EsduinoXtreme: A low cost data procurement, processing, and transmitting device

Harshdeep, Guraya, gurayah, 400060619, L02

Abstract— In this paper the potential for Technological Instrument's EsduinoXtreme to serve as a data acquisition, manipulation, and communication tool is explored. The microcontroller is at the center of the entire system, it receives analog signals from an accelerometer which it then converts into digital values. Custom calculations are then applied on the digital data to determine the angle of the accelerometer. This data is then serially communicated to a MATLAB program running on an external computer for real time display while also being presented as a binary coded decimal or a linear bar on 9 light emitting diodes (LEDs). The system was tested by changing the angle of accelerometer and examining the displayed output, the results indicated that EsduinoXtreme is a viable and low-cost data procurement, processing, and transmitting instrument.

I. Introduction and Background

DATA acquisition, manipulation and display play a crucial role in the functioning of many technological systems that appear in our everyday lives. Manipulating and display acquired data from physical phenomenon inherently requires that the system have a process to convert the data from analog values to digital values. Although there are many implementations of such a system, they are usually too complex, too expensive, or too limited in capability.

Motion detection is one of potential physical phenomenon that such a system could track. The most common device used to measure motion is an accelerometer which measures acceleration by functioning as a transducer. These acceleration values can then be used to determine a range of other movement parameters.

Products like the Fitbit's digital pedometers are examples of systems that implement the above-mentioned motion tracking systems. These devices utilize a 3-axis accelerator to track movement in the X,Y, and Z directions. The accelerometer converts the movement into analog voltages that can then be processed. An analog to digital convertor converts these analog values. This digital value is analyzed and the corresponding output is displayed on a mobile app via a Bluetooth module.

In this paper I will propose a relatively low-cost system for acquiring angle inclination data that will be displayed on light emitting diodes as a binary coded decimal or a linear bar while being communicated to a computer running a graphing program, all in real time.

In section 2, I will outline the system being used to perform the data acquisition and real time display. In section 3, I will review the experimental procedure and corresponding results. In section 4, I will discuss the performance of the system. In section 5, I will provide a conclusion.

Harshdeep Guraya is with McMaster University, Hamilton, ON L8S 4L8 Canada (e-mail: gurayah@ mcmaster.ca).

II. Design Methodology

The accelerometer is the transducer, converting the movement into an analog voltage value. This voltage is then read as an input by a microcontroller's analog to digital convertor. This convertor outputs a digital value to which custom algorithm will be applied to determine an angle value. This angle value will be serially transmitted to a computer running a MATLAB real time graphing program. Meanwhile other custom algorithms will also be applied to output the angle onto pins that feed light emitting diodes as either a binary coded decimal or a linear bar.

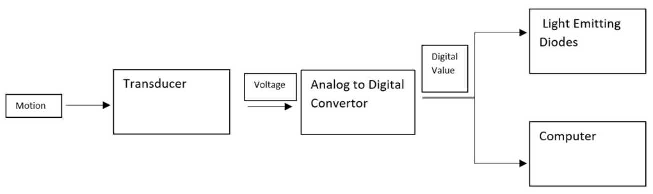


Fig. 1. System block diagram.

A. Pin Assignment Map

Pin	Assignment
A0: AN0	LED Bit 0
A1: AN1	LED Bit 1
A2: AN2	LED Bit 2
A3: AN3	LED Bit 3
A4: AN4	ADC Input
A5: AN5	LED Bit 4
DIG 2: PT0	Push Button 1
DIG 3: PT1	Push Button 2
DIG 4: AN 6	LED Bit 5
DIG 5: PP1	LED Bit 6
DIG 6: AN7	LED Bit 7
DIG 11: PP5	LED Bit 8
DIG 13: PJ0	Serial Communication Check

Fig. 2. Pin Assignment Chart.

The microcontroller chosen for the project is the 16-bit EsduinoXtreme developed by Technolgoical Art. It is part of the 9S12GA240 family. It has an operating voltage ranging between 3.3 to 5 V as such could power the 3.3 V accelerometer. The max bus speed of 24MHz was more than enough to reach the project specified bus speed of 4 MHz. The programmable ADC channel 4 (AN4) that can go up to 12-bit was sufficient to meet a 10-bit ADC resolution project specification. The microcontroller also had serial communicating capabilities which was essential to transfer the data to a computer for display. The microcontroller provided sufficient amount of input/output pins to interface with all the external hardware to build the full system. The microcontroller could also be programmed in C which was a known language. All in all, the microcontroller was able to meet all the project requirements while being relatively cost friendly retailing at only $69 USD.

