Data Acquisition Systems: The Revolutionary Method of Processing & Analyzing Data

Maaz, Azam, azamm3, 400069421, L06

***Abstract* – In this paper, a data-acquisition system was implemented in order to obtain data from the accelerometer, that outputs a voltage reading in reference to the gravitational force on earth, and from here obtain the angle of inclination in reference to the gravitational surface. This was done through converting the analog signal reading into a digital signal that can be processed by the PC from the Esduino, and finally be graphically represented through transmitting this data via serial communication. This paper goes through the certain steps in order to implement this system with high accuracy. In addition, components included in this system setup will be validated separately in order to ensure that the graphically reproduced signal seen through serial communication is accurate and correct. Finally, other applications of this system and improvements will be taken into consideration and further discussed.**

I. Introduction and Background

In this society, technology is a part of our everyday lives, and the role and impact of technology continues to grow as humans become more dependent on technology throughout their daily routines. One of the most astounding inventions of the technology era however was the computer. Many people use computers for several applications, such as obtaining information from the web. In the past, acquiring data was done through a series of calculations done by hand. In addition, some applications required people to use alternate materials such as strings, papers, and sticks and make inferences on their data based on the movement or positioning of these materials.

With the invention of the computer, acquiring data from experiments, sensors, and many other applications is not done by hand anymore. Instead, the computer is used for data acquisition processing from where analog signals or information from the real-world is converted into quantitative data that the computer outputs to the user. Therefore, technology has significantly helped in areas such as research and development as now, we can obtain instant data and easily manipulate or perform further analysis.

Internally, the process of converting the analog signal to a digital signal through an analog-to-digital converter is more complicated and can be prone to several errors or deviations. This is due to the computer itself not acquiring the data, and simply acting as a means to process the data as the computer acts as an embedded system.

A popular example of a data-acquisition system to obtain analog data in the aerospace industry is with strain gauge measurements. Typically, strain gauges are used to observe the stress and strain forces that the plane feels in a particular section of the aircraft, such as the fuselage. These voltage readings are converted into digital signals by the computer, from where the user can perform from post-processing calculations to get an accurate reading of the stress and strain to ensure the safety of the aircraft in extreme weather conditions in the air. Many other sensors work the same way and are used across various industries, such as pressure sensors, temperature sensors, light sensors, ultrasonic sensors, and many more.

In this paper's specific application, the data-acquisition system consists of an accelerometer responsible for outputting a voltage reading that is processed by the computer through the Esduino microcontroller to give the user data of the angle of inclination of the device in reference to the gravitational force.

A common example where this application is important is in mobile devices, as our cell phones now have the feature to detect the inclination of the device to rotate the screen to meet the user's point of view.

The question is, how is this done to always meet the user's expectation without failure? This paper goes through the process of setting up and testing this application, and discusses relevant concepts related to this design. In section 2, a system overview of the data-acquisition system will be given, outlining specifications, key terms and specific components and their operations in the system. In section 3, our design will be implemented, and results will be outlined. Lastly, in section 4, we will verify our results to confirm that the readings observed are correct, and comment on how to improve performance and reduce our deviations.

II. System Overview

As previously stated, the purpose of this project is to design a data-acquisition system capable of outputting the inclination angle of the device. The transducer, being the accelerometer, will be used to output the angle reading as a voltage value to the A/D convertor that will finally convert the analog signal to a digital signal that will be read by the computer. From here, the computer will graphically display the result of the angle through serial communication. This relationship and system overview are shown in figure 1.

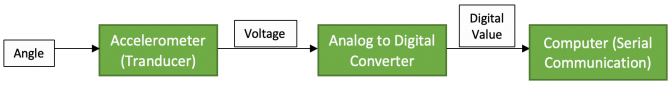

Figure 1: Breakdown Process of Data-Acquisition System

A. Input

The input to this system is the reading of the device known as the ADXL337 accelerometer (as shown in figure 2), which is done through the accelerometer measuring the acceleration

forces. In this project, the acceleration force is referring to the gravitational force that the accelerometer reads. This force is a dynamic force, as this reading is obtained through moving the accelerometer.

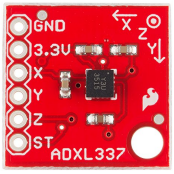

Figure 2: ADXL337 Accelerometer

Whenever an input from the real-world is taken or sensed, a transducer is required to convert this signal into an electrical signal. However, in this scenario the transducer's conversion is done by using the accelerometer as we are only measuring the output signal, being the voltage reading. Therefore, we do not need to use a transducer in this experimental setup.

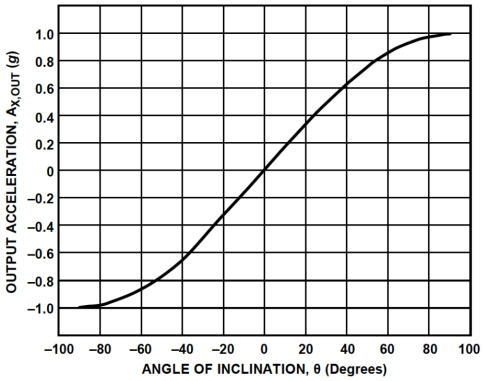

Figure 3: Output Acceleration vs. Angle of Inclination for Single Axis Inclination Sensing

Due to the accelerometer outputting a voltage value and the inclination angle being done digitally, the acceleration is presented as a constant acceleration that is obtained through an ADC. The output resolution is the output acceleration, and the graph represents that the resolution is best around 0 degrees and is very sensitive or worst at 90 degrees. This difference in sensitivity is crucial to take into account when obtaining the inclination angle as this can result in serious deviation in our results if not accounted for.

The approximation method used to convert the analog signal into a digital signal is the successive approximation method. This method simply uses a reference voltage and either multiples by 0.5 if the value is greater than the desired voltage or multiplies by 1.5 if the value is lower than the desired voltage. This process continues for a certain amount of cycles that is dependent on the number of bits of your microcontroller device, that will be explained in the next section. This method is more accurate and faster in comparison to the linear approximation method, which uses different ranges of values to approximate the angle.

