lab2_thermistor

September 14, 2019

1 Lab 2: Thermistor

The purpose of this lab is to measure the temperature of two ”coffee” (boiling water) cups over time as ”milk” (ice water) is added to it.

[1]:
```
import pandas as pd
import numpy as np
%matplotlib inline
```

1.0.1 Raw Data

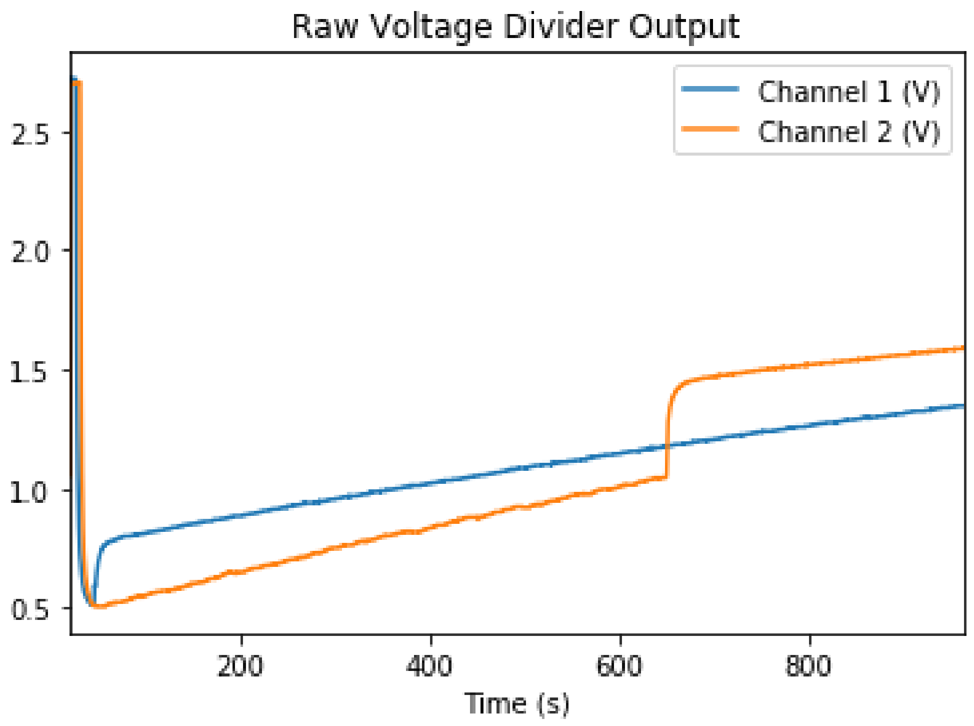
The data was collected by putting a thermistor in a voltage divider circuit. The output voltage was collected and read by using the oscilloscope function of the Analog Discovery.

After the lab was completed, the total volume of water in each cup was measured. This includes the 40 mL of water that was added during the lab.

[2]:
```
# Read Raw Data

raw_data_filename = 'thermistor_data.csv'
data = pd.read_csv(raw_data_filename)

# Configuration Data
v_in = 5   # V
r_1 = 1000   # Ohms

# Volume data
# Total volume was measured at the end of the lab, after one
# "unit" of ice water was added to each cup.
c1_total_volume = 188   # mL
c2_total_volume = 191   # mL
ice_volume = 40   # mL
```

[3]:
```
# Plot voltage vs time
data.plot(x='Time (s)', y=['Channel 1 (V)', 'Channel 2 (V)'], title='Raw␣
 ↪Voltage Divider Output')
```

```
[3]: <matplotlib.axes._subplots.AxesSubplot at 0x7fb61ce9e198>
```