B. *Quantifying Signal Properties*

The analog signal being produced by the accelerometer is a voltage. Using an oscilloscope, it was determined that this voltage value ranged from 1.6 to 1.9 V as the accelerometer was inclined at 0 to 90 degrees in the x-axis.

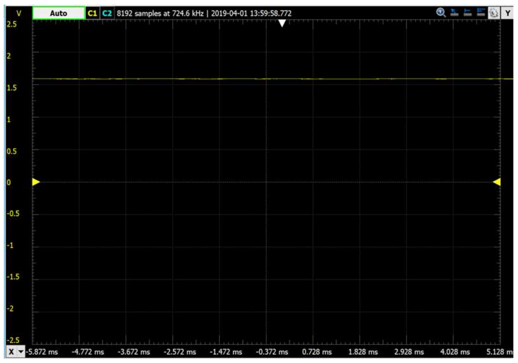


Fig. 3. Accelerometer output at an angle of 0 degrees.

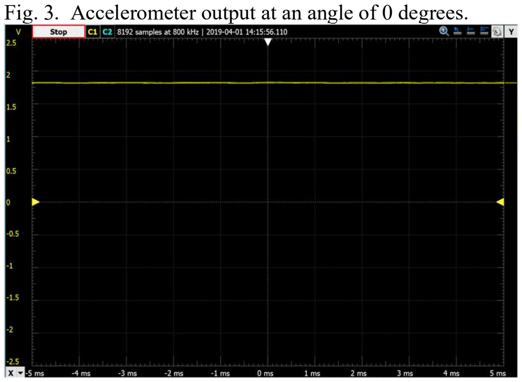


Fig. 4. Accelerometer output at an angle of 45 degrees.

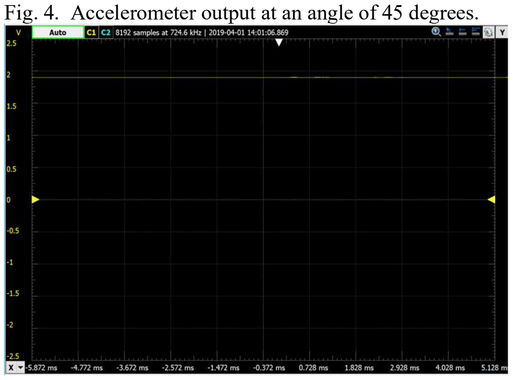


Fig. 5. Accelerometer output at an angle of 90 degrees.

The three angle positions (0,45,90 degrees) and their corresponding measured voltage values (1.6,1.8,1.9 V) did not showcase a linear relationship. However, using these points a linearly approximated transfer function was determined in excel.

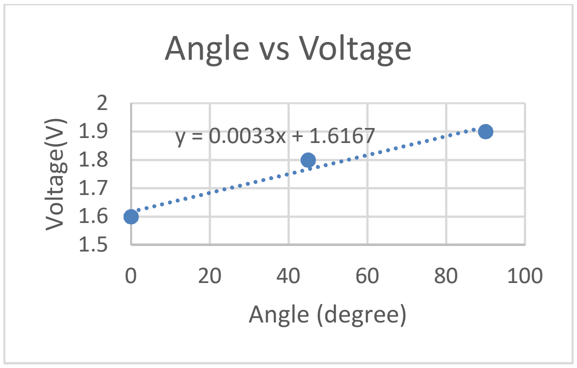


Fig. 6. Angle approximation based on accelerometer output voltage

C. *Transducer*

The exact transducer being used is the Analog Devices ADXL337. This is a 3.3 V device with a frequency of 500 Hz. Although the ADXL337 can measure acceleration in the X, Y, and Z axis, only data from the X axis is being sampled. As the acceleration increases so does the output signal. The device is capable of measuring within the range of ±3 g minimum. It contains a polysilicon surface micromachined sensor and signal conditioning circuitry to implement an open-loop acceleration measurement architecture.

The structure is suspended over the wafer using polysilicon springs, which provide resistance against accelerating forces. A differential capacitor with independent fixed plates and plates attached to the moving mass is utilized to measure deflection of the structure is measured. 180° out-of-phase square waves drive the fixed plates. Acceleration causes the moving mass to be and unbalances the differential capacitor. This creates a sensor output that is proportional in magnitude to acceleration. The magnitude and direction of the acceleration is determined using phase-sensitive demodulation techniques. The resulting output is brought out using a 32 kΩ after amplification. Due to addition of 0.01 uF built in capacitor at the X axis output, the signal bandwidth is set to 500 Hz.

