FACULTY OF ENGINEERING AND APPLIED SCIENCE

MECE 2030U Electronics Applications for Mechatronics

LABORATORY REPORT

Instructor: [redacted]

Lab TA: [redacted]

Experiment: Lab Project: Temperature Monitoring System, Lab Group: 8

	GROUP MEMBERS			
#	Surname Name	Email	Student ID	Signature
1	Rehman Muzzammil	muzzammil.rehman@ontariotechu.net	100864100	M.R

Table of Contents

I. Introduction & Objectives

The laboratory work will focus on the fundamental concepts of sensor application and signal processing with an electronic system focusing on the design of a temperature monitoring system. The main objective of this lab session related to sensors, signals, and temperature monitoring is to be familiarized with sensor technology and its basic principles, as well as with its practical applications. Via hands-on experiments, the goal is to obtain a strong knowledge of thermistors and the associated resistance-temperature relationship. The aim is to know how to process the acquired temperature data from the temperature sensor (thermistor) with Arduino Mega. As well as improve capabilities in showing the actual temperature readings with LCD display and discover solutions for more stable circuits and less distortions. Through the use of Arduino, the intent is to enrich the knowledge of microcontroller systems and their practical implementation in monitoring and control.

II. Equipment & Materials

This lab project is the design of an Arduino-based Temperature Monitoring System that uses electronic components and programming. It encompasses three main parts: emulate the behavior of a thermistor, put the thermistor and Arduino together for temperature monitoring, and improve the system with a sensor preamplifier and LED alarm indicators:

- Op Amp 741
- LCD display
- Arduino Mega
- Thermistor
- Connecting wires and breadboard
- Multimeter
- Oscilloscope (for optional advanced analysis)
- LEDs
- Potentiometer

III. Procedure

Part 1: Monitoring the Temperature

- Make Arduino Mega, LCD display, and thermistor be part of the workspace, together with wires, and breadboard
- The resistance of a room temperature thermistor can be determined using a multimeter
- Record the value
- Determine an NTC or PTC thermistor to which type it belongs based on its temperature characteristics
- Connect the thermistor to the Arduino board using the input circuit diagram that can be found out in Figure 3 of the lab manual
- Upload Thermistor code on the Arduino Mega board as the provided code
- See the LCD monitor for the temperature reading on the display screen
- The interactions are made by touching the thermistor directly to the LCD to observe the corresponding changes with temperature readings
- note temperature as a key data point and confirm that the system depicts the actual room temperature correctly

Part 2: Sensor Preamplifier

- Remove the thermistor from the Arduino Mega and breadboard, and prepare to install the components
- To fulfill this purpose, the op-amp 741 circuit has to be connected to the breadboard according to Figure 1 of the Lab Manual
- Make the power available for the op-amp circuit
- Measure the output-input relationship of the preamp with a multimeter
- Replace the thermistor circuit in Part 1 with the preamplified setup
- Introduce the noise input in an approximate distance of 30 cm from the thermistor.
- On the oscilloscope or multimeter, check the voltage's stability of the output
- Take note of the impact the disturbance is likely to produce on the signals

Part 3: Adding LED Alarm Indicators

- Install two LEDs into the circuit and put them on the Arduino outputs A and B

- Design the Arduino code for control of the LED temperature indicators using the threshold values
- Encode the blue LED to trigger when the temperature is below 24°C
- If the temperature is at 28°C or higher, then will configure the red LED to flash
- Install the Arduino code into the Arduino Mega
- Supply the power to the system and note the patterns of the LED indicators while it is modulating the temperatures
- Test by LED to see if the changes in temperature are registered by the devices

IV. Results & Discussion

A. Monitoring Temperature

To identify the type of thermistor, the resistance at room temperature (nominal value) and at a higher or lower temperature needs to be measured. The nominal value at approximately 25° Celsius was 10,000 Ω, and at approximately 35° Celsius it was a much lower 5,680 Ω. From these values, it can be concluded that the thermistor is of type NTC (negative thermal coefficient), since an increase in temperature results in a decrease in resistance.

