In The Name Of God

Instrumentation

CA3

Professor: Mohammad R. Nayyeri

Producer:
Mohammad Ali Shakerdargah

Student number: 810196487

Summer 2020

Main TA: Arash Irafar

Question 1:

❖ Part A

This question is about **LM35** sensor, we will discuss its Scale Factor, also known as sensitivity, its accuracy, its range and finally how it should be used and biased.

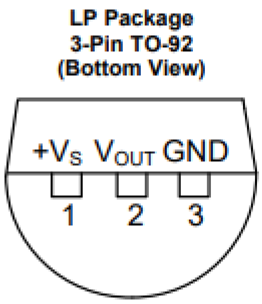
It Based on **Texas instrument** Data Sheet, **LM35** is Linear + 10-mV/°C Scale Factor.

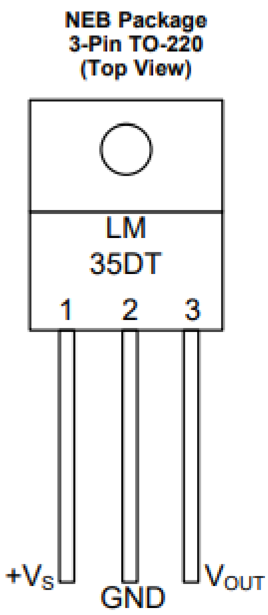
It has 0.5°C Ensured Accuracy.

Its range is from −55°C to 150°C.

And Best range for input voltage is between 4Volts and 30Volts.

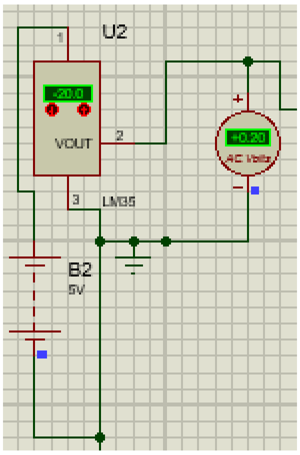
Now how we should use the pins in general:





There you can see that there are three pins. One should be connected to Vcc which is mostly 5Volts the other is GND which is ground and the other pin is the output of our sensor.

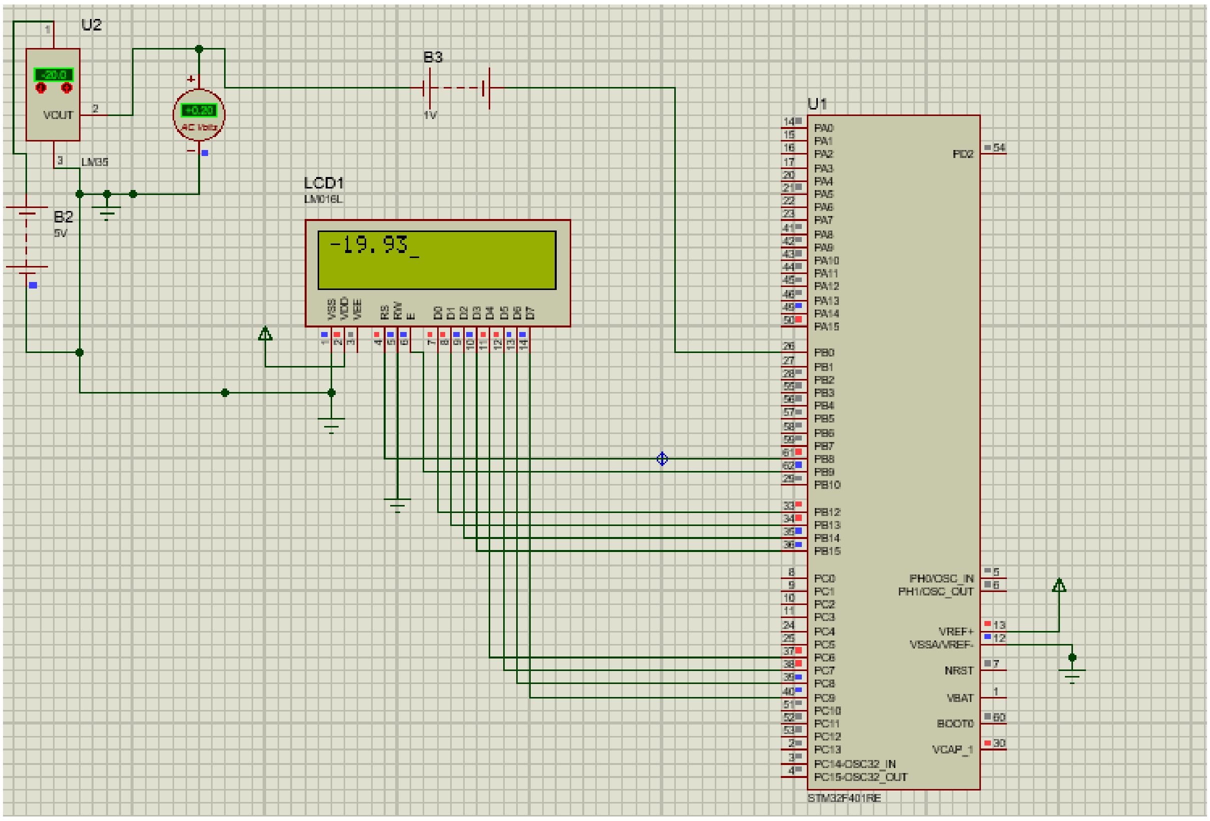
Now how we should use the pins in Proteus:



In above figure we have LM35 that shows $-20°C$ and you can see the short difference, in proteus, 1st pin is Vcc, 2nd pin is output and 3rd pin is GND.

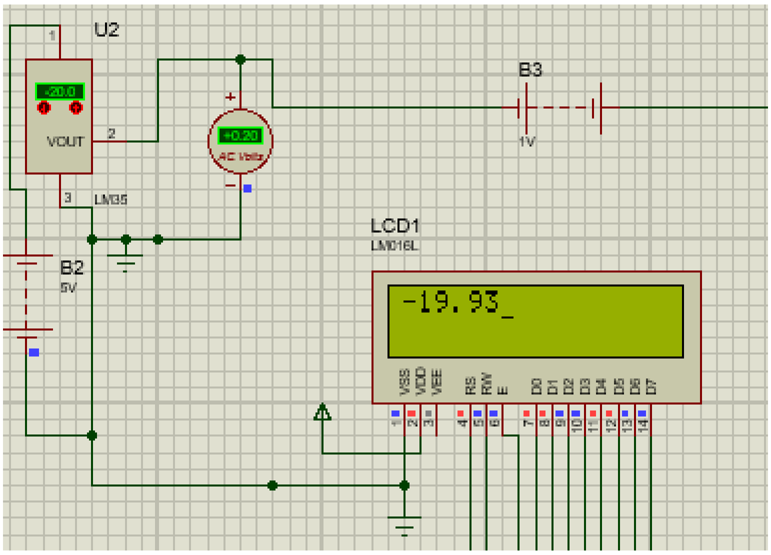
❖ Part B

We wrote the code and Here is the simulation for it:



❖ Part C

And if we zoom it we will see the accuracy:



We set the temperature to $-20℃$ and we got $-19.93℃$ which has $0.0.07℃$ error and we can say that its accuracy rate is close to $\pm 0.1℃$ that is close to $\pm 0.5℃$.

The reason that our accuracy is good is that we are only simulating, in real world and real project there might be a bigger possibility that we get larger error.

