0. Abstract

In this report, we acquire data based off of various inclination angle measurement ranging from 0° to 90°. To process the data, an Esduino Xtreme embedded prototype is used to utilize the data read and convert into proper values and display. This system is designed to provide accurate angles on inclination of an object replacing the need for physical tools of measurement (e.g. protractor).

1. Introduction and background

The purpose of this project was to create an ESDX embedded prototype to acquire angle data, process the data, and transmit the data for display and recording. The project combines hardware and software applications such as Esduino Xterme, Analog Discovery 2, Code Warrior, and Matlab. The project utilizes the knowledge acquired in 2DP4 Microprocessors Systems to combine the various devices and software to achieve the desired outcome and product. This project gives exposure and develops knowledge in data acquisition systems while heavily emphasizing the use of analog to digital signal conversions (ADC).

Many devices we rely on in our everyday lives utilize ADC. Such devices would include cameras and microphones. In cameras, the light detected by sensors would be converted into a digital signal to compute its image on its internal computer. Similarly, a microphone =would convert audio levels to digital signals which a computer can then read and record.

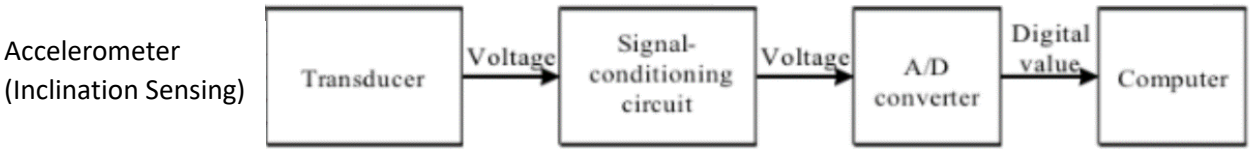
In this project, we are tasked with acquiring inclination angle measurement data (0° to 90°) and displaying that data in a legible manner to a high degree of accuracy. Without a physical tool for measuring angles like a protractor, the human eye can't accurately depict the angles of a physical object. This is where many that come across this problem rely on various devices to accurately measure these values for us. We're interested in this type of problem because our world is constantly shifting with new usage for data and this small project provides a strong stepping stone to lead many to develop various inventions and advancements. We will be interested in using the data received from the accelerometer for numerous custom algorithms to return/display desired data. Thus, our responsibility is to develop an ESDX embedded prototype with knowledge built around this course to achieve our goal.

The purpose of this paper is to discuss the conversion of analog signals to digital format using and ADC and various knowledge and components demonstrated with in the 2DP4 Microprocessors course. In section 2, we introduce our system and give an overview of what each part does. In section 3, we discuss the experiment and results. This includes testing and verifying the measured values to verify the accuracy of our measured values.

2. System Overview

In this section we discuss the different parts that make up the systems and the specific functionally of each component. Each component works in junction with each other to enable proper communication between the analog signals and the computer. These key components include a transducer, an optional

Figure 1: The ADC Conversion Process



signal-conditioning circuit, an A/D converter and a personal computer to read the data.

Referring to the diagram in figure 1, the transducers main functionality is to convert a specific type of analog signal into electrical signals. The signal from the transducer is then conditioned to be suitable for the ADC. The ADC then converts the analog signal to digital equivalent data where it can then be communicated and processed by a computer.

2.1. Transducer

The input for transducers can be a one of many different physical phenomena such as temperature, weight, light, and pressure. However, this projects input is the inclination of the accelerometer acting as the non electric quantity and the transducer respectively. Accelerometers are devices that measure acceleration (rate of change of velocity) in an object in meters per second squared. The acceleration is computed in reference to the zero g offset and with the inverse sine function (figure 2) to determine the sign and magnitude of the inclination.

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

Figure 2: Inverse sine function

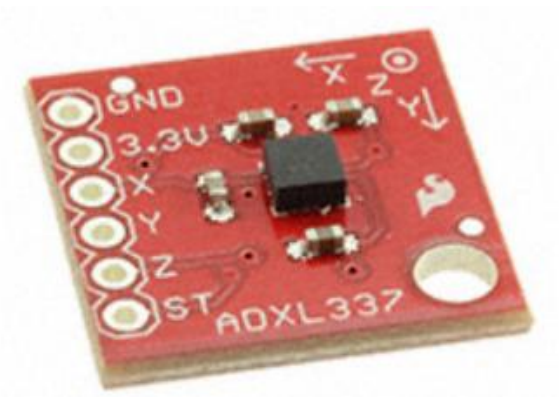


Figure 3: The ADXL337 Breakout Board

The angle θ can be then calculated in degrees with variables $A_{x, OUT}$ being the 1-dimensional acceleration output on the x-axis of the accelerometer and g being the ideal value of gravity.

