Module Four Laboratory Report

Heat Transfer

ME3902 - Project-Based Engineering Experimentation

Submitted by:

Matthew Adam

mladam@wpi.edu

Submitted to:

Professor Ahmet Can Sabuncu

6/18/2023

Summary

Summary **2**
List of Figures **2**
Introduction **3**
Objectives: 3
Procedure: **3**
A) Assemble circuit 3
B) Modify circuit and implement code 4
C) Calibrate 7
D) Run Experiment 8
Simple Comparison 9
Heat Transfer Comparison 10
Conclusions **11**
Facilities **11**

List of Figures

Figure 1: Original Circuit Diagram for Thermistor
Figure 2: Documented Source Code for Thermistor
Figure 3 & 4: Graph and Table of True Sensor Data vs Thermistor Measured Data
Figure 5: Can with 1 set dissipation fins
Figure 6: Control Temperature Measurement vs Time
Figure 7: Two Fins Measurement Vs Time

Introduction

This set of lessons was designed to expose users to how to measure and interact with temperature using microcontrollers, thermistors, and thermocouples. From designing the circuit to implementing working code to do precisely what is needed for the experiment, I am able to understand some of the basics of how to measure temperature for any future applications.

The goal of this lab was to design, implement, and analyze a circuit to measure the temperature of an object.

Objectives:

- Implement a thermistor circuit using a thermistor and an ADS1115 chip designed to measure resistance changes in the thermistor.
- Implement code to measure the temperature using the resistance difference in the thermistor.
- Calibrate the sensor such that the temperature is within an acceptable range of the temperature of a standard alcohol thermometer.
- Run an experiment testing the capabilities of different cooling mechanisms.

Procedure:

A) Assemble circuit

Based on the lab assignment, I had a basic setup for the circuit as seen in figure 1. This is a circuit containing a thermistor, an ADS1115 chip, a reference resistor, and an arduino.

