Daniel Diamont

TA: Saadallah Kassir

EE 445L

Lab 9 Report

A) OBJECTIVES

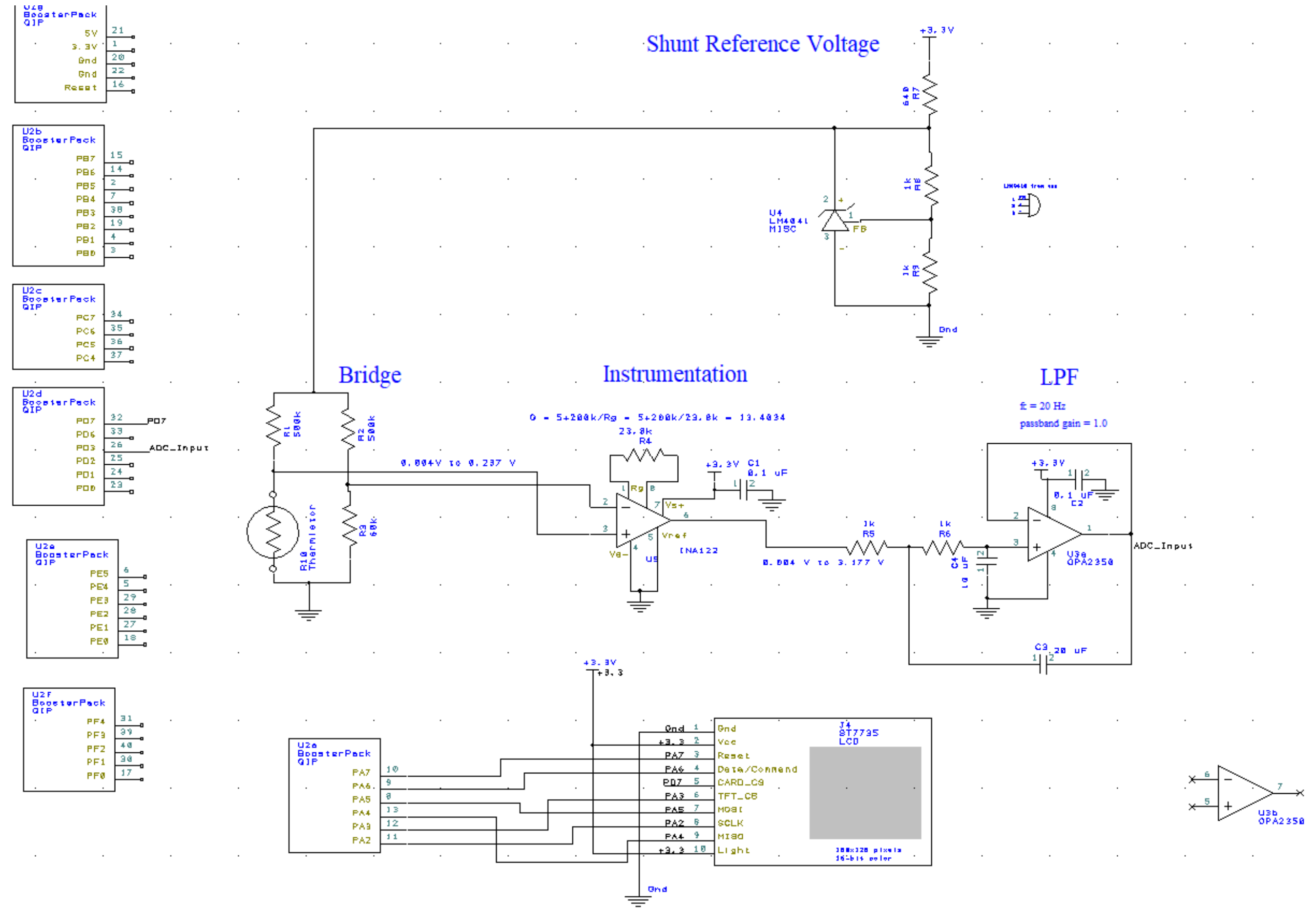
Goals

The goals of this lab are to study ADC conversion, the Nyquist Theorem, and to develop a data acquisition system involving transducers, instrumentation amplifiers, and filters. Specifically, the lab involves the design and implementation of a digital thermometer with a temperature range from 20 to 40 degrees Celsius with an accuracy of 1 degree Celsius and a resolution of 0.1 degrees Celsius or better. The knowledge needed to do this job include – but is not limited to – the following:

- Understanding bridge circuits for precise instrumentation
- Understanding the non-linear temperature-resistance curve of thermistors
- Ability to construct a calibration procedure for a transducer
- Signal conditioning with instrumentation amplifiers and filters
- Sampling analog signals and understanding the conditions for full reconstruction of a signal
- Time-domain and frequency domain analysis of signals, noise, and the effect of filtering while sampling
- 2-pole Butterworth LPF design for signal conditioning
- Using shunt reference voltages for instrumentation
- Performing software conversion of transducer data to a meaningful digital format using equations and/or look-up tables
- Real time sampling using a hardware timer to trigger ADC conversion
- Graphic display of temperature vs. time on LCD screen

B) HARDWARE DESIGN

Circuit Schematic ***(check Lab09_Artist.sch in .zip file for better resolution)***



C) SOFTWARE DESIGN

Calibration Calculations

The NTC thermistor that we used for this lab (Part #: BC2432-ND) follows a resistance-temperature relationship that can be modeled by the following equation:

$$R = R_o e^{\frac{\beta}{\frac{1}{T} - \frac{1}{T_o}}}, \qquad where\ R_o is\ the\ resistance\ at\ reference\ temperature\ T_o$$

