

THE OHIO STATE UNIVERSITY

# Wireless Accelerometer

---

ECE 682

*Dynamic Acceleration*

Clay Cochran, Tim Duly, Brandon Ervin, & Mike Jender

3/13/2009

## Executive Summary

### Purpose

This document proposes the design and development of a wireless circuit network to record and analyze the acceleration of some system.

### Problem Statement

An alarming statistic that is increasing today is the number of students not entering engineering after high school. High school students with interest in engineering leave high school and enter college with little or no engineering experience. A small programmable microcontroller could be used to create a fairly simple application such as measuring acceleration to be integrated into a high school lab setting.

### Background Research

By using the TI eZ430-rf2500 Wireless Development Tool, it is possible to create a more user-friendly, easy-to-use accelerometer; this device could be used in a variety of ways that high school students would find interesting.

Once the accelerometer device is designed and built, using the unit discussed just previously, novel and interesting ways to use it in the classroom must be devised. One such application could be use on a model roller coaster. In this case, the accelerometer could be used to analyze speeds or g-forces at various times throughout the roller coaster.

### Design Approach

We will investigate wireless transmitters and buy one to evaluate. After we are familiar with the product, we will integrate the accelerometer to send data. Once we have a good means of sending and receiving acceleration data, the door is wide open for applications. The spiral design process will be used; many parts of the design will be revisited.

### Design

We plan to use the eZ430-RF2500T to send data from the accelerometer. Our initial design included a digital accelerometer but it was decided to revisit the design and we settled on an analog accelerometer. It was much easier to debug and fix problems. We used MATLAB to record acceleration in the X, Y, and Z directions and plotted it on a screen. We plan to apply this system to a model roller coaster with Coaster Dynamix and integrate it into a high school lab project modeled after the Infinity Project.

### Conclusion

The design and development of a fairly simple and affordable wireless analog accelerometer network has important applications, one of which to measure acceleration of a model roller coaster for a high school lab setting.

## TI Parts Discussion

### RF Transmitter and Receiver

The CC2500 RF transceiver provided communication from the accelerometer on the roller coaster to the receiver node. It was low power which ensured a longer battery life for the system. This is also important to consider since there are a multitude of transmissions occurring. The communication link provided sufficient distance from our transmitter and receiver—perfect for using inside a room.

### MSP430 Microcontroller

The MSP430F2274 was used. This microcontroller provided the necessary analog to digital conversion as well as the processing power to read the data. On the transmitting end, the microcontroller provided the analog to digital conversion, necessary data manipulation, and set up to be sent wirelessly through the CC2500. On the receiving end, the chip acted as a means to pass data through to the USB port on the computer, which was read by our MATLAB program.

This particular microcontroller was chosen for a variety of reasons. It was low power which meant our system could work longer before changing the battery. Its size was ideal for attaching to the accelerometer board and for future implementation in a PCB board.

### Voltage Regulator

The TPS71530 was used to allow different voltages to be connected to the system. This part accepted voltages up to 24 Volts and provided a constant output of 3 V; therefore no adjustments were needed to maintain the voltage level. This ensured consistent operation. This part had an error or 4%, which still provides acceptable voltages for the accelerometer. The TPS71530 was designed to work with MSP430 microcontrollers and is very suitable for our purposes.

# Table of Contents

- Executive Summary ..... 1
  - Purpose ..... 1
  - Problem Statement ..... 1
  - Background Research ..... 1
  - Design Approach ..... 1
  - Design ..... 1
  - Conclusion ..... 1
- TI Parts Discussion ..... 2
  - RF Transmitter and Receiver ..... 2
  - MSP430 Microcontroller ..... 2
  - Voltage Regulator ..... 2
- Table of Contents ..... 3
- Introduction ..... 4
  - Purpose ..... 4
  - Problem Statement ..... 5
  - Scope ..... 5
- Background Research ..... 6
- Requirements Analysis ..... 7
- Design Approach ..... 7
- Initial Design Plan ..... 7
- Final Design ..... 8
  - Explanation of Code ..... 10
  - Calibration ..... 11
- Sample Image Capture ..... 11
- Report of Work ..... 12
- Resources ..... 12
  - Personnel ..... 12
  - Facilities, Equipment, and Costs ..... 13
- Schedule ..... 14
- Design Review ..... 15

Future Planned Developments .....	16
Conclusion.....	16
Appendix: Code.....	17
C Code Used in IAR Embedded Workbench.....	17
Access Point .....	17
End Device.....	22
MATLAB Code to Display Plots.....	26

## Table of Figures

Figure 1: Block Diagram of Wireless Accelerometer Network .....	8
Figure 2: Schematic of Analog Accelerometer Endpoint .....	10
Figure 3: Sample Acceleration in X,Y, & Z .....	11
Figure 4: SparkFun 3-D Accelerometer .....	14
Figure 5: TI MSP430 Wireless Development Tool.....	14
Figure 6: Table of Design Work to Be Completed .....	15

## Introduction

### Purpose

This document proposes the design and development of a wireless circuit network to record and analyze the acceleration of some system. This is to be done by implementing the use of a wireless capable microcontroller and a small accelerometer. This circuit would be valuable because it could be used for a wide variety of projects and applications. The wireless accelerometer network could be used if development is fairly straightforward to introduce and promote interest in high school students to engineering by implementing the development of the circuit in a class room. This is important because many high school students are not interested in engineering programs when they graduate because they are not exposed to them and think that it is too complicated to complete the large number of complex science and math courses. The accelerometer circuit could also be used in many other more common engineering applications. A rollercoaster model one of these examples of a project that the wireless accelerometer could be implemented into. This could be used to measure the acceleration in three directions to see how intense the actual rollercoaster experience would be for someone riding the actual rollercoaster. If the wireless accelerometer network is found to be effective and fairly compact, it would also be possible to implement in many other extremely important and marketable applications.

## Problem Statement

An alarming statistic that is increasing today is the number of students not entering engineering after high school. As the world continues to become more technologically centered and complex, it is important to make certain that there are an abundance of potential engineers leaving our country's high schools.

High school students with interest in engineering leave high school and enter college with little or no engineering experience. As a result, there are a large percentage of students that drop out of engineering once they get into the major because they feel overwhelmed or unprepared. Furthermore, many students lack the initial interest in engineering because they are not exposed to it. They are discouraged by the way math and science are taught and try to avoid a field of study or career that will be using these for the rest of their lives. In order to combat this, students need to be exposed to complex but achievable engineering projects in high schools. These projects will also demonstrate a real life application using math and science as opposed to only working trivial problems.

Teachers feel frustrated with teaching because many lab kits are expensive and overly complex for the high school level. Therefore it is important to make sure the teachers are well prepared and educated regarding engineering lab kits so that they can effectively teach their students. It is just as important to make the kits low cost so public schools can afford them to be implemented in their labs and provide their students with the same opportunities as private schools.

