April 2007

Ultrasonic 3D Wireless Computer Mouse

Christian John Banker
Worcester Polytechnic Institute

Jamie E. Mitchell
Worcester Polytechnic Institute

Jeffrey D. Tucker
Worcester Polytechnic Institute

Jeffrey Vincent DiMaria
Worcester Polytechnic Institute

Michael A. Cretella
Worcester Polytechnic Institute

Ultrasonic 3D Wireless Computer Mouse

An improved three-dimensional computer-human interface device

A Major Qualifying Project submitted to the faculty of the
Worcester Polytechnic Institute
Electrical and Computer Engineering Department
in partial fulfillment of the requirements for the
Degree of Bachelor of Science

April 26, 2007

Christian Banker
cbanker@wpi.edu

Michael Cretella
cretella@wpi.edu

Jeff Dimaria
jvd07@wpi.edu

Jamie Mitchell
jamiem@wpi.edu

Jeff Tucker
tux@wpi.edu

magicmouse@wpi.edu

Project Advisor: Professor Brian King
Project: BYK-MOUS

Abstract

The aim of this project is to develop a three-dimensional computer input device which provides a better interface between a user and their computer. The user wears a ring on his or her finger which transmits an ultrasonic signal to a receiver array. A microcontroller then calculates the three-dimensional coordinates using time-difference-of-arrival methods. These coordinates are input to the computer as a standard human interface device (HID) USB peripheral. X and Y dimensions control the mouse cursor on the screen, and the Z dimension can be used in three-dimensional applications.

Executive Summary

The purpose of this project is to develop a "Magic Mouse", which is a three-dimensional wireless mouse that is both small in profile and intuitive in use. The standard computer mouse has been in use for over 40 years, and recent developments in 3D software are testing the limits of the mouse's design. To compensate for the two-dimensional nature of the mouse, a scroll wheel has been added to the mouse, which can traverse the third dimension when needed. However, this design does not allow for simultaneous movement in three dimensions, nor does it have a high resolution along the depth axis.

There are a few products previously developed that provide similar functionality to the Magic Mouse. A wireless gyroscopic mouse by Gyration allows the user to hold the mouse in their hand and eliminates the need for a table or other form of a mouse pad. However, this product is cumbersome and lacks three-dimensional movement. Logitech also developed a 3D mouse based on ultrasonic technology, but it is expensive, bulky and does not have wireless functionality. The most similar device to the Magic Mouse that is currently on the market is the Wiimote by Nintendo, which intuitively combines three dimensions and wireless functionality. The difference between this product and the Magic Mouse lies in the fact that the Magic Mouse is small, unobtrusive and consumes a minimal amount of power.

Preliminary research explored the areas of ultrasonic transmission, radio frequency identification, magnetic triangulation, the use of accelerometers, and gyroscopes to determine 3D coordinates. One of the main design requirements is to have a low-profile solution, so immediately gyroscopes were eliminated. Accelerometer-based mice have been implemented in projects at other universities as well as commercially proven. This design seeks a novel approach, so accelerometers are not the ideal choice. The use of magnetics involves a few complications. For a small, completely passive device, the use of large electromagnets is needed, and these would be next to a personal computer which could pose problems to its normal operation. After significant research, it was found that the design challenges of implementing an accurate and robust three-dimensional position tracking system using magnetic field sensing outweighs its advantages.

The use of radio frequency identification is a promising implementation for future work, but at this time the technology to track a signal within one meter and with millimeter accuracy is not available. To compensate for this, ultrasonic transmission, which propagates about 1 million times slower than electromagnetic waves, is used as the tracking mechanism.

The final design of the Magic Mouse consists of an ultrasonic signal transmitted from a ring on the user's finger. The signal arrives at an array of receivers, at a different time on each receiver based upon its distance from the transmitter. These receivers convert the acoustic energy into electrical pulses. Based upon these pulse times, a time difference of arrival (TDOA) algorithm determines the transmitter's 3D position in space. This is realized in four major blocks, with the first being signal transmission. To transmit the signal, a 40 kHz pulse is sent via a microcontroller and a small ultrasonic transducer. This unit is powered by a rechargeable battery in the shape of ring, which allows the device to achieve a low profile. The entire circuit board for this block is less than 1 square inch, and weighs less than 10 grams.

The second stage begins by receiving the signal on five different transducers set in an optimized array configuration. Analog circuitry then amplifies and shapes the pulse on the five channels. These conditioned signals provide an acceptable input to a digital signal processor (a Microchip dsPIC), which is the third system block. The signals are then sampled by an Analog Devices AD9220 10 MSps 12-bit analog-to-digital converter and read by the dsPIC.

The signals are then compared to a stored characteristic signal shape through a curve fitting algorithm to determine the time associated with the peak of each channel. Since each pulse has a similar curve shape, the peak of each pulse is a reliable estimate of the time that the pulse arrived on each receiver, relative to the other pulses. Finding an accurate peak is essential for TDOA, and is also computationally intensive. The applied curve fitting algorithm allows peak detection with a root-mean-squared (RMS) error of less than 5 μs, which translates to a positional accuracy of less than 1 cm. The relative peak times for each receiver are then input to the TDOA algorithm. By knowing the precise location of each receiver, the TDOA algorithm converts these unique time delays into a accurate estimate of the 3D coordinates of the transmitter. The dsPIC

has now generated X, Y, and Z coordinates, and the fourth and final stage sends data to the computer using the Human Interface Device (HID) Plug-n-Play mouse standard, with the Z-axis mapped to the scroll wheel. This is accomplished by sending the raw coordinates to a second microprocessor with its only purpose being to put the coordinates in the correct format and to send them to the host computer.

Although in the time frame of this project full product implementation was not achieved, the essential developments for a proof of concept have been implemented. The transmitted signal is received by the five receiving transducers, and their peaks are accurately detected. The first block to perform less than ideally is the TDOA algorithm. This is not due to flaws in the mathematics, but rather a failure to fully calibrate the receiver positions. Knowing the exact receiver positions is critical for this algorithm to function properly. A 1 mm discrepancy in the position of one receiving transducer can translate to an error of over 20 mm in the position of the transmitter after the executing the TDOA algorithm. However, the coordinates have been determined to an accuracy that meets the basic needs for a proof of concept. The second shortcoming is that a USB interface has not been fully developed. Synchronization between the microcontrollers using UART protocol has proved to be more problematic than expected, but with more time, this issue could be resolved.

This device has many applications for a wide variety of users, including students, professional engineers, gamers, and general consumers. At this point the limitations of the device can be easily solved after a moderate amount of time and debugging. Future work such as hardware gesture recognition can also be readily implemented. Indeed, the versatility and small form factor of this design are the first steps to a unique and innovative device.

Acknowledgments

We would like to thank WPI and specifically the ECE department and shop for the use of their facilities.

We would also like to thank the WPI marketing team and Popular Science for giving us the opportunity to be published in the June issue of the Popular Science magazine.

Most importantly, we would like to thank our advisor, Professor Brian King, for his invaluable help and support throughout the year. Without his motivation, expert guidance, and seemingly unlimited insight, the completion of this project may not have been possible.

Contents

List of Figures

List of Tables

1 Introduction

Computer interfaces comprise one of the three major divisions within computer theory: input, processing, and output. Every device or procedure involved in computing can be categorized as performing one of these functions. Input devices provide an interface to allow a human to interact with the computer. Interface efficiency is an important factor of the design of an input device. The Dvorak keyboard standard, for example, arranges letters in a way that is not uniform to the alphabetical order of the characters, or to the standard QWERTY layout. As a result, learning to use the device competently requires time and training. After some experience with the device, however, one realizes that the keys are arranged in a very efficient way; the most used characters lie in easily accessible locations for the hands. This exemplifies the trade-off between instinctive interaction and device functionality. The ideal interface can balance this relationship, resulting in seamless interaction between the computer and the user. The driving goal of this project is to develop an input device for the PC that is intuitive to use, and provides increased functionality over a traditional 2D computer mouse.

The idea of the computer mouse was developed in the 1960s at the Stanford Research Institute as a token which could be moved about a surface to translate movement data to the computer [16]. The sensory device was implemented using gears that rested on the table on which the mouse moved. Although mouse technology has seen significant progress in communication, accuracy, and functionality in the past forty years, there have been minimal changes to its method of implementation. In the late 1990s, the addition of the scroll wheel was a significant improvement to the mouse at the time, as it provided two-dimensional shortcuts as well as the option to implement a virtual third axis of motion. Most applications today utilize the wheel on the mouse to scroll information on a page, traversing the Y-axis. In other applications, the wheel represents zooming in and out in a third dimension.

Our goal is to develop a device that can be used with personal computers that is an intuitive and functional human-computer interface. It must be versatile enough to track three dimensions of motion, yet not inhibit the motion of the user. The user places a ring that contains a small

battery and circuitry onto their finger to transmit ultrasonic pulses to the receiver microphones set near the monitor. The receivers then compute the position of the ring, and transmit that information to the computer, which moves the pointer on the screen. While similar 3D mice exist, none excel in simplicity or discrete design to date. This report documents the process of designing and constructing the device, including background research, methodology, implementation, and testing. Recommendations are also made regarding further development of the device and possible revisions to its design.

2 Background

The first step to meeting the design challenges of this project was to perform extensive background research. The purpose of this research was to create a deeper understanding of the scope of the project so that creative solutions to the design challenge could be achieved that were within the scope of implementation. Previous designs for a 3D position tracking system were investigated, and the technologies behind each product were scrutinized in detail. The most important characteristics of these technologies as they apply to a 3D mouse are highlighted and discussed in the following sections.

2.1 Current Technology

Prior art research is the first step into understanding the design challenge. Technologies that have already been developed for 3D tracking have succinctly defined what is already possible and what may be improved upon by this project. Technical reports, design patents, and commercial products were reviewed and analyzed. This information was a basis for further research into transmission, sampling, and interface technologies.

Review of 2003 MQP

In 2003, Advanced Media Technology (AMT) in Ireland sponsored a WPI Major Qualifying Project (MQP) entitled "Magneto-resistive Sensors Applied to the Development of a Three Dimensional Wireless Mouse" which had similar goals to this project. The group was to design a passive three-dimensional wireless mouse, with the added conditions of multiple object tracking, and the exclusion of optical tracking as a technology. The group researched inductive, magneto-resistive, and capaciflector field sensing, as well as radio frequency and ultrasonic time delay tracking.

