Temperature Monitoring System using ESP32 and Winstar LCD

ab4674 as6198

April 30, 2025

Abstract

This report presents the development of a real-time temperature monitoring system using an ESP32(Embedded Serial Peripheral 32-bit Microcontroller) microcontroller, a TMP36 temperature sensor, and a WINSTAR LCD(Liquid Crystal Display) display. The system was designed to convert analog sensor readings into voltage and temperature values, which were displayed on the LCD using I^2C(Inter-Integrated Circuit) communication. It successfully demonstrated low-power continuous monitoring, logical threshold activation via an LED, and accurate data conversion using voltage-to-temperature equations. Comparison with a commercial thermometer showed that the readings remained within a reasonable margin of error. The structured development process included component selection, PCB(Printed Circuit Board) design, system assembly, testing, and performance evaluation. The project concluded with proposals for future improvement including wireless connectivity and improved calibration. Overall, the system validates the feasibility of building simple, cost-effective, and scalable embedded monitoring systems.

Contents

1 Introduction **2**
- 1.1 Context & Background . . . 2
- 1.2 Objective . . . 2
- 1.3 Scope . . . 2

2 Methodology **3**
- 2.1 Experimental Setup . . . 3
  - 2.1.1 Hardware Used . . . 3
- 2.2 Circuit Design and Assembly . . . 3
  - 2.2.1 Circuit Schematic . . . 3
- 2.3 Software Implementation . . . 4
  - 2.3.1 Data Acquisition . . . 4
- 2.4 Testing Procedure and Performance Evaluation . . . 4

3 System Performance & Accuracy **5**
- 3.1 Key Equations . . . 5
  - 3.1.1 Voltage Calculation . . . 5
  - 3.1.2 Temperature Conversion . . . 5
- 3.2 Comparison of Temperature Readings . . . 6
- 3.3 Performance Analysis . . . 7

4 Conclusion **8**

A Supplementary Diagrams **9**
- A.1 Circuit Schematic . . . 9

B Arduino Code **10**

C ESP32 Pin Configuration Reference **12**

D List of Acronyms **13**

Bibliography **13**

Chapter 1

Introduction

1.1 Context & Background

In today's technology-driven world, the demand for microcontrollers in real-time monitoring applications has grown significantly. Users require real-time applications on functions such as temperature sensing. This report represents a system used for temperature monitoring using an ESP32 Arduino microcontroller and a Winstar LCD.

1.2 Objective

The report aims to achieve the following goals:

- Develop a system that accurately displays temperature in real-time using a low-power source.
- Utilize I2C, the most common communication protocol, to interface between the LCD and the ESP32.
- Evaluate the system performance using experimental validation.

1.3 Scope

The scope is restricted within the following constraints:

- The hardware is limited to the ESP32 or similar Arduino hardware such as the Arduino Nano.
- The Arduino IDE(Integrated Development Environment) framework is used for programming and development.
- Data visualization is performed using the Winstar LCD.
- Performance and validation are focused on testing the TMP36 sensor for reasonable temperature ranges.

Chapter 2

Methodology

2.1 Experimental Setup

The experimental phase began with sourcing and preparing all system components.

2.1.1 Hardware Used

The hardware that would be used is:

- ESP32 Micro Controller
- TMP36 Temperature Sensor
- A WINSTAR LCD
- USB-C Cable
- Arduino IDE

2.2 Circuit Design and Assembly

The circuit was assembled by connecting the ESP32 to all peripheral components based on the planned schematic. This included linking the TMP36 temperature sensor and the WINSTAR LCD to the correct GPIO pins, as well as incorporating a trimmer and resistor to fine-tune voltage levels. Connections were made on a breadboard for prototyping purposes, though the design can also be implemented on a PCB for increased reliability.

Place the circuit on a breadboard for easier implementation. However, it can also be done on a PCB if preferred.

2.2.1 Circuit Schematic

The full schematic diagram demonstrating the connections between all components—ESP32, WINSTAR LCD, trimmer, LED, resistor, and TMP36—is provided in Appendix A.1.

2.3 Software Implementation

The software was implemented using the Arduino IDE. The program was structured to initialise the system, read sensor values, and display temperature in real time. The main program flow should be:

1. Initialize the ESP32 and LCD.
2. Read the data from the TMP36.
3. Convert the analog signals to voltage and from voltage to temperature.
4. Get those temperature readings and put them onto the LCD in real time.
5. Do this repetitively in a loop.