The range of the potential voltage values that the accelerometer can output are shown on the datasheet to be 1.35V to 2.0V. This represents a range between -90 to 90 degrees. However, in our application, we will only be finding values between 0 to 90 degrees. We will also see that the experimental values of the output voltages may potentially differ from the theoretical values given on the ADSXL337 datasheet. The transfer function between the ADC voltage values and the angle for the accelerometer is given in figure 4 shown below.

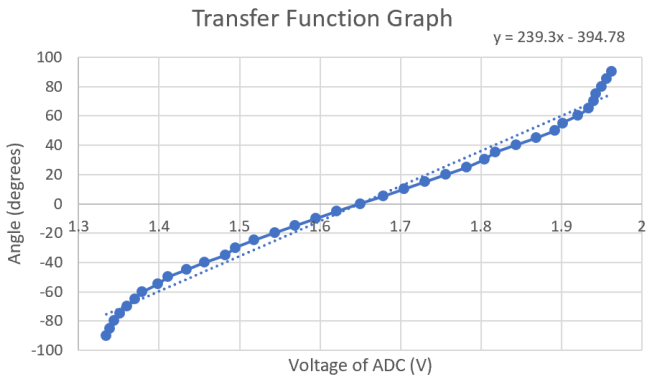

Figure 4: Relationship Between ADC Voltages and Angle Values

B. Esduino: Hardware Overview

The microcontroller chosen in this project is the EsduinoXtreme microcontroller, which we will refer to as simply the Esduino, shown in figure 5.

Figure 5: Esduino Xtreme Microcontroller

The Esduino is a 16-bit microcontroller that is a part of the 9S12GA240 family, with EPROM, SRAM and flash memory capabilities. In addition, the Esduino has 16-bit enhanced timer with input capture, output capture, counter, and pulse accumulator options that will be used in this project. Also, the Esduino has a serial peripheral interface (SPI), serial communication interface (SCI), controlled area network (CAN) communication subsystems, and multi-channel analog-to-digital (ADC) conversion capabilities. The ADC capabilities is a very crucial aspect of this experiment, as we will explore later. The Esduino microcontroller contains several other features and capabilities as shown in figure 6. The Esduino microcontroller is available for purchase for an average price of USD $62.

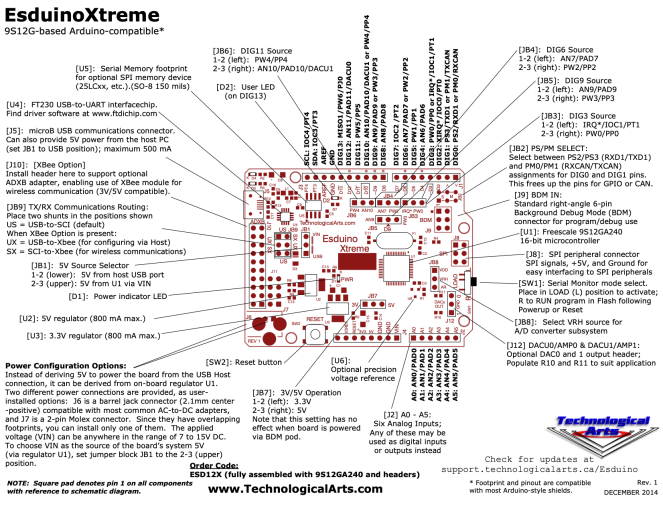

Figure 6: Esduino Datasheet and Features Overview

The bus speed refers to the speed of number of pulses per second generated by the internal oscillator. This speed represents the speed at which the transmission of data can be performed, which can also be described as the number of clock cycles the Esduino can perform. By default, the Esduino clock speed is 6.25 MHz although this bus speed can be modified, as we will explore more later in the report.

The Esduino operates at either 3.3V or 5V, which depends on the user's preference or application. This voltage is jumper selectable on the microcontroller.

As said before, there are 3 different memory resources available on the Esduino. The first is electrically erasable programmable read only memory (EEPROM) with a total of 4K bytes available. The second is static random-access memory (SRAM), with 11K bytes available to use. Thirdly, there is 240K bytes of flash memory.

There is a total of 12 ADC pins, representing 12 locations on the microcontroller where the analog-to-digital conversion can take place. This can be done by using either 8, 10 or 12-bit resolution. These pins can also be configurated to general purpose input or output pins as digital signals, that will be done in this project as well. However, there may be slight errors in this conversion due to a phenomenon known as the quantization error, which is related to the ADC resolution, which we will discuss later.

The Esduino can be programmed using either C or assembly programming. This can be done using one of the Esduino's integrated development environments (IDE), such as CodeWarrior, Cosmic Software's cas12x and Dirk Heisswolf's HW12. In this paper, we will focus on CodeWarrior as this IDE is considered the industry standard and is available for free. In addition, CodeWarrior offers interactive debugging options with the Esduino, hence a USBDMLT will be used, which is similar to a plug-in USB that connects between the Esduino and the PC being used.

C. *Esduino: Program*

The Esduino microcontroller is being used to perform the analog to digital conversion of the output voltage signal that we receive from the accelerometer. The program will help in automating this process so that the Esduino is constantly converting this signal and outputting the results through serial communication using the MATLAB program. There are also two push-buttons used as additives to the hardware. The first button is used to either start or stop the serial communication. On the other hand, the second button is used to change the mode or method of how the LED's are displayed. The first mode will display the LED's as a binary number. For example, the value 63 will be represented as 0110 0011, with the 4 lower bits (0011) representing the one's digit (3) and the higher 4 bits (0110) representing the ten's digit, being 6. The second mode will display these LED's as an increasing approximation. For example, the value 26 will be rounded to approximate 30 degrees and will be displayed by the lower 3 LED's turning on. In this mode, the one's digit is always rounded either up or down. The buttons will be implemented through interrupts, which are used to break-away from the main code and help successfully operate the buttons function.

There are several registers such as the ADC and TIMER registers that will be configured based on our design. The Esduino has pre-set values for several functions although this can be configured to match the user's specifications. The flowchart shown in figure 7 helps visually show the flow of the program that will be programmed using C on CodeWarrior's IDE.

