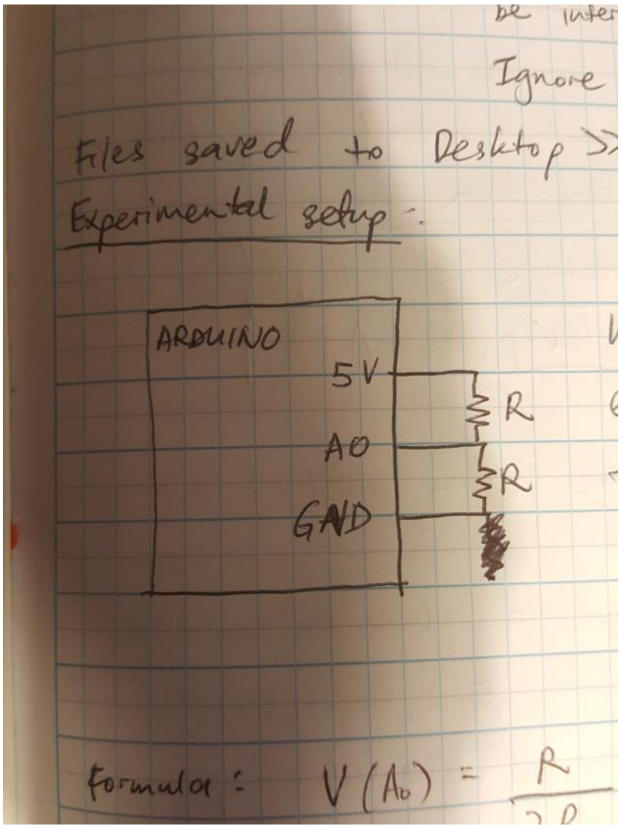
ESI

Workshop 1

Introduction

- Usage of Arduino and voltage dividers
- Download program from Arduino.cc
- Start program. Select Tools >> Board >> Select Board
- Arduino Uno was used
- Connect device to PC . Select Tools >> Serial Port >> Tick option
- File >> Example to test
- After opening file, upload file to Arduino. Use serial monitor to observe progress
- For exporting data, copy (ctrlA, ctrlC) and paste in Microsoft Excel (ctrl V). (MATLAB can be used)
- Problems encountered: Not able to copy data using ctrl A, ctrl C
- First row contains characters that cannot be interpreted by desired program. Ignore unrecognisable characters

Experimental Setup:



Wires connect the 2 resistors (of almost equal values) to a breadboard with the appropriate ports of the Arduino. The data generated by the program is giving the voltage received at port A0

Formula:

V(A0) = VR/2R = V/2 (theoretically)

Questions:

1. What is the sample rate?
2. How can we change it?
3. How much noise is there?
4. Is the noise Gaussian?
5. How does it depend on the resistor values?
6. Does the voltage source make a difference?

Procedure:

1. Built circuit as per experimental setup on previous page
2. Use resistor values of R = 200kohms and R = 100 kohms. Repeated using resistor values of R = 300 kohms, 560kohms, 30 kohms, 20 kohms. (To increase amount of data) [range not big enough] Include R = 150 ohms, 1Mohms, 1.5 Mohms, and 2.2Mohms. Took data over more than 350 samples each.
3. Code used was a template from Examples (AnalogRead). Code was uploaded and serial monitor opened every time R was changed. Measurements taken from serial monitor.
4. Used Microsoft Excel's =average(Bi:Bj) to find mean of data and
5. =stdev.p(Ai:Aj) to find standard deviation (s.d.)
6. Plotted mean (bits representing voltage across resistor) vs value of resistance (R).
7. Used bit value rather than voltage to reduce calculation errors introduced via rounding (i.e. 512, 513 etc)
8. Also, plotted standard deviation (of bit values representing voltage) against value of resistance.

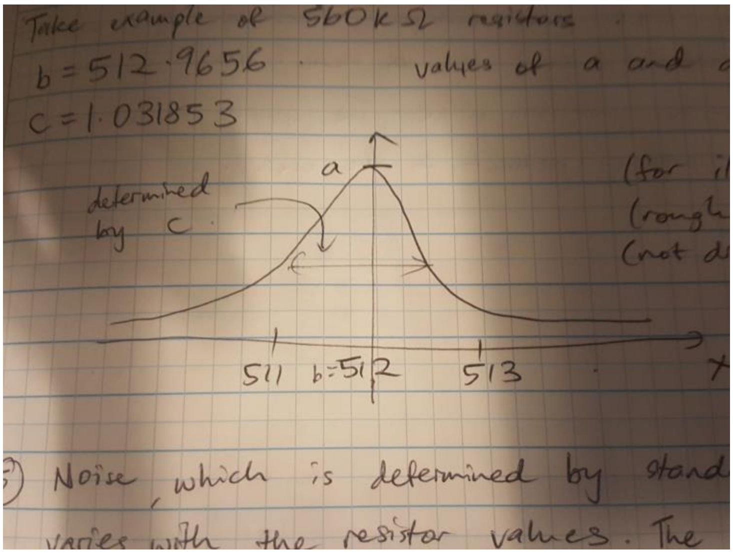
Analysis (General comments)

- When two 560kohms resistors are connected to the ports, we first recorded a mean of 512.97 with s.d. of 1.03. However, upon switching the two resistors, a mean of 509.85 with s.d. of 1.13 is recorded. The difference in mean value recorded is attributed to the difference in actual resistance of the resistors. Hence, the voltage divider does not equally "divide" the voltage (not exactly in half). We observe similar patterns when the experiment was repeated using a range of resistor values (560 kohms, 1 Mohms, 1.5 Mohms, 2.2 Mohms).

- From the plotted graph, we found that the mean voltage recorded is slightly correlated to the resistance. However, the s.d. has a strong correlation to the value of resistance. In other words, s.d. increases as the resistance value increases (e.g. some tolerance but gets bigger when resistor value is bigger)
- Generic phenomenon: When two resistors are interchanged, the mean will increase to above 512, if the value is less than 512 (i.e. 508 -> 513) and vice versa. This is consistent with the actual resistance not being exactly that of its specifications. However, the s.d. remains within the stated degree of tolerance.

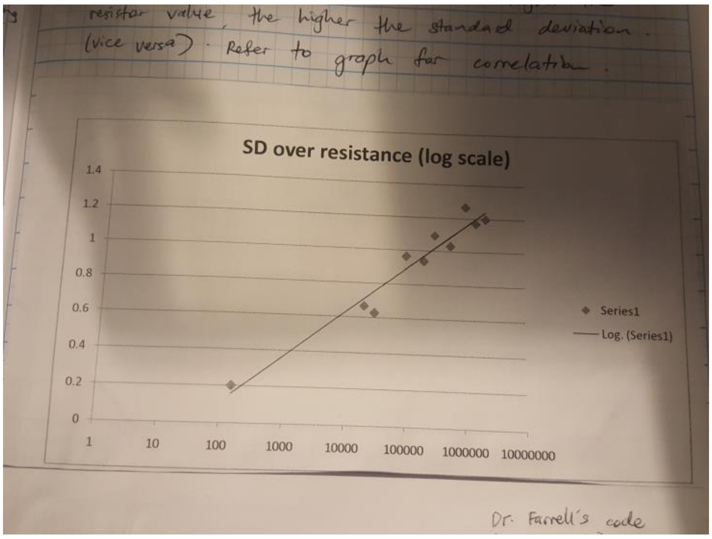
Analysis (Answers)