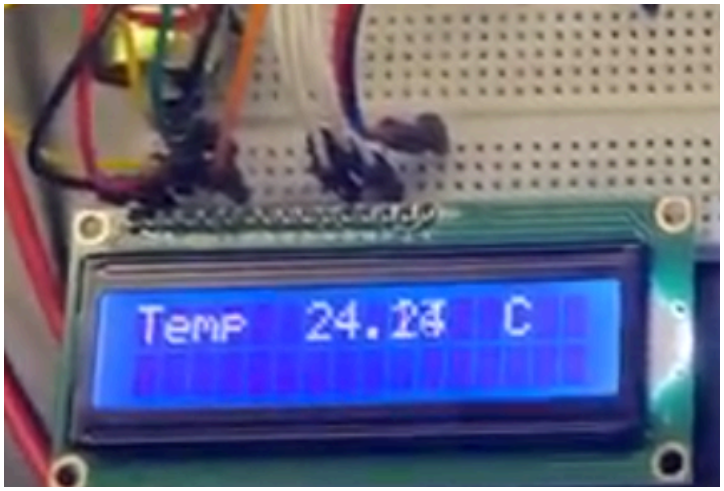

Figure 1: Temperature Reading After Initial Set-Up

The displayed value before any additional touching (to add heat) was extremely close to room temperature. Then after touching the thermistor, the response is almost immediately noticeable, with the displayed temperature adjusting to the added heat. This is to be expected, as the voltage across the thermistor is being read directly by the Arduino.

The Arduino code is simple in this first stage of the experiment. However an understanding of how the Arduino reads analog values is important to understanding how the temperature is actually collected. In the global scope, the library to effectively use the LCD (liquid crystal display) is imported for use, as well as an object for the LCD is created from the library with the pins passed in its parameters. Register-Select (pin 7) decides whether or not character or command data is being sent, Enable (pin 8) allows for data transfer to the LCD if pulled high. The rest of the pins (9-12) are connected to send 4-bit data to the display. Other than those main points, the rest of the code regarding the LCD is just printing any values to it in the loop() function or initialization in the setup() function.

What is quite interesting is the code regarding reading the voltage value of the thermistor and accurately deriving a temperature from that value. Essentially, the analogRead() function converts the analog signal to a 10-bit resolution digital signal, with values ranging from 0 - 1023. However to find the temperature of the thermistor, the resistance is required and to calculate the resistance the analog voltage is needed, meaning that the ADC capability of the Arduino is actually a hindrance. To accomplish temperature measurement, an understanding of how the Arduino collects the analog signal is required. It can be theoretically assumed that the Arduino gets the voltage from the voltage divider the thermistor has between itself and the 10 kOhm pull-down resistor. Dividing that value by the analog reference value (specific to the operating voltage of the device, in this case for the thermistor it is 5V), provides a ratio of the instantaneous voltage to the maximum possible voltage for the device. This ratio can then be 'mapped' to the 0-1023 10-bit digital ratio by multiplying it by 1024, as such the voltage divide equation (including the mapping to digital) can be rearranged to find $R_{thermistor}$, allowing for the calculation of the temperature.

B. Sensor Preamplifier

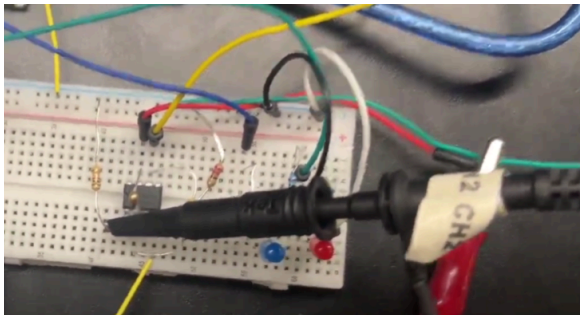

Figure 2: Operational Amplifier Circuit

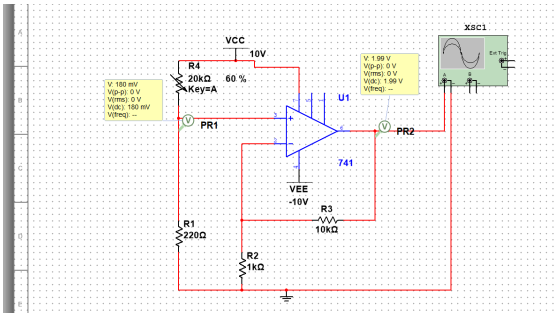

Figure 3: Series-Shunt Circuit Schematic

After creating the difference amplifier op-amp circuit that was simulated in the pre-lab, Figure 3, measurements of the input voltage and output voltage were taken to see if the gain of the circuit aligned with the simulated values. Using the multimeter, it was found that the input voltage to the op-amp was 230mV and the output voltage was 2.64V, which means that the gain is approximately 12. This is in line with the simulated gain of 11, and the minuscule difference has most likely been caused by the tolerances of the components (resistors, op-amp, thermistor), as well as the small added resistance from the connecting wires.

To integrate the amplifier into the previous temperature monitoring circuit only the point at which the analog voltage is read needs to be adjusted. Initially, the analog read pin (A0) was connected to read the voltage across the thermistor directly from the voltage divider with a 10 kOhm pull-down resistor. With the op-amp, the analog voltage signal is read from the output of the amplifier circuit (pin 6 of the op-amp). This allows for the change in signal from the thermistor, which is typically very small, to be amplified and accurately monitored. The small change in signal was confirmed in the pre-lab simulation, only varying a couple of percent with every 500 Ohm increment of the thermistor value.

To test the stability of the circuit, air from the mouth was blown onto the thermistor in its configuration with the preamplifier stage. The output remained fairly stable, however, there were fluctuations in the output voltage from the circuit. To prevent external noise from getting through, a filter circuit is required. Although beyond the scope of this experiment and what has currently been covered in class, a possible solution could be an active filter circuit as they not only provide amplification but also can filter out unwanted signals. However, whether the noises are of high or low frequency decides what form of filter to use, in this scenario, it can be concluded that a low-pass filter would be the best choice. The reasoning behind this is the fact that overall the circuit is functioning with an expected DC input and output (low frequency), so any high frequencies that will interfere with the readings can be attenuated.

C. LED Alarm Indicators

A common application of thermistors in circuitry is to provide real-time readings of a system's temperature. This is important because components, like resistors and capacitors to name a few, have an ideal operating temperature that should not be surpassed to maintain proper function. This could mean either too low or too high a temperature would be detrimental to the electronics. The following figures showcase the function of the LED alarm system that was implemented to display what operating zone the circuit (thermistor) was in.