ERROR of instrument:

We know that ADC has binary 12bits , so we would have $2^{12} = 4096$

Our equation for temperature is:

$$\theta = \frac{(Value_{detected} - 819.2)}{100 * V_{ref}}$$

And we gave it a 5Volts so the equation would be like bellow:

$$\theta = \frac{(Value_{detected} - 819.2)}{500}$$

And because we have 0.5 ensured accuracy, so our error would get:

$$0.5 * \frac{100 * 5}{4096} = 0.061℃$$

Now we would declare it in to percent:

$$\frac{0.061}{150 - (-55)} = \ 0.000297$$

Question 2:

- Part A

This question is about **TSL251** sensor, we will discuss its Scale Factor, also known as sensitivity, its accuracy, its range and finally how it should be used and biased.

We have same data sheet for TSL250 and TSL251 and TSL252 Light-to-Voltage Optical Sensors.

It Based on **OMS** Data Sheet, TSL251 is Linear with High Irradiance Responsivity **137mV/(µW/cm2)** at λp = 635nm (TSL250R and TSL251R), we will discuss this in the simulation of proteus too and find a better constant to multiply so that our output would get closer to real value.

In other data sheets like Texas instrument it has said that its **scaler factor** is

$\frac{3020}{13.66.12} = 2.21 \left[\frac{mv}{lux}\right]$ and its **accuracy** is 1.366 lux (0.2/(µW/cm2)) , which is different from 137mV/(µW/cm2) at λp = 635nm, the reason is their difference in λ.

This sensor is more complicated than LM35, let us see use the description to fully describe this sensor.

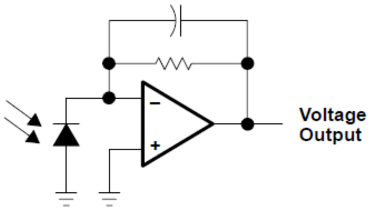
This sensor is a single supply.

Its low dark offset voltage is **10mv**, and its wide **supply voltage Range is from 3Volts to 9Volts.**

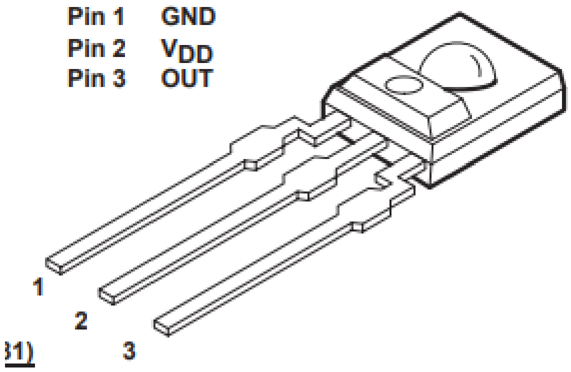
And base on data sheet:

Supply voltage, V_{DD} (see Note 1)	10 V
Output current, I_O	±10 mA
Duration of short-circuit current at (or below) 25°C (see Note 2)	5 s
Operating free-air temperature range, T_A	−25°C to 85°C
Storage temperature range, T_{stg}	−25°C to 85°C
Lead temperature 1,6 mm (1/16 inch) from case for 10 seconds	240°C

Its functional block diagram is like bellow:

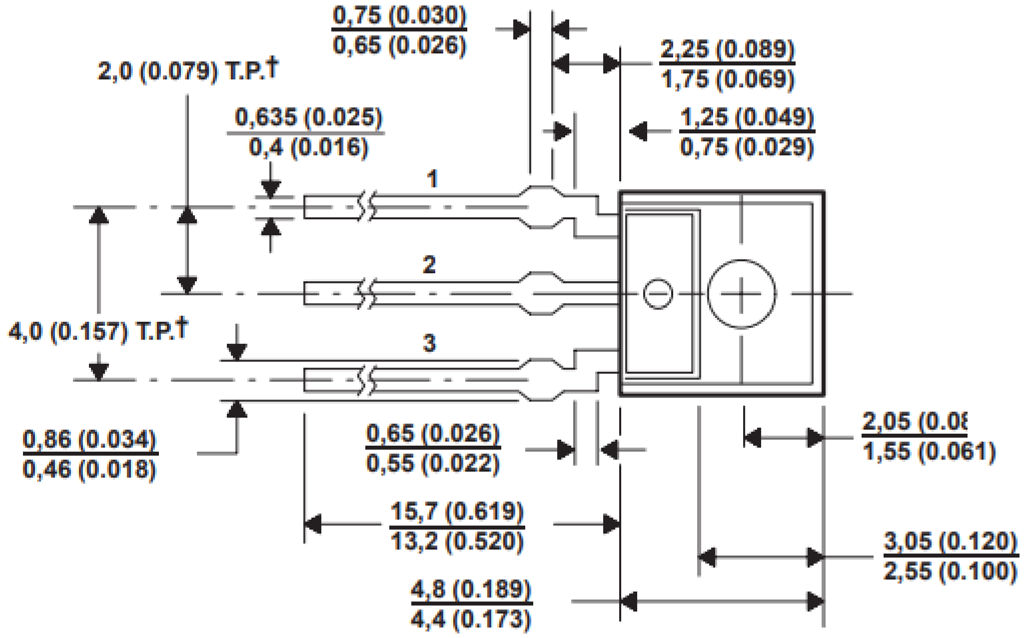


Now how we should use the pins in general:

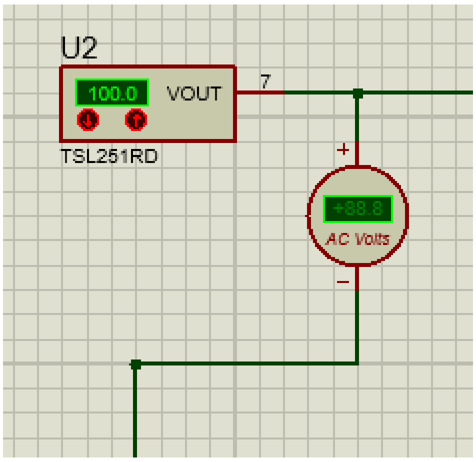


We can totally see that pin 1st is for GND which is ground, 2nd pin is for VDD which we would get 5Volts, and pin 3rd is the output voltage.

In bellow figures, you can see other chematics:



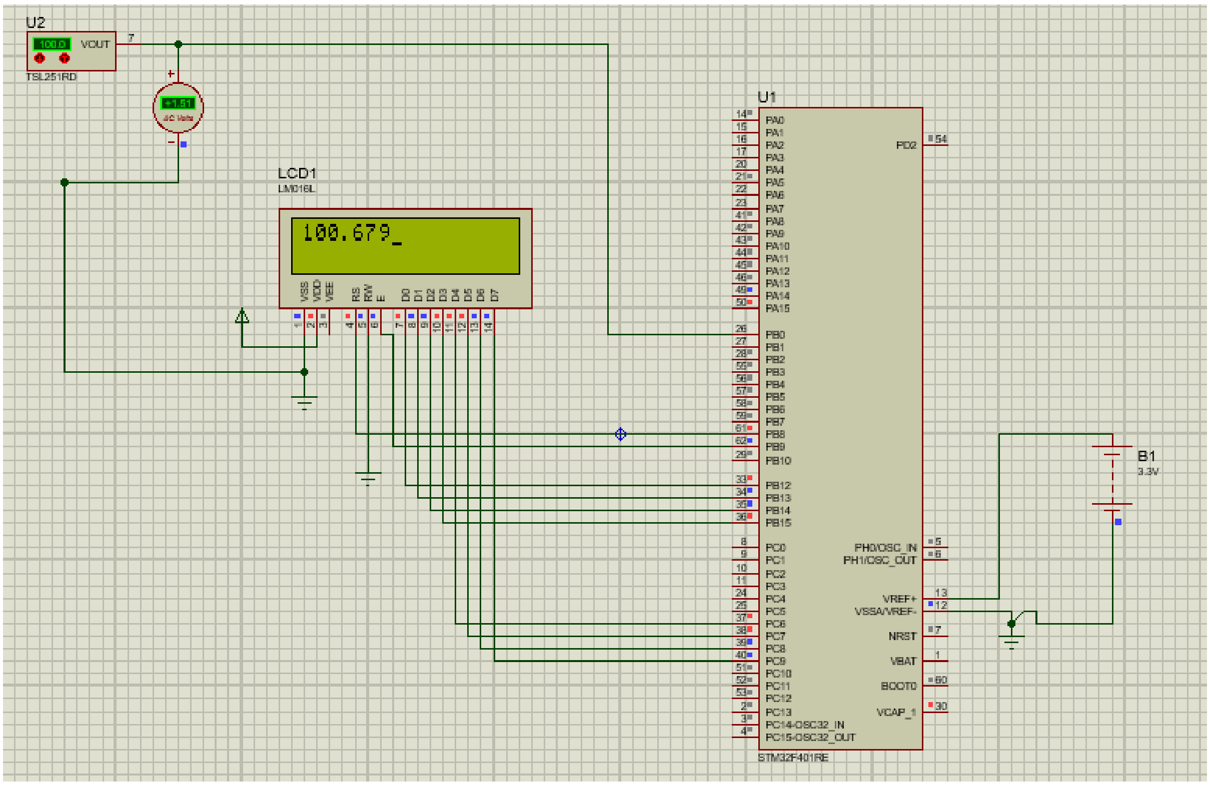
Now how we should use the pins in Proteus:



We only have to use the output voltage.

❖ Part B

We wrote the code and Here is the simulation for it:



We set the sensor to 100lux and after processing the output of sensor (which we turned to digital) we got a close amount. The main thing that we changed in this simulation is sensitivity, I examined 15 different outputs of our sensor and it had the relation like bellow:

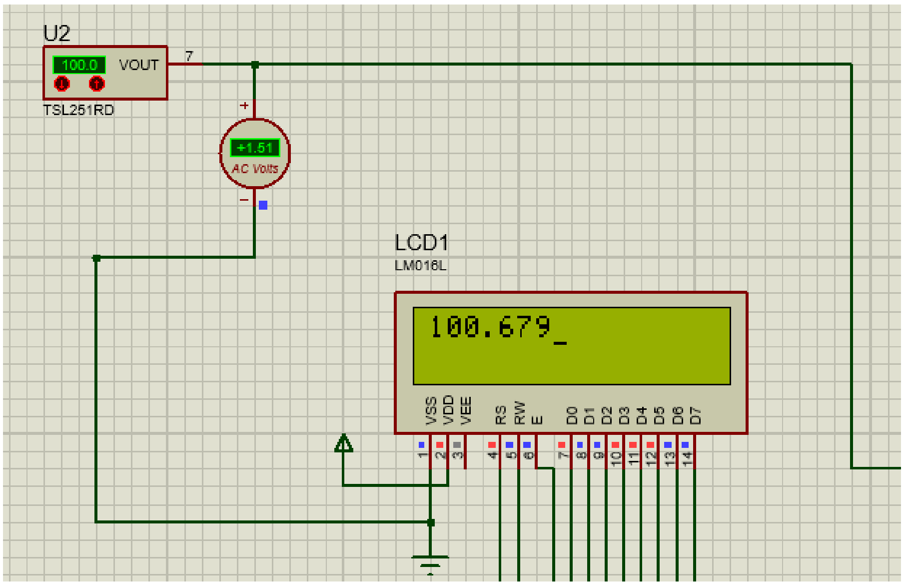
- Part C

Since we used a 3.3Volts battery for our MCU,

$$\text{signal} = (\text{Digital}_{Value} * 3.3) * 100$$

$$\text{signal}_{\text{desired}} = (\text{signal} * \frac{2}{3})$$

And we would get a great accuracy showed bellow:



We set the input to $100lux$ and we got $100.679lux$ which has $0.679lux$ error and we can say that its accuracy rate is close to $\pm 0.7lux$ $(not\ even\ 1lux)$.

The reason that our accuracy is good is that we are only simulating, in real world and real project there might be a bigger possibility that we get larger error.

ERROR of instrument:

We know that ADC has binary 12bits , so we would have $2^{12} = 4096$

Our equation for output is:

$$\text{signal} = (\text{Digital}_{Value} * 3.3) * 100$$

$$\text{signal}_{\text{desired}} = (\text{signal} * \frac{2}{3})$$

So we would have $\frac{5}{4096}$ which is 1.2mv and based on what we have in above equation:

$$\left(1.2 * \frac{2}{3}\right) = 0.8mv$$

Then based on datasheet:

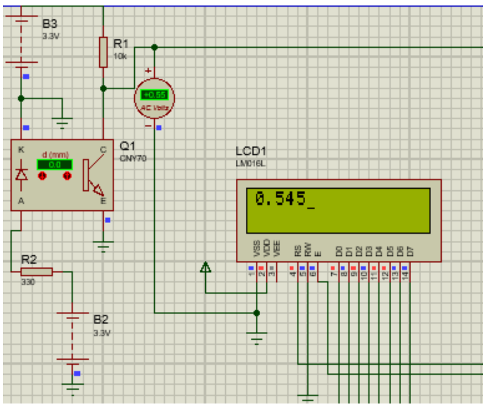
$$(0.8/2.21) = 0.361mv$$

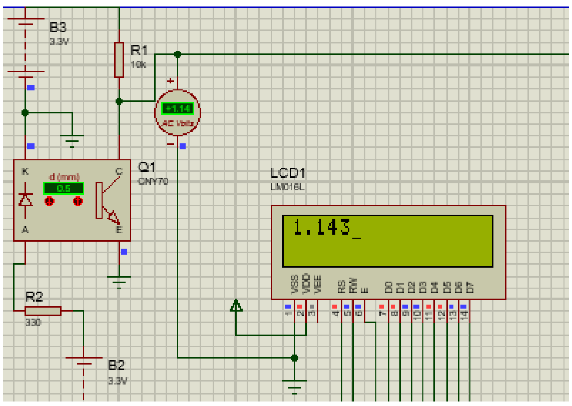
Question 3:

❖ Part A

This question is about **CNY70** sensor, we use a 3.3Volt battery for its VCC, now we will fulfill the desired table:

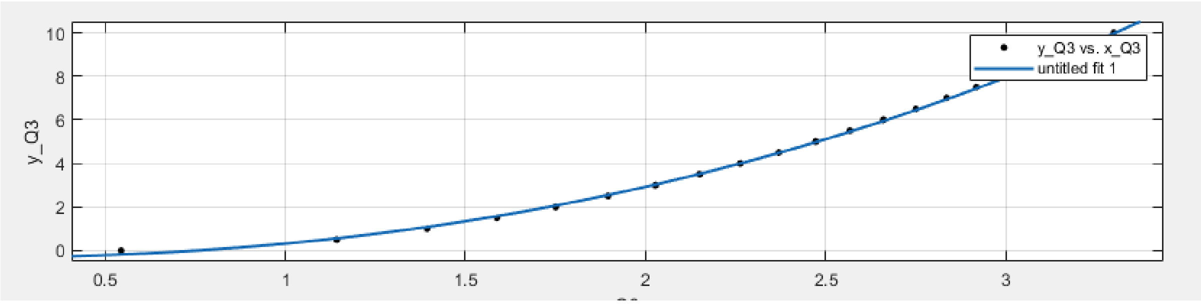
D(cm)	Vout(v)
0	0.545
0.5	1.143
1	1.394
1.5	1.588
2	1.751
2.5	1.896
3	2.028
3.5	2.150
4	2.263
4.5	2.370
5	2.472
5.5	2.567
6	2.660
6.5	2.750
7	2.835
7.5	2.918
8	2.999
8.5	3.077
9	3.153
9.5	3.277
10	3.299