1. Baud rate is 9600 bps. Baud rate is the maximum bits per second on the wire.
2. Sample rate recorded = 195.83 (max)
3. Sample rate can be adjusted by manipulating delay in code (i.e. delay(1); or delay(1000);). (Sample rate recorded = 20.47 when delay = 500). The higher the delay, the lower the sample rate.
4. Sample rate also increased when “Serial.println” was removed from code.
5. Assume that noise is the standard deviation in this case. Standard deviation ranges from 0.202812% to 1.288424%. The “noise” or standard deviation falls within (or close to) the tolerance level of the resistors of 1%.
6. Gaussian function is given as:
7. F(x) = a exp(-(x-b)^2/2c^2)+d
8. Where a = height of peak
9. B = position of center
10. C = standard deviation
11. D = value that function asymptotically approaches far from peak
12. From observation, yes, the noise is Gaussian. Take example of 560 kohms resistors.
13. B = 512.9656, c =1.031853
14. Values of a and d are unknown
15. Gaussian distribution
16. Noise, which is determined by standard deviation, varies with the resistor values. The higher the resistor value, the higher the standard deviation (vice versa). Refer to graph for correlation. (GRAPH AND CODE)
17. Mean (bit values) might seem to have a correlation with resistor value BUT from analysis, remember that switching two resistors changes the mean. Mean is not correlated with resistor value. (GRAPH OF MEAN OVER RESISTANCE)
18. We were given a 5V voltage source on the Arduino . (Expected) Using a lower 3V voltage might give less sensitive bit values. But this is only a conjecture, it is inconclusive. Noise mainly depends on tolerance of resistors. So, no, voltage source does not affect results (based on data we have).



DISCUSS SIGNAL TO NOISE RATIO – REFER TO POZAR

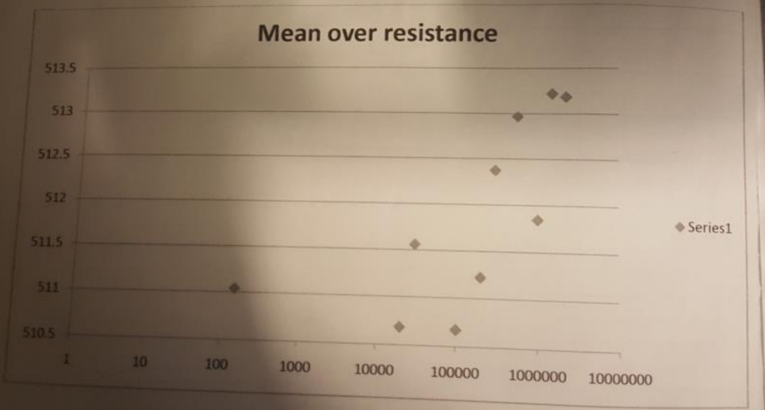
(SCREENSHOT + CODE)

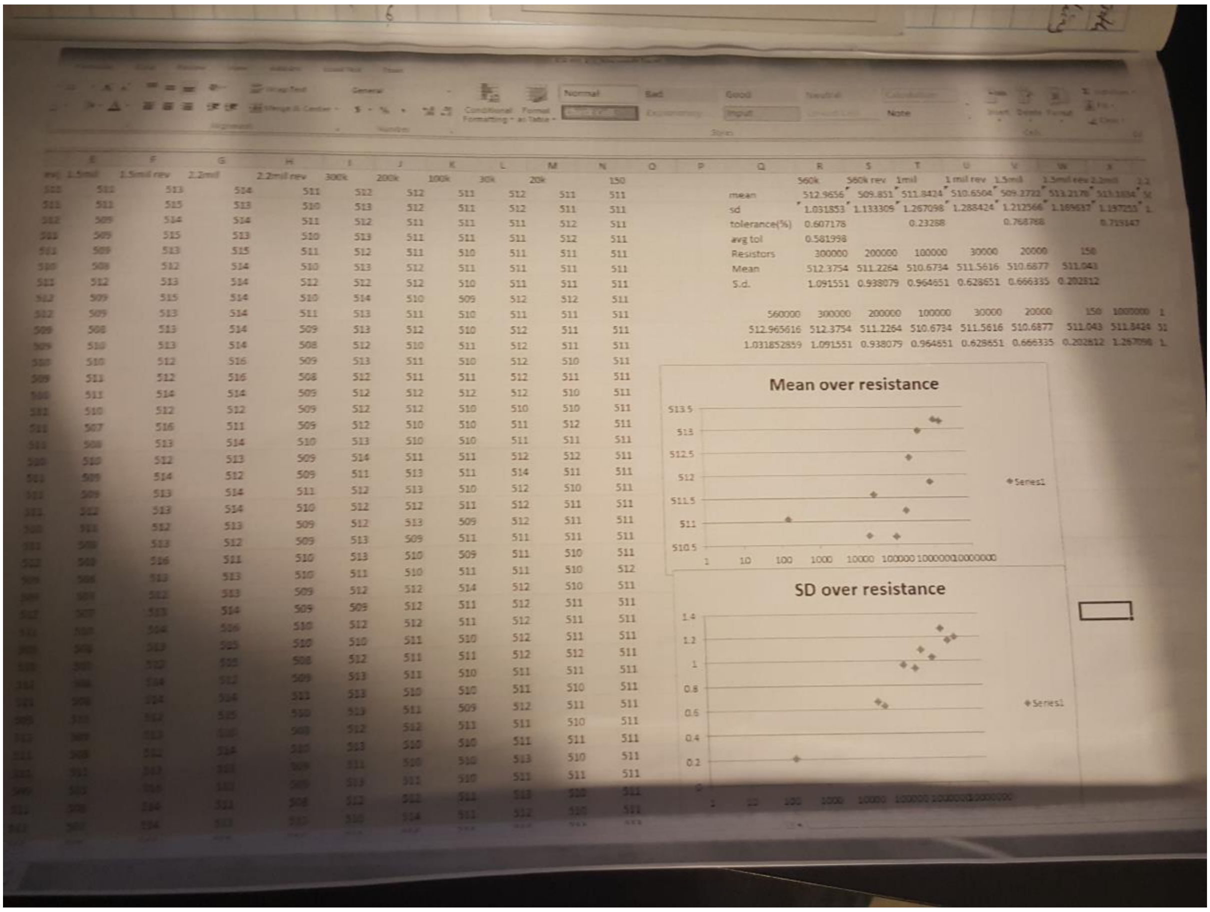


ESI WS1 code2

```
/*
  AnalogReadSerial
  Reads an analog input on pin 0, prints the result to the serial monitor.
  Attach the center pin of a potentiometer to pin A0, and the outside pins to
+5V and ground.

 This example code is in the public domain.
 */

// the setup routine runs once when you press reset:
float mn=511;
double sd=0;
void setup() {
  // initialize serial communication at 9600 bits per second:
  Serial.begin(9600);
}

// the loop routine runs over and over again forever:
void loop() {
  long int t=millis();
  for(int i=0;i<10000;i++){
  // read the input on analog pin 0:
  int sensorValue = analogRead(A0);
  // print out the value you read:
  //Serial.println(sensorValue);
  mn=(1-0.1)*mn+sensorValue*0.1;

  sd=(1-0.1)*sd+(sqrt((float(sensorValue)-mn)*(float(sensorValue)-mn))-sd)*0.1;
  delay(0.1);        // delay in between reads for stability
  }
  t=millis()-t;
  float fs=1.0e6/t;
  //sd=sqrt(sd);
  Serial.println(fs);
  Serial.println(mn);
  Serial.println(sd);
  while(1){};
}
```

correlated with resistor value. The graph below is by chance, looking pleasant to the eyes.