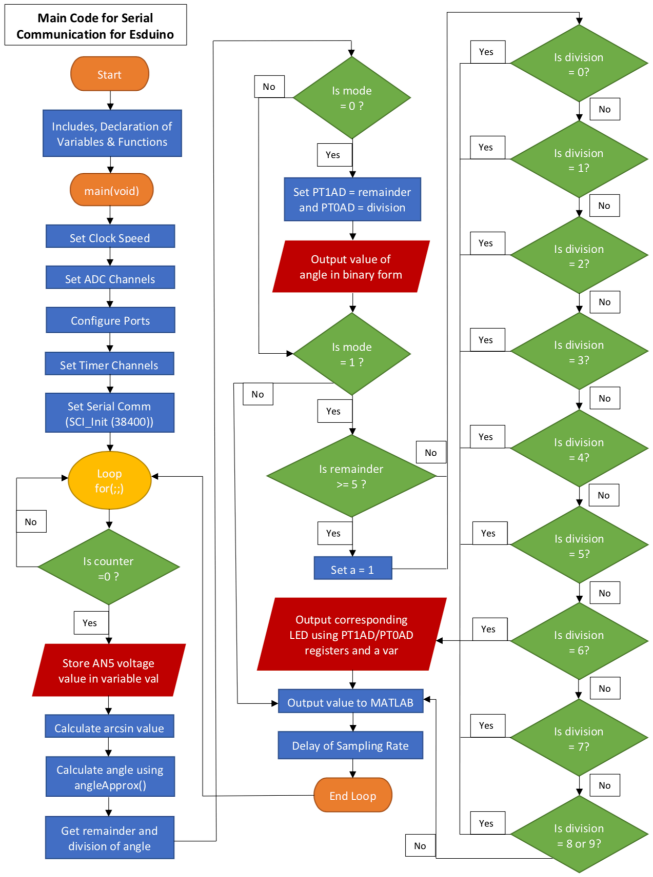

Figure 7: Main Code for Serial Communication on CodeWarrior

In addition, figure 8 shows the function of the interrupts which help in the operation of the buttons. Other channel and port configurations are also shown in figure 9 and 10.

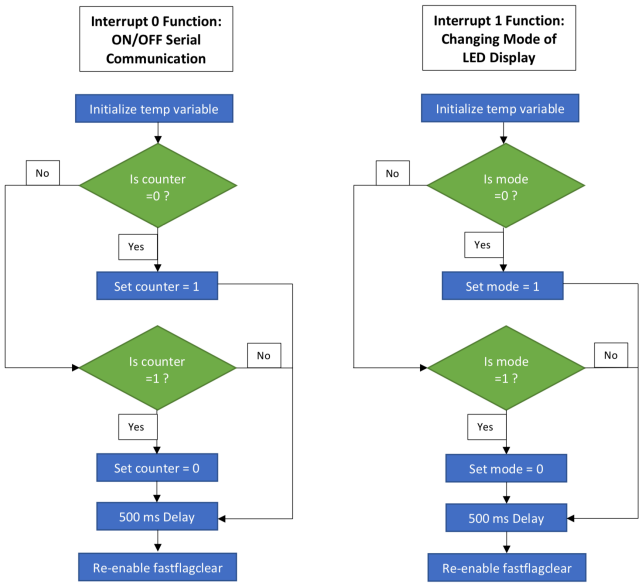

Figure 8: Interrupt Function Flowchart on CodeWarrior

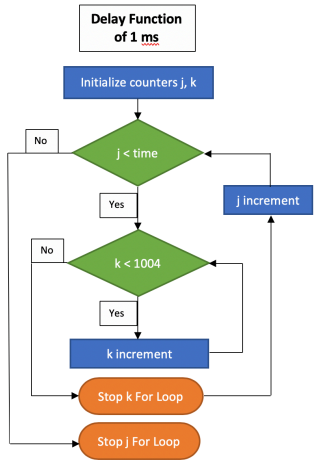

Figure 9: Delay Function of 1ms

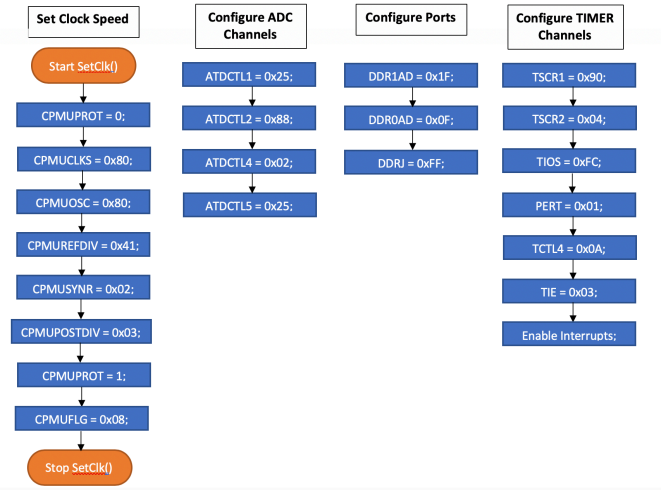

Figure 10: Clock Speed, ADC, TIMER and Port Configurations

The assigned parameters for this project include an assigned ADC channel at AN5 (refer to figure 5 for exact location), a bus speed of 6MHz and a resolution of 10 bit. These requirements are summarized in table 1 below.

The ADC channels are crucial in modifying the analog to digital conversion to satisfy our requirements. First of all, since our bit resolution is set to 10, ATDCTL1 = 0x2F. Secondly, due to ATDCTL2 = 0x88 simply right justifies this result. ATDCTL4 = 0x02 determines the pre-scalar frequency, which is set to 2 MHz as this value matches our bus speed of 6 MHz. Lastly, due to AN5 being our assigned ADC channel, ATDCTL5 = 0x25 to make this channel perform continuous conversions in obtaining data from the assigned pin.

Due to the usage of interrupts for the buttons, the TIMER channels are also set up to catch the button signal at any time of the code's operation. The TIMER channels as shown in figure 10 allow us to capture this data and enable the interrupts.