2.3.1 Data Acquisition

Data acquisition was configured to occur at one-second intervals to ensure real-time temperature monitoring. Communication between the ESP32 and the LCD display was achieved using the I^2C protocol, which allowed for efficient data transmission using minimal wiring.

2.4 Testing Procedure and Performance Evaluation

Testing was conducted in a controlled indoor environment to ensure stable temperature conditions. This helped to minimize external environmental factors that could influence the sensor's accuracy during data collection. The independent variable in this setup was the temperature, while the dependent variables were the recorded temperature outputs from the ESP32 system compared to the commercial digital thermometer.

To assess the system's performance, recorded temperatures from both devices were measured under the same conditions and compared. The data was then exported to Microsoft Excel for comparative analysis. A graph was created to compare the temperature responses and evaluate the accuracy of the ESP32 system relative to the commercial reference.

Chapter 3

System Performance & Accuracy

3.1 Key Equations

3.1.1 Voltage Calculation

The TMP36 temperature sensor outputs an analog voltage that corresponds to the ambient temperature. This analog signal is read by the microcontroller's ADC (Analog-to-Digital Converter), which converts it into a digital value ranging from 0 to 4095 when using a 12-bit ADC (as in many ARM-based microcontrollers). To convert this digital value into an actual voltage, we apply the following equation:

$$V = S \times \frac{3.3}{4095} \tag{3.1}$$

where:

- V is the resulting voltage in volts,
- S is the sensor's analog output value as read by the ADC (a number between 0 and 4095),
- 3.3 is the reference voltage of the system (assuming a 3.3V logic board),
- 4095 is the maximum ADC value for a 12-bit resolution.

This step is essential for translating raw sensor readings into a usable voltage output, which is then used to determine the temperature.

3.1.2 Temperature Conversion

Once the voltage output from the sensor is determined, we can convert this voltage into a temperature value. The TMP36 sensor follows a linear relationship where 0.5V corresponds to 0°C, and each additional 10mV represents a 1°C increase. Thus, the conversion from voltage to temperature in degrees Celsius is calculated using:

$$T = (V - 0.5) \times 100 \tag{3.2}$$

where:

- T is the temperature in degrees Celsius,

- V is the voltage obtained from the ADC conversion.

Here, 0.5 is subtracted to account for the 500mV offset of the TMP36 at 0°C, and the result is multiplied by 100 to convert volts to degrees Celsius (as 10mV = 0.01V per °C). This equation enables real-time temperature monitoring using simple linear scaling.

3.2 Comparison of Temperature Readings

Table 3.1: Comparison of Temperature Readings

Time (s)	**Commercial Sensor (°C)**	**Temperature Reading (Ours) (°C)**
0	22	17.6
1	22	17.5
2	22	17.0
3	22	17.0
4	22	17.0
5	22	17.6
6	22	17.5
7	22	17.6
8	22	17.4
9	22	17.0
10	22	17.0
11	22	17.5
12	22	16.9
13	22	17.6
14	22	17.5
15	22	17.0
16	22	17.5
17	22	17.5
18	22	17.6
19	22	17.5
20	22	17.5
21	22	17.6
22	22	17.5
23	22	17.5
24	22	17.6
25	22	17.7
26	22	17.6
27	22	17.5
28	22	17.0
29	22	17.0
30	22	17.0
31	22	17.6
32	22	17.5
33	22	16.9

Continued on next page

Table Continued from Previous Page

Time (s)	Commercial Sensor (°C)	Temperature Reading (Ours) (°C)
34	22	17.6
35	22	17.5
36	22	17.0
37	22	17.5
38	22	17.0
39	22	17.2
40	22	17.5
41	22	17.6
42	22	17.3
43	22	17.2
44	22	16.9
45	22	16.8
46	22	17.5
47	22	17.0
48	22	17.5
49	21	17.0
50	21	17.2
51	21	17.5
52	22	17.6
53	22	17.6
54	21	17.3
55	21	17.2
56	21	16.9
57	21	16.8
58	21	17.5
59	21	17.3
60	21	16.9

3.3 Performance Analysis

The LED was programmed to activate at 20°C, and this condition was successfully met during testing, confirming accurate implementation of the control logic. The LCD display functioned correctly, displaying real-time temperature readings, however due to the TMP36 not being perfectly accurate the temperature readings of our system were slightly lower than the **commercial** temperature sensor. The temperature sensor recorded between a **16.9°C and 17.6°C**, which, while lower than the expected 22°C, the value still remains within a reasonable deviation given the specifications of the sensor. A detailed comparison of these readings can be seen in Table 3.1.