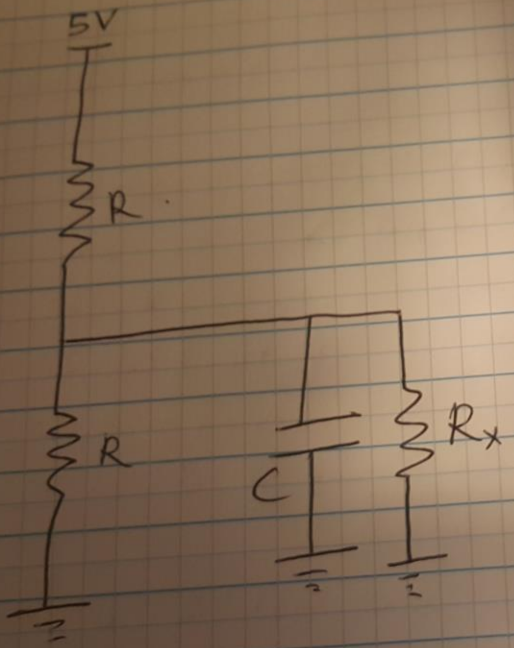




Extras

Our model for this lab is not accurate. Amore likely model is as follows:

Our mo...
model is as follows :



This is ~~how~~ to account for the unexact actual resistance of R.

(Manufacturers of R are responsible. Not my fault that resistors differ in resistance from their specifications).

In terms of voltage, we expect to see 2.5 V

ESI WS1 code

```
/*
  AnalogReadSerial
  Reads an analog input on pin 0, prints the result to the serial monitor.
  Attach the center pin of a potentiometer to pin A0, and the outside pins to
+5V and ground.

 This example code is in the public domain.
 */

// the setup routine runs once when you press reset:
void setup() {
  // initialize serial communication at 9600 bits per second:
  Serial.begin(9600);
}

// the loop routine runs over and over again forever:
void loop() {
  long int t=millis();
  for(int i=0;i<1000;i++){
  // read the input on analog pin 0:
  int sensorValue = analogRead(A0);
  // print out the value you read:
  Serial.println(sensorValue);
  delay(1);        // delay in between reads for stability
  }
  t=millis()-t;
  float fs=1.0e6/t;
  Serial.println(fs);
  while(1){};
}
```

This is to account for the inexact actual resistance of R.

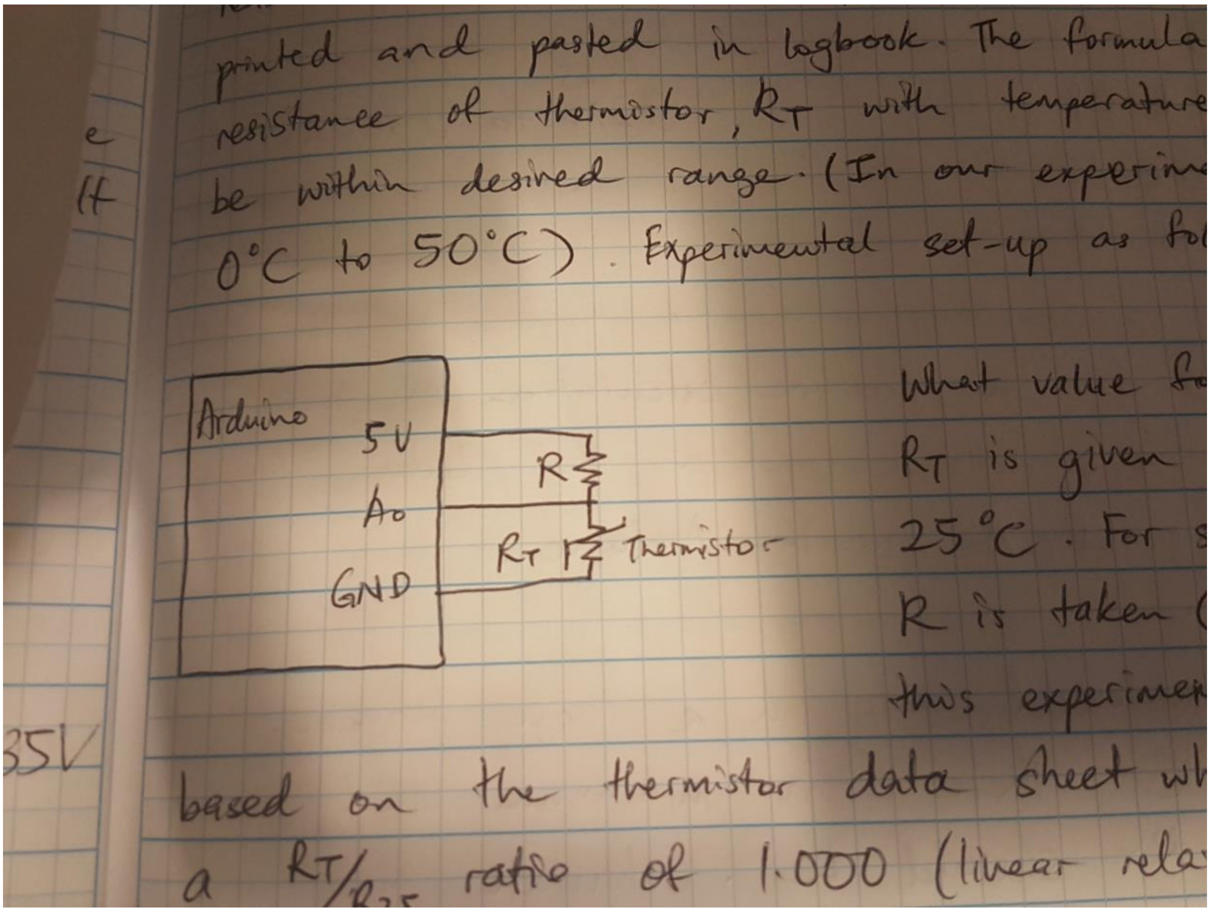
Differs from manufacturers specifications.

In terms of the voltage, we expect to see 2.5V but due to the different values of resistance, we get values ranging from 509(2.5)/512 = 2.48535 V to 515(2.5)/512 = 2.51465 V.

The voltage ranges from 2.48V to 2.52V. Arduino code is attached for reference.

Workshop 2

Aim: Design a circuit to read the ambient temperature.

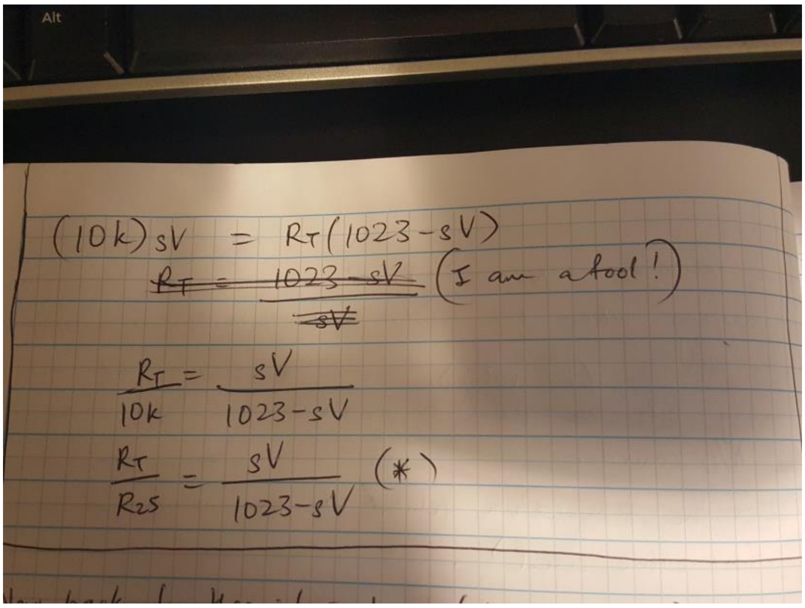


Introduction:

- Introduction to usage of thermistor (temperature dependent resistor)
- Will need to encode formulae for temperature calculation into Arduino.
- Use similar circuit to that in Workshop 1. Replace one of the resistors with thermistor. Also, thermistor data sheet is printed and pasted into logbook. The formula relates to resistance of thermistor, R_T with temperature and must be within desired range. (In our experiment, choose 0 degrees to 50 degrees). Experimental setup is as follows:
- PCITUREREEEEEEEEEEEEEEE
- What value for R? R_T is given as 10kohms at 25 degrees. For simple calculations, R is taken at 10 kohms for this experiment. This is based on the thermistor datasheet where 25 degrees gives a $RT/R25$ ratio of 1.000 (linear relation with a 1 to 1 relationship). Also, we are only interested in temperatures in the range of 5 degrees to 45 degrees.
- From thermistor datasheet, we may use two formula:
- $RT/R25 = \exp(A+B/T+C/T^2+D/T^3)$ where T = temperature in K and K = degrees +273.15
- The desired temperature range conveniently falls into 0 degrees to 50 degrees so:
- $A = -1.4141963E+01$
- $B = 4.4307830E+03$
- $C = -3.40789983E+04$
- $D = -8.8941929E+06$
- $1/T = a+b(\ln RT/R25)+C(\ln RT/R25)^2+d(\ln RT/R25)^3$
- Again use A,B,C, D for 0 to 50 degrees

- A = 3.3540154E-03
- B = 2.5627725E-04
- C = 2.0829210E-06
- D = 7.3003206E-08
- RT/R25 range: 3.274 to 0.36036
- Use formula 2 for temperature calculation in Arduino. We know the RT/R25 ratio based on the data obtained from the reading of A0 in the Arduino.

Since we are interested only in a range between 5 degrees and 45 degrees, there is a minimum and maximum that RT should be. Calculate once thermistor type (NTC or PTC) is determined.

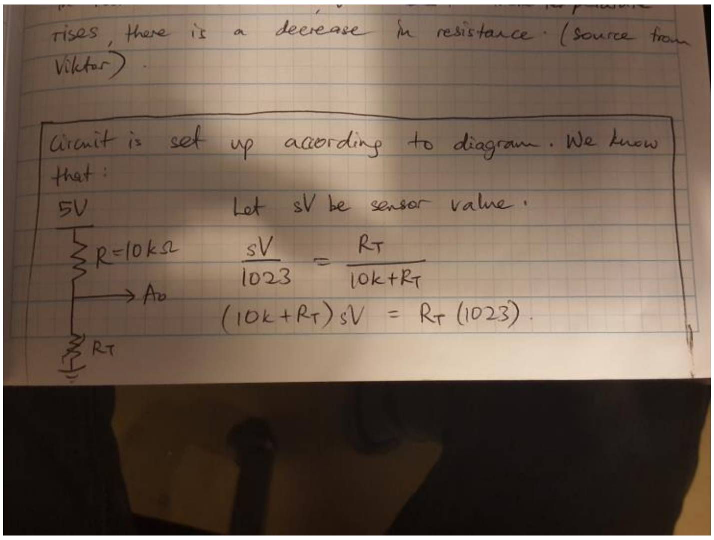


From Q1, the thermistor is NTC.

R5 = RT at 5 degrees = 735/(1023-735)(10K) = 25520.83 ohms

R45 = RT at 45 degrees = 312/(1023-312)(10K) = 4388.19 ohms

Where sV = 735 at 5 degrees and sV = 312 at 45 degrees


rises, there is a decrease in resistance. (Source from Viktor).

Circuit is set up according to diagram. We know that:

5V

$R = 10k\Omega$

A_0

R_T

Let sV be sensor value.

$$\frac{sV}{1023} = \frac{R_T}{10k + R_T}$$

$$(10k + R_T)\, sV = R_T\,(1023).$$

Questions:

1. Is the thermistor NTC or PTC?
2. What happens when the resistor and thermistor are switched (interchanged)?
3. How many bits (range of bit values) are used to represent the temperature in the A/D converter?
4. How does the temperature relate to the number of bits used? (bit value used)
5. How does the sensitivity relate to the temperature range?
6. How accurate/precise are the temperature measurements?

Procedure will be a part of the answers to the questions

Analysis (Answers):

1. There are two types of thermistors: Positive Temperature Coefficient (PC) and NTC. For PTC, when temperature rises, there is also an increase in resistance. For NTC, when the temperature rises, there is a decrease in resistance.

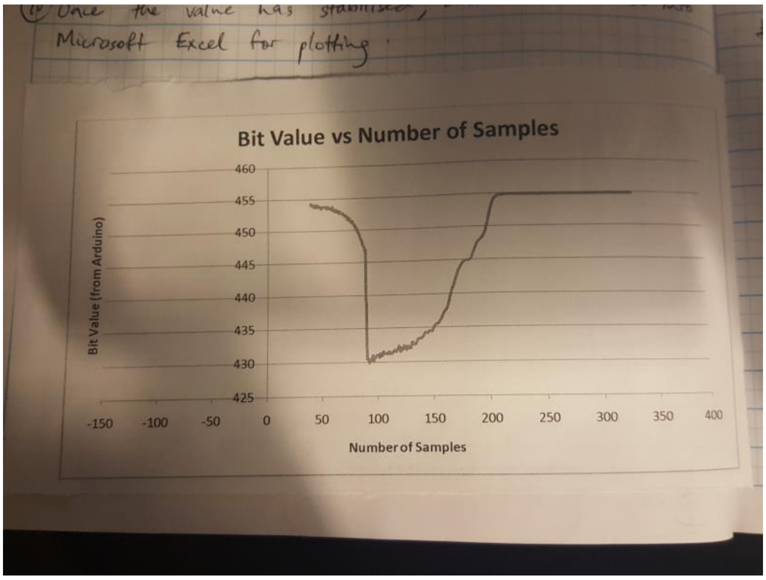
Circuit is set up according to diagram. We know that:

DIAGRAMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMM

Now back to thermistor type (NTC or PTC)

1. Circuit is set up as in diagram. RT is attached between A0 and GND. AnalogRead code is uploaded to Arduino.
2. Serial Monitor is brought up.
3. The thermistor is touched with bare fingers for a few seconds and then released
4. Once the value has stabilised, data is inserted into Microsoft Excel for plotting.

