Parth Kheni
U89149991
EK131
Professor Ousama Aamar
Section A2

Design Process of Room Temperature Monitor

Summary

This report summarizes the design, construction, and demonstration of a temperature monitoring device. The project involved creating a system that measures temperature using a TMP36 sensor and displays the readings in Celsius and Fahrenheit on an LCD screen. The device triggers an alarm consisting of a piezo buzzer and a flashing red LED if the temperature deviates from a predefined range (72°F–75°F). The circuit is powered by a 9V battery, controlled by an Arduino, and features a 2-way rocker switch for activation, accompanied by a green LED to indicate proper functionality. The components are housed in a custom ABS enclosure with a 3D-printed battery holder. Wiring connections were completed using soldering and insulated with heat shrink tubing, ensuring durability and safety. The project successfully demonstrates the integration of electrical and mechanical engineering principles with programming, resulting in a functional, user-friendly device. The outcome highlights the importance of multidisciplinary approaches in engineering design and serves as a practical application of course concepts.

Introduction

This report details the design, construction, and functionality of a room temperature monitoring device that integrates mechanical and electrical engineering with programming. The device monitors room temperatures, alerting users when they deviate from a comfortable range of 72°F–75°F. It displays ambient temperature in Fahrenheit and Celsius on an LCD screen and activates a buzzer and flashing red LED for alerts. Powered by a 9V battery and enclosed in an ABS housing, the project features precise wiring, circuit design, Arduino programming, and custom CAD-based packaging. This report provides an overview of the design process, construction techniques, and outcomes, highlighting the value of interdisciplinary engineering in creating practical solutions.

Design Elements

a) List of Components

i) **3D Printed Enclosure and Lid:**

Encases and protects all the internal components of the temperature monitor.

ii) **3D Printed Custom Battery Holder (CAD in Appendix):**
Secures the battery to prevent shifting inside the enclosure.

iii) **Arduino Uno Microcontroller:**
Executes the programmed code and manages the functions of the device by processing inputs (e.g., LEDs, TMP36, Buzzer) and generating outputs (e.g., displaying temperature on the LCD).

iv) **Alphanumeric I2C LCD (16x2 Characters):**
Displays the temperature readings in both Fahrenheit and Celsius.

v) **2-Way Rocker Switch:**
Controls the power for the entire circuit, allowing it to be turned on or off.

vi) **Green LED:**
Illuminates when the switch is turned on, indicating that the circuit is active. This feature is particularly useful during circuit testing.

vii) **Red LED:**
Lights up when the temperature is outside the predefined range, indicating a "HIGH" state. When the temperature is within the range, the LED remains off ("LOW").

viii) **Piezo Capsule (Buzzer):**
Emits a 1KHz sound signal when the temperature falls outside the predetermined range.

ix) **1000Ω (Ohm) Resistor:**
Regulates the electrical current flowing through the green LED.

x) **200Ω (Ohm) Resistor:**
Manages the electrical current flowing through the red LED.

xi) **TMP36 (Temperature Sensor):**
Reads the ambient temperature and outputs a voltage proportional to the temperature in Celsius, with a range from -40°C to +125°C.

xii) **9V Battery:**
Provides power to the entire temperature monitoring device.

xiii) **9V Battery Snap:**

Connects the battery terminals to the Arduino while also insulating and protecting the battery.

xiv) **Stranded Jumper Wires:**

Connect all components to the Arduino via soldering and header pins. These wires are 26 AWG, with red for 5V lines, white for data lines, and black for ground connections.

xv) **22 AWG Solid Core Wires:**

Used to create common ground and 5V connections within the circuit. These wires are routed through wire nuts to the Arduino for a secure connection.

xvi) **Wire Nuts:**

Twist onto the bare wire ends to ensure tight and secure connections. They group the 5V wires together and the ground wires together.

xvii) **Heat Shrink:**

Insulates soldered connections to enhance safety and durability.

xviii) **Spade Connectors:**

Enable quick and reliable electrical connections for the 2-way rocker switch without the need for soldering.

xix) **Breadboard:**

Serves as a platform to test and prototype all components before final assembly.

b) In Appendix

c) In Appendix

d) In Appendix

e) The Arduino Uno board serves as the central controller, converting analog signals into digital signals. It stores and executes the code that governs the device's functions. The microcontroller processes inputs (e.g., LEDs, TMP36, Buzzer) and generates corresponding outputs (e.g., displaying temperature on the LCD).

f) Picture in Appendix

- **Soldering:** Utilized to join two metal components, creating a secure electrical connection. Soldering was applied to the LEDs, Buzzer, and TMP36.

- **Jumper Wires:** Employed for all data connections, with white wires designated for transmitting data to the red LED, Buzzer, and LCD, as well as receiving data from the TMP36. One end of the jumper wires was cut, stripped, and soldered to the red LED, Buzzer, and TMP36 to facilitate easy connections to the Arduino.
- **Solid Core Wires:** Recommended for all 5V and Ground connections due to their durability and ability to support higher currents.
- **Twist Nuts:** Used to securely connect all Ground wires and all 5V wires.
- **Spade Connectors:** Enabled convenient and efficient connections between 22 AWG Solid Core wires and the 2-way rocker switch.

g) The project utilized 22 AWG wires for power and ground connections due to their ability to handle up to 920 mA, far exceeding the circuit's <100 mA requirement, ensuring reliability and durability. Jumper wires were initially used for data connections between the LCD and Arduino during testing for flexibility. In the final assembly, 22 AWG wires were soldered for permanent connections, except for the LCD data lines, where jumper wires were retained for modularity.

Resistor values for the LEDs were selected for safe operation and optimal brightness. For the green LED, a 1000Ω resistor limited the current to ~3 mA, sufficient for its power-indicator role. For the red LED, a 200Ω resistor allowed 14 mA for enhanced visibility as a visual alarm. Using Kirchhoff's Voltage Law (KVL), the calculations were as follows:

Green LED: Voltage drop = 5.0V − 2.0V = 3.0V; Required resistance = 3.0V ÷ 0.003A = 1000Ω.

Red LED: Voltage drop = 5.0V − 2.2V = 2.8V; Required resistance = 2.8V ÷ 0.014A ≈ 200Ω.

These resistor values ensured the LEDs operated safely, with the green LED indicating power and the red LED providing sufficient brightness for alerts.

h) The device is powered by a 9V battery, selected for its compatibility with the Arduino Uno's input voltage range of 6–20V. This battery provides a portable, compact, and

widely accessible power source, making it ideal for a device designed for easy deployment. A standard 9V alkaline battery typically has a charge capacity of approximately 500 mAh, sufficient for short-duration usage. With the device consuming an average current of 80 mA, the runtime is estimated using the formula: Runtime=Capacity/Current=500 mAh/80 mA=6.25 hours. Although the 9V battery is convenient and adequately powers the device for portable applications, its limited capacity makes it more suitable for intermittent or short-term use rather than prolonged operation. For extended or continuous use, the device can be powered externally via the Arduino's USB port, which connects to a computer or USB power adapter. This dual power option provides flexibility, allowing the device to function effectively in both portable and stationary settings.

i) In Appendix

j) The prototype operates on a 9V battery regulated to 5V by the Arduino. The device draws 85 mA, measured using a DMM across the switch terminals with the circuit off. With a 9V battery capacity of 500 mAh, the runtime is approximately 5.88 hours, calculated as Runtime = Capacity ÷ Current = 500 mAh ÷ 85 mA. The TMP36 temperature sensor operates from -40°C to +125°C, and the prototype triggers alarms when the temperature falls outside 72°F–75°F, adjustable in the Arduino code. LED resistors were calculated using Kirchhoff's Voltage Law (KVL):

Green LED: Forward voltage = 2.0V, resistor voltage = 5V − 2.0V = 3.0V. Required resistance = 3.0V ÷ 0.003A = 1000Ω; a 1 kΩ resistor ensures safe operation as a power indicator.

Red LED: Forward voltage = 2.2V, resistor voltage = 5V − 2.2V

Evaluation of Results

The temperature monitor prototype successfully met its objectives by integrating key functionalities. It can be switched on and off, measures and displays temperature in Fahrenheit and Celsius, and triggers alarms—a flashing red LED and piezo buzzer—when the temperature falls outside 72°F–75°F. Built with an Arduino, TMP36 sensor, LEDs, a 9V battery, and a custom ABS enclosure, it demonstrates effective circuitry, assembly, and programming.

While functional, the prototype has limitations. The TMP36 sensor's slower response and limited accuracy could be improved with a higher-quality sensor. The 6.25-hour battery life could be extended with a larger-capacity battery or low-power design. A smaller, lighter enclosure would improve portability. The device's customizable Arduino code allows users to modify the temperature range or adapt it for other applications, offering versatility beyond standard thermometers. This project showcases the potential of combining hardware and software and prepares me for advanced engineering courses.

Supporting Materials(appendix)

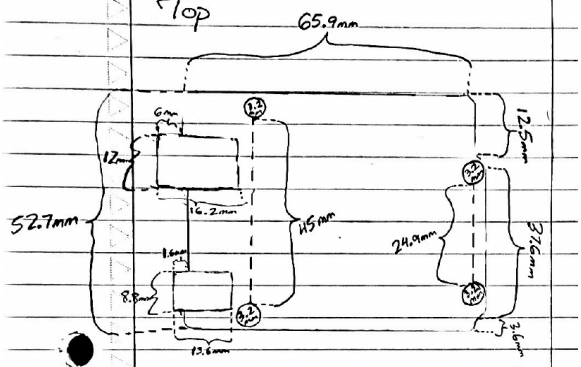
a. **From 2.b)**

Table 1: Precision Measurements					
Item	**Sketch Identification**	**W (mm)**	**L (mm)**	**H (mm)**	**Diameter (mm)**
ABS Enclosure	Top, Side, Front views	123.0	145.1	56.3	N/A
Arduino Board	Top, Front views	52.7	65.9	12.5	N/A
2-way Switch	Top View	20.6	14.7	20.1	N/A
LCD 2x16	Top, Front View	34.4	79	7.3	N/A
Buzzer	Top View	N/A	N/A	10.6	21.5
Temperature Sensor	Top View	N/A	N/A	4.3 Without stem	3
LEDs	Top View	N/A	N/A	8.8 Without stem	5.4

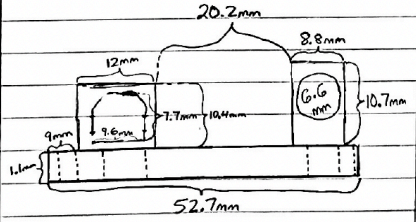
b. **From 2.c)**

Component 1: Arduino Uno Board

Top

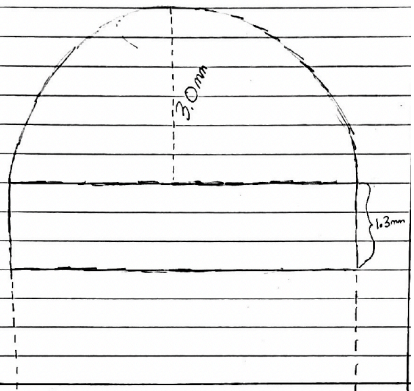


Front

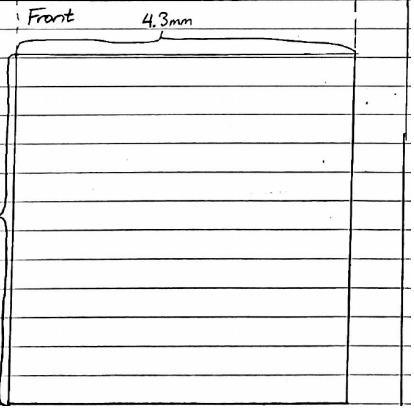


Component 2: Temperature Sensor

Top

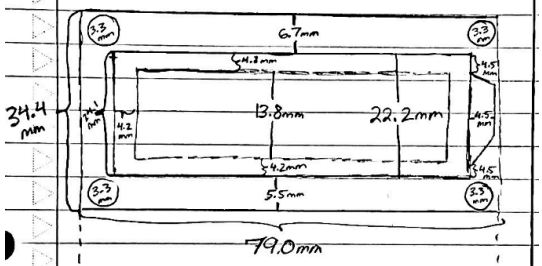


Front

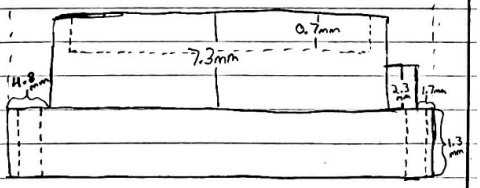


Component 3: Alphanumeric I2C LCD

Top

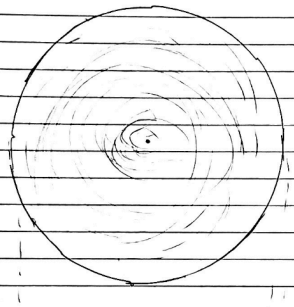


Front

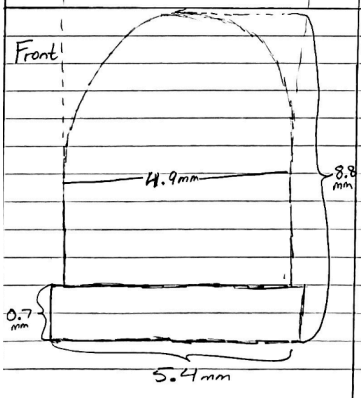


Component 4: LEDs

Top

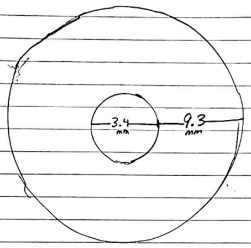


Front

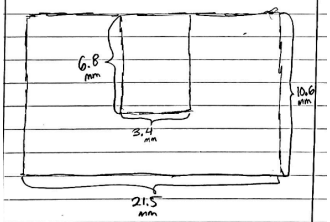


Component 5: Piezo Capsule/Buzzer

Top

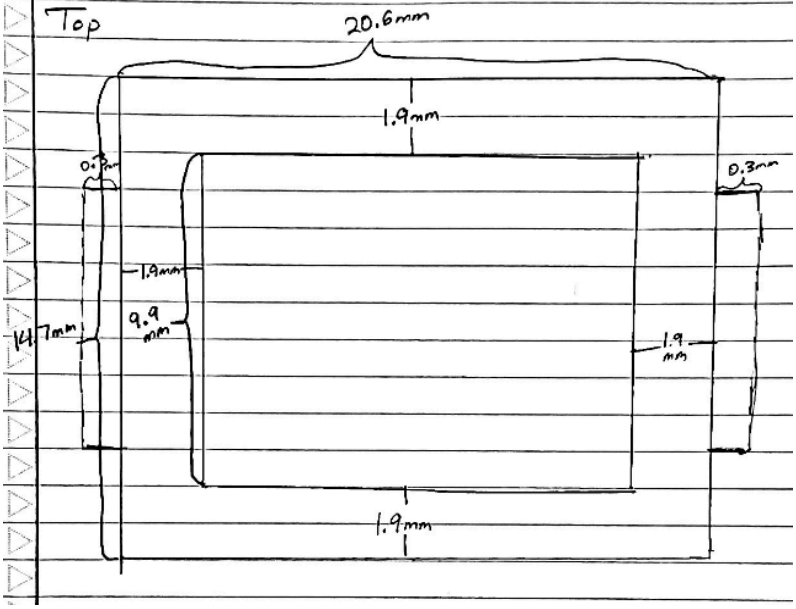


Front

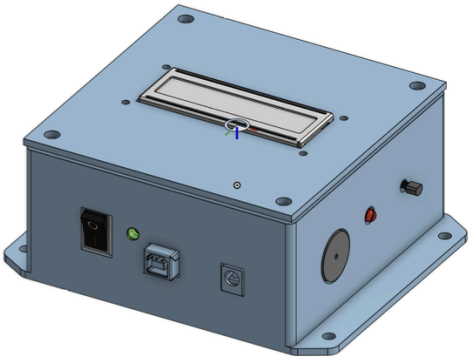


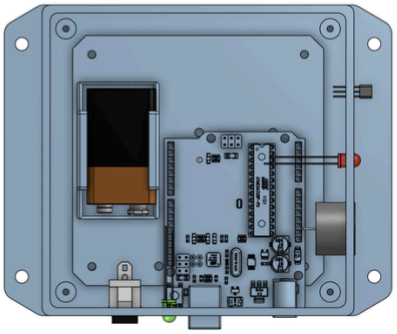
Component 6: 2-Way Switch

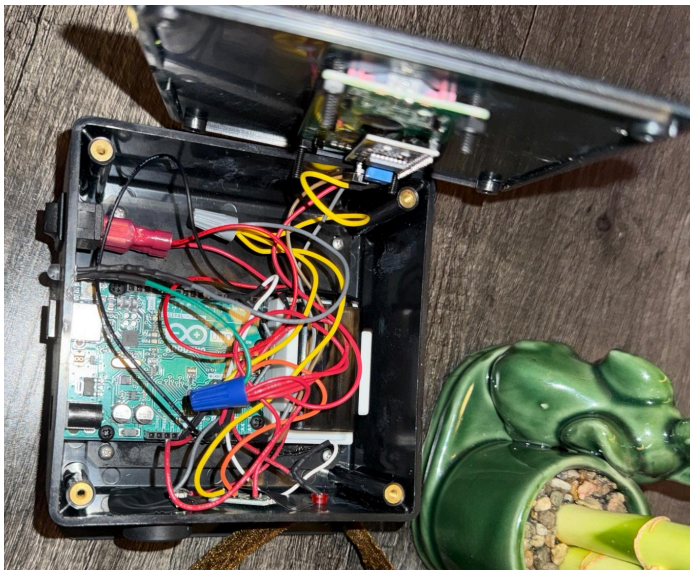
Top



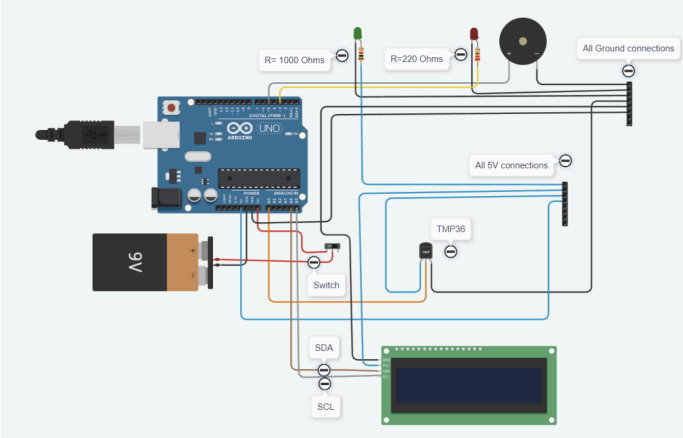
c. **From 2.d)**







d. **From 2.f)**



e. From 2.i)

```
#include <LiquidCrystal_I2C.h>

int TMP_PIN = A0;
int RED_LED_PIN = 3;
int BUZZER_PIN = 5;
LiquidCrystal_I2C lcd(0x27,16,2);
void setup() {
  // put your setup code here, to run once:
  pinMode(BUZZER_PIN, OUTPUT);
  pinMode(RED_LED_PIN, OUTPUT);

  // LCD Setup
  lcd.init();
  lcd.backlight();
}

void loop() {
  // Temperature Detection
  int v_adc = analogRead(TMP_PIN);
  float voltage = 5*v_adc/1023.0;
  float tempC = 100*(voltage-0.5);

  // Temperature in F
  float tempF = 32 + (9*tempC)/5;

  // Printing Temperature into LCD
  lcd.setCursor(0,0);
  lcd.print("Temp is ");
  lcd.print(tempC);
  // lcd.print(tempF);
  lcd.setCursor(0, 1);
  lcd.print("deg Celcius");
  // lcd.print("deg Fahrenheit");

  // if (tempF < 70 || tempF > 75) {
  if (tempC > 20 || tempC < 25) {
  // Starting Buzzer & RED LED
  digitalWrite(RED_LED_PIN, HIGH);
  tone(BUZZER_PIN, 1000);
  delay(1000);

  // Stopping Buzzer & RED LED
  digitalWrite(RED_LED_PIN, LOW);
  noTone(BUZZER_PIN);
}
else {
  digitalWrite(RED_LED_PIN, LOW);
  digitalWrite(BUZZER_PIN, LOW);
}

  delay(100);
}
```