Project Report on

ESP-32 based Electronic Voting Machine

Indian Institute of Information Technology
Design and Manufacturing, Kurnool

Submitted by

Pranjal Upadhyay
523EC0012
Integrated B. Tech and M. Tech
Department of Electronics and Communication Engineering

April 17, 2025

Abstract

This project presents the design and development of an **ESP32-based Electronic Voting Machine (EVM)** with integrated IoT connectivity using the HTTP protocol. The system aims to enhance the efficiency, transparency, and security of small-scale election processes through electronic automation and real-time cloud monitoring. It provides an intuitive interface using push buttons for voting, an I2C-based 8x2 LCD for displaying results, and integrated LEDs for instant feedback.

The ESP32 microcontroller serves as the core processing unit, handling voting logic, result computation, and HTTP-based data transmission to the **ThingSpeak** cloud platform. This enables real-time visualization of election data and remote monitoring of results. The project ensures a balance between usability, accuracy, and reliability, making it suitable for institutional, community, and small-scale governmental use cases.

To promote scalability and security, the system design anticipates future integrations such as biometric voter authentication, encrypted communication, and cloud-based multi-node result aggregation. The proposed EVM demonstrates how embedded systems and IoT can converge to modernize voting mechanisms with minimal human intervention and high integrity.

Pranjal Upadhyay

(pranjal2004upadhyay@gmail.com)

Contents

1. Objective

The Electronic Voting Machine (EVM) with HTTP Integration aims to ensure a secure, reliable, and real-time voting process by integrating live vote monitoring. The key features of the system are:

- **Push-Button Voting:** Provides a simple electronic voting interface, eliminating the need for physical ballots.
- **Instant Vote Display:** An integrated LCD screen shows real-time vote counts for immediate verification.
- **Live Cloud Monitoring:** Utilizes the HTTP protocol to transmit data to the ThingSpeak platform for remote tracking and analysis.
- **Standalone Feedback System:** An LED indicator provides immediate confirmation that a vote has been successfully cast.
- **Integrated ESP32 Control:** The ESP32 microcontroller manages all core processes, including vote counting, display updates, and cloud data transmission.
- **Secure & Reliable Operation:** Implements error handling and data logging to ensure a smooth and robust voting process.
- **Future Expansion:** The system is designed to be scalable for future upgrades, such as face recognition, fingerprint authentication, blockchain security, and mobile/web application integration.

This system provides a framework for secure, efficient, and tamper-proof electronic voting.

. . . .

2. Scope

Advantages

- **Real-Time Monitoring:** Enables live vote tracking through cloud integration using the HTTP protocol and the ThingSpeak platform.
- **Scalability:** The system is designed to be expanded with advanced features like face recognition, blockchain security, and mobile voting capabilities.
- **Cost-Effective & Efficient:** Eliminates the need for paper ballots, significantly reducing election costs and the potential for manual errors.

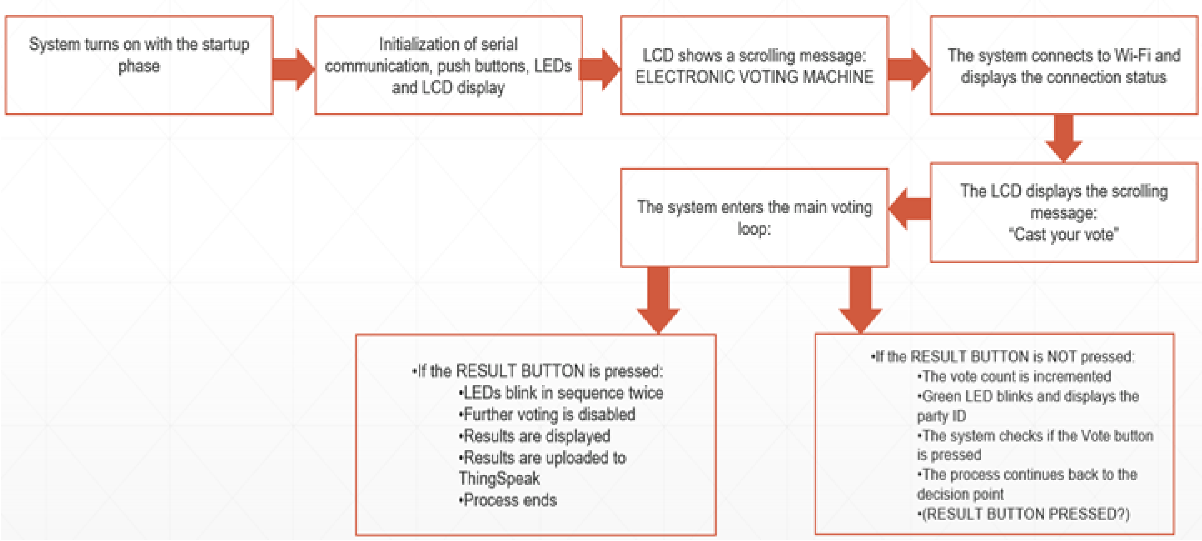
- **Fully Automated System:** By removing the need for manual vote counting, the system improves both the accuracy and speed of announcing results.