❖ Part B

After giving the above table to **cftool** of MATLAB, we received this output:

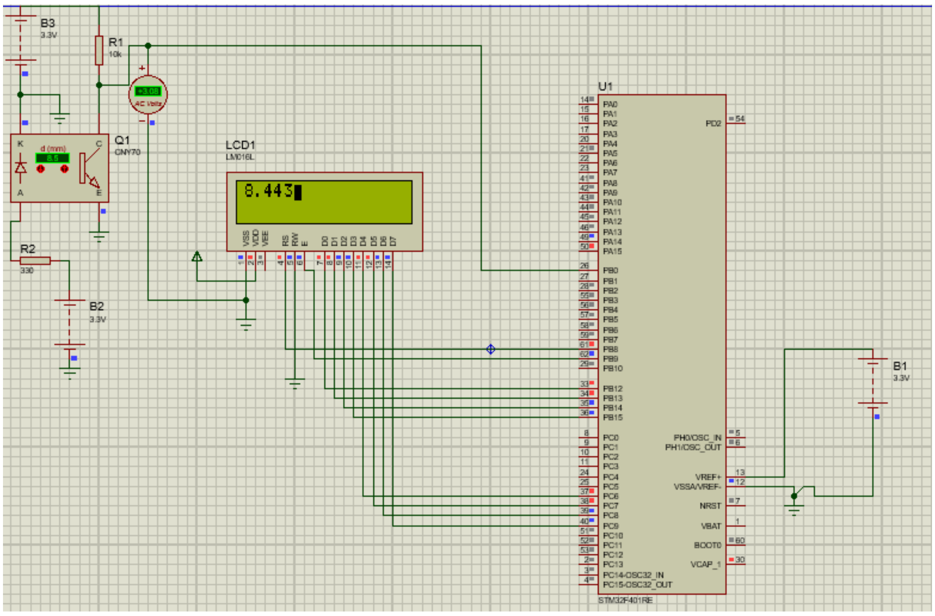


Results

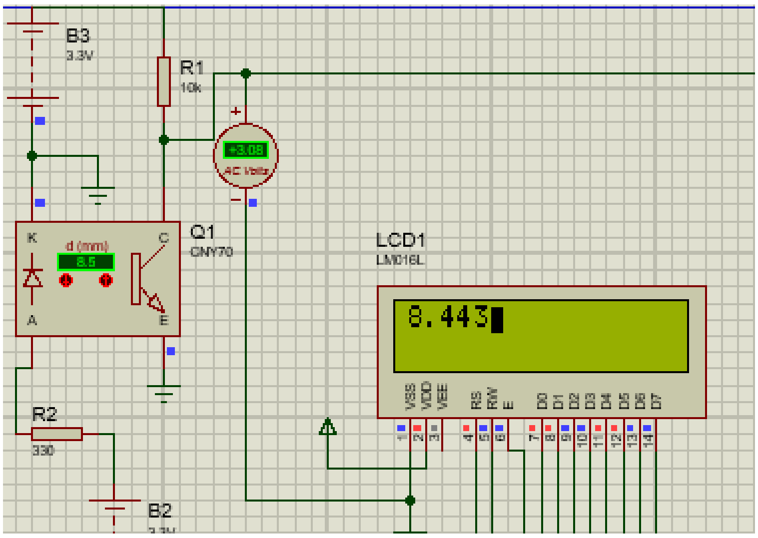
```
General model Power2:
    f(x) = a*x^b+c
Coefficients (with 95% confidence bounds):
      a =      0.6672  (0.5793, 0.7552)
      b =       2.294  (2.192, 2.396)
      c =     -0.3457  (-0.5322, -0.1593)

Goodness of fit:
  SSE: 0.1805
  R-square: 0.9991
  Adjusted R-square: 0.999
```

❖ Part C

We used "math.h" in our code and built the equation, the results are as followed:



If we zoom in we would see:



❖ Part D

We set the input to $8.5mm$ and we got $5.443mm$ which has $0.007mm$ error and we can say that its accuracy rate is close to $\pm 0.01mm$.

The reason that our accuracy is good is that we are only simulating, in real world and real project there might be a bigger possibility that we get larger error.

ERROR of instrument:

We know that ADC has binary 12bits , so we would have $2^{12} = 4096$

In this case our V_{ref} is 3.3Volts.

Our equation for output is:

$$\text{signal} = (\text{Digital}_{Value} * 3.3) * 100$$

$$\text{signal}_{\text{desired}} = 0.6 * (\text{signal}^{2.3}) - 0.34$$

So we would have $\frac{3.3}{4096}$ which is 0.8mv and based on what we have in above equation:

$$(0.6 * (0.8)^{2.3}) - 0.34 = 0.0191$$

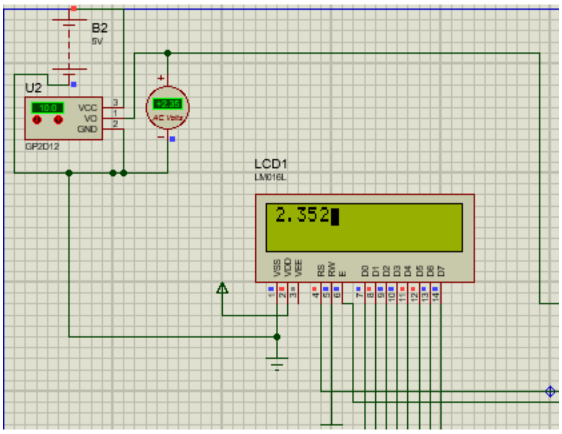
So this will be our error.

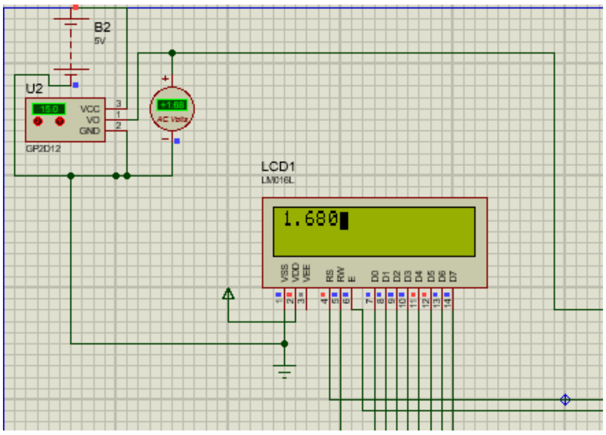
Question 4:

❖ Part A

This question is about **GP2D12** sensor, we use a 5Volt battery for its VCC, now we will fulfill the desired table:

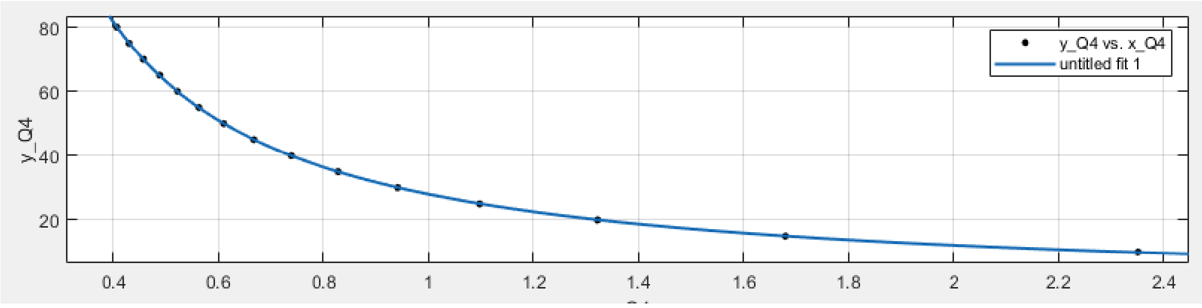
D(cm)	Vout(v)
10	2.352
15	1.680
20	1.322
25	1.098
30	0.942
35	0.828
40	0.739
45	0.668
50	0.610
55	0.563
60	0.522
65	0.488
70	0.457
75	0.430
80	0.407