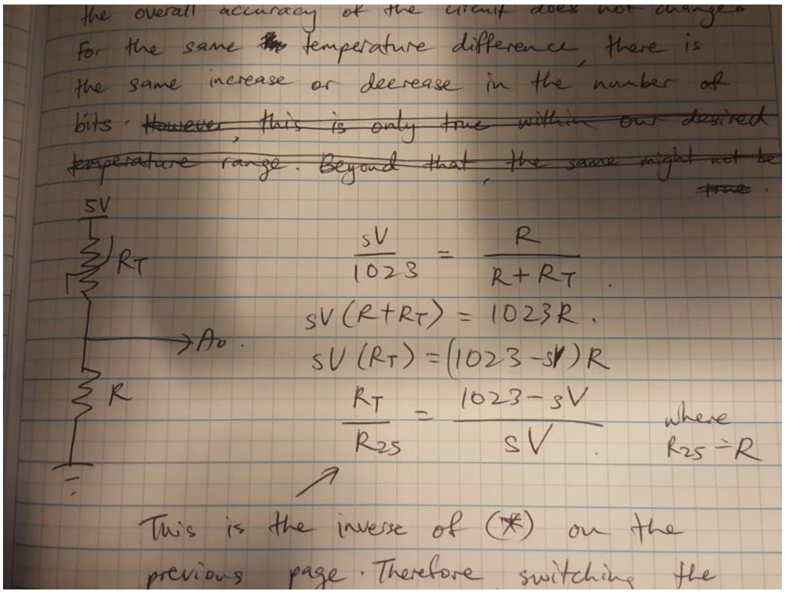
BIT VALUE vs Number of Samples



We know that from above,

When thermistor was touched, sensorValue decreased. This means that RT decreased when the temperature increased. This is consistent with an NTC thermistor.

When the thermistor and resistor are interchanged, the overall accuracy of the circuit does not change. For the same temperature difference, there is the same increase or decrease in the number of bits.



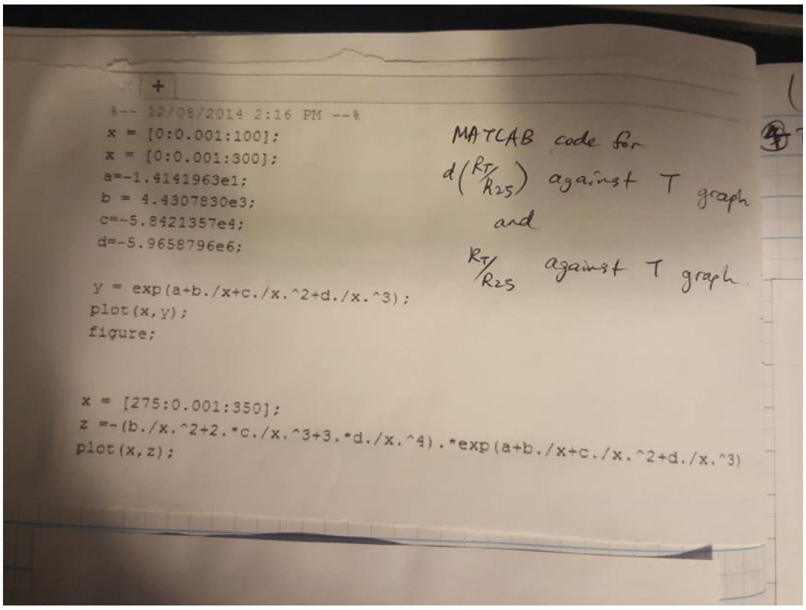
PICTUREEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEE

Question 4 answered before Question 3

4. The bit range we have on the Arduino is from 1 to 1023 to give a valid temperature. The relationship between temperature and bit value used is nonlinear.

TEMPERATURE READING vs BIT VALUE

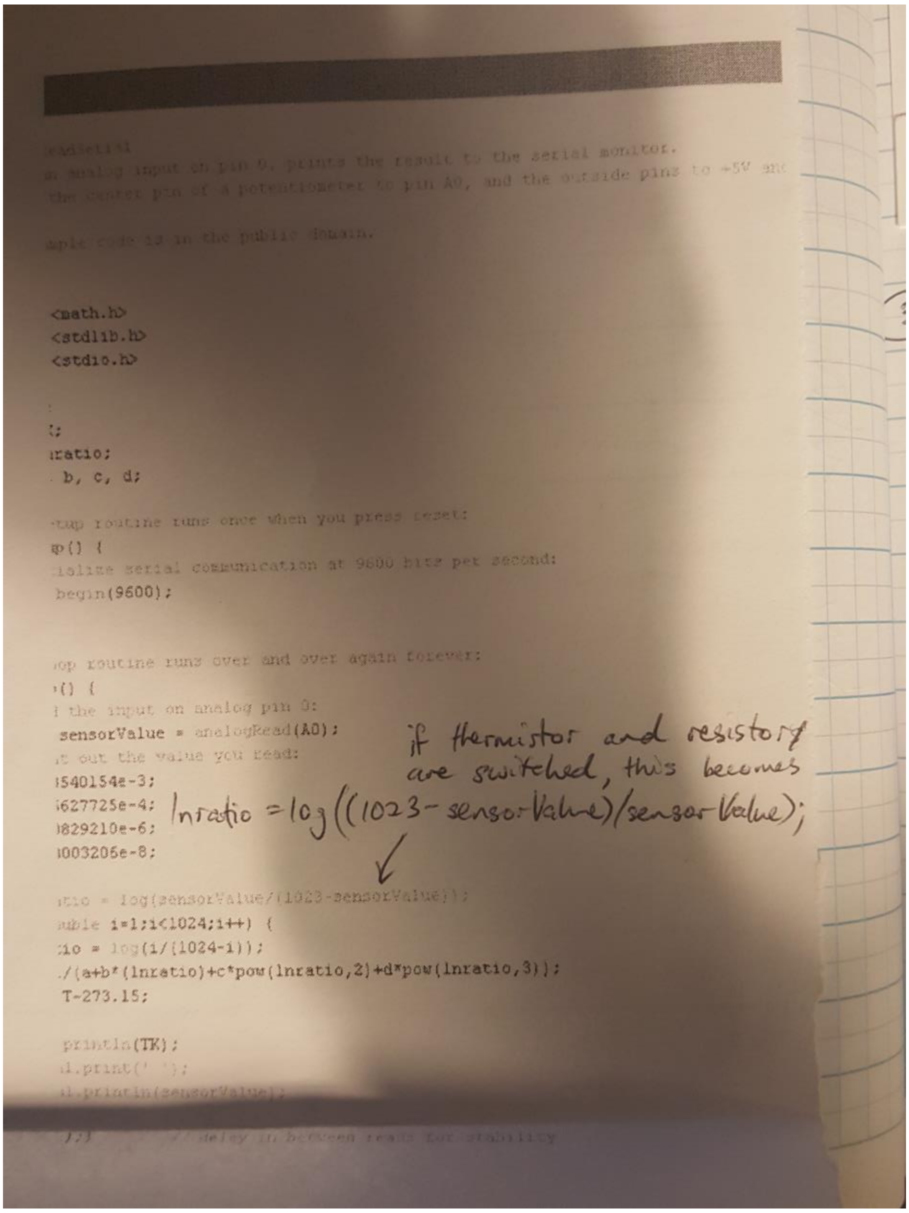
3. Within 5 degrees and 45 degrees, the bit values that represent this temperature range is between 312 and 735. That is ~424 bits used to represent 5 degrees to 45 degrees in the A/D converter. Also, within this region it is roughly linear (not exactly linear as we will see in Q5).



PICTUREEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEE

5. We are given that:

PICTUREEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEE



```
<math.h>
<stdlib.h>
<stdio.h>

ratio;
b, c, d;

begin(9600);

sensorValue = analogRead(A0);
5540154e-3;
5627725e-4;
3829210e-6;
1003206e-8;

T-273.15;

println(TK);
```

if thermistor and resistor are switched, this becomes
lnratio = log((1023-sensorValue)/sensorValue);

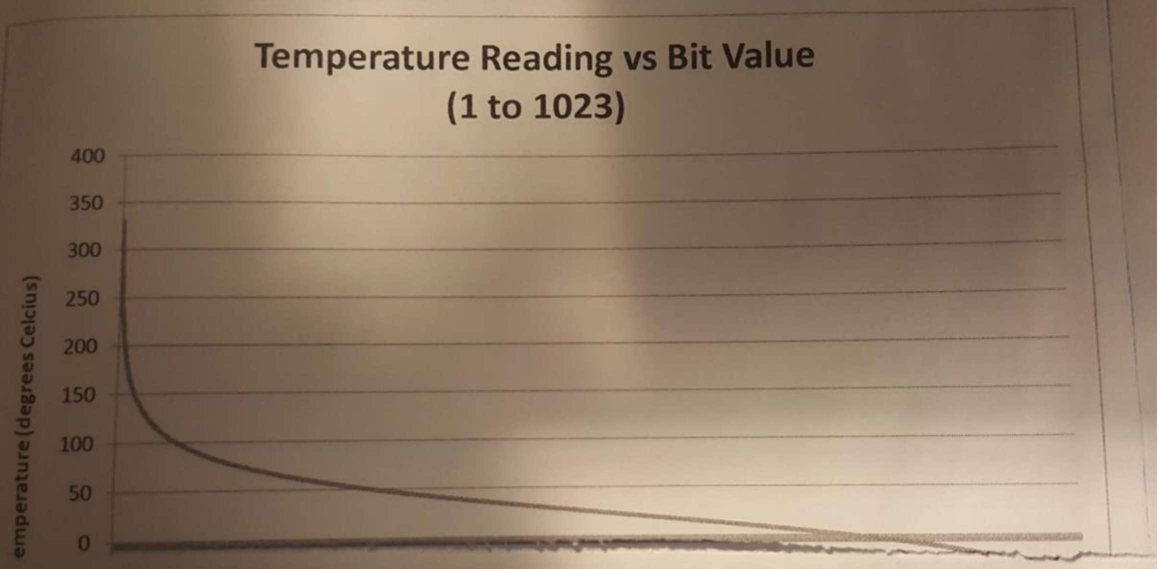
The second figure shows that as temperature increases (270K to 350K), the rate of change of RT/R25 decreases that is, delta T increases, delta R decreases.

A change in temperature at lower temperatures give a bigger change in resistance than a change in temperature at higher temperatures.

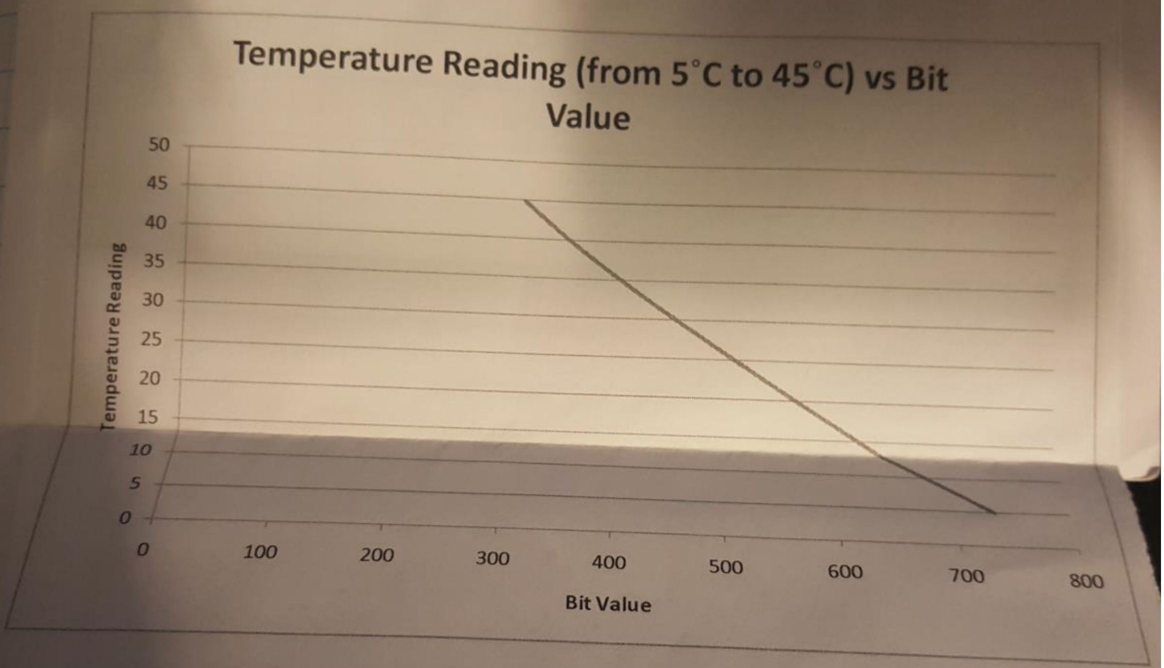
6. From the plot below, the difference in temperatures represented by adjacent bit values is between 0.08 to 0.12. (for 5 degrees to 45 degrees)

PICUTREEEEEEEEEEEEEEEEEEEEEEEE

used is non-linear.



Within 5°C and 45°C, the ~~bits~~ values that represent



Hence, the sensitivity is given by the biggest possible change in temperature (i.e. 0.12). The change in 1 bit (increase or decrease) causes a change of 0.12 degrees in temperature (increase or decrease).

5 We are given that:

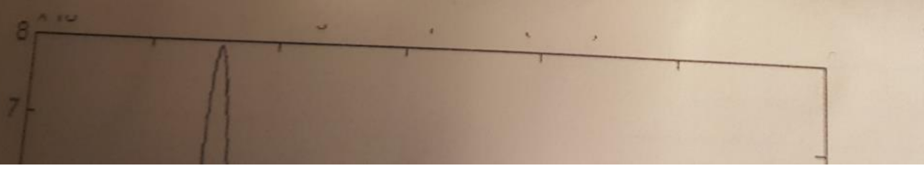
$$\frac{R_T}{R_{25}} = \exp\left(A + B/T + C/T^2 + D/T^3\right)$$

$$\frac{d(R_T/R_{25})}{dT} = -\left(\frac{B}{T^2} + \frac{2C}{T^3} + \frac{3D}{T^4}\right)\exp\left(A + B/T + C/T^2 + D/T^3\right)$$

$$\frac{dR}{dT} \approx \frac{\Delta R}{\Delta T}$$

Plotting these two equations in MATLAB, we obtain:

R_T/R_{25} against Temperature.



$$\frac{d(R_T/R_{25})}{dT} = \left(\frac{\ }{T^2} + \frac{\ }{T^3} \quad T^4 \right.$$

give

che

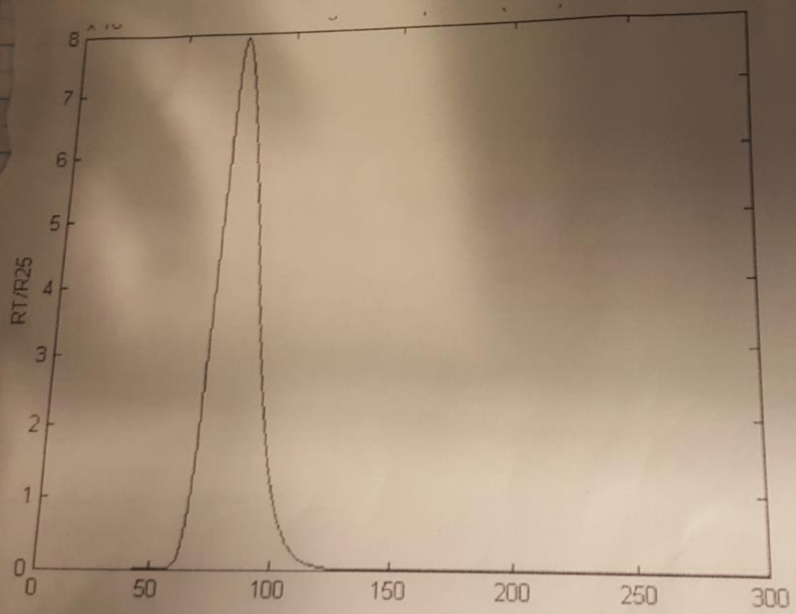
$$\frac{dR}{dT} \approx \frac{\Delta R}{\Delta T}$$

