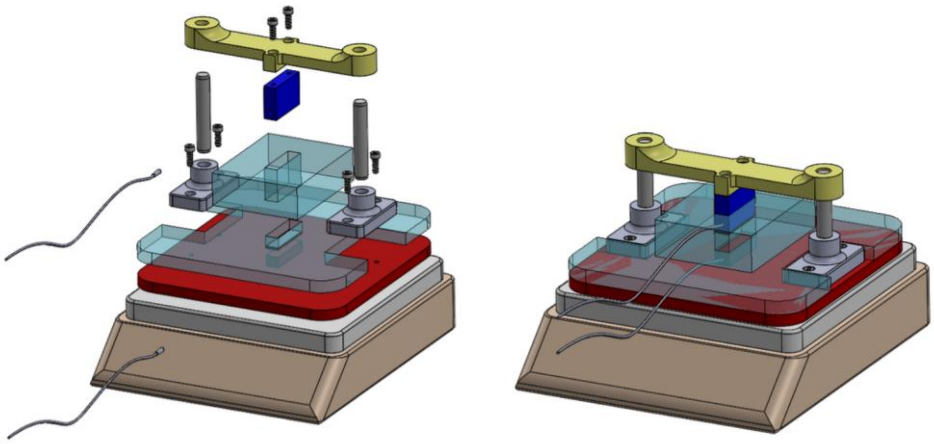
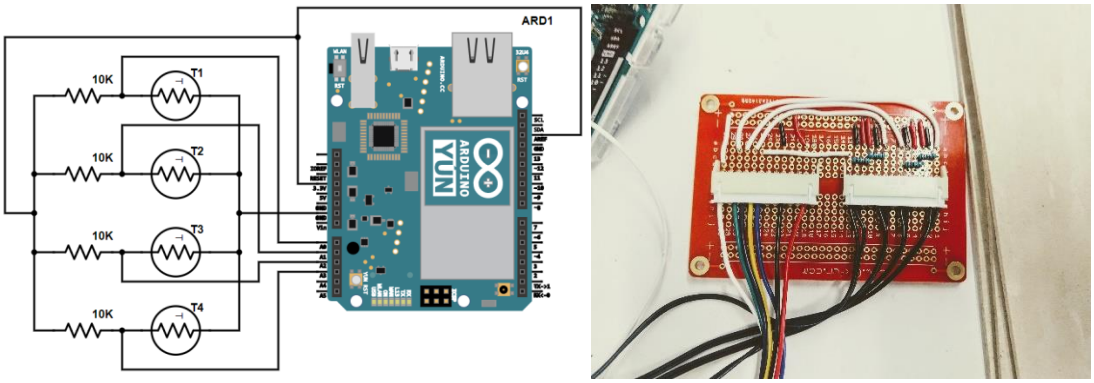
Thermal Conductivity Testing

A large part of EV battery pack/module design is cooling. If the battery cells are not kept at the right temperature, they will have a reduced lifetime and higher risk of failing. In addition to this, the more heat the cooling system can pull from the cells, the more current can be drawn from them safely. The cooling tube however, needs to be electrically isolated from the cells either using a non conductive tube material or a non conductive coating. I was tasked with designing a thermal conductivity test rig to test the relative thermal conductivities of different cooling tube/coating designs. This was not meant to be an absolute test, but rather a simple way to benchmark the current design against new designs.

A hot plate set to a fixed temperature Is used as the heat source, polyethylene foam for insulation, and thermistors for temperature measurment.



I wanted to monitor not only the input and output temperatures, but also the temperature of the environment. Using an Arduino for data acquisition, I was able to easily read temperatures from 4 thermistors to the serial port on my laptop and import the data into python for more analysis. In python I save the raw data as a .csv file, then plot the temperatures against time (and save the plot), then finally save a .txt with the time constant, and a few other important details about the data.



Sensor wiring diagram and soldered prototype board: The crimped connectors on the left connect to the Arduino, and the ones on the right are the 4 thermistors.

Arduino code

```
static const uint8_t Pins[] = {A0, A1, A2, A3}; // resistance at 25 degrees C
#define THERMISTORNOMINAL 10000
// temp. for nominal resistance (almost always 25 C)
#define TEMPERATURENOMINAL 25
// how many samples to take and average, more takes longer
// but is more 'smooth'
#define NUMSAMPLES 5
#define NUMTHERMISTORS 4
// The beta coefficient of the thermistor (usually 3000-4000)
#define BCOEFFICIENT 3950
// the value of the 'other' resistor
#define SERIESRESISTOR 10000
int startTime = millis();
// time step in milliseconds
//******************************** change time step here
int timeStep = 100;
int samples[NUMTHERMISTORS][NUMSAMPLES];

void setup(void) {
  Serial.begin(9600);
  analogReference(EXTERNAL);
}
void loop(void) {
  uint8_t i;
  uint8_t j;
  // take N samples in a row, with a slight delay
  int t = millis() - startTime;
  for (i = 0; i < NUMSAMPLES; i++) {
    for (j = 0; j < NUMTHERMISTORS; j++) {
      samples[j][i] = analogRead(Pins[j]);
    }
    delay(5);
  }

  // average all the samples out
  float  average[NUMTHERMISTORS] = {0};
  for (i = 0; i < NUMTHERMISTORS; i++) {
    for (j = 0; j < NUMSAMPLES; j++) {
      average[i] += samples[i][j];
    }
  }

  for (i = 0; i < NUMTHERMISTORS; i++) {
    average[i] /= NUMSAMPLES;
  }

  // convert the value to resistance
  for (i = 0; i < NUMTHERMISTORS; i++) {
    average[i] = 1023 / average[i] - 1;
    average[i] = SERIESRESISTOR / average[i];
  }
  float steinhart[NUMTHERMISTORS];

  for (i = 0; i < NUMTHERMISTORS; i++) {
    steinhart[i] = average[i] / THERMISTORNOMINAL;     // (R/Ro)
    steinhart[i] = log(steinhart[i]);                  // ln(R/Ro)
    steinhart[i] /= BCOEFFICIENT;                   // 1/B * ln(R/Ro)
    steinhart[i] += 1.0 / (TEMPERATURENOMINAL + 273.15); // + (1/To)
    steinhart[i] = 1.0 / steinhart[i];                 // Invert
    steinhart[i] -= 273.15;                         // convert to C
  }

  for (i = 0; i < NUMTHERMISTORS; i++) {
    Serial.println(steinhart[i]);
  }
  Serial.println(t);

  delay(timeStep);  //set timestep in miliseconds
}
```