Some kits are available such as the Infinity project that utilizes a digital signal processor to create various projects. Programming a digital signal processor is fairly complicated, especially at the high school level. Therefore another approach that is easier to work with and that requires less background knowledge should be used in the kits to make the program more effective. A small programmable microcontroller could be used to create a fairly simple application such as measuring acceleration. Perhaps this could be applied to a model roller coaster application such as a Coaster Dynamix roller coaster. Engineering education would also be complimented by physics and other science disciplines.

The accelerometer circuit could also be implemented into many more important applications if it is fairly accurate. There are many applications such as testing of automobiles and safety and health of human beings just to name a few that could be improved by the use and implementation of an accurate wireless accelerometer network.

## Scope

This report discusses the design and implementation of the wireless accelerometer circuit. It discusses the different pieces of equipment and the hardware selected for the implementation of the circuit and how they will be connected and implemented to get the desired results. It will only discuss the applications of the circuit very minimally. There is much more of a focus on the actual conceptual design than on the actual applications of the accelerometer network. Although only a small number of applications are mentioned, there are unlimited applications for this wireless accelerometer circuit.

The accelerometer was designed and built throughout the course of the design series of ECE 582 and ECE 682/683. A fairly simple Graphical User Interface (GUI) was developed in MATLAB to monitor the acceleration readings in the X, Y, and Z – directions. This provided a way of collecting the acceleration data so it could be used for analysis of the system later. The circuit and data collection

system was completed and is discussed in addition to the possible advancements and improvements to the system.

## Background Research

It is essential to continue emphasizing exposure to the sciences and engineering during high school. A major obstacle standing in the way of this goal is the complexity of many existing technology kits. By using the TI eZ430-rf2500 Wireless Development Tool, it is possible to create a more user-friendly, easy-to-use accelerometer; this device could be used in a variety of ways that high school students would find interesting.

The TieZ430-rf2500 Wireless Development Tool is actually a device with three main parts. At the heart of the device is the MSP430F2274 Microcontroller. This is a very low-power microcontroller that includes several useful peripheral components. These include a 10-bit A/D converter, two general-purpose operational amplifiers, two 16-bit timers, and 32 I/O pins. Thus, this microcontroller is very useful for applications that require an analog signal to be digitized and subsequently processed. This is an important ability, because it would enable the processing of analog accelerometer data.

Also included in the Wireless Development Tool is the CS2500 Transceiver. This is a low-cost electronic component that both transmits and receives wireless data; it is designed especially for use in very low-power applications. It is equipped with a modem capable of adjustable data rates of up to 500 kBaud, as well as various modulation schemes. There are 64-bit FIFO's for both transmit and receive. This component would allow the accelerometer to be wireless, opening up a variety of exciting possible uses.

Lastly, and perhaps most importantly, the eZ430-RF Debugging Interface is included with the device. This piece allows the user to use a USB connection to program and debug the attached microprocessor. This component can be separated from the rest of the unit. This is one of the most important advantages of this device, as it allows access to an easy-to-use interface with the microprocessor and allows step through debugging.

Also included in the device is a "target board", which is a group of 18 development pins that allows easy access to the MSP430F2274 microcontroller. This provides a simple way for the user to connect various types of hardware to the microcontroller, and allows for a wide range of uses for the unit.

Once the accelerometer device is designed and built, using the unit discussed just previously, novel and interesting ways to use it in the classroom must be devised. One such application could be use on a model roller coaster. In this case, the accelerometer could be used to analyze speeds or g-forces at various times throughout the roller coaster.

CoasterDynamix, a small company based in Virginia, makes high-quality roller coaster models that would work very well in conjunction with the accelerometer unit. These well-made scale models would expose students to interesting engineering and physics concepts which they could identify with.

## Requirements Analysis

This system will begin as a general accelerometer data transmitter, and apply it to specific applications. We plan to implement this device into a classroom setting for a high school or even junior high laboratory, which would be more effective than a complex tool kit that the Infinity project now has. Furthermore, this system has selling potential to companies such as Roller Coaster Dynamix in which they want show in real time the acceleration of their model roller coasters.

The system must be able to transmit acceleration data wirelessly to a receiver within a hundred meters. This data will be stored in a microcontroller and will be easily available for calculations (i.e., velocity, distance). The wireless transmit system is required to be physically small: no more than 10 cm x 10 cm in area so that it can be implemented discretely into systems such as a model roller coaster.

## Design Approach

Our team is using a bottom up approach to designing this system. We will start with an accelerometer evaluation board to test and characterize. Next we will investigate wireless transmitters and buy one to evaluate. After we are familiar with the product, we will integrate the accelerometer to send data. Then we will work on the receive device and work on how to talk to the transmitter. Once we have a good means of sending and receiving acceleration data, the door is wide open for applications, and we shall approach each application individually and branch off from there.

A common theme in our design approach is taking small steps. This way, we can separate the problem and debug it individually. This is much easier to do rather than building the system as a whole and attempting to fix it when something goes wrong. Also we intend for the programs to be relatively simple for debugging purposes and for creating more complex systems in the future.

A spiral design approach will be used for the development of the system. As time progresses, many parts of the design process will be revisited. As the design process is advanced, the actual solution defined will change as will many other aspects of the design. This is easily shown in a spiral design representation. Also, the debugging process will prove to be extremely important due to actually seeing what the microcontroller is capable of and the programming skills of all team members.

## Initial Design Plan

The initial plan was to create a design for a wireless accelerometer implementing TI parts into the design. We planned to use two eZ430-RF2500T target boards for the design, using one for the main node and one for the end point of the system collecting data from the accelerometer. These devices allow data to be sent wirelessly by the transceivers conveniently located on each target board. The accelerometer data was planned to be output either digital or analog data. This decision proved to be the most important decision because it was discovered that the design was much easier implemented using the analog accelerometer. The digital accelerometer proved to be much more difficult due to the SPI communication scheme. The eZ430-RF2500T has an A/D converter that was to be used to convert the analog signal from the accelerometer to digital for signal processing. This led to much easier troubleshooting of the circuit during testing and was much easier to develop software for.

Once the signal was received, it was planned to be sent to another microcontroller where the data would be collected for processing, and then sent to a computer for interpretation. A fairly simple GUI was to be developed that would make it possible to analyze the data collected from the accelerometer for analysis purposes. It was to be developed to show the X, Y, Z movement on the computer screen and save all of the data collected. Next it was planned to apply a specific application to the wireless accelerometer. For example, with the roller coaster, Coaster Dynamix would like to see the amount of g's exerted on the model rollercoaster in real time as it moves along the track. Also, it could be implemented into a second generation Infinity Project for High School students to learn math and science. Once the system was completed, numerous opportunities for applications arise, with little or no tweaking. Figure 1 shows the block diagram for the project showing both the possibilities of using either an analog or digital accelerometer.