TABLE I: SPECIFIED EXPERIMENTAL PARAMETERS

Specific Experimental Requirements	
Assigned ADC Channel	AN5
ADC Resolution	10-bit
Bus Speed	6 MHz

The bus speed requirement of 6 MHz refers to the Esduino clock speed of 6.25 MHz being modified. This is done through creating and modifying the SetClk() function as shown in figure 9 above. First, CPMUPROT = 0 disables the clock speed write protections. CPMUOSC = 0x80 sets the oscillator frequency or clock reference of the Esduino to 8 MHz, and from here CPMUREFDIV = 0x41 divides this reference frequency by 2 to get 4 MHz. From here, CPMUSYNR = 0x02 and CPMUPOSTDIV = 0x03 both multiply and divide the VCOCLK frequency in order to get the 6 MHz bus speed we want. To end this function, we set CPMUPROT = 1 to enable clock write protections.

When observing the data sheet of the accelerometer, we see that the range of voltages that we can get from the accelerometer is approximately 1.35 V – 2.0V. On the other hand, the Esduino's range of acceptable voltages is 0V to 3.3V. We will also see later that in specific; a 0-degree angle is represented by 1.65V while 1.35V to 1.65V represents -90 to 0 degrees, and 1.65V to 2.0V represents 0 to 90 degrees. The datasheet of the accelerometer also states that the frequency of operation is approximately 550 Hz. According to the Nyquist rate, the appropriate sampling frequency would be equal or greater than 2 times the input frequency. This relationship is shown in figure 11 below.

$$Nyquist\ Rate \geq 2 * Input\ Frequency$$

Figure 11: Nyquist Rate Relationship

When calculating for this value using an input frequency of approximately 550 Hz, we get a Nyquist frequency that must be equal or greater to 1100 Hz. Therefore, the sampling period can be calculated by doing the inverse of this value as T = 1/f, giving as a sampling time of 0.001 seconds.

As stated before, the application used to serially communicate this data graphically is MATLAB. The rate at which serial communication of the data is performed is known as the baud rate, measured in bits per second. The baud rate is dependent on the bus speed of operation. Therefore, we will need to calculate and find an appropriate baud rate for serial communication of our data. The equation to obtain the baud divisor is shown in figure 12 below.

$$Baud\ Divisor = BusClock/(16 * Baud\ Rate)$$

Figure 12: Baud Divisor Equation

The goal is to use the highest baud rate possible that is lower than a 6% error margin. The different baud rates that we can use are 2400, 4800, 9600, 19200, 38400, 57600, and 115200. Calculations for a few of these baud rates are shown below.

$$Baud\ Divisor = \frac{6000000}{16 * 115200} = 3.255$$

$$Baud\ Divisor = \frac{6000000}{16 * 57600} = 6.510$$

$$Baud\ Divisor = \frac{6000000}{16 * 38400} = 9.766$$

TABLE II: BAUD DIVISOR WITH BAUD RATES

Baud Rate	Baud Divisor
2400	156.25
4800	78.125
9600	39.063
19200	19.531
38400	9.766
57600	6.510
115200	3.255

The baud divisor can only be a whole number, hence must be rounded and recalculated to see if the given baud rate is less than 6% error from the actual theoretical baud rate. These calculations are done for the 3 highest baud rates below. As said before, we are trying to achieve the highest baud rate possible within the error margin.

$$Baud\ Rate = \frac{\frac{Bus\ Speed}{Baud\ Divisor}}{16}$$

Figure 13: Baud Rate Calculation

$$Baud\ Rate = \frac{\frac{6000000}{3}}{16} = 125000$$

$$Error = \frac{125000 - 115200}{115200} * 100 = 8.51\%$$

$$Baud\ Rate = \frac{\frac{6000000}{7}}{16} = 53571.4$$

$$Error = \frac{53571 - 57600}{57600} * 100 = 6.99\%$$

$$Baud\ Rate = \frac{\frac{6000000}{6}}{16} = 62500$$

$$Error = \frac{(62500 - 57600)}{57600} * 100 = 8.51\%$$

$$Baud\ Rate = \frac{\frac{6000000}{10}}{16} = 37500$$

$$Error = \frac{37500 - 38400}{38400} * 100 = 2.34\%$$

From the calculations above, we can see that a baud rate of 38400 gives us an error of 2.34%, hence satisfies the error margin.

The code will be implemented differently based on what program is being used for the serial communication. This is due to each program using a different terminator. The terminator is an operator that tells the program where to separate certain digits into separate values. MATLAB uses a "CR" terminator to distinguish this operation. The bits containing the angle value that we calculate will be serially communicated and represented using our MATLAB program.

D. Computer

For this project, the MacBook Pro 2018 with touch bar. This laptop contains 512GB flash memory, in addition to 8GB memory capability. This laptop also has a 2.3 GHz Intel core i5 processor. This laptop also has Boot Camp installer that was used to download Windows operating system in order to run the CodeWarrior IDE. This laptop contains 4 thunderbolt ports; therefore, an additional dongle was purchased in order to enable USB port connections to the laptop. This dongle can be purchased from Lention or Amazon for around $50 USD.

As shown in figure 6, the code on CodeWarrior first starts with obtaining the value of the output voltage from the accelerometer. From here, the arc sine value is calculating using a reference zero voltage of 1.65V, a full-scale voltage of 3.3V, and a sensitivity factor of 0.3. Linear approximation is then used in order to calculate for the angle of inclination. From here, this value is serially communicated through the USB serial port identified as "COM3" which the MATLAB program reads. The serial port of the device by going into settings, and to device manager that shows the serial USB port that the Esduino is using. The flowchart for the program on MATLAB that connects to the serial port, obtains the angle value, and graphically displays this value is shown below in figure 14.