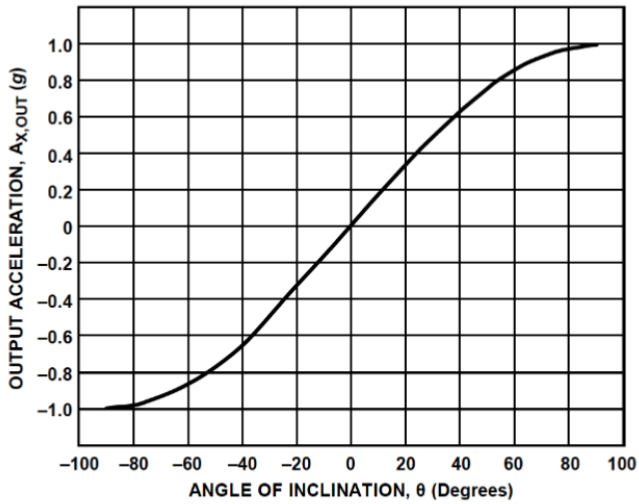


Figure 4: Output Acceleration vs. Angle of Inclination (single-axis) from ADXL337

The output of the accelerometer is not linear as it resembles the plot of a sine wave seen in figure 4. An output greater than the reference offset acceleration represents a positive acceleration and an output lower than the reference offset represents a negative acceleration. The acceleration data gathered can be computed into its respective degree output through implementing the inverse sine function shown in figure 2 or by using an approximation method. Such an approximation was found by writing down the values that the accelerometer would output (average) when it was resting at 0° and when the inclination was raised to 90°. From there we could calculate a slope given our two numbers and formula an equation that could represent the amount of inclination from the accelerometer to a high degree of accuracy. This method is referred to linear approximation and it a great way to avoid the fact that we could not use an inverse sine equation to calculate our angles.

2.2. Esduino Xtreme

The key component to this project would be the Esduino Xtreme. This microcontroller ($69 USD) powers all other devices its connected to and allows all the other components to communicate and work in conjunction with each other completing the desired task at hand. Knowledge from the Esduino analog to digital converter module allowed us to encode analog signal to digital, process the data and computer it to information of use, and return that data through various means that would be easily legible for anyone. This device is responsible for displaying our desired output data by using LEDs in binary-coded-decimal format and a linear bar format. Furthermore, we were able to serially transmit the waveform data to a PC and graphically

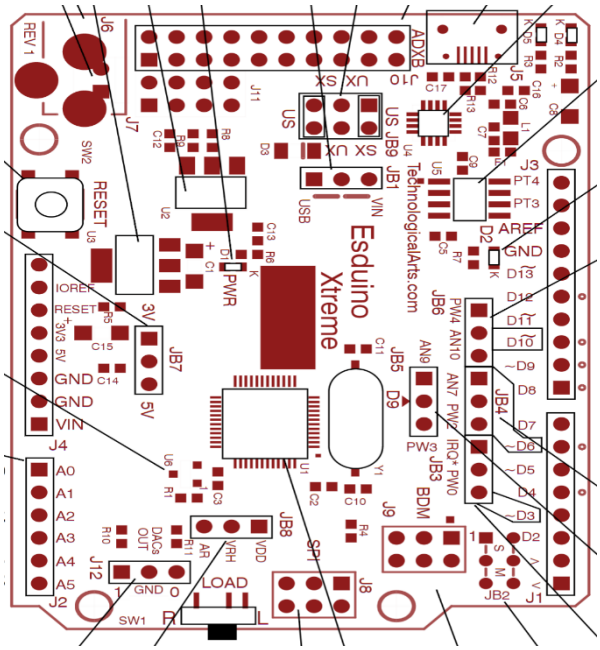


Figure 5: Esduino Xtreme

display the inclination and in the worm of a waveform on various applications (like Matlab). The Esduino Xtreme comes with a variety of onboard interfaces and features. This includes a 8MH crystal, providing a 4MHz nominal bus speed, memory resources such as 240k Flash 11k RAM and 4K EEPROM. Its typical operating voltage is 5V or 3.3V which could be physically adjusted with a jumper. Power is supplied to the Esduino through a micro USB by default, but it can also be powered externally with an optional input jack with the on-board 5V regulator and using a 6-12 VDC power source. The ESDX can utilize its 6-pin standard header for a use of a separate BDM pod for programming all Flash and EEPROM memories and debugging programs in real-time. The ESDX can communicate to a computer through its FTDI chip by utilizing a micro USB. There are several different languages that could be used for implementing and algorithm in the ESDX