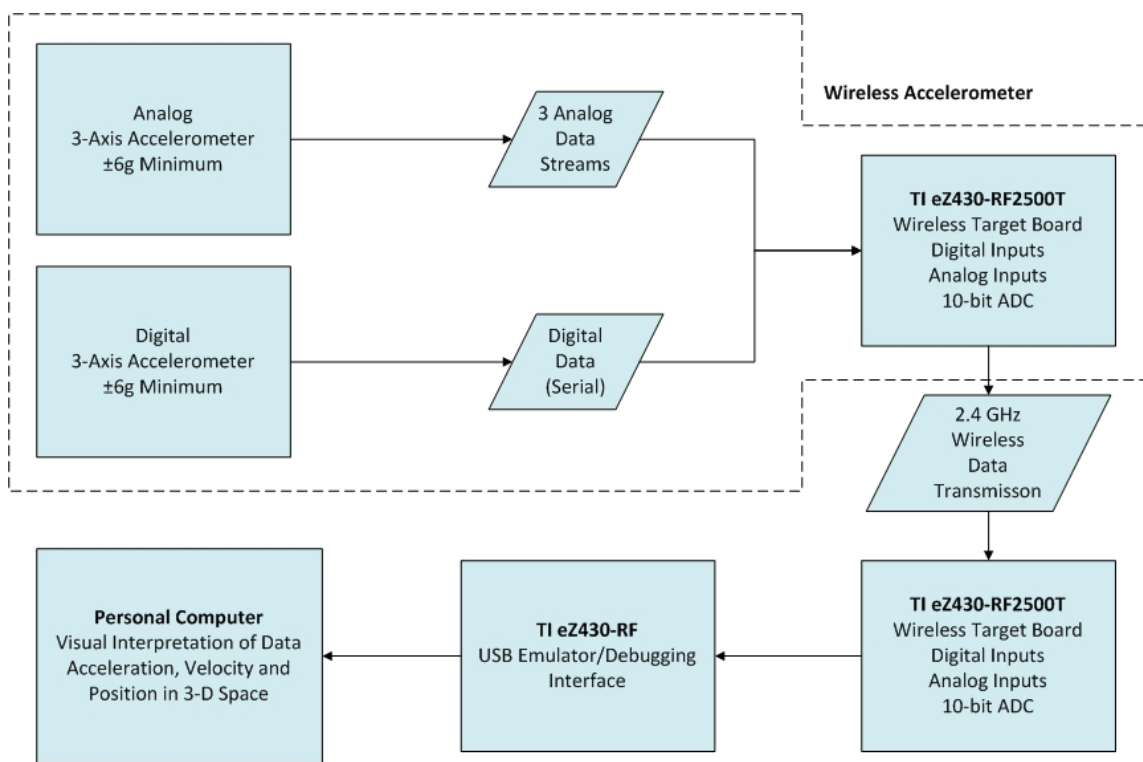


Figure 1: Block Diagram of Wireless Accelerometer Network

## Final Design

The final hardware design was fairly straightforward and is shown in Figure 3. An analog 3-dimensional accelerometer was picked for the final design because it was much easier to test and work with than the digital accelerometer using SPI communications. The pins labeled X, Y, and Z are the actual analog outputs of the accelerometer. These pins were then connected to analog inputs for Port A of the microcontroller target board. This kept all of the data confined to the first three channels of Port A. In order to actually read the data, the SLP pin of the accelerometer had to be set to high. This was done by outputting a high value from pin 8 continuously from the target board. This was using more

power by keeping the accelerometer active all of the time, but it was easiest and most effective for this application.

The schematic of the connections for the endpoint is shown in Figure 2 below. This shows the necessary connections to collect the acceleration data from the accelerometer and then prepare it for wireless transfer to the main node. The Vcc wire is used to power the accelerometer by supplying it with approximately 3 VDC from the power supply of the target board. The GS1 and GS2 pins select the acceleration range for the accelerometer. In this application, both pins were set to high in order to set it for a +/- 6g range of acceleration data. There are other ranges that can be set by changing the configuration of these two pins. In order to get these two pins to high voltage, they are tapped off of the Vcc pole on the breadboard. The ground is obviously then fed from the ground on the target board which is supplied by the power supply of the target board. This would serve as the minus of the battery. The X, Y, and Z pins on the accelerometer are the output pins for the acceleration in that specified direction. These are fed into the analog PORT A on the microcontroller target board as shown below. The SLP pin on the accelerometer could be used to save power by only sending data when the voltage at that pin is high. In order to simplify this, it was set to high all of the time, which would result in a shorter battery life, but since it uses such small power, it was decided that ease of development took precedence. In order to set this high it could be approached two ways. One way was to directly connect it to the Vcc pole, while the other way was to output a high signal from pin 8 on the target board all of the time to that pin.

After a working prototype was built, it was decided that in order to make the system more versatile, a voltage regulator could be used which would accept various voltage levels and output the constant 3 V that the system required. The TPS71530 is suitable for this task, accepting voltages up to 24 V and providing 3 V output with adequate current for the system.