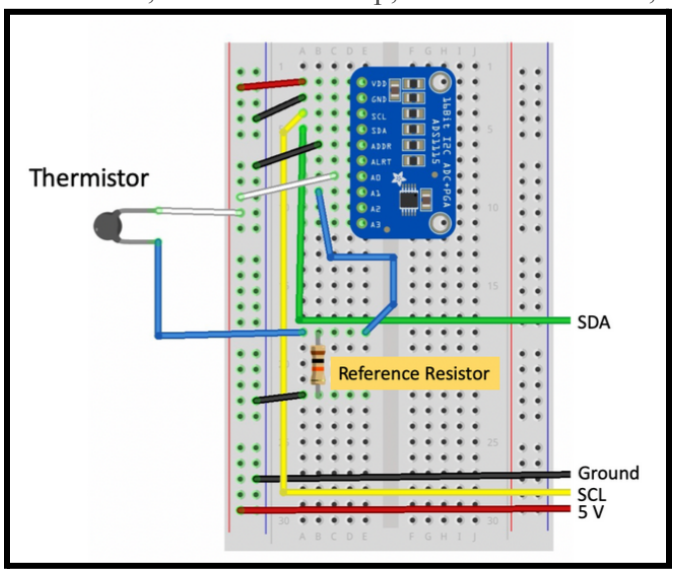


Figure 1: Original Circuit Diagram for Thermistoruajcn

B) Modify circuit and implement code

Modify the existing circuit to ensure that the thermistor works. In order to do this, I had to alter where the resistor was in the circuit to ensure that the ADS1115 chip worked correctly. I also switched A0 and A1 pins from the ADS1115 chip from the diagram above. Additionally, during this step the code was written such that the ADS chip could monitor the thermistor resistance and successfully report and record the information to the computer through a serial output. The code can be found below in Table 1.

```
#include <Wire.h>
//  by John Sullivan and Ahmet Can Sabuncu, ME3902 Summer 2021
//
#include <Adafruit_ADS1X15.h>  // same library for both the ads1015 and ads1115
Adafruit_ADS1115 ads1115;  // Declare an instance of the ADS1115 at address slot 0x48
int16_t rawADCvalue;  // The is where we store the value we receive from the ADS1115
                      //  int16_t is a 16 bit signed integer range = -32768 to +32767
                      // scalefactor = max Voltage /(  (2^15)-1 = max Voltage/(32767) for 16 bit with most
                      //   significant bit reserved for sign (+ or -)

float volts = 0.0;          // The result of applying the scale factor to the raw value

float bit_res = 0.1875000;
//float bit_res = 0.1250000;
//float bit_res = 0.0625000;
//float bit_res = 0.0312500;
//float bit_res = 0.0156250;
//float bit_res = 0.0078125;     //  This is the bit resolution in [mV] will change with the gain, please refer to the table below

float uV = 0.0;             //  This is just volts times a million [uV]
float a0 = 0, a1 = 2.5928e-2, a2 = -7.602961e-7, a3 = 4.637791e-11;
//  These are the NIST coefficients for converting voltage readings to temperature
```

```
float a4 = -2.165394e-15, a5 = 6.048144e-20, a6 = -7.293422e-25;  // These are the NIST coefficients for converting voltage readings to temperature

// PLEASE LOOK UP THE NIST COEFFICIENTS FOR YOUR THERMOCOUPLE


float TempDegC=0;
unsigned long StartTime = 0;
//                                //  Gain   Max Volt   ads1015      ads1115
// ads1115.setGain(GAIN_TWOTHIRDS);// 2/3x gain +/- 6.144V  1 bit = 3mV (default)  1 bit = 187.5 micro-V
// ads1015.setGain(GAIN_ONE);    // 1x gain   +/- 4.096V  1 bit = 2mV      1 bit = 125. micro-V
// ads1015.setGain(GAIN_TWO);    // 2x gain   +/- 2.048V  1 bit = 1mV      1 bit =  62.5 micro-V
// ads1015.setGain(GAIN_FOUR);   // 4x gain   +/- 1.024V  1 bit = 0.5mV      1 bit =  31.25 micro-V
// ads1015.setGain(GAIN_EIGHT);  // 8x gain   +/- 0.512V  1 bit = 0.25mV     1 bit =  15.625 micro-V
// ads1015.setGain(GAIN_SIXTEEN);  // 16x gain  +/- 0.256V  1 bit = 0.125mV    1 bit =  7.8125 micro-V

void setup(void)
{
  Serial.begin(9600);
  ads1115.setGain(GAIN_TWOTHIRDS);  // Set gain to 16x
  ads1115.begin(0x48);
  //  start a timer
  StartTime = millis();
}

float do_calculations(float uV){

//  V = I * R
//  current stays same in series

//  voltage and current in resistor
```

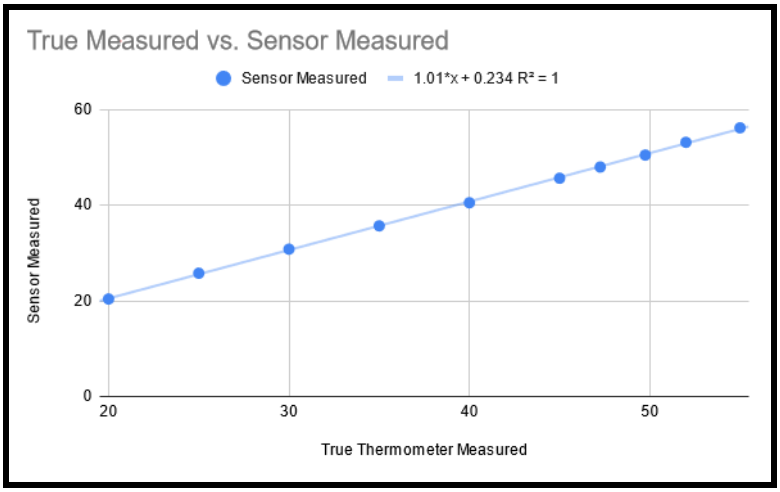
```
  float Vr = uV / 1000000;
  float I = Vr / 10000; // 10,000 Ohm Resistor

//  find voltage in thermistor
  float Vt = 5.0 - Vr;
  float Rt = Vt / I;

//  using the Steinhart Hart Equation from Canvas:
//  setting for this thermistor for this equation
  float Beta = 3950;
  float To = 22.75+273; // kelvin
  float Ro = 10530;

//  Equation (cut up into intermediates)
  float a_log = log( Rt / Ro);
  float bottom = ( a_log / Beta ) + (1/To);
  float T = 1/bottom;

//  return Rt; // for calibration of Ro
  return T - 273;
}

float perform_calibration(float T){
  return (1.01*T) + 0.234;
}

void loop(void)
{
  rawADCvalue = ads1115.readADC_Differential_0_1(); // Differential
voltage measurement between A0 and A1 on the ADC chip
  volts = rawADCvalue * bit_res; // Convert rawADC number to voltage
in [mV]
  uV = volts*1e3; // Express the voltage in microVolts
//  TempDegC = a0 + a1*uV + a2*pow(uV,2) + a3*pow(uV,3) +
a4*pow(uV,4) + a5*pow(uV,5) + a6*pow(uV,6);
  unsigned long CurrentTime = millis();
  float ElapsedTime = (CurrentTime-StartTime)/1000.0;
//  Serial.print("Time (sec) ");
  Serial.print(ElapsedTime,3);
```

```
//  Serial.print(",   microVolts Measured = ");
//  Serial.print(uV,2);

  float T = do_calculations(uV);

  T = perform_calibration(T);

  Serial.print(",");
//  Serial.print(",   Temperature (deg C) = ");
  Serial.println(T,2);
//  Serial.println();
  delay(500);
}
```