1.0.2 Calculate thermistor resistance

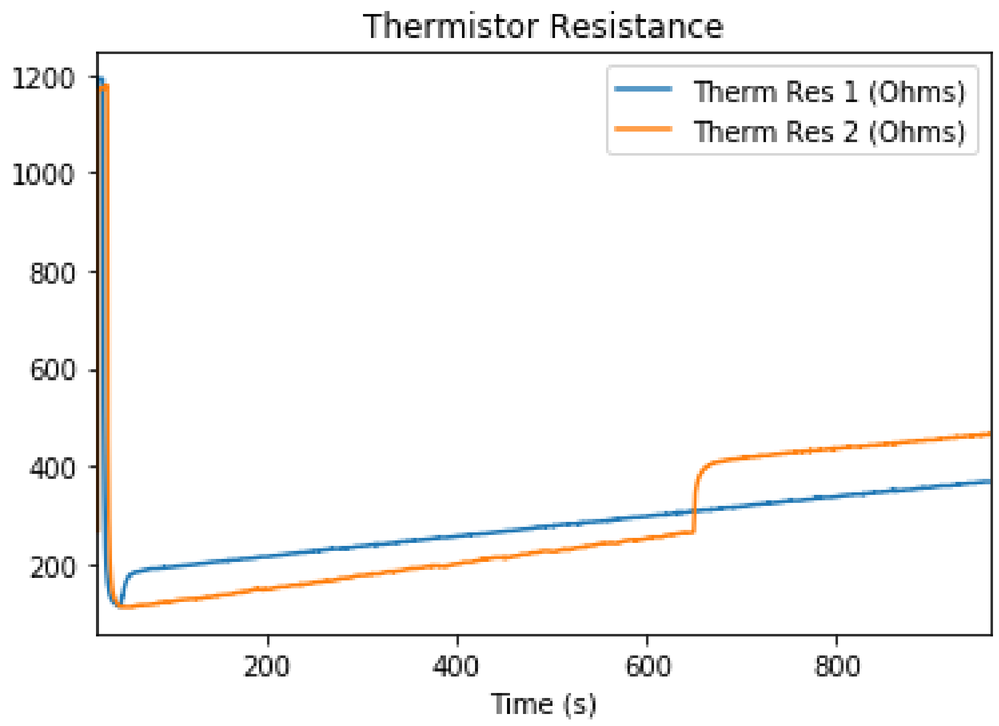
The thermistor resistance can be defined as a function of R_1 and V_{out} by using the voltage divider equation.

$$V_{out} = V_{in} \times \frac{R_{therm}}{R_1 + R_{therm}} R_{therm} = \frac{V_{out} \times R_1}{V_{in} - V_{out}}$$

```
[4]: r_therm_1 = data['Channel 1 (V)'] * r_1 / (v_in - data['Channel 1 (V)'])
     r_therm_2 = data['Channel 2 (V)'] * r_1 / (v_in - data['Channel 2 (V)'])

     data['Therm Res 1 (Ohms)'] = r_therm_1
     data['Therm Res 2 (Ohms)'] = r_therm_2
```

```
[5]: # Plot resistance vs time
     data.plot(x='Time (s)', y=['Therm Res 1 (Ohms)', 'Therm Res 2 (Ohms)'],␣
      ↪title='Thermistor Resistance')
```

```
[5]: <matplotlib.axes._subplots.AxesSubplot at 0x7fb61ada9668>
```



1.0.3 Calculate temperature

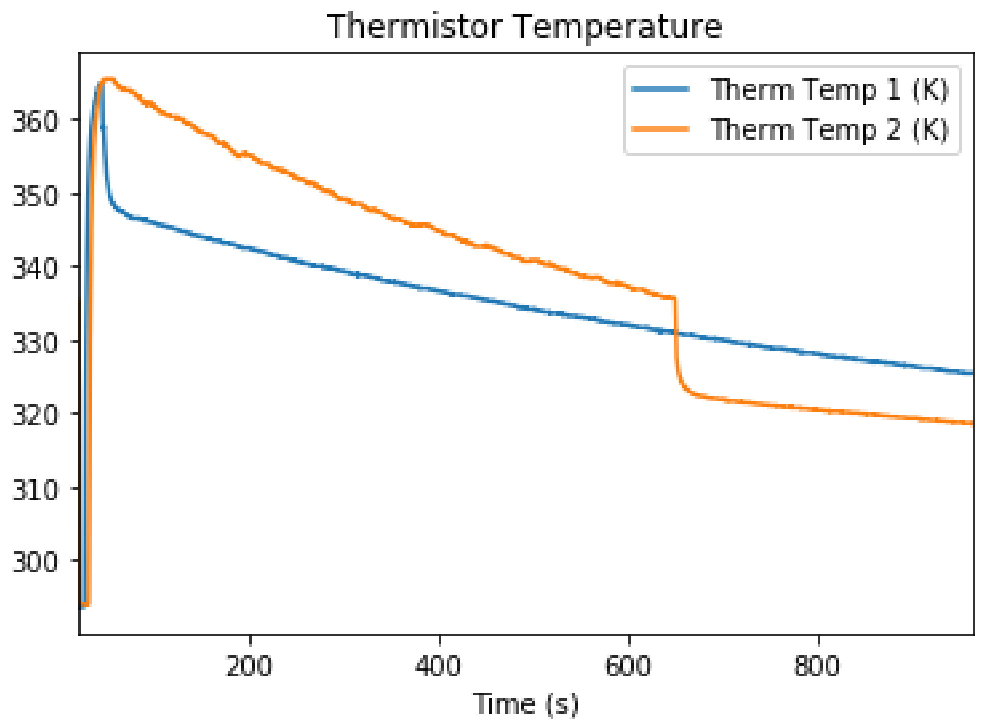
The thermistor has a defined resistance as:

$$R = 1000\Omega \times e^{-3528(\frac{1}{298}-\frac{1}{T})} T = \frac{1}{\frac{1}{298} + \frac{ln(\frac{R}{1000})}{3528}}$$

```
[6]: temp_therm_1_k = 1/(1/298 + (np.log(data['Therm Res 1 (Ohms)'] / 1000)) / 3528)
     temp_therm_2_k = 1/(1/298 + (np.log(data['Therm Res 2 (Ohms)'] / 1000)) / 3528)

     data['Therm Temp 1 (K)'] = temp_therm_1_k
     data['Therm Temp 2 (K)'] = temp_therm_2_k
```

```
[7]: # Plot temperature vs time
     data.plot(x='Time (s)', y=['Therm Temp 1 (K)', 'Therm Temp 2 (K)'],␣
      ↪title='Thermistor Temperature')
```

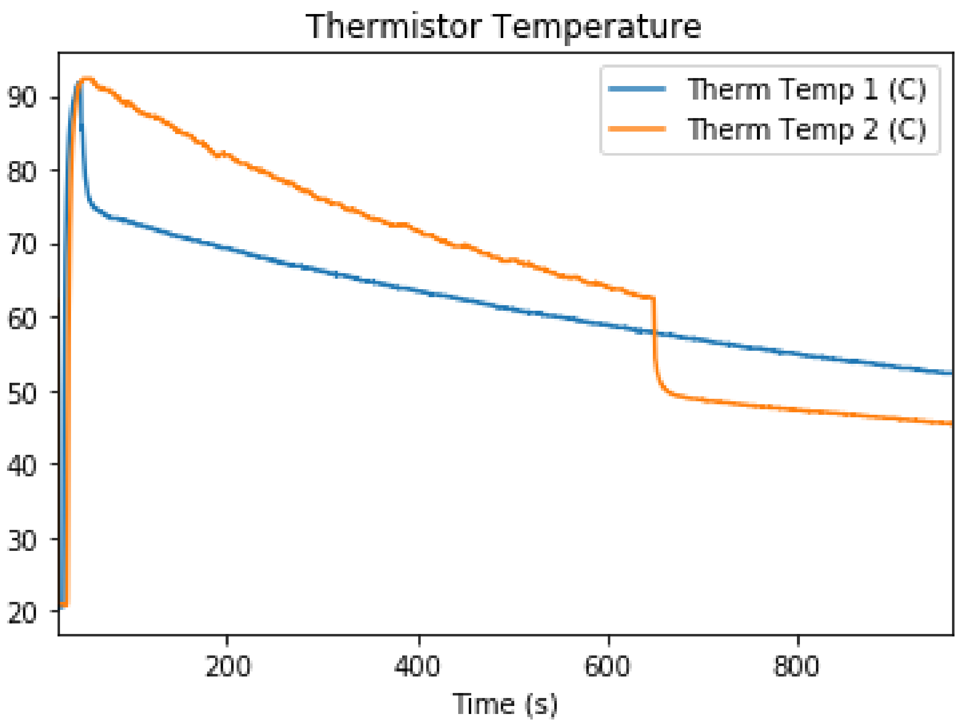
```
[7]: <matplotlib.axes._subplots.AxesSubplot at 0x7fb61a672908>
```



```
[8]: # Convert to celcius
     temp_therm_1_c = temp_therm_1_k - 273.15
     temp_therm_2_c = temp_therm_2_k - 273.15

     data['Therm Temp 1 (C)'] = temp_therm_1_c
     data['Therm Temp 2 (C)'] = temp_therm_2_c
```

```
[9]: # Plot temperature vs time
     data.plot(x='Time (s)', y=['Therm Temp 1 (C)', 'Therm Temp 2 (C)'],␣
      ↪title='Thermistor Temperature')
```

```
[9]: <matplotlib.axes._subplots.AxesSubplot at 0x7fb61ad777f0>
```



