Angle Inclination measure system

Abstract – In this paper we trying to make up a device based on 3.3V voltages which can measure the angle based on horizon in the range of -90 degree to 90 degree with some additional function. The input is determined by user, rotating the device horizontally and collect data by ADC (analog-digital conversion) is one of the efficient ways. In my system, I used "Esduino Xtreme" and an Accelerometer (the angle sensor) for Inclination Sensing (These two devices basically complete the function of angle measurement), a toggle switch to change modes of operation, a momentum switch to stop data acquisition, and 9 LEDs to represent the degree we got in different modes. For the testing step, I use my PC to interact with my device. Using "Matlab" to get plot and "Realterm" to test value. The data has been collected, proceed, and showed on the PC terminal. For the result, we got the correct output in both value and plot.

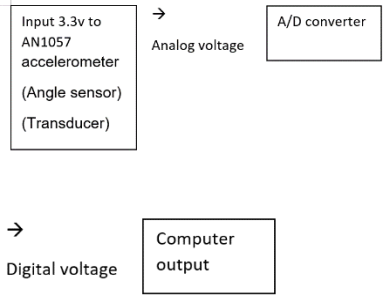
I: Introduction:

In the real world, there are many problems would be solved efficiently if we use sensor, angle analysis especially. Area like military, medical, industry and daily-life use would use angle analysis by ADC. In the real world, most of our signal used to interact with computer / machine. There are many different types of ADC we are using now, measurements of things like light, sounds, acceleration, speed, height, humidity etc. must be transferred to digital signal from analog signal in order to process them with computer. In this project, we are going to have the angle sensor using ADC which can be used in many areas, one of the example is the game consoles we use, it need to read the angle of its angle of inclination in some case, and our project have meet the requirements by doing small improvement on accuracy. There already has the kind of sensor as the solution of this problem, which has the similar function of our project and it is called the "G-sensor". It reads the angle ass input and give the programme in the computer as feedback to process. For the detailed example, please visit the website: https://blogofwishes.com/tag/g-sensor/. In this paper, we will introduce the system which can acquire angle data and process them,

record and display them on PC and LEDs using ESDX. The layout of the following section: section II will explain the whole system. Section III will let us check the outputs of some test cases. Section IV will do some discussion based on the project and in section V we are going to have the conclusion.

II. System Overview

In this section, I will introduce the complete function and design of my project. The whole process will be explained first.



We are going to have subsection talking about input calculation, ESDX setup, and output analysis.

A. the input calculation

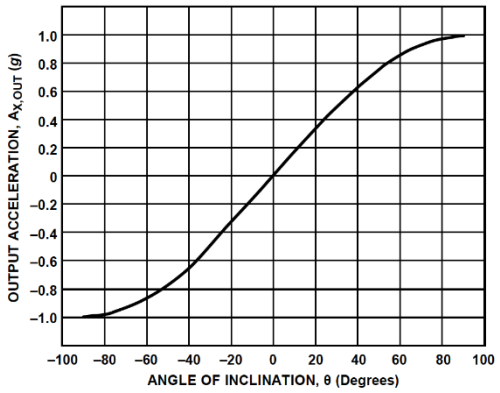
In order to measure the angle of inclination, we need to use the 3-Axis low power accelerometer AN1057 with 500Hz sampling rate, we only need to use one axis to do this (x axis). The input of it is 3.3 V_{PP} and the output is still analog signal depends on the angle of in inclination. The accelerometer will manipulate the input analog voltage and we use ESDX to complete the ADC part.

First, we need to deal with the requirements we have to set up as given. The bus clock I use is 6MHz, which means that I must change the default clock (6.25MHz). In order to change the configuration of the bus clock register, I must set CPMUPROT to 0 so that I can manipulate it with out protection. Next, I want to have a oscillator with 8MHz frequency, so I set PLLSEL =1 and OSCE=1 by setting CPMUCLKS and CPMUOSC registers to 0x80. According to the formula given for the CPMUREFDIV register, I set it to 0x41 to get 4MHz reference frequency. Then, it went to CPMUSYNR register, setting to 0x05 to get VCOCLK frequency = 2*4*(5+1) = 48MHZ. finally, we need to get PLL frequency by setting CPMUPOSTDIV to 0x03 to get 48 / (3+1) = 12MHz. So, the result would be 12 / 2 = 6MHz as we want. Last step is to wait PLL engage and enable the clock write protection (CPMUPROT = 1). So far, I had already set my bus speed to 6MHz. I also need an accurate delay loop to use. In order to d that, I implement my 6MHz bus speed.

Snice I have 6MHz bus frequency, I bus cycle takes 1 / 6*10^6 s. Assume that I want 1ms delay, then I need 1ms / (1 / 6*10^6) = 6000 add to the TCNT time register.

The we come to the calculation for the accelerometer. When the output signal from the accelerometer goes to ESDX, we need to have ADC convert in our ESDX. As we know the lowest voltage is 0V and the highest voltage is 3.3V for input, we have the range of 3.3V. Then we can come to the ADC formula: **$V_K = V_{RL}$ + (range * K) / ((2 ^ n) - 1)**. So, we can get the largest value is 2420 since we use 0+(3.3*2420) / ((2^12)-1) = 1.95V, and the smallest is 1675 since 0+(3.3*1675) / ((2^12)-1) = 1.35V. For the value of the middle point, we can also get 1.65V as 0+(3.3*2048) / ((2^12)-1) = 1.65V.

Then, we can see the relation of angle of inclination and output acceleration.



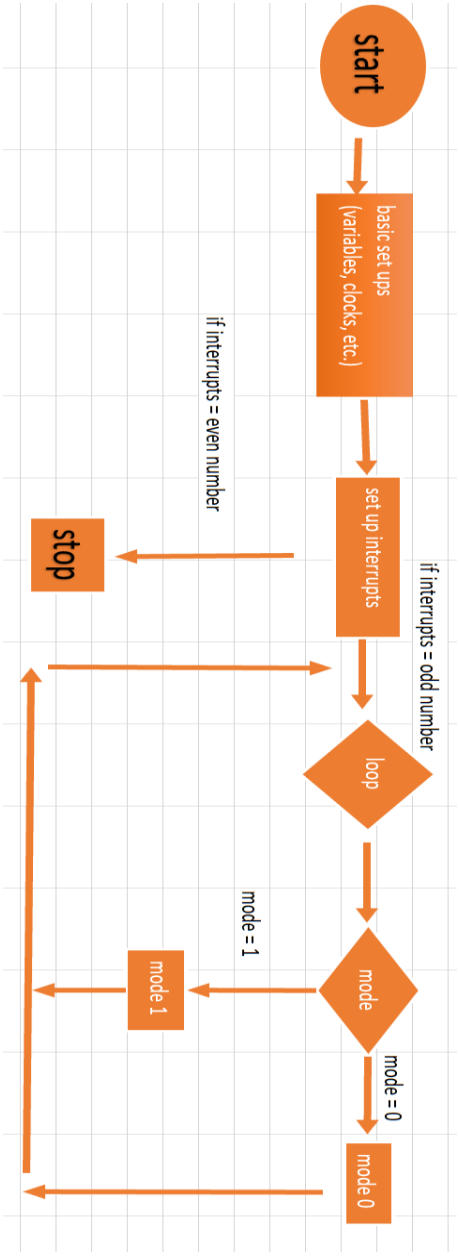
For the accelerometer, we use the successive approximation method, and there are some errors in later analysis.

B. ESDX settings

The accelerometer has already been introduced previously, then we need to know more about the ESDX settings.

- Bus speed: 6.25MHz (as default)
- 6MHz (after setting)
- Operating voltage: 5V or 3.3V, jumper-selectable.
- Memory resources: 240K Flash, 4K EEPROM, 11K RAM
- ADC: multi-channel 12-bit analog-to-digital conversion capability
- Cost: $90
- Language: C, Absolute assembly, and SBASIC

Then, we can come up with a flow chart of how our ESDX working for ADC and process outputs.



The major function of the ESDX in our project is Analog-Digital conversion (ADC). So, we need to set up the ADC registers.

From the ATDCTL1, we set it to 0x4F, so it set to be 12-bit resolution as required. To right justified, and has one sample per sequence, we set ATDCTL3 to 0x88. ATD clock need to be 1MHz, so ATD clock = 6MHz / 2 * (2 + 1) = 1MHZ by setting the prescaler register ATDCTL4 to 0x02. The last step, we need to set up the register ATDCTL5 to 0x24 in order to let ADC continuous conversion on channel 4 as AN4 channel needed to be used as required.

We have already gotten the range of voltages, [1.35, 1.95] represent -90 degree to 90 degree separately. In order to represent the minus sign in terms of degree, we could set the offset as 1.65V so that every voltage I could have below middle voltage (1.65V). We need to use the digital value to get the angle. We use the formula: analog= ((digital * 3.3/4095)-1.65)/0.3 to sin (angle), then use the arcsine equivalent approximation equation:

$$\arcsin(x) = x + \frac{1}{2}\frac{x^3}{3} + \frac{1\cdot 3}{2\cdot 4}\frac{x^5}{5} + \frac{1\cdot 3\cdot 5}{2\cdot 4\cdot 6}\frac{x^7}{7}$$

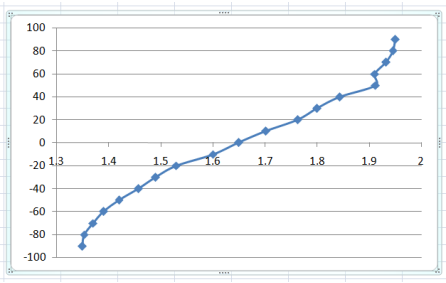
to get the angle.

By using these two equations, we could come up with a table just for some example ADC conversion.

Angle (deg)	Voltage (V)

-90	1.35
-80	1.353
-70	1.37
-60	1.391
-50	1.420
-40	1.457
-30	1.49
-20	1.53
-10	1.60
0	1.65
10	1.702
20	1.753
30	1.8
40	1.843
50	1.912
60	1.91
70	1.932
o	1.945
90	1.95

Then, we could plot this data out to check if it corresponds with the pervious figure.



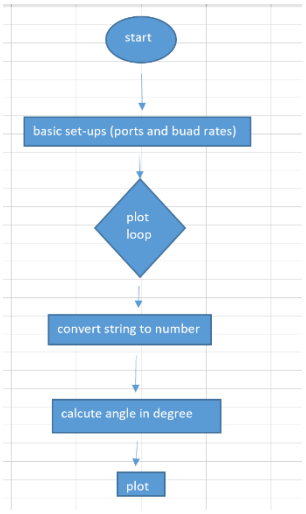
From the output graph, we can conclude that the output basically corresponds with the given figure.

C. Output analysis

After we introduce the major function of the system (ADC and accelerometer). We can come to the serial communication.

As we change the default bus clock 6.25MHz to 6MHz, we need to change the baud divisor in the “SCI.c” file using the formula: baud divisor = 6MHz / (16 * baud rate). So that the result in Realterm could be read properly.

For the communication with PC, I use software “Realterm” (Serial Capture Program 2.0.0.70) and “MATLAB” (R2016a) in 64-bit operating system. The flow chart of data communication with PC will be flowing as below.



I choose 9600 baud rates cause the error of 9600 is one of the smallest cases.

Baud rate	Baud divider (original)	Baud divider (final)	Error
2400	156.25	156	1.6026e-3
4800	78.125	78	1.6026e-3
9600	39.0625	39	1.6026e-3
19200	19.53125	20	0.0265625
38400	9.765625	10	0.0765625

We use the “device manager” to determine which port we need to use. We have only one port “COM1” available, so we need to justify this every time I need to use software to interact.

Realterm: In the serial communication, SCI_Init 9600) needed to be set in order to match with the sampling rate to get the value in Realterm.

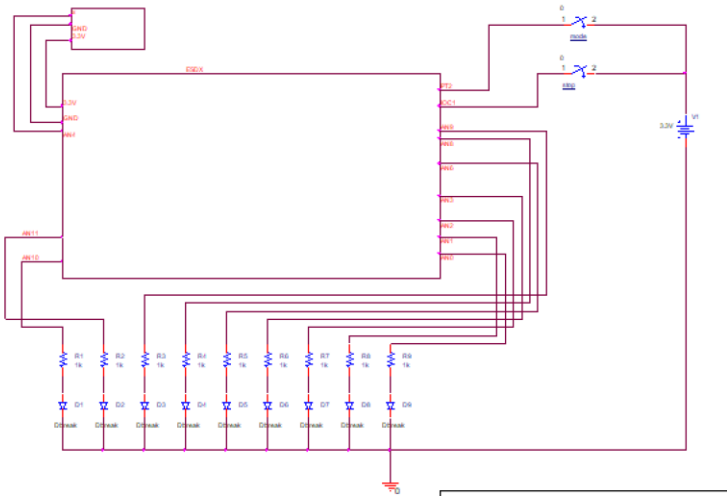
MATLAB: Same as the “Realterm” settings, we need to define the port “COM1” and baud rate (9600), storing the data into variables and plot the out versus time using equation.

III. Experiment and Results

For the Experiment and results, this section will show you the discussion of input, clock speed and delay loop, ADC system, serial communication, and the overview of the entire system.

A. Inputs

In this section, we can get the circuit schematic first.



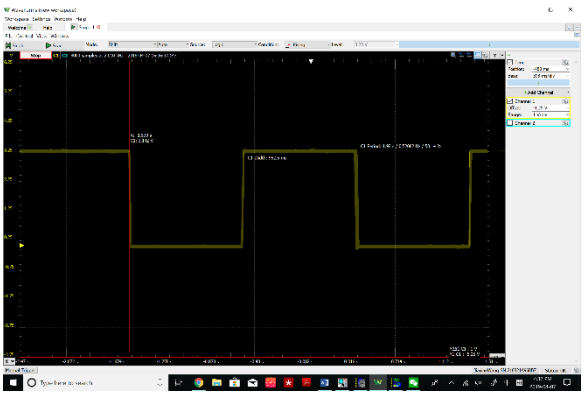
The pins we use for LED display are AN0,1,2,3,8,9,10,11. IOC1 for interrupt, PT2 for switching mode.

B. clock speed and delay loop

First, we would show the working voltage for accelerometer.

From the AD2, we can see that the voltage for working accelerometer is 3.3V.

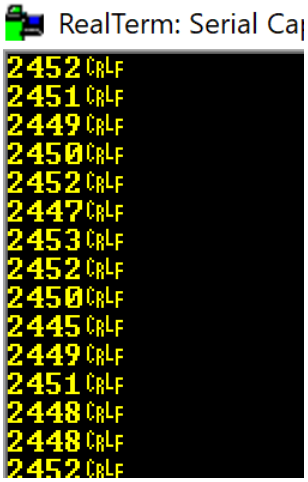
Then, we can check how our delay loop works, does it satisfy our requirement.



We can see the delay is 962.6ms (test DDRJ) if we want 1s delay. And 37.4ms error is acceptable.

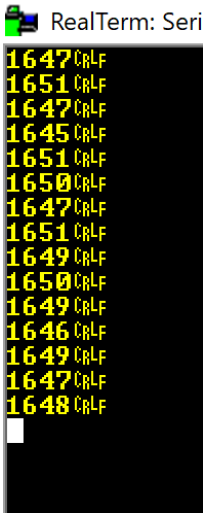
Then, we can verify our ADC value from Realterm software.

90-degree ADC conversion



From the ADC formula 0 + (3.3 * ADC) / (2^12 - 1) we can see that the result angles are slightly higher than 90 degree, but the error is acceptable.

-90-degree ADC conversion

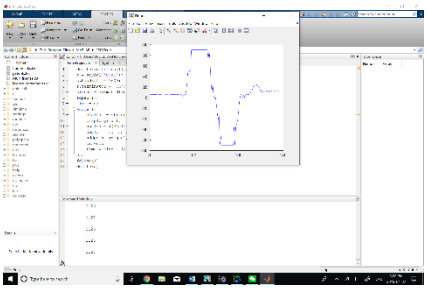


From the ADC formula 0 + (3.3 * ADC) / (2^12 - 1) we can see that the result angles are slightly smaller than -90 degree, but the error is acceptable.

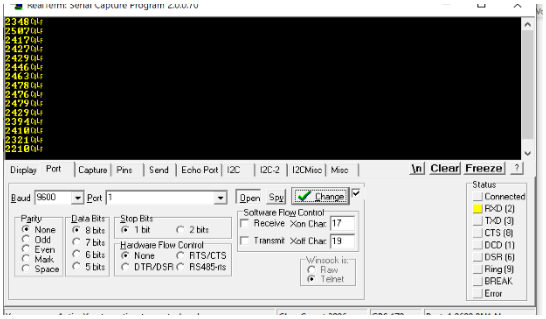
Then, we want to check the PC output plot on Matlab and check the graph.

C. Serial commutation

Matlab



Realterm



For the baud rate:

Equation:

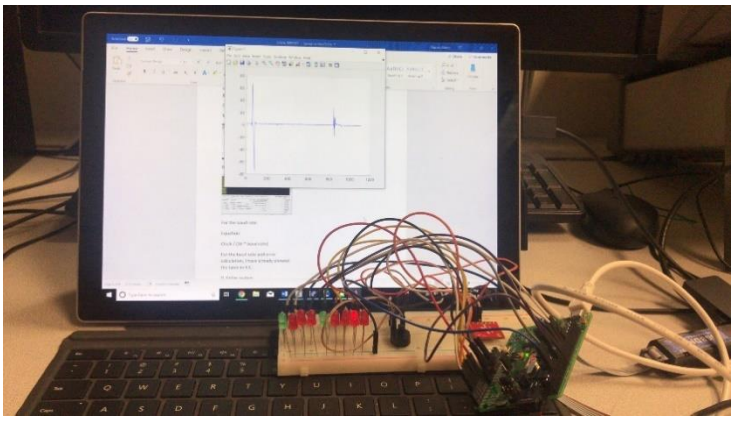
Clock / (16 * baud rate)

For the baud rate and error calculation, I have already showed the table in II.C.

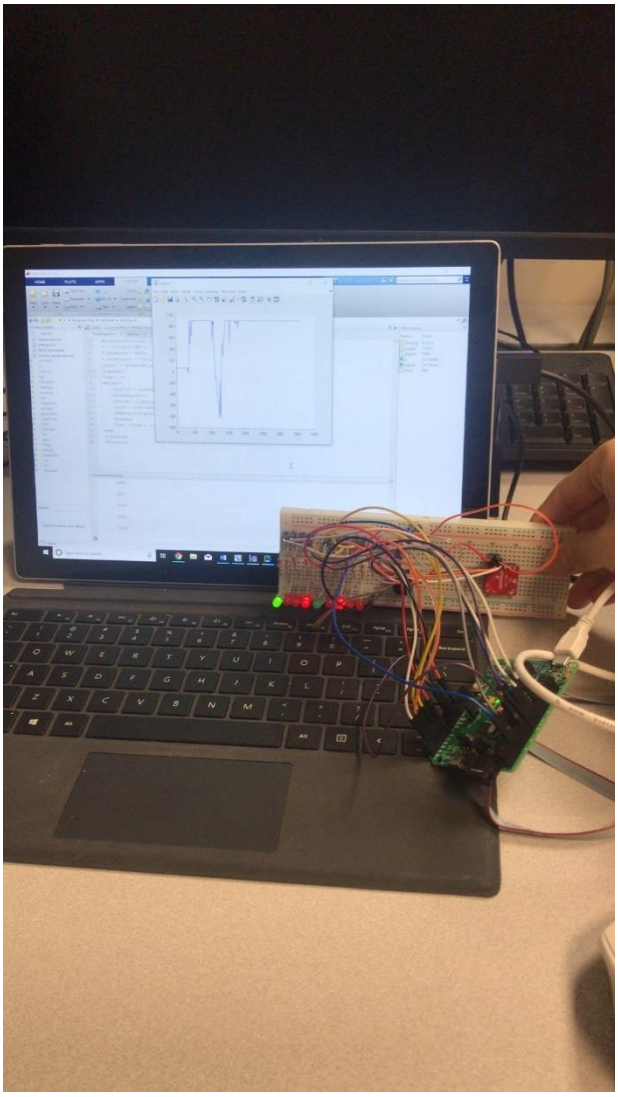
D. Entire system

Mode 0:

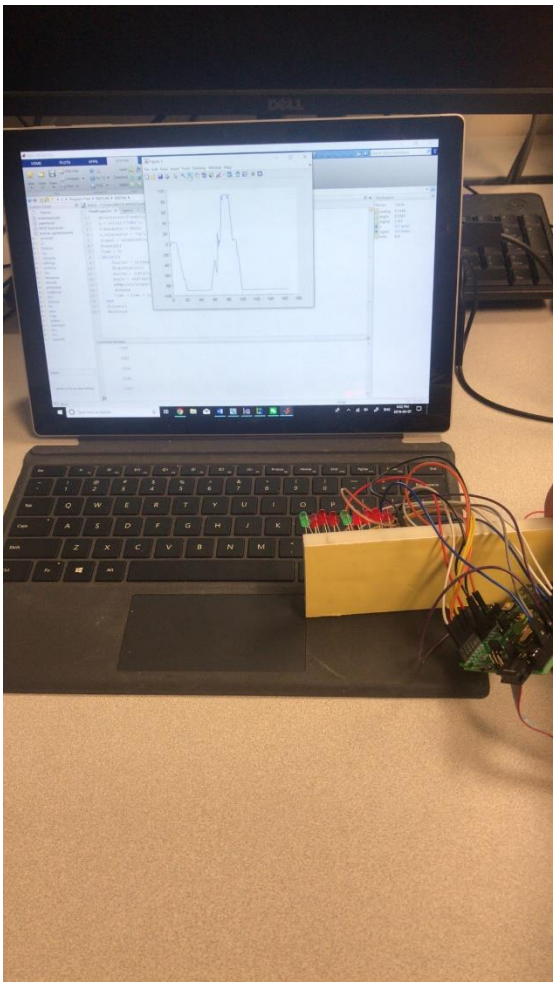
0 degree

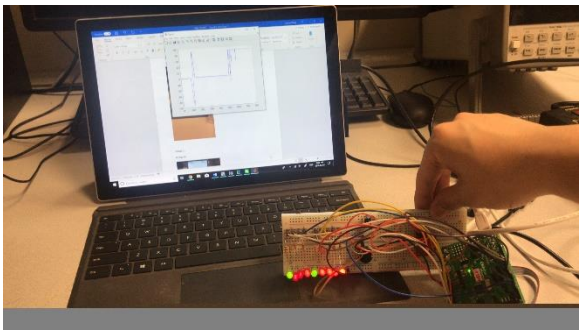


90 degree

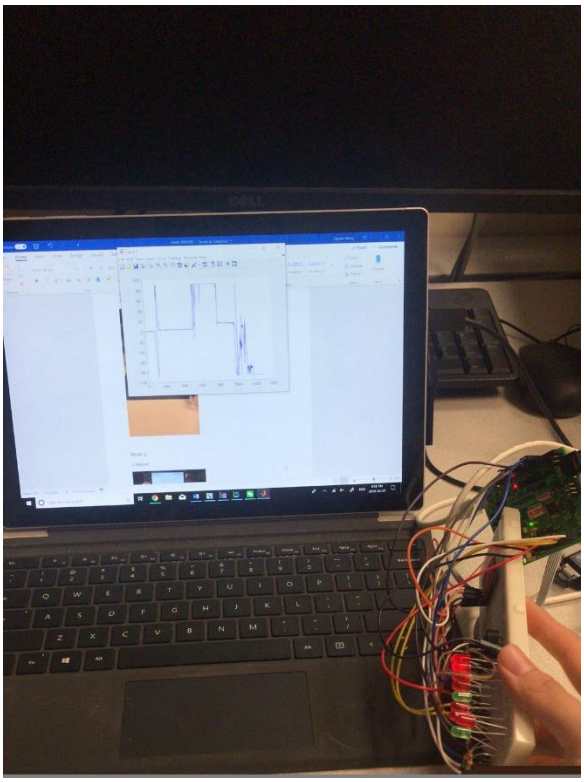


-90 degree





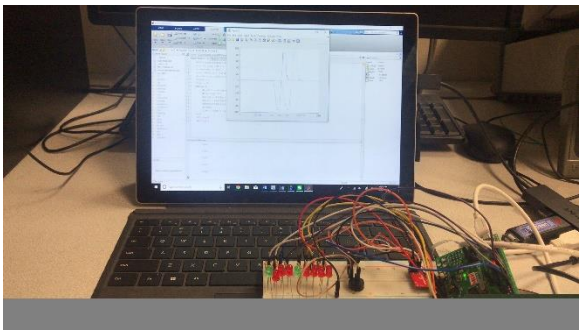
-90 degree



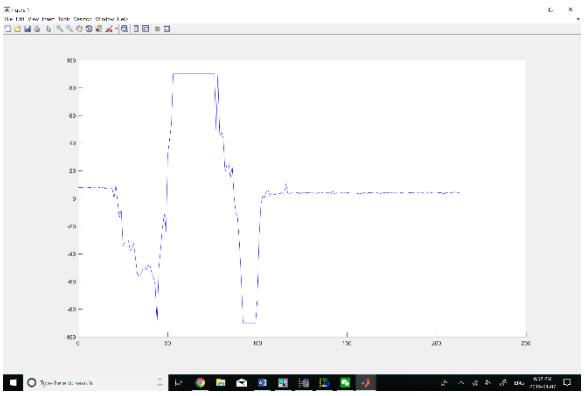
PC display:

Mode 1:

0 degree



90 degree



IV. Discussion

We have already checked each component of our project. There is some slight error, but we can still conclude that the output is correct, and error is acceptable. Since we use approximation to get the angle, the error of ADC conversion is unavoidable, so the result of ADC is not absolute accurate. In order to do some improvements, we might want to change the equation for approximation of the angle, so that we can get the more accurate result.

Q1: In order to solving that, we build our project base on the new codewarrior file which contains the function of IEEE32 double and small memory, then, we can use the arcsine approximation based on that.

Q2: Using the maximum quantization error: resolution = $V_{FS} / 2^m$. V_{FS} is 3.3V here and m is bits of resolution (12). So, the result is 8.06*10^ (-4).

Q3: The maximum communication speed of 6MHz should be 38400, and it can be checked using Realterm.

Q4: the primary limitation on baud rate is the baud frequency, because we calcite the baud rate based on the baud frequency.

Q5: sampling rate must be larger or equal to the twice of the input signal or it might cause the lost of data.

Q6: the analog input signal will not be reproduced accurately digitally. Shorter bandwidth might cause the lost of data and fail to capture the signal.

V. Conclusion

In this paper, we have already showed the function of our project, which is the ADC conversion and output on PC terminal.

In conclusion, the whole system works as we expected with some acceptable error, and still need to be improved.