This output voltage can then be translated into an angle based on the following relationship:

$$\theta = \frac{\arcsin\left(A_{X,OUT}[g]\right)}{1g} \times \frac{180}{\pi}$$

Upon graphing this relationship, it can be noted that the function is only fairly linear in the middle of the range. Linear approximation was used to convert the digital values to an angle as this approximation is a reasonably accurate method of determining the angle of inclination, however it is not perfect. The error is especially amplified at the extremes of the range where the relationship is the least linear and the lowest in the middle of the range where the relationship is fairly linear.

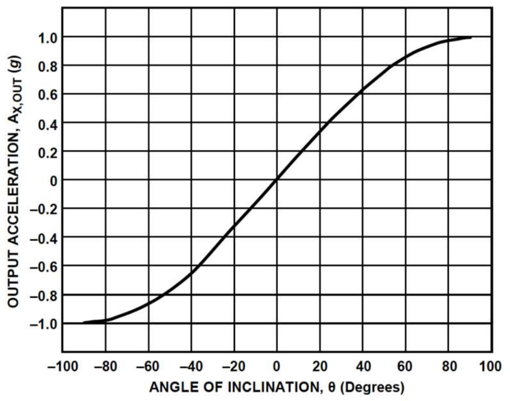


Fig. 7. Angle determination based on accelerometer output voltage

D. *Analog to Digital Conversion*

Analog to digital conversion involves quantization of continuous analog signal into a finite amount of digital values. In order to perform this conversion, the EsduinoXtreme microcontroller uses a technique known as successive approximation. This method first involves utilizing a sample and hold circuit to store the analog voltage. A digital to analog convertor is then used is then used to convert a digital guess into an analog voltage that can be entered into comparator along with the input voltage. If the guess was smaller than input voltage then the guess is multiplied by a factor of 1.5, if the guess was larger than the input voltage then the guess is multiplied by a factor of 0.5. This process repeats n amount of times n is the ADC resolution. Higher the resolution the more accurate the final guess. However, increasing the resolution will also increase the overhead and processing time.

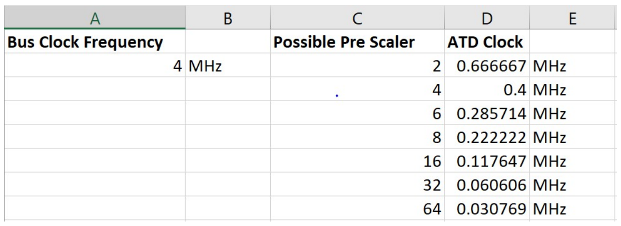
The highest possible input voltage into the ADC is 5V while the lowest input possible voltage into the ADC in 0V.

Voltage (Decimal)	**10 Bit ADC (Binary)**
0	00 0000 0000
1.25	01 0000 0000
2.50	10 0000 0000
3.75	11 0000 0000
5	11 1111 1111

Fig. 8. ADC conversion table.

The fastest ADC clock was chosen, this clock ran at 2MHz when the pre scaler was 0.

$$Fastest\ ADC = \frac{Bus\ Clock\ Frequency}{2 \times (prescaler + 1)} = 2\ MHz$$



A	B	C	D	E
Bus Clock Frequency		**Possible Pre Scaler**	**ATD Clock**	
4	MHz	2	0.666667	MHz
		4	0.4	MHz
		6	0.285714	MHz
		8	0.222222	MHz
		16	0.117647	MHz
		32	0.060606	MHz
		64	0.030769	MHz

Fig. 9. ADC clock calculations

According to Nyquist's sampling theorem sampling should occur at a rate at least twice the frequency of the analog signal in order to produce the best results.

$$f_{Nyquist} = 2 \times 500Hz = 1000\ Hz$$

$$T_{Nyquist} = \frac{1}{f_{Nyquist}} = 1\ ms$$

Based on 10-bit resolution, delay of 100 ms, baud rate 19200, serial communicating 3 characters (2 digits and a newline terminator), the Nyquist criteria is not being held as we sample at a frequency much lower than the minimum.

$$Transmission\ Rate = \frac{Resolution \times (\#\ of\ Transmission\ Bits\ )}{baud\ rate} = 1.5625\ \ ms$$