1.0.4 Characterizing heat loss over time

Because heat loss is a linear time invariant system, it can be modeled as an exponential decay function, which approaches room temperature, which we are assuming to be 22C.

$$T(t) = T_A + (T_0 - T_A)e^{\frac{-t}{\tau}}$$

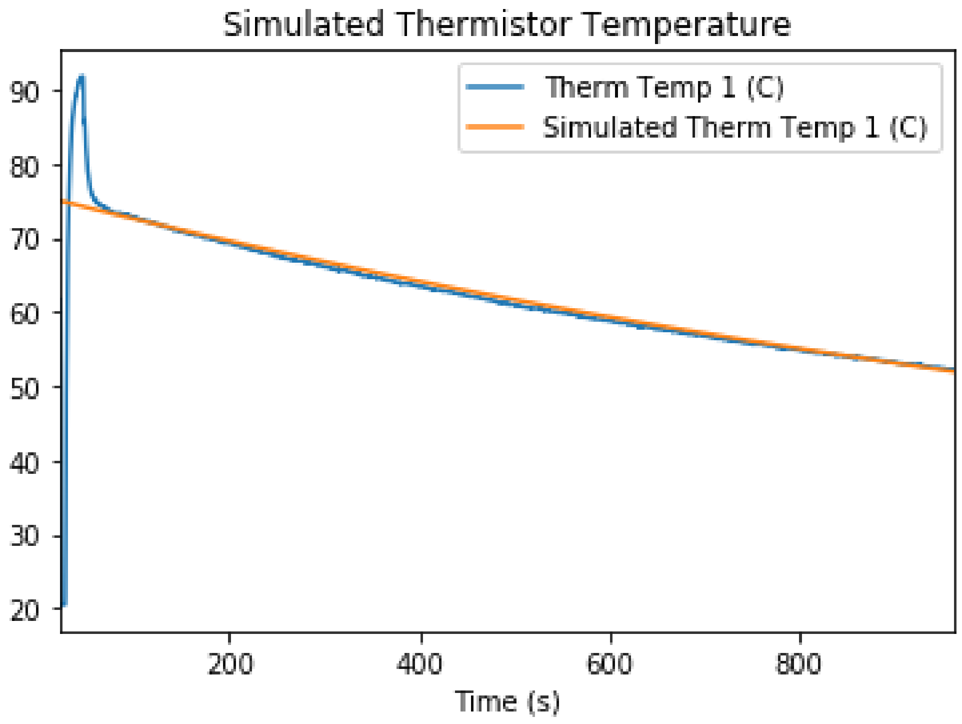
Where τ is the time constant. τ needs to be tuned to adjust the data gathered, but then can be used to extrapolate the heat loss over more time.

[10]:
```
def gen_heat_loss(times, tau, t_0=75, t_a=22):
    output = pd.Series(np.zeros(len(times)), times)
    for i in times:
        output[i] = t_a + (t_0-t_a) * np.exp(-(i - times[0]) / tau)

    return output
```

[11]:
```
tau = 1650
heat_loss = gen_heat_loss(data['Time (s)'], tau)
data['Simulated Therm Temp 1 (C)'] = heat_loss.values

# Plot simulated therm 1
```

```
data.plot(x='Time (s)', y=['Therm Temp 1 (C)', 'Simulated Therm Temp 1 (C)'],␣
↪title='Simulated Thermistor Temperature')
```

```
[11]: <matplotlib.axes._subplots.AxesSubplot at 0x7fb61a672208>
```



1.0.5 Conclusions

This lab shows significant evidence that the best way to optimize the heat of your coffee is to dump the milk in as soon as possible. This agrees with Newton's Law of Cooling, which implies that the heat loss is higher when there is a greater temperature difference with the environment.

In addition, the heat loss system was modeled using an exponential decay function, with a time constant of $1650\frac{1}{C}$.