The first two technologies ruled out were capaciflector field sensing and ultrasonic time delay tracking. Capaciflector sensing would rely on a tracking object moved inside the electric field generated by a large air capacitor. The presence of a foreign object in the field would change

properties of the dielectric, which would be monitored to track position. It was found, however, that any material inside the field, including the user's hand, would have a significant change on the electric field, so tracking a single object within the field would not be possible. Ultrasonic time delay tracking proved to be technically viable, but unsuitable for the project, because the pointing device could not meet AMT's specification of being passive. The intricacies of ultrasonic tracking are described in further detail in Section 2.7.1.

The next technology to be eliminated was radio frequency time delay tracking. There were no experiments performed, but research quickly led the group to the conclusion that the reflection time would be too small to track accurately.

The group then turned to electromagnetics, and performed experiments involving mutual inductance and magneto-resistive sensing. The first experiment was to detect a change in the voltage across a conductor coil due to the induced electric field of a nearby LC circuit. Their first attempt showed no results, so they constructed a Wheatstone bridge to isolate the electric field produced by the LC circuit. With the addition of the Wheatstone bridge, when a metal object was placed inside the inductor coil a small, but usable voltage appeared across the bridge's outputs. However, the group decided to eliminate this option as well, after experimenting with magneto-resistive sensors.

Using a magnet constructed from a screw and magnet wire, the group tested the viability of the magneto-resistive integrated circuits (ICs) they purchased from Honeywell. A magnetic field was detected seven centimeters from the ICs, which provided the assurance needed to proceed with magneto-resistive sensing.

For the final tracking system, three sets of three-dimensional sensors were used to track the position of a $7 \times 1 \times 1.5$ cm bar magnet. Due to time limitations concerning the software development, the group was only able to implement two-dimensional tracking. They did this by constructing a plastic box around their circuit boards, with a 10×10 cm grid drawn on its top cover. The magnet's south pole was placed face down on the grid, and its movements were tracked. Repeatability was demonstrated by creating a unique identifier for each cell in the grid,

finding the output values of the magneto-resistive sensors at that cell. These values were put into a SQL database, and when the magnet was again placed on a cell, the software would search for the corresponding voltages in the database to find the cell the magnet was over. They were able to achieve 95 percent accuracy with this method.

Limitations of their project include the need for calibration of the magneto-resistive circuits, and the fact that multiple object tracking would be difficult if not impossible to implement.

2.1.1 Logitech 3D Mouse

A product that was developed by Logitech of Fremont, CA in 1992 called the "3D Mouse and Head Tracker" is a three-dimensional tracking mouse based on ultrasonic signals. The user takes this device and places one piece of the device on their head, they can then use the mouse and head movement to control a virtual reality application. The transmitter unit sits on the desk and points toward the user. Each of the three speakers on the devices sends an ultrasonic pulse in a cone area. The mouse has three microphones which detect the pulse and triangulate the position of the mouse in three-dimensional space. The kit comes with a control unit which connects to the computer via RS-232, and the mouse and head gear attach to this control unit. Although the mouse itself is not wireless, the position of the mouse is calculated using wireless transmission.

Although the application of this product is specifically for virtual reality applications and not as a general-purpose mouse, there is still much that can be learned from this device. The user manual for the device is more extensive than a typical guide; it provides calculations, schematics, and an application programming interface (API) for the device [5].

2.1.2 Gyration Air Mouse

Gyration's air mouse is a two-dimensional input device that is controlled by wrist motions in the air. It senses angular movement using a two-axis gyroscope and does not require any mousing surface. It also has the ability to switch into a standard optical mouse mode for desktop use. Current versions use 2.4 GHz wireless communication to provide approximately a 30 foot range without requiring line of sight [14]. The current retail price of these devices is $70 for the home

version and $180 for the professional version, which adds an additional battery, in-line battery charger and more advanced professional media and presentation control software [14].

These devices use a miniature two-axis gyroscope IC called the MicroGyro 100, or MG100 for short [14]. This IC senses rotational force in two axes: up-down and left-right. The chip contains a miniature tuning fork that is vibrated by an electromagnet. When the device is moved, the movements of the tuning fork are sensed to determine how the device is being moved [14].

A focus group was held with three air mouse users to determine the effectiveness of these devices. All three spoke very positively of the overall performance of their mice. They stated that accuracy is very good once you become accustomed to using the device. They also noted that with some practice, it can be effectively used for drawing in image editing applications as well as in some first-person shooter computer games. The trigger button that turns on and off the motion tracking is an essential feature due to the tendency for the cursor to move when clicking buttons on the mouse. They also found the additional programmable buttons to be desirable and liked the ability to program mouse gestures. Their only complaints involve a common problem where the mouse has trouble switching between air mouse and desktop modes and will not always switch immediately.

The users interviewed noted that there is a noticeable learning curve, but that it is not too difficult to overcome. When asked whether this device causes any problems with arm or wrist fatigue they explained that it was not an issue because of the minimal movement required and the fact that only the wrist is moved, allowing users to rest their arm on the armrest of their chair.

2.2 Patents

One resource for prior art research into existing products or technologies pertaining to a three-dimensional mouse is the United States Patent and Trademark Office (USPTO) patent database. The patent database is available to search from the USPTO website at (`http://patft.uspto.gov/netahtml/PTO/search-adv.htm`) and contains electronic versions of all patents filed in the

United States from 1790 to the present day (2007), and access to most patents filed in the 20th century in either Europe or Japan. The database is an excellent source of previously implemented ideas, and their complexity ranges from full-scale technical documents that outline every detail in constructing a working prototype to outlines of general concepts, the understanding of which is aided by figures and diagrams. The searches conducted on this database sought to highlight the content most relevant to implementing a three-dimensional virtual pointing device.

The USPTO advanced search page filters the patent database according to Boolean logic and uses search strings input by the user. The first search string entered into the USPTO search page was:

> interface AND mouse AND (three AND dimensional OR 3 AND dimensional) AND position AND (wireless OR radio) AND device ANDNOT (rights OR face AND recognition OR hair OR protein OR robot OR method OR scanner OR barcode OR keyboard)

The keywords were chosen in the hope that they would refine the database down to patents that pertain to three-dimensional mouse interfaces. Additional words were added to refine the search further, such as wireless or position. The ANDNOT term of the search string filtered out patents that pertained to face recognition, barcodes, scanners, protein, robots, methods, hair, and rights. These search terms appeared frequently in the titles of many patents in a preliminary search without the ANDNOT term, so they were included in patents to remove from the search. This limited the patents for further scrutiny down to 23. Twenty-three patents is a very small percentage of over 7 million patents in the database and a new search was conducted for more results. The new search was conducted with the following search string:

> interface AND mouse AND (three AND dimensional OR 3 AND dimensional) AND position AND (wireless OR radio) AND device ANDNOT (rights OR face AND recognition OR hair OR protein OR robot OR scanner OR information OR barcode OR keyboard)

This search returned 14 additional patents.

Table 1 lists the patents that were reviewed as part of this background research.

Eight of the twenty-five total patents listed in Table 1 are selected and summarized to highlight key features. These patents highlight the devices in the patent search that most closely match the goals of this MQP. Their summaries are roughly ordered according to the method by which the 3D position of the device was determined. The first three patents correspond to optical recognition or sensor systems. The next three patents utilize magnetic positioning systems. The last two patents utilize hybrid systems. One device uses ultrasound triangulation and an RF transmitter to transmit position data. The final device uses accelerometers to track the position of the device and pressure sensors to track the orientation of a wand.

United States Patent 7,098,891: "Method for providing human input to a computer"

This patent uses a visual tracking system. An electro-optic sensor detects the position of two targets on a user's body and determines the distance between them. From this information, a computer calculates the two-dimensional motion of the user in the area defined by the two targets. This invention exhibits the ability to determine position and distance optically, with sensors that operate at very low power.

United States Patent 5,815,411: "Electro-optic vision system which exploits position and attitude"

This device is an electronic vision system. The entire system includes a camera to gather optical information about a particular environment, a computer processor, a device to measure camera position and attitude, and a database of stored images. The invention uses this vision system to delete, add, and supplement images to a scene that it is currently observing and displays these changes in real-time. The algorithms behind image addition, deletion, and supplementation show that complex optical systems can also determine three-dimensional position based upon relative object size and location with respect to camera orientation and a database of recorded images.

United States Patent 6,222,465: "Gesture-based computer interface"

Patent Number	Patent Title
6,373,492	Computer-assisted animation construction system and method and user interface
5,854,634	Computer-assisted animation construction system using source poses within a pose transformation space
6,678,546	Medical instrument guidance using stereo radiolocation
6,222,465	Gesture-based computer interface
6,469,633	Remote control of electronic devices
7,034,804	Computer pointing device employing a magnetic field source and magnetic field sensors
5,453,785	Measurement camera with fixed geometry and rigid length support
5,444,917	Sensing device
5,296,871	Three-dimensional mouse with tactile feedback
5,541,621	Mouse or trackball system
5,875,257	Apparatus for controlling continuous behavior through hand and arm gestures
6,159,101	Interactive toy products
7,098,891	Method for providing human input to a computer
7,096,148	Magnetic tracking system
6,710,770	Quasi-three-dimensional method and apparatus to detect and localize interaction of user-object and virtual transfer device
6,774,624	Magnetic tracking system
6,845,241	Relevance assessment for location information received from multiple sources
6,453,246	System, method, and computer program product for representing proximity data in a multi-dimensional space
6,445,943	Position tracking and imaging system for use in medical applications
6,019,725	Three-dimensional tracking and imaging system
6,211,863	Method and software for enabling use of transcription system as a mouse
6,008,798	Method of determining an object's position and associated apparatus
4,654,648	Wireless cursor control system
4,814,552	Ultrasound position input device
5,815,411	Electro-optic vision system which exploits position and attitude

Table 1: Patent Search Results

This patent outlines a system that uses a video capture device to recognize hand gestures and manipulate elements on a computer screen. The video capture device observes a limited zone in which it can see the user's hand and software determines the location and gesture that the hand represents. There are different commands that the recognition system interprets, including gripper (thumb and forefinger pressed together), resize, draw, delete object, and menu commands. These commands can either be traditionally selected with a computer mouse or gestured within the video capture zone.