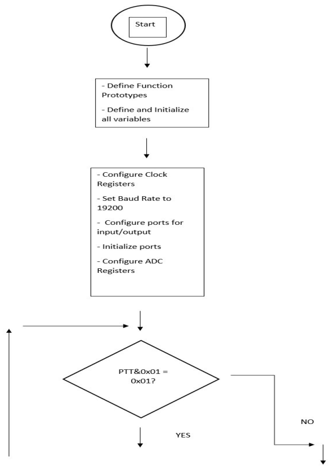
$$Conversion\ Time = \frac{(resolution + \#\ of\ programmed\ sample\ clocks + 2)}{ATD\ Clock} = \ 8\,us$$

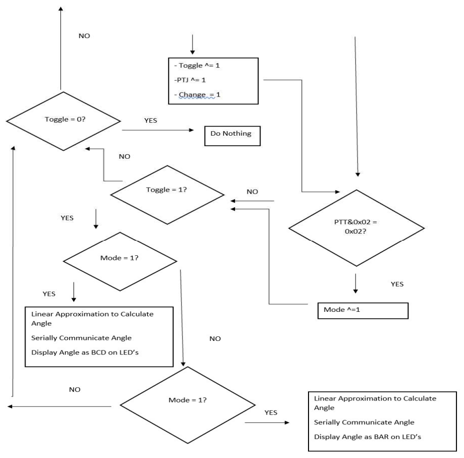
$$Sampling\ Time = 1.5625\,ms + \ 8\,us + 100\,ms = 0.1015705\ s$$

$$Sampling\ Frequency = \frac{1}{Sampling\ Rate} = 9.84537833\ \text{Hz}$$

E. *Led Display*

Custom algorithms were developed in order to display the calculated angle on either a linear bar or as a binary coded decimal based on the status of the variable mode (BCD = 0 and Linear Bar =1). This process only occurred based on the status of the variable toggle (Off = 0 and On = 1).


Start
- Define Function Prototypes
- Define and Initialize all variables
- Configure Clock Registers
- Set Baud Rate to 19200
- Configure ports for input/output
- Initialize ports
- Configure ADC Registers
PTT&0x01 = 0x01?
YES
NO



NO
- Toggle ^= 1
-PTJ ^= 1
- Change = 1
Toggle = 0?
YES
Do Nothing
NO
Toggle = 1?
NO
PTT&0x02 = 0x02?
YES
Mode = 1?
YES
YES
Mode ^=1
Linear Approximation to Calculate Angle
Serially Communicate Angle
Display Angle as BCD on LED's
NO
NO
Mode = 1?
YES
Linear Approximation to Calculate Angle
Serially Communicate Angle
Display Angle as BAR on LED's

Fig. 10. Flow chart for the Main program.

F. *Data Processing*

In order to convert the ADC output to an angle first a linear approximation model was built. This was done by recording the ADC output and several different angles measured using a protractor. These angles and corresponding ADC outputs were plotted in excel and a line of best fit was produced.

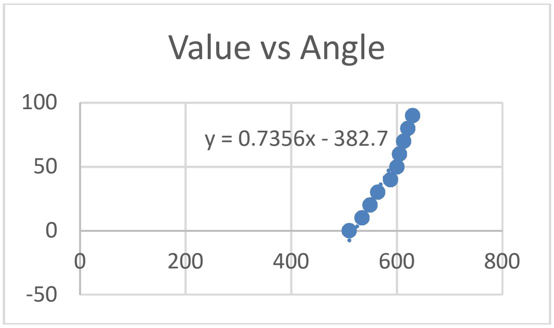


Fig. 11. Angle approximation based on ADC value.

G. *Control and Communication*

This model was then programmed into code warrior to convert the ADC outputs into angles. These angles could directly be serially communicated to an external computer running a MATLAB 2019 program. The computer utilized was a Microsoft Surface Pro running an Intel Core i7 – 7660U CPU @ 2.50 GHz. It holds 16 GB of RAM. Running a 64 bit version of windows 10. The data was serially communicated via COM Port 2 on the computer.