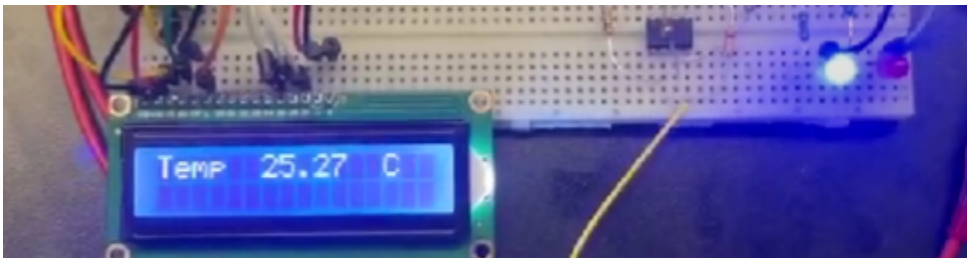

Figure 4: Blue LED On For Under 26 Degrees Celsius

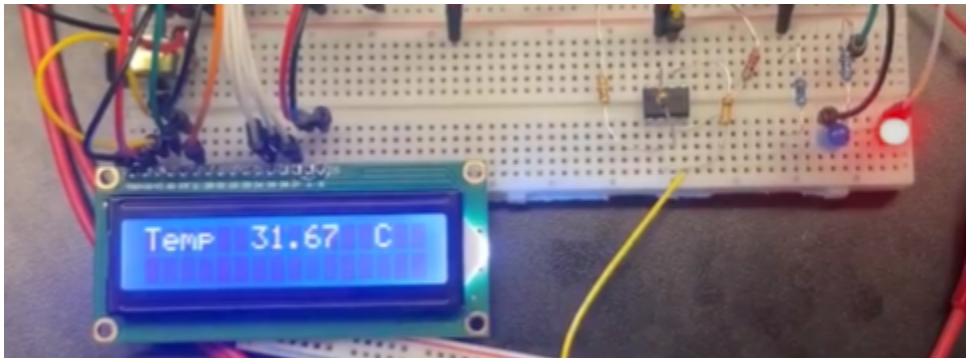

Figure 5: Red LED Flashing For Above 28 Degrees Celsius

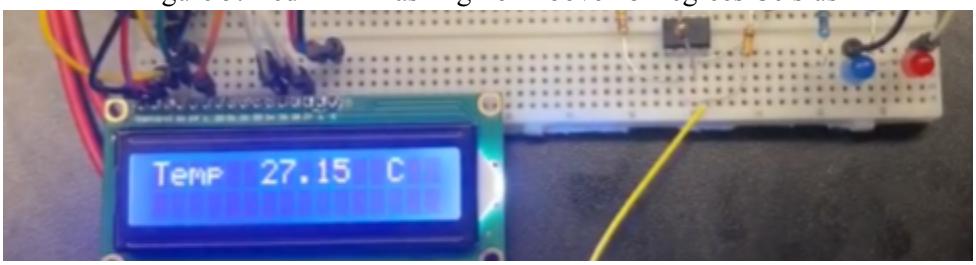

Figure 6: No LED On Between 26 and 28 Degrees Celsius

After careful consideration, three zones (temperature ranges) were decided to represent the different levels of temperature that the circuit would be operating in. For below 26 degrees Celsius, the circuit was in optimal operating conditions. Above 26 but below 28 degrees Celsius was a ‘gray zone’, in which the temperature was not too hot or too cold, nothing that would be worth identifying. Above 28

degrees was the ‘red zone’, in which temperatures above this were considered to be critical and would harm the circuitry. All functions were done with simple if statements.

V. Conclusion

The purpose of this three-part lab was to create a robust temperature monitoring system using a thermistor, low pass filter, and Arduino to create a visual indicator. In part one, a thermistor was used to sample the temperature in the surrounding environment. An Arduino was used to read the voltage across the thermistor and calculate the thermistor resistance value. The Arduino code was then used to convert that resistance value to a corresponding temperature value in degrees Celsius, and output this temperature to an LCD screen. Upon analyzing the behavior of the resistance of the thermistor across varying temperatures, it was found that the thermistor resistance value decreases with an increase in temperature. Therefore, the behavior of the resistance with a change in temperature suggested that the given thermistor was an NTC thermistor.

In part two, a 741 op-amp was used to amplify the voltage across the thermistor prior to it being measured by the analog A0 pin on the Arduino. Doing so made the small variations in the signal more prominent, therefore allowing the Arduino to accurately measure small changes in the signal from the thermistor. This helped make the temperature readings more precise since those small signals were being accounted for. The temperature readings were not as precise in part one because the variation in the signal from the thermistor is typically very small, which is difficult for the Arduino to monitor.