United States Patent 7,096,148: "Magnetic Tracking System"

This invention uses a magnetic field generator and sensor to detect changes in a magnetic field caused by a metal object within that field. The object is placed between the field generator and an array of magnetic field sensors, which triangulate position. The system is calibrated for each situation of use under ideal conditions. A shortfall of this technology is that metal objects within twice the distance between the magnetic field generator and sensor will affect the field. Another shortfall is that this system must be calibrated under ideal conditions before field use.

United States Patent 7,034,804: "Computer Pointing Device Employing a Magnetic Field Source and Magnetic Field Sensors"

This device includes a magnetic field source and a magnetic field sensor. The change in position of the magnetic field source is measured by the magnetic field sensor as the source is moved in space. The magnetic field source is a permanent magnet or an electromagnetic solenoid. A Hall Effect element is present in the magnetic field sensor of the device that has the ability to measure the magnitude of the magnetic field along three axes. The transmission system for this patent is a connector cable attached to the magnetic field sensor. The purpose of this invention is to provide an alternative to the traditional two-dimensional computer mouse.

United States Patent 5,854,634: "Computer-assisted Animation Construction System Using Source Poses Within a Pose Transformation Space"

The object of this patent is to create a system to generate natural computer animation. It utilizes a set of predefined source poses of an animation sequence. A wand is included and uses

magnetic positioning, with a magnetic field generator and magnetic field sensors to detect the three-dimensional position of the wand. With the wand, the user can move through the source poses of the animation sequence, and manipulate their display orientation on a computer.

United States Patent 4,814,552: "Ultrasound Position Input Device"

This patent outlines a stylus or pen design that utilizes ultrasound to determine its two-dimensional position. The device uses a pressure sensor to determine whether the pen is in contact with a surface, and contains an ultrasonic receiver and wireless RF transmitter. The ultrasonic receiver is used in conjunction with base unit transmitters to triangulate the position of the stylus. The wireless RF transmitter sends this position data to a wireless module, which is then interpreted by a computer into a two-dimensional writing space. One disadvantage of this device is that it uses an ultrasonic transmitter/receiver pair and an additional wireless RF transmitter/receiver pair.

United States Patent 5,875,257: "Apparatus for Controlling Continuous Behavior Through Hand and Arm Gestures"

This invention utilizes accelerometers embedded in a wand to determine the beat and position of the rod while it is being used to conduct music. The bulbous, cupped portion of the wand contains five pressure sensors which are aligned to each of the user's fingers. The device transmits accelerometer and pressure sensor data wirelessly to a receiver unit that interfaces with a computer. From this raw data, the absolute three-dimensional position of the baton can be determined, as well as its speed and the "beat" of its motion. The aim of this invention is to improve the user's conducting abilities with the aid of computer analysis and feedback.

These patents are examples as to how others have implemented three-dimensional position tracking systems. The summarized designs serve as a basis for potential methods of implementing the tracking system for this project. The selected patents are by no means a comprehensive list of the available options, but they show what has been done in the past and what may be accomplished in the future.

2.3 Gyroscopic Tracking

Gyroscopic tracking involves using gyroscopes to sense rotational changes. This is very effective for rotational movements, but does not map so well to lateral movement. One example of a device that uses this technology is the Gyration Air Mouse that was described earlier.

This design works well for two-dimensional applications, however it would not work very well for a three-dimensional design. Adding a third dimension would require the rotational axis of the wrist to be used. This would be difficult in conjunction with the left-right rotation. Because of the similarity between these axes, the three rotational axes could not be mapped intuitively to the Cartesian coordinate system. Another disadvantage is that the tracking is done by the hand-held device, so it would not be possible to implement this technology passively or in a very small device.

2.4 Radio Frequency Identification and Geometry (RFIG)

Radio Frequency Identification and Geometry, or RFIG, is a new technology that is an extension of RFID. It uses photosensitive RFID tags to make millimeter-accurate positioning possible. This technology is still under development and is being designed by Mitsubishi Electric Research Labs. RFIG allows objects to be precisely located in two-dimensional space and can also be extended to provide three-dimensional positioning.

The basic operation of RFIG involves tags mounted to objects being tracked along with a sensor and projector unit. The photo-sensing RFID tags are queried through RF as normal RFID tags would be. The projector then beams a unique time-varying code that is decoded by the tags [11]. Based on the code received, the tags then respond through RF with their precise pixel location [11]. The projector can then display information on the items being tracked for visual feedback.

Some of the currently envisioned applications include package tracking for stockrooms and book positioning for libraries. RFIG can be used to locate packages and display relevant information, such as expiration status. In libraries, books can be checked for proper position and

orientation using a handheld projector. If the books are out of place, the projector can even display arrows to show where they should be moved [11].

RFIG has the benefit that the tags do not necessarily require a battery. Since the tags only receive light and do not emit any, they have very low power requirements and can be implemented as passive tags. The current prototypes use a battery because active tags are easier to program, however future revisions can be passive. With passive tags, the size will be able to be reduced to the point where a tag can be as small as a grain of rice [11].

Three-dimensional positioning can be accomplished with the addition of a second projector. The two projectors are spaced apart and both aimed at the object from different angles [11]. By combining the information from the two separate projectors, positioning can be tracked in 3D space.

RFIG has some benefits and drawbacks for use as a 3D computer mouse. It has the advantages of the tags being small and not requiring power. It is also already fairly accurate and can be used for 3D applications. Unfortunately, there are some major drawbacks that would make this technology somewhat ineffective for the purposes of this project. The size of the two projectors required would be prohibitive, as would be their cost. Additionally, this is very new technology that is still in its early development stages and has not yet been released. Although most of these problems will be eliminated in the future, they are currently too detrimental to the goals of this project for RFIG to be a competitive option.

2.5 Accelerometers

While in search of previous products or research in three-dimensional tracking, it was found that a popular technology involves tracking with accelerometers. These devices are small, autonomous, low power chips making them well suited for this application. Accelerometer chips which detect three-dimensional acceleration send their signals to a processor which then must use a pair of integrators to determine the three-dimensional position of the mouse. This position can then be sent to the computer as a Human Interface Device (HID). Several projects at other

engineering universities have developed accelerometer based mice influenced by the film industry which exhibit the control of a computer interface using movements and gestures in the hands and fingers.

Students at the University of California at Berkeley developed a two-dimensional wireless glove that used accelerometers on each finger. The data from the accelerometers is sent to the base station (driven by an Atmel MPU) which is connected to the PS/2 interface on the PC; the devices are powered by two AA batteries. The application for the glove was mainly used for detecting individual finger movement and gestures, although the team had developed a basic Human Interface Device driver for the glove [8]. Graduate students at Cornell University developed a three-dimensional accelerometer based mouse which was a wired device and supported limited two-finger gestures. The device is controlled using an Atmel Megacontroller MPU, and connects to the PC over the PS/2 interface. This device is perhaps small enough to be discrete, however it lacks wireless communication abilities [3]. Of the existing accelerometer based tracking devices found, MIT students have developed the most comprehensive device. Operating on an FPGA and connected to the PC over the PS/2 interface, this device incorporates all three axes of motion and uses a single accelerometer board which communicates over RF with the base station [4]. These well-documented projects provided much insight in the consideration of using accelerometers for this project.

Accelerometers do not depend on any other devices for their operation and are autonomous by detecting individual changes in G-forces in each direction. The chips are small enough to incorporate many sensors on one hand, increasing the precision of the device. For the application of a simple HID mouse this does not benefit the device, however for applications that involve tracking of hand gestures, an accelerometer based approach would work well.

In order to interpret acceleration into positional data, accelerometers require fast integrators which should be located between the sensory equipment and the computer. The integrators must be quick and efficient enough to produce a consistent output for the user; inaccurate calculations of position could lead toward undesired movement of the cursor. The device must

also be consistently calibrated in order to find the logical zero for the devices.

For the application of this project, the use of accelerometers is not the most logical decision. The goal for a very discrete device relies on wireless communication and ideally a passive device. Using accelerometers, the chips must be constantly powered and must be equipped with an RF transmitter/receiver combination. This would not be conducive to a discrete solution. In the projects completed at other universities the sophistication of the devices was one of the desired goals of the projects. The purpose of this project is simple three-dimensional movement of a mouse cursor and perhaps a basic gesture for clicking, which does not require the implementation of a sensor on each finger.

2.6 Magnetics

The distance between a source and receiver can be determined in some cases based on the amplitude of the received signal. The amplitude of most analog signals and fields varies inversely with the distance from the source. Therefore, if the signal amplitude at the source is known, the distance from the receiver to the source can be calculated based on the amplitude received.

A magnetic field is a good example, since field strength varies inversely with the square of the distance from the source, as is shown in equation 1. If the value at one point is known, then the distance from the source can be calculated for any other point using this equation. By taking the electric field value at four different points, the distance from each point is obtained. Each of these distances represent the radius of a sphere, and the intersection of four or more of these spheres will uniquely define a point in 3D space.

$$\mathbf{B} = \frac{\mu I L \sin(\Theta)}{4\pi \mathbf{r}^2} \tag{1}$$

For each receiver, an expression of the form shown in equation 2 can be created to determine a sphere of radius r_a, where a denotes the specific receiver. The intersection of four or more of these spheres will uniquely define a point in space relative to the coordinate locations of the receivers.

$$r_a^2 = (x_a - x)^2 + (y_a - y)^2 + (z_a - z)^2 \quad (2)$$

To increase accuracy, additional receivers can be added. The additional data provided by an extra receiver can allow averaging and correction for errors that is not possible with the minimally determined case of only four receivers.

Electromagnetics

There are several ways to use electromagnetics to track an object. If a primary inductive coil is used to create a large magnetic field, eddy current detection of a metal object or an LC oscillating circuit can be used. Alternatively, if the tracking object is itself a magnet, magneto-resistive sensors can be used. The former relies on the induced current in a metal object in a time varying magnetic field, while the latter relies on a material that has a change in resistance in the presence of a magnetic field.

To track a metal object or an LC circuit, a large coil of ferrous material is necessary. When current is run through the coil, a magnetic field is induced. The shape of the magnetic field follows the right hand rule, and a tightly wound coil will have a field through its radial axis. When a metal object or a tuned LC circuit is placed inside the magnetic field a current will be induced in the object, and hence it will create its own magnetic field, which opposes the original. This opposing magnetic field changes the voltage across the inductor, and the magnitude of the change is proportional to the distance from the coil to the tracking object. Measuring the change in voltage allows the object to be tracked in one dimension. One-dimensional tracking is simple to implement, and the Biot-Savart law (which states that a tightly wound solenoid behaves in the same manner as a magnetic dipole) and Faraday's law provide equations to determine the changing voltages and magnetic fields.

A similar approach has been developed at Waseda University in Tokyo, Japan. To detect a position tracker inside the human body, they used primary coils of 1.3 mH carrying 2.5 A of current to produce a magnetic field around the body where the secondary tracking coil resides.

The induced voltage in the secondary coil is then modulated to FM frequencies using circuitry powered by a small watch battery, and the FM signal is picked up by detectors on the body to triangulate its position. The result of the experiment was a detectable range of 500 mm with an accuracy of 10 mm. This accuracy and range are within an acceptable range, and despite the reported large size of the system, its success proved promising for our experiments.

In order to track in three dimensions, the use of 4 coils plus an exciter object that would change the voltage across each coil is needed. The first coil would serve the same purpose as explained above, plus there would be three other coils placed along three axes. The first coil serves to power the three other coils, and an exciter object would be placed inside the magnetic field near the three perpendicular coils. Following the same principle described above, when the object moved closer to each coil the voltage across that coil would be lessened, and the three changes in voltages can be tracked to triangulate the object's position. At first this theory seemed viable and fairly easy to implement. However, with research we learned that when more magnetic fields are introduced to track an object in multiple axes, finding a function for the position becomes far more complicated, as every additional coil contributes to the magnetic field. Tracking would then have to be done through experimentation, noting voltage across the coils at each position and comparing the past and present values to find position. While this method is possible to implement, because of its complexity it was not a promising option.

An apparatus was constructed in order to test the theory that a metal object in a changing magnetic field could alter that field according to its position. The impetus of this experiment was based upon the idea that eddy currents in a metal object could alter the perceived field of a sensing coil whereby creating a voltage dependent upon the position of the metal object. The basis behind these magnetic field experiments was primarily due to the knowledge of electromagnetic field principles, and was validated by the content of prior art, namely US patents 7,096,148, 7,034,804, and 5,854,634. Patent 7,096,148, described in section 2.2 outlines a system of three excitation and three sensing coils to determine three-dimensional position.

Three coils were constructed, one transmitter and two receivers, as a proof of concept that

a metal object would distort the magnetic field between transmitter and receiver based upon its position in the field. The transmission coil X was constructed from 22 gauge magnet wire, wrapped 70 times around a hollow spool 8 cm in diameter. Receiver coil A was constructed from 18 gauge glass-cloth insulated transformer wire, wrapped around a spool 10 cm in diameter, which was later removed, leaving an air core. Receiver coil B was constructed with similar dimensions and materials as coil A, but did not have as many windings as coil A. The practical parameters for coils X, A, and B are described in table 2 where the resistance values of the wire are measured as DC resistances.

Coil	Inductance (L)	Resistance(R)
X	422.8 uH	0.7 Ω
A	4.08 mH	1.3 Ω
B	1.54 mH	0.7 Ω

Table 2: Practical Coil Parameters

In the first phase of the experiment, coil X and coil A were connected as the inductors of two separate parallel RLC circuits. The R value of each circuit was 1 kΩ. Using equation 3 with a desired frequency of 20 kHz, the capacitance value of the circuit containing coil X was determined to be approximately 98 nF. Because of capacitance value rounding, the new resonant frequency of the system shifted from 20 kHz to 25.4 kHz. This was determined by sweeping a 0.5 V peak-to-peak sine wave on the transmission circuit and testing the frequency that caused a voltage maximum. The capacitance value of the circuit containing coil A was 9.55 nF based upon the new resonant frequency.

$$f = \frac{1}{2\pi\sqrt{LC}} \tag{3}$$

With both coil X and coil A tuned to resonance at an input frequency of 25.4 kHz, coil A was separated from coil X by a distance of 1 m. The voltage measured across coil X was 100 V peak-to-peak at its resonance frequency of 24.5 KHz and amplitude of 0.5 V for the input voltage. This voltage was roughly measured using the peak-to-peak voltage measurement on oscilloscope channel 1. The oscilloscope was used to measure the voltage of coil A on channel

2. This voltage was measured to be 10 V peak-to-peak. A thin copper plate, 15 cm $\times$ 10 cm, was moved in the field region between coil X and coil A. At first, the plate was oriented such that the large flat surface was covering the opening of coil X. A significant change was noted in coil A voltage as the plate was moved both vertically and horizontally through the magnetic field in this orientation. Different orientations of the plate were tried, and it was determined that if the plate was parallel to the magnetic field, its area was reduced in correlation to the field, and its effect on coil A voltage minimized. Next, a copper bar was placed in the field at different orientations. The change in voltage on coil A was not as noticeable on the scope, and had a similar characteristic to the copper plate parallel to the magnetic field.

In the second phase of the experiment, coil A and coil B were oriented such that the opening of each coil was perpendicular to the other. They were oriented orthogonally to the opening of coil X approximately 20 cm above. Under this experiment, coil A and coil B were within a close enough proximity that resonance in these circuits was not necessary to induce a significant voltage from the magnetic field. A voltage of 0.5 V AC with a frequency of 24.5 KHz was applied to coil X, as in the first phase of the experiment. With this orientation, the copper plate was passed through the magnetic field created by coil X. The voltages of coil A and B were measured on the oscilloscope and were noted to change according to the distance and orientation of the copper plate in the field with respect to each coil. It was difficult to determine whether the magnetic fields produced by coils A and B from the induced voltages were interfering with each other, since the object within the field was non-uniform to all axes.

These magnetic field experiments have shown that an easily measurable voltage could be induced in a single coil at distances less than 1 m. The voltage was shown to be directly dependent upon the position of the object that was distorting the magnetic field. The apparatus that was constructed for this case was relatively easy to implement, and did not require complex circuitry, other than the equipment provided in any ECE lab in the Atwater Kent Building.

The magnetic field experiments also showed that the magnetic field decays rapidly over large distances. Ideally, a maximum voltage of 100 V peak-to-peak would be desired on the receiver

coil. In resonance, the receiver coil was only able to sense about 10% of the input voltage. The difficulty of attaining resonance for each coil, and keeping the entire system in tune was a unique challenge of this type of position sensing. The interference of the magnetic fields of coils A and B was not determined empirically, but theoretically they should interfere with each other since they are creating their own magnetic fields, whereby inducing a voltage on coils in close proximity. What was noted, most importantly, was that the voltages on each coil also changed with the orientation of the object in the magnetic field. The change in voltage on the coil also depended on the inductance of each coil. This resulted in different magnitudes of voltages being induced in each coil, and different rates of change in voltage due to the distance of the obstructing object.

The design challenges of implementing an accurate and robust three-dimensional position tracking system using magnetic field sensing outweigh its advantages. The main complexity of this system is that the voltages represented by each axis would not correspond to the absolute distance of the tracked object to each field sensor. The sensors, would otherwise provide a dynamic voltage output, customized to the precise construction of each coil, its orientation in the apparatus, and the type of interference of additional objects in the apparatus. The device would require calibration at the beginning of each session of use, unless other means were conceived that would reduce or eliminate the non-idealities introduced by this method of position sensing.

Magneto-Resistive

Magneto-resistive sensing would be the simplest electromagnetic choice, as magneto-resistive sensors exist in prepackaged ICs. The most popular manufacturer of these devices is Honeywell, and they start at a base of 2.5 volts, and have sensitivities of 1mV/V/Gauss (1 Tesla is 10,000 Gauss). Experimentation from the 2003 MQP found a usable change in voltage 10 cm from the ICs. Our own experimentation was planned, however the IC did not arrive in time to perform adequate tests. We believe this approach would be viable, but the magnet would have to be strong enough to create a change in the IC's output voltage at twelve or more inches from the PCB, and such a magnet would be a burden to use as a pointing device.

2.7 Time Difference of Arrival (TDOA)

Time Difference of Arrival (TDOA) at its most basic level involves the difference in arrival times of a signal at multiple receiver locations. The TDOA principle relies on calculating the distance from each point based on the time it takes for the signal to travel between the source and receiver. One of the points is specified as the reference point and all other points are compared to this one. Based on the propagation speed of the signal and the time difference between each point and the reference point, the distance of each point from the transmitter can be calculated. By receiving the signal at four or more points to determine three dimensions, this method allows the location to be calculated without knowing the initial time, T_0, that the signal was sent. This is especially beneficial in systems where there is no direct line of communication between the transmitter and receiver, because it would be difficult to determine the start time of the signal.

For our purposes, the handheld device would emit or reflect a signal that would be received by at least 4 receivers. The receivers provide the time of arrival for processing, which gives the time difference between the arrival of the signal at the different receivers. The following 4 equations relate the propagation time to the distance between each receiver and the transmitter. In these equations, *c* denotes the propagation speed of the signal and *x*, *y*, and *z* denote the unknown coordinates of the transmitter.

$$T_L = \frac{1}{c}\sqrt{(x_L - x)^2 + (y_L - y)^2 + (z_L - z)^2} \quad (4)$$

$$T_R = \frac{1}{c}\sqrt{(x_R - x)^2 + (y_R - y)^2 + (z_R - z)^2} \quad (5)$$

$$T_Q = \frac{1}{c}\sqrt{(x_Q - x)^2 + (y_Q - y)^2 + (z_Q - z)^2} \quad (6)$$

$$T_C = \frac{1}{c}\sqrt{x^2 + y^2 + z^2} \quad (7)$$

The following three equations calculate the TDOA between the reference receiver and each other receiver. By simultaneously solving these three equations for x, y, and z, the position of the transmitter can be accurately determined.

$$\tau_L = T_L - T_C = \frac{1}{c}\sqrt{(x_L - x)^2 + (y_L - y)^2 + (z_L - z)^2} - \sqrt{x^2 + y^2 + z^2} \quad (8)$$

$$\tau_R = T_R - T_C = \frac{1}{c}\sqrt{(x_R - x)^2 + (y_R - y)^2 + (z_R - z)^2} - \sqrt{x^2 + y^2 + z^2} \quad (9)$$

$$\tau_Q = T_Q - T_C = \frac{1}{c}\sqrt{(x_Q - x)^2 + (y_Q - y)^2 + (z_Q - z)^2} - \sqrt{x^2 + y^2 + z^2} \quad (10)$$

Although positioning can be adequately accomplished using only four receivers, the accuracy can be improved with additional receivers. If interference is introduced, it may cause the equations to not specify a single point, which would leave some room for uncertainty between the curves. Additional data points can help in these situations by allowing averaging or error correction.

Positioning using TDOA is generally quite accurate because it is relatively easy to measure time with a high degree of precision. Accuracy is primarily affected by geometry and timing. The location of the transmitter and the configuration of the receivers can have an effect on how accurate the system is. The accuracy of the system decreases at certain points due to geometrical effects and is worse when the transmitter is further from the center of the receiver array. Uncertainty about the locations of the receivers can also be a cause of error. Since this is a time-based operation, timing is the most prominent cause of inaccuracy. The timing accuracy for the received pulses is essential, but there also must be timely communication between the sensors and the processor. If the system is not designed well, unforeseen latency can occur within the processing unit.

The propagation speed of the signals being used can also have an effect on system performance. Sound signals will propagate more slowly than waves that move at the speed of light, increasing the time differences observed. The example below derives the required time resolution to achieve a given spatial accuracy. The constant r represents the propagation speed of the signal and d and t represent distance and time, respectively.

$$2\Delta d = r\Delta t \quad (11)$$

$$\Delta t = \frac{2\Delta d}{r} \tag{12}$$

The two solutions below show the time resolution needed for 1 mm spatial accuracy in a single dimension with different propagation speeds.

r = speed of light:

$$\Delta t = \frac{2 \times 10^{-3}}{2.998 \times 10^{8}} = 6.67\text{ps} \tag{13}$$

r = speed of sound:

$$\Delta t = \frac{2 \times 10^{-3}}{340.29} = 5.88\mu\text{s} \tag{14}$$

These examples emphasize the six orders of magnitude difference between using a low speed signal such as ultrasound versus an optical or electromagnetic signal that propagates at the speed of light. Having slower speeds to work with is beneficial, but there are also trade-offs with other factors that will be discussed later in this section.

Radio Frequency and RFID

Radio frequency and RFID tracking involve high frequency signals that are transmitted by one or more transmitters, altered by a passive device being tracked and received back at the transmitters. The primary difference between RF and RFID deals with how the signal is altered. With RF, the signal is simply reflected, whereas RFID tags modulate the transmitted signal with another one to send information. Both of these technologies would rely on TDOA principles to determine position and for the purposes of this discussion can be considered the same because they differ only in specifics of implementation.

The way RFID would be implemented for tracking in 3D is to have one reader capable of transmitting and receiving, one RFID tag and three or more receivers. A signal is sent from the reader and is modulated by the tag. The modulated signal would be received back at the reader and the other receivers at different times based on their distance from the tag. From the time

differences between the receivers, the position of the tag relative to the known receiver locations could be determined. The same principle applies to RF, using reflections instead of modulation.

To determine whether it would be possible to read an RFID signal with multiple receivers, we designed some basic tests using a 125 kHz RFID reader and tags. We contacted Jay Farmer, a WPI graduate student, about designing a near field receiver for our application. He walked us through some calculations to determine inductance and capacitance parameters. The equations below determine inductance and shunt capacitance based on frequency and relate the two values based on operating frequency f_0.

$$f_0 = \frac{1}{2\pi\sqrt{LC}} \tag{15}$$

$$LC = \frac{1}{4\pi^2 {f_0}^2} \tag{16}$$

$$C = \frac{1}{4\pi^2 L {f_0}^2} \tag{17}$$

$$L = \frac{1}{4\pi^2 C {f_0}^2} \tag{18}$$

Jay recommended using these equations to determine the necessary capacitance to complete a coil of known inductance and offered us use of an inductance meter. He was able to provide us with a pre-made coil, designed for similar frequencies. We measured the inductance of this coil to be 398.5 μH. The calculations below show how we determined the necessary value for the shunt capacitance, C.

$$C = \frac{1}{4\pi^2 \times 398.5 \times 10^{-6} \times (125 \times 10^3)^2} = 4.0681 \times 10^{-9} \approx 4\text{nF} \tag{19}$$

The primary goal of testing the multiple reader system was to determine whether we can receive the communication occurring between the RFID reader and tag with an additional read-only coil. The first test was performed with just the coil and capacitors connected to the oscilloscope.

It was possible to see signals being received by the coil and there was a change in the signal when the tag was introduced. However, at this point the signal could not be positively identified as modulation being introduced by the tag.

In an attempt to obtain more intelligible signals, we tried filtering the signal to make it easier to view. Following the information provided by the RFID Proximity Security System [12], we added some circuitry that acts like an envelope detector to make the modulation more visible. A series capacitor was also added to remove the DC offset of the signal. Figure 1 shows the result of applying this filter and clearly shows modulation being detected.

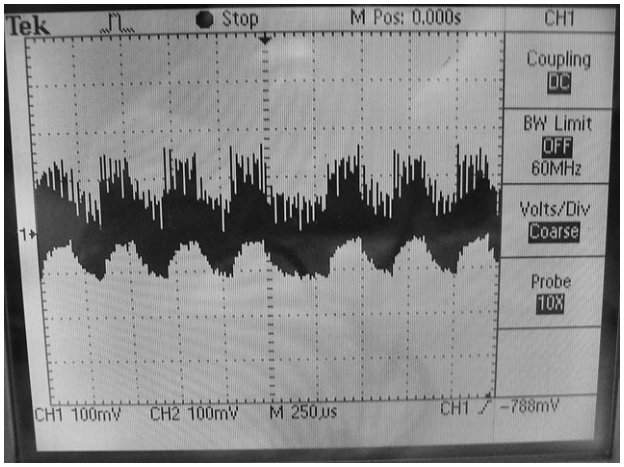


Figure 1: RFID Modulation

The above modulation appears to be a digital signal and is present whenever the RFID tag is communicating with the reader. When the tag is not within range, only the 125 kHz carrier signal is seen.

The fact that modulation can be seen on the additional antenna shows that using a read-only antenna is a valid means of picking up the communication between a reader and tag. This could be extended with additional read antennas to receive the modulated signal reflected off of the tag and the received signals can be compared to determine the tag's 3D location.

RFID can operate at one of a few established frequencies. After evaluating the different options, we determined that 13.54 MHz would be most effective. This frequency has a range of about one meter and is not prone to interference problems associated with higher frequencies.

In addition to only having a three inch read range, the 125 kHz reader we tested uses too long of a wavelength to discern the phase difference for small distances. With a wavelength of over 2 km, it would be very difficult to determine any phase difference from differences of only several centimeters.

$$\lambda = \frac{c}{f} = \frac{2.998 \times 10^8}{125000} = 2398.4\text{m} \tag{20}$$

The next logical step is to test a system using a 13.56 MHz carrier frequency. At 13.56 MHz, the wavelength is more manageable:

$$\lambda = \frac{c}{f} = \frac{2.998 \times 10^8}{13.56 \times 10^6} = 22.11\text{m} \tag{21}$$

Using RFID as the technology for our project has some potential, but would certainly come with many problems. The major difficulties lie in accurately measuring the phase difference between signals and sampling the signals. These difficulties would certainly take some work to overcome and must be weighed against other technologies.

2.7.1 Ultrasonic

TDOA positioning can also be performed using ultrasonic waves. Ultrasonic waves propagate at the speed of sound, which simplifies detection, however there are still some implementation concerns. This section details the testing of a basic ultrasonic TDOA system.

The ultrasonic transducers we used for initial testing were graciously provided by the Machine Vision Lab and include one transmitter and two receivers. Each of the transducers had been built into a complete unit with appropriate support circuitry and necessary interconnects and have the following specifications.

Transmitter Specifications:

- Supply Voltage: $\pm$ 12V
- Input Signal: 3V pk-pk continuous, 5V pk-pk intermittent (60 sec)

  Max DC Offset: 0.5V

- Pass Band: Approx 42 kHz to 48 kHz (-10 dB)

Receiver Specifications:

- Supply Voltage: $\pm$ 12V
- Output Signal: 12V pk-pk max
- Pass Band: Approx 42 kHz to 48 kHz (-10 dB)

A sine wave of appropriate frequency (42-48 kHz) from a signal generator was provided as input to the transmitter and viewed on channel 1 of the oscilloscope. The output of the receiver was connected to channel 2 of the oscilloscope and the signals were monitored and compared with the scope triggered on channel 1. Basic transmission was effective for both receivers, however at these frequencies, obstructions such as a human hand cause significant interference. Since these specific transducers are only omnidirectional in two dimensions, significant offset in the Z direction will cause signal deterioration.

After basic communication was established, transmission to two receivers was tested by viewing the two received signals as the transmitter was moved. The two receivers were set up along a measured line and the transmitter was slid alongside the line going between them. Observing the time delay between the two signals provides information about the difference in distance from the transmitter to the two receivers, following the equation below (d in meters, t in seconds):

$$\Delta d = 344 \times \Delta t \tag{22}$$

Transmission to two receivers was effective and showed very useful results. Both signals could be seen on the scope and if the receivers were at different distances, they can be told apart by their amplitudes. By triggering the scope on one of the channels and moving the transmitter along the workbench, the phase between the two signals could be seen to change proportionately to the movement of the transmitter.

Although the movement could be viewed on the oscilloscope, there was no way to determine the exact location of the transmitter due to the small wavelength as calculated below.

$$\lambda = \frac{c}{f} = \frac{344}{45000} = 7.64\text{mm} \tag{23}$$

Having such a short wavelength means that single phase tracking is only effective for within a single wavelength, however, time difference in this case could be determined by measuring arrival time of a pulse rather than the phase of a constant signal.

Ultrasonic tracking worked very effectively and is relatively easy to work with because of the slow signal speed. Since the device being tracked is a transmitter, it requires power and cannot be a passive device. The transmitter we tested was rather large and required a lot of power, however there are other options available that are much smaller. A carefully designed custom transmitter that is designed for close-range use could potentially be made very small.

2.7.2 TDOA Conclusions

The best TDOA options appear to be RFID and Ultrasonic, however, they both have their benefits and drawbacks. RFID has the advantages of being passive and not susceptible to most common forms of interference, however the signals propagate very quickly and would be difficult to develop the device. Ultrasound has a much slower propagation speed, but suffers from problems with objects blocking the communication path and cannot be implemented passively. To decide between these two, we will have to decide which qualities we value the most and choose which technology we feel would be most successful considering our goals and time frame.

3 Methodology

The aim of this section is to document the logical methods, theory, and group discussion behind important aspects of the design. The technology choice for the transmission method is discussed and selected. This section outlines the requirements necessary for the design to complete a working prototype, and is organized into functional blocks. It also discusses the considerations taken that best meet these requirements. Furthermore, this section discusses integration of all of the blocks into a complete system and testing plans to determine the level of performance for individual blocks and the entire system.

3.1 Project Management

With a team of five students working with many facets of the project from different computers, it is crucial to have a plan for data management to prevent loss of data and reversible changes. We have utilized open source tools such as LaTeX in conjunction with Subversion (SVN) for data management for this project. Specifically, an SVN server was run through the department's UNIX machine `maxwell.ece.wpi.edu`. This allowed each of the members of the `magicmouse` group to access the data files through Secure Shell (SSH).

By using LaTeX to author the document, we can benefit from the excellent rendering of figures, equations and the structure it provides. Also, by placing the master document on the server directly, we can rely on the redundancy of the ECE servers for data reliability. The document was structured in a hierarchy with the master file referencing each section and subsection. This allows different members of the team to work on different portions of the paper simultaneously. Each TeX file for the document was under version control using subversion. Using subversion, each member could check out a local copy of the document and only commit files which they have edited. Also, because the document is maintained on the server, reviews by the advisor were as simple as checking out and compiling the document. Although only one member of the group had extensive LaTeX experience, the others were able to learn the application during the course of the project.

Using Subversion to establish and maintain the document became second nature to the team, and it was logical to extend its capabilities. We have employed version control on our firmware code for the microcontrollers used in the design. This provides the opportunity to commit changes when the code has reached certain milestones. When the code is then broken, it can be easily reverted back to a working state without lost time. Additionally, due to the ease of using subversion repositories on the ECE servers, It was simple to create a way to keep track of revisions to the printed circuit boards. Although these files are binary, we can still keep track of prior versions and easily reference or revert to them. Also saving the state of the board when it was sent to be produced is an advantage in case one needs to refer to an old board's schematic.

Overall the small amount of additional time spent establishing the skeletal structure for the LaTeX document and creating SVN repositories prevented any data loss catastrophes that could have occurred otherwise.

Project Overview

Bringing a design concept from an idea to the creation of the device begins with a foundation in background research pertinent to that idea. From this foundation, a decision must be made that determines the direction of the project. In many designs, this is the critical step, and eventually determines whether or not the idea may be feasibly realized. If the idea passes this step, this decision will set the stage for the rest of the design. A handy metaphor for this process is comparing the design to pitching a tent. The foundation of background research provides stable, solid ground on which to construct the design. The debate over transmission technology discussed in the background section is a debate over where to stake down the first pole of the tent. Since subsequent decisions on the design are directly related to the transmission technology, it is a rational place to start. From the first pole, all other poles may be placed in reference to it. From here, the remaining challenge is building the tent, or physically constructing the design.

The Competitive Value Analysis located in Figure 2 is a graphical depiction of the thought process that went into the transmission technology decision. The most important criteria are ease

of implementation, position accuracy, size of the transmitter, transmitter power consumption, and total cost. The two major goals of the project are to create a device that accurately detects the position of a wireless object while using as little power as possible to do so. These criteria are weighted more favorably than the other, secondary criteria. Each technology is rated on a scale from 1 to 5, where 5 is the best score and their weighted scores in each area are summed. In theory, the technology with the highest score is considered the best option.

Competitive Value Analysis

	Weight	Magnetic		RF/RFID		Ultrasonic		Accelerometer		Gyroscopic	
Criteria:		Raw	Weighted	Raw	Weighted	Raw	Weighted	Raw	Weighted	Raw	Weighted
Ease	4	3	12	2	8	4	16	4	16	3	12
Accuracy	5	2	10	4	20	4	20	5	25	4	20
Size	4	3	12	5	20	4	16	4	16	3	12
Power	5	4	20	5	25	4	20	2	10	2	10
Cost	4	3	12	2	8	3	12	3	12	3	12
Total Score:			66		81		84		79		66

Figure 2: Competitive Value Analysis

Ultimately, the transmission technology selected for this design is ultrasound. The electromagnetic options were tested in the lab, and proved to be too inaccurate for 3D position measurements. The implementation of these designs also proved to be challenging, especially in creating a passive device that would create fluctuations in an induced magnetic field. Three-dimensional positioning using gyroscopic sensors does not fit into the limits of a low or no-power device which is one of the major goals of the project. The use of accelerometers fits into the same category. Implementation of either of these strategies, though they are remarkably accurate, would require active RF communication. RFID, or passive RF technologies are difficult to implement, and depending upon the implementation may prove costly. Ultrasound fits the middle of the design criterion, while only excelling in a single category, emerged as an overall winner.

The implied ultrasonic mouse design has been broken down into several functional blocks. Each block achieves a different function toward realizing a final prototype. With the exception of the power block the design proceeds linearly with the transmitted pulse of the signal, and provides a manageable structure to divide workload. These functional blocks are broken down

into transmitter, receiver signal processing, TDOA processing, PC interface, and power blocks. The system block diagram is shown in Figure 3.

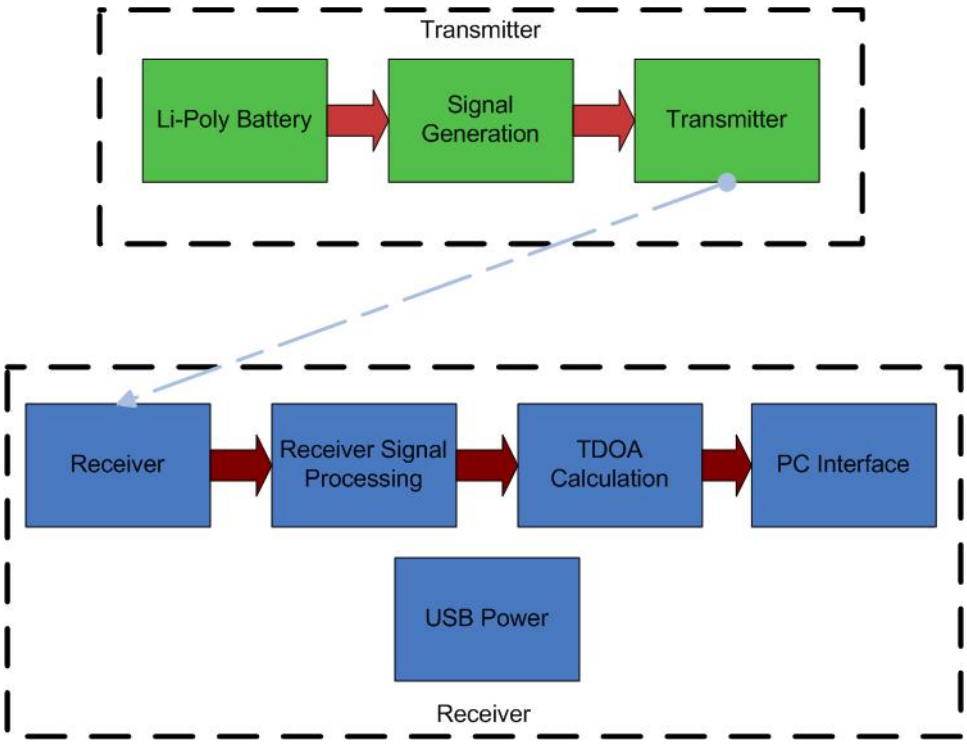


Figure 3: System Block Diagram

3.2 Power Requirements

This section details the power requirements for the transmitter and receiver portions of the system.

3.2.1 Transmitter Power

The power requirements for the transmitter are largely dictated by the physical specifications needed to meet the design goals. The device needs to be small, and for prolonged use, it must run on low power. Conventionally the best way to meet these goals would be to use a button cell lithium-ion battery.

Button cell batteries, which are commonly used in watches, hearing aids, and other small, low-power devices, have typical outside diameters ranging 4 mm to 32 mm, and have nominal

voltages of 1.5 V or 3 V. The capacity of non-rechargeable button cells is typically in the 100's of mAh range for the size we need, but the rechargeable options have on average one-tenth of that capacity. For this reason we chose to pursue further research, and found Lithium Polymer batteries to be a viable option.

Created in the late 1990's Lithium Polymer batteries have a lower capacity, but more flexibility and lighter weight than their traditional Lithium Ion counterparts. While they are still gaining popularity, the typical applications for Lithium Polymer batteries are in small-scale electronics, such as cellular phones and remote-controlled aircraft. They suit this purpose because they can be custom made in any shape, are inherently rechargeable, and priced competitively with existing technology. They are uniquely flexible because in polymer cells external pressure is not required as the electrode sheets and the separator sheets are laminated onto each other. They are typically available to consumers in rectangular blocks, with smaller cells at the range of 900 mm^3 ($5 \times 12 \times 24$ mm). The nominal voltage of nearly all small-scale Lithium Polymer batteries is 3.7 V, and their capacity ranges from 50 mAh at 900 mm^3 to 620 mAh at 6600 mm^3.

These blocks would be too large for this project, and ordering a custom battery would not be feasible for a prototype, but further research led us to PowerStream Technology, a distributor of ultra-thin Lithium Polymer cells with thicknesses ranging from 1 mm down to 500 microns. Normally these cells would have to be ordered in quantities of at least 100, but a small quantity of one model, the PGEB0052081, can be ordered for electrical testing. This cell has dimensions of $0.5 \times 81 \times 20$ mm, and has a capacity of 45 mAh. After calculating the expected power consumption of the transmitter unit to be less than 1 mA this capacity yields over 45 hours of continuous use and is appropriate for the requirements of the device.

These batteries would not be of much use if they cannot be easily recharged. Research on the charging of Lithium Polymer batteries led to the MAX1555 IC, which will safely charge the battery from either a computer USB port or 5 V DC power supply. The small size of this IC allows for the possibility of including a charging station on the receiver module without interfering with the rest of the components.

3.2.2 Receiver Power

Powering the receivers is a much simpler task. Section 7.2.1 of the USB 2.0 specification document defines a unit load to be 100 mA, and states that each port can source "up to five unit loads" [15], and based on our preliminary power consumption estimates, the four receiver modules, PIC microcontroller, and battery charger will use significantly less than that, particularly because the charger, which draws 100 mA, will not be used at the same time as the rest of the circuitry, which uses less than 100 mA.

3.3 Transmitter

The purpose of the transmitter is to send a periodic signal that can accurately be detected by the receiver. This section describes the requirements for the transmitter subsystem and how the methods of implementation were selected.

Figure 4 shows a block diagram of the transmitter subsystem. A signal is generated which is then put through some drive circuitry to create a better signal for the transducer, which in turn converts the signal into ultrasound waves capable of propagating through the air.

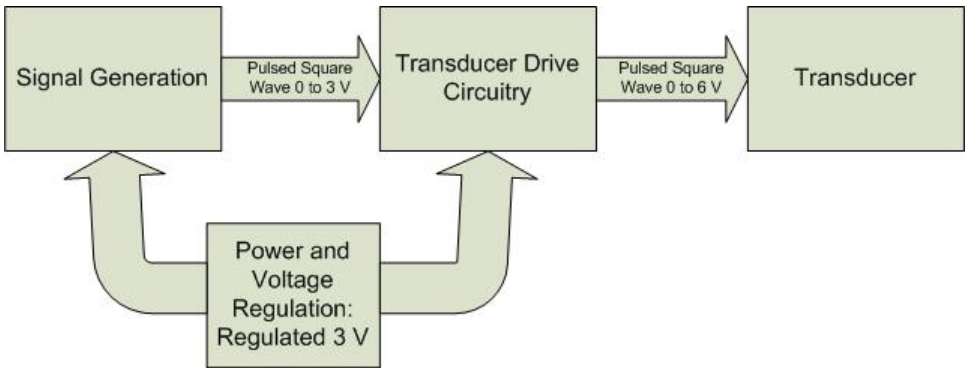


Figure 4: Transmitter Block Diagram

Transmitter Requirements

One of the main goals for this project requires that the transmitter is a small, unobtrusive device. It is clearly important to keep this in mind during the entire process of designing the

transmitter. To this end, the transmitter needs to be made as small and low-power as possible, while still fulfilling the necessary requirement of producing a pulsed signal in the ultrasonic frequency range.

3.3.1 Transducer Selection

Ultrasound is defined as sound waves in the frequency range above that of human hearing, or over 20 kHz. An ultrasonic transducer works exactly like a speaker or microphone, converting electrical energy to and from mechanical waves. These devices are designed to have a specific resonant frequency that is in the range above typical human hearing. This section details the relevant characteristics of these devices and describes the selection of an appropriate set of transducers for this system.

There are a number of different frequencies that can be used for ultrasonic systems, however nearly all economically priced transducers operate at 40 kHz. Typical transducer prices for 40 kHz devices are in the $5 to $20 range, whereas transducers for other frequencies tend to be $50 or more. Because the needs of this project are not extremely specific in regards to frequency and several transducers are required, it makes sense to use the most cost effective option, which is 40 kHz.

There are two main types of ultrasonic transducers: open and enclosed. Open transducers have their piezo device and cone exposed to the air behind a screen. Enclosed transducers have the piezo device mounted directly to the outer case, which is designed to resonate. Enclosed transducers are useful for harsh conditions, but have the drawbacks of being less sensitive and more expensive. Open transducers are generally inexpensive and have very good performance. For these reasons, an open transducer is the clear choice for this application.

When selecting ultrasonic transducers there are a few different performance parameters to consider:

- Sensitivity and Sound Pressure
- Directivity

- Bandwidth

Sensitivity is a measurement of a receiver's ability to convert mechanical waves into an electrical signal. It is typically measured as a negative decibel (dB) value relative to the transmitted signal, indicating the attenuation at the receiver. Sound pressure level (SPL) is the equivalent measure for a transmitter. This value, in decibels, describes the output sound pressure that the transmitter is capable of producing. These parameters often include graphs to show how the sensitivity and SPL vary as a function of frequency.

Directivity describes how directionally dependent a transducer's response is. It is generally expressed as a graph showing the attenuation in dB for 360° around the transducer. A transducer with more omnidirectional directionality will have a wider range with little or no attenuation. For some applications, it would be desirable to have a directional transducer to avoid interference, however that is not the case for this design. The configuration of this system will require that the transducers are operated primarily in the 30° to 45° range. To achieve minimal attenuation, it is important that the transducers have good directivity and do not attenuate the signal too much within this range of angles. Some transducers have dead spots in their range where very high attenuation occurs between the center and side lobes. For this system, this is not desirable because it could produce undetectable areas within the range of the mouse.

Bandwidth determines the range of frequencies outside of the center frequency where the attenuation is less than 6 dB. This essentially dictates the usable frequency range and is most important in applications where frequency modulation is used. For this design, only the center frequency will be used, so bandwidth is not an important consideration for transmission, however it should be considered in terms of how the transducers act as a bandpass filter for the receiver.

There exists a range of different sizes for ultrasonic transducers, the smallest being cylinders about 10 mm in diameter and 10 mm deep. SPL and sensitivity tend to decrease with smaller sizes and in many cases qualities such as directionality vary between differently sized transducers. Although small size is definitely desirable for this design, the other qualities of the transducer must also be adequate.

3.3.2 Signal Generation Method

There are essentially two general methods to generate the desired signal consisting of pulses of a 40 kHz square wave or sinusoid. The first option is to generate a signal external to the microcontroller and switch it using a transistor and a digital output on the microcontroller. The second method involves generating a 40 kHz clock signal as an output of one of the microcontroller ports.

A couple different methods of generating signals externally were first explored, including crystal oscillators and timer integrated circuits. We tried a variety of different crystal oscillator configurations and were unable to get the crystal to produce the desired signal. It is likely that these problems were due to not having the proper components to excite the crystal. A timer circuit was also created with a 555 timer. This circuit produced a 40 kHz signal with nearly a 50% duty cycle, however the current draw of about 3 mA would be completely unacceptable for this battery-powered application.

After exploring options for external signal generation, it became apparent that even though it may be possible this way, there would be more effective options. This led to the exploration of options using a microcontroller to generate the entire signal. The microcontroller option has the advantages of low power consumption, fewer components and a more flexible design that can generally be altered with a few lines of code rather than changing components. One way to accomplish this is to manually toggle an output port in firmware, however implementing this can be time consuming and sensitive to code errors.

TI's MSP430 chips offer a simpler and more effective option for accomplishing this goal by routing the clock signal to an output port. During the course of our research, we came across a document produced by Texas Instruments (TI) entitled Ultrasonic Distance Measurement with the MSP430 [10]. This document provides schematics and code for the implementation of a complete one-dimensional ultrasonic ranging system. It is designed to bounce an ultrasonic pulse off of an object and then receive the reflected signal back to calculate the distance to the object. Most importantly, it describes how to configure an MSP430 to route the signal from its clock

crystal to an output port. This method is the best choice for this application because it can be controlled very precisely and requires few components and minimal power.

Hardware Design

Although TI's document has a wealth of useful information, it is not a complete design for this exact application. Some circuit design is necessary to set up the microcontroller appropriately and adapt the circuitry TI uses to fit the needs of this project.

The first step to designing the hardware is to determine the necessary blocks within this part of the system. It is then necessary to determine which microcontroller pins are needed and how they must be connected. After these steps have been accomplished it is possible to design the specific hardware for each block, draw up a schematic and build a prototype. A working prototype can then be made into a printed circuit board, which would permit the use of small surface mount components and could shrink the design to an acceptable size.

3.3.3 Firmware Design

Figure 5 shows a block diagram of the transmitter firmware. It begins with an initialization stage, where all of the outputs and functionality are configured. The main loop performs the transmit pulses. The interrupt service routines handle interrupts to bring the microcontroller out of low power modes that it enters when it is between pulses.

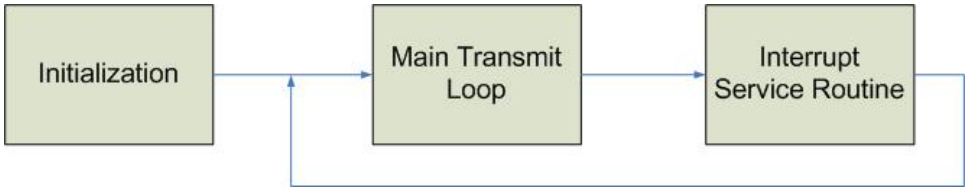


Figure 5: Transmitter Firmware Block Diagram

3.4 Receiver and Signal Conditioning

The goal of the receiver is to preserve the timing information of each 40kHz signal pulse and convert the pulses into a format that the decision device within the dsPIC can translate into time

delays. This section defines the specific requirements of such a system and outlines the process behind its implementation.

The block diagram of the system may be seen in Figure 6. The pulsed signal is received through the receiver transducer array, amplified through an AC amplifier, rectified, filtered, and digitized. The block diagram represents the operative flow of the signal through the system for one of the five receivers. All five receivers are constructed from this basic design.

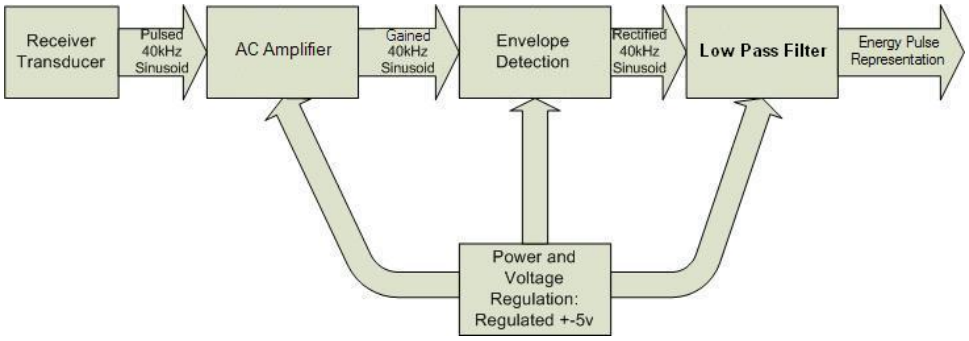


Figure 6: Receiver Block Diagram

3.4.1 Signal Conditioning Requirements

The receiver signal processing is required to amplify and convert each pulse into information that may be digitized. It is also required to preserve the pulse peak time throughout the system, by both minimizing distortion of the signal and propagation delay deviation.

3.4.2 Signal Demodulation

The demodulation hardware approach is comparable to that of other acoustic TDOA systems. These approaches involve minimal analog front-end amplification, and immediate digital acquisition. [9] They differ from the design used in this project in that they utilize standard digital acquisition cards to sample the data and run TDOA calculations on a PC. [17] Within the constraints of conducting all TDOA calculations outside of the PC, this leaves calculations to be conducted on a DSP chip, or other dedicated microprocessor.

There are two major approaches to demodulation architecture. The first approach is to use

Digital Signal Processing and coherent phase detection algorithms in software to determine precise phase delay and to minimize error. The second approach is to use discrete analog components to pre-process the signal before it becomes digitized. Pre-processing may involve coherent or incoherent detection with a square-law device.

Coherent detection first multiplies the incoming signal with the expected pulse signal, integrates the product, and detects the peak of the signal, either directly or through a curve fitting algorithm. The time at which the pulse has arrived may be ascertained from the peak, as it is directly related to the final edge of the received pulse. Since the pulse length is the same on all receivers, the peak should correspond to the time at which the pulse is received on each receiver. When the initial receiver peak time is subtracted off of any subsequent receiver peak time, the time delay associated with distance may be acquired.

Coherent phase detection on a DSP chip has the advantage over analog coherent phase detection in that it can normalize the amplitude of the result and determine the peak of each signal more accurately in software. The processing that occurs within a DSP chip is also more resistant to circuit noise than analog components if properly configured. Since many operations that would normally require three or four discrete hardware components can be created and changed easily in software, the development time of processes that involve these functions is greatly reduced. The only drawback of using a DSP is that most DSP chips only offer ADC speeds of about 400 KSPS. Using a simple direct sampling method requires that each channel is sampled at least 1 MSPS for accurate peak detection and position determination. A regressive curve fitting method of peak estimation may allow slower sampling speeds with little decrease in performance. The ADC is required to sample at at least 1 MSPS per channel in order to accommodate direct sampling peak detection since curve fitting peak detection has higher development time, complexity, and risk.

Incoherent detection utilizes a square law device to rectify the signal. The major difference between coherent and incoherent detection is that the signal is rectified instead of multiplied for correlation. The rectification process is similar to taking the square root of the square of the

signal, or the absolute value function. The signal is then filtered, in this case, to standardize the pulse shape at 40 kHz. Similar to coherent detection, the peak of the pulse is then determined, either through direct sampling or curve fitting regression. The main advantage of the incoherent detection approach is that it is inexpensive compared to coherent detection. This is due to the minimization of receiver complexity by not requiring a sync to the carrier frequency, otherwise necessary with coherent detection methods. Disadvantages of incoherent detection are that the phase information of the carrier is distorted, and it is not as resistant to multipath.

Balancing factors of cost, sampling speed, and accurate pulse time detection, a unique mixed signal approach was taken. While coherent detection with a DSP chip or discrete analog hardware is accurate and more efficient, both options were priced. For a coherent detection method implemented with discrete components each receiver would cost $40, totaling $200 for 5 necessary receivers. This is almost one third of the project budget, and prototyping costs weigh heavily against this option. DSP chips that meet the requirements of the design are expensive, require supporting components, and often come in packages that are difficult to solder. For these reasons, incoherent detection techniques were employed. First, the signal is received on the receiver transducer. It is amplified using an AC amplifier, with a tuned maximum gain at 40 kHz. The amplified signal is rectified, removing negative components. The signal is then filtered to create a recognizable pulse, with the peak corresponding to the end of the received signal pulse. This pulse has a much sharper peak than the original that was received and aids the sampling block in peak determination. The sampling block digitizes the signal and utilizes a microcontroller to achieve peak detection.

3.4.3 Data Transmission

The first priority of the system was to determine positional data, with digital data transmission as a secondary goal. The transmission protocol was selected with these considerations in mind. For data transmission, however, FSK, BPSK, and OOK protocols were considered. The receiver transducers involved in this project are bandwidth limited to ± 1 kHz and may be modeled as

high order bandpass filters around a center frequency of 40 kHz. The factors of low bandwidth, a low-power transmitter, and receiver simplicity were of highest priority when making the selection. An FSK receiver would require a phase locked loop, or other phase matching component to accurately define transmitted symbols. A BPSK receiver would require a phase detection scheme. With an acoustic wave that is constantly changing phase with transmitter position, BPSK would be difficult to implement. For the reasons of receiver simplicity, scalability, and practicality, the OOK approach was selected. The trade off for this method was a decrease in signal to noise ratio versus FSK or BPSK methods. The operating range of the device is small enough such that the received signal may be amplified and filtered from any ultrasonic room noise at 40 kHz, and the drop in signal to noise ratio is an issue that can be managed. With an OOK transmission method, multiple pulses could be transmitted per transmission cycle. The first pulse could be used for location determination, with additional pulses transmitted to encode mouse click or mode data. While the possibility of digital data transmission was accounted for in the transmission protocol, and a button input built into the transmitter for this purpose, encoded data transmission is not necessary and was left as a future improvement.

3.5 Signal Processing

The purpose of the processing block is to calculate the location of the transmitter based on the received signals. Using the output of the receivers, this block generates five time delays. From these delays it uses TDOA calculations to determine the location of the transmitter.

3.5.1 Processing Requirements

Many of the functionality goals of the mouse must be met in this block. The main constraints are how much time is available for processing data and how fast the data can be sampled. Both of these factors determine the resolution of the mouse. An updated transmitter location must be sent to the computer at least once every 16 ms to ensure that the cursor movement appears smooth on a typical 60 Hz monitor. To calculate the location of the transmitter we need to know when the five signals were received in relation to one another. The differential time values must

have a high resolution for the TDOA calculations to work with minimal error. The faster the channels are sampled, the higher the time resolution will be, which will in turn improve spacial resolution.

This block is broken down further into three smaller sections. The three blocks concerned with processing are sampling, curve fitting and TDOA calculations. The sampling block takes in the five receiver channels and samples them into a data array in memory. The curve fitting block takes this data and processes it to determine the precise peak time for each signal. The TDOA block uses these time delays to calculate the 3D location of the transmitter.

3.5.2 Sampling and Curve Fitting

The sampling block takes in five conditioned pulses from the analog signal conditioning. To process the data further, the five channels must be converted into digital data. To accomplish this, the analog stream of data is fed into a device that can convert from analog voltages into quantized digital numbers representing these voltages.

The analog-to-digital converter (ADC) can only convert one channel at a time so the five channels need to be combined into a single stream. Using a multiplexer, the five channels can be separately selected to be passed on to the output. With proper timing, the multiplexer channel is selected and once the ADC has converted that signal, the processor receives a digital version of the signal. Using this method, the processor is able to receive data from all five channels.

Although the amplitude of the pulses varies based on their power, these pulses have a consistent shape based on the previous stage. This attribute can be used to determine the peak of the pulses very precisely by comparing the received signal to a precise known signal.

The next step is to determine precisely when each signal was received. Because the square wave pulses sent by the transmitter are bandwidth-limited by the transducers, a sharp edge cannot be produced and the received pulse resembles a smooth curve. Since this curve's voltage is based on the power of the input signal and varies with the distance and angle between the transmitter and receiver, a voltage level can not be used to determine the peak of a received signal. However,

the signal's shape is not based on the input power and remains consistent from one pulse to the next. The time at which the signal reaches a peak only varies with the time at which the signal is received. This means the peak of the signal can be used to determine when the signal was received.

A method called curve fitting can be used to determine the time a pulse is received very accurately. This algorithm compares a low resolution received signal to a stored higher resolution signal and uses the relationship between the two to determine the time that the peak is received. After detecting the time that pulses were received on all five channels, they must be converted to time differences. By subtracting the time one pulse was received from all of the others, the time difference can be quickly determined.

3.5.3 TDOA Calculation

Once all of the receiver time delays have been determined, TDOA calculations are used to develop an estimate of where the transmitter is located. In these equations, c represents the propagation speed of sounds waves in air. The L,R,Q,W, and C starting coordinates are all initial conditions in the system. The C on corresponds to the center point which determines the relative location for all delays to be calculated. The center point will be a static receiver because all of the relational positions of the receivers are based on the center receiver being at the coordinate origin (0,0,0). This center point does not necessarily need to be located in the center of all the receivers, but it is considered the center because all other time delays are based on their relation to the time delay measured at this receiver.

Solving for the (x, y, z) coordinates with the standard TDOA equations is computationally intensive to implement on a microcontroller, and the processor would take far too long to complete the computations for this to be an effective solution. Fortunately, a resource for a more efficient solution came from Ben Woodacre, a WPI graduate student, who has worked on a similar TDOA issue. He uses an algorithm for TDOA calculations by John D. Bard, which avoids difficult differential equations and instead determines a solution using matrix and algebraic math [1].

From this paper Ben Woodacre wrote a MATLAB program that calculates the location of a transmitter called `bard_solver`.

Appendix B.5 shows a MATLAB version of the code that can be used to calculate positions using the TDOA method. The program takes three inputs, the first of which is a matrix containing the `[x,y,z]` coordinates of the receivers in the system. The second input is time delay between the selected receiver and the receiver that is defined as the origin. The third input is the pulse transmission rate. Since the transmission is performed using ultrasound, the speed of sound in millimeters per second is used for this value. This value is dependent on the ambient temperature as shown in equation 24, and for room temperature $(25°C)$ results in the speed of sound being approximately 346,000 mm/s.

$$c_{sound} = 0.3315\sqrt{1 + \frac{T}{273.15}}\text{mm/s} \tag{24}$$

The `bard_solver` has no unit requirement, so the units that are entered are carried through to the output. The data is entered in units of millimeters, seconds, and millimeters per second so the differential output of the transmitter location is