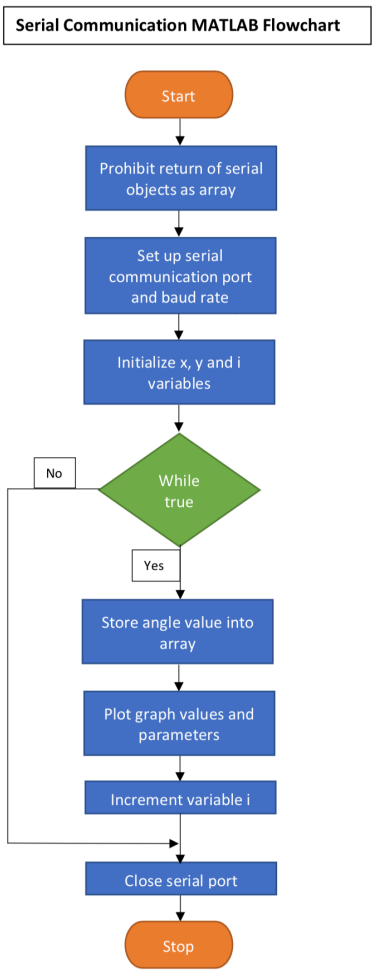

Figure 14: MATLAB Serial Communication Process Flowchart

As shown, first the 'instrfind' MATLAB function is deleted in order to prohibit the return of any serial objects or instruments as an array. This is done as in the next line, we connect to our specific serial port at the specified baud rate and using the terminator option as discussed before, using the serial function. A 'while' loop is then formed, which simply is one of many forms to implement an infinite loop to continue the serial communication of the data until the user specifically specifies to end the transmission. The 'while' loop consists of an array that holds the value of the angle that is then plotted against the corresponding time. The data is sent serially one bit at a time over the communication port. This operation continuously occurs in real-time to give the user a graphical representation of the inclination of the accelerometer.

III. Experiment and Results

In this section, we will observe the data-acquisition system in operation and observe the role of all the components discussed earlier. Each of these components will be validated separately to ensure that the system is working correctly at each stage of the process. This will also help in ensuring that each component is working effectively and not resulting in any error or deviation in our experimental results. First, we will validate the input of the accelerometer to the system. From here, we will validate the clock speed, delay function, analog-to-digital convertor and the serial communication. After observing these separate functions, we will observe the entire system working to ensure the operation of the system.

A. Input

The first crucial component to validate is the input to the system, being the accelerometer. The voltage outputs must be accurate in order to obtain the correct inclination angle of the device. In order to validate the accelerometer, we need to ensure that there is no error with the internal function of the accelerometer. This is done through connecting the accelerometer to an oscilloscope to ensure that the output voltages are theoretically correct at certain values.

First, the accelerometer is left at an inclination of zero degrees. At this point, the output voltage should theoretically be half of the full-scale voltage of the Esduino. In our scenario, this would mean that the accelerometer should output a voltage of 3.3/2 V = 1.65V, which is validated in figure 15 below.

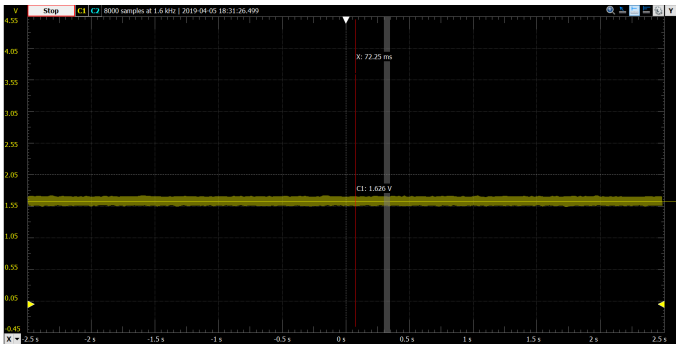

Figure 15: Oscilloscope Reading at 0 Degree Inclination

The exact reading that the oscilloscope is giving is 1.626V, which is close enough to our theoretical assumption, and we can attribute the slight deviation from 1.65V to experimental errors, such as other factors influencing the accelerometers reading or the device not being at exactly 0 degrees with the gravitational surface when tested.

Next, we need to ensure that the range of output voltages lie within our assumptions that were derived from the datasheet of the accelerometer. This is done by testing the accelerometer at the maximum value, being 90 degrees. The oscilloscope reading is shown in figure 16 below.

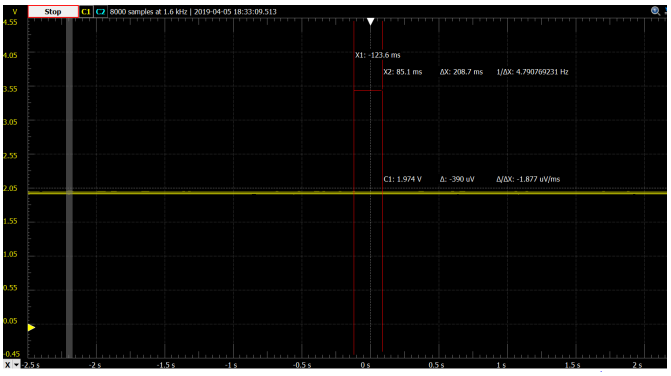

Figure 16: Oscilloscope Reading at 90 Degree Inclination

The exact reading given by the oscilloscope is 1.974V at a 90-degree inclination, which is very close to the expected value of 2.0V. Again, many experimental factors could have resulted in the slight deviation as mentioned before. Finally, a value within the range is tested at approximately 75 degrees, and the output voltage given corresponds to what we would expect the accelerometer to output. This is shown in figure 17 below.

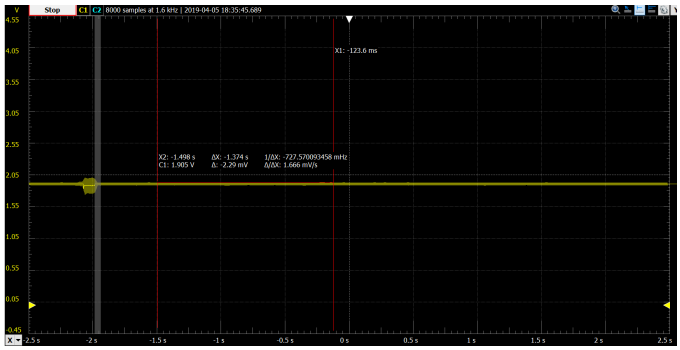

Figure 17: Approximation Oscilloscope Reading Within Range

From the oscilloscope readings shown and cross-referencing these values with the theoretical values given on the ADXL337 datasheet, we can confirm that our accelerometer operates as expected. This also helps validate the input to the data-acquisition system to confirm that the values are correct and within a small error margin.