Python code

```
#Clear variables
from IPython import get_ipython
get_ipython().magic('reset -sf')

import serial
import time
import matplotlib.pyplot as plt
import numpy as np
import pandas as pd

plt.cla()   # Clear axis
plt.clf()   # Clear figures
plt.close() # Close a figure window


# ***************Set total run time
here*******************************************************
totalTime = 10     # Run time
#
*****************************************************************************************************
**

# ***************Change export filename
here*****************************************************
fileName = "out"   # export file name
#
*****************************************************************************************************

stamp = 0
i = 0
j = 0
temp1 = []; #input temperature
temp2 = []; #air temp 1
temp3 = []; #air temp 2
temp4 = []; #output temperature

#arduinoSerialData.read(arduinoSerialData.inWaiting())
times = []
startSeconds = time.time()
ardTimes = []

arduinoSerialData = serial.Serial('com4',9600) #Create Serial port object called
arduinoSerialData(com port may have to be changed)

arduinoTimeStamp = 0.0
type(arduinoTimeStamp)
while (arduinoTimeStamp<totalTime):
    if (arduinoSerialData.inWaiting()>0):
         t = arduinoSerialData.readline()
         string_n = t.decode()  # decode byte string into Unicode
         string = string_n.rstrip() # remove \n and \r
         t1 = float (string)
         time.sleep(.0001)

         t = arduinoSerialData.readline()
         string_n = t.decode()  # decode byte string into Unicode
         string = string_n.rstrip() # remove \n and \r
         t2 = float (string)
         time.sleep(.0001)

         t = arduinoSerialData.readline()
         string_n = t.decode()  # decode byte string into Unicode
         string = string_n.rstrip() # remove \n and \r
         t3 = float (string)
         time.sleep(.0001)

         t = arduinoSerialData.readline()
         string_n = t.decode()  # decode byte string into Unicode
         string = string_n.rstrip() # remove \n and \r
```

```
            t4 = float (string)
            time.sleep(.0001)

            temp1.append(t1) #Refrence temperature
            temp2.append(t2)
            temp3.append(t3)
            temp4.append(t4) #Output temperature

            arduinoTimeStamp = float(arduinoSerialData.readline())/1000

            print ([temp1[i],temp2[i],temp3[i],temp4[i]])

            ardTimes.append(arduinoTimeStamp)
            i +=1

plt.plot(ardTimes, temp1, label='temp1')
plt.plot(ardTimes, temp2, label='temp2')
plt.plot(ardTimes, temp3, label='temp3')
plt.plot(ardTimes, temp4, label='temp4')

# Add a legend
plt.legend(loc='center left', bbox_to_anchor=(1, 0.5))
plt.title(fileName + 'plot')
plt.xlabel('Time [S]')
plt.ylabel('Temperature [C]')

# Show the plot
plt.show()
plt.savefig(fileName + '.png', dpi=300, bbox_inches='tight')    #save plot as image

temps = [ardTimes,temp1, temp2, temp3, temp4] #create single list of lists containina all data

with open(fileName + '.csv', "w") as o:   #save data as CSV
    for el in temps:
        for item in el:
            o.write (str(item)+',')
        o.write ('\n')


dat = pd.DataFrame(temps)  #create pandas dataframe containing all data

startOutTemp = dat[0][4]+.5  # to find when the temperature starts rising
startInTemp = dat[0][1]      # initial input (hot side) temperature

endOutTemp =  dat[len(temp1)-1][4]
endInTemp = dat[len(temp1)-1][1]

checkTemp = startOutTemp + .632*(startInTemp-startOutTemp)

print ('')
print ('the 63.2% temperature is:')
print (checkTemp)

startTime = 0

for i in range(len(temp1)):
    if dat[i][4] > startOutTemp:
        startTime = ardTimes[i]
        break

timeConstant = 0
for i in range(len(temp1)):
    if dat[i][4] > checkTemp:
        timeConstant = ardTimes[i]-startTime
        break

print (' ')
print ('the time constant is:')
print (timeConstant)

print ('')
```

```
print ('the input temperature changed from:')
print (startInTemp, ' to ', endInTemp, sep=' ')

print ('')
print ('the output temperature changed from:')
print (startOutTemp, 'to', endOutTemp, sep=' ')


with open(fileName + '.txt', "w") as o:
    o.write ('the input temperature changed from: ' + str(startInTemp)+ 'to' +str(endInTemp))
    o.write ('\n')
    o.write ('the output temperature changed from: ' + str(startOutTemp)+ 'to' +str(endOutTemp))
    o.write ('\n')
    o.write ('the time constant (tau) is: ' + str(timeConstant) + ' and the temperature at t=tau
is: ' + str(checkTemp))

arduinoSerialData.close()
print ('end')
```