Figure 2: Documented Source Code for Thermistor

C) Calibrate

I calibrated the circuit by running a small experiment. I placed the thermistor and an alcohol based thermometer in the same heated water bath. Using the assembled circuit to measure data from the thermistor, I took measurements from both devices and recorded them in excel. Using the chart software, I was able to find the line of best fit for the data. In the code in table 1, I used the equation of the line of best fit to find the true temperature from the thermistor.



True Measured	Sensor Measured
55	56.2
52	53.17
49.75	50.5
47.25	48
45	45.65
40	40.52
35	35.7
30	30.82
25	25.79
20	20.39

Figure 3 & 4: Graph and Table of True Sensor Data vs Thermistor Measured Data

D) Run Experiment

Using the calibrated sensor, I was able to accurately measure the effectiveness of heat dissipation fins on a heated can on water. Using cans, I made 2 sets of dissipation fins that fit onto another can as seen in figure 5. I then conducted a control with > 90 degree water in a can with no fins and measured the temperature at every given interval. Finally, I fixed 2 sets of dissipation fins to the can and repeated the experiment with > 90 degree water. I was now able to compare the cooling rates of the control with no fins and the trial with 2 sets of fins.

Figure 5: Can with 1 set dissipation fins

Results and Discussion

PLEASE PROVIDE YOUR RESULTS IN THIS SECTION (FIGURES AND TABLES)

The direct data from the experiment was the temperature vs time graphs for both the control and the two-fin trial. (Figure 6 and Figure 7) From this data, we can clearly see that The fins provided faster cooling time. To empirically measure this, some information was calculated based off the collected data.