Applications

- **Institutional Voting:** Ideal for colleges, universities, and other organizations to conduct internal elections securely and efficiently.
- **Corporate Decision-Making:** Useful for board meetings, shareholder voting, and other internal corporate polls where integrity is crucial.
- **Community & Club Elections:** Can be deployed to facilitate transparent elections for housing societies, local clubs, and NGOs.
- **Secure Access Control:** The foundational technology can be adapted for other security purposes, such as biometric-based attendance and authentication systems.

3. Introduction

3.1 Flow of Working (Block Diagram)

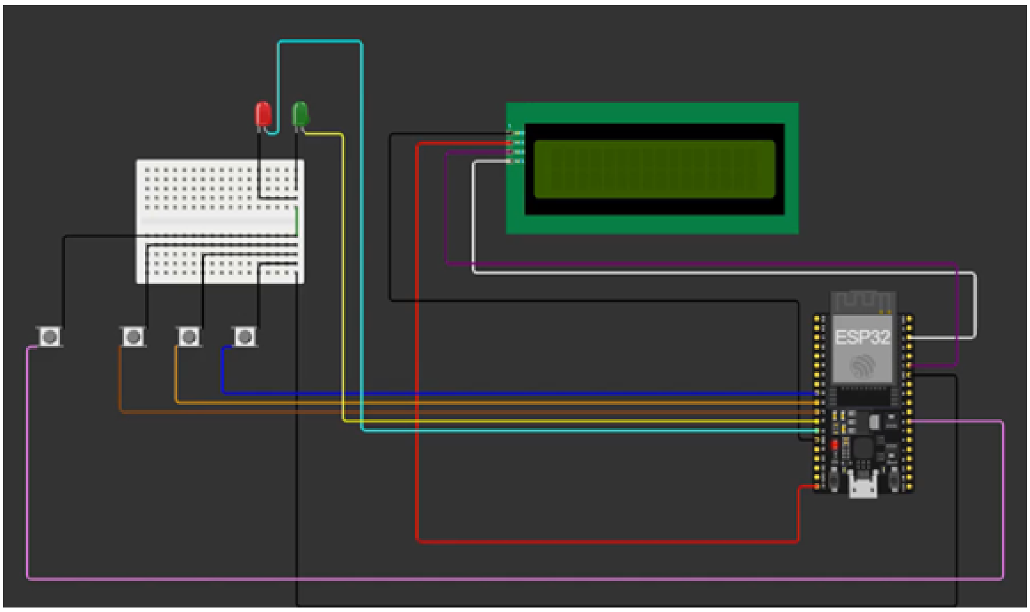


3.2 Pin-out Table

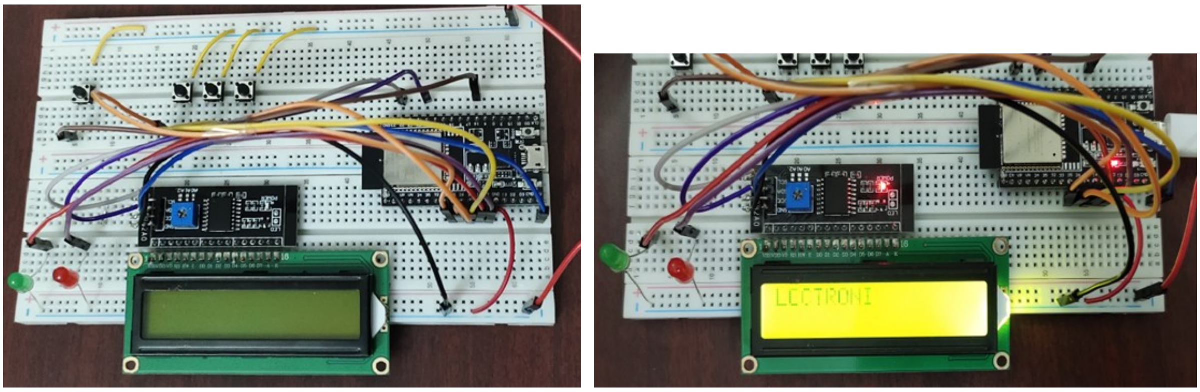
Table 1: Component Pin Mapping for the ESP32 EVM

Component	**ESP32 Pin**	**Description**
Green LED	GPIO 14	Blinks to confirm a successful vote has been registered.
Red LED	GPIO 12	Blinks along with the green LED during the result display animation.
Result Button	GPIO 16	When pressed, it displays the final results on the LCD and initiates the data upload to ThingSpeak.
Vote Button - Party 1	GPIO 25	Registers one vote for Party 1.
Vote Button - Party 2	GPIO 26	Registers one vote for Party 2.
Vote Button - Party 3	GPIO 27	Registers one vote for Party 3.
I2C SDA (for LCD)	GPIO 21	The serial data line for I2C communication with the LCD.
I2C SCL (for LCD)	GPIO 22	The serial clock line for I2C communication with the LCD.
LCD (8x2, I2C)	I2C (Address 0x27)	Displays startup messages, voting prompts, and the final results.

3.3 Circuit Diagram



3.4 Images of the EVM



(a) System in OFF state

(b) System in ON state

3.5 Embedded C Code

The final firmware for the project was developed in Embedded C++ using the Arduino IDE. The code includes custom functions for handling scrolling text on the LCD, managing Wi-Fi connectivity, processing votes, and uploading data to the ThingSpeak cloud platform.

Listing 1: Final Firmware for the ESP32 EVM

```
#include <WiFi.h>
#include <Wire.h>
#include <LiquidCrystal_I2C.h>

// === WiFi credentials ===
const char* ssid = "IDEAPAD 3028";
const char* password = "7964g9B@";

// === ThingSpeak ===
const char* server = "api.thingspeak.com";
const char* apiKey = "2P811K6IBDJEGMRT";

// === Pin setup ===
const int greenLED = 14;
const int redLED = 12;
const int resultButton = 16;
const int voteButton1 = 25;
const int voteButton2 = 26;
const int voteButton3 = 27;

// === Variables ===
```

```
int votesParty1 = 0;
int votesParty2 = 0;
int votesParty3 = 0;

bool votingActive = true;
bool resultsShown = false;

WiFiClient client;
LiquidCrystal_I2C lcd(0x27, 8, 2); // 8x2 LCD

// === Scroll text control ===
unsigned long lastScrollTime = 0;
int scrollIndex = 0;
bool scrollActive = false;
String scrollMessage = "";
int scrollRow = 0;
bool loopScroll = false;

void setup() {
    Serial.begin(115200);
    delay(2000);

    pinMode(greenLED, OUTPUT);
    pinMode(redLED, OUTPUT);
    pinMode(resultButton, INPUT_PULLUP);
    pinMode(voteButton1, INPUT_PULLUP);
    pinMode(voteButton2, INPUT_PULLUP);
    pinMode(voteButton3, INPUT_PULLUP);

    digitalWrite(greenLED, LOW);
    digitalWrite(redLED, LOW);

    Wire.begin();
    lcd.begin(8, 2);
    lcd.backlight();
    lcd.clear();

    startScroll("ELECTRONIC VOTING MACHINE", 0, true);

    unsigned long startTime = millis();
    while (millis() - startTime < 5000) {
```

```
        handleScroll(millis());
    }
    scrollActive = false;
    lcd.clear();

    connectToWiFi();

    startScroll("Cast your vote", 0, true);
}

void loop() {
    handleScroll(millis());

    if (digitalRead(resultButton) == LOW) {
        resultButtonAnimation();
        if (votingActive) {
            votingActive = false;
            resultsShown = true;
            lcd.clear();
            lcd.setCursor(0, 0);
            lcd.print("Voting");
            lcd.setCursor(0, 1);
            lcd.print("Results");
            delay(1500);

            showResults();
            sendToThingSpeak();
        } else if (resultsShown) {
            showResults(); // Re-show on next press
        }
        while (digitalRead(resultButton) == LOW);
        delay(300);
    }

    if (votingActive) {
        if (digitalRead(voteButton1) == LOW) {
            votesParty1++;
            registerVote("P1");
        } else if (digitalRead(voteButton2) == LOW) {
            votesParty2++;
            registerVote("P2");
```

```
        } else if (digitalRead(voteButton3) == LOW) {
            votesParty3++;
            registerVote("P3");
        }
    }
}

void connectToWiFi() {
    lcd.clear();
    startScroll("Connecting to WiFi...", 0, false);
    while (scrollActive) handleScroll(millis());

    WiFi.begin(ssid, password);
    while (WiFi.status() != WL_CONNECTED) {