	Language	Assembler	IDE	Terminal Program	BDM Com	Other
CodeWarrior Special	C	S12	x		x	free 32K limit for C unlimited assembler
CodeWarrior Pro	C	S12	x		x	commercial product
HSW12	Assembler	S12	x		x	Linux
AsmIDE	Assembler	S12	x	x		Windows
MiniIDE	Assembler	asm12	x	x		Windows
SBASIC	BASIC	as12				DOS-based
GCC-Syncode	C	as12				Windows
GCC-Eclipse	C	as12				Windows
Imagecraft C	C	as12	x	x	some	commercial product
Cosmic C	C	S12X	x		x	commercial product

Figure 6: Development platforms available for ESDX

such as C and Assembler with popular platforms consisting of Code warrior Special or CodeWarrior Pro. Other platforms are listed below in figure 6.

2.3. Esduino: Program

We were given many specifications that needed to be followed with executing this project all based on our student number. The ADC channel that we were assigned to was AN5 with other requirements including the implementation of the project in a resolution of 12 bits and a bus speed of 8 MHz. We used these given

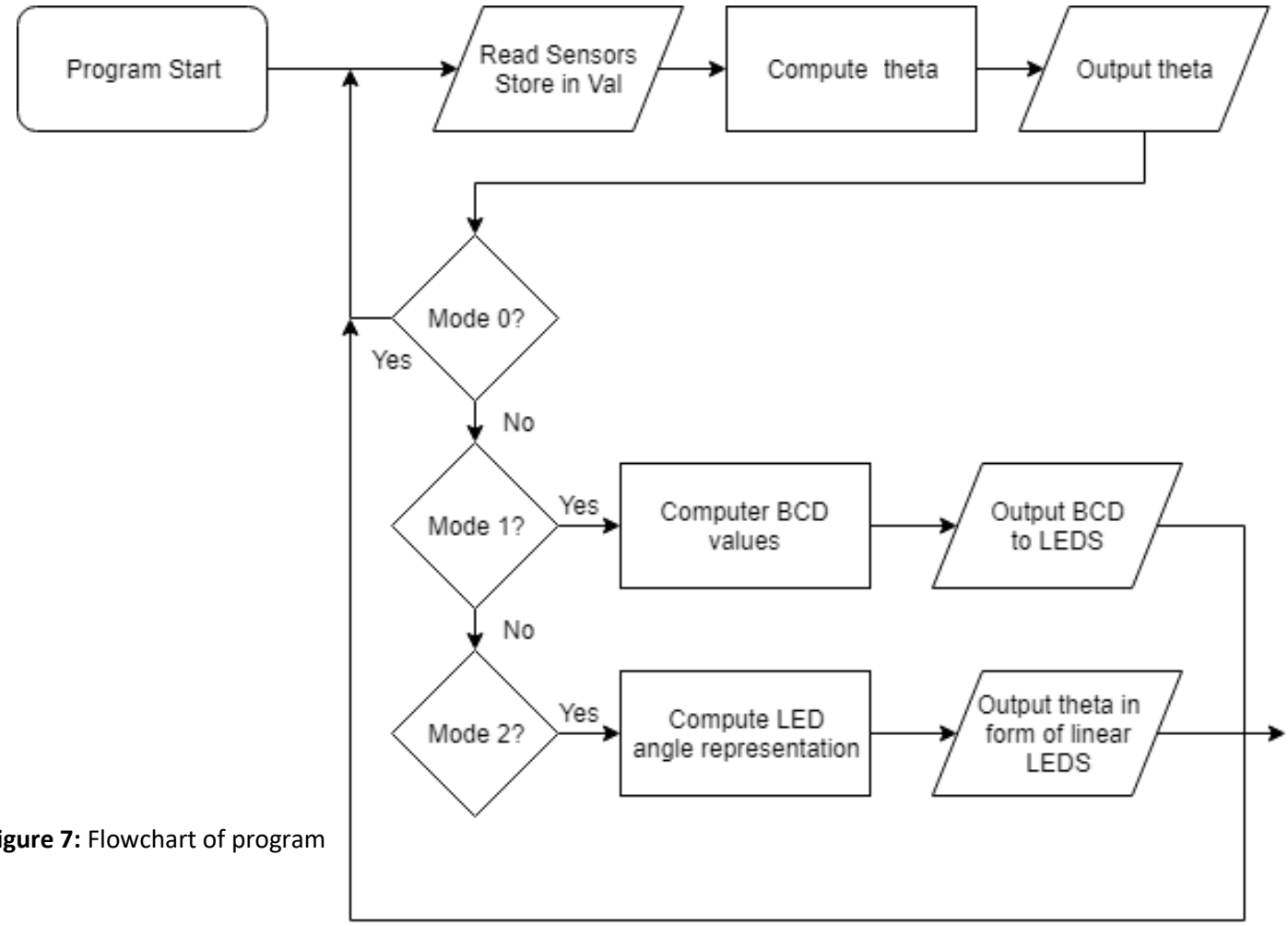
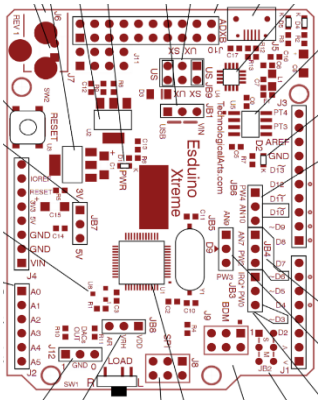


