Datasheet: Arduino-based System

ab4674 as6198

February 24, 2025

1 Introduction

This datasheet describes the key features, circuit schematic, software implementation and performance testing features of an Arduino-based LCD Temperature Sensing System. The system should be able to track the temperature in a room and display it on an LCD, furthermore it should be able to use an LED to turn on when a certain temperature threshold is crossed. A trimmer is included to control the LCD contrast to allow flexibility in different light conditions. **Note:** If the LCD cannot be seen within the existing lighting even after changing the trimmer use a 5V power supply for greater power.

1.1 LCD Testing

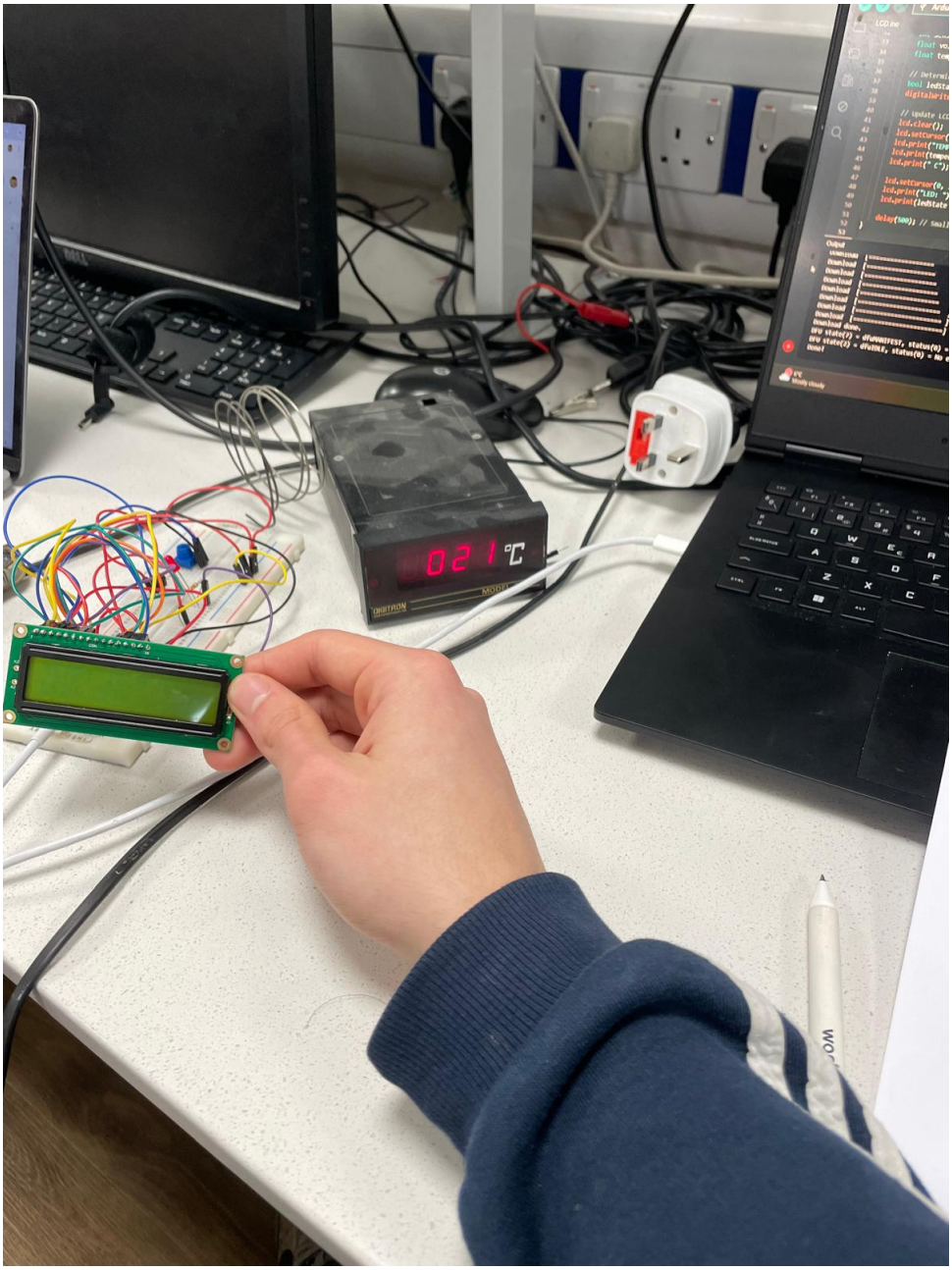

Figure 1: Testing of the LCD display during system setup.

