Design Process of Room Temperature Monitor

Giacomo Cappelletto
BU ID: U91023753

1 Summary

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 $\mathrm{I^2C}$ 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 $\mathrm{I^2C}$ 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^{2}C 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 × 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 2-8)

Item	Fig. ID	W [mm]	L [mm]	H [mm]	Ø [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^{2}C LCD	9	36,0	80,0	22,0	-
TMP36 Analog Temperature Sensor	10	-	-	14,6	2,5
Piezo Capsule Buzzer	**??**	-	-	21,3	20,0
LEDs (Red and Green)	**??**	-	-	36,5	3,0
2-Way Switch	**??**	14,8	20,8	23,3	-

C. **CAD drawings**

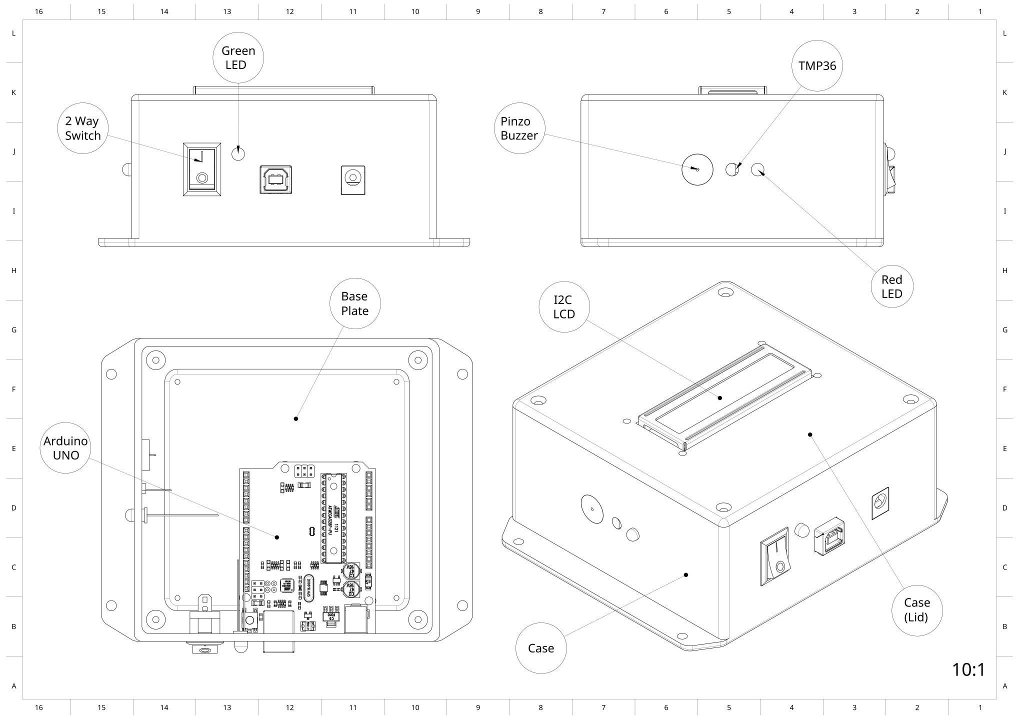

Figure 1: CAD assembly of the prototype.

Figures 2-8 in Appendix (see Section A)

D. **Prototype photographs**

Top view of actual prototype, with and without lid, powered on.

(a) With lid (b) Without lid

Figure 2: Top views of the working prototype.

E. **Purpose of using an Arduino board**

Explain role of the Arduino in the circuit.

F. **Wiring diagram and methods**

Show diagram and discuss soldering, jumper wires, twist nut caps, spade connectors, etc.

Figure 3: Circuit wiring diagram.

G. **Wire gauge and resistor values**

Specify wire gauge used and justify choice (jumper vs. 22 AWG). State resistor values in series with green/red LEDs, calculate operating currents using KVL.

H. **Internal power supply**

Discuss why a 9 V battery, its charge capacity, expected runtime, and external power options.

I. **Arduino code**

Provide the code listing used in your design.