Figure 7: Flowchart of program

specifications to calculate our baud rate with the following formula: Baud Divisor = Bus Clock/(16*Baud Rate). We had to ensure that we were following the SCI timing tolerance with is typically =/- 6% error with selecting our baud rate and thus we determined that the specific baud rate that would be used would be 19200. The value met our requirements enabling us to have proper communication and calculation of data. The pins we are using for the program are shown in figure 8 as they provide functionality to components such as the accelerometer, LEDS, and buttons. Our linear approximation gave us an equation of: theta = 9*val/28 – 425 with the variable val representing the ADC values. The programs main component is the on going loop that will have three major if-statements/states mode 0, 1, and 2. Mode 0 acts as the off switch/pause where the system will not output any data back to the user, mode 1 will output the data in terms of its BCD equivalent to 8 LEDs, and mode 2 will output



GND – Breadboard terminal

GND – Button Lead

D2 – Button Lead

3.3v – 3.3 accelerometer lead

A5 – X pin on accelerometer

A0 – A4 – LEDS

D8 – D10 – LEDS

D12 - LEDS

Figure 8: Pin placements

the data in a linear angle representation with the 9 LEDs (each LED representing 10° of inclination). Within each mode, specific calculations and pin assignments are made to output its distinct output that would make it different from the other modes. Each mode can be switched through with the use of an interrupt function and buttons. Each time the interrupt was called, the program would traverse through the different modes. The only input this program takes are that of the single button and the data sent from the accelerometer and this leads to the output that corresponds to what ever mode the program is currently in.

2.4. Computer

The project was done using the CodeWarrior IDE and was ran on a Del XPS running Windows 10 OS. The specs include Intel® Core™ i7-7560 CPU at 2.40GHz, 16 GB RAM, 256GB of SSD storage as well with integrated graphics. The data from the ESDX was transferred serially to the PC with a micro USB connector enabling the data to be displayed on Realterm 12C.

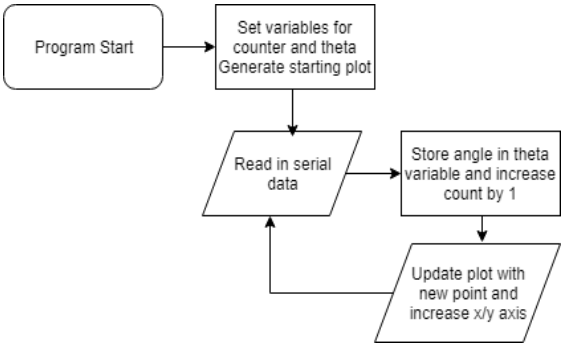


Figure 9: Matlab

For out computer and its program, the only tool we used was Matlab (2016a). Our data display program is coded to automatically read in the angular data and plot with no user intervention. The plot will show both axis (time and theta) where there is a constant scale of 0° to 90° and the time axis will constantly expand as time goes by while the program is running. This plotting program will this generate an updating plot that can be used in conjunction with our Esduino as another form of displaying data for experiment. The functionality and process of the Matlab code can be seen in figure 9 mainly consisting of one loop constantly taking inputs and outputs.