1.2 System in Operation

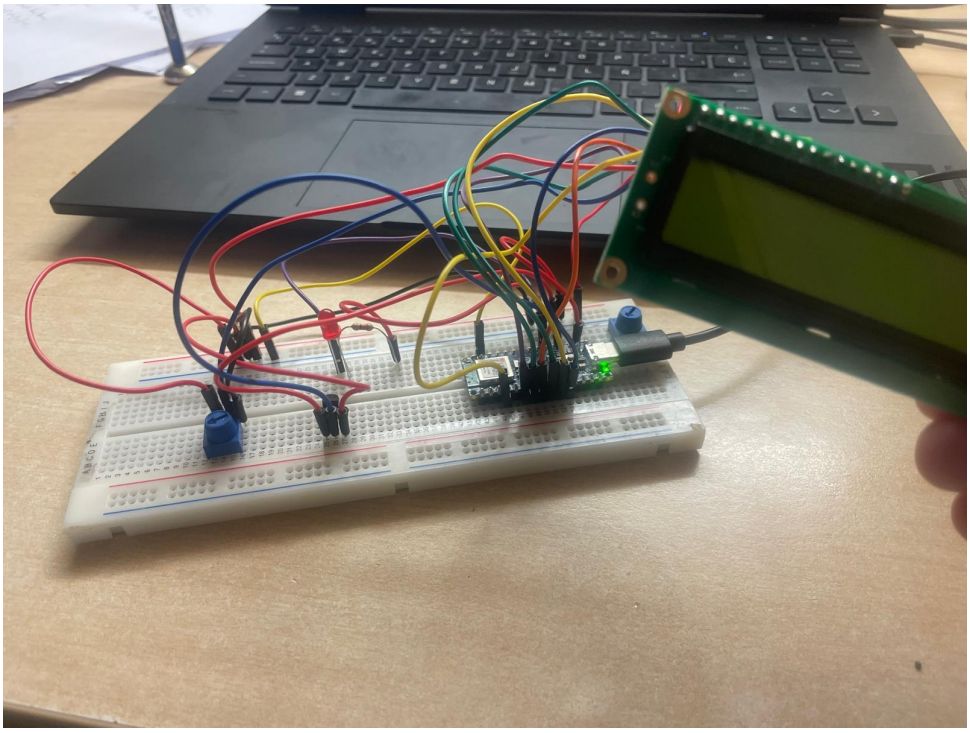

Figure 2: LCD Temperature Sensing System in operation.

2 System Overview

2.1 Key Features

- Microcontroller: **Arduino ESP32**
- Power Supply: 3.3V
- Interfaces: I2C
- Components Used: 10kohm Trimmer, LED, Resistor 220 ohms, Winstar H1602F(LCD 16X2).
- Programming Language: C++ (Arduino IDE)

2.2 Block Diagram

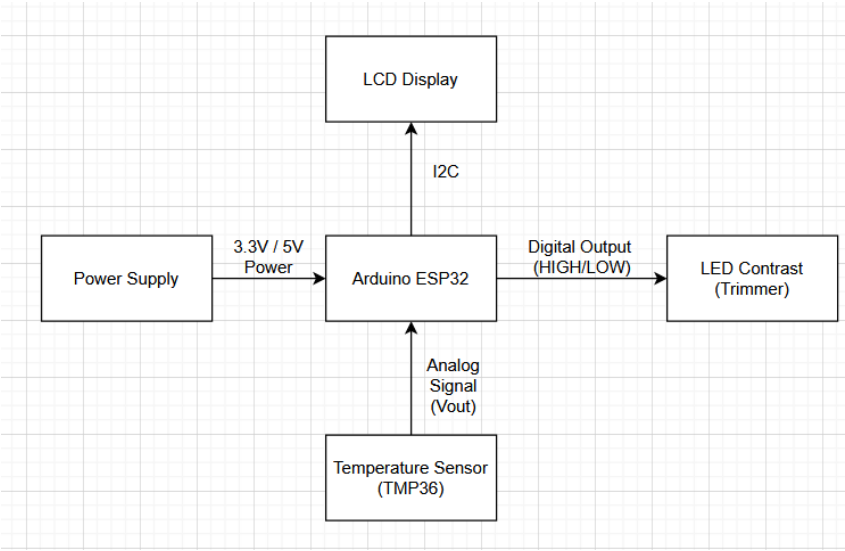

Figure 3: System Block Diagram showing the ESP32, OLED Display, Temperature Sensor, and LED Indicator

3 Key Equations

3.1 Voltage Calculation

The voltage is calculated based on the sensor's analog reading:

$$V = S \times \frac{3.3}{4095} \tag{1}$$

where V is the voltage in volts and S is the sensor's analog output value.

3.2 Temperature Conversion

The temperature in Celsius is then derived from the voltage as:

$$T = (V - 0.5) \times 100 \tag{2}$$

where T is the temperature in degrees Celsius.

4 Pin Configuration and Wiring

Tables 1 and 2 list the wiring connections between the Arduino ESP32 and its components.

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

5 Circuit Schematic

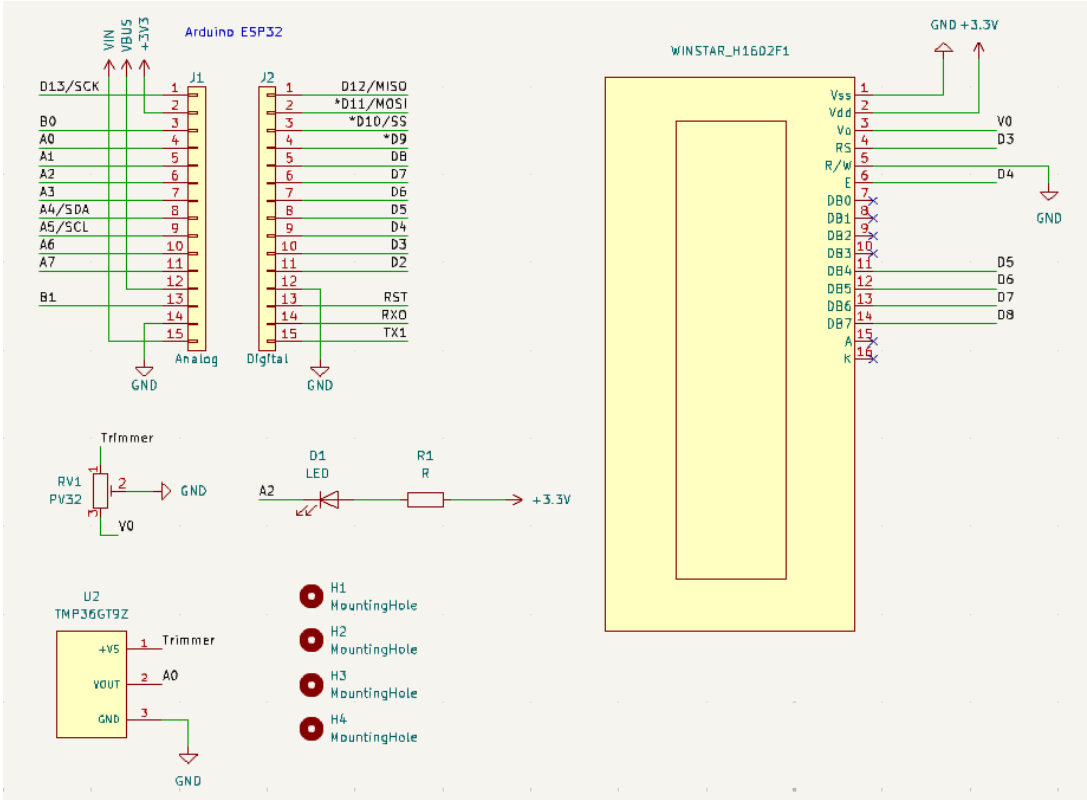

Figure 4: Circuit Diagram

6 Software Implementation

The following code snippet initializes the hardware and displays the temperature and the LED condition on the LCD. The LED condition is when the room temperature is above 20 degress Celsius the LED turns HIGH. The trimmer is used to help **contrast adjustment** thats controlled by the hardware **Pin 3**.

```
#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 voltage
    float temperature = (voltage - 0.5) * 100.0;  // now convert the voltage to Celsi

    // get the LED state and set a condition for it (>20C)
    bool ledState = (temperature > 20);
    digitalWrite(LED_PIN, ledState);

    // although LCD been cleared do it again
    lcd.clear();
    lcd.setCursor(0, 0);
    lcd.print("TEMP: ");
    lcd.print(temperature, 1); // display temperature onto LCD with one 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); // create a delay - avoiding flickering
}
```

7 Performance and Testing

7.1 Expected Performance

- The LCD should display real time temperature readings and the LED status - The LED should turn ON above 20 degress celsius and OFF below that. - The temperature sensor should read values between **-40°C to 150°C**, this follows the TMP36 datasheet, however reaching these values may damage the component and the values may become inaccurate.

7.2 Performance Comparison

To validate the accuracy of the system, measurements were compared against a commercial temperature sensor. The results are presented in The results are presented in Table 3.

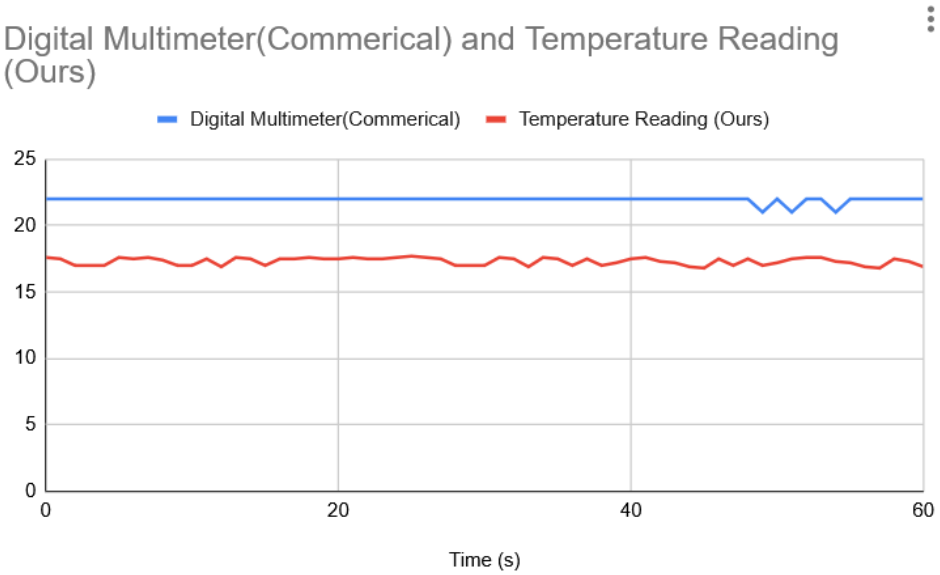

Figure 5: Comparison of system performance against a commercial temperature sensor. See Table 3 for detailed values.

Table 3: 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

Continued on next page

Table Continued from Previous Page

Time (s)	Commercial Sensor (°C)	Temperature Reading (Ours) (°C)
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
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

7.3 Testing Summary

The LED was expected to turn ON only at 20 which resulted true, represeting accurate implementation of the LED function. 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 slighter 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.

8 Conclusion

This datasheets documents the realization, implementation, performance of an Arduino-based LCD Temperature Sensing System. It successfully can perform the task of displaying the current room temperature to an acceptable standard, while also using the LED and trimmer implementations.

Although minor deviations can be seen with the current implementation of the system, which exist due to inherent limitations of the TMP36 sensor.

For future improvements, a more proficient temperature sensor can be used and a system for data logging using a software such as MATLAB would be effective. In addition, wireless communication with the ESP32 built in bluetooth, would make the system more suitable for the IoT applications.