Plotting these two equations in MATLAB, we obtain:

Th

⑥

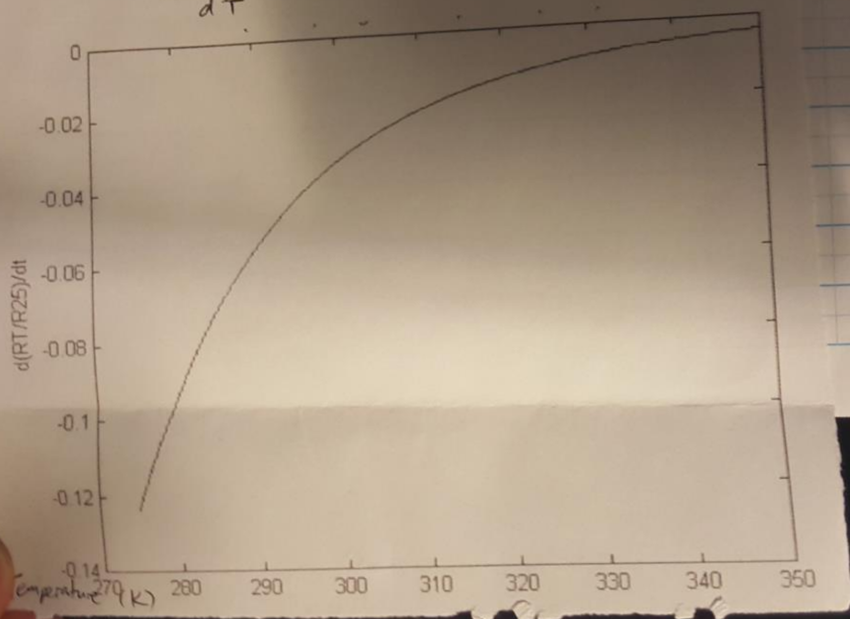
R_T/R_{25} against Temperature.



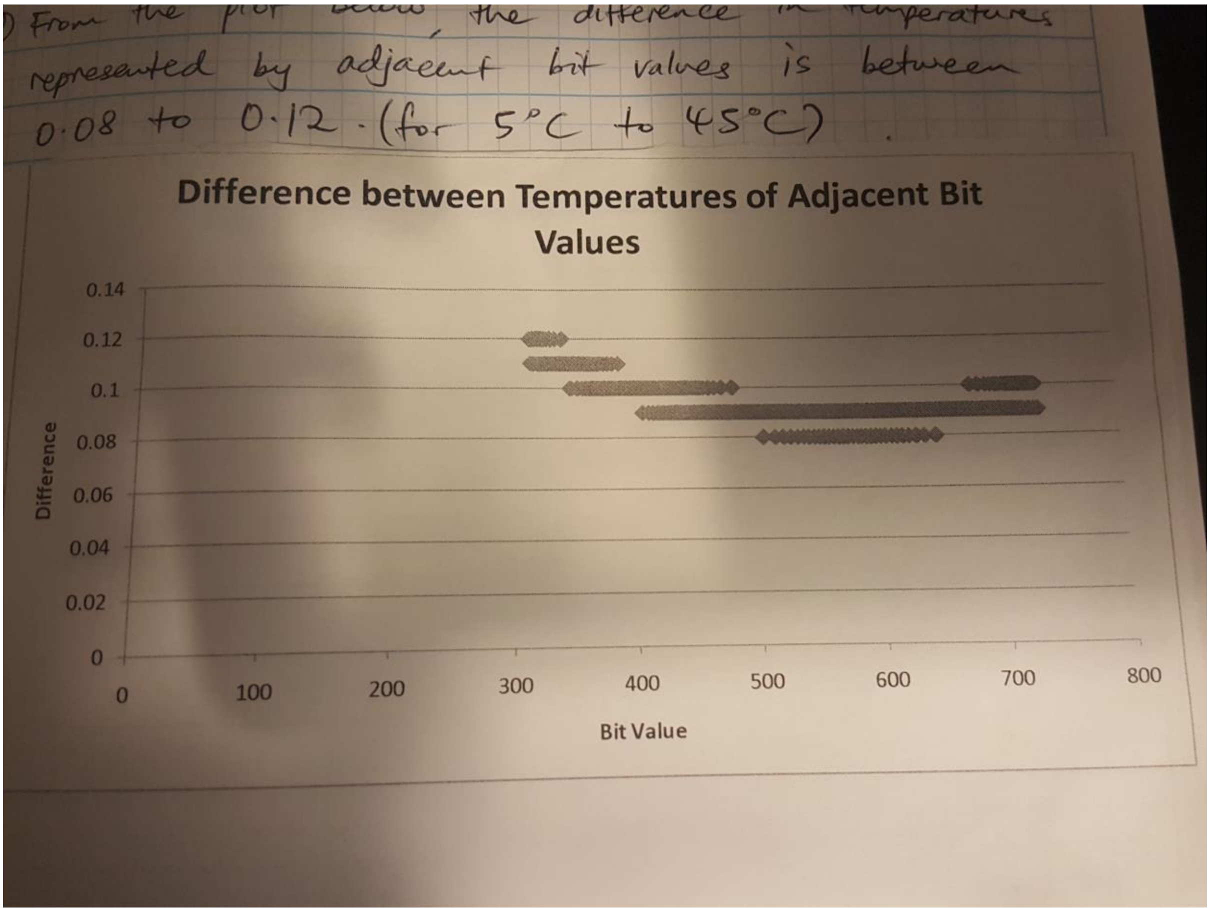
Temperature (Kelvin)

Temperature (Kelvin)