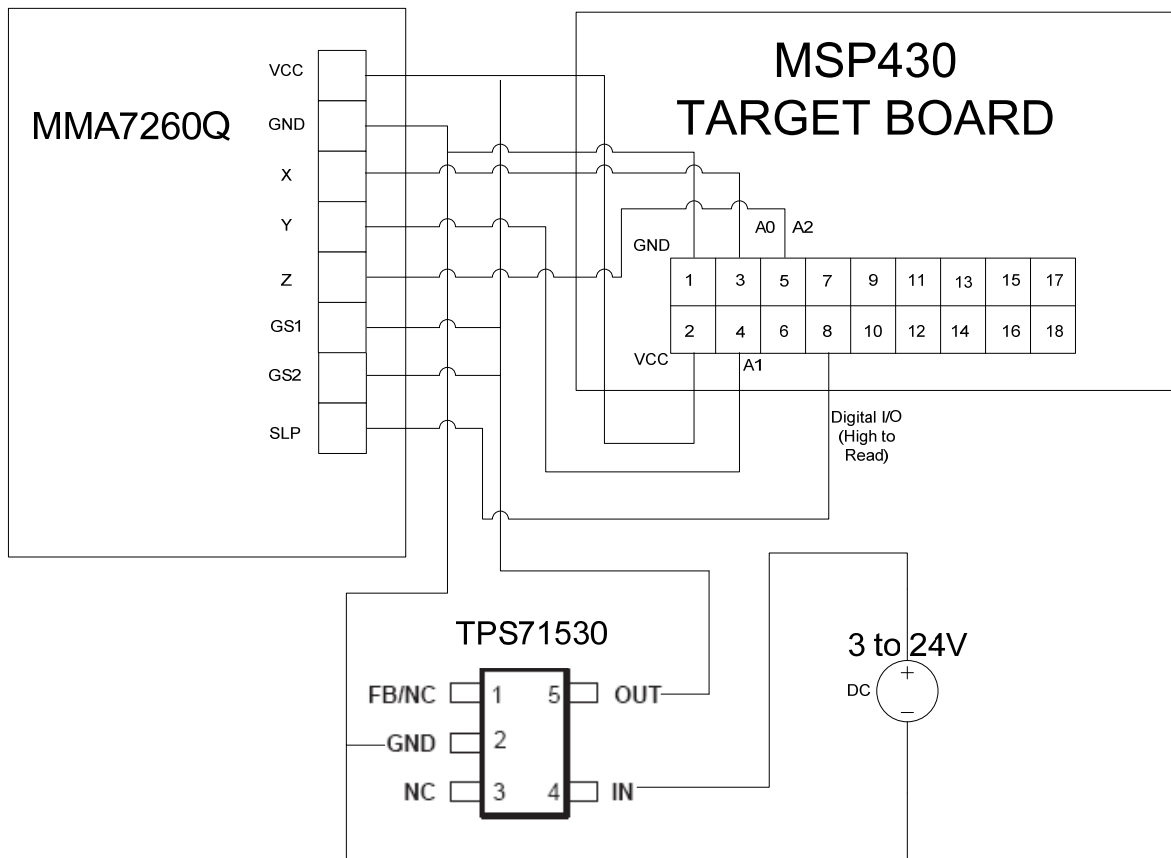


Figure 2: Schematic of Analog Accelerometer Endpoint

## Explanation of Code

The microcontroller code was divided into two parts, access point and end device. The access point code is run on the microcontroller that is attached to a PC and serves to gather the data being transmitted from the end device. This data is then conditioned and sent to the PC through a USB connection. The access point code starts by continuously looping until it detects a signal from an end device. Once a signal is acquired and a connection is made, the access point enters the data processing loop. In this loop the access point continuously gathers data from the end device and conditions the character output to be sent to the PC over the USB connection. The end device code starts by initializing the microcontroller and the protocols needed for wireless data transmission. The end device then enters the data gathering loop where it reads the analog accelerometer voltages using three separate analog inputs. The voltages are converted using the 10-bit ADC to digital numbers ranging from 0-1023. These values are then transmitted to the access point and new values are then acquired. The Matlab code is used to present the acceleration data in a user readable form. The Matlab code starts by opening a COM port that is linked to the microcontroller attached to the PC. The code then continuously polls the COM port for data and enters each data reading into the respective X, Y and Z data matrices. The advantage of having separate matrices for each axis is that the data is stored

separately and can therefore be manipulated later if necessary. After a set number of data points are gathered, the data gather loop exits and the acceleration data for each axis are plotted together. All codes are included in the Appendix at the end of the report.

## Calibration

In order to try to calibrate the acceleration values to the correct values, the accelerometer was placed in the XY plane so that the only force acting on the accelerometer was gravity and it was in the Z-direction. Therefore, the values were then normalized by subtracting in MATLAB by a constant to account for the offset of the actual output of the accelerometer. This calibration was done by taking the Z-axis data and then subtracting by a value to make it equal to  $-1g$ . This same procedure was done to make the values of the X and Y axes equal to  $0g$  since they were both not acted on by gravity at all and the accelerometer was not in motion.

## Sample Image Capture

Figure 3 shows a sample output of running the MATLAB script that captures the X, Y, and Z acceleration and displays in a subplot. This provides the end user with accurate, easily visible acceleration of their system. The maximum acceleration in either direction is  $6g$ 's.

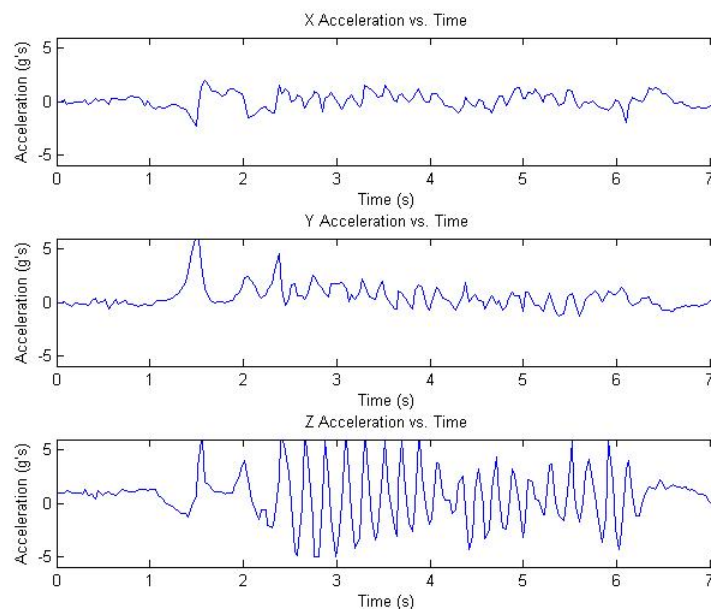


Figure 3: Sample Acceleration in X,Y, & Z

## Report of Work

### 582

The main goal of 582 was to define the problem and requirements of the wireless system/accelerometer. The overall design process was defined and plans parts were ordered during the break so we could begin prototyping in the beginning of 682. A significant amount of time was devoted to developing the requirements that the accelerometer and software will have to meet. The wireless transceiver portion of the design is an off the shelf part from TI and therefore it can be ordered at the end of 582 to provide more build time in 682. At the end of 582 the plans and design should be sufficient enough to begin building as soon as the parts arrive.

### 682

Prototyping was started as quickly as possible. We used the MSP430-RF2500 and spent a significant time learning how it operated and how it sent and received data. We initially ordered a digital accelerometer from SparkFun but after reiteration of the design process decided to use an analog accelerometer. We also ordered a small coin cell battery to power the system.

The design continued to be refined based upon tested conducted with the working prototype. A robust software application using MATLAB was used to gather the data and plot. A graph of the sensors acceleration in real time was successfully completed and gave a demonstration.

### Beyond 682

The prototype was fairly large on a breadboard and if we wanted to run the module on a roller coaster with CoasterDynamix the design will be implemented on a PCB board. Also the report plans to be rigorously reviewed in order to submit to TI contest.

## Resources

### Personnel

The project and tasks previously mentioned will be performed by four senior engineering students at The Ohio State University. They each have a wide variety of skills that when intertwined can be used to successfully develop the wireless accelerometer circuit. The following is a list of all team members and what they are bringing to the team to result in a successful circuit development.

**Clayton Cochran:** He has gained experience in the development and design process by working as an intern with Emerson Network Power: Liebert Corporation in the research and development department. He has also completed a wide variety of coursework including, microcontroller programming and communications that will likely be helpful in the development of the wireless accelerometer circuit.