Lastly, for part three, LEDs were incorporated into the circuit, and connected to the Arduino, to create a visual indicator for the surrounding temperature. Two LEDs were used; a blue LED and a red LED. The Arduino was programmed to illuminate only the blue LED for a measured temperature of under 26 degrees Celsius and illuminate only the red LED in a flashing sequence for a measured temperature of greater than 28 degrees Celsius. For a measured temperature of between 26 and 28 degrees Celsius, both LEDs were turned off. This effect was achieved with if and else statements in the Arduino code to turn pins for the corresponding LEDs on or off, based on the temperature reading. This combination of LEDs served as a visual indicator and alarm for the surrounding temperature. The blue LED indicated a cool temperature, both LEDs being off indicated a moderate temperature (dead zone), and a flashing red LED indicated a high temperature.

The completion of this lab provided students with a holistic understanding of applying circuit theory, amplifiers, and programming in a practical data sampling project to create a robust temperature monitoring system.

VI. References

[1] METE 2030 Electronic Application for Mechatronics - LAB Project: A Temperature Monitoring System, Ontario Tech, Oshawa, ON, 2023

VII. Appendices

A. Appendix A: PreLabs

1. Muzzammil's PreLab

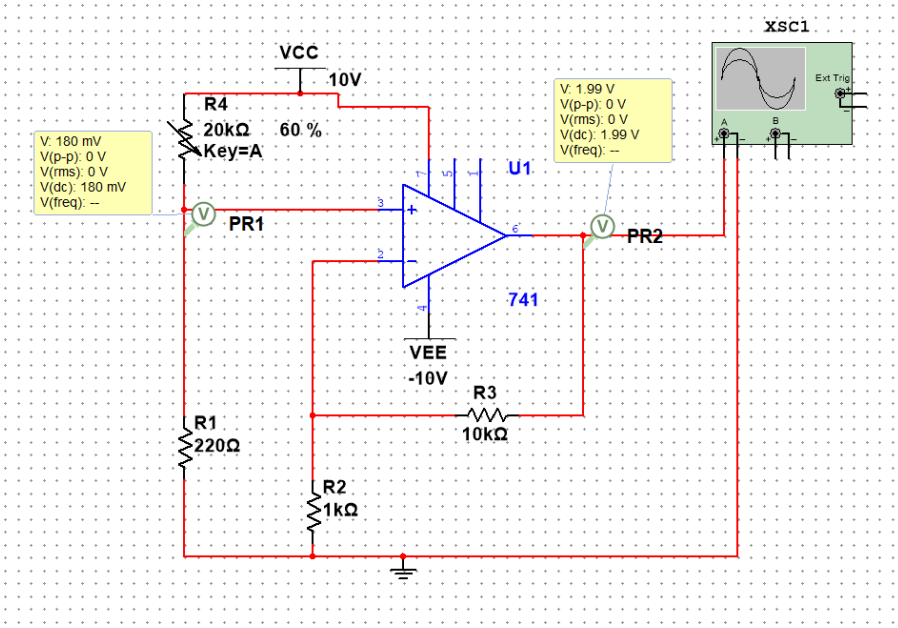

Figure 7: Negative Feedback Op-Amp Configuration Showcasing 11.1 Gain

Table 1: Tabulated Values Relating Thermistor Resistance to Output Voltage

Thermistor Resistance	12k	11.5k	11k	10.5k	10k
Output Voltage	1.992	2.077	2.169	2.269	2.380
Voltage Difference From Previous Resistance	0%	4.26%	4.43%	4.61%	4.89%

Sample Calculation:

12kΩ - 11.5 kΩ:

$$Percent\ Difference = \frac{2.077-1.992}{1.992} * 100 = 4.26\%$$

2. Taha's PreLab

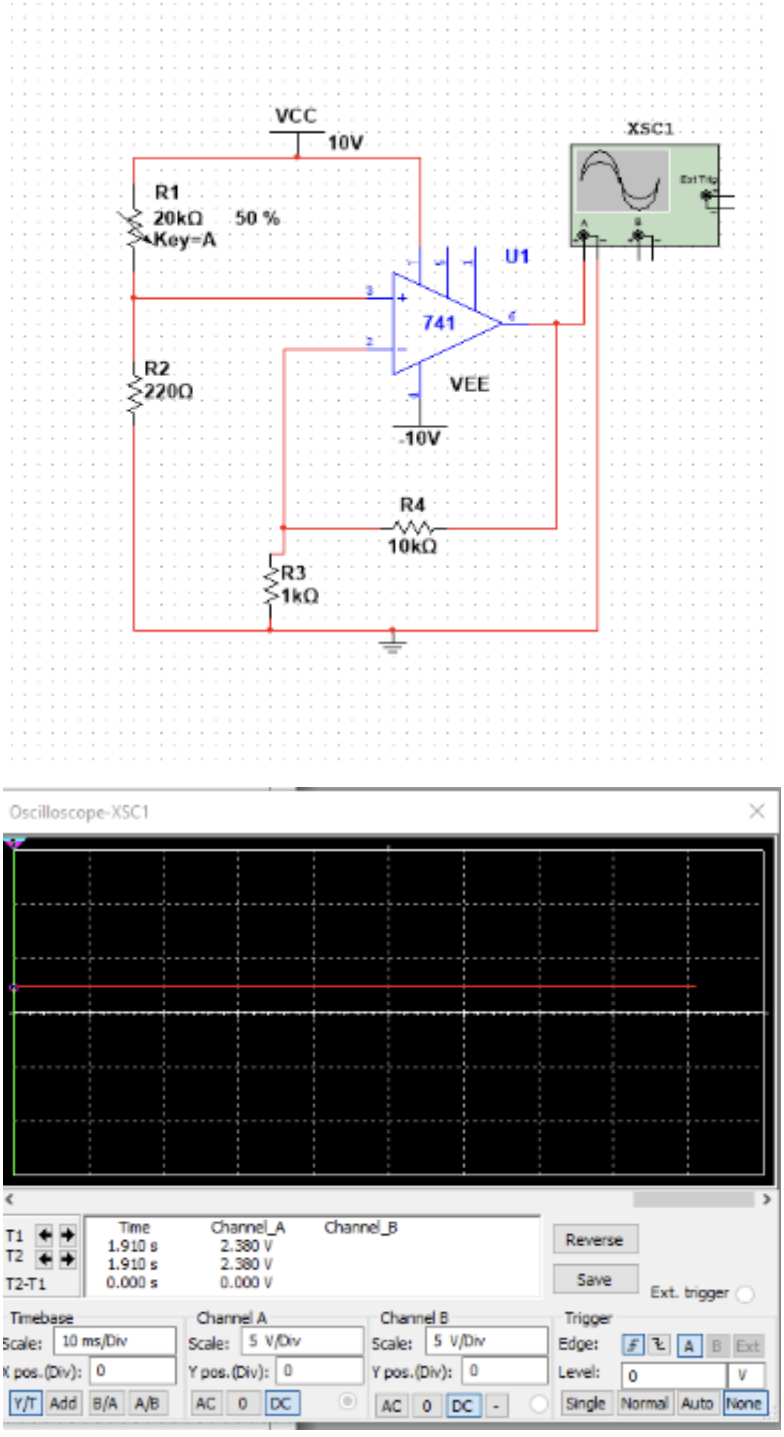

Figure 8: Thermistor circuit

Table 2: simulation results

Thermistor Resistance	12kΩ	11.5kΩ	11kΩ	10.5kΩ	10kΩ
Output Voltage	1.992 V	2.077 V	2.169 V	2.269 V	2.380 V

Percentage Change:

From 12 KΩ to 11.5 KΩ : 4.27%
From 11.5 KΩ to 11 KΩ : 2.99%
From 11 KΩ to 10.5 KΩ : 4.61%
From 10.5 KΩ to 10 KΩ : 4.89%