Chapter 4

Conclusion

The project successfully implemented the TMP36 and WINSTAR into the ESP32 to create a system for temperature monitoring. The LED and trimmer were useful in adding conditions or tweaking contrast for specific light conditions, respectively. The temperature findings were slightly off from the commercial digital thermometer; however, the values are within an acceptable range considering the limitations of the TMP36. Future improvements include adding wireless connections to make the system more effective as an IoT device, using better calibration techniques to improve result accuracy, and testing different sensors to evaluate their performance within the system. Additionally, improvements could be made to the Arduino code to enable direct and real-time data logging onto an Excel sheet, possibly using MATLAB.

Appendix A

Supplementary Diagrams

A.1 Circuit Schematic

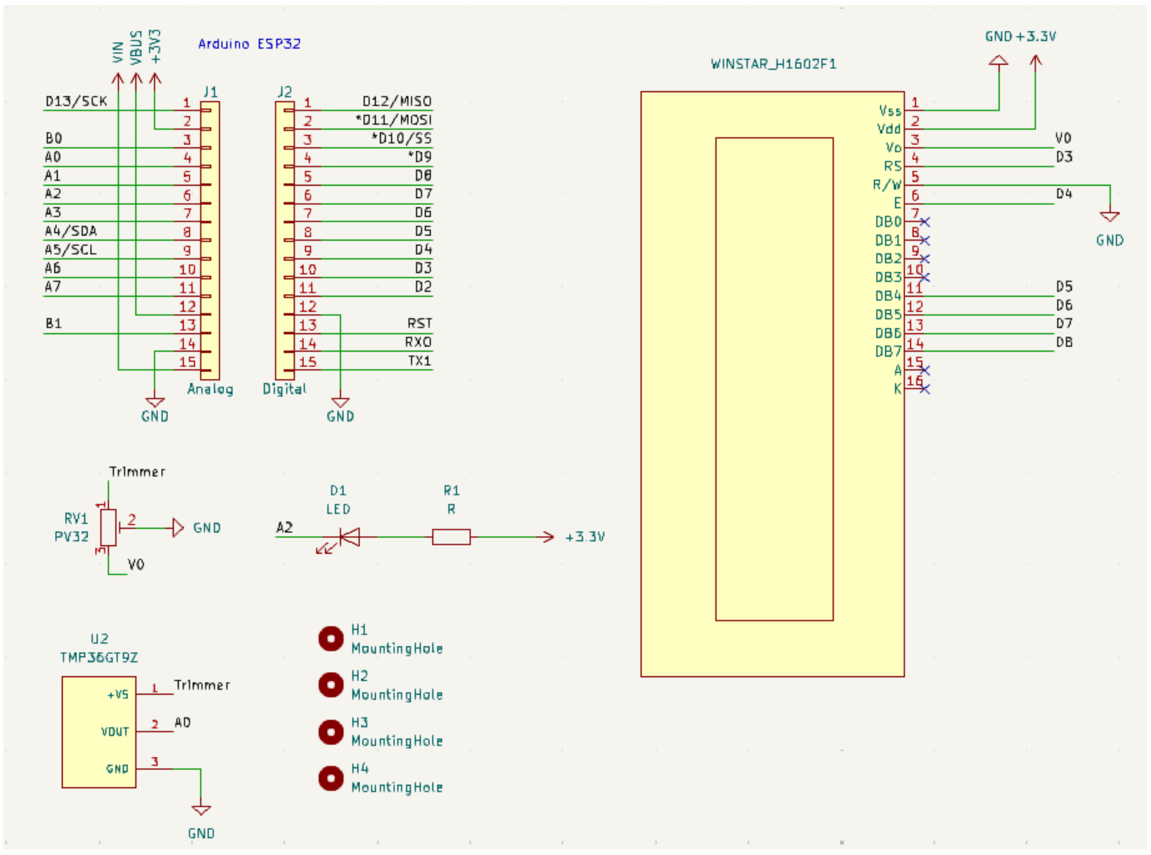

Figure A.1: Full schematic of the ESP32 temperature monitoring system showing connections to the WINSTAR LCD, TMP36 sensor, LED, and supporting components.