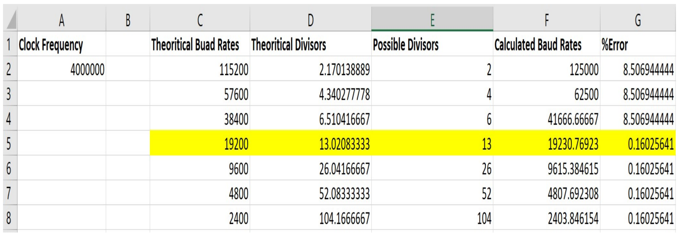
Excel was used to determine a baud divisor that corresponds highest possible baud rate possible without breaking the $\pm 6\%$ SCI timing tolerance.

$$Theoritical\ Divisor = \frac{Clock\ Frequency}{Theoritical\ Baud\ Rate}$$

$$Possible\ Divisor = Truncate\ (Theoritcal\ Divisor)$$

$$Calculated\ Baud\ Rate = \frac{Clock\ Frequency}{Possibel\ Divisors}$$

$$\%\ Error = \frac{Calculated\ Baud\ Rate - Theoritical\ Baud\ Rate}{Theoritcal\ \ Baud\ Rate}$$



	A	B	C	D	E	F	G
1	Clock Frequency		Theoritical Buad Rates	Theoritical Divisors	Possible Divisors	Calculated Baud Rates	%Error
2	4000000		115200	2.170138889	2	125000	8.506944444
3			57600	4.340277778	4	62500	8.506944444
4			38400	6.510416667	6	41666.66667	8.506944444
5			19200	13.02083333	13	19230.76923	0.16025641
6			9600	26.04166667	26	9615.384615	0.16025641
7			4800	52.08333333	52	4807.692308	0.16025641
8			2400	104.1666667	104	2403.846154	0.16025641

Fig. 12. Maximum standard serial communication calculation

Based on these calculations the highest possible baud rate of 19200 bps was utilized.

The angle data was transmitted at the 19200 baud rate as 2 digits followed by a newline character. The MATLAB 2019 program was set up to read from COM Port 2 at baud rate with the newline character being used as a terminator to determine when the next data point was encountered. A stopwatch was started, the variable x was initialized to 0, and an array for each axis was initialized before entering into an infinite loop. The loop stored each read angle as an integer and then updated x to the current value on the stop watch. The time value was appended to the x axis array while the angle value was appended to the y axis array. A plot was then created based on these arrays. This process went on forever, effectively updating the arrays with the current values and then plotting over the previous graph.

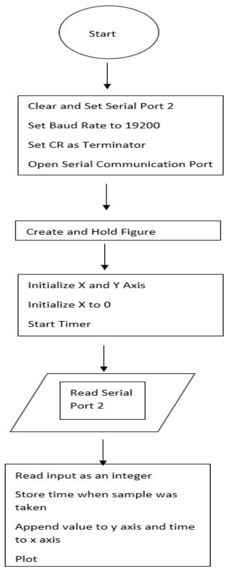


Fig. 13. Flow chart for the MATLAB 2019 program.

Meanwhile custom algorithms displayed the values to LED's.

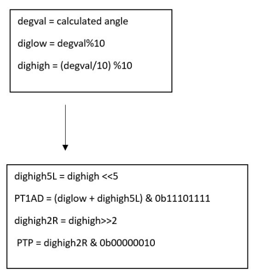

degval = calculated angle
diglow = degval%10
dighigh = (degval/10) %10
dighigh5L = dighigh <<5
PT1AD = (diglow + dighigh5L) & 0b11101111
dighigh2R = dighigh>>2
PTP = dighigh2R & 0b00000010

Fig. 14. Flow chart for the BCD program.

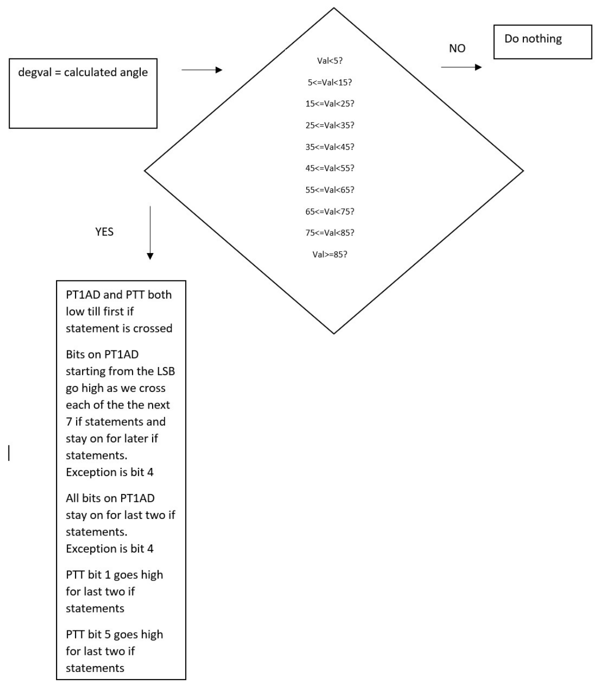
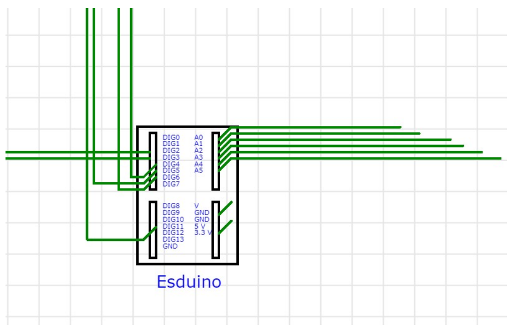

degval = calculated angle
Val<5?
5<=Val<15?
15<=Val<25?
25<=Val<35?
35<=Val<45?
45<=Val<55?
55<=Val<65?
65<=Val<75?
75<=Val<85?
Val>=85?
NO
Do nothing
YES
PT1AD and PTT both low till first if statement is crossed
Bits on PT1AD starting from the LSB go high as we cross each of the the next 7 if statements and stay on for later if statements. Exception is bit 4
All bits on PT1AD stay on for last two if statements. Exception is bit 4
PTT bit 1 goes high for last two if statements
PTT bit 5 goes high for last two if statements

Fig. 15. Flow chart for the linear bar program.

H. Full System Circuit Schematic



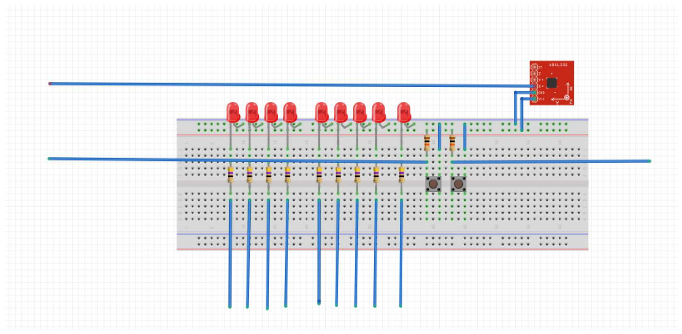


Fig. 16. Full system schematic and wiring

III. Experiment and Results

For each component of the system, a test was developed to validates its functionality. The system was then put together to determine whether the project criteria could be meet.

To test the functionality of the accelerometer, it was powered by 3.3 V DC source, grounded, and its x axis was connected to on oscilloscope. By examining figures 3 to figure 6, it can be noted that as the angle changed from 0 to 90 degrees the output voltage changed from 1.6 to 1.9 V in a nonlinear fashion.

A simple time delay function (delayms) is used throughout the main program. This function loops 4000 due to the 4 MHz bus speed clock specification.

$$Bus\ Clock\ Period = \frac{1}{Bus\ Clock\ Frequency} = 0.25\ us$$

$$Number\ of\ Iterations = \frac{Desired\ Delay}{Bus\ Clock\ Period} = 4000$$

This function inherently relies on another function (SetClk) which sets the clock registers to produce a 4 MHz bus clock speed using the PLL clock whose source is set to the Internal Reference (1MHz).

$$SYSCLK = PLLCLK = 2 \times Bus\ Clock = 8\ MHz$$

$$PLLCLK = 2\ \times OSCCLK\ \times \frac{SYNR + 1}{POSTDIV + 1}$$

$$4\ MHz = \frac{SYNR + 1}{POSTDIV + 1}$$

In order to satisfy this equation SYNR was chosen to be 15 and POSTDIV was chosen to be 3. In order to test if all of these functions worked as expected, a simple program was written that turned the onboard LED on then called delayms for 100 ms then turned the onboard LED off and called delayms for 100 ms gain in an infinite loop. Since the value of the onboard LED is also outputted to PTJ0, an oscilloscope measured this to validate that the delay functioned as expected.

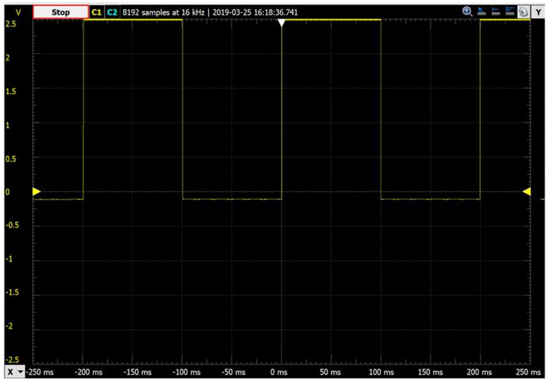


Fig. 17. Delay accuracy test result

To test the analog to digital convertor a 3.3 V peak to peak sinusoid function was inputted into the ADC and the corresponding values that were outputted into the Realterm serial communication program. Then these values were captured and exported to excel to be plotted in order to verify the ADC is capable of sampling the analog signal and recreating a digitized signal properly.

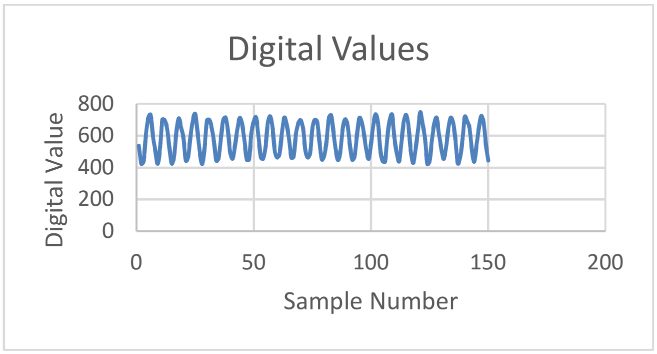
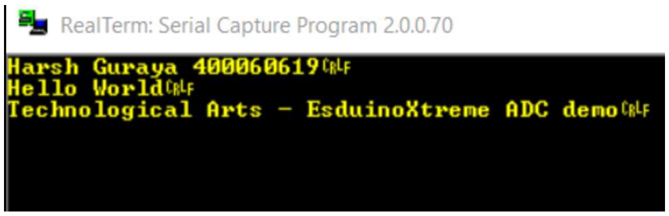


Fig. 18. ADC functionality test result

Based on a 10-bit resolution and a full-scale voltage of 3.3 V a max quantization error in converting from analog values to digital values.

$$Max\ Quantization\ Error = \frac{V_{FS}}{2^n} = \frac{3.3\ V}{2^{10}} = 3.22265625\ \text{ms}$$

To determine that serial communication works correctly, a string was outputted to Realterm.



```
Harsh Guraya 400060619CRLF
Hello WorldCRLF
Technological Arts - EsduinoXtreme ADC demoCRLF
```

Fig. 20. Serially communicating string to Realterm.

The entire system was then tested together

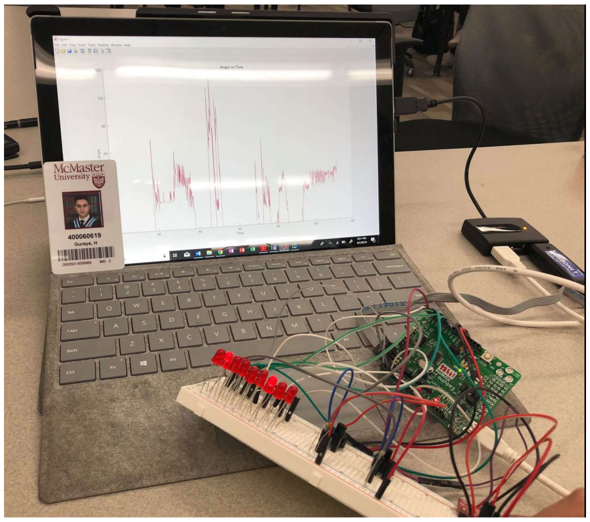


Fig. 21. Entire system working together, serial communication is on and BCD mode is selected.

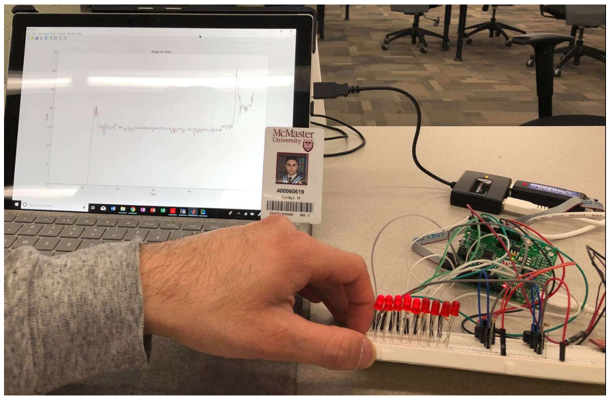


Fig. 22. Entire system working together, serial communication is on and linear bar mode is selected.

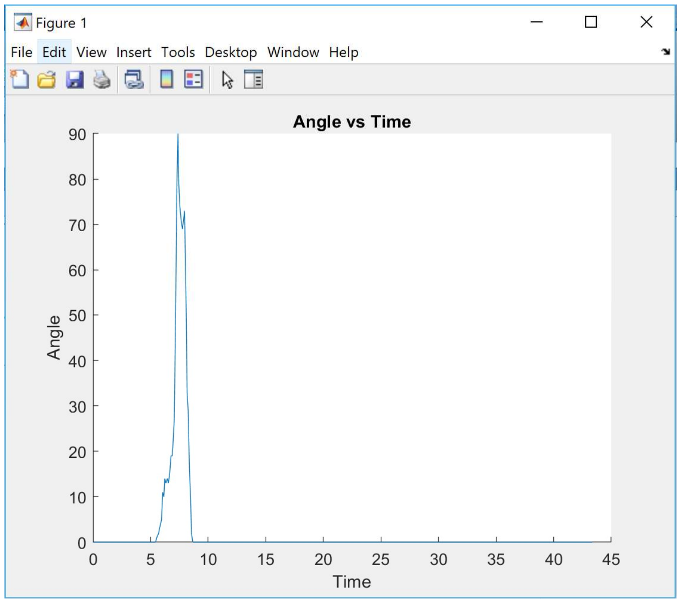


Fig. 23. 0 to 90 degree displayed on computer.

IV. Discussion

In analyzing the results from section III, it can be proven that the entire system meets the project specification criteria. The system was further examined to determine its limitations, accuracy, potential problems, and possible improvements.

1.The microcontroller does not have access to trigonometric functions, as such the formula to calculate the angle stated in section II part C could not be implemented directly. Instead, a linear approximation approach was utilized as described in section II part F. The function generated by this approximation includes floating point values which the microcontroller is uncapable of handling. To overcome this obstacle the function was scaled by a factor of 100 before being applied and then divided by a factor of 100 before being serially communicated. This reduces the amount of rounding error as opposed to rounding the original approximation itself.

2.The maximum quantization error as calculated in section III was 3.22265625 ms.

3.Using the process described in section II part G the maximum standard serial communication rate for a 4 MHz bus clock frequency that does not break the $\pm 6\%$ SCI timing tolerance is 19200 bps.

4.The primary limit on speed of the system is the serial communication rate. This rate is essentially dependent on the baud rate value. The baud rate value is based off of the bus clock speed which is the length of time each operation takes. Overall, the entire system would be faster if the bus clock speed was increased. This was verified by implementing the same system with smaller and larger baud rates than the current implementation. The higher the value the better the results

5.As described in Section II part D, the minimum sampling frequency is 1000 Hz according to Nyquist's theorem. However ideally, we would like to sample as 10 times this rate, leading to a ideal sampling rate of 10 000 Hz. However, based on the calculation provided in the same section and part we sampled at 9.8423 Hz.
Exceeding this maximum will result in the amplitude and signal shape preservation to be lost. Aliasing will occur if the sampling frequency is less than twice the input frequency.

6.Accurately reproducing analog signals with sharp transitions such as square or sawtooth functions into digital signals is possible with our current system. However, these signals can only be reproduced at high frequencies. This is because with slower sampling there is greater chance to miss the sharp transitions.

V. Conclusion

In this paper I successfully proved that a complete system for data acquisition, manipulation, and communication can be created using a low-cost microcontroller such as the EsduinoXtreme. Such a system could prove useful in designing devices that require this sort of data control but are able to sacrifice accuracy in order to save on cost. Based on my results generated by testing each crucial component of the project and the entire system as a whole, the system was able to meet the design criteria outlined by the project specifications, nevertheless was not a perfectly accurate system.

VI. References

[1] T.E. Doyle. (2018, January). Lecture slides (First edition). [Online]. Available: avenue.mcmaster.ca
[2] Fitbit Help. (2019). *How does my Fitbit device calculate my daily activity?*.[online]
Available at:https://help.fitbit.com/articles/en_US/Help_article/1141 [Accessed 8 Apr. 2019].