Listing 1: Arduino sketch for running average temperature monitoring

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

const int TEMP_PIN = A0;
const int RED_LED_PIN = 3;
const int BUZZER_PIN = 5;
const float TEMP_THRESHOLD_C = 26.0;
const int LCD_ADDR = 0x27;
const int LCD_COLS = 16;
const int LCD_ROWS = 2;
const unsigned long READ_INTERVAL_MS = 1000;
const int BUZZER_FREQ_HZ = 1000;
const int NUM_SAMPLES = 5; // sample size (NUM_SAMPLES * READ_INTERVAL_MS) is 5s

float TempC = 0.0;
float TempF = 0.0;
bool isAlarmActive = false;
unsigned long lastReadTime = 0;
float tempSumC = 0.0;
int sampleCount = 0;

LiquidCrystal_I2C lcd(LCD_ADDR, LCD_COLS, LCD_ROWS);

void setup() {
        lcd.init();
        lcd.backlight();
        lcd.setCursor(3, 0);
        lcd.print("Starting...");

        pinMode(RED_LED_PIN, OUTPUT);
        pinMode(BUZZER_PIN, OUTPUT);

        digitalWrite(RED_LED_PIN, LOW);
        noTone(BUZZER_PIN);

        delay(1500);
        lcd.clear();
        printNormalLabels();
}
```

```
void loop() {
        unsigned long currentTime = millis();
        if (currentTime - lastReadTime >= READ_INTERVAL_MS) {
                lastReadTime = currentTime;

                int TADC = analogRead(TEMP_PIN);
                float VTEMP = 5.0 * (TADC / 1024.0);
                float newTempC = 100.0 * (VTEMP - 0.5);

                // running average filter for smooth temperature displaying
                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();
            }
        }
    }
}

void printNormalLabels() {
    lcd.setCursor(9, 0);
    lcd.print((char)223);
    lcd.print("C");
    lcd.setCursor(9, 1);
    lcd.print((char)223);
    lcd.print("F");
}

void updateNormalDisplay() {
    lcd.setCursor(5, 0);
    lcd.print(TempC, 1);
    if (TempC < 10.0 && TempC >= 0.0) lcd.print(" ");
    if (TempC < 0.0 && TempC > -10.0) lcd.print(" ");

    lcd.setCursor(5, 1);
    lcd.print(TempF, 1);
    if (TempF < 100.0 && TempF >= 10.0) lcd.print(" ");
    else if (TempF < 10.0 && TempF >= 0.0) lcd.print("  ");
```

```
        else if (TempF < 0.0 && TempF > -10.0) lcd.print("-");
}
```

J. **Prototype specifications**

List:

- Voltage of power supply
- Operating voltage of circuit
- Total current drawn (measured with DMM)
- Battery operating time
- Sensor temperature range (from datasheet)
- Comfortable temperature range
- Use KVL to explain resistor choices (1 k Ω for green LED, 220 Ω for red LED)

4 Evaluation of Results

- Degree to which design objectives were met
- Summary of outcomes and comparison to standard thermometers
- Limitations and recommendations for future work (battery life, size, weight, etc.)
- Highlight usefulness of the design

A Detail CAD Drawings of Components

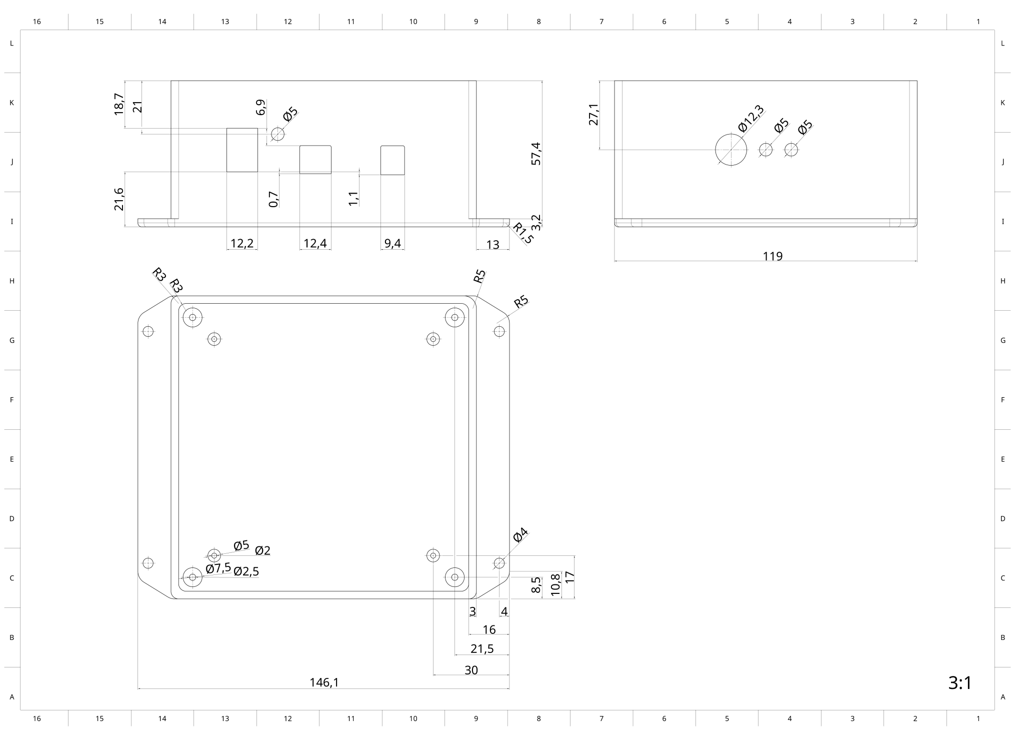

Figure 4: Full assembly of the prototype.

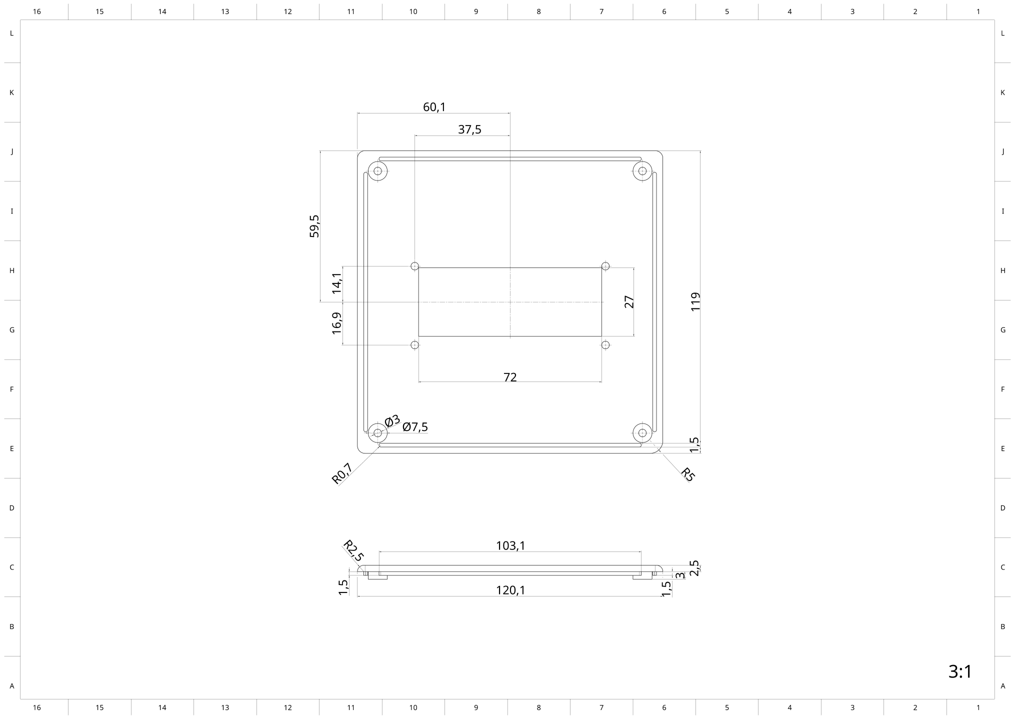

Figure 5: Full assembly of the prototype.

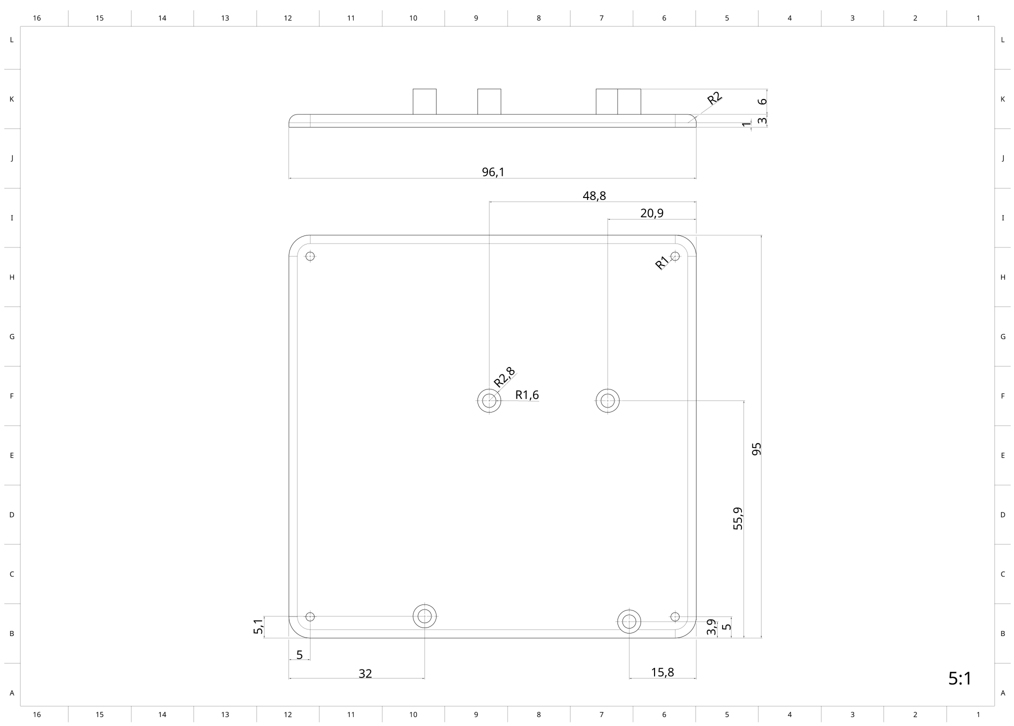

Figure 6: Full assembly of the prototype.