3. Muhammad's PreLab

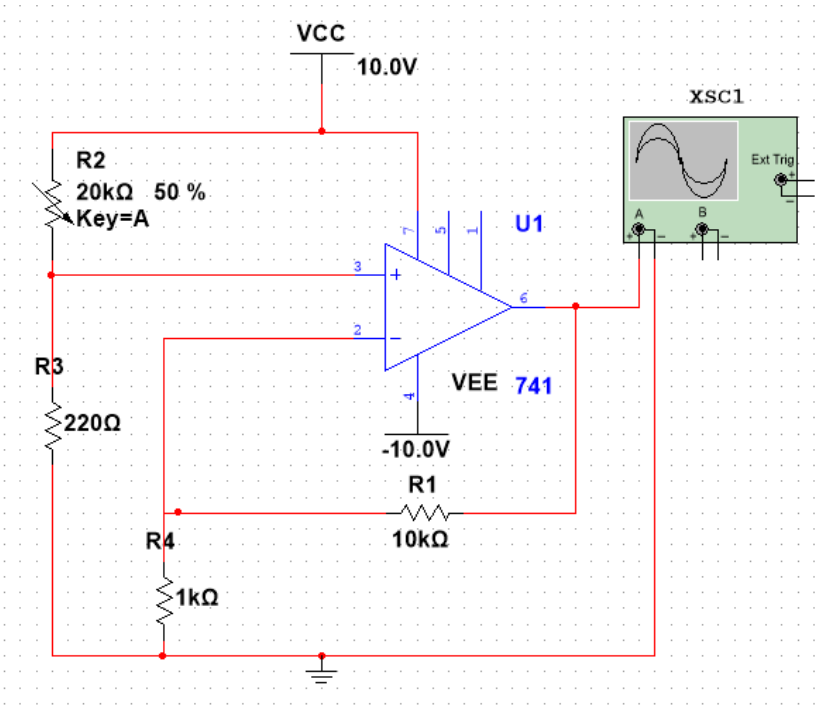

Figure 9: Op-Amp Circuit Construction in MultiSim

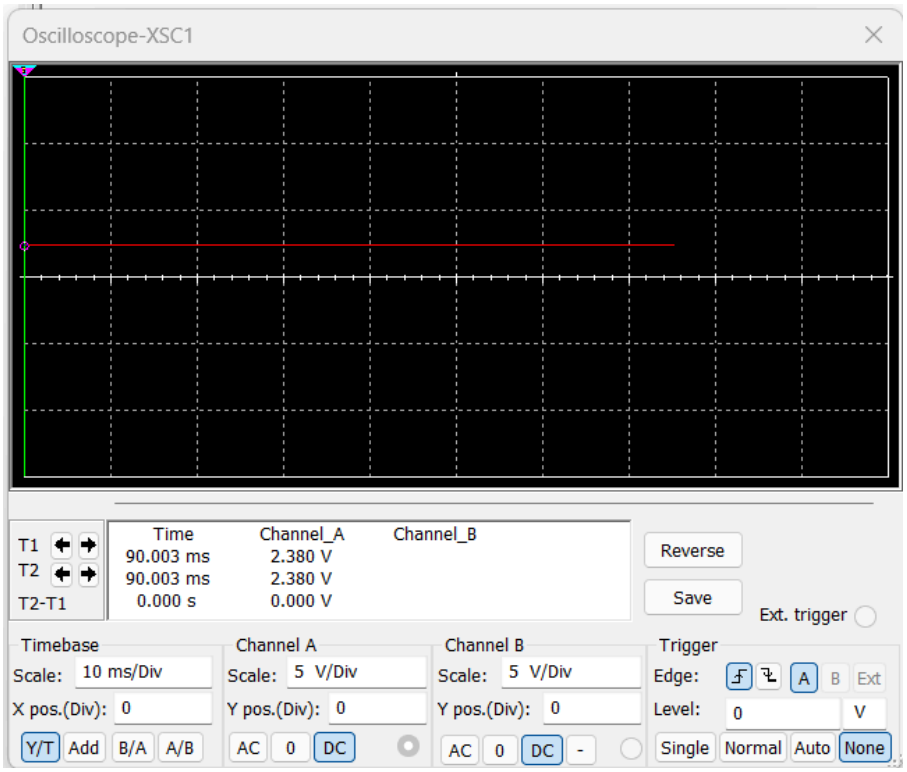

Figure 10: Op-Amp Circuit Simulation in MultiSim

Table 3: MultiSim Simulation Output Voltages for Thermistor Resistances 10 kΩ to 12 kΩ