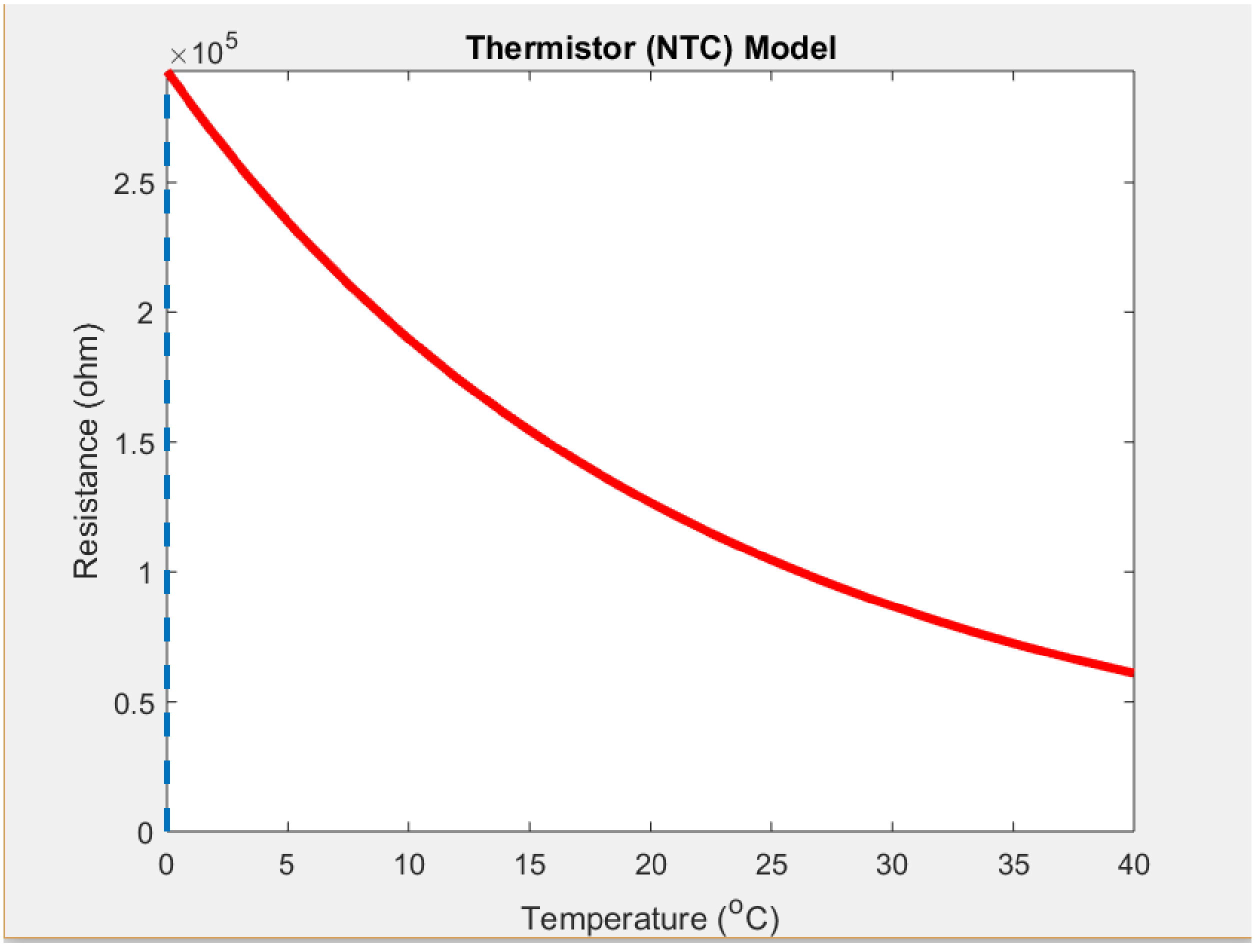
To obtain the β-term of the NTC Thermistor equation, I measured the resistance of my thermistor at two temperatures as follows:

Resistance	Temperature
104.5 kΩ	25^oC
70 kΩ	36^oC

Turning the temperature into Kelvin, the β-term (in Kelvin) can be calculated to be:

$$\beta = 3357.55\ K$$

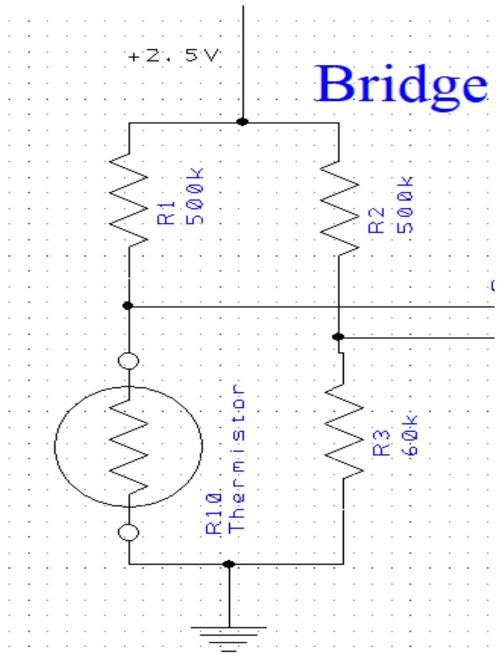
This produces our approximate thermistor curve as follows:



We can see that at the end of our temperature range of interest ($40^{o}C$), the resistance of our thermistor approaches ~ 60 kΩ, and at the beginning of our region of interest ($20^{o}C$) it approaches ~ 125 kΩ. Indeed, when calculating the resistances with our acquired β-value, we arrive at:

Resistance	**Temperature**
126.629 kΩ	$20^{o}C$
60.932 kΩ	$40^{o}C$

Given these values, we can choose the resistor values of the Wheatstone bridge used to transform the variable resistance of the thermistor into a voltage that we can condition and process. The bridge circuit is as follows:

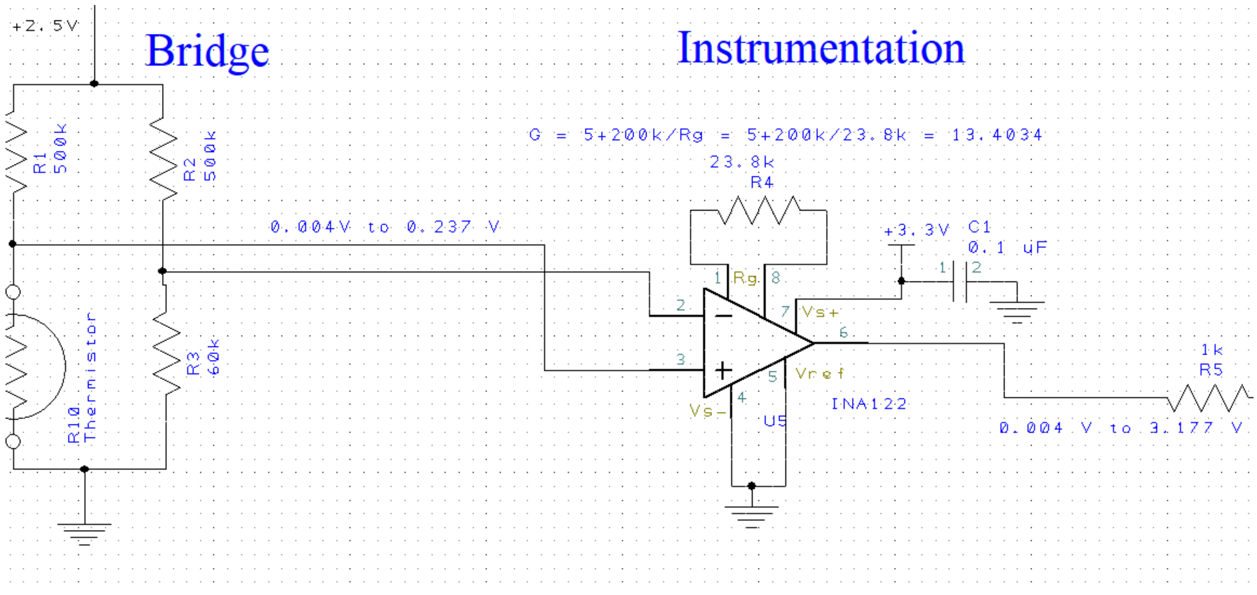


Using voltage division equations, the range of input resistances to the bridge can be represented as a voltage as follows:

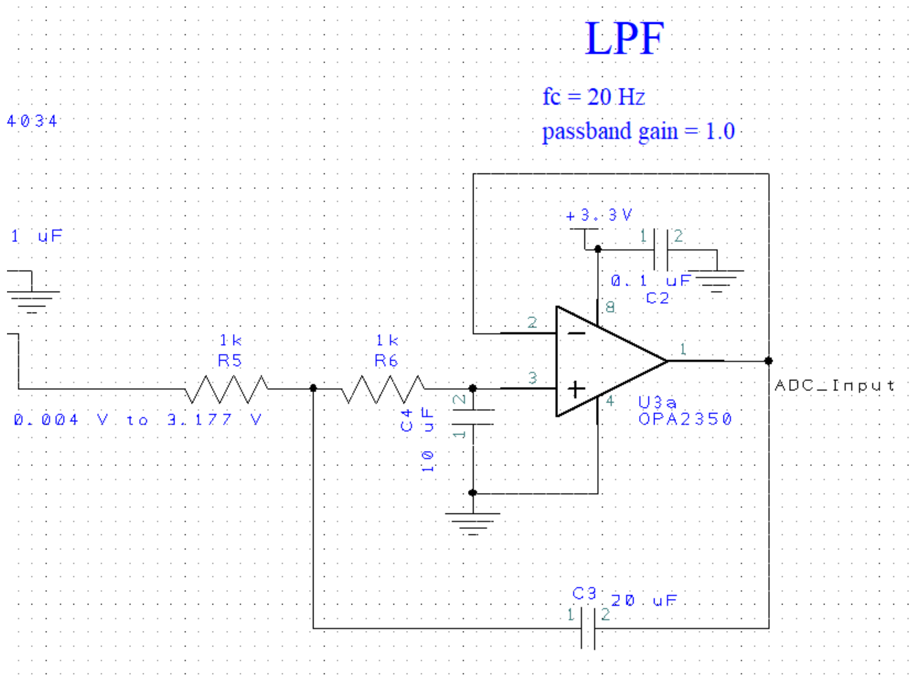
$$V_{@20^oC} = 2.5V\left(\frac{500k\Omega}{500k\Omega + 126.629k\Omega}\right) - 2.5V\left(\frac{500k\Omega}{500k\Omega + 60k\Omega}\right) = 0.237\ V$$

$$V_{@40^oC} = 2.5V\left(\frac{500k\Omega}{500k\Omega + 60.932k\Omega}\right) - 2.5V\left(\frac{500k\Omega}{500k\Omega + 60k\Omega}\right) = 0.004\ V$$

Since our Analog-to-Digital converter module can only take voltages from 0.0V to 3.3V as inputs, we run the instrument signal from the resistor bridge through an instrumentation amplifier with a gain factor of 13.4034 to map the voltage range from 0.004 V to 3.177V. The gain of the instrumentation amplifier is calculated as shown in the diagram on the next page:



The instrumentation amplifier has the advantage of neglecting common-mode voltage between its inputs. This increases the signal-to-noise ratio by removing any noise shared by the two input nodes and only amplifying the voltage between them. Additionally, the signal conditioned by a 2-pole Butterworth Low-Pass-Filter chosen with a cut-off frequency of ~20 Hz to eliminate any unwanted high-frequency noise from our transducer signal. Note the frequency of our transducer signal is expected to be less than or equal to 10 Hz. The passband gain of the filter is 1.0. After this conditioning stage, the signal is ready to be sampled by our Analog-to-Digital converter.



LUT Generation

The ADC on the TM4C123GH6PM has 2^{12} input codes (4096 alternatives) available to convert an analog signal. Since the format of interest of the data is in units of temperature, we must find some function *f[n]* that maps the discrete input codes of the ADC to temperature.

$$f: x \in [0,4095] \rightarrow temperature \in [20^oC, 40^oC]$$

We can do this by first finding the transfer function T which maps the continuous transducer resistance values to temperatures. This is the function obtained from running the signal through the bridge circuit.

$$T: V_{temperature \in [20^oC,40^oC]} \rightarrow temperature \in [20^oC, 40^oC]$$

This requires rearranging the β-term equation to the form:

$$T = \frac{\beta}{\ln\left(\frac{R_{temperature}}{R_o * e^{\frac{\beta}{T_o}}}\right)}$$

The calculation of V_{bridge} is dependent on the thermistor resistor input $R_{thermistor}$ as follows:

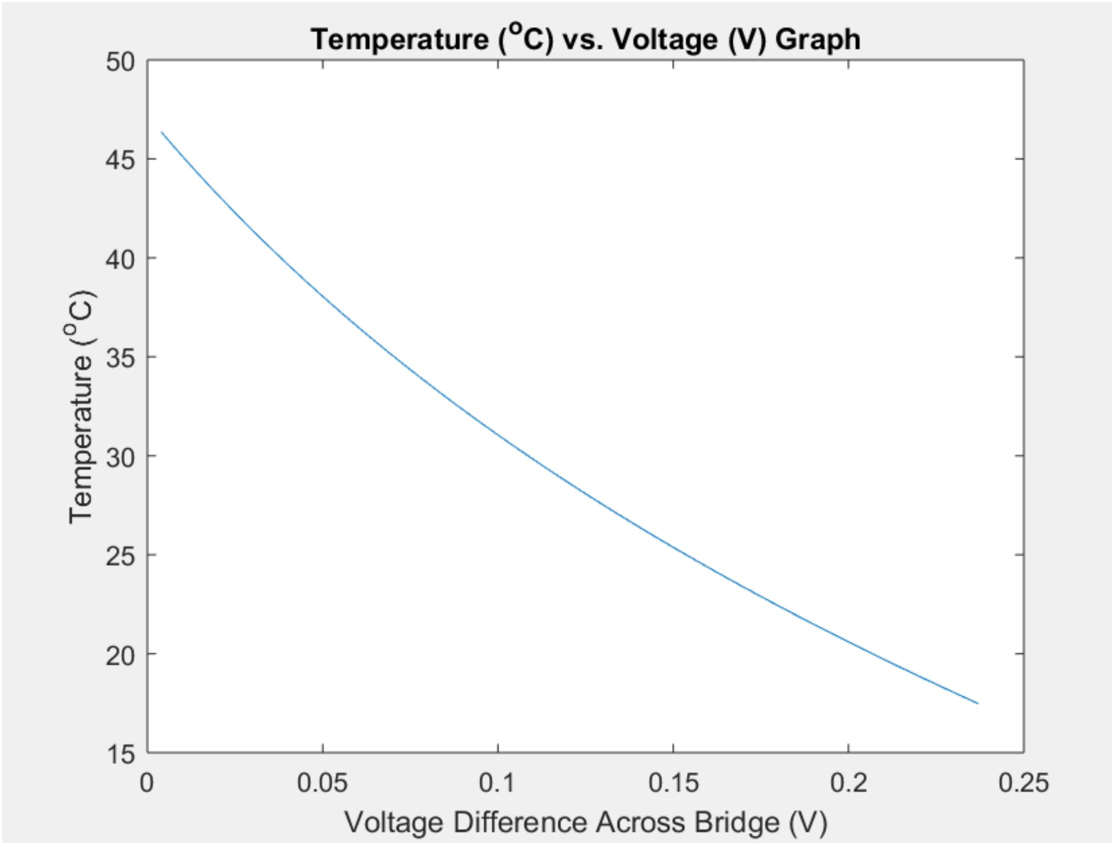
$$V_{bridge} = 2.5\,V\left(\frac{R_{temperature}}{R_{temperature} + 500k\Omega}\right) - V_o$$