**Brandon Ervin:** He is a Senior in Electrical Engineering, graduating Spring 2009, who is focusing on Power Systems with minor focuses in Computers and Control. He participated in the FEH program, placing 4<sup>th</sup> in the robot design competition, Spring 2006.

**Tim Duly:** He is focusing his Electrical Engineering studies in Communications/DSP and Electromagnetics. He has had experience programming MSP430 microcontrollers in an internship in 2008.

**Mike Jender:** He is a fifth-year electrical engineering student at The Ohio State University. His coursework concentrations include: Digital Signal Processing, Communications, Control Systems, and Electromagnetics.

## Facilities, Equipment, and Costs

The development of the accelerometer circuit will require a very small amount of equipment and essentially no special facilities. The only facilities that will be necessary will be a place to meet as a group and a place to build and test the circuit. This place needs to only consist of a table large enough to complete the building and testing and then a computer containing the software needed to develop the program for the microcontroller. The following is a list of hardware needed to develop the wireless accelerometer network.

- Microcontroller: Texas Instruments MSP430-RF2500 Development Tool (\$49.00)
  - o MSP430 development tool with target boards (USB development)
  - o CC2500 transceiver for wireless communications
  - o Integrated Development Environment Software for compiling the code
- Small Digital Accelerometer: SparkFun 3-D Accelerometer LIS302DL (\$19.95)
  - o Selectable +/- 2g to +/- 8g range
  - o 8-bit resolution
  - o Extremely small
- Small Analog Accelerometer: SparkFun 3-D Accelerometer MMA7260Q (\$19.95)
  - o Selectable +/- 1.5g to +/- 6g range
  - o Extremely small
- Voltage Regulator (DigiKey) (\$1)
  - o Allows for multiple input voltages up to 24 V
- SparkFun coin cell 3 V batteries and battery holder (\$10.00)
- Wiring and headers for developing the system with the target boards (\$20.00)
  
- TOTAL COST: \$139.90

Both accelerometers from SparkFun that were used are very small, as seen in Figure 4. The analog accelerometer, as mentioned earlier, has a selectable range of acceleration readings from +/- 1.5 g to +/- 6 g. This range of acceleration should be large enough for most of its expected applications.

Although the digital accelerometer encountered performance problems during the course of the project, it is our hope that we will eventually be able to implement a digital version of our project. The 8-bit resolution should also provide the necessary accuracy for the system, but if a more accurate accelerometer system is needed there are accelerometers out there that do have higher resolution. Because this specific accelerometer gives its output in digital form, it will not be necessary to perform an analog to digital conversion to send the data.

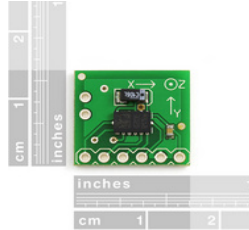


Figure 4: SparkFun 3-D Accelerometer

The main component of the design is the MSP430 Wireless Development tool shown in Figure 5. This kit comes with everything shown in the figure and has an easy to use USB connection to use with a computer. This will come in very helpful while developing the code for the microcontroller and testing the design. This device already has the transceiver and the microcontroller already integrated into one device. This should prove much easier than trying to implement them on our own which could prove to be fairly difficult.



Figure 5: TI MSP430 Wireless Development Tool

As seen above, the cost of the project will be reasonably affordable. Also, some of the parts will most likely be paid for by Coaster Dynamix with the agreement that if the accelerometer works, it can be used on their model rollercoaster.

## Schedule

According to the original schedule, the majority of the project was going to be built and developed through the duration of 682. The planning for the project was mostly done during the duration of 582. Therefore, the goal was to get the preliminary plan and mental idea that were developed during the first quarter of the series, and then actually implement the design and ideas in the last half of the series.

The following Figure 6 shows the original rough temporary plan for the design process of the wireless accelerometer network. It is easily seen that as the project progressed, the goal was to revisit tasks to make certain that the project was progressing where it needed to. This table again essentially shows the spiral design plan.

		Week									
		1	2	3	4	5	6	7	8	9	10
Task	Define Solution Applications	■						■			
	Draw Schematics, Purchase	■							■		
	Part Exploration		■						■		
	Component Communication			■						■	
	Test				■	■				■	
	Debugging						■				■

Figure 6: Table of Design Work to Be Completed

The actual design process followed this schedule exactly. During the first 6 weeks, the main focus was on the digital accelerometer. However, once the sixth week concluded and it was decided that it was too difficult to work with SPI in this short amount of time, the solution was redefined to include the analog accelerometer. After this was decided, the rest of the plan fell into place and resulted in a functional wireless analog accelerometer.

## Design Review

After a very brief design review it was decided that if the wireless accelerometer system is developed correctly and functioning as expected a gyroscope element could be added to the system. This would make certain that the acceleration readings are all consistent by keeping the orientation of the accelerometer at a constant. Therefore z, x, and y-direction acceleration would be in that true direction the entire duration of the test. If there is no gyroscope and the accelerometer is testing a roller coaster model as mentioned earlier, the directions of acceleration will change with the orientation of the accelerometer. Such as when the accelerometer is going uphill the z-direction will be different than the z-direction when the accelerometer is on flat track. This would make the wireless accelerometer system much more accurate and more valuable.

The status review meetings proved to be extremely valuable in keeping the group on track. It was important to get weekly feedback on our project and to get suggestions. For example, it was recommended that we use an analog accelerometer after our initial digital design was not working. The digital design was difficult to debug and we decided that an analog device would be easier to track problems, and therefore easier to fix.

## Future Planned Developments

Since the one quarter of work for development of the project was a fairly short time, there were many other developments that could have been approached for the project. These developments are listed as follows:

- Implement a gyroscope into the circuit to get true acceleration readings to account for the rotation of the accelerometer during testing.
- Develop a program to manipulate the data for calculation of velocity and position in three dimensions, possibly modeling the motion in three dimensional space.
- Minimize the size of the entire circuit by combining the target board and the accelerometer of the endpoint to a single fabricated printed circuit board.
- Make a more detailed and user friendly GUI that can be used to set the range of accelerometer and that can start and stop the data collection with a push of the button.

## Conclusion

The design and development of a fairly simple and affordable wireless accelerometer network could prove to have extremely important and endless applications. A wireless accelerometer can be designed using TI components, such as a microcontroller and transceiver, and an accelerometer. It is important to keep the elements of the system physically small to make certain that it can be applied to most situations where acceleration should be measured. One application used to develop the size and performance of the system was a model rollercoaster. In this application the accelerometer must be small, lightweight, and able to measure up to 6g. Also, if the microcontroller programming is proven to be fairly simple, it could be used in a high school setting to get students interested in the fields of engineering and physics. The wireless accelerometer network could retrieve data on the acceleration of a certain object and then provide this data to a computer to run calculations on the data and determine much more important information about the system under test. The wireless accelerometer network could be a very valuable circuit that could be used for many important applications.