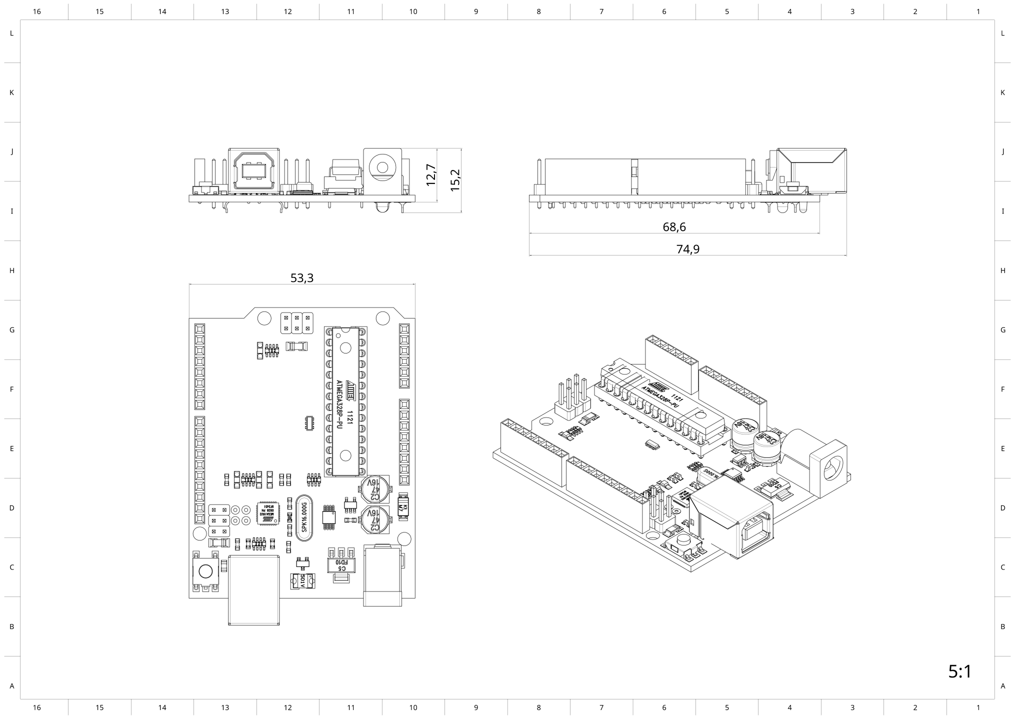

Figure 7: Full assembly of the prototype.

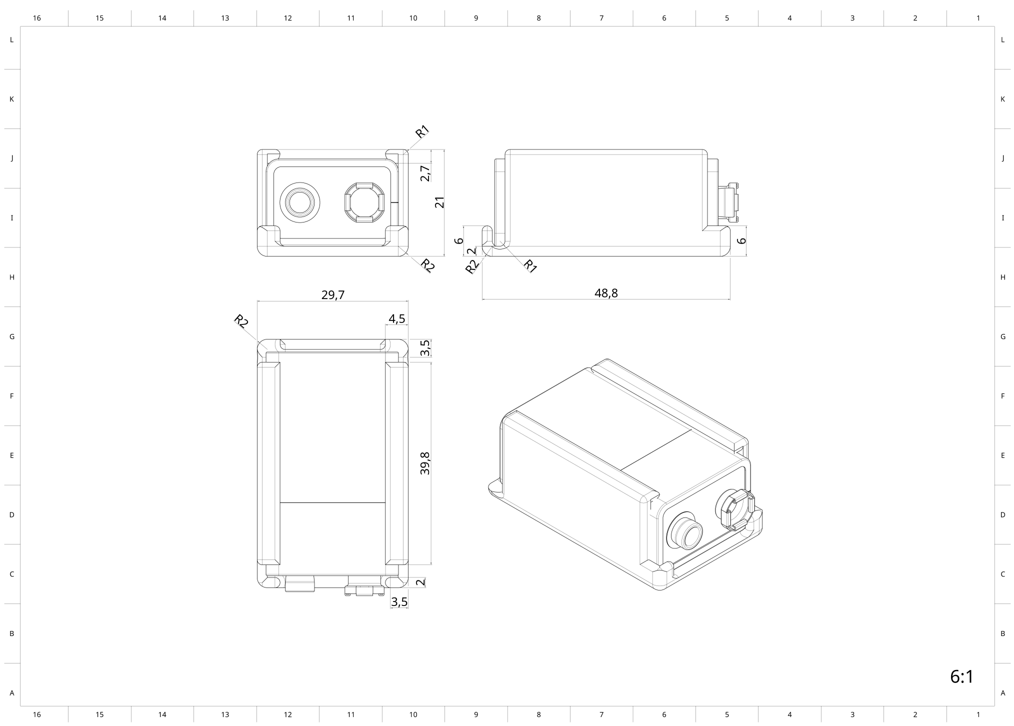

Figure 8: Full assembly of the prototype.

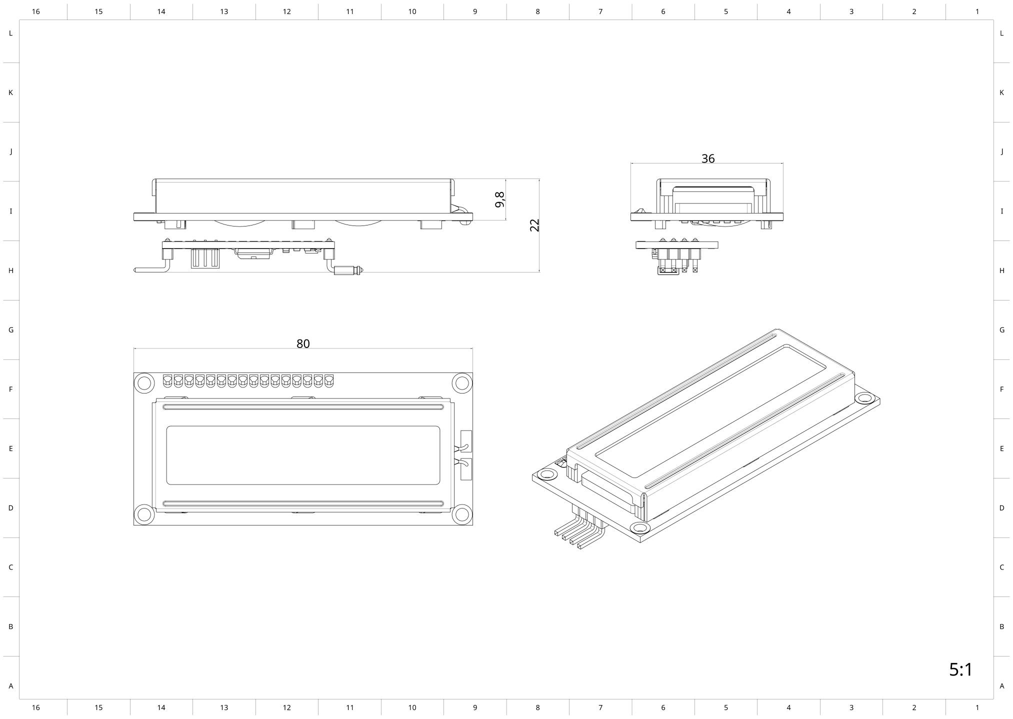

Figure 9: Full assembly of the prototype.

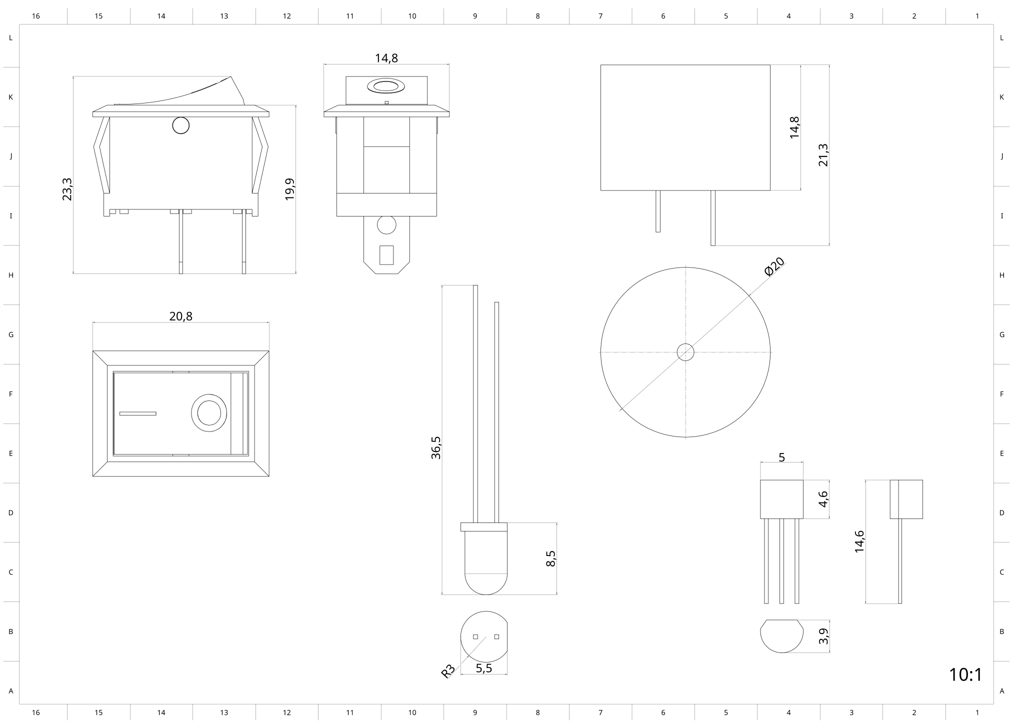

Figure 10: Full assembly of the prototype.