$$(V_{bridge} + V_o)\frac{500k\Omega}{2.5\,V} = R_{temperature} - (V_{bridge} + V_o)\frac{R_{temperature}}{2.5\,V}$$

$$(V_{bridge} + V_o)\frac{500k\Omega}{2.5\,V} = R_{temperature}(1 - \frac{\mathrm{V_{bridge}} + \mathrm{V_o}}{2.5\,V})$$

$$R_{temperature} = \frac{(V_{bridge} + V_o) * \frac{500k\Omega}{2.5\,V}}{1 - \frac{(\mathrm{V_{bridge}} + \mathrm{V_o})}{2.5\,V}}$$

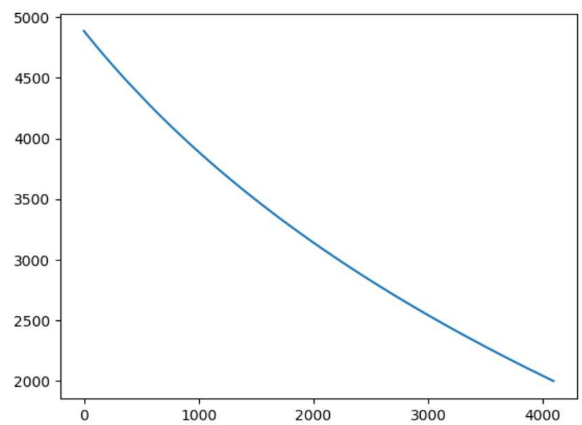
Now, we can substitute the resistance into the rearranged β-term equation and plot the temperature response for our full-range of transducer voltage signals as such:

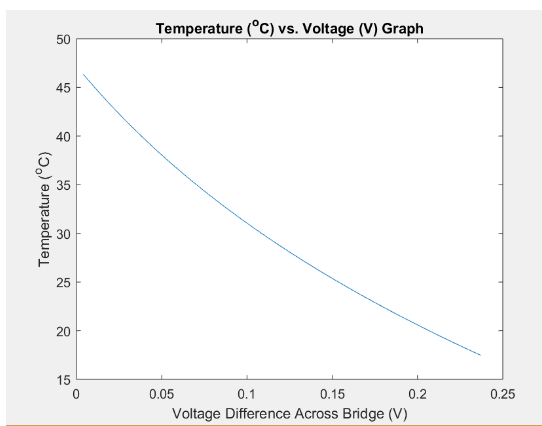


This table can then be discretized to generate function *f* as follows to generate a 4096 element look-up table that will map the values of the ADC to the temperature measured by the thermistor.

```python
#Daniel Diamont
#EE445L

#This file was made for the purposes of generating a thermistor bit to temperature LUT

#import numpy and matlab plotting capabilities

import math as m
import numpy as np
import matplotlib.pyplot as plt

wav = []
LUT = []

B = 3357.55;
V0 = 0.163;
T0 = 298.15;
R = 500e3;
R0 = 104.5e3;

Vt = 0.004
step = 0.233/4096

denom = R0*np.exp(-B/T0);

for i in range(0,4095):
        y = (B/(np.log(((R*((Vt + (step*i) + V0)/2.5))/(1 - ((Vt + (step*i) + V0)/2.5))/denom)))) - 273.15
        wav.append(int(y*100))

#plot of sine wave
print(wav);
plt.plot(wav);
plt.show();
```

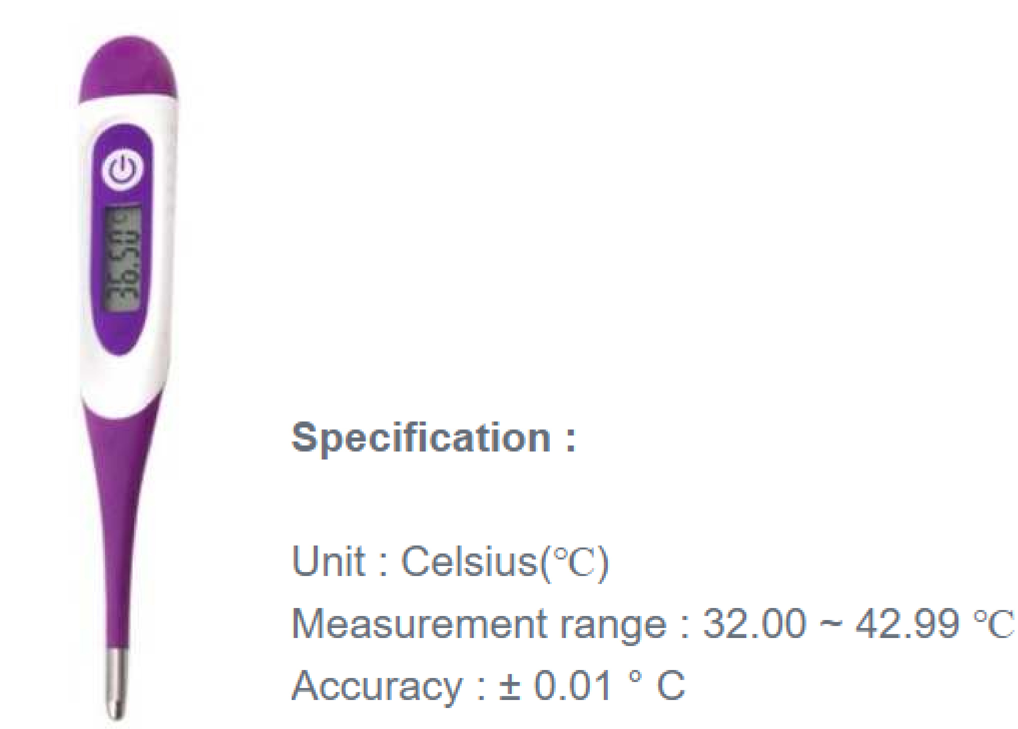
This code generates the temperature vs. voltage plot except it is discretized:





Calib.h

When implemented on the micro-controller, to achieve a similar curve as shown above by using the look-up table, the function had to be implemented with an offset of $-2.47^{o}C$ in order to achieve the same temperature as the following thermometer under the same conditions:



[thermometer link]

This offset was stored as a #define in calib.h

ADC.c and ADC.h (check attached files to .zip folder)

```c
void ADC0_InitTimer0ATriggerSeq3PD3(uint32_t period){
  volatile uint32_t delay;
  SYSCTL_RCGCADC_R |= 0x01;     // 1) activate ADC0
  SYSCTL_RCGCGPIO_R |= 0x08;    // Port D clock
  delay = SYSCTL_RCGCGPIO_R;    // allow time for clock to stabilize
  GPIO_PORTD_DIR_R &= ~0x08;    // make PD3 input
  GPIO_PORTD_AFSEL_R |= 0x08;   // enable alternate function on PD3
  GPIO_PORTD_DEN_R &= ~0x08;    // disable digital I/O on PD3
  GPIO_PORTD_AMSEL_R |= 0x08;   // enable analog functionality on PD3
  ADC0_PC_R = 0x01;             // 2) configure for 125K samples/sec
  ADC0_SSPRI_R = 0x3210;        // 3) sequencer 0 is highest, sequencer 3 is lowest
  SYSCTL_RCGCTIMER_R |= 0x01;   // 4) activate timer0
  delay = SYSCTL_RCGCGPIO_R;
  TIMER0_CTL_R = 0x00000000;    // disable timer0A during setup
  TIMER0_CTL_R |= 0x00000020;   // enable timer0A trigger to ADC
  TIMER0_CFG_R = 0;             // configure for 32-bit timer mode
  TIMER0_TAMR_R = 0x00000002;   // configure for periodic mode, default down-count settings
  TIMER0_TAPR_R = 0;            // prescale value for trigger
  TIMER0_TAILR_R = period-1;    // start value for trigger
  TIMER0_IMR_R = 0x00000000;    // disable all interrupts
  TIMER0_CTL_R |= 0x00000001;   // enable timer0A 32-b, periodic, no interrupts
  ADC0_ACTSS_R &= ~0x08;        // 5) disable sample sequencer 3
  ADC0_EMUX_R = (ADC0_EMUX_R&0xFFFF0FFF)+0x5000; // 6) timer trigger event
  ADC0_SSMUX3_R = 4;            // 7) PD4 is channel 4
  ADC0_SSCTL3_R = 0x06;         // 8) set flag and end
  ADC0_IM_R |= 0x08;            // 9) enable SS3 interrupts
  ADC0_ACTSS_R |= 0x08;         // 10) enable sample sequencer 3
  NVIC_PRI4_R = (NVIC_PRI4_R&0xFFFF00FF)|0x00004000; // 11)priority 2
  NVIC_EN0_R = 1<<17;           // 12) enable interrupt 17 in NVIC
```

```
volatile uint32_t ADCvalue;
void ADC0Seq3_Handler(void){
  ADC0_ISC_R = 0x08;          // acknowledge ADC sequence 3 completion

    //store value in fifo
    FiFo_Put(ADC0_SSFIFO3_R);

}
```

The key to the ADC conversion process lies in triggering the ADC with a general-purpose timer. When the conversion process is finished, the value of the ADC is put into a software FIFO for later access by main.c

Main.c

Main performs the initialization routine to begin collecting samples at 1kHz.

```c
extern uint32_t LUT [];

uint32_t Convert(uint32_t * ptr){

    return LUT[*ptr] - OFFSET;
}

int main(void){

    PLL_Init(Bus80MHz); //bus to 80 MHz
  Switch_Init(); //initialize switches

  SYSCTL_RCGCGPIO_R |= 0x00000020;          // activate port F
  ADC0_InitTimer0ATriggerSeq3(0, 8000000); // ADC channel 0, 10 Hz sampling
  GPIO_PORTF_DIR_R |= 0x04;                 // make PF2 out (built-in LED)
  GPIO_PORTF_AFSEL_R &= ~0x04;              // disable alt funct on PF2
  GPIO_PORTF_DEN_R |= 0x04;                 // enable digital I/O on PF2
                                            // configure PF2 as GPIO
  GPIO_PORTF_PCTL_R = (GPIO_PORTF_PCTL_R&0xFFFFF0FF)+0x00000000;
  GPIO_PORTF_AMSEL_R = 0;                   // disable analog functionality on PF
  GPIO_PORTF_DATA_R &= ~0x04;               // turn off LED

    ST7735_InitR(INITR_REDTAB); //initialzie LCD screen
    ST7735_FillScreen(ST7735_BLACK);

    //initialize software fifo
    UART_Init();
    FiFo_Init();
    uint32_t period = 80000; //sample at 1 kHz
    ADC0_InitTimer0ATriggerSeq3PD3(period);

  EnableInterrupts();

    uint32_t data;

    //uint8_t counter = 0;
    uint32_t j = 0;
    uint8_t N = 128;

    #define BUF_SIZE 50 //10 hz cutoff frequency for digital LPF
```