## Appendix: Code

### C Code Used in IAR Embedded Workbench

#### Access Point

```
//Access Point File

#include "bsp.h"
#include "mrfi.h"
#include "bsp_leds.h"
#include "bsp_buttons.h"
#include "nwk_types.h"
#include "nwk_api.h"
#include "nwk_frame.h"
#include "nwk.h"

#include "msp430x22x4.h"
#include "vlo_rand.h"

#define MESSAGE_LENGTH 6
void TXString( char* string, int length );
void MCU_Init(void);
void transmitData(char msg[MESSAGE_LENGTH] );
void transmitDataString(char msg[MESSAGE_LENGTH]);
void createRandomAddress();

//data for terminal output
const char splash[] = {"\r\n-----\r\n
****\r\n      eZ430-RF2500\r\n      *****o***** Temperature Sensor
Network\r\n*****_//_**** Copyright 2007\r\n *****/_//_***** Texas Instruments
Incorporated\r\n ** **(_/***** All rights reserved.\r\n *****
Version 1.02\r\n      *****\r\n      ***\r\n-----
-----\r\n"};

__no_init volatile char Flash_Addr[4] @ 0x10F0; // Flash address set randomly

// reserve space for the maximum possible peer Link IDs
static linkID_t sLID[NUM_CONNECTIONS];
static uint8_t sNumCurrentPeers;

// callback handler
static uint8_t sCB(linkID_t);

// work loop semaphores
static uint8_t sPeerFrameSem;
static uint8_t sJoinSem;
//static uint8_t sSelfMeasureSem;
int run;

void main (void)
{
    addr_t lAddr;
    bspIState_t intState;

    WDTCTL = WDTPW + WDTHOLD; // Stop WDT

    // delay loop to ensure proper startup before SimpliciTI increases DCO
```

```

// This is typically tailored to the power supply used, and in this case
// is overkill for safety due to wide distribution.
volatile int i;
for(i = 0; i < 0xFFFF; i++){

if( CALBC1_8MHZ == 0xFF ) // Do not run if cal values are erased
{
    volatile int i;
    P1DIR |= 0x03;
    BSP_TURN_ON_LED1();
    BSP_TURN_OFF_LED2();
    while(1)
    {
        for(i = 0; i < 0x5FFF; i++){
            BSP_TOGGLE_LED2();
            BSP_TOGGLE_LED1();
        }
    }

BSP_Init();

if( Flash_Addr[0] == 0xFF &&
    Flash_Addr[1] == 0xFF &&
    Flash_Addr[2] == 0xFF &&
    Flash_Addr[3] == 0xFF )
{
    createRandomAddress(); // set Random device address at initial
startup
}
lAddr.addr[0]=Flash_Addr[0];
lAddr.addr[1]=Flash_Addr[1];
lAddr.addr[2]=Flash_Addr[2];
lAddr.addr[3]=Flash_Addr[3];
SMPL_Ioctl(IOCTL_OBJ_ADDR, IOCTL_ACT_SET, &lAddr);

MCU_Init();
//Transmit splash screen and network init notification
TXString( (char*)splash, sizeof splash);
TXString( "\r\nInitializing Network....", 26 );

SMPL_Init(sCB);

// network initialized
TXString( "Done\r\n", 6);

//Some of our original code begins here

BSP_TOGGLE_LED2();
// main work loop
while (1)
{
    // Wait for the Join semaphore to be set by the receipt of a Join frame from a
    // device that supports and End Device.

    if (sJoinSem && (sNumCurrentPeers < NUM_CONNECTIONS))
    {
        // listen for a new connection
        SMPL_LinkListen(&sLID[sNumCurrentPeers]);
        sNumCurrentPeers++;
        BSP_ENTER_CRITICAL_SECTION(intState);
        if (sJoinSem)
        {

```

```

        sJoinSem--;
    }
    BSP_EXIT_CRITICAL_SECTION(intState);
}

// Have we received a frame on one of the ED connections?
// No critical section -- it doesn't really matter much if we miss a poll
if (sPeerFrameSem)
{
    uint8_t    msg[MAX_APP_PAYLOAD], len, i;

    // process all frames waiting
    for (i=0; i<sNumCurrentPeers; ++i)
    {
        if (SMPL_Receive(sLID[i], msg, &len) == SMPL_SUCCESS)
        {
            ioctlRadioSiginfo_t sigInfo;
            sigInfo.lid = sLID[i];
            SMPL_Ioctl(IOCTL_OBJ_RADIO, IOCTL_ACT_RADIO_SIGINFO, (void *)&sigInfo);
            transmitData( (char*)msg );
            BSP_TOGGLE_LED2();
            BSP_ENTER_CRITICAL_SECTION(intState);
            sPeerFrameSem--;
            BSP_EXIT_CRITICAL_SECTION(intState);
        }
    }
}
}

/*-----*/
*
-----*/
void createRandomAddress()
{
    unsigned int rand, rand2;
    do
    {
        rand = TI_getRandomIntegerFromVLO();    // first byte can not be 0x00 of 0xFF
    }
    while( (rand & 0xFF00)==0xFF00 || (rand & 0xFF00)==0x0000 );
    rand2 = TI_getRandomIntegerFromVLO();

    BCCTL1 = CALBC1_1MHZ;                // Set DCO to 1MHz
    DCOCTL = CALDCO_1MHZ;
    FCTL2 = FWKEY + FSSEL0 + FN1;        // MCLK/3 for Flash Timing Generator
    FCTL3 = FWKEY + LOCKA;               // Clear LOCK & LOCKA bits
    FCTL1 = FWKEY + WRT;                 // Set WRT bit for write operation

    Flash_Addr[0]=(rand>>8) & 0xFF;
    Flash_Addr[1]=rand & 0xFF;
    Flash_Addr[2]=(rand2>>8) & 0xFF;
    Flash_Addr[3]=rand2 & 0xFF;

    FCTL1 = FWKEY;                       // Clear WRT bit
    FCTL3 = FWKEY + LOCKA + LOCK;        // Set LOCK & LOCKA bit
}

/*-----*/
*
-----*/
void transmitData(char msg[MESSAGE_LENGTH] )
{