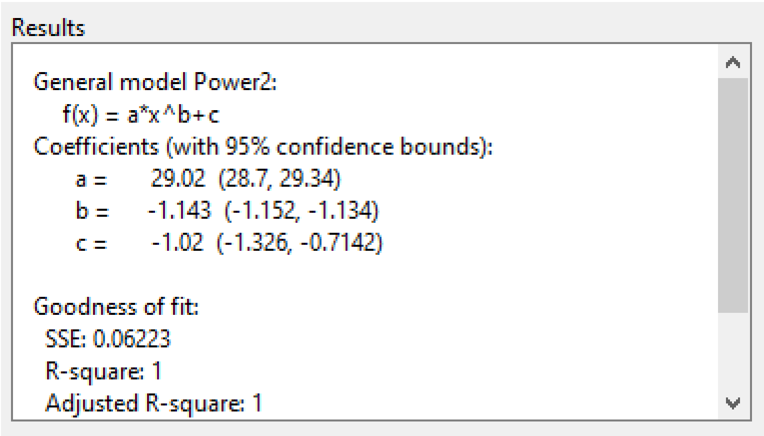




❖ Part B

After giving the above table to **cftool** of MATLAB, we received this output:





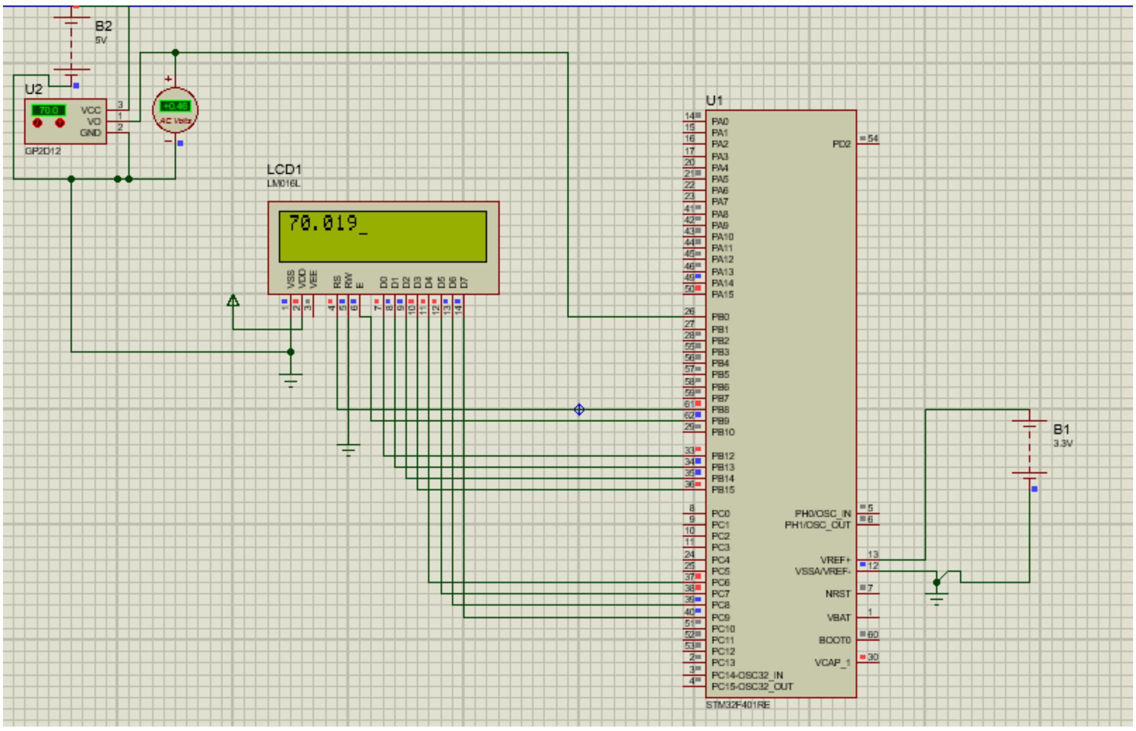
Results

General model Power2:
f(x) = a*x^b+c
Coefficients (with 95% confidence bounds):
a = 29.02 (28.7, 29.34)
b = -1.143 (-1.152, -1.134)
c = -1.02 (-1.326, -0.7142)

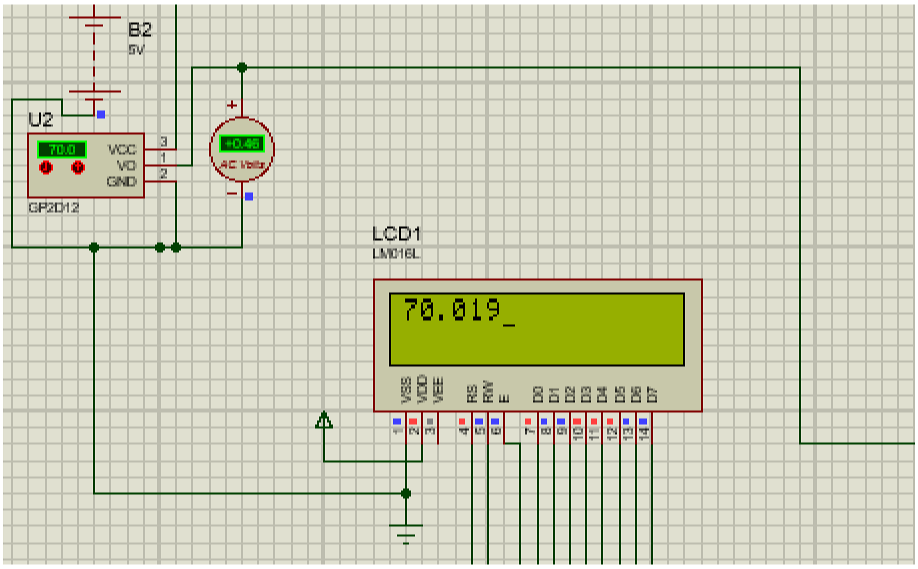
Goodness of fit:
SSE: 0.06223
R-square: 1
Adjusted R-square: 1

❖ Part C

We used "math.h" in our code and built the equation, the results are as followed:



If we zoom in we would see:



❖ Part D

We set the input to $70cm$ and we got $70.019cm$ which has $0.0.19cm$ error and we can say that its accuracy rate is close to $\pm 0.02cm$.

The reason that our accuracy is good is that we are only simulating, in real world and real project there might be a bigger possibility that we get larger error.

ERROR of instrument:

We know that ADC has binary 12bits , so we would have $2^{12} = 4096$

In this case our V_{ref} is 5Volts.

Our equation for output is:

$$\text{signal} = (\text{Digital}_{Value} * 3.3) * 100$$

$$\text{signal}_{\text{desired}} = 29 * (\text{signal}^{-1.14}) - 1.02$$

So we would have $\frac{5}{4096}$ which is 1.2mv and based on what we have in above equation:

$$29 * (1.2^{-1.14}) - 1.02$$

So this will be our error.

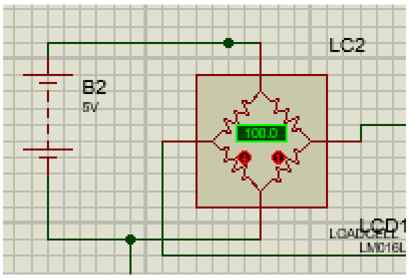
Question 5:

❖ Part A

This question is about **Loadcell** sensor, we use a 5Volt battery for its VCC, now we will fulfill the desired table:

In this part we are going to try to calculate sensitivity in full load (100Kg) mode:

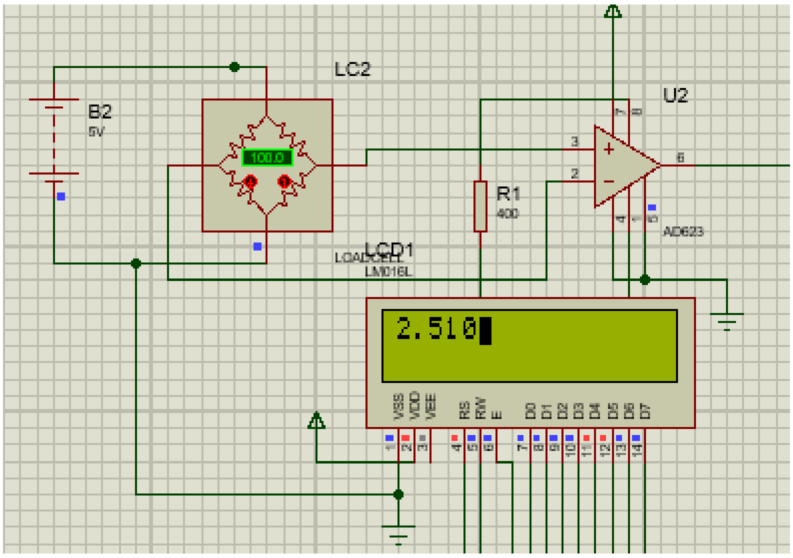
We will take loadcell from proteus library and add it to our workspace and the shape would be like bellow:



After several experiments and based on what we got in results, its scaler rate or sensivity is close to $2[\frac{mv}{v}]$

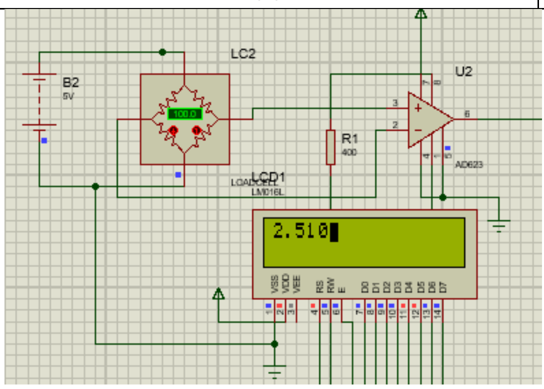
❖ Part B

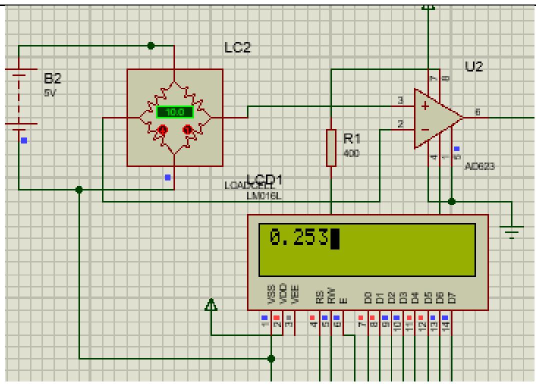
We used AD623 to get the projection that we wanted, now we have projected 100kg to (we can say close to) 2.5Volts, to do so we needed a 400Ω resistor.



Now everything is set, we would fulfill the table that is desired,

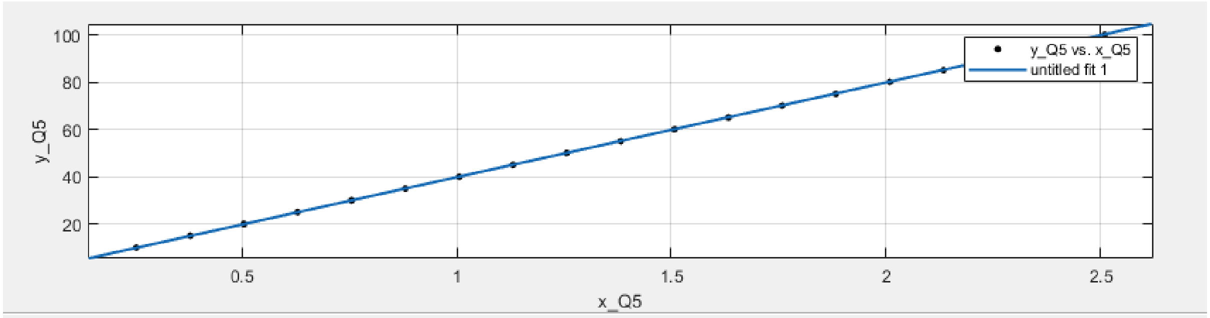
W(kg)	Vout(Amplifier)(v)
10	0.253
15	0.379
20	0.504
25	0.629
30	0.755
35	0.880
40	1.006
45	1.131
50	1.256
55	1.382
60	1.507
65	1.633
70	1.758
75	1.883
80	2.009
85	2.134
90	2.260
95	2.385
100	2.510

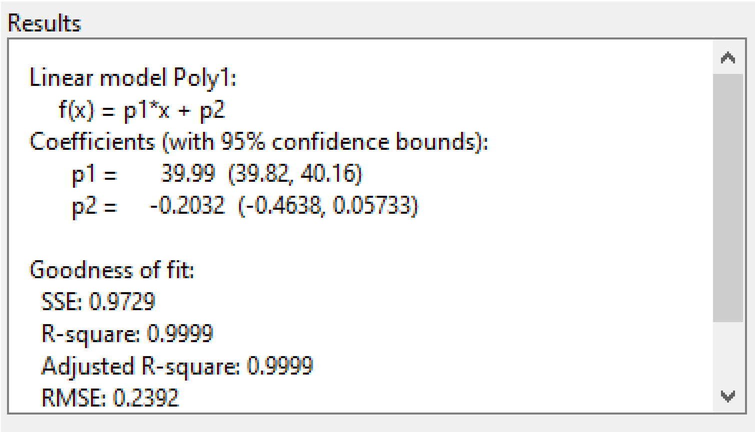




❖ Part C

After giving the above table to **cftool** of MATLAB, we received this output:



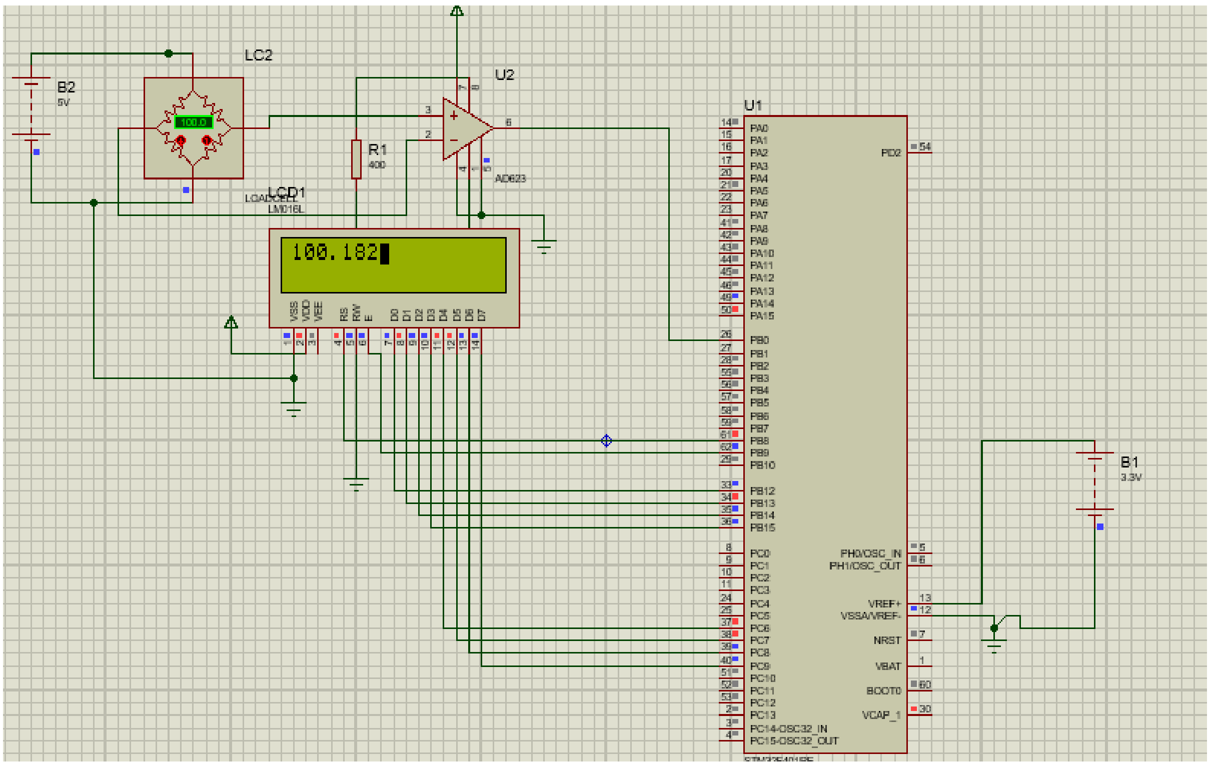

Results

Linear model Poly1:
    f(x) = p1*x + p2
Coefficients (with 95% confidence bounds):
    p1 =   39.99 (39.82, 40.16)
    p2 =   -0.2032 (-0.4638, 0.05733)

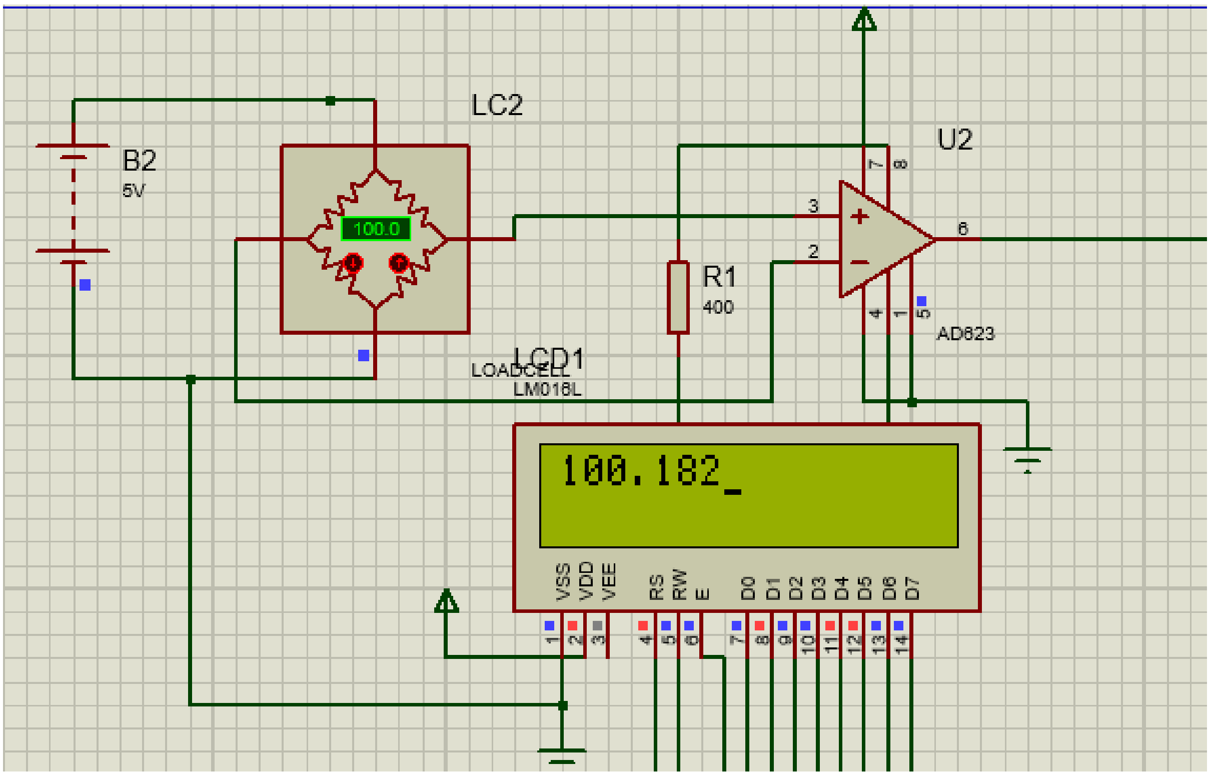
Goodness of fit:
  SSE: 0.9729
  R-square: 0.9999
  Adjusted R-square: 0.9999
  RMSE: 0.2392

❖ Part D

After setting the results from previous part in our code, we will be able to get the desired output, the figure is shown bellow:



And if we zoom in we will see:



❖ Part D

We set the input to $100Kg$ and we got 100.082 which has $0.082Kg$ error and we can say that its accuracy rate is close to $\pm 0.1Kg$.

The reason that our accuracy is good is that we are only simulating, in real world and real project there might be a bigger possibility that we get larger error.

ERROR of instrument:

We know that ADC has binary 12bits , so we would have $2^{12} = 4096$

In this case our V_{ref} is 5Volts.

Our equation for output is:

$$\text{signal} = (\text{Digital}_{Value} * 3.3) * 100$$

$$\text{signal}_{\text{desired}} = 40 * (\text{signal}) - 0.2$$

So we would have $\frac{5}{4096}$ which is 1.2mv and based on what we have in above equation:

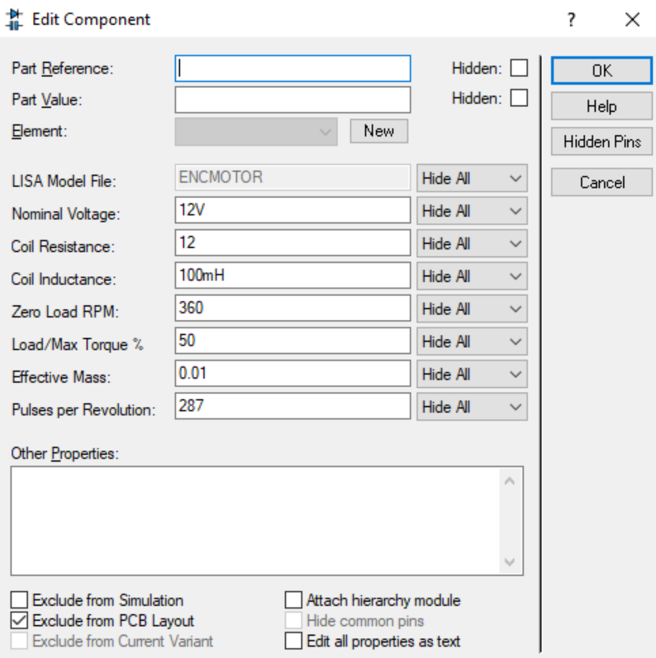
$$40 * (1.2) - 1.02$$

So this will be our error.

Question 6:

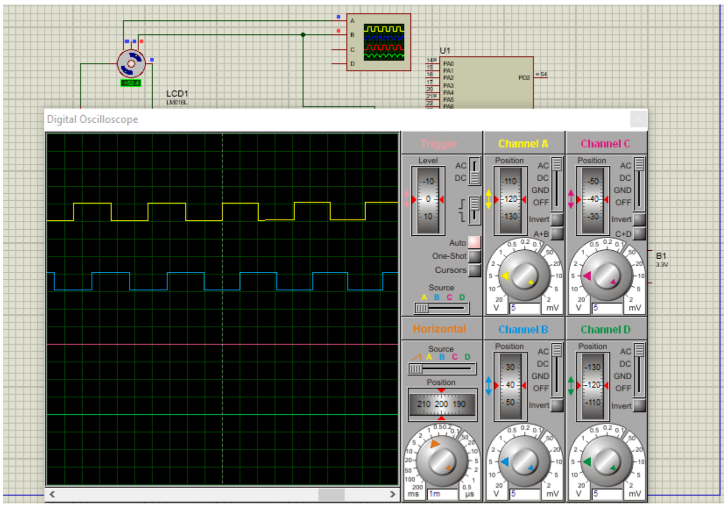
❖ Part A

In this question we are going to work with **MOTOR-ENCODER,** in the first part we would set the parameter to 200 +87(which is last 2 digits of my student number) and we would connect it to 12Volts battery the result is shown bellow:



❖ Part B

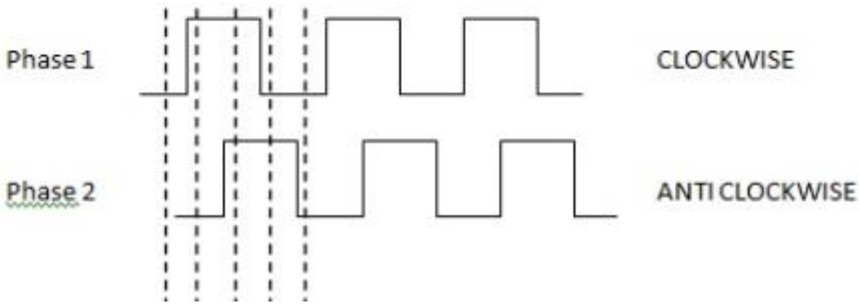
In this part after connecting it to 12Volts battery, we would see the pulses of A and B in the scope:



We can see that the yellow one that is connected to A of scope is truly A because it is earlier than B. the frequency increases in time.

❖ Part C & D

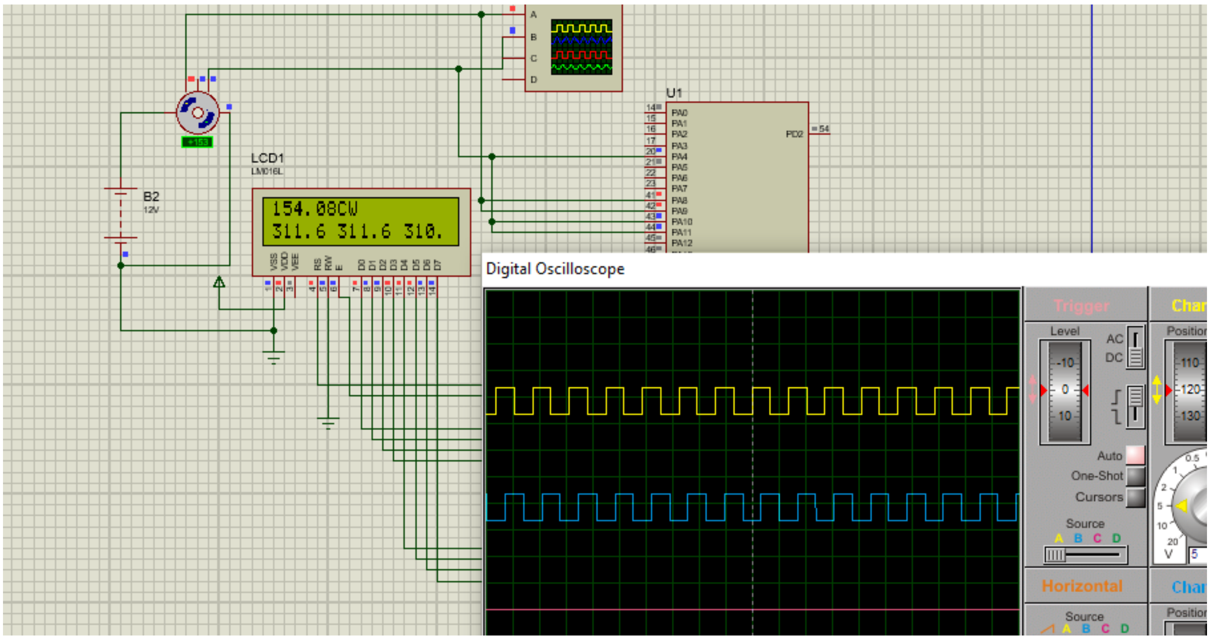
It is easy to show that if A was earlier than B, the rotation would be clockwise and if it was not earlier, then rotation will be counter clockwise like the figure bellow:



❖ Part E & F & G

After doing all that is wanted Result of our code is shown bellow:

From left to right in the second line, its mesure_1x, mesure_2x and mesure_4x.



❖ Part H & I

For measuring speed we have bellow equation:

$$RPM = \left(\frac{freq * 60.0}{287.0}\right)$$

For measuring angle we have these formulas:

Mesurement_1:

$$\frac{measurement_{x1} * 365.0}{287.0}$$

Has been counting only on 1 EXTERNAL INTERRUPTS in cycle.

Mesurement_2:

$$\frac{measurement_{x2} * 365.0}{287.0 * 2}$$

Has been counting only on 2 EXTERNAL INTERRUPTS in cycle.

Mesurement_4:

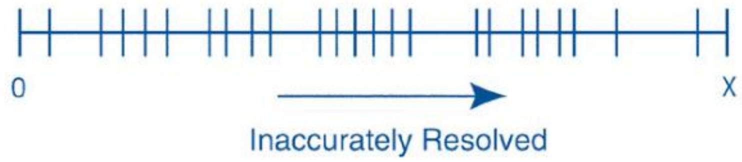
$$\frac{measurement_{x4} * 365.0}{287.0 * 4}$$

Has been counting on 4 EXTERNAL INTERRUPTS in cycle.

And based on the accuracy of measuring frequency, our accuracy for speed would differ.

One point to mention:

A common misunderstanding regarding encoders is to assume that a higher resolution improves the accuracy of the system as a whole. It is important to note that increasing resolution will not compensate for the latter type of error (see figure 1). Adding more pulses per revolution may improve the ability of the encoder to report position but if there is a systemic error, higher resolution will not correct it.



Based on sources, Encoder accuracy is a measure of the error between the value read out by the encoder and the actual physical value being measured. Encoder accuracy is measured in arcminutes or arcseconds with 20 arcminutes (0.3 degrees) or better generally considered a high accuracy encoder with some precision devices on the order of 5 arcseconds (0.0014 degrees).

For speed we got the accuracy now maximum and minimum amounts:

$$maximum: \frac{(frequency_{\max}\ \ ) * 60.0}{287.0}$$

$$minimum = \frac{freq\_\min\ \ * 60.0}{287.0}$$

For getting error in each measurements:

Mesurement_1:

$$\frac{0.3 * 365.0}{287.0} = 0.381$$

Has been counting only on 1 EXTERNAL INTERRUPTS in cycle.

Mesurement_2:

$$\frac{0.3 * 365.0}{287.0 * 2} = 0.190$$

Has been counting only on 2 EXTERNAL INTERRUPTS in cycle.

Mesurement_4:

$$\frac{0.3 * 365.0}{287.0 * 4} = 0.095$$

And we can totally see that our accuracy is increased in Mesurement_4.