Appendix B

Arduino Code

Listing B.1: Full Arduino Source Code for ESP32 Temperature Monitoring System

```
#include <LiquidCrystal.h> // include the library for the LCD

// define some macros connected to ESP32
#define RS 3
#define E 4
#define D4 5
#define D5 6
#define D6 7
#define D7 8

// define peripheral connections
#define TMP36_PIN A0
#define LED_PIN A2

// start the LCD
LiquidCrystal lcd(RS, E, D4, D5, D6, D7);

void setup() {
  // create pin modes for components
  pinMode(TMP36_PIN, INPUT);
  pinMode(LED_PIN, OUTPUT);

  // make a little start up message
  lcd.begin(16, 2);
  lcd.print("System Ready");
  delay(1000); // gives user time to read it
  lcd.clear(); // clear LCD for further messages
}

void loop() {
  int sensorValue = analogRead(TMP36_PIN); // get sensor value
  float voltage = sensorValue * (3.3 / 4095.0); // convert analog to voltag
  float temperature = (voltage - 0.5) * 100.0; // now convert voltage to C
  // get the LED state and set a condition for it (>20C)
  bool ledState = (temperature > 20);
  digitalWrite(LED_PIN, ledState);

  // update LCD
  lcd.clear();
  lcd.setCursor(0, 0);
  lcd.print("TEMP: ");
  lcd.print(temperature, 1); // display temperature with 1 decimal place
  lcd.print(" C");

  // move cursor to next line for LED condition
  lcd.setCursor(0, 1);
  lcd.print("LED: ");
  lcd.print(ledState ? "ON " : "OFF");

  delay(500); // short delay to prevent flicker
}
```

Appendix C

ESP32 Pin Configuration Reference

Pin Configuration Tables

The following image summarises the pin configurations and connections used in this project, based on the actual hardware setup and referencing ESP32-compatible peripherals:

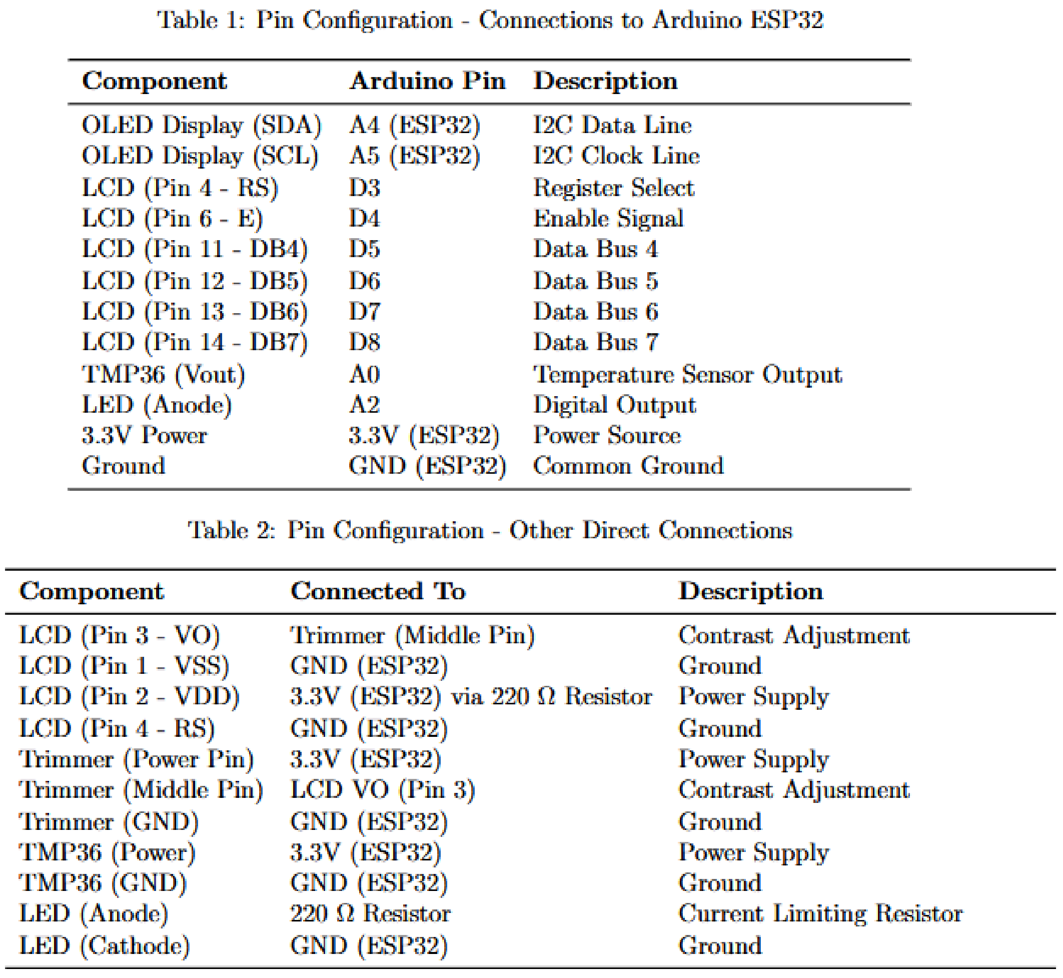

Table 1: Pin Configuration - Connections to Arduino ESP32

Component	Arduino Pin	Description
OLED Display (SDA)	A4 (ESP32)	I2C Data Line
OLED Display (SCL)	A5 (ESP32)	I2C Clock Line
LCD (Pin 4 - RS)	D3	Register Select
LCD (Pin 6 - E)	D4	Enable Signal
LCD (Pin 11 - DB4)	D5	Data Bus 4
LCD (Pin 12 - DB5)	D6	Data Bus 5
LCD (Pin 13 - DB6)	D7	Data Bus 6
LCD (Pin 14 - DB7)	D8	Data Bus 7
TMP36 (Vout)	A0	Temperature Sensor Output
LED (Anode)	A2	Digital Output
3.3V Power	3.3V (ESP32)	Power Source
Ground	GND (ESP32)	Common Ground

Table 2: Pin Configuration - Other Direct Connections

Component	Connected To	Description
LCD (Pin 3 - VO)	Trimmer (Middle Pin)	Contrast Adjustment
LCD (Pin 1 - VSS)	GND (ESP32)	Ground
LCD (Pin 2 - VDD)	3.3V (ESP32) via 220 Ω Resistor	Power Supply
LCD (Pin 4 - RS)	GND (ESP32)	Ground
Trimmer (Power Pin)	3.3V (ESP32)	Power Supply
Trimmer (Middle Pin)	LCD VO (Pin 3)	Contrast Adjustment
Trimmer (GND)	GND (ESP32)	Ground
TMP36 (Power)	3.3V (ESP32)	Power Supply
TMP36 (GND)	GND (ESP32)	Ground
LED (Anode)	220 Ω Resistor	Current Limiting Resistor
LED (Cathode)	GND (ESP32)	Ground

Figure C.1: Pin Configuration Tables – Connections to and from the Arduino ESP32

Appendix D

List of Acronyms

ADC Analog-to-Digital Converter

ESP32 Embedded Serial Peripheral 32-bit Microcontroller

LCD Liquid Crystal Display

I^{2}C Inter-Integrated Circuit

IDE Integrated Development Environment

PCB Printed Circuit Board

Bibliography

[1] Winstar Display Co. *16x2 Character LCD Display Module*. Available at: `https://www.winstar.com.tw/products/character-lcd-display-module/lcd-display-16x2.html` [Accessed 19 Feb 2024].

[2] Analog Devices Inc. *TMP35, TMP36, TMP37 Datasheet*. Available at: `https://www.analog.com/media/en/technical-documentation/data-sheets/tmp35_36_37.pdf` [Accessed 19 Feb 2024].

[3] Arduino. *Nano 33 IoT Datasheet*. Available at: `https://docs.arduino.cc/resources/datasheets/ABX00083-datasheet.pdf` [Accessed 19 Feb 2024].

[4] NXP Semiconductors. *I^2C Bus Specification and User Manual*. Available at: `https://www.nxp.com/docs/en/user-guide/UM10204.pdf` [Accessed 19 Feb 2024].

[5] Arduino. *Wire Library Reference*. Available at: `https://www.arduino.cc/reference/en/language/functions/communication/wire/` [Accessed 19 Feb 2024].

[6] Analog Devices Inc. *TMP36 Sensor Application Notes*. Available at: `https://www.analog.com/en/analog-dialogue/articles/temperature-sensors.html` [Accessed 19 Feb 2024].