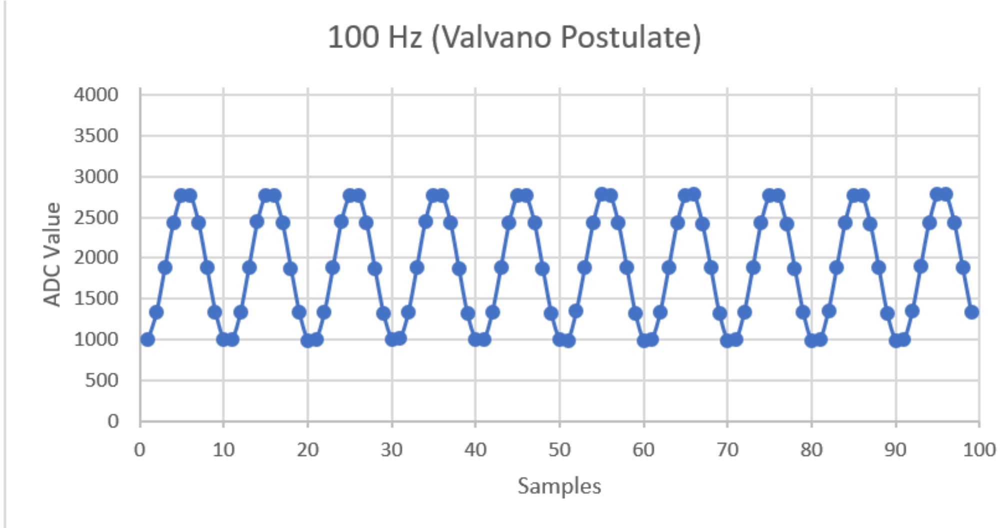
Before plotting the temperature, the values are fed through a digital low-pass filter implemented as the average of the most recent 50 values in order to cause gain drop-off after 10 Hz.

```
  uint32_t buf[BUF_SIZE] = {2500,2500,2500,2500,2500,2500,2500,2500,2500,2500,2500,2500,2500,250

  ST7735_PlotClear(1000,4000);

while(1){

    if(FiFo_Get(&data) == 1){
        //begin critical section
        DisableInterrupts();
        //perform data conversion
        uint32_t temp = Convert(&data);

        uint32_t sum = 0;
        uint32_t avg = 0;

        for(int i = 1; i < BUF_SIZE; i++){
            buf[i] = buf[i-1];
        }

        buf[0] = temp;

        for(int i = 0; i < BUF_SIZE; i++){
                sum = sum + buf[i];
        }

        avg = sum/BUF_SIZE;

        ST7735_SetCursor(0,0);
        //display results on LCD screen
        ST7735_OutString("Temp: ", ST7735_YELLOW);
        ST7735_sDecOut2(temp, ST7735_YELLOW);
        ST7735_OutString(" degC\nADC value:", ST7735_YELLOW);
        ST7735_OutUDec(data, ST7735_YELLOW);

        ST7735_PlotPoint(avg);  // Measured temperature
        ST7735_PlotNext();
        if((j&(N-1))==0){          // fs sampling, fs/N samples plotted per second
            ST7735_PlotClear(2000,4000);  // overwrites N points on same line
        }
        j++;                       // counts the number of samples

        EnableInterrupts();
```

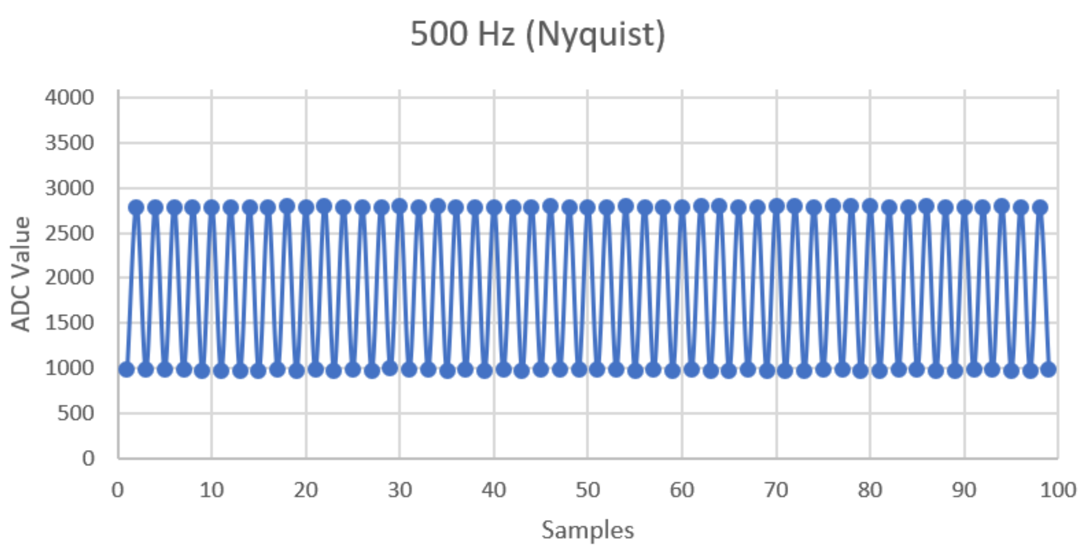
D) MEASUREMENT DATA

Waveform Analysis

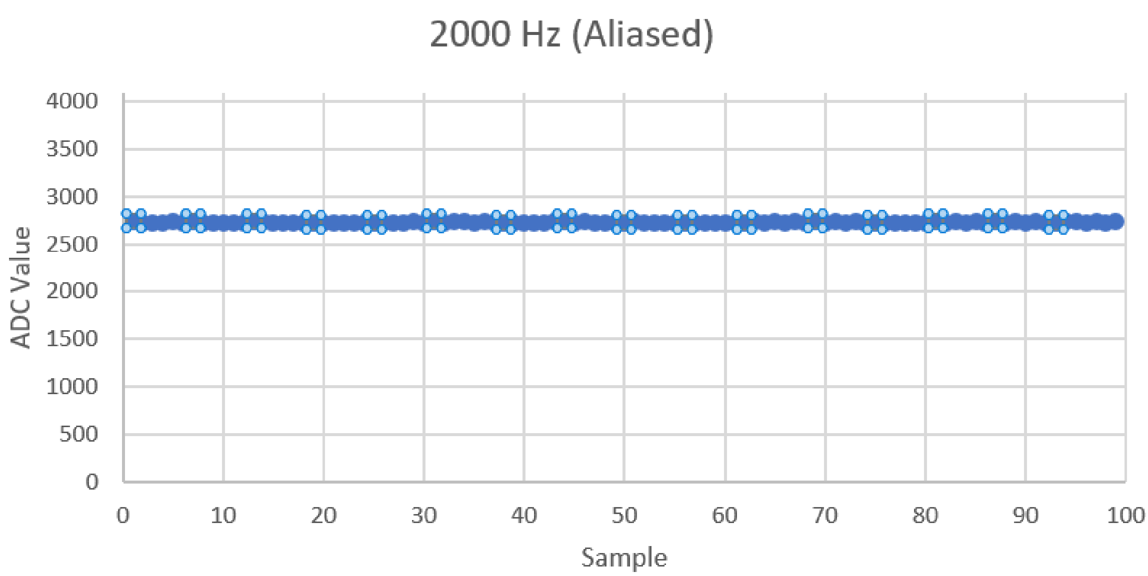
All waveforms below were sampled at 1kHz



A 100 Hz frequency can be accurately reproduced when sampled at a rate 10 times faster.