Experiment and Results

In this section we detail our experiment and the results of the project as discussed earlier. Furthermore, the specifications of each component and student specific requirements are detailed and verified to ensure they all meet the standard.

3.1. Input

Our input was tested show in figure 10 through hooking up our device to the AD2 and using its onboard oscilloscope to test and checked if we were getting any voltage from the accelerometer to verify it was working.

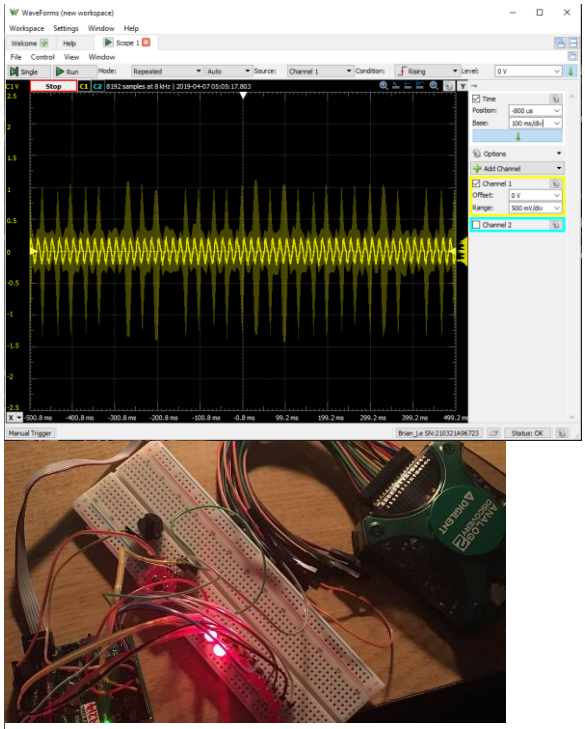


Figure 10: Input test using oscilloscope

3.2. Clock Speed and Delay Loop

The following registers were used to set up the proper clock speed for the ESDX: CPMUPROT, CPMUCLKS, CPMUOSC, CPMUREFDIV, CPMUPOSTDIV, and CPMUSYNR. We verified the proper clock speed of 8 MHz by setting up a time delay loop and LEDS on the ESDX. The clock speed is verified through the use of an oscilloscope rigged up with the device with the data being displayed on the plot helping us visualize and confirm the duration at which the LEDs are toggled On or Off.

3.3. ADC System

Another student specific requirement was to set the ADC Channel and resolution. The selected channel for this project was AN5, with a 12-bit resolution. The ADC Channel AN5 was to be used with continuous conversion using 12-bits while following ADC registers were configured to get the required outcomes.

ADC Register	Set Value	Description
ATDCTL1	0x46	Set resolution of 12 bts
ATDCTL3	0x88	Right justified
ATDCTL4	0x02	Code from Lab4
ATDCTL5	0x25	Set ADC Channel AN5

Similarly, a button interrupt was used to toggle through the different operational states of the system. The following interrupt registers were configured to get the desired outcome:

Interrupt Register	Set value
TSCR1	0x90
TSCR2	0x00
TI0S	0xFE
TCTL3	0x00
TCTL4	0x02
TIE	0x02

3.4. Serial Communication

As we previously stated in section 2.3, we had calculated our baud rate with the formula Baud Divisor = Bus Clock/(16*Baud Rate). Additionally, mentioned earlier was the fact that SCI's timing tolerance is typically +/- 6% otherwise the serial communication would not function properly, and data would not be correctly transferred. The following calculations were performed to get our desired baud rate with a bus speed of 8 MHz:

Baud Divisor = (8x10^6)/(16*19200) = 20.04

Divisor of 26:

Baud Rate = [(8x10^6)/26]/16 = 19230.77

Error:

[(19230 – 19200)/19200] * 100 = 0.16% error

Through these calculations, we can verify that the SCI timing tolerance is met, and the serial communication is correct as shown in figure 11.

