IDE**

Arduino IDE is the primary development environment used in this project for programming and uploading code to the ESP32 microcontroller. It provides an easy-to-use interface for writing, compiling, and debugging the code that controls the four-way traffic light system. With its extensive library support and compatibility with ESP32, the Arduino IDE streamlines the development process, allowing for efficient code iteration and testing.

**Thingzmate Cloud**

ThingzMate enables real-time simulation of a 4-way traffic light control system, allowing you to monitor and manage traffic lights via cloud connectivity.It provides tools for configuring traffic light sequences, ensuring accurate simulation of traffic flow and light transitions.With ThingzMate, you can easily visualize and control traffic light statuses, optimizing traffic management and ensuring efficient simulation scenarios.

**Block Diagram**



**Code**

#include <WiFi.h>

#include <HTTPClient.h>

// Replace these with your ThingzMate settings

const char \*serverUrl = "https://console.thingzmate.com/api/v1/device-types/esp3264/devices/esp3264/uplink"; // Replace with your server endpoint

String AuthorizationToken = "Bearer 3b2ff09756866857e8ed6d2b24685af6";

// Replace these with your WiFi credentials

const char\* ssid = "Remi9A";

const char\* password = "subash123";

// Define the GPIO pins for the ultrasonic sensor

const int trigPin = 23;

const int echoPin = 22;

int Relay = 26;

// Define variables for measuring distance

long duration;

float distance;

void setup() {

Serial.begin(9600);

pinMode(trigPin, OUTPUT);

pinMode(echoPin, INPUT);

pinMode(Relay, OUTPUT);

// Attempt to connect to Wi-Fi

connectToWiFi();

}

void connectToWiFi() {

WiFi.begin(ssid, password);

Serial.println("Connecting to WiFi...");

int attempts = 0;

while (WiFi.status() != WL\_CONNECTED && attempts < 20) { // Try for 10 seconds (500ms \* 20)

delay(500);

Serial.print(".");

attempts++;

}

if (WiFi.status() == WL\_CONNECTED) {

Serial.println("Connected to WiFi");

Serial.print("IP Address: ");

Serial.println(WiFi.localIP());

} else {

Serial.println("Failed to connect to WiFi");

// Optional: Add code to handle a failed connection (e.g., reset the device, or enter deep sleep)

}

}

void loop() {

// Only run the main logic if connected to WiFi

if (WiFi.status() == WL\_CONNECTED) {

// Measure distance

digitalWrite(trigPin, LOW);

delayMicroseconds(2);

digitalWrite(trigPin, HIGH);

delayMicroseconds(10);

digitalWrite(trigPin, LOW);

duration = pulseIn(echoPin, HIGH);

distance = (duration \* 0.0344) / 2; // Calculate distance in cm

// Print distance to Serial Monitor

Serial.print("Distance: ");

Serial.print(distance);

Serial.println(" cm");

String status;

if (distance < 10) {

status = "PUMP OFF";

Serial.println(status);

digitalWrite(Relay, LOW);

} else {

status = "PUMP ON";

Serial.println(status);

digitalWrite(Relay, HIGH);

delay(1000);

}

// Send data to ThingzMate

sendToCloud(status);

} else {

Serial.println("WiFi not connected, attempting to reconnect...");

connectToWiFi();

}

}

void sendToCloud(String status) {

HTTPClient http;

http.begin(serverUrl);

http.addHeader("Content-Type", "application/json");

http.addHeader("Authorization", AuthorizationToken);

// Create JSON payload with the status text and distance

String payload = "{\"status\":\"" + status + "\",\"distance\":\"" + String(distance) + "\"}";

// Send POST request

int httpResponseCode = http.POST(payload);

if (httpResponseCode > 0) {

String response = http.getString();

Serial.println("HTTP Response code: " + String(httpResponseCode));

Serial.println(response);

} else {

Serial.print("Error code: ");

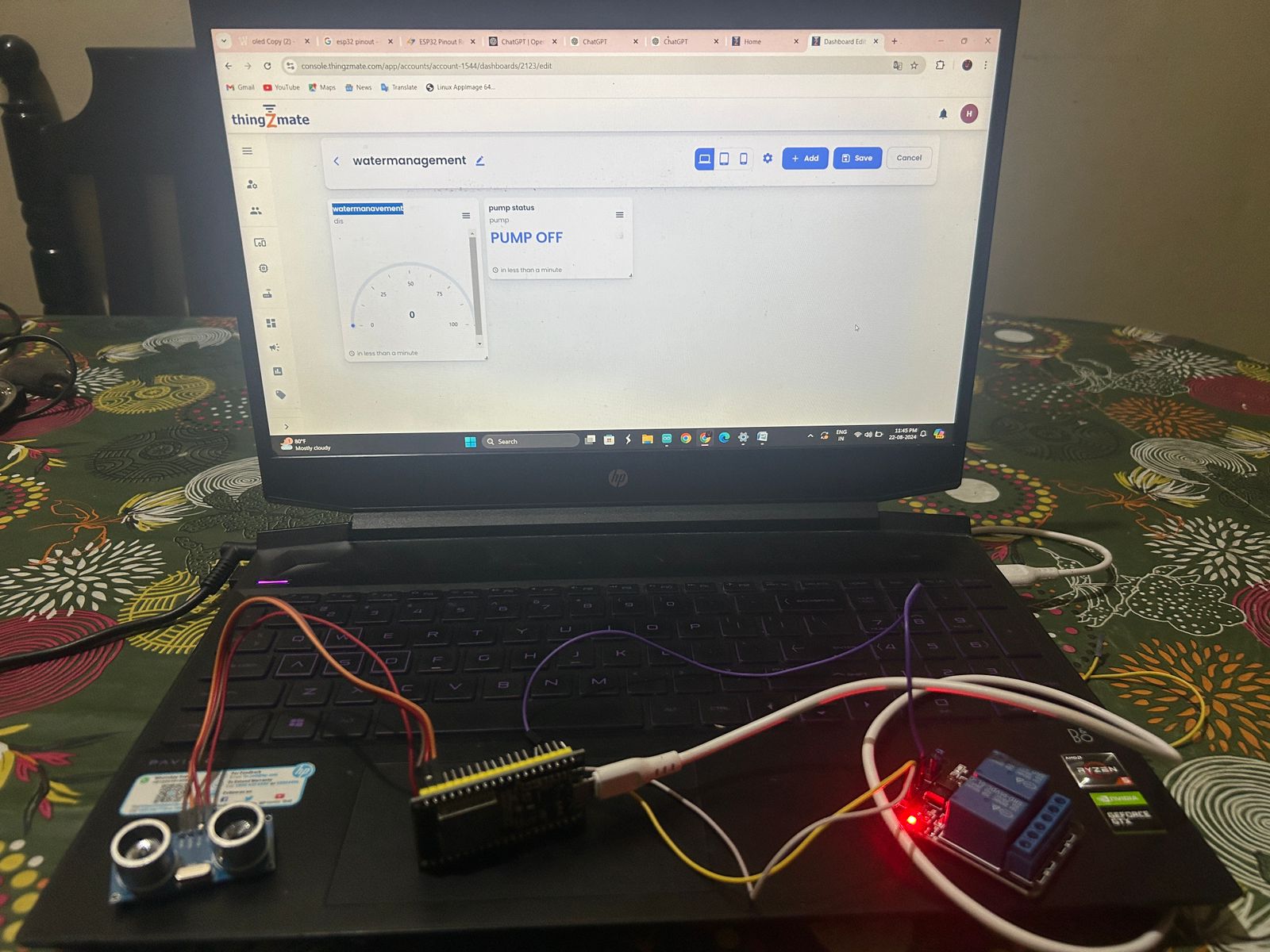
Serial.println(httpResponseCode);

}

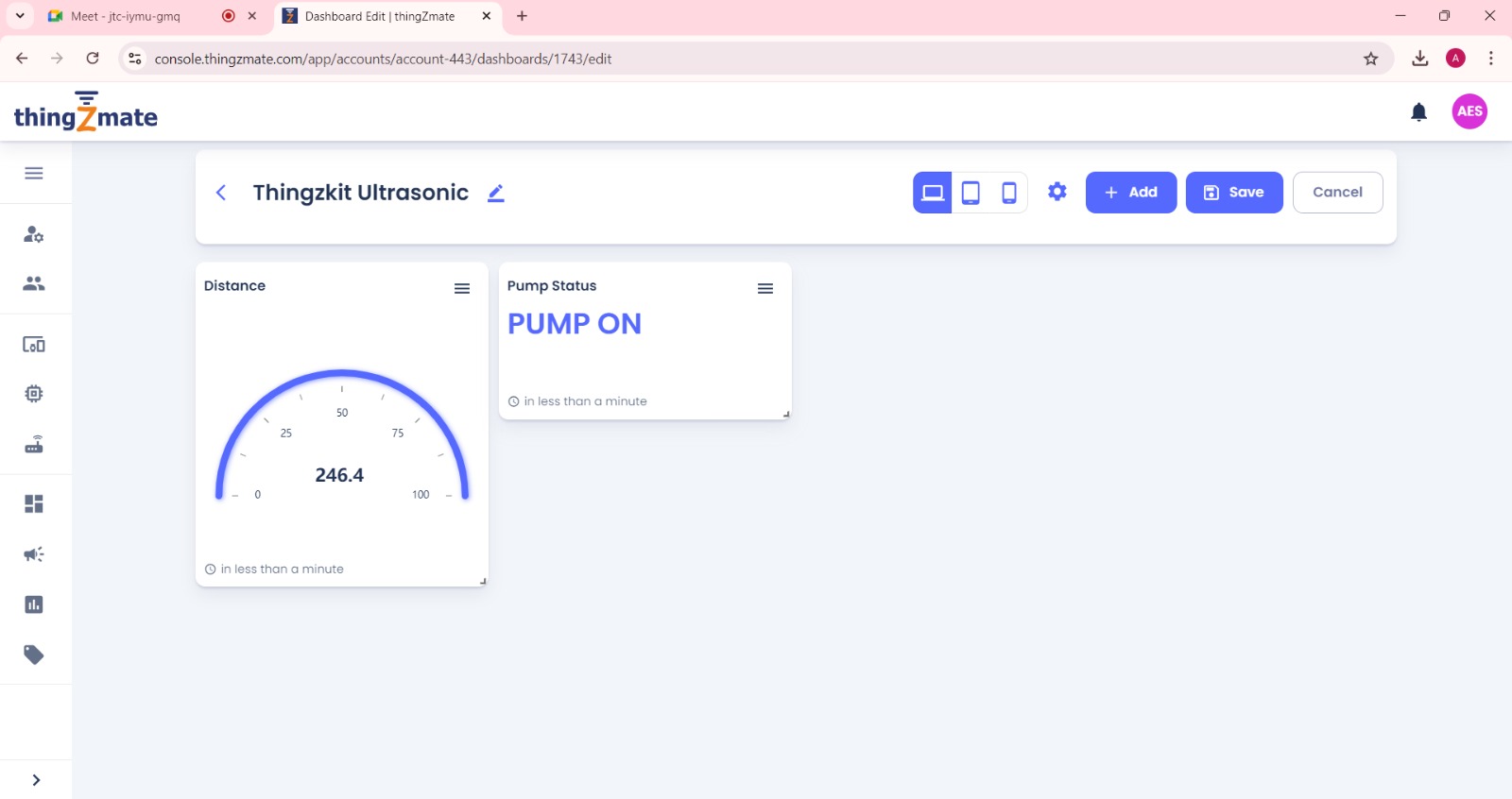
http.end();

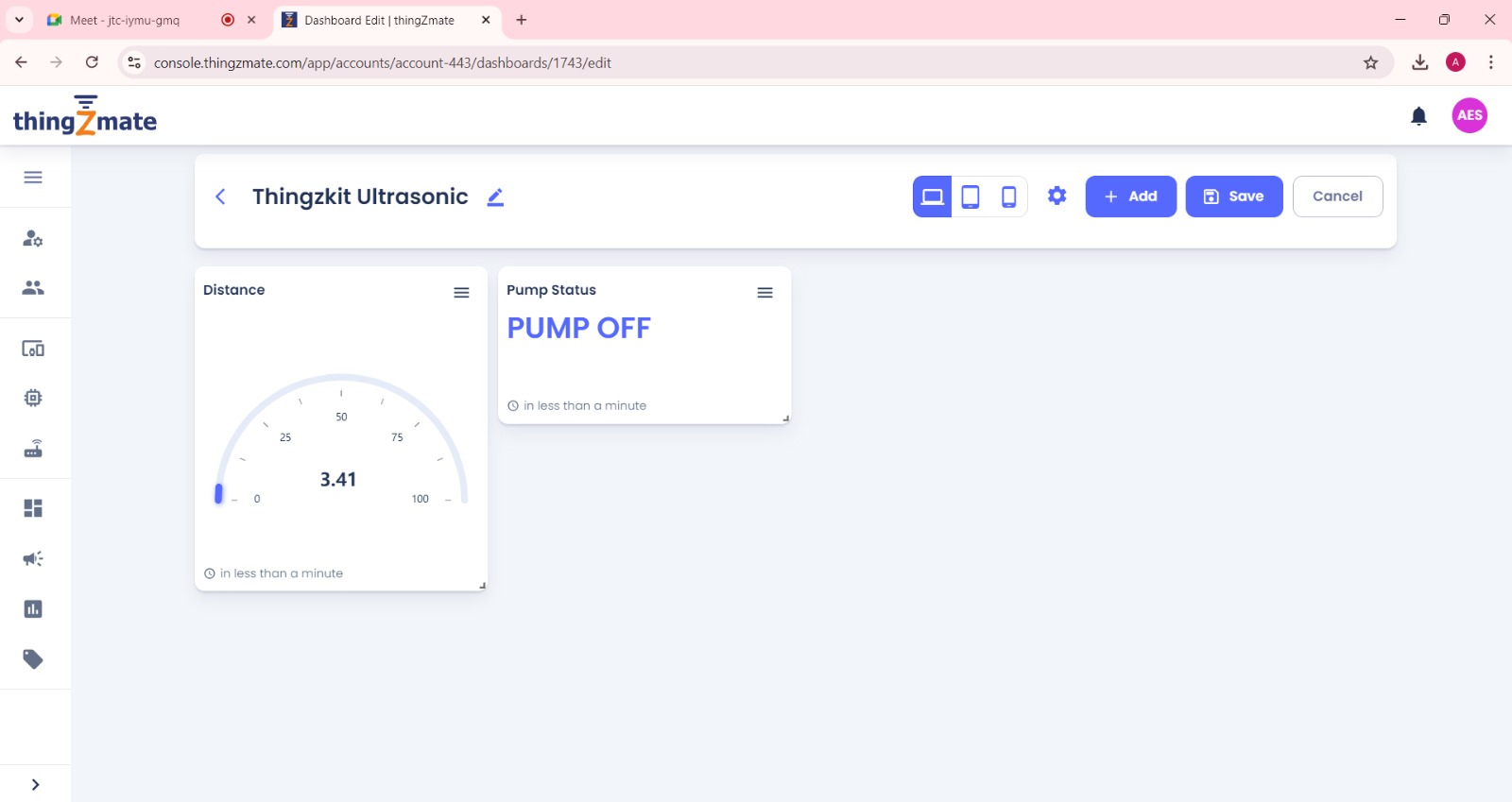
}

**Output Results**

****

**Cloud Output**

****

****

**Conclusion**

The integration of the ultrasonic sensor provides accurate and reliable water level measurements, while the ESP32’s capabilities enable automated control of the water pump based on these measurements. The inclusion of a relay allows the system to safely and efficiently manage the operation of high-power devices like pumps. Furthermore, a display module enhances user interaction by providing real-time data, alerts, and system status, making the system intuitive and user-friendly.

****