B. Clock Speed and Delay Loop

As mentioned before, the assigned bus speed for our configuration is 6 MHz. The setClk() function as previously explained is used to configure the Esduino's clock speed to our preference. In other words, we modify the Esduino's default 6.25 MHz clock speed to 6 MHz. To confirm that the clock speed works accurately as expected, an oscilloscope is used with only the delay function to ensure that the 1ms delay function gives us 1ms as we expect. The oscilloscope reading of the 1ms delay is shown in figure 18 below.

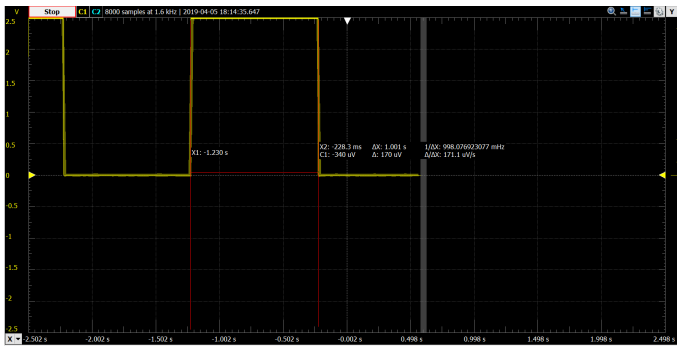

Figure 18: Oscilloscope Reading of Delay Function

The oscilloscope reading shows a delay of 1.001 seconds when the 1ms delay function is run 1000 times. Therefore, this helps prove that the bus speed is configured correctly within CodeWarrior and in the setClk() function specifically. This is an important parameter to check as ensuring the delay function can help us set the sampling rate according to the Nyquist rate, and hence effectively sample the data of the inclination angle through serial communication.

C. ADC System

The ADC system is verified through observing the data plotted using MATLAB through the serial communication port. The Esduino uses the successive approximation method in order to convert the analog data from the accelerometer into a digital value. This method is limited based on the number of bits the ADC channel is configured at. Therefore, there will be some error in the conversion, known as the quantization error. The maximum quantization error can be calculated to ensure that the values we plot are within an acceptable error margin. The equation to calculate this error is shown in figure 19 and is also commonly referred to as the smallest step size or the ADC resolution between the theoretical and experimental results.

$$Maximum\ Quantization\ Error = \frac{Full\ Scale\ Voltage}{2^N}$$

Figure 19: Quantization Error Equation

N refers to the ADC resolution bits. In our experiment, we have a full-scale voltage of 0.348V, which is found by finding the difference between the maximum voltage value we found at 90 degrees, and the minimum voltage value found at 0 degrees, which were found using the oscilloscope to be 1.974V and 1.626V respectively. with an ADC resolution of 10 bits. Therefore, calculating for the maximum quantization error gives us:

$$Maximum\ Quantization\ Error\ (\%) = \frac{1.974 - 1.626}{2^{10}} * 100 = 0.034\%$$

Therefore, the error margin for the ADC operation is approximately 0.32%. This value is very low in comparison to the other errors and deviations experienced in the experiment. However, this error is caused from the internal ADC method of the Esduino and hence cannot be improved any further without changing the requirements of the project, such as the resolution.

D. Serial Communication

The final component that requires validation to ensure that the entire system is working efficiently is the serial communication. This is done through an application called RealTerm, that connects to the serial communication port to ensure that the values of the angles are bring transmitted correctly. The transmission of data is also done in real-time, similar to MATLAB. RealTerm has the option of setting the port and effective baud rate, that we will first set to 'COM3' and a baud rate of 38400 respectively. The output on RealTerm read by the serial communication port is shown in figure 20 below.

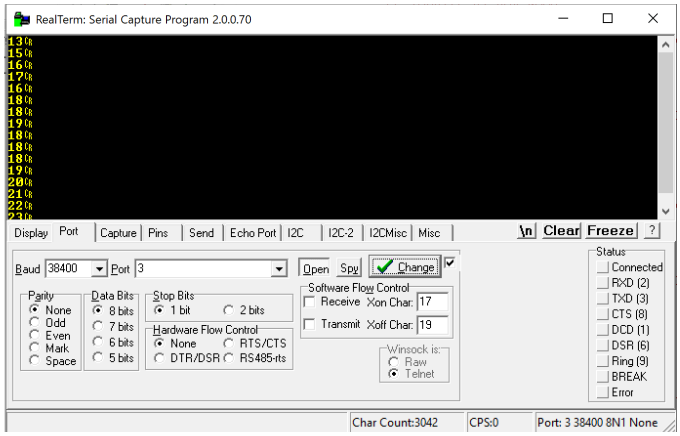

Figure 20: RealTerm Output at Baud Rate of 38400

To ensure that we have selected the correct serial communication port, we will change the baud rate to 57600 and observe the output of the angle and see if this data is transmitted effectively. The RealTerm output reading of this is shown in figure 21 below.

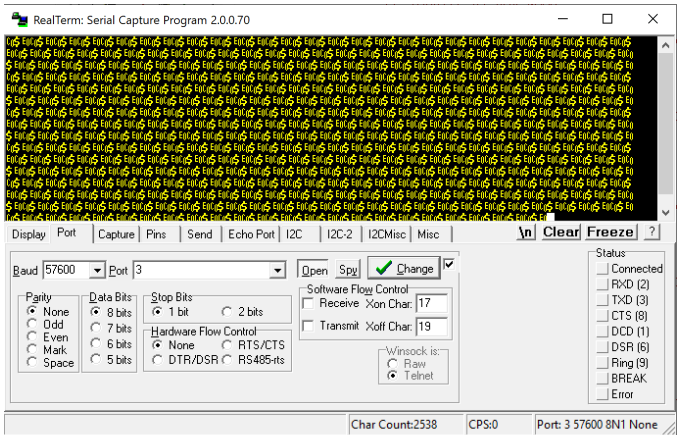

Figure 21: RealTerm Output at Baud Rate of 57600

From the output shown above, we can confirm that the baud rate that we have selected is correct. Using equations for the baud divisor and baud rate as shown in figures 11 and 12 above, we can calculate the baud divisor and baud rate error for each possible baud rate. The calculations are summarized and shown in table 3 below.