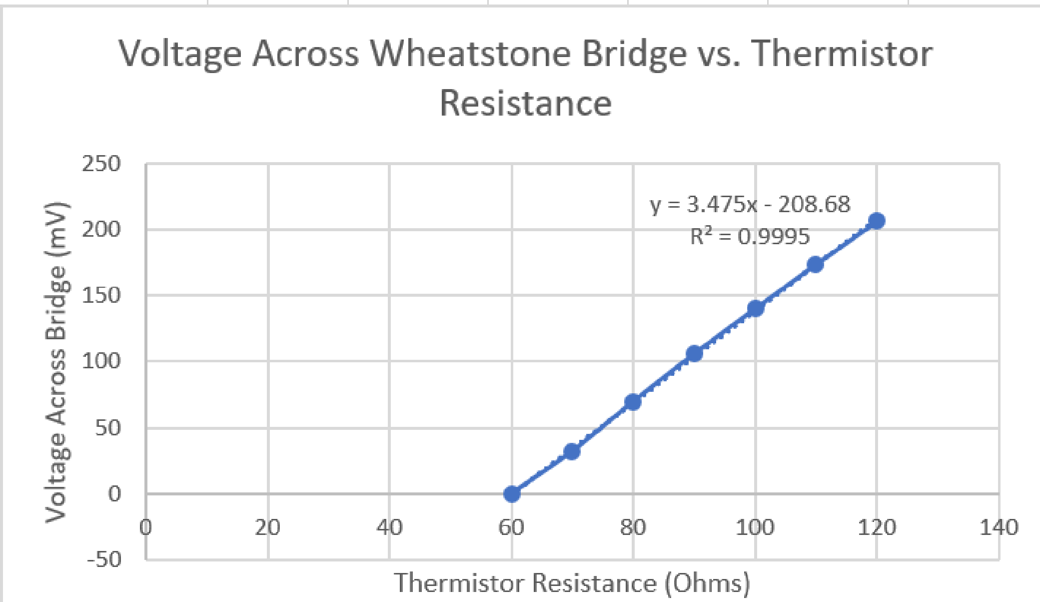
A 500 Hz frequency cannot be accurately reconstructed when it is at exactly half of the sampling frequency. The signal was slightly out of phase in comparison to the sampling signal and as a result, the ADC picked up the peaks and the troughs of the signal, and in this case preserved the frequency of the signal. However, it would be impossible to reconstruct the original sinusoid without additional information. This violates the Nyquist Theorem.



This completely violates the Nyquist Theorem, as the frequency of the input signal is twice the sampling frequency. The input signal shows up aliased as a DC offset, therefore it cannot be reconstructed under any conditions.

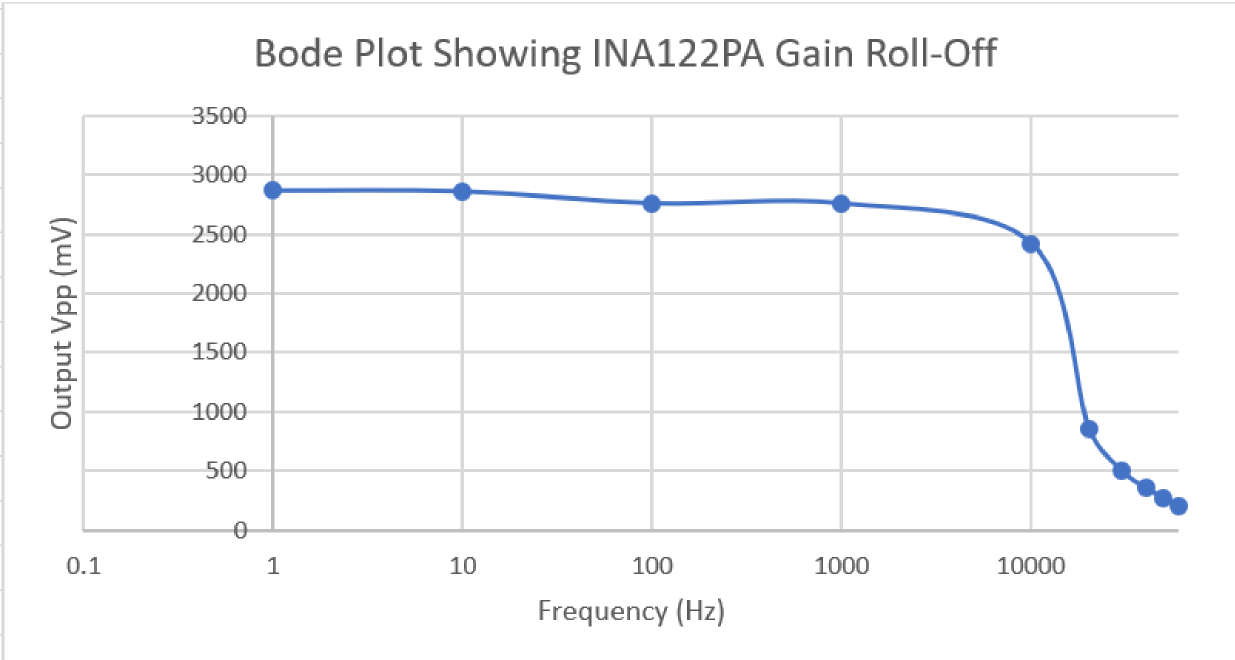
Static Circuit Performance

Resistance (Ohms)	120	110	100	90	80	70	60
Voltage (mV)	206.4	173.8	140.3	106	69.9	32.1	0



As evidenced by the voltage vs. thermistor curve above, the relationship between the two quantities is relatively linear, which justifies the mapping we achieved earlier from $R_{thermistor}$ values to those of V_{bridge}.

Dynamic Circuit Performance



From the plot above, we can conclude that our instrumentation amplifier will stifle any signals past 10 kHz. This gain roll-off is not relevant to our immediate application, as our signal of interest (temperature) is expected to be 10 Hz or less. However, in high-frequency applications like audio, it is important to choose instrumentation that will not inherently interfere with the magnitude or phase of our signal of interest.

Accuracy

Trials	Meter	Raw Measurement	Scaled Temperature (degC)
1	33.01	3300	33
2	33	3303	33.03
3	33.03	3300	33
4	32.98	3299	32.99
5	33	3293	32.93
			0.01
			0.03
			0.03
			0.01
		sum xti - xmi	0.08
		avg accuracy (degC)	0.016
		avg acc. %	1.6%

Once again using the thermometer used for calibration, I managed to obtain an average accuracy of $0.016^{o}C$.

Reproducibility

Trials	Raw Measurement	Temperature
1	2235	22.35
2	2230	22.3
3	2230	22.3
4	2232	22.32
5	2233	22.33
6	2231	22.31
7	2232	22.32
8	2234	22.34
9	2232	22.32
10	2234	22.34
	std dev.	0.017029386

Taking a set of measurements under the same conditions, the standard deviation of the distribution of values is $0.02^{o}C$. This is well within the $0.1^{o}C$ benchmark specified as a requirement of the lab.

E) ANALYSIS AND DISCUSSION

1) What is the Nyquist theorem and how does it apply to this lab?