```

```

    transmitDataString( msg );
}

/*-----
*
-----*/
void transmitDataString(char msg[MESSAGE_LENGTH] )
{
    char x_string[] = {"XXXX"};
    char y_string[] = {"XXXX"};
    char z_string[] = {"XXXX"};

    int tempx = msg[0] + (msg[1]<<8);
    x_string[3] = '0' + (tempx%10);
    x_string[2] = '0' + ((tempx/10)%10);
    x_string[1] = '0' + ((tempx/100)%10);
    x_string[0] = '0' + ((tempx/1000)%10);

    int tempy = msg[2] + (msg[3]<<8);
    y_string[3] = '0' + (tempy%10);
    y_string[2] = '0' + ((tempy/10)%10);
    y_string[1] = '0' + ((tempy/100)%10);
    y_string[0] = '0' + ((tempy/1000)%10);

    int tempz = msg[4] + (msg[5]<<8);
    z_string[3] = '0' + (tempz%10);
    z_string[2] = '0' + ((tempz/10)%10);
    z_string[1] = '0' + ((tempz/100)%10);
    z_string[0] = '0' + ((tempz/1000)%10);

    //char output[] = {"\r\nX: XXXX, Y: XXXX, Z: XXXX"};
    char output[] = {"\r\nXXXX XXXX XXXX"};
    output[2] = x_string[0];
    output[3] = x_string[1];
    output[4] = x_string[2];
    output[5] = x_string[3];

    output[7] = y_string[0];
    output[8] = y_string[1];
    output[9] = y_string[2];
    output[10] = y_string[3];

    output[12] = z_string[0];
    output[13] = z_string[1];
    output[14] = z_string[2];
    output[15] = z_string[3];

    TXString(output, sizeof output );
}

/*-----
*
-----*/
void TXString( char* string, int length )
{
    int pointer;
    for( pointer = 0; pointer < length; pointer++)
    {
        volatile int i;
        UCA0TXBUF = string[pointer];
        while (!(IFG2&UCA0TXIFG));           // USCI_A0 TX buffer ready?
    }
}

```

```

/*-----
*
-----*/
void MCU_Init()
{
    BCSCTL1 = CALBC1_8MHZ;           // Set DCO
    DCOCTL = CALDCO_8MHZ;

    BCSCTL3 |= LFXT1S_2;           // LFXT1 = VLO
    TACCTL0 = CCIE;                // TACCR0 interrupt enabled
    TACCR0 = 2000;                 // ~1/6 second
    TACTL = TASSEL_1 + MC_1;       // ACLK, upmode

    P3SEL |= 0x30;                 // P3.4,5 = USCI_A0 TXD/RXD
    UCA0CTL1 = UCSSEL_2;           // SMCLK
    UCA0BR0 = 0x41;                // 9600 from 8Mhz
    UCA0BR1 = 0x3;
    UCA0MCTL = UCBRS_2;
    UCA0CTL1 &= ~UCSWRST;          // **Initialize USCI state machine**
    IE2 |= UCA0RXIE;              // Enable USCI_A0 RX interrupt
    __enable_interrupt();
}
/*-----
* Runs in ISR context. Reading the frame should be done in the
* application thread not in the ISR thread.
-----*/
static uint8_t sCB(linkID_t lid)
{
    if (lid)
    {
        sPeerFrameSem++;
    }
    else
    {
        sJoinSem++;
    }
    // leave frame to be read by application.
    return 0;
}

/*-----
* ADC10 interrupt service routine
-----*/

#pragma vector=ADC10_VECTOR
__interrupt void ADC10_ISR(void)
{
    __bic_SR_register_on_exit(CPUOFF); // Clear CPUOFF bit from 0(SR)
}

/*-----
* Timer A0 interrupt service routine
-----*/

#pragma vector=TIMER_A0_VECTOR
__interrupt void Timer_A (void)
{
    run = 1;}

```

## End Device

```
//End Device File

#include "bsp.h"
#include "mrfi.h"
#include "nwk_types.h"
#include "nwk_api.h"
#include "bsp_leds.h"
#include "bsp_buttons.h"
#include "vlo_rand.h"

void linkTo(void);
void MCU_Init(void);

__no_init volatile char Flash_Addr[4] @ 0x10F0; // Flash address set randomly

void createRandomAddress();

void main (void)
{
    addr_t lAddr;
    WDTCTL = WDTPW + WDTHOLD; // Stop WDT
    {
        // delay loop to ensure proper startup before SimpliciTI increases DCO
        // This is typically tailored to the power supply used, and in this case
        // is overkill for safety due to wide distribution.
        volatile int i;
        for(i = 0; i < 0xFFFF; i++){
        }
    }
    if( CALBC1_8MHZ == 0xFF ) // Do not run if cal values are erased
    {
        volatile int i;
        P1DIR |= 0x03;
        BSP_TURN_ON_LED1();
        BSP_TURN_OFF_LED2();
        while(1)
        {
            for(i = 0; i < 0x5FFF; i++){
                BSP_TOGGLE_LED2();
                BSP_TOGGLE_LED1();
            }
        }
    }

    // SimpliciTI will change port pin settings as well
    P1DIR = 0xFF;
    P1OUT = 0x00;
    P2DIR = 0x20;
    //P2OUT = 0x00;
    P3DIR = 0xC0;
    P3OUT = 0x00;
    P4DIR = 0xFF;
    P4OUT = 0x00;
    //P2SEL = 0x07;
    P4OUT |= 0x08; //Sets Pin 8 (P4.3) high, turns on the accelerometer, set low to put
to sleep

    BSP_Init();

    if( Flash_Addr[0] == 0xFF &&
        Flash_Addr[1] == 0xFF &&
        Flash_Addr[2] == 0xFF &&
        Flash_Addr[3] == 0xFF )
```

```

    {
        createRandomAddress(); // set Random device address at initial
startup
    }
    lAddr.addr[0]=Flash_Addr[0];
    lAddr.addr[1]=Flash_Addr[1];
    lAddr.addr[2]=Flash_Addr[2];
    lAddr.addr[3]=Flash_Addr[3];
    SMPL_Ioctl(IOCTL_OBJ_ADDR, IOCTL_ACT_SET, &lAddr);
    BCSCTL1 = CALBC1_8MHZ; // Set DCO after random function
    DCOCTL = CALDCO_8MHZ;

    BCSCTL3 |= LFXTS_2; // LFXT1 = VLO

    // keep trying to join until successful. toggle LEDS to indicate that
    // joining has not occurred. LED3 is red but labeled LED 4 on the EXP
    // board silkscreen. LED1 is green.
    while (SMPL_NO_JOIN == SMPL_Init((uint8_t (*)(linkID_t))0))
    {
        BSP_TOGGLE_LED1();
        BSP_TOGGLE_LED2();
        __bis_SR_register(LPM3_bits + GIE); // LPM3 with interrupts enabled
    }
    // unconditional link to AP which is listening due to successful join.
    linkTo();
}

void createRandomAddress()
{
    unsigned int rand, rand2;
    do
    {
        rand = TI_getRandomIntegerFromVLO(); // first byte can not be 0x00 of 0xFF
    }
    while( (rand & 0xFF00)==0xFF00 || (rand & 0xFF00)==0x0000 );
    rand2 = TI_getRandomIntegerFromVLO();

    BCSCTL1 = CALBC1_1MHZ; // Set DCO to 1MHz
    DCOCTL = CALDCO_1MHZ;
    FCTL2 = FWKEY + FSSEL0 + FN1; // MCLK/3 for Flash Timing Generator
    FCTL3 = FWKEY + LOCKA; // Clear LOCK & LOCKA bits
    FCTL1 = FWKEY + WRT; // Set WRT bit for write operation

    Flash_Addr[0]=(rand>>8) & 0xFF;
    Flash_Addr[1]=rand & 0xFF;
    Flash_Addr[2]=(rand2>>8) & 0xFF;
    Flash_Addr[3]=rand2 & 0xFF;

    FCTL1 = FWKEY; // Clear WRT bit
    FCTL3 = FWKEY + LOCKA + LOCK; // Set LOCK & LOCKA bit
}

void linkTo()
{
    linkID_t linkID1;
    uint8_t msg[6];

    // keep trying to link...
    while (SMPL_SUCCESS != SMPL_Link(&linkID1))
    {
        __bis_SR_register(LPM3_bits + GIE); // LPM3 with interrupts enabled
        BSP_TOGGLE_LED1();
        BSP_TOGGLE_LED2();
    }
}

```

```

}

// Turn off all LEDs
if (BSP_LED1_IS_ON())
{
    BSP_TOGGLE_LED1();
}
if (BSP_LED2_IS_ON())
{
    BSP_TOGGLE_LED2();
}
while (1)
{
    char msg[6];
    int data_x,data_y,data_z;
    volatile long temp;
    int results[3];
    SMPL_Ioctl( IOCTL_OBJ_RADIO, IOCTL_ACT_RADIO_SLEEP, "" );
    //__bis_SR_register(LPM3_bits+GIE); // LPM3 with interrupts enabled
    SMPL_Ioctl( IOCTL_OBJ_RADIO, IOCTL_ACT_RADIO_AWAKE, "" );

    //read x: x is A0 which is P2.0
    BSP_TOGGLE_LED2();
    ADC10CTL1 = INCH_0 + ADC10DIV_2; //set ADC to A0 with clock divide of 2, we might
not need the clock divide
    ADC10CTL0 = SREF_0 + ADC10SHT_3 + ADC10ON + ADC10IE; //using Vcc and gnd for
reference, 16 clks/sample, turn ADC on and enable interrupts
    //for( int a = 100; a > 0; a-- ); // delay to allow reference to settle, might
not be necessary
    ADC10CTL0 |= ENC + ADC10SC; // Sampling and conversion start
    __bis_SR_register(CPUOFF + GIE); // LPM0 with interrupts enabled, this means
that it enters low power until ADC conversion is done
    results[0] = ADC10MEM; //copy ADC results to x data
    ADC10CTL0 &= ~ENC; //reset encode bit

    //read y: y is A1 which is P2.1
    ADC10CTL1 = INCH_1 + ADC10DIV_2; //set ADC to A1 with clock divide of 2, we might
not need the clock divide
    ADC10CTL0 = SREF_0 + ADC10SHT_3 + ADC10ON + ADC10IE; //using Vcc and gnd for
reference, 16 clks/sample, turn ADC on and enable interrupts
    //for( int a = 100; a > 0; a-- ); // delay to allow reference to settle, might
not be necessary
    ADC10CTL0 |= ENC + ADC10SC; // Sampling and conversion start
    __bis_SR_register(CPUOFF + GIE); // LPM0 with interrupts enabled, this means
that it enters low power until ADC conversion is done
    results[1] = ADC10MEM; //copy ADC results to y data
    ADC10CTL0 &= ~ENC; //reset encode bit

    //read z: z is A2 which is P2.2
    ADC10CTL1 = INCH_2 + ADC10DIV_2; //set ADC to A2 with clock divide of 2, we might
not need the clock divide
    ADC10CTL0 = SREF_0 + ADC10SHT_3 + ADC10ON + ADC10IE; //using Vcc and gnd for
reference, 16 clks/sample, turn ADC on and enable interrupts
    //for( int a = 100; a > 0; a-- ); // delay to allow reference to settle, might
not be necessary
    ADC10CTL0 |= ENC + ADC10SC; // Sampling and conversion start
    __bis_SR_register(CPUOFF + GIE); // LPM0 with interrupts enabled, this means
that it enters low power until ADC conversion is done
    results[2] = ADC10MEM; //copy ADC results to z data
    ADC10CTL0 &= ~ENC; //reset encode bit
}

```

```

ADC10CTL0 &= ~(REFON + ADC10ON);      // turn off A/D to save power

/*
The analog data is converted using the following formula:

Digital = 1023*(V analog - Vref-) / (Vref+ - Vref-)
The 1023 comes from the 10 bit ADC (2^10)
For our case as it's setup, Vref- = Gnd and Vref+ = Vcc
An input voltage of Vcc/2 corresponds to zero acceleration
so zero acceleration is 512 in digital (1023/2 rounded up, not sure if supposed
to round up)
*/

data_x = results[0];
data_y = results[1];
data_z = results[2];

msg[0] = data_x&0xFF;
msg[1] = (data_x>>8)&0x03;
msg[2] = data_y&0xFF;
msg[3] = (data_y>>8)&0xFF;
msg[4] = data_z&0xFF;
msg[5] = (data_z>>8)&0xFF;
BSP_TOGGLE_LED1();

if (SMPL_SUCCESS == SMPL_Send(linkID1, msg, sizeof(msg)))
{
    BSP_TOGGLE_LED2();
}
else
{
    BSP_TOGGLE_LED2();
    BSP_TOGGLE_LED1();
}
}

/*-----
* ADC10 interrupt service routine
-----*/
#pragma vector=ADC10_VECTOR
__interrupt void ADC10_ISR(void)
{
    __bic_SR_register_on_exit(CPUOFF);      // Clear CPUOFF bit from 0(SR)
}

```

## MATLAB Code to Display Plots

```
%serial data capture and plot
clear all
s = serial('COM1');
set(s, 'BaudRate',9600, 'DataBits',8, 'Parity', 'none');%, 'InputBufferSize',28);
fopen(s);
s.ByteOrder = 'bigEndian';
l = 200;
i = 2;
t = [];
x = [];
y = [];
z = [];
t(1) = 0;
data = fgets(s);
data = fgets(s);
tic
data2 = str2num(data);
x(1) = data2(1);
y(1) = data2(2);
z(1) = data2(3);
while (i < (l + 2))
    data = fgets(s);
    data = fgets(s);
    t(i) = t(i-1) + toc;
    tic
    data2 = str2num(data);
    x(i) = data2(1);
    y(i) = data2(2);
    z(i) = data2(3);
    i = i + 1;
    disp(data)
end
n = toc;
fclose(s);delete(s);clear s
x = (x - 474)./85.3;
y = (y - 481)./85.3;
z = (z - 478)./85.3;
subplot(3,1,1)
plot(t,x)
axis([0 t(1) -6 6])
subplot(3,1,2)
plot(t,y)
axis([0 t(1) -6 6])
subplot(3,1,3)
plot(t,z)
axis([0 t(1) -6 6])
```