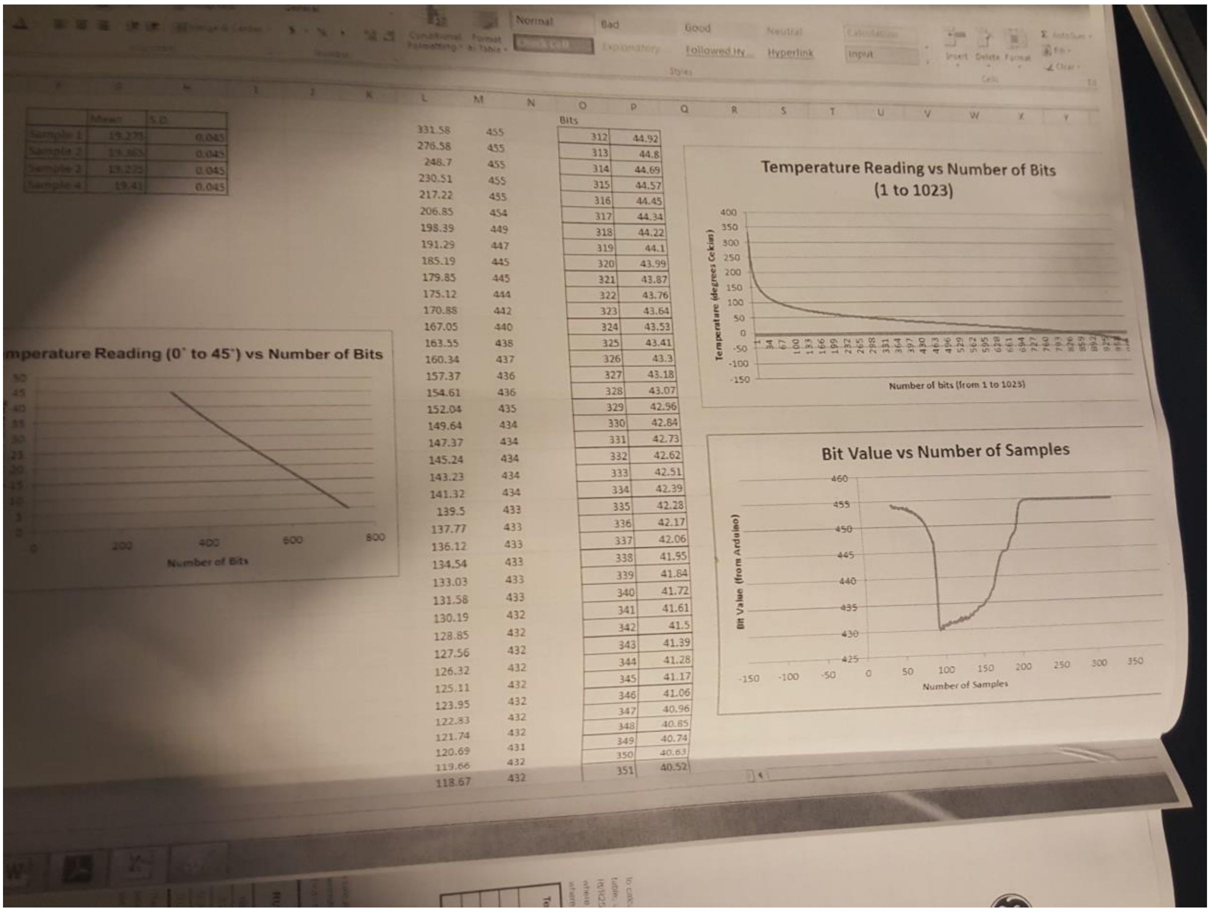
$\frac{d(R_T/R_{25})}{dT}$ against $T(K)$.



Temperature (K)



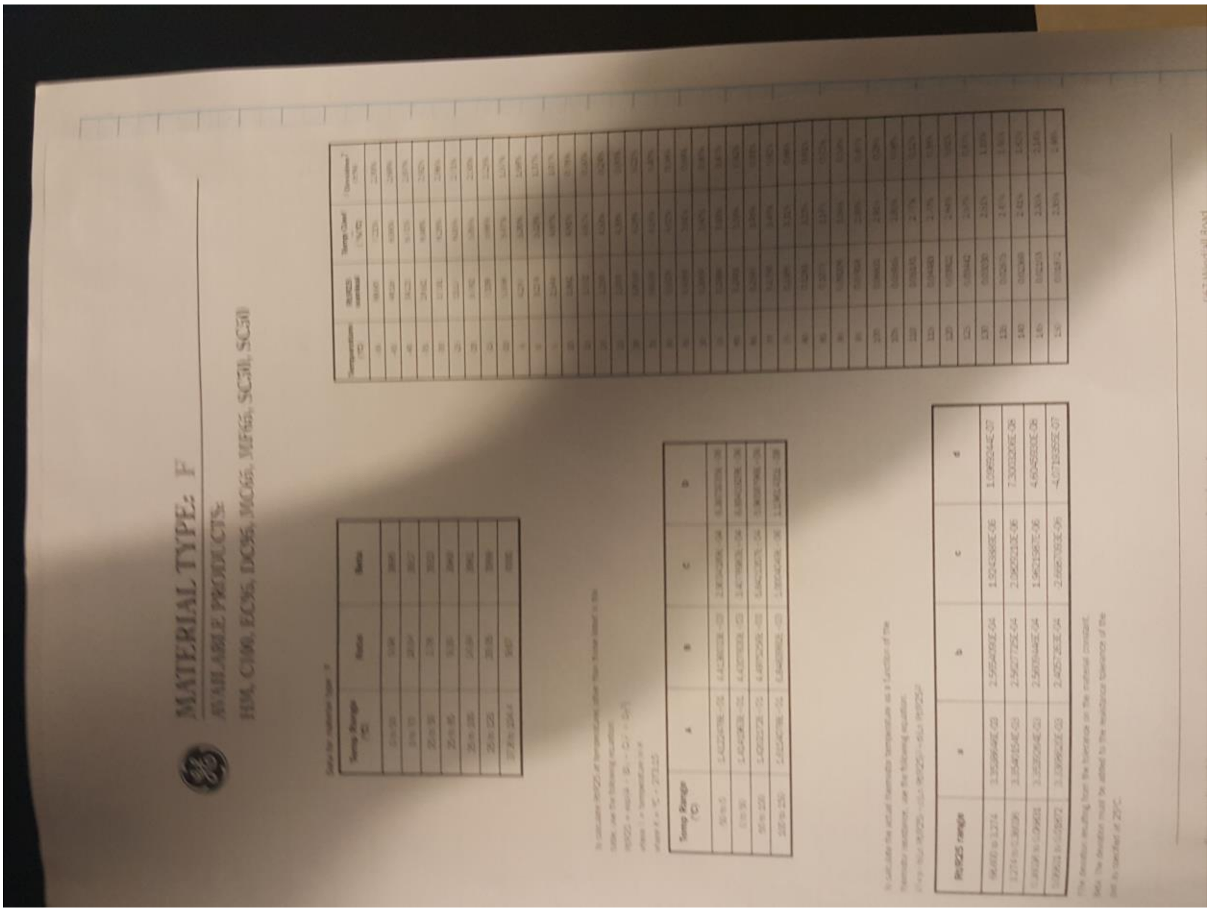
SCREENSHOT AND DAATASHEETTTTTTTTTTTTTTTTTTTTTT



	Mean	S.D.
Sample 1	19.275	0.045
Sample 2	19.365	0.045
Sample 3	19.275	0.045
Sample 4	19.41	0.045

331.58	455
276.58	455
248.7	455
230.51	455
217.22	455
206.85	454
198.39	449
191.29	447
185.19	445
179.85	445
175.12	444
170.88	442
167.05	440
163.55	438
160.34	437
157.37	436
154.61	436
152.04	435
149.64	434
147.37	434
145.24	434
143.23	434
141.32	434
139.5	433
137.77	433
136.12	433
134.54	433
133.03	433
131.58	433
130.19	432
128.85	432
127.56	432
126.32	432
125.11	432
123.95	432
122.83	432
121.74	432
120.69	431
119.66	432
118.67	432

Bits	
312	44.92
313	44.8
314	44.69
315	44.57
316	44.45
317	44.34
318	44.22
319	44.1
320	43.99
321	43.87
322	43.76
323	43.64
324	43.53
325	43.41
326	43.3
327	43.18
328	43.07
329	42.96
330	42.84
331	42.73
332	42.62
333	42.51
334	42.39
335	42.28
336	42.17
337	42.06
338	41.95
339	41.84
340	41.72
341	41.61
342	41.5
343	41.39
344	41.28
345	41.17
346	41.06
347	40.96
348	40.85
349	40.74
350	40.63
351	40.52



Workshop 4: Phase Shift Oscillator

Introduction: