Design Process of Room Temperature Monitor

Giacomo Cappelletto

BU ID: U91023753

1 Summary

This report presents a portable room temperature monitor built with a TMP36 sensor and Arduino Uno, displaying readings on a 16×2 I^2C LCD and triggering LEDs and a buzzer when a running-average setpoint is exceeded. It outlines the design approachsensor integration, firmware development, circuit assembly, and custom enclosure CAD modeling, The key outcomes of this project: $\pm 0.5°C$ linear accuracy over the operational range and approximately 6.7 h runtime on a single 9 V battery. These results demonstrate a low-cost, energy-efficient solution for real-time temperature monitoring and alerting.

2 Introduction

Maintaining a comfortable and stable indoor temperature is crucial to modern building management and directly impacts both energy consumption as well as the wellbeing of those who occupy it. Traditional thermostats regulate heating and cooling systems based on set threshold temperatures, but commercial units can be costly and therefore inaccessible for small-scale or experimental applications. In response to growing concerns over energy efficiency and sustainability, it is important to explore low-cost, modular solutions that allow fine-grained temperature monitoring and control in real world environments.

This project addresses the need for an accessible, do-it-yourself room temperature monitoring system by integrating a TMP36 analog temperature sensor, a 16×2 I^2C LCD, and an Arduino UNO microcontroller. By automating the measurement and display of ambient temperature, the device can inform control actions in the event loop (such as triggering a fan or heater), enabling users to avoid over-adjusting the temperature in their spaces, reducing unnecessary energy use. A 9 V battery, switch, green and red LEDs, and a buzzer provide basic power, status indication, and alerts without reliance on wall plug electricity.

This project has two main aims: first, to replicate the core functionality of a simple thermostat using components and techniques covered in our course. Second, to develop skills in assembling electronic hardware, writing microcontroller code (in C++), and designing enclosures in CAD. For that part of the project enclosure, lid, battery holder, and internal baseplate were modeled and fabricated to store and organize all components, wired using 22 AWG and female-to-female jumper cables for I^2C data connections.

The purpose of this report is to document the design process, from component selection and design to circuit assembly and firmware development, and to evaluate the system's performance against expected objectives. By following the design methodology outlined here, we demonstrate how simple and easily sourceable parts can be combined into a cohesive temperature monitoring tool.

3 Design elements

Explain your design decisions. Address each of the following:

A. **List Of Used Components**

- 1 × Injection Mold ABS Enclosure (Bottom and Lid)
- 1 × Transparent Acrylic Lase Cut Base Plate
- 8 × Polycarbonate Flat Top Phillips Screws
- 8 × Polycarbonate Bolts
- 8 × Metal Round Head Phillips Screws
- 8 × Plastic 0.5 mm spacers
- 1 × $9V$ Battery
- 1 × PLA FDM Printed $9V$ Battery holder
- 1 × Arduino UNO microcontroller
- 1 × Alphanumeric 16×2 I^2C LCD with IIC
- 1 × TMP36 Analog Temperature sensor
- 1 × Piezo Capsule Buzzer
- 1 × Red LED
- 1 × Green LED
- 1 × 2 way switch
- 2 × Female-Female $4''$ jumper wires
- $15 \times 6''$ 22 AWG Wires (White and Red)
- 1 × 220Ω resistor
- 1 × $1k\Omega$ resistor
- 2 × Twist Nut Caps
- 2 × Spade Connectors

B. **Precision measurements**

Table 1: Relevant dimensions of major components (see Appendix A for Figures 4 - 10)

Item	Fig. ID	W [mm]	L [mm]	H [mm]	⌀ [mm]
Full Assembly	1	123,8	146,1	63,1	-
Injection Mold ABS Enclosure Bottom	4	119,0	146,1	57,4	-
Injection Mold ABS Enclosure Lid	5	119,0	120,1	5,5	-
Transparent Acrylic Laser-Cut Base Plate	6	95,0	96,1	9,0	-
Arduino UNO Microcontroller	7	53,3	74,9	15,2	-
9 V Battery and PLA-Printed Holder	8	29,7	48,8	21,0	-
Alphanumeric 16×2 I^2C LCD	9	36,0	80,0	22,0	-
TMP36 Analog Temperature Sensor	10	-	-	14,6	2,5

Full Table in Appendix (see Section A)

C. CAD drawings

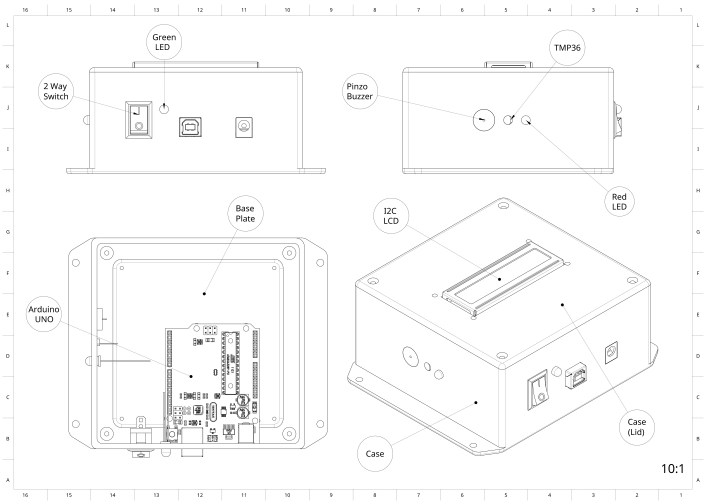
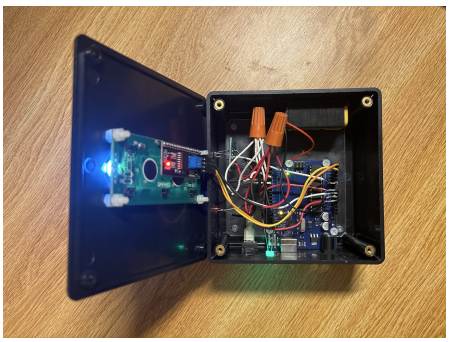


Figure 1: CAD assembly of the prototype.

Figures 4 - 10 in Appendix (see Section B)

D. Prototype photographs

(a) With lid



(b) Without lid

Figure 2: Top views of the working prototype.

E. Purpose of using an Arduino board

In this project, the Arduino UNO serves as the central controller: it reads the TMP36's analog voltage on pin A0 using its 10-bit ADC ($0 - 1023$ counts for $0 - 5V$), converts that value to temperature in °C and °F, and drives the 16×2 I^2C LCD over SDA/SCL (pins A4/A5) via the LiquidCrystal_I2C library. Inside the `loop()` function (sampling once per second), the firmware compares the measured running average temperature (over 5 seconds) against a setpoint and, when thresholds are crossed, toggles digital pin D3 to light the red (220 Ω series) LED and pulses pin D4 to sound the buzzer. This programmable feedback mechanism both displays real-time readings and issues visual/audible alerts, all managed by the Arduino's microcontroller. Furthermore, we are able to execute arithmetic operations such as the running average of the temperature over a set interval, effectively addressing the initial issue of over regulation of temperature in thermostats.

F. **Wiring diagram and methods**

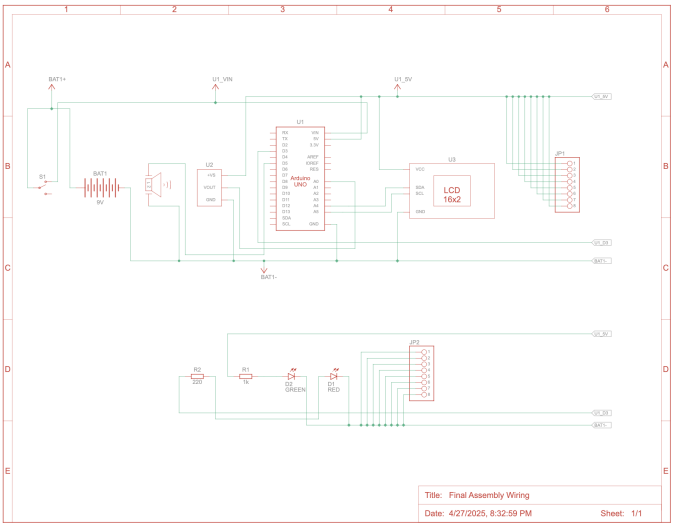


Figure 3: Circuit wiring diagram.

Figure 3 shows the final assembly. All GND and positive lines, as well as the analog TMP36, are soldered to 90° male header pins on the Arduino Uno for a secure, low-profile connection. The SDA/SCL lines to the 16×2 I^2C LCD use female-female jumper wires for easy removal. Two twist-on wire nuts combine the multiple battery positive and negative leads, permitting quick disassembly and reconfiguration. The power switch is fitted with insulated spade connectors to maintain a clean, removable interface.

G. **Wire gauge and resistor values**

All main power and signal lines use 22 AWG to keep series resistance low and provide good strength since the LCD lines need to bend when removing the lid. The I^2C SDA/SCL connections use thinner jumper wires (26 AWG) since their $< 1m\,A$ currents tolerate higher resistance and allow quick reconnection.

Series resistors: green LED $R_G = 1k\Omega$, red LED $R_R = 220\,\Omega$.

By KVL in each LED loop:

$$5V - I_G R_G - V_{F,G} = 0 \quad \Longrightarrow \quad I_G = \frac{5 - 2.1}{1000} = 2.9\,mA,$$

$$5V - I_R R_R - V_{F,R} = 0 \quad \Longrightarrow \quad I_R = \frac{5 - 2.0}{220} = 13.6\,mA,$$

where $V_{F,G} = 2.1\,\mathrm{V}$ and $V_{F,R} = 2.0\,\mathrm{V}$ are the LED forward voltages.

H. **Internal power supply** A 9 V battery was chosen for its compact size, ease of integration and replacement. Typical 9 V batteries have a capacity of $C \approx 500\,\mathrm{mAh}$. Given the device's draw (as per DMM) are

$$I_{\mathrm{avg}} \approx I_{\mathrm{Arduino}} + I_{\mathrm{LCD}} + I_{\mathrm{TMP+LEDs}} \approx 50\,\mathrm{mA} + 20\,\mathrm{mA} + 5\,\mathrm{mA} = 75\,\mathrm{mA},$$

the expected runtime t is

$$t = \frac{C}{I_{\mathrm{avg}}} = \frac{500\,\mathrm{mAh}}{75\,\mathrm{mA}} \approx 6.7h$$

This is ok for a prototype, but would be unusable for real world applications, as users would need to replace the devices power supply up to 4 times a day. For longer operation, the Arduino can be powered externally via the barrel jack (7-12 V) or USB/USB-C 5V input.

I. **Arduino code**

See Appendix for code (Section C)

J. **Prototype specifications**

- Voltage of power supply: 9 V
- Operating voltage of circuit: 5 V
- Total current drawn (measured with DMM): $\approx 75\,\text{mA}$
- Battery operating time: $t = \dfrac{C}{I_{\text{avg}}} \approx 6.7h$ as previously shown
- Sensor temperature range (from datasheet): $-40\,°\text{C}$ to $125\,°\text{C}$
- Comfortable temperature range: $20\,°\text{C}$ to $25\,°\text{C}$
- KVL for resistor choices: (*see Section G.*)

4 Evaluation of Results

- The objectives of this project were met: the device can effectively measure correct temperature and initiate an interrupt signal in the event loop which triggers warnings when the temperature running average exceeds the setpoint of 26 °C. The LCD's effectively display the calculated temperature from the Analog input voltage of the TMP36 sensor, as well as some basic initialization status and warning signals. The device effectively mitigates the issue of over-regulation of temperature by utilizing a running average of the recorded temperature of the TMP36 sensor by polling it every second, but triggering the warning signals only when the running average exceeds the setpoint. This allows for a more stable and accurate reading of the temperature, and prevents the device from triggering false alarms due to short spikes in temperature.

- In a real-world application, this would allow the device to be used in a more efficient manner, as it would not trigger false alarms and would only alert the user when the temperature exceeds the setpoint for a sustained period of time. To increase its effectiveness, the interval of time between loops `READ_INTERVAL_MS` could be increased to 5-10 seconds, since quick spikes of temperature in indoor environments are uncommon. Leveraging AVR low-power modes on the Arduino UNO to set it to idle between readings would reduce power consumption even further, extending its battery life. Furthermore, the I^2C LCD backlight could be turend off when the temperature is in the comfortable range, and turned on when the temperature exceeds the setpoint, which would also greatly reduce power consumption.

- Future designs could employ a custom one-piece custom PCB that integrates the microcontroller, TMP36 footprint, power regulation, LCD header, LEDs, and buzzer. This approach would eliminate wiring, reduce signal noise and resistance, shrink the form factor, and simplify assembly and maintenance, enhancing both performance stability and manufacturing efficiency.

- The employed TMP36 sensor is not the most accurate temperature sensor available, but for its low cost it achieves an accuracy of $\pm 1°C$ over its full range. By contrast, commercial room thermostats such as the Honeywell FocusPRO 5000 advertise a display accuracy of $\pm 0.5°C$. Despite this being an improvement in accuracy by 50%, such minute differences are not noticeable to the human and do not heavily impact the performance of the device. Its main weaknesses are its form factor and operating time, but these are all results of the component choices made in this project and can be improved in future iterations.

Appendix

A Full Table of Precision Measurements

Table 2: Relevant dimensions of major components (see Appendix A for Figures 4 - 10)

Item	Fig. ID	W [mm]	L [mm]	H [mm]	⌀ [mm]
Full Assembly	1	123,8	146,1	63,1	-
Injection Mold ABS Enclosure Bottom	4	119,0	146,1	57,4	-
Injection Mold ABS Enclosure Lid	5	119,0	120,1	5,5	-
Transparent Acrylic Laser-Cut Base Plate	6	95,0	96,1	9,0	-
Arduino UNO Microcontroller	7	53,3	74,9	15,2	-
9 V Battery and PLA-Printed Holder	8	29,7	48,8	21,0	-
Alphanumeric 16×2 I^2C LCD	9	36,0	80,0	22,0	-
TMP36 Analog Temperature Sensor	10	-	-	14,6	2,5
Piezo Capsule Buzzer	10	-	-	21,3	20,0
LEDs (Red and Green)	10	-	-	36,5	3,0
2-Way Switch	10	14,8	20,8	23,3	-

B Detail CAD Drawings of Components

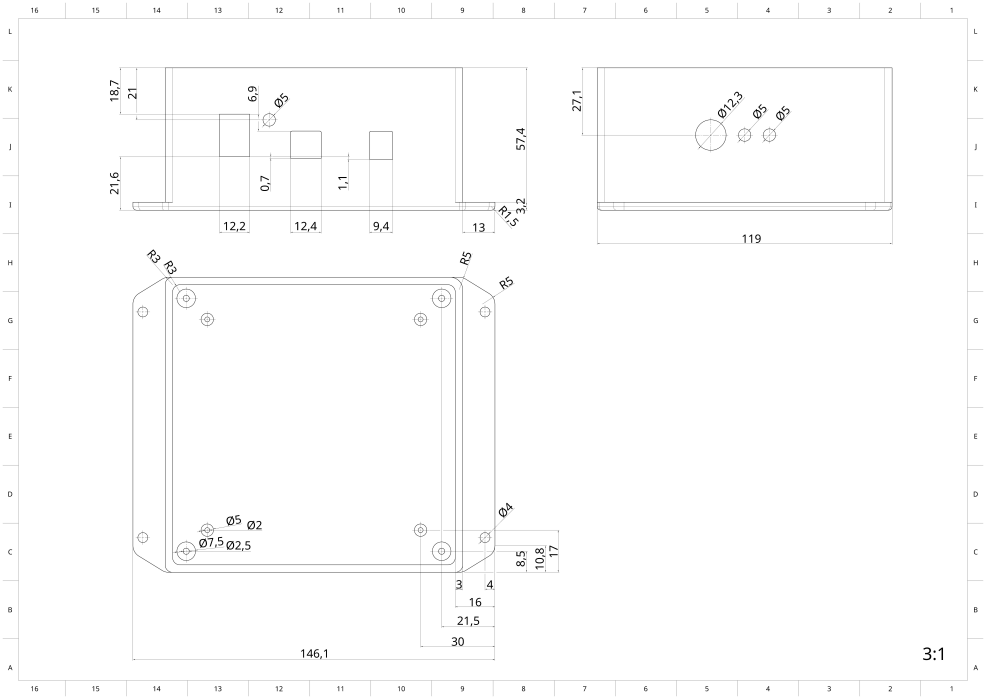


Figure 4: Injection Mold ABS Enclosure Bottom

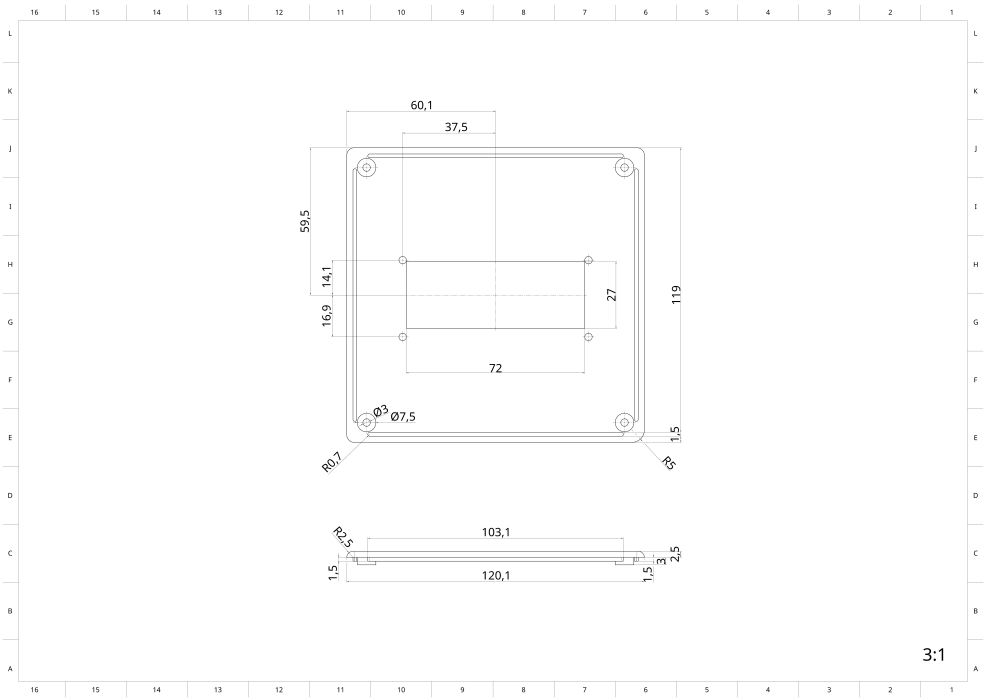


Figure 5: Injection Mold ABS Enclosure Lid

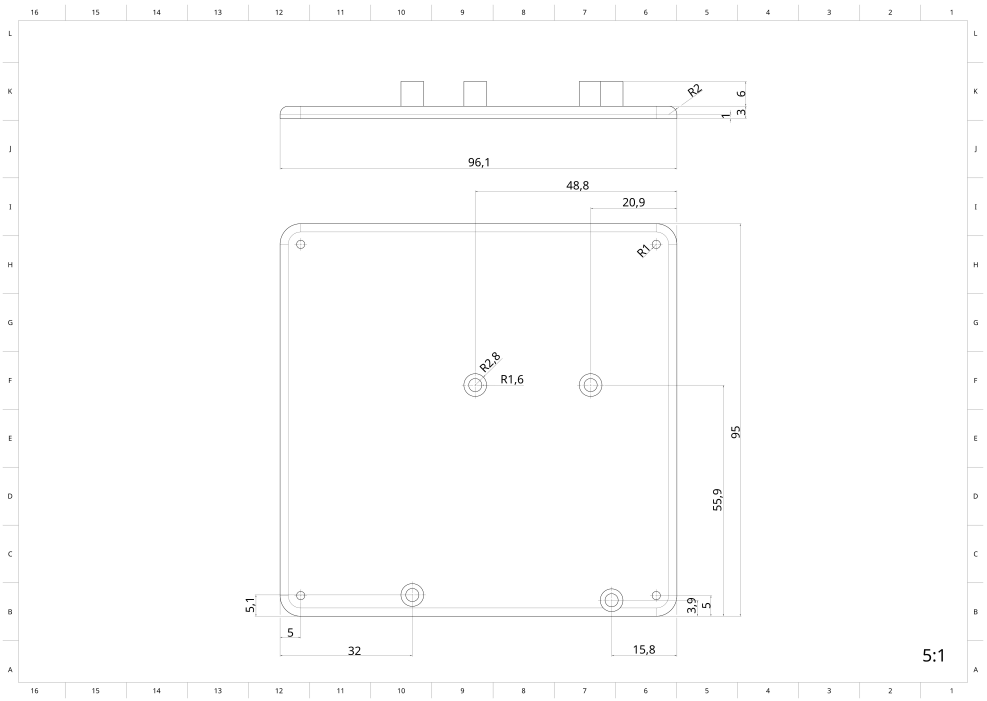


Figure 6: Transparent Acrylic Laser-Cut Base Plate

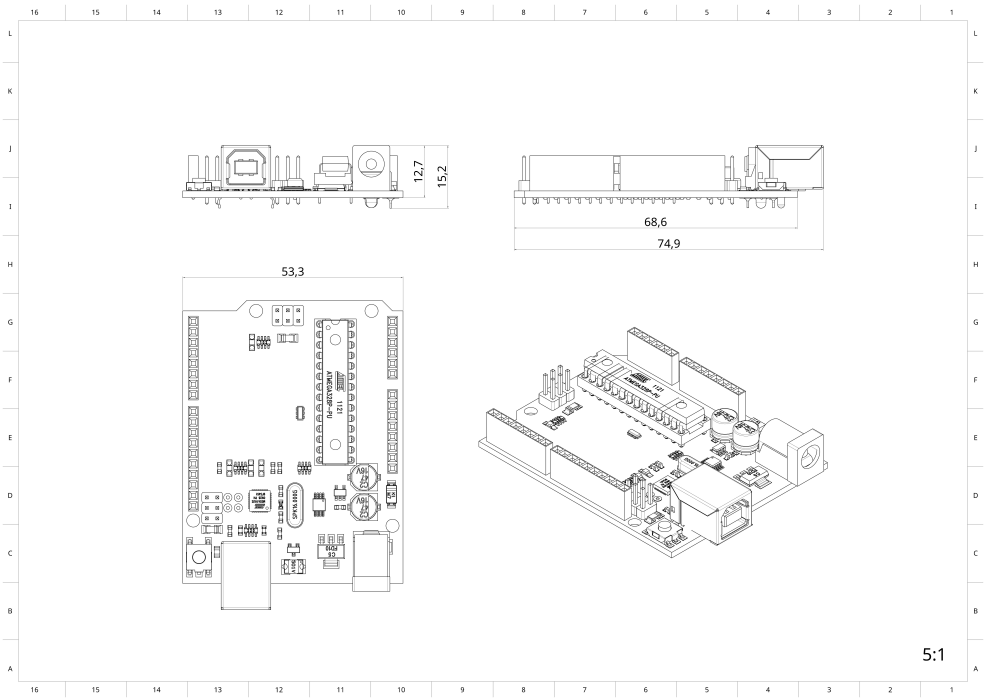


Figure 7: Arduino UNO Microcontroller

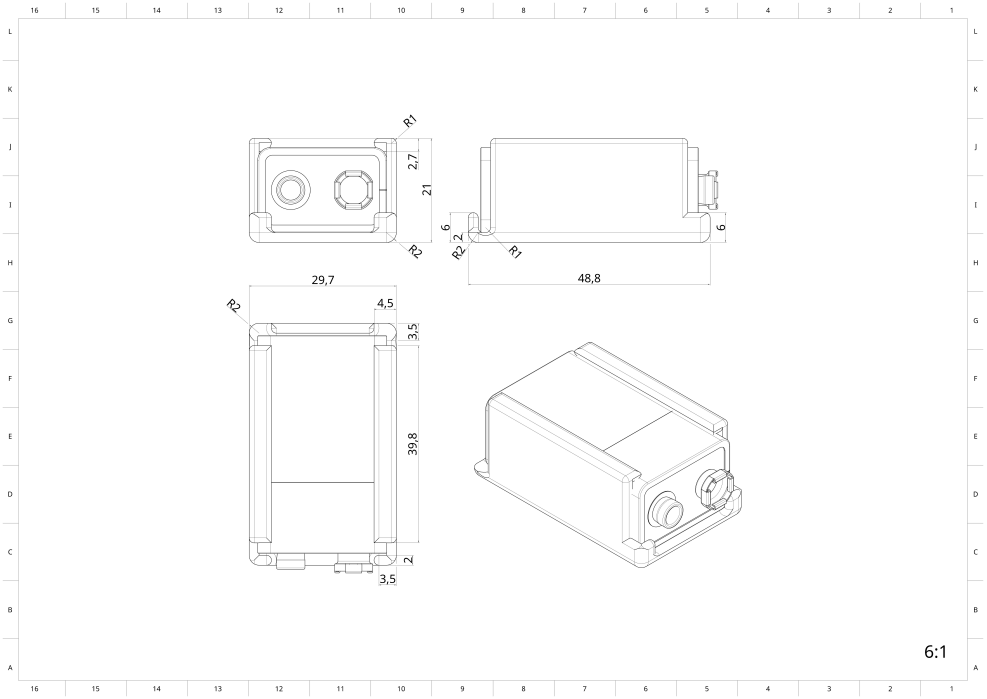


Figure 8: 9 V Battery and PLA-Printed Holder

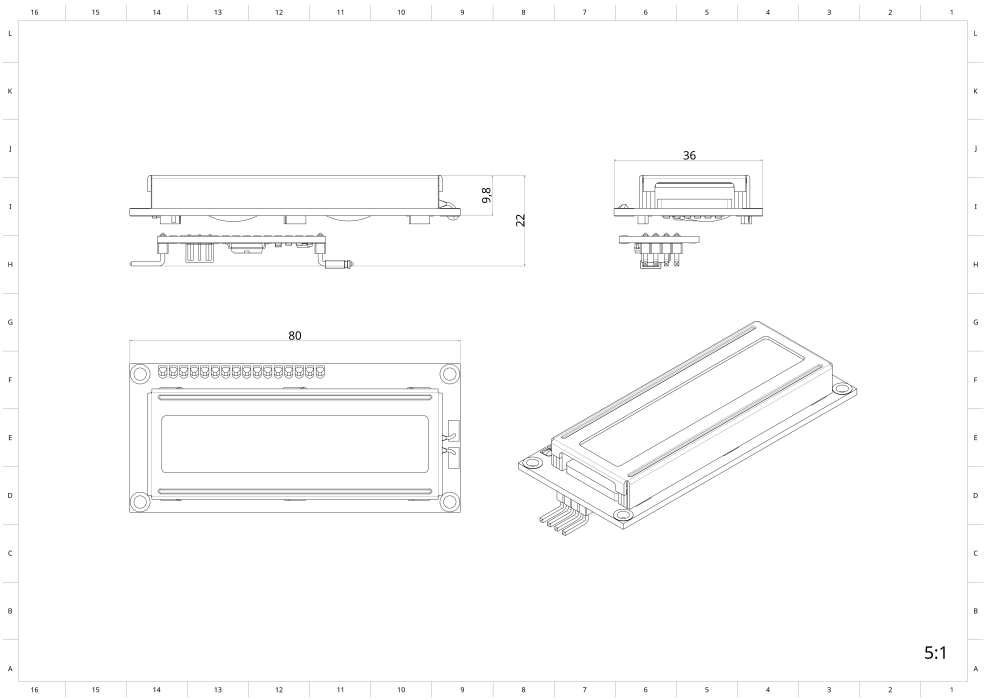


Figure 9: Alphanumeric 16×2 I^2C LCD

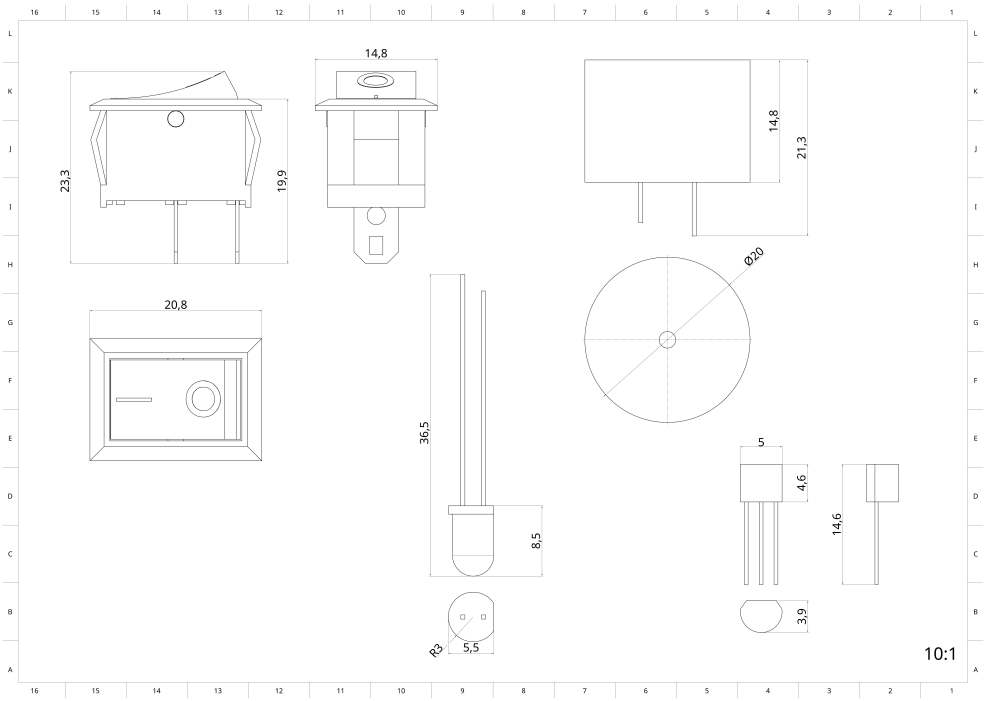


Figure 10: TMP36 Analog Temperature Sensor, Piezo Capsule Buzzer, 2 way switch and LEDs

C Arduino Code

Listing 1: Arduino sketch for running average temperature monitoring

```
1  #include <Wire.h>
2  #include <LiquidCrystal_I2C.h>
```

```
3
4       const int TEMP_PIN = A0;
5       const int RED_LED_PIN = 3;
6       const int BUZZER_PIN = 5;
7       const float TEMP_THRESHOLD_C = 26.0;
8       const int LCD_ADDR = 0x27;
9       const int LCD_COLS = 16;
10      const int LCD_ROWS = 2;
11      const unsigned long READ_INTERVAL_MS = 1000;
12      const int BUZZER_FREQ_HZ = 1000;
13      const int NUM_SAMPLES = 5; // sample size (NUM_SAMPLES * READ_INTERVAL_MS) is 5s
14
15      float TempC = 0.0;
16      float TempF = 0.0;
17      bool isAlarmActive = false;
18      unsigned long lastReadTime = 0;
19      float tempSumC = 0.0;
20      int sampleCount = 0;
21
22      LiquidCrystal_I2C lcd(LCD_ADDR, LCD_COLS, LCD_ROWS);
23
24      void setup() {
25          lcd.init();
26          lcd.backlight();
27          lcd.setCursor(3, 0);
28          lcd.print("Starting...");
29
30          pinMode(RED_LED_PIN, OUTPUT);
31          pinMode(BUZZER_PIN, OUTPUT);
32
33          digitalWrite(RED_LED_PIN, LOW);
34          noTone(BUZZER_PIN);
35
36          delay(1500);
37          lcd.clear();
38          printNormalLabels();
39      }
40
41      void loop() {
42          unsigned long currentTime = millis();
43          if (currentTime - lastReadTime >= READ_INTERVAL_MS) {
44              lastReadTime = currentTime;
45
46              int TADC = analogRead(TEMP_PIN);
47              float VTEMP = 5.0 * (TADC / 1024.0);
48              float newTempC = 100.0 * (VTEMP - 0.5);
49
50              // running average filter for smooth temperature displaying
51              tempSumC -= TempC;
```

```
            tempSumC += newTempC;
            TempC = tempSumC / NUM_SAMPLES;
            TempF = TempC * 9.0 / 5.0 + 32.0;

            bool shouldAlarmBeActive = (TempC > TEMP_THRESHOLD_C);

            if (shouldAlarmBeActive != isAlarmActive) {
                isAlarmActive = shouldAlarmBeActive;
                lcd.clear();

                if (isAlarmActive) {
                    digitalWrite(RED_LED_PIN, HIGH);
                    tone(BUZZER_PIN, BUZZER_FREQ_HZ);

                    lcd.setCursor(0, 0);  // Top left
                    lcd.print("!!");

                    lcd.setCursor(5, 0);
                    lcd.print(TempC, 1);
                    lcd.print((char)223);
                    lcd.print("C");

                    lcd.setCursor(14, 0); // Top right (leave space)
                    lcd.print("!!");

                    lcd.setCursor(0, 1); // Bottom Left
                    lcd.print("!!");

                    lcd.setCursor(5, 1);
                    lcd.print(TempF, 1);
                    lcd.print((char)223);
                    lcd.print("F");

                    lcd.setCursor(14, 1); // Bottom Right (leave space)
                    lcd.print("!!");
                } else {
                    digitalWrite(RED_LED_PIN, LOW);
                    noTone(BUZZER_PIN);
                    printNormalLabels();
                    updateNormalDisplay();
                }
            } else {
                if (!isAlarmActive) {
                    updateNormalDisplay();
                }
            }
        }
    }

```

```
101     void printNormalLabels() {
102         lcd.setCursor(9, 0);
103         lcd.print((char)223);
104         lcd.print("C");
105         lcd.setCursor(9, 1);
106         lcd.print((char)223);
107         lcd.print("F");
108     }
109 
110     void updateNormalDisplay() {
111         lcd.setCursor(5, 0);
112         lcd.print(TempC, 1);
113         if (TempC < 10.0 && TempC >= 0.0) lcd.print(" ");
114         if (TempC < 0.0 && TempC > -10.0) lcd.print(" ");
115 
116         lcd.setCursor(5, 1);
117         lcd.print(TempF, 1);
118         if (TempF < 100.0 && TempF >= 10.0) lcd.print(" ");
119         else if (TempF < 10.0 && TempF >= 0.0) lcd.print("  ");
120         else if (TempF < 0.0 && TempF > -10.0) lcd.print(" ");
121     }
```