TABLE III: BAUD RATE CALCULATIONS WITH % ERROR

Baud Rate	Baud Divisor (Rounded)	Calculated Baud Rate	Percent Error
2400	156	2404	0.17%
4800	78	4808	0.17%
9600	39	9615	0.16%
19200	20	18750	2.34%
38400	10	37500	2.34%
57600	7	53571	6.99%
115200	3	125000	8.51%

The highest baud rate that is within a 6% error must be chosen for the serial communication. As calculated before, using a baud rate of 38400 is proven to be the effective rate of serial communication that reduces the error of the transmission of data compared to other baud rates. Therefore, from our calculations and the output observed on RealTerm, we can confirm that the serial communication is working effectively.

E. Entire System

Now that we have confirmed all our separate components and functions are working effectively with only slight deviations from our result, we can test the entire data-acquisition system as a whole. The hardware and setup of the entire system is shown in figure 22 below. In addition, the graphical representation of the angle is also shown in figure 23, proving that the serial communication of the inclination angle is working effectively.

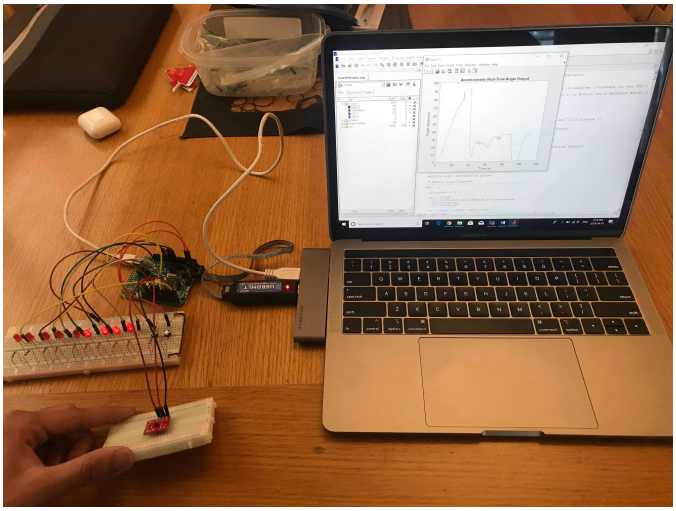

Figure 22: Setup and Configuration of Esduino and Components

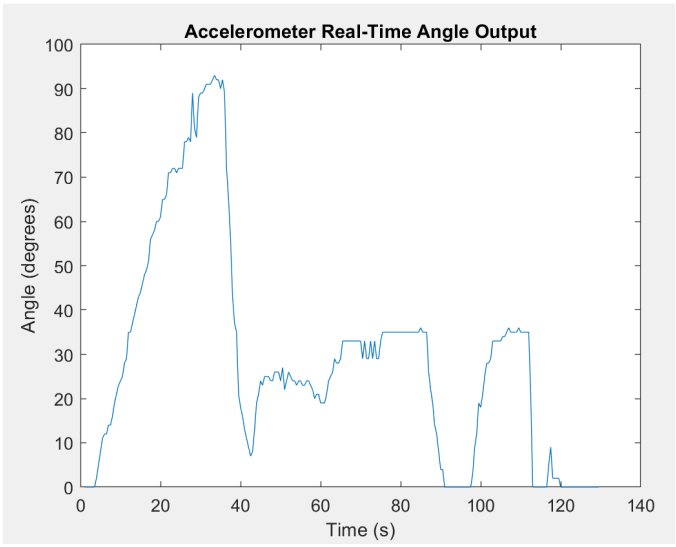

Figure 23: MATLAB Graphical Output of Inclination Angle

IV. Discussion

As seen above, we have been able to produce a graphical representation of the inclination angles given in real time. In addition, we were also able to turn on the LED's appropriately depending on the value of the angle. One simple way to confirm

that the values are consistent between CodeWarrior and MATLAB is to verify that the LED's value of the angle corresponds to the value of the angle that MATLAB outputs graphically through the serial communication port. Another verification method can be to observe the inclination angle of the device with an external support, such as a protractor, and from here compare the angle observed in relation to the gravitational surface and what is plotted on MATLAB. Our validation results showed that there is potential for a slight deviation of approximately 3% from the value plotted, which must also be taken into account to ensure that the reproduced signal is correct.

Using a function generator along with an oscilloscope is also an effective approach to determine if the reproduced signal is correct. This is done through setting the function generator to the correct peak to peak voltage that we determined experimentally (1.65V to 2V full scale voltage) and by connecting an oscilloscope to the output to determine if the inclination angle is correct compared to the output on MATLAB. However, from our MATLAB graph and component validations, we can confirm that the reproduced signal is correct with slight deviations that do not affect the signal greatly and are within a 5% error margin.

We were able to overcome the Esduino microcontroller not having any trigonometric functions through a linear approximation approach using Desmos graphing software. First, an equation is used in order to approximate the arcsine value from the analog reading.

$$arcsine = \frac{\frac{voltage\ reading * 3.3}{2^{10}} - 1.626}{0.31}$$

Figure 25: Arcsine Approximation Equation

In this equation, the 3.3 represents the operating voltage of 3.3V, the 2^10 represents that the ADC is set at a 10-bit resolution. The 1.626 value represents the zero-voltage value of the accelerometer. Experimentally, as shown in section 3.1, we found this value to be 1.626V although theoretically this value is shown on the datasheet to be 1.65V. Lastly, the 0.31 represents the sensitivity factor of the accelerometer, that can also be found on the datasheet to be 0.3, although experimentally we found that a valuer of 0.31 is more effective in accordance with the zero reference voltage we found. Therefore, the output voltage of the accelerometer can help us find the arcsine value of the angle using this equation.

As said before, due to the Esduino not providing any trigonometric functions, we used a linear approximation approach in order to find the angle value. This was done by plotting the arcsine graph on Desmos, as shown below.