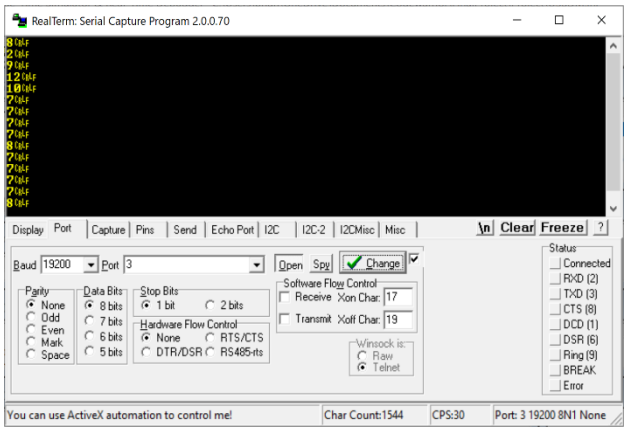


Figure 11: Serial communication output in RealTerm

3.5. Entire System

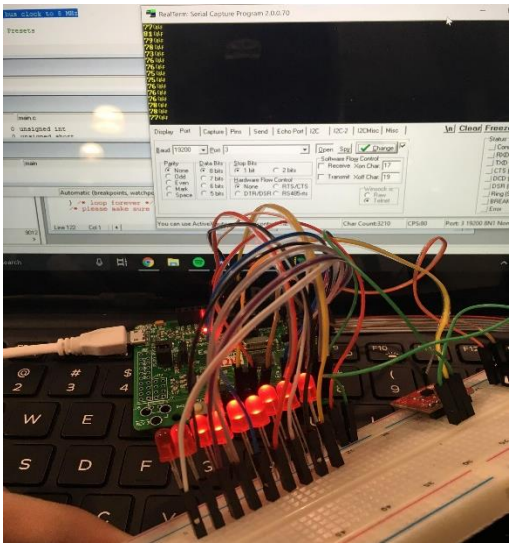


Figure 11: Entire device

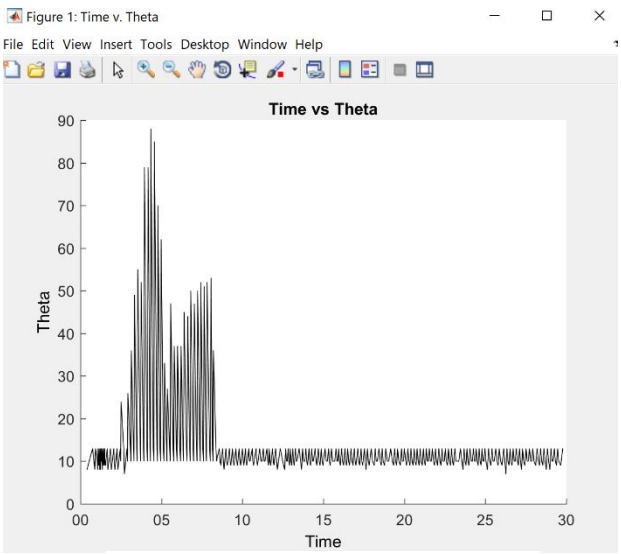


Figure 14: PC display screenshot

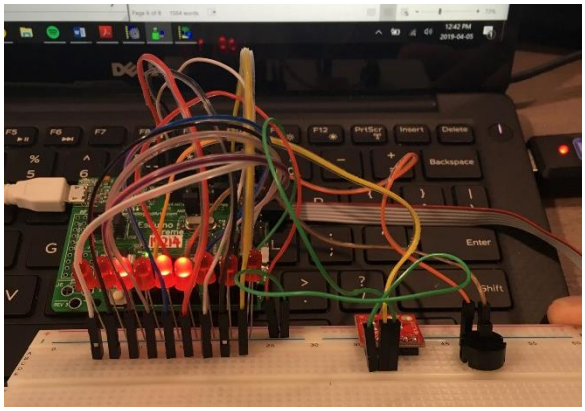


Figure 12: Mode 1 (BCD LEDs)

