Portable Range Doppler Radar System

Michigan State University
ECE 480
Fall Semester 2011
Design Team 7

Shaun Eisenmenger
Micheal Zito
Adam Rogacki
Enwei Gu

Project Sponsor: Gregory L. Charvat from Massachusetts Institute of Technology Lincoln Laboratory
Executive Summary

This Report goes into detail about how our team decided to go about tackling the issue of a Data Acquisition Unit to expand the functionality of a Laptop-based RADAR system already developed and in use by MIT-LL. The system is a low-cost system that is mainly used in a short-course taught at MIT about RADAR systems and their basic functionality. To keep with the original design requirements of the MIT system, our addition must be as low power as possible, and provide a higher resolution than the current data gathering mechanism (utilizing the Laptop’s Line-in Port). The other high level goal is to provide the data in as real time as possible via a GUI which the user can interact with. Various methods of data acquisition were investigated before the group finally settled on a high resolution Analog-to-Digital Convertor, coupled with a MSP430 Microcontroller. This system will then stream the sampled data to the GUI for data processing. The GUI utilizes C# and was developed using Visual Studio, along with the .NET 4 Framework, and DirectX Functionality.

Acknowledgements

We of Design Team #7, would firstly like to thank Dr. Gregory Charvat, our team Sponsor from MIT-LL. His knowledge and guiding words have truly helped us realize our goals this term to the best of our abilities. We would also like to thank Dr. Joydeep Mitra, of Michigan State University. He was our team facilitator during this project, and helped keep us on track along with connecting us with the great knowledgebase which we have here as students at MSU. We would like to thank the electromagnetics department here at MSU, including Dr. Prem Chahal, and Dr. Edward Rothwell, for connecting us with their graduate students in order to tune our Cantennas. We would also like to thank the other graduate students that came to our aid in trying to troubleshoot our USB to SPI convertor system.

Lastly, Adam Rogacki would like to thank Steven Rogacki, and his company EngXT for providing Adam with C# example code for GUI manipulation, and FFT code, and pseudocode to improve upon.
# Contents

Chapter 1 - Introduction .................................................................................................................. 3

Chapter 2 – Solution Space Analysis, Approach Selection ............................................................... 5

Chapter 3 – Technical Descriptions ................................................................................................ 6

  ADC, MCU, and USB Adapter ....................................................................................................... 6

  RADAR Theory ........................................................................................................................... 12

  GUI Design Requirements .......................................................................................................... 17

Chapter 4 – Test Data and proof of a functional design ................................................................. 19

  ADC, MCU and USB adapter ....................................................................................................... 19

  RADAR Hardware Testing and Validation .................................................................................. 20

Chapter 5 - Conclusions .................................................................................................................. 26

  Data Acquisition Unit ................................................................................................................ 26

  Conclusions Regarding GUI ....................................................................................................... 26

  Final Cost .................................................................................................................................. 27

  Other Future Work ..................................................................................................................... 27

USB Timing – Issues for the Future .............................................................................................. 28

Appendix 1 – Technical Roles, Responsibilities, and Work Accomplished ................................... 30

Appendix 2 – Literature and Website References .......................................................................... 33

Appendix 3 – Detailed technical attachments .............................................................................. 35
Chapter 1 - Introduction

Security concerns have prompted a sharp increase in demand for low-cost, low-power detection and imaging systems. Our goal is to create a laptop based low-cost, low-power range-Doppler imaging system based on an existing coffee can “cantenna” design. The actual radar hardware is based on an existing design that is currently being used in a Massachusetts Institute of Technology Lincoln Laboratory Opencourseware course. Because this system is to be used by students, low cost, and low power consumption are critical requirements. The existing design sends analog radar data to a PC via the audio line-in port. The data is saved on the PC as a .wav file. The user can then process the data using a MATLAB script to display range and velocity measurements. A problem with this set-up is that the data cannot be processed in real time. The user must record data, save the data, then process the data at a later time.

Our project is to solve this issue by implementing real time data acquisition and signal processing for the radar. The final system shall be capable of plotting, in real time, the range vs. Time, and Doppler (or Velocity) vs. Time spectrum of moving targets at ranges that vary between 0-100m. A graphical user interface (GUI) will be developed on a laptop specifically to display this information. The resolution of the system needs to be sufficiently high as to allow the user, and or GUI to identify the targeted object as a pedestrian, bicycle, automobile or other moving target. The GUI will classify these objects based on their velocities. For the system to be better able to differentiate objects, a 16-bit analog-to-digital converter will be required. Despite the improved performance, the final design must maintain the low cost, low power consumption characteristics of the original design.
In order to achieve these goals, and achieve higher resolution, the audio line-in will no longer be used to transfer analog data from the radar to the PC. Instead, a serial communication (in this case (USB) port will be used. Raw analog data will be sent from the radar to a microcontroller, which will send the data to an external analog-to-digital converter (ADC), which sends digital data to the microcontroller. The microcontroller then sends the digital data to the PC via USB, where the signal processing will take place in real time. The microcontroller that was selected was the MSP-430 because of its ease of use, low cost, fast processing capabilities, and low power consumption. Likewise, the ADC that was selected for use in the project is the Linear Technologies LTC2382-16 because of its low-power consumption, high sampling rate, and 16-bit resolution.

Another component of the system that will need to be redesigned is the signal processing component. MATLAB is simply unable to process the data at a high enough rate to produce real-time range and velocity plots. To solve this problem, the signal processing and plotting functions will be written in C# code. Open-source Fast Fourier Transform (FFT) algorithms are widely available online and can be easily integrated for use in Microsoft Visual Studio. The GUI will be designed using C#, which is highly supported, and packages for plotting data are integrated within the .NET framework which is present in every Windows-based system.

The success of this project will result in a more powerful and easier to use portable laptop-based radar system. The user will be able to get instant range and velocity data, at a higher resolution than the current design. This will make the radar more useful, as it can be used to track objects as they are moving, not after the object has passed. The final system will also be just as portable as the original design, while increases in cost and power consumption will be minimized. The increased performance however, will more than justify the slightly increased cost and power consumption.
Chapter 2 – Solution Space Analysis, Approach Selection

There are several different components to the final design of the portable laptop based radar system. The FAST diagram below displays graphically, the functional requirements of the project.

![Figure 3 Portable laptop based radar system FAST diagram](image)

The main function of the portable laptop based radar system is to track objects. This will be done by displaying graphics that the user can see and interpret. In order to produce graphics, velocity, position must be measured. This is accomplished by acquiring and processing signals from the RADAR system. These are then processed via FFT algorithms. In order to acquire usable data, signals must first be transmitted, then received and converted from analog signal to digital.

Other, non-functional requirements were given to the team by the project sponsors. The first non-functional requirement is safety. The system utilizes and S-band radar, which operates in the range of 2.4-2.5 GHz. This is also known as the S Band, or “wifi-band”, as it is a range of frequencies commonly used in wifi networks. There is little evidence to suggest that exposure to such frequencies of electromagnetic radiation pose any significant health risk to humans or wildlife, especially at the low output power levels utilized by this design.

Some electrical components, particularly lead-based solder, can pose a significant health risk if ingested. For that reason, it is expected that all team members wash their hands with hot soapy water after handling any of the electronic components.

Another design requirement is low power consumption. The system is designed to be portable, so large, heavy battery packs are highly undesirable. In the interest of low power
consumption and portability, the radar device, microcontroller and ADC unit shall be powered by two battery packs, each consisting of four standard 1.5V AA batteries.

As mentioned above, portability is another critical design constraint. The radar unit, which includes the circuitry and coffee can antennas is mounted on a 12in x 12 in wooden board. The microcontroller and ADC unit can both be mounted on the same wooden board, which will allow the final design to maintain the same 12in x 18in footprint as the previous design. The extra six inches comes from the coffee can antennas, which stick out about six inches from the edge of the board. The microcontroller and ADC unit are extremely lightweight, so their contributions to the weight of the device is minimal.

**Chapter 3 – Technical Descriptions**

**ADC, MCU, and USB Adapter**

Requirements for the data acquisition design:

- Analog source signal
- Digitalized destination signal
- Integrated with PC
- 16-bit resolution
- Data rate of 80K+ samples per second

Design realizations

- External analog to digital converter
- Low power microcontroller
- Microcontroller to USB adaptor

Description of hardware components

- ADC (LTC2382-16)

In order to achieve the requirements of 16-bit resolution and 80K+ samples per second data rate, we choose an external ADC (LTC®2382-16) as analog to digital converting unit. The LTC®2382-16, from Linear Technology, is a low noise, low power, high speed 16-bit successive approximation register ADC.
Besides, it has 500 Ksps data rate which is much higher than that of our requirement, and good enough for our design.

•Microcontroller (MSP430G2452)

MSP430G2452 is a member of the Texas Instruments MSP430™ family of ultra-low-power microcontrollers. It consists of several devices featuring different sets of peripherals targeted for various applications. Our radar system is a battery-based design, so there is no doubt that power saving is one of the most important considerations during our design process. MSP430G2452 is such an ultra-low power microcontroller that only costs 220 μA in active mode, which will fairly meet our requirement.

•USB adaptor (MAX3420E Evaluation kit-2)

According to the requirement, we have to realize 16-bit resolution with 80K+ sampler per second. This means the data link of the data acquisition unit should have the capability to handle 1.28 Mbit/s (16*80K) at least. In order to achieve this capability, the USB connection is coming to our sight. The MAX3420E evaluation kit-2 provides a proven design to evaluate the MAX3420E USB peripheral controller with SPI interface. It adds USB functionality to our MSP430G2452 microcontroller with an SPI master interface. Besides, it complies with USB 2.0 (full speed) and gives us 12 Mbit/s of bandwidth. This adaptor can guarantee the connection between the microcontroller and PC.

Hardware Implementation

•Connection between LTC2382-16 and MSP430G2452

Figure 1 shows the pin-connections between MCU and ADC. The ADC works in Normal Mode, and as a slave. MCU is configured as a master and provides clock signal as well as control signal (SS) to the ADC. As a result, ADC sends back converted data from its SDO pin to SDI pin of MCU.

![Figure 4 - pin-connections between ADC and MUC for SPI communication](image)
• Connection between MSP430G2452 and MAX3420E Evaluation kit-2

In order to simplify our task, the communication between MSP430G2452 and MAX3420E is configured as half-duplex mode. The MAX3420E is put into half-duplex mode at power on, or when the SPI master clears the FDUPSPI bit. In half-duplex mode, only three wires are needed for the talking, they are SS, SCLK and MOSI. Figure 3 shows the pin-connections between ADC, MCU and MAX3420E.
Programming for microcontroller to communicate with ADC

• Introduction to SPI

The Serial Peripheral Interface (SPI) bus is a synchronous serial data link that was introduced by Motorola and is the simplest communication protocol in general use. Devices communicate in master and slave mode where the master device initiates the data frame and provides clock signals to slaves. In its full form SPI requires four wires and transmits data simultaneously in both directions between two devices.

• Operation of SPI

The SPI can operate with a single master device and with one or more slave devices. However, for our project, the SPI communication is single master and single slave (Figure 5). The microcontroller acts as master and ADC acts as slave. The slave will be activated and prepared to work, when SS pin was low. And master provides clock signals (SCLK) to the slave in order to synchronous the data transmission.
Two shift registers can be used to represent the concept of SPI data transmission (Figure 6). Each device places a new bit on its output from the most significant bit of the shift register when the clock has negative edge and reads its input into the lsb of the shift register on a positive edge of the clock. As a result, one bit of data is transferred in each direction during one clock cycle. After eight clock cycles, the contents of the shift registers have been exchanged and transfer is complete.

![Figure 9 - SPI data transmission represented by two registers](Image)

**SPI clock polarity and phase**

The master must configure the clock polarity (CPOL) and phase (CPHA) with respect to the data based on the property of slave. Table 1 describes the four different modes of CPOL and CPHA. Figure 7 shows a complete transfer of 4 bits using SPI in mode 3:

- CPHA = 0: Read (written) on the leading (trailing) edge of each clock pulse.
- CPHA = 1: Written (read) on the leading (trailing) edge of each clock pulse.
- CPOL = 0: Clock idles low between transfers.
- CPOL = 1: Clock idles high between transfers.

<table>
<thead>
<tr>
<th>Mode</th>
<th>CPOL</th>
<th>CPHA</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>2</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>3</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>

*Table 1- The four standard modes for the clock in SPI*
• ADC conversion timing using the SPI

The simplest way to operate the LTC2382-16 is to set it in Normal Mode with single device. In this mode, CHAIN and RDL/SDI connected to ground, SDO is enabled and the MSB(D15) of the new conversion data available at the falling edge of BUSY. Then, MCU can use SCK to take data from SDO pin of ADC (Figure 8).

• Coding for microcontroller

P2 is configured as an output and drove it high while the USI is active to act as CNV(SS). P7 provides clock signal in order to synchronous the whole system. P15 is the data entrance of MCU. The followings are the details on software configuration:

- Input clock-

• The SMCLK is set to operate in 16MHz.

- USI(SPI mode)-
The USIMST bit is set because this is an SPI master.

The USIWRST bit is set during the write to USICTL0 to prevent any unwanted activity.

The USICKPH bit is set, so it is equivalent to CPHA = 0, which means read on the leading edge and written on the trailing edge.

The clock is taken from SMCLK(USISSEL_2) and divided by 1(USIDIV_0). It idles low, so clear USICKPL bit.

After releasing USI by clearing USISWRST, _low_power_mode_0( ) put the MCU into LPM0 sleep mode and wait for interrupt.

The USI16B is set in order to take 16-bit data from ADC.

**RADAR Theory**

Radars are systems that are used for detecting objects. They accomplish this goal by using electromagnetic waves to determine a variety of detection parameters. Such parameters include altitude, speed, range and direction. The project that we choose uses range and Doppler (direction) to detect objects. But how do you detect objects with electromagnetics? There are many different configurations in which a radar can exist. Among the many, the few that are most common include Monopulse Radar, Bistatic Radar, Doppler Radar, and Frequency Modulated Continuous Wave Radar. Of the above stated configurations, this RADAR system uses a FMCW configuration to determine range and direction of an object.

**Figure 12 - Receive and Transmit Antennas**

Figure 12 represents a simple antenna design of the radar system implemented by this project. It consists of two cantennas (antennas made of coffee cans) one being a receiver and one being a transmitter. The radar transmitter induces a time-varying microwave signal along a coaxial cable to the transmit cantenna. The time-varying signal will induce an electrical current and produce electromagnetic radiation. The microwave photons will flow away at the speed of light and reflect off an object. The reflected energy will illuminate the receiver cantenna and induce an electrical current to the radar receiver. Figure 12 shows the basic idea of how this is accomplished:
The radar system begins at Modulator 1. Modulator 1 is the circuitry that drives our FMCW signal; it uses an XR2206 IC that produces high quality sine, square, triangle, ramp and pulse waveforms. Using this IC is ideal for producing our FMCW signal because FMCW radars are capable of slewing up and down with any of the four different wave configurations capable by the IC. However for this application we’ll use a triangle modulated waveform because it will provide us with range and velocity capability.

Modulator 1 has two functions in this system. First, it produces a linear ramp which modulates OSC1 of figure 13 with a variable called Vtune. The Vtune voltage is proportional to our transmit frequency (between 2.4-2.45 GHz). Modulator 1’s linear ramp of Vtune will cause OSC1 to produce a linear FM chirp which is used to transmit and receive the signal. The given ramp up time and triangle wave period is 20ms and 40ms respectively. The magnitude of the ramp is also given; we will set our potentiometers to give us a 100 MHz transmit bandwidth. Secondly, it sends a pulse to our data acquisition device and sends it to the PC to be displayed on our GUI by perform an FFT with it. The Modulator 1 signal is synchronized with the linear ramp. Figure 14 shows the schematic and soldered version of the Modulator 1 circuitry.
After the transmit receiver illuminates a target and receives the reflected signal, it is then passed into our op-amp circuitry, which amplifies the signal. The block diagram of Figure 13 represents our video amplifier (Figure 15). The signal will first enter our gain stage op-amp circuit and amplify our signal about 28dB. The output of our amplified circuit will then be inputted into our 4th order Low Pass Filter. The Low Pass filter configuration is called a Sallen-Kelly LPF. This type of configuration is easy to implement and will not load the circuit like a passive filter would. After the gain and filtering stages, the output is sent to the data acquisition circuitry and processed within the GUI on the PC. A MAX414 quad op-amp with a single-supply configuration to suit the given design requirements, this particular IC is ideal because both the gain stage and LPF can be implemented using active op-amp components.
Lastly, there is the power supply. Two battery packs which hold 4 AA batteries each are used to produce the 6v and 12v dc outputs needed. However, for our circuitry, 12V, 10V, and 5V are needed. This can be accomplished by implementing 2 voltage regulators in the power supply, as seen in figure 16. The RF components are fed by a 5V DC low-dropout regulator. The regulator is fed by 6V from the battery pack. The 10V DC output will power the MAX414 IC and the 12V DC will be used to power the FMCW circuit.
Figure 16 - Power Supply
Figure 17 is a picture of the complete radar system. The components were soldered onto a PCB for a more robust and permanent system.

GUI Design Requirements

As was discussed above, the GUI portion of this project needs to do two main tasks: Complete data acquisition in real time, and display this data to the user. MATLAB is unusable, as it cannot process data fast enough in order to give a real-time (or nearly real-time) plot of the data. Since compiled code works much faster (in most cases) than interpreted code, either C++ or a C derivative language would be desired. C# was settled on because Visual Studio is able to create and modify GUI’s very easily, and there is a lot of supported functionality built into C# via the .NET framework.

However, transitioning over to C# coming from a MATLAB environment is more difficult, since the datatypes are a lot less dynamic, and require more planning to achieve similar functionality. The main signal processing support needed is an efficient FFT algorithm. Unlike MATLAB, C# does not directly include FFT support. However, its functionality can be expanded upon, and a library of FFT and
other Complex Data Format utilities was found and added onto the GUI. The Exocortex Digital Signal Processing library was used for FFT support. Secondary to an FFT algorithm, getting data into the GUI was another difficulty to be surpassed. Though the .NET framework does support serial interfacing, which can be extended to USB support, it does not directly support taking data from Line-in ports. To accomplish this, another 3rd party library was used, which interfaces between the DirectX functionality in Windows, and the C# GUI. This open source library came from The Code Project, and had very good examples to work from.

Once those two tasks were surmounted, the next task was to convert MATLAB code into C# code, and to modify the MATLAB code in order to make it into a real-time version. A basic outline of the code flowchart is included below:

![Basic Flowchart for GUI code](image-url)
Chapter 4 – Test Data and proof of a functional design

ADC, MCU and USB adapter

- Waveform of control and output signals

<table>
<thead>
<tr>
<th>Waveforms</th>
<th>Represented</th>
</tr>
</thead>
<tbody>
<tr>
<td>Orange</td>
<td>CNV(SS)</td>
</tr>
<tr>
<td>Green</td>
<td>BUSY</td>
</tr>
<tr>
<td>Blue</td>
<td>SCK</td>
</tr>
<tr>
<td>Violet</td>
<td>16-bit Data</td>
</tr>
</tbody>
</table>

Table 2 - Waveforms for below Figure

![Waveform Image]

Figure 19 - Waveforms for CNV(SS), BUSY, SCK and data

![Data Points]

Figure 20 - Sampling Rate of ADC
Sampling rate $= 1/\text{period} = 1/4.107\text{us} = 243.487\text{ K samples/sec.}$

- An example of sampling a constant Voltage

![Sampled analog signal(Violet)](image)

**Figure 21 - A sampled analog signal(Violet)**

<table>
<thead>
<tr>
<th>Ref. (V)</th>
<th>Input+(V)</th>
<th>Input-(V)</th>
<th>Result(DEC)</th>
<th>Result(V)</th>
<th>Error%</th>
</tr>
</thead>
<tbody>
<tr>
<td>2.4964</td>
<td>2.3570</td>
<td>0</td>
<td>30851</td>
<td>2.3504</td>
<td>0.281</td>
</tr>
</tbody>
</table>

Table 3 - An Example

Result(V) = (30851/(65536/2))*2.4964 = 2.3504 V

**RADAR Hardware Testing and Validation**

The first items to be tested in the constructed design were the radar circuitry components. This was done to ensure that the radar was working properly so that accurate data could be collected for processing. The first component tested was the ramp output. An oscilloscope was used to verify that the ramp output was a 5 V_{pp} triangle wave, with a 20 ms up-ramp time. A screenshot of the oscilloscope readout has been included below:
The next component of the radar circuitry to be tested was the active filter of Video Amp1. A sin wave generator was connected to the input of the active filter, and a two channel oscilloscope was attached to the input and outputs of the active filter. For the filter to be working properly, attenuation of at least 3 dB must be achieved at 15 kHz. Below is a screenshot of the oscilloscope readout, with the input waveform on top, and the output waveform on the bottom:
A calculation confirming that attenuation greater than 3 dB has been achieved is given below:

\[ G \text{ (dB)} = 20 \times \log \left( \frac{\text{input}}{\text{output}} \right) \]

\[ G(dB) = 20 \times \log \frac{4.882}{8.710} \]

\[ G = -5.028 \text{ dB} \]

The final radar circuitry component to be tested was the gain stage of the Video Amp1. Like the active low-pass filter, a function generator was attached to the input and a 2 channel scope was attached to both the input and outputs of the circuit. High gain was observed from the amplifier, as shown on the oscilloscope screenshot below, with the input waveform on the top and the output waveform on the bottom.
A simple calculation of the gain has been included below:

\[
Gain = \frac{Output}{Input} = \frac{7.867}{0.375}
\]

\[
Gain = 20.978
\]

After the radar circuitry components were tested and proven to be functional, the actual antennas in their coffee can waveguides were tested. The cantenna assemblies consist of a coffee can with a diameter of 3.9in and a length of 5.25in. The antenna is a simple monopole and is mounted in an SMA bulkhead receptacle jack that is in turn, mounted inside the coffee can via a thru hole. The spacing between the monopole and the waveguide back wall was set to 1.8in, which is a quarter-wavelength at 2.4 GHz. Reflection coefficient (S11) measurements were taken for both transmit and receive antennas. In order for the antennas to be considered functional, the measured S11 should be at or below -10dBm. 1-port measurements were taken using a network analyzer. Pictures of the network analyzer results have been included below:
In the above photo, markers 1 and 3 were placed at 2.4106 GHz and 2.5202 GHz, respectively. The respective measured S11s for the two markers are -12.061 dBm and -16.873 dBm. Both markers are display S11 readings that are less than -10 dBm, and both markers combined span the 2.4 Ghz – 2.5 GHz operating range for the radar.

S11 measurements were then taken of the receive antenna. Markers 1 and 3 were placed at 2.4106 GHz and 2.5202 GHz respectively. The respective S11 measurements are -9.9934 dBm and -20.242 dBm. Both markers account for the 2.4 GHz to 2.5 GHz range. It can be concluded that the receive antenna is functioning properly.
The entire radar system was tested using MATLAB scripts created at MITLL for the Opencourseware course. A team member with a metal trashcan stood at the end of a long hallway and walked towards the radar. The metal trashcan has a higher reflection coefficient than a human and thus will allow for a stronger signal to be received by the radar. Raw data from the radar was sent to a PC via audio line-in, recorded using Audacity then saved as a .wav file so that a MATLAB range vs. time plot could be constructed. The results of this test have been included below:

As shown in the plot above, the target started at a distance just less than 60m away from the radar. 45 seconds later, the target was at a distance of around 1m from the radar. The starting and ending positions of the target were measured with a tape measure and were found to match the results of the MATLAB ranging test. The MATLAB code for this test has been included in Appendix 3.

As a Proof of Concept test, of the GUI, the following figure shows the simple FFT test done to show the FFT is working properly. The GUI passes data of a sin curve from 0 to 4\*\pi, and plots the sin curve and the FFT, which is mostly dominated by the imaginary component.
Chapter 5 - Conclusions

Data Acquisition Unit

Achieved 16-bit data output from ADC with 243K samples per second data rate under the control of MUC. A data link established between ADC and MCU with 3.9 Mbit/s transmission rate. However establishing a similar link between the MCU and the PC over a USB interface has been met with difficulties. The MSP430 processor used contains support for a SPI interface. A USB to SPI conversion evaluation board was used, by MAXIM, but configuring it properly has been difficult due to poor documentation and support.

Conclusions Regarding GUI

Beyond the difficulty of learning a completely new language, and figuring out how to pass data throughout the various classes needed for data processing in C#, the GUI design and implementation has been relatively straightforward. Of course, as most implementations composed of learning a new system, the GUI has been behind from the start, and just in late November, it was decided to shift towards supporting the Audio Line-in port in order to show the real-time code works. Future work regarding the GUI could go a number of ways. Make the code more efficient, and memory conscious
would be a primary concern. Also, porting the code out to a more universally supported language would also be a good ‘next step’, since C# will only work on windows systems.

**Final Cost**

Our Radar circuitry was met with success. By mid-October, our protoboard radar circuitry was completed and tested and by late November, our soldered PCB was completed and tested. Therefore, the radar portion of our circuitry met its objectives on time. The budget for our radar circuitry was as follow. The total cost of our project was $470, making our project under budget. The following table shows the list of our expenses.

<table>
<thead>
<tr>
<th>Equipment</th>
<th>Price</th>
</tr>
</thead>
<tbody>
<tr>
<td>Radar</td>
<td>$235.55</td>
</tr>
<tr>
<td>“Can”tennas</td>
<td>$53.48</td>
</tr>
<tr>
<td>Analog Components</td>
<td>$50.00</td>
</tr>
<tr>
<td>SPI to USB Controller</td>
<td>$50.00</td>
</tr>
<tr>
<td>Shipping</td>
<td>$80.00</td>
</tr>
<tr>
<td><strong>Total</strong></td>
<td>$469.03</td>
</tr>
</tbody>
</table>

Table 4 - Budget

Lastly, future projects with this radar system could explore the effective radiation patterns of the signals feeding the antenna. By toughening the radiation pattern of the arrays that are desired and stifling the radiation patterns that are destructive, future groups could study the phased array of the radar system.

**Other Future Work**

If, perhaps, this system were to be commercialized in the future, a business model canvas is a very helpful tool to determine how best to pursue a proper methodology on which to focus a commercialization idea.
USB Timing – Issues for the Future

As a whole, system integration posed a significant challenge to the implementation of serial port communication for our system. Our data acquisition system had 4 main hardware components that all needed to communicate with one another in a constructive manner: a microprocessor, USB controller, analog-to-digital converter (ADC) and a laptop computer. The microprocessor was an MSP430 from Texas Instruments, the ADC chip was from Linear Technologies, the USB controller from MAXIM Electronics and the PC was a Dell Laptop running Windows XP. This made the entire integration process much more difficult than it would have been had there only been one or two vendors for the entire system. For instance, if the USB controller, microprocessor and ADC chips had all come from the same vendor, it would be very likely that they would be designed to be easily integrated with one another, with minimal configuration by the user.

The team was successfully able to enable proper communication and data transfer from the microcontroller and analog-to-digital converter, and the signal processing has been properly set up on the PC. The problem was that the microcontroller could not be configured to communicate to the PC via serial port because the USB controller could not be properly configured. The data from the radar remained “stuck” on the microcontroller.

USB device development is a complicated and time-consuming process and out of our knowledge scope. For the UBS interface, we have to deal with a large amount of settings. Such as Speed
control, power management, endpoints, communication protocol, data packaging, descriptors and so on. In order to achieve those settings, a wide range of coding skills is needed. So, this came an unconquerable barrier for a team that lack of computer science engineerage

Besides, MAX3420E chip has a sophisticated controlling process. The microcontroller could establish USB connection by reading and writing 21 registers in MAX3420E. And those registers are related to USB communication mechanism, like endpoints, data packaging and descriptors that mentioned above. Develop USB communication based on MAX3420E requires a lot of relevant working experience, which is inadequate for our team.

Our project did not meet all of its deliverables. Our microprocessor and ADC chip were communicating as we had proposed but during testing we were using an MSP430 evaluation board that had USB connectivity. However, the speed of our data was slower than that of what was required. Since this was overlooked, our timing in our Gantt chart was underestimated. We only allowed 6 days for our computer radar system to be implemented, where we should have allowed up to 30 days. Additionally, we lost a team member during our project. If we would have had one more person on our team to divvy up the workload, we may have had more time to research and bounce around ideas. Lastly, the complexity of USB interfacing was overlooked, we did not allocate enough time to study USB and how SPI could be implemented to interface with a computer.
Appendix 1 – Technical Roles, Responsibilities, and Work Accomplished

Enwei Gu

Enwei Gu is a Senior at MSU studying Electrical Engineering. His main work, as a member of team-7, was to develop the data acquisition unit for the radar system. The data acquisition unit consisted of an ADC, MUC and USB adapter. In order to complete his work, he worked on four specific tasks, which are analyzed the design requirement, studied on each component, implemented pin-connections for ADC and MCU, and Coding for MCU.

Analyze the design requirement is the first step for the data acquisition unit development. According to sponsor, the minimum data resolution is 16-bit. And he found the frequency of analog signal from the radar circuit is 40 KHz, so he thought the sampling rate must greater than 80 KHz. With those two critical parameters, he reached on different kinds of ADCs and found LTC2382-16, from linear technology, is a reasonable choice for the requirement. On the other hand, he realized power saving is vital to a battery-based radar system. So the MSP430 ultra low power microcontroller is selected as MCU.

The second task he completed is study on ADC and MCU, and figure out the SPI communication for the data link. ADC and MCU have several communication modules, such as UART, I2C and SPI. For the project, high data rate and simplicity are two considerations on choosing an appropriate communication module. After pondering, he decided to take SPI as the communication module between ADC and MCU.

In order to achieve pin-connections for ADC and MCU, he studied on the data sheets for more details, such as electrical characteristics, power requirements, pin configurations and so on. As a result, he mounted those components on a breed-board and successfully achieved the pin-connections for ADC and MCU.

The last task is the coding for MCU. There are several coding tools exists on the web, like IAR and CCS. For the sake of limited budget, he selected the free version of CCS as the coding tool for MCU. After countless coding, testing and recoding, he implemented the talking between ADC and MUC.
After those tasks, he achieved 16-bit data output from ADC with 243K samples per second data rate under the control of MUC. A data link established between ADC and MCU with 3.9 Mbit/s transmission rate.

Besides, he also worked on the radar antenna testing, like measuring resonate frequency and reflection coefficient (S11) of the antenna.

Shaun was primarily responsible for assembling and testing the radar circuitry, RF circuitry and coffee can antennas. Shaun also helped other team members research microcontrollers and ADC chips, examining the feasibility of each option, as well as providing feedback on the C# code used for the signal processing and the graphical user interface.

Testing of the radar circuitry was conducted in the ECE 480 lab and the results of these tests are available in chapter 4 of this report. The bench power supplies were used to power the test circuitry for the majority of the semester. This was done to preserve the AA batteries that were supplied. Battery packs were assembled and tested during the second to last week of the semester to do simple ranging and velocity tests in the hallways of the engineering building. An Agilent Infiniium DSO9064A 4-channel oscilloscope was the primary testing tool used to test the performance of the individual radar circuitry components. A vector network analyzer was utilized, with the assistance of a graduate student to ensure that the coffee can antennas were functioning properly.

Testing the functionality of the entire radar system was also conducted. This testing was done after the individual radar components were tested and proven to be functional. The serial communication portion of the project had not been completed at the time of this test, so the audio line-in had to be used to transfer data from the radar to the PC. The data sent to the PC was recorded in Audacity, saved as a .wav file, then analyzed in MATLAB to produce a plot of range vs. time. The radar was positioned to measure targets as they walked down the hallway. A team member started out a distance of ~60m from the radar and proceeded to walk towards the radar unit. The team member carried a metal trashcan to increase the strength of the signal that would be reflected back to the radar unit.
Michael was responsible for assembling the final radar deliverable. In addition to his final deliverable, Michael also assisted in testing the prototype by using a function generator and oscilloscope in the ECE480 lab to verify op-amp and FMCW circuitry. Additionally, Michael provided feedback on the C# code by helping transposing functions in MATLAB into C# readable code. The C# code was generated for our GUI. Lastly, after communication between microcontroller and ADC came online, Michael assisted in SPI to USB research by reading data sheets for the MAX3042 and searching forums.

Assembling the radar deliverable involved many steps, including time in the machine shop and many hours in the lab testing and troubleshooting. Michael collected all of the necessary components to build the breadboard circuitry and assembled the prototype. Mike also milled the can-tennas and radar components to the wooden board that mounts the system. Michael assisted in testing the radar connectivity with a PC using MATLAB as a graphing program.

After having a working prototype, our sponsor asked us to create a PCB of our circuitry. Michael then spent a few weeks trying to solder the team’s circuitry onto Radio Shack evaluation boards. However, after networking with the other radar team in ECE480 our group obtained a manufactured PCB from MIT. Michael then completed the soldered PCB and tested the functionality in the ECE 480 lab using a function generator and oscilloscope. Testing with the function generator is necessary before trying to test in MATLAB. By inputting a 200mv sine wave at 15KHz into the input stage of the video amplifier, you can observe the gain stage and the low pass filters doing its job. The output of our gain stage was roughly 28dB. Additionally, it was seen that there was a 3db drop at 15KHz from our filters.
Adam was responsible for programming the GUI, as he had previous experience in C++, from previous hobby projects, which is similar in syntax to C#. Adam was the one that first decided on implementing the GUI in wxWidgets, but then once the feasibility and lack of documentation made wxWidgets a non-viable solution, he decided that C# would be the better route to go. This choice was mainly due to the fact that C# is very well supported in a windows environment, and that there is a great deal of example code on the web in which to use. While searching for a better alternative, he wrote an FFT algorithm to test with, as to not hinder programming progress. He also attempted to make the code, and tried to make explanations regarding what the code was accomplishing, open to the rest of the group, and accepted their feedback and suggestions on how the GUI should operate.

Due to his previous experience with the MSP430 launchpad for hobby projects, Adam brought the processor, and its line, to his group’s attention, as it seemed to fit the design requirements related to the processor, and processing power required. The MSP430 also had the advantage of being an extremely low cost solution, and has a relatively easy to use PC programming interface. Similarly, he also researched with his group to ensure that the processor did meet the group’s specifications. In addition, he also collaborated greatly with the group on choosing the proper ADC chip based on processing capability, resolution, and power consumption.

He also assisted in testing the prototype prior to the PCB version, and helped to troubleshoot issues that were encountered. He also helped his group immensely during the tuning of the group’s cantenna’s, by helping to trim the antenna’s themselves so that the rest of the group could measure the antenna’s S11 parameters.

Appendix 2 – Literature and Website References


[2] MSP430G2452 Data Sheet
[3] LTC2382-16 Data Sheet

http://www.ti.com/lit/ug/slau144h/slau144h.pdf


[6] MAX3420E Data Sheet


[8] Charvat, Gregory L. "Free Online Course Materials | Resource Home | MIT
OpenCourseWare." Free Online Course Materials | MIT OpenCourseWare.
Massachusetts Institute of Technology Lincoln Laboratory. 04 Dec. 2011
<http://ocw.mit.edu/resources/res-ll-003-build-a-small-radar-system-capable-of-
sensing-range-doppler-and-synthetic-aperture-radar-imaging-january-iap-
2011/index.htm>.


http://www.exocortex.org/dsp/

[12] DirectX.Capture Open Source C# Library utilizing DirectSound for Audio Capture over Line In
Appendix 3 – Detailed technical attachments

Coding for MCU:

```c
#include <msp430g2452.h>
#include <intrinsics.h>
#include <stdint.h>
uint16_t RXdata;
int count;
void main(void)
{
    WDTCTL = WDTPW + WDTHOLD; // Stop watchdog timer
    BCSTCL1 = CALBC1_16MHZ; //Set clock
    DCOCR = CALDC0_16MHZ; //Set clock
    P1OUT = BIT1;
    P1DIR |= BIT1;
    USICTL0 |= USIPE7|USIPES|USIMST|USIOE; //Set USICTL_0
    USICTL1 = USICKPH|USIE; //Set USICTL_1
    USICKCTL = USIDIV_0 + USISSEL_2; // Set USICKCTL
    USIINT = USII16B; // 16-bit data length
    USICTL0 &= ~USISWRST; // USI released for operation
    USIINT = 16; // init-load counter
    for (;;){
        _low_power_mode_0(); //Sleep and wait for wake up
    }
}

#pragma vector = USI_VECTOR //Wake up to get data
__interrupt void USI_ISR(void)
{
    P1OUT &= ~BIT1; // Disable ADC
    RXdata = USISR; // Copy data
    P1OUT |= BIT1; // Enable ADC
    for (count = 0; count < 1; ++count); // Delay for clock
    USIINT = 16; // re-load counter
}
```

%MIT IAP Radar Course 2011
%Resource: Build a Small Radar System Capable of Sensing Range, Doppler,
%and Synthetic Aperture Radar Imaging
%
%Gregory L. Charvat

%Process Range vs. Time Intensity (RTI) plot

clear all;
close all;
```
%read the raw data .wave file here
[Y,FS,NBITS] = wavread('running Outside_20ms.wav');

%constants
c = 3E8; %(m/s) speed of light

%radar parameters
Tp = 20E-3; %(s) pulse time
N = Tp*FS; %# of samples per pulse
fstart = 2260E6; %(Hz) LFM start frequency
fstop = 2590E6; %(Hz) LFM stop frequency
BW = fstop-fstart; %(Hz) transmit bandwidth
f = linspace(fstart, fstop, N/2); %instantaneous transmit frequency

%range resolution
rr = c/(2*BW);
max_range = rr*N/2;

%the input appears to be inverted
trig = -1*Y(:,1);
s = -1*Y(:,2);
clear Y;

%parse the data here by triggering off rising edge of sync pulse
count = 0;
thresh = 0;
start = (trig > thresh);
for ii = 100:(size(start,1)-N)
    if start(ii) == 1 & mean(start(ii-11:ii-1)) == 0
        %start2(ii) = 1;
        count = count + 1;
        sif(count,:) = s(ii:ii+N-1);
        time(count) = ii*1/FS;
    end
end
%check to see if triggering works
%plot(trig,'.b');
%hold on;si
%plot(start2,'.r');
%hold off;
%grid on;

%subtract the average
ave = mean(sif,1);
for ii = 1:size(sif,1);
    sif(ii,:) = sif(ii,:) - ave;
end

zpad = 8*N/2;

%RTI plot
figure(10);
v = dbv(ifft(sif,zpad,2));
S = v(:,1:size(v,2)/2);
m = max(max(v));
imagesc(linspace(0,max_range,zpad),time,S-m,[ -80, 0]);
colorbar;
ylabel('time (s)');
xlabel('range (m)');
title('RTI without clutter rejection');

%2 pulse cancelor RTI plot
figure(20);
sif2 = sif(2:size(sif,1),:] - sif(1:size(sif,1)-1,:);
v = ifft(sif2,zpad,2);
S = v;
R = linspace(0,max_range,zpad);
for ii = 1:size(S,1)
    %S(ii,:) = S(ii,:).*R.^(3/2); %Optional: magnitude scale to range
end
S = dbv(S(:,1:size(v,2)/2));
m = max(max(S));
imagesc(R,time,S-m,[ -80, 0]);
colorbar;
ylabel('time (s)');
xlabel('range (m)');
title('RTI with 2-pulse cancelor clutter rejection');
## Bill of Materials (BOM) 1/3

<table>
<thead>
<tr>
<th>Callout</th>
<th>Qty/Kit</th>
<th>Part#</th>
<th>Description</th>
<th>Supplier</th>
<th>Supplier Part #</th>
<th>Cost Each</th>
<th>Subtotal</th>
</tr>
</thead>
<tbody>
<tr>
<td>OSC1</td>
<td>1</td>
<td>ZX95-2536C+</td>
<td>2315-2536 MC VCO, +6 dBm Out</td>
<td>Mini-Circuits</td>
<td>ZX95-2536C+</td>
<td>$42.50</td>
<td>$42.50</td>
</tr>
<tr>
<td>ATT1</td>
<td>1</td>
<td>VAT-3+</td>
<td>3dB SMA M-F attenuator</td>
<td>Mini-Circuits</td>
<td>VAT-3+</td>
<td>$9.95</td>
<td>$9.95</td>
</tr>
<tr>
<td>PA1/LNA1</td>
<td>2</td>
<td>ZX60-272LN-S+</td>
<td>Gain 14 dB, NF=1.2 dB, IP1= 18.5 dBm</td>
<td>Mini-Circuits</td>
<td>ZX60-272LN-S+</td>
<td>$39.95</td>
<td>$79.90</td>
</tr>
<tr>
<td>SPLTR1</td>
<td>1</td>
<td>ZX10-2-42+</td>
<td>1900-4200 Mc, 0.1 dBinsertion loss</td>
<td>Mini-Circuits</td>
<td>ZX10-2-42+</td>
<td>$34.95</td>
<td>$34.95</td>
</tr>
<tr>
<td>MXR1</td>
<td>1</td>
<td>ZX05-43MH-S+</td>
<td>13 dBm LO, RF to LO loss 6.1 dB, IP1</td>
<td>Mini-Circuits</td>
<td>ZX05-43MH-S+</td>
<td>$46.45</td>
<td>$46.45</td>
</tr>
<tr>
<td>SMA M-M Barrels</td>
<td>4</td>
<td>SM-SM50+</td>
<td>SMA-SMA M-M barrel</td>
<td>Mini-Circuits</td>
<td>SM-SM50+</td>
<td>$5.45</td>
<td>$21.80</td>
</tr>
</tbody>
</table>

### Cannenas

<table>
<thead>
<tr>
<th>Callout</th>
<th>Qty/Kit</th>
<th>Part#</th>
<th>Description</th>
<th>Supplier</th>
<th>Cost Each</th>
<th>Subtotal</th>
</tr>
</thead>
<tbody>
<tr>
<td>Can</td>
<td>2</td>
<td>TBD</td>
<td>TBD</td>
<td>local grocery store</td>
<td>$5.00</td>
<td>$10.00</td>
</tr>
<tr>
<td>L bracket</td>
<td>2</td>
<td>NA</td>
<td>L-bracket, 7/8&quot;, zinc plated</td>
<td>McMaster Carr</td>
<td>$0.35</td>
<td>$0.70</td>
</tr>
<tr>
<td>SMA F bulkhead</td>
<td>2</td>
<td>901-9889-RFX</td>
<td>SMA bulkhead F solder cup</td>
<td>Mouser</td>
<td>523-901-9889-RFX</td>
<td>$4.27</td>
</tr>
<tr>
<td>6-32 screws</td>
<td>1</td>
<td>NA</td>
<td>6-32 machine screw, 5/8&quot; length, pk of 100</td>
<td>McMaster Carr</td>
<td>$3.49</td>
<td>$3.49</td>
</tr>
<tr>
<td>6-32 nuts</td>
<td>1</td>
<td>NA</td>
<td>6-32 hex nuts, pk of 100</td>
<td>McMaster Carr</td>
<td>$1.09</td>
<td>$1.09</td>
</tr>
<tr>
<td>6-32 lockwashers</td>
<td>1</td>
<td>NA</td>
<td>lock washers for 6-32 screws, pk of 100</td>
<td>McMaster Carr</td>
<td>$0.71</td>
<td>$0.71</td>
</tr>
<tr>
<td>6&quot; SMA M-M Cables</td>
<td>3</td>
<td>086-12SM+</td>
<td>SMA-SMA M-M 6&quot; cable</td>
<td>Mini-Circuits</td>
<td>086-12SM+</td>
<td>$9.65</td>
</tr>
<tr>
<td>Component</td>
<td>Quantity</td>
<td>Description</td>
<td>Supplier</td>
<td>Part Number</td>
<td>Price 1</td>
<td>Price 2</td>
</tr>
<tr>
<td>-------------------------------</td>
<td>----------</td>
<td>------------------------------------------</td>
<td>-------------------</td>
<td>-------------</td>
<td>---------</td>
<td>---------</td>
</tr>
<tr>
<td>Wood Screws</td>
<td>1 NA</td>
<td>brass #2 wood screws 3/8&quot; long, pk 100</td>
<td>McMaster Carr</td>
<td>98685A225</td>
<td>$3.70</td>
<td>$3.70</td>
</tr>
<tr>
<td>measuring tape</td>
<td>1 NA</td>
<td>50' long measuring tape</td>
<td>McMaster Carr</td>
<td>6839A68</td>
<td>$11.22</td>
<td>$11.22</td>
</tr>
<tr>
<td>Wood</td>
<td>1 NA</td>
<td>12&quot; wide by 1&quot; thick 8' long wood</td>
<td>Home Depot</td>
<td>458538</td>
<td>$14.37</td>
<td>$14.37</td>
</tr>
<tr>
<td>Modulator</td>
<td>1 XR-2206</td>
<td>Function Generator Chip</td>
<td>Mouser</td>
<td>701-XR2206CP-F</td>
<td>$4.05</td>
<td>$4.05</td>
</tr>
<tr>
<td>Video Amp</td>
<td>1 MAX414CP+</td>
<td>low-noise quad op-amp</td>
<td>Digi-Key</td>
<td>MAX414CP+ND</td>
<td>$14.46</td>
<td>$14.46</td>
</tr>
<tr>
<td>solderless breadboard</td>
<td>1 EXP-300E</td>
<td>6.5x1.75&quot; solderless breadboard</td>
<td>Mouser</td>
<td>510-EXP-300E</td>
<td>$7.00</td>
<td>$7.00</td>
</tr>
<tr>
<td>C1</td>
<td>4 4SA105A102JAR</td>
<td>1000 pf 5% capacitor</td>
<td>Digi-Key</td>
<td>478-3147-1-ND</td>
<td>$0.22</td>
<td>$0.86</td>
</tr>
<tr>
<td>R1a</td>
<td>1 MFR-25FBF-8Kd5</td>
<td>6450 ohm 1% resistor</td>
<td>Digi-Key</td>
<td>8.45KXBK-ND</td>
<td>$0.11</td>
<td>$0.11</td>
</tr>
<tr>
<td>R1b</td>
<td>1 MFR-25FBF-102K</td>
<td>102K ohm 1% resistor</td>
<td>Digi-Key</td>
<td>102KXBK-ND</td>
<td>$0.11</td>
<td>$0.11</td>
</tr>
<tr>
<td>R2</td>
<td>1 MFR-25FBF-7K15</td>
<td>7150 ohm 1% resistor</td>
<td>Digi-Key</td>
<td>7.15KXBK-ND</td>
<td>$0.11</td>
<td>$0.11</td>
</tr>
<tr>
<td>RF_1_2</td>
<td>3 MFR-25FBF-1K00</td>
<td>1K ohm 1% resistor</td>
<td>Digi-Key</td>
<td>1.00KXBK-ND</td>
<td>$0.11</td>
<td>$0.34</td>
</tr>
<tr>
<td>Rg</td>
<td>1 MFR-25FBF-12K1</td>
<td>12.1K ohm 1% resistor</td>
<td>Digi-Key</td>
<td>12.1KXBK-ND</td>
<td>$0.11</td>
<td>$0.11</td>
</tr>
<tr>
<td>R1a_2</td>
<td>1 MFR-25FBF-17K4</td>
<td>17.4K ohm 1% resistor</td>
<td>Digi-Key</td>
<td>17.4KXBK-ND</td>
<td>$0.11</td>
<td>$0.11</td>
</tr>
<tr>
<td>R1b_2</td>
<td>1 MFR-25FBF-28K0</td>
<td>28K ohm 1% resistor</td>
<td>Digi-Key</td>
<td>28.0KXBK-ND</td>
<td>$0.11</td>
<td>$0.11</td>
</tr>
<tr>
<td>R2_2</td>
<td>1 MFR-25FBF-4K12</td>
<td>4120 ohm 1% resistor</td>
<td>Digi-Key</td>
<td>4.12KXBK-ND</td>
<td>$0.11</td>
<td>$0.11</td>
</tr>
<tr>
<td>Rg_2</td>
<td>1 MFR-25FBF-1K62</td>
<td>1620 ohm 1% resistor</td>
<td>Digi-Key</td>
<td>1.62KXBK-ND</td>
<td>$0.11</td>
<td>$0.11</td>
</tr>
<tr>
<td>Decoupling cap</td>
<td>2 2K104Z15YSVE5TH5</td>
<td>0.1 uf</td>
<td>Mouser</td>
<td>594-K104Z15YSVE5TH5</td>
<td>$0.05</td>
<td>$0.10</td>
</tr>
</tbody>
</table>
private void fftTest_Click(object sender, EventArgs e) {
    tabControl1.SelectTab(0);
    chartNorm.Series["Sin"].(Points).Clear();
    chartFFT.Series.Add("Imag");
    chartFFT.Series["Imag"].(ChartType = SeriesChartType.FastLine);
    chartNorm.Series.Add("Cos");

    this.Refresh();
    int n = 512;
    int fftn = 9;

    int xLow = 0;
    double xHi = 4 * Math.PI;

    double numPerPoint = (double)(xHi - xLow) / n;
    double currPoint;
    Complex[] SinTest = new Complex[n];
    for (int i = 0; i < n; i++)
    {
        currPoint = numPerPoint * i;
        SinTest[i].Real = (float)Math.Sin((double)currPoint);
        chartNorm.Series["Sin"].(Points).AddY(Math.Sin(currPoint));
//chartNorm.Series["Cos"].Points.AddY(Math.Cos(currPoint));
}

chartNorm.Series["Sin"].ChartType = SeriesChartType.Line;
this.Refresh();

FFT DoFFT = new FFT(fftn);
DoFFT.Run(ref SinTest);

int upperBnd = (int)Math.Pow(2,fftn);
for (int i = 0; i < (upperBnd); i++)
{
    //Math.Sqrt( Math.Pow(SinTest[i].Real, 2) +
    //Math.Pow(SinTest[i].Imaginary,2))
    chartFFT.Series["Imag"].Points.AddY(SinTest[i].Real);
    chartFFT.Series["Imag"].Points.AddY(SinTest[i].Imaginary);
}
chartFFT.Series["Imag"].ChartType = SeriesChartType.Line;
chartFFT.Series["Imag"].ChartType = SeriesChartType.Line;

chartFFT.Titles.Add("FFT");
chartNorm.Titles.Add("SIN Function");
this.Refresh();