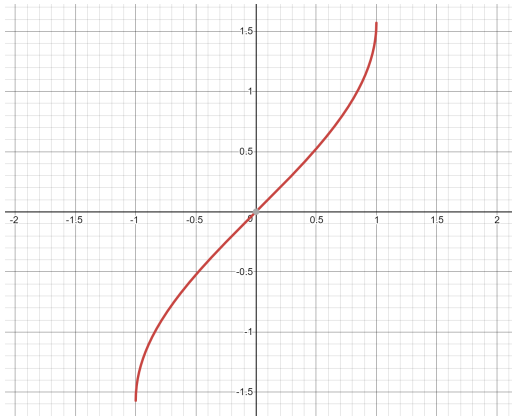

Figure 25: Desmos Graph of Arcsine Function

This graph shows a constant slope from (0,0) to (0.5,0.5). From here, the remainder of the function was split up into different sections depending on the slope of the line. Overall, this function was split into 9 different regions which each performed an approximation to get the value of the angle. For example, a constant slope of 1 was shown in the first region. Therefore, if the arcsine value was calculated to be 0.5, the value would be multiplied by 60 to obtain an inclination angle of 30 degrees. Cross referencing this value with the theoretical calculation, we see that we also get 30 degrees. Therefore, the linear approximation was found to be effective in giving us the correct angle based on the arcsine value calculated.

As calculated earlier, the maximum quantization error is dependent on the experimental full-scale voltage calculated in addition to the ADC resolution, which in our case is 10 bits. This calculation is shown in section 3.3, and again shown below. We get a value of 0.034%, which shows that the maximum quantization error is very low and a small factor in deviating our angle results.

$$Maximum\ Quantization\ Error = \frac{Full\ Scale\ Voltage}{2^N}$$

$$Max\ Quantization\ Error\ (\%) = \frac{1.974\ - 1.626}{2^{10}} * 100$$
$$= 0.034\%$$

As previously calculated, the maximum baud rate that satisfied a 6% error margin must be chosen as the standard serial communication rate. By using the equations found in figure 12 and 13, we found that the best baud rate used based on our assigned bus speed of 6 MHz is 38400. The calculations and errors of all the potential baud rate selections are shown in table 3 in section 3.4. This serial communication rate was verified afterwards using the RealTerm application that further proved that our calculations were correct, as a baud rate greater than 38400 did not accurately communicate the value of the angle through serial communication.

After reviewing the entire system, the primary limitation on the speed is the baud rate. As said before, the baud rate is the rate at which each bit is transmitted per second through serial communication. This was verified through performing a series of calculations to determine the standard serial communication

rate, which we found to be 38400. In addition, increasing this baud rate gives us an inaccurate output of the angle, which further proves that the baud rate we selected was appropriate. We found the selected baud rate to have a 2.3% error, which can also attribute to the limitation or reduction in speed of the serial communication.

The Nyquist rate states that the sampling frequency should be at least equal to two times the input frequency from the accelerometer, which on the datasheet is shown to be 550 Hz. This relationship is also shown in figure 11 above. Therefore, the sampling frequency should be 1100 Hz or slightly greater. This is a crucial rule to follow when transmitting and plotting the data of the angle as or else, there may be aliasing in the signal. Aliasing may cause inaccurate results in the signal and causes different signals to be identified as the same, which can further increase errors or deviations. Due to the sampling rate, the maximum frequency that can be used as an input is 550 Hz. Any greater frequency from the input will cause aliasing and inaccurate data.

From this experiment, we can conclude that in general, analog input signals with sharp transitions are not accurately reproduced digitally. This is due to their being a lot of fluctuations of the rising and falling edge of the signal's waves. We have also noticed that there are many factors that can contribute to potential errors or delays in the plotting of the graph. With sharp transitions, this can greatly affect the user's interpretation of the data and can hence lead to further deviations in the data. Therefore, it is more accurate to recreate shapes such as a sine wave, in comparison to a sawtooth or square wave representation.

There are also many areas of improvement for the experiment. First, using interrupts is helpful in reducing the run-time and processing time of the code which can be beneficial when plotting real-time data and transmitting data at the sampling rate. Secondly, the test apparatus can be improved by being more precise with the placement of the accelerometer in relation to the gravitational surface. It is very possible for the accelerometer to be slightly tilted upward or downward on the zero position, which can greatly affect results. As said before, the Esduino uses successive approximation in order to get the digital value from the ADC channel. This process is dependent on the ADC resolution. Therefore, increasing the ADC resolution can also help in getting more accurate values from the analog to digital conversion. In addition, following the Nyquist relationship helps in the sampling process of plotting the serially transmitted data, hence ensuring that this relationship is satisfied also greatly improves the results.

V. Conclusion

In this paper, we walked through how to create and implement a data-acquisition system responsible for graphically representing the inclination angle using an accelerometer through serial communication. The system first converted the analog signal of the acceleration into a digital signal using ADC conversion from the Esduino, which was then serially transmitted to MATLAB to plot the inclination angle value. The system worked efficiently, and we see that the angle of inclination is appropriately plotted in real-time, along with the corresponding LED's to the value of the angle turning on. In addition, the buttons implemented through interrupts also efficiently toggled the serial communication and changed the display mode of the LED's. As we continue to progress within technological innovations and performance testing, systems similar to the one implemented in this paper will be used more commonly for various applications. Transferring data between the real world and a PC is very efficient and with further findings, can be further improved to reduce any deviations in the process. Such a system can serve very useful in the real world, such as in the orientation of any technological device to meet the user's point of view at any given time, or in real-time. We further tested our components and system using an oscilloscope to ensure the validity of the reproduced signal. As time progresses, these systems will be more common in our daily lives and both data-acquisition and ADC conversion can help to serve as a positive influence in the quality of our life and in the technological world.

REFERENCES

[1] Ni.com. (2019). *What Is Data Acquisition? - National Instruments*. [online] Available at: http://www.ni.com/data-acquisition/what-is/ [Accessed 7 Apr. 2019].

[2] Rudo, P. (2019). *6 Important Stages in the Data Processing Cycle*. [online] Enterprise Features. Available at: http://www.enterprisefeatures.com/6-important-stages-in-the-data-processing-cycle/ [Accessed 7 Apr. 2019].

[3] Analog.com. (2019). [online] Available at: https://www.analog.com/media/en/technical-documentation/application-notes/AN-1057.pdf [Accessed 7 Apr. 2019].

[4] Analog.com. (2019). [online] Available at: https://www.analog.com/media/en/technical-documentation/data-sheets/ADXL337.pdf [Accessed 7 Apr. 2019].