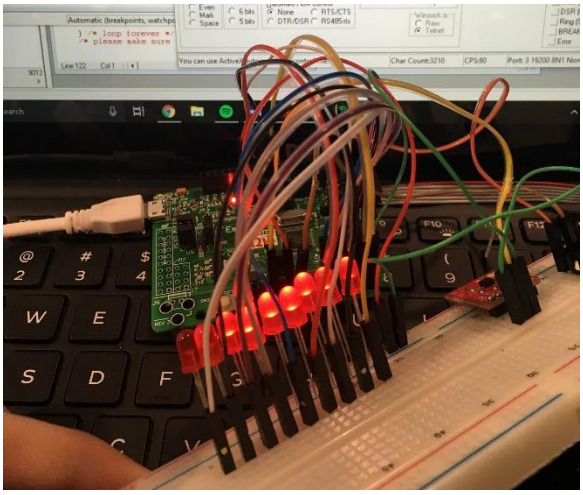


Figure 13: Mode 1 (Linear LEDs)

Discussion

Although we have validated that each component works the real features that we look at that can truly verify that the reproduced signals are correct is physically testing the device. Our linear approximation is accurate to a certain extent however, a minor inconsistency in the output come from the surface at which it is tested on and human error in handling and tilting the device for measurements. Collectively, the reproduced signal is correct with the only limitations being that we only receive data from 1 axis of rotation and the introduction of another axis would result in adjustments to the algorithms and calculations made in the program. The only problem we ran into when testing this device was the physical properties of our build seeing that the wires/jumpers would make it difficult sometimes to rotate the accelerometer. Some recommendations we have reflected upon would be to include another axis of rotation as input to enable angle to be displayed no matter how the device is rotated and tested.

1) We were able to overcome the ESDX not having floating point capabilities and not having

trigonometric functions through the implementation of linear approximation. We calculated a slope given the points that the accelerometer would output when its resting at 0° and tilted to 90°. From there we reused one of the points in junction with our newly found slope to create an equation that would return a proper angle based from the input of the accelerometer. This way, we are getting results accurate enough without the need of floating-point capabilities and not having trig functions.

2) The maximum quantization error is: Q = VFS/(2^m). Therefore, 5/(2^12) = 1.2207E-3

3) Maximum serial communication rate implemented was a baud rate of 19200 bits per second. This value was calculated with the assigned E Clock frequency of 8MHz and was determined (alone with percent error) using these equations:

Baud Divisor = Bus Clock / (16*baud rate)

Percent error = (Theoretical Baud Divisor – Actual Baud Divisor)/Actual Baud Divisor)*100

This was easily verified by altering the baud rate so that the percent error was greater than 6% and we could then see incorrect data displayed on ReamTerms output and when it was changed back to 19200, the data would be correct and legible.

4) Reviewing the entire system, the baud rate is the primary limitation on the sped of the system. The baud rate is the rate in which information is transmitted through a communication channel. Therefore, by increasing the baud rate, it would in-turn increase the speed in which information is being transmitted. This was proven by plotting the data transmitted into Matlab and comparing different baud rates. A low baud rate would update the graph at a slower rate compared to a higher baud rate. A fairly high baud rate would update the graph with virtually no latency or lag.

5) Based on the Nyquist Rate, the sampling rate must be twice the highest analog frequency in order to reproduce values effectively and efficiently. When your input signal exceeds this frequency would your data would give inaccurate outputs.

6) Signals with sharp transitions can be accurately reproduced at low frequencies. When the frequency is small enough, the sampling frequency is fast enough to read the sharp drops. When the frequency is high, the sampling frequency may not be fast enough to read such a sharp change in the signal.

Conclusion

In this paper, we demonstrated the basic steps required for proper data acquisition with the set components. The project successfully acquired the inclination data of an accelerometer, processed the data, and transmitted the data to a user through LEDS in a BCD format or linear format and in Matlab. This project combined the Esduino Xtreme, Analog Discovery Kit, CodeWarrior and Matlab in order to achieve a fully functional device and desired outcome. The benefits of this device compared to others with similar functionality are efficiency, portability, implementation, and ease of use as it plays a strong stepping stone for future developments in this branch of technological advancements. Performing a simple yet key action like measuring an angle is important for future work as it can easily translate to large scale problems like airplanes or other vehicles where all the pre-stated benefits play a key factor. The final three modes, Realterm and Matlab displays are a result of proper data acquisition and computer engineering execution. The finished product is an angle measuring device that can be implemented in numerous of different products

and scenarios. Thus, we were able to verify that an ESDX embedded prototype was properly created to acquire angle data, process the data, and transmit the data for display and recording.