        delay(500);
        Serial.print(".");
    }

    Serial.println("\nConnected to WiFi!");
    startScroll("Connected to WiFi", 0, false);
    while (scrollActive) handleScroll(millis());
    lcd.clear();
}

void startScroll(String text, int row, bool loopForever) {
    scrollMessage = "    " + text + "    ";
    scrollRow = row;
    scrollIndex = 0;
    scrollActive = true;
    loopScroll = loopForever;
    lastScrollTime = millis();
}

void handleScroll(unsigned long currentMillis) {
    if (!scrollActive) return;

    if (currentMillis - lastScrollTime >= 250) {
        lcd.setCursor(0, scrollRow);
        lcd.print(scrollMessage.substring(scrollIndex,
           scrollIndex + 8));
        scrollIndex++;
```

```
        lastScrollTime = currentMillis;
        if (scrollIndex > scrollMessage.length() - 8) {
            if (loopScroll) {
                scrollIndex = 0;
            } else {
                scrollActive = false;
                lcd.clear();
            }
        }
    }
}

void registerVote(String party) {
    blinkGreenLED();
    Serial.println("Vote registered for " + party);
    lcd.clear();
    lcd.setCursor(0, 0);
    lcd.print("Voted!");
    lcd.setCursor(0, 1);
    lcd.print(party);
    delay(1000);
    lcd.clear();

    startScroll("cast your vote", 0, true);

    while (digitalRead(voteButton1) == LOW || digitalRead(
       voteButton2) == LOW || digitalRead(voteButton3) == LOW);
    delay(300);
}

void blinkGreenLED() {
    digitalWrite(greenLED, HIGH);
    delay(300);
    digitalWrite(greenLED, LOW);
}

void resultButtonAnimation() {
    for (int i = 0; i < 2; i++) {
        digitalWrite(redLED, HIGH);
        delay(300);
        digitalWrite(redLED, LOW);
```

```
        digitalWrite(greenLED, HIGH);
        delay(300);
        digitalWrite(greenLED, LOW);
    }
}

void showResults() {
    Serial.println("\n==== FINAL RESULTS ====");
    Serial.print("Party 1: "); Serial.println(votesParty1);
    Serial.print("Party 2: "); Serial.println(votesParty2);
    Serial.print("Party 3: "); Serial.println(votesParty3);
    Serial.println("=======================");

    lcd.clear();
    lcd.setCursor(0, 0);
    lcd.print("Results:");
    lcd.setCursor(0, 1);
    lcd.print("P1:"); lcd.print(votesParty1);
    delay(2000);

    lcd.clear();
    lcd.setCursor(0, 0);
    lcd.print("P2:"); lcd.print(votesParty2);
    lcd.setCursor(0, 1);
    lcd.print("P3:"); lcd.print(votesParty3);
    delay(3000);
    lcd.clear();

    if (votingActive) {
        startScroll("Cast your vote", 0, true);
    } else {
        startScroll("RESULTS SENT TO CLOUD", 0, false);
        while (scrollActive) handleScroll(millis());
    }
    lcd.clear();
    startScroll("THANK YOU", 0, true);
}

void sendToThingSpeak() {
    if (WiFi.status() == WL_CONNECTED) {
        WiFiClient client;
```

```
        const int httpPort = 80;
        if (!client.connect(server, httpPort)) {
            Serial.println("Connection to ThingSpeak failed.");
            return;
        }
        String postData = "api_key=" + String(apiKey) +
                          "&field1=" + String(votesParty1) +
                          "&field2=" + String(votesParty2) +
                          "&field3=" + String(votesParty3);

        client.println("POST /update HTTP/1.1");
        client.println("Host: api.thingspeak.com");
        client.println("Connection: close");
        client.println("Content-Type: application/x-www-form-
           urlencoded");
        client.print("Content-Length: ");
        client.println(postData.length());
        client.println();
        client.println(postData);
        Serial.println("Sending data to ThingSpeak...");
    } else {
        Serial.println("WiFi not connected. Data not sent.");
    }
}
```

4. Result Display and Cloud Integration

Once the voting period is concluded, the system provides comprehensive feedback through its integrated peripherals and cloud connection. The process is initiated by pressing the dedicated Result Button.

4.1 On the LCD Display (8x2)

When the Result Button (connected to GPIO 16) is pressed for the first time after voting has occurred:

- The main voting process is immediately disabled to prevent any further votes.
- The LCD displays the final vote count for each party sequentially, ensuring clarity on the small screen. For example, the display will show:

```
P1: 01
P2: 02
```

followed by:

```
P2: 02
P3: 02
```

- After displaying the vote counts, a confirmation message, “RESULTS SENT TO CLOUD”, scrolls across the screen. This indicates that the data has been successfully transmitted and is now available for viewing on the ThingSpeak platform.
- **Note:** If the ESP32 is not connected to Wi-Fi, the results will still be shown on the LCD, but the subsequent message will be “WiFi not connected. Data not sent.” to inform the operator that the cloud upload failed.

4.2 On the ThingSpeak Platform

The vote counts are uploaded via a Wi-Fi connection to a pre-configured ThingSpeak channel using a unique API key.

- Each party’s vote count is mapped to a separate field within the ThingSpeak channel (e.g., Party 1 to Field 1, Party 2 to Field 2, etc.).
- This allows for a live, timestamped data stream that can be visualized as a graph or table, for example:

Time	**Party 1**	**Party 2**	**Party 3**
11:00 AM	1	2	2

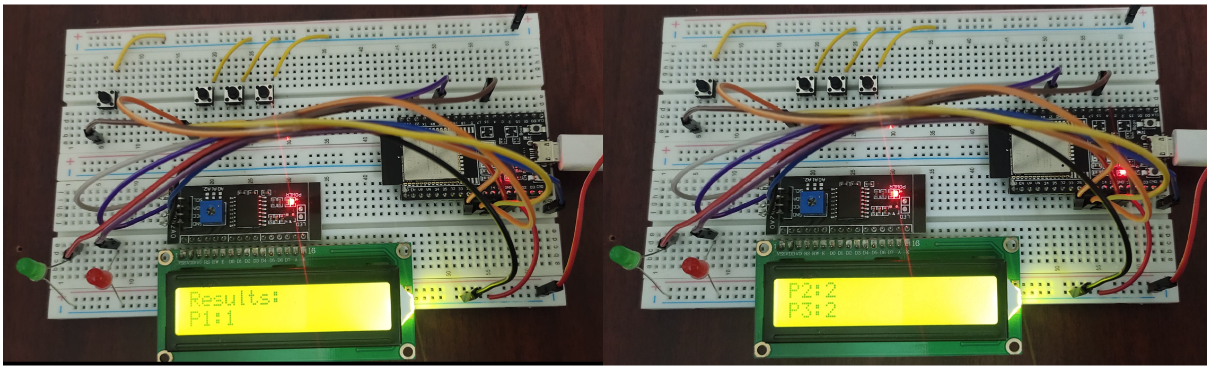
4.3 LED Indicators

Visual feedback is provided by the onboard LEDs:

- The Green and Red LEDs blink in unison to provide a clear signal that the voting process has officially ended and the system is now in the result-display mode.

If the result button is pressed again after the initial display, the system will repeat the process of showing the results on the LCD. This allows the results to be checked multiple times without needing to restart the device.

4.4 Results on LCD



4.5 ThingSpeak Visualization

. . . .

5. Summary

This project successfully demonstrates the design and implementation of an ESP32-based Electronic Voting Machine (EVM) with IoT capabilities. The system provides a secure, efficient, and transparent alternative to traditional paper-based voting for small-scale elections. The key aspects of the project are summarized below.

5.1 Hardware Components

The system is built using a minimal set of standard and cost-effective components:

1. ESP32 microcontroller with built-in Wi-Fi.
2. An 8x2 I2C LCD for user interface and data display.
3. Green and Red LEDs for immediate visual feedback.
4. Three push buttons designated for casting votes for three different parties.
5. A single, dedicated result button to finalize the voting process and trigger the result display.

5.2 Core Features

The firmware integrates several key functionalities to create a robust voting solution:

1. **Wi-Fi Connectivity:** The ESP32 connects to a local network to upload election results to the cloud.
2. **ThingSpeak Integration:** Provides seamless cloud storage and real-time monitoring of voting data.
3. **Scrolling Text Display:** A custom function allows for clear, readable messages on the compact 8x2 LCD.
4. **Vote Counting:** The system accurately counts and stores votes for three distinct parties (P1, P2, P3).
5. **Result Finalization:** A secure function to end the election, display the final tally, and transmit the results.

5.3 Program Flow

The operational logic of the system follows a clear, sequential process:

1. On startup, the system initializes all hardware components, displays a welcome message, and connects to the configured Wi-Fi network.
2. It then enters the main voting loop, continuously displaying a ”Cast your vote” prompt.
3. When a vote button is pressed, the system increments the appropriate party’s counter, blinks the green LED, and shows a confirmation message on the LCD.

4. When the result button is pressed, the firmware executes a finalization sequence: it performs an LED animation, disables further voting, displays the results on the LCD, and uploads the data to ThingSpeak.

5. After the results are processed, a final ”THANK YOU” message is displayed, concluding the session.

5.4 Helper Functions

The code is modularized with several key helper functions to manage specific tasks:

1. A text scrolling function designed for the limited space of the 8x2 LCD.

2. A voter registration function that handles vote counting and provides visual feedback.

3. Dedicated routines for displaying the final results on the LCD.

4. A ThingSpeak data upload function that sends the results via an HTTP POST request.

. . . .

6. Future Enhancements and Possible Modifications

This section outlines practical and impactful extensions you can implement to enhance security, scalability, usability, and inclusivity.

6.1 Biometric Authentication

Integrate a fingerprint sensor (e.g., R307, GT-521F, or similar) to authenticate voters before allowing them to cast a vote. Biometric authentication prevents duplicate voting and strengthens voter identity verification. Typical steps:

- Enroll voter fingerprints in a secure local or cloud-stored database.
- Authenticate fingerprint on each voting attempt.
- Log authentication events with timestamps for auditability.

6.2 Scalability

Scale the system to support multiple booths and centralized result aggregation:

- Use MQTT, LoRaWAN, or a RESTful API to aggregate data from distributed devices.

- Replace ThingSpeak with a scalable backend (e.g., Firebase, AWS IoT, or a custom server) for large deployments.
- Implement device provisioning and OTA (over-the-air) firmware updates.

6.3 Enhanced Security

Improve data integrity and privacy by:

- Enabling HTTPS/TLS for data transmission.
- Encrypting stored logs and API keys using AES or secure element chips.
- Considering blockchain-based tamper-evident logging for an immutable audit trail.

6.4 Touchscreen Interface

Upgrade the UI by replacing the 8x2 LCD and push buttons with a capacitive or resistive touchscreen:

- Provide a clearer candidate list and confirmation screens.
- Add on-screen accessibility options (font size, contrast).

6.5 Mobile App Integration

Develop a companion mobile app or web dashboard to:

- Visualize live results and historical trends.
- Manage device network settings and API keys remotely.
- Provide administrative controls and role-based access.

6.6 Power Management

Add resilience and energy efficiency:

- Implement battery backup and UPS-based switching.
- Use low-power modes between sessions to extend battery life.
- Add power-fail-safe logging to avoid data loss.

6.7 Accessibility Features

Make the system inclusive:

- Add audio prompts and tactile buttons for visually impaired voters.
- Provide language selection and simple confirmation steps.

6.8 Testing & Validation

For production-ready deployments, include:

- Unit and integration tests for firmware.
- Security audits and penetration testing.
- Usability testing with a diverse user group.

. . . .

7. References

[1] Espressif Systems. (2023). *ESP32 Technical Reference Manual*. Retrieved from https://docs.espressif.com/projects/esp-idf/en/latest/esp32/

[2] Hitachi. (n.d.). *HD44780U LCD Controller/Driver Datasheet*.

[3] MathWorks. (2023). *ThingSpeak Documentation*. Retrieved from https://www.mathworks.com/help/thingspeak/

[4] Espressif Systems. (n.d.). *Arduino core for the ESP32*. GitHub Repository. Retrieved from https://github.com/espressif/arduino-esp32

[5] Marco Schwartz. (n.d.). *LiquidCrystal_I2C Library*. GitHub Repository. Retrieved from https://github.com/johnrickman/LiquidCrystal_I2C