Thermistor Resistance	12 kΩ	11.5 kΩ	11 kΩ	10.5 kΩ	10 kΩ
Output Voltage (% Change)	V_{o1} = 1.992 V	V_{o2} = 2.077 V (4.26% increase from V_{o1})	V_{o3} = 2.169 V (4.43% increase from V_{o2})	V_{o4} = 2.269 V (4.61% increase from V_{o3})	V_{o5} = 2.380 V (4.89% increase from V_{o4})

Calculations:

For 11.5 kΩ:

$$Percent\ Difference\ = \frac{2.077-1.992}{1.992} * 100 = 4.26\%$$

For 11 kΩ:

$$Percent\ Difference\ =\ \frac{2.169-2.077}{2.077} * 100\ =\ 4.43\%$$

For 10.5 kΩ:

$$Percent\ Difference\ =\ \frac{2.269-2.169}{2.169} * 100\ =\ 4.61\%$$

For 10 kΩ:

$$Percent\ Difference\ =\ \frac{2.380-2.269}{2.269} * 100\ =\ 4.89\%$$

B. Appendix B: Arduino Code

Thermometer | Arduino 1.8.19

File Edit Sketch Tools Help

Thermometer §

```
//Lab Group 8 (or 18, cant remember right now) METE 2030U
#include <LiquidCrystal.h>
int tempPin = 0;
//                BS  E  D4 D5  D6 D7
LiquidCrystal lcd(7, 8, 9, 10, 11, 12);

//pin which the temperature will be read from
int tempPin = 0;

//pins that are used for the LED indicators
int blueLed = 3;
int redLed = 2;

void setup()
{
  lcd.begin(16, 2);

  //set led identifier pins as output
  pinMode(blueLed,OUTPUT);
  pinMode(redLed,OUTPUT);
}
void loop()
{
  int tempReading = analogRead(tempPin);
  // This is OK
  double tempK = log(10000.0 * ((1024.0 / tempReading - 1)));
  tempK = 1 / (0.001129148 + (0.000234125 + (0.0000000876741 * tempK * tempK )) * tempK );       //  Temp Kelvin
  float tempC = tempK - 273.15;            // Convert Kelvin to Celcius
  float tempF = (tempC * 9.0)/ 5.0 + 32.0; // Convert Celcius to Fahrenheit
  /*  replaced
    float tempVolts = tempReading * 5.0 / 1024.0;
    float tempC = (tempVolts - 0.5) * 10.0;
    float tempF = tempC * 9.0 / 5.0 + 32.0;
  */

  //BLUE LED CODE
  if (tempC < 26)
    {
      digitalWrite(blueLed, HIGH);
      digitalWrite(redLed, LOW);
    }


   //RED LED CODE
   if(tempC > 28)
   {
    digitalWrite(redLed, HIGH);
    delay(1000);
    digitalWrite(redLed, LOW);
   }

   //In between code
 else if (tempC >= 26 && tempC <=28)
    {
      digitalWrite(blueLed, LOW);
      digitalWrite(redLed, LOW);
    }

  // Display Temperature in C
  lcd.setCursor(0, 0);
  lcd.print("Temp         C  ");
  // Display Temperature in F
  //lcd.print("Temp         F  ");
  lcd.setCursor(6, 0);
  // Display Temperature in C
  lcd.print(tempC);
  // Display Temperature in F
  //lcd.print(tempF);
  delay(500);
}
```

Done Saving.