The Nyquist Theorem states that for a signal to be accurately represented in digital form, the sampling rate must be strictly greater than twice the maximum frequency component of the signal of interest. This applies to our lab in that our signal of interest has an expected frequency of 10 Hz or less. Therefore, for our system to accurately represent the temperature in digital form, it must sample the analog signal given by the thermistor in the bridge circuit at a rate greater than 20 Hz.

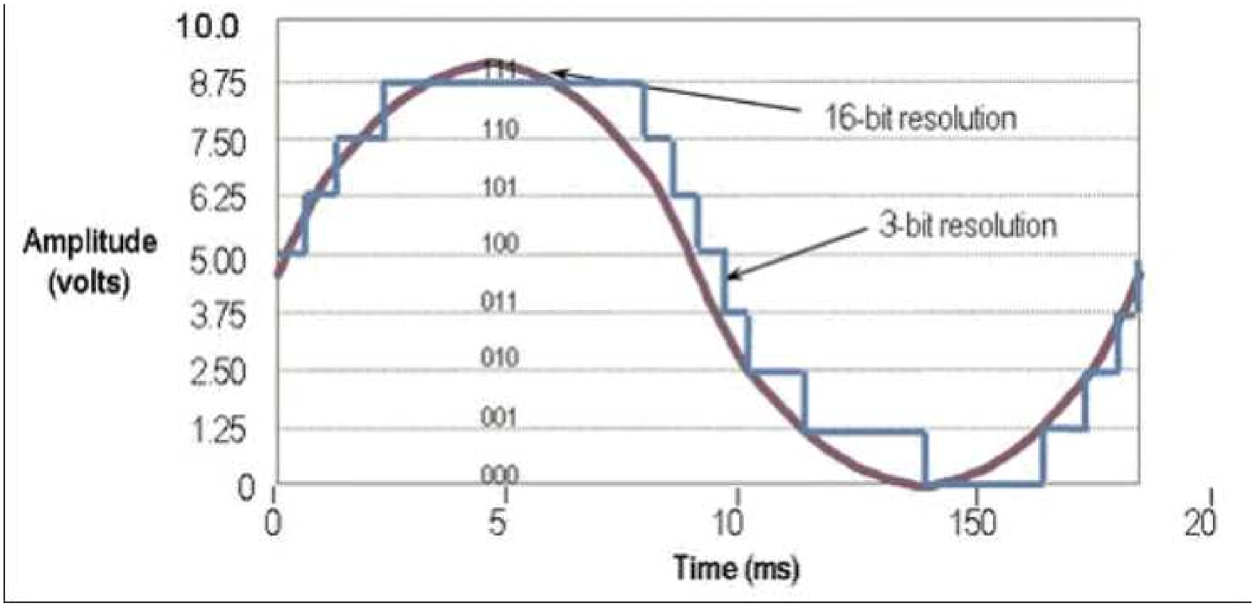
2) Explain the difference between resolution and accuracy?

Resolution refers to the smallest step change in our signal of interest that our data acquisition system can detect. Accuracy refers to the difference in actual value to the measured value. It is a measure of our device's correctness.

3) Derive an equation to relate reproducibility and precision of the thermometer.

The relationship between reproducibility and precision can be modeled in terms of the signal-to-quantization-noise-ratio approximation for analog-to-digital converters.

$SQNR = 20\log_{10} 2^{Q}$, where Q is the number of bits used in the ADC for quantization. As the number of bits used by the ADC increases, the sampled waveform more closely resembles the original continuous time waveform because it uses more levels with a smaller step size. This can be seen comparing the difference between a 3-bit ADC and a 16-bit ADC as shown below:



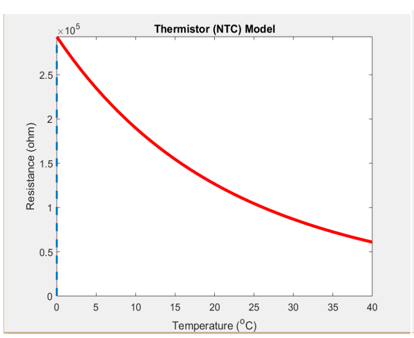
http://www.ni.com/white-paper/5356/en/

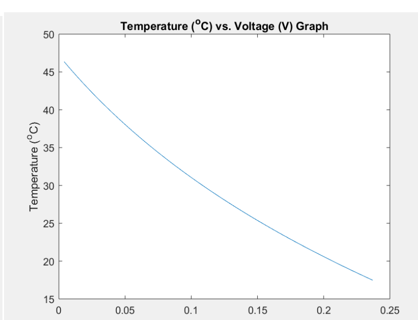
Notice that the 3-bit resolution only uses $2^3 = 8$ levels to represent the sinusoid, which causes the sampled signal to have very sharp peaks that increase the quantization noise. This, in turn, causes the SQNR of the 3-bit measurement to be much smaller than that of the 16-bit measurement, which does not introduce as much quantization noise. It can be noted that each bit added to the ADC will increase the SQNR of the signal by about 6.02 dB. As a result of this, we are better able to reproduce the signal acquired from the thermistor by adding more bits to our analog-to-digital converter. In practice, we would purchase an ADC with more than the 4096 levels that we already have available to us.

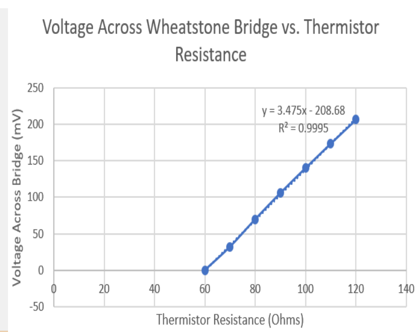
4) What is the purpose of the LPF?

The purpose of the LPF is to increase the signal-to-noise ration of our sampled data by preventing the propagation of high frequency noise when sampling the signal.

5) If the R versus T curve of the thermistor is so nonlinear, why does the voltage versus temperature curve look so linear?







Performing a side-by-side comparison, we can see that the temperature vs. voltage curve, when calculated directly from the NTC thermistor model, is in fact also non-linear. This makes sense because there is a linear relationship between the voltage difference across the Wheatstone bridge and the thermistor's resistance.

*6) There are four methods (a,b,c,d) listed in the **4) Software Conversion** section of methods and constraints. For one of the methods you did not implement, give reasons why your method is better, and give reasons why this alternative method would have been better.*

In comparison to the method of small look-up table with linear interpolation, my method is better in that it was developed directly from the transfer functions of the thermistor and the Wheatstone bridge, thus the large LUT is more accurate for every measurement that a small look-up table. The disadvantage of my method is that it takes 8,192 bytes of memory to store the table (4096 entries of 2 bytes each), and a small LUT of 50 entries would only take 100 bytes of memory. In an application where the processor is pressed for memory, a trade-off would have to be made between accuracy and space complexity of the conversion algorithm.