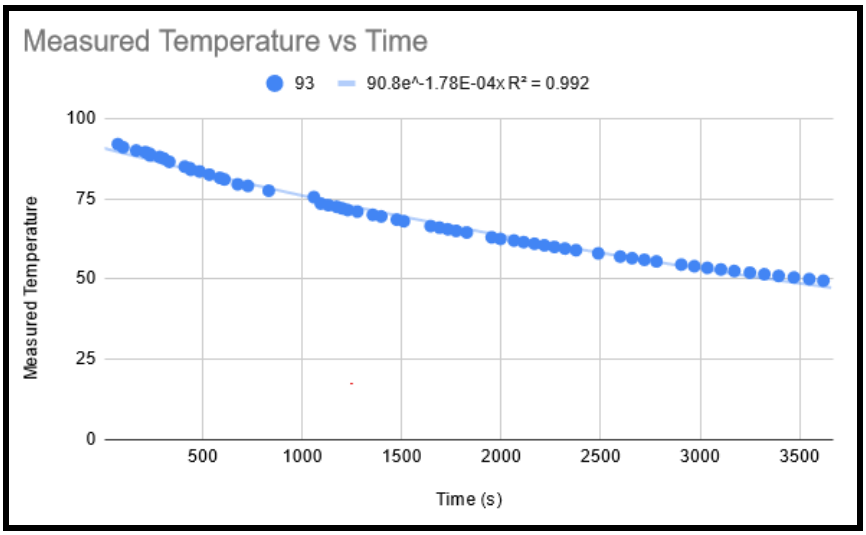


Figure 6: Control Temperature Measurement vs Time

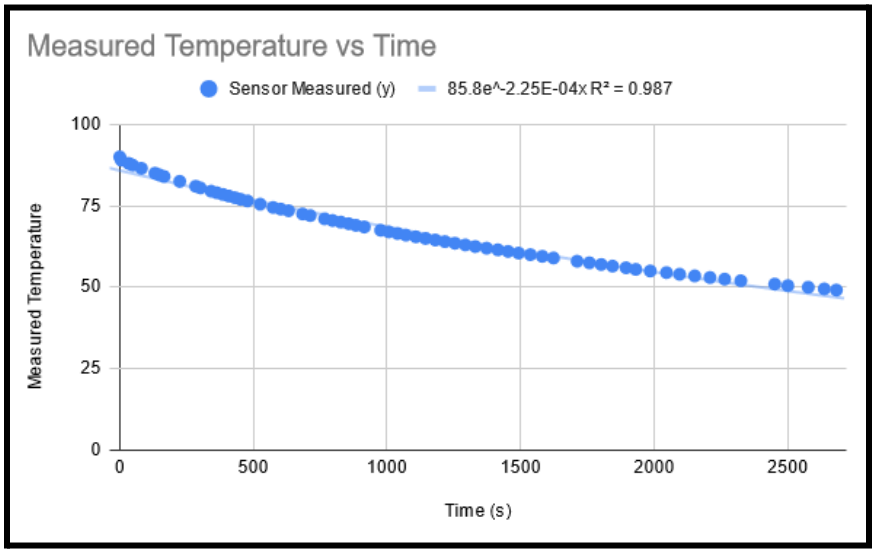


Figure 7: Two Fins Measurement Vs Time

Simple Comparison

The simplest comparison of the data is by picking a starting and ending point and comparing the time it took for each trial to achieve the temperature difference. Arbitrarily picking 90 degrees celsius and 50 degrees celsius, I can confirm that

$$Time_{\ 90 -> 50\ control} = \frac{Time_{50\ degrees} - Time_{90\ degrees}}{60\ seconds\ /\ min} = \frac{3548 - 165}{60\ seconds\ /\ min} = 56\ minutes$$

$$Time_{\ 90 -> 50\ fin\ trial} = \frac{Time_{50\ degrees} - Time_{90\ degrees}}{60\ seconds\ /\ min} = \frac{2576 - 0}{60\ seconds\ /\ min} = 43\ minutes$$

It is now a straightforward comparison of time; the fins caused the fin trial to cool from 90 degrees to 50 degrees 13 minutes faster than if there were no fins on the device.

Heat Transfer Comparison

A more complicated comparison of the data is by using the heat transfer coefficient. Using Fourier's law, I know that the heat flux q is equal to the difference in the temperature to its surroundings times the heat transfer coefficient. Additionally, I know that the heat flux is equal to the specific heat times the difference in the temperature from time 1 to time 2. From these two pieces of information, I was able to calculate the K heat transfer coefficient between every two data points within both of my datasets. Most of these heat transfer coefficients were very similar, so finding the average gave the final average of the heat transfer coefficient for the trial.

By analyzing the data, I was able to find that the fins provided a heat transfer coefficient 48.48 while the control provided a heat transfer coefficient of only 48.18.

$$K\ trial\ = \frac{\sum K_{\ t}}{total\ K}$$

$$K\ = \frac{q_{\ 1-2}}{Tcan - Troom}$$

$$q_{\ 1-2} = Cu\ (T1\ -\ T2)$$

Conclusions

From the difference in the heat transfer coefficients, we can conclude that the fins were much better at dissipating heat from the heated can of water than without fins. Using a thermistor, it is possible to conduct complex experiments based on temperature. Microcontrollers are able to interact with the world in many ways.

The objectives of this lab were to:

- Implement a thermistor circuit using a thermistor and an ADS1115 chip designed to measure resistance changes in the thermistor.
- Implement code to measure the temperature using the resistance difference in the thermistor.
- Calibrate the sensor such that the temperature is within an acceptable range of the temperature of a standard alcohol thermometer.
- Run an experiment testing the capabilities of different cooling mechanisms.

Each of these objectives were completed during the course of this lab. We were able to construct the circuit, write the driving code, calibrate the sensor using the code, and run different kinds of experiments with the code. I now understand better how microcontrollers can be used in an engineering application.

Facilities

1 Arduino Uno
1 thermistor
1 resistor(s) (465 ohms, 5% tolerance) > 40 ohms (given that the power source is 3.3V).
1 half-breadboard
Analog to Digital Converter, 16-bit I2C ADC+PGA (ADS-1115)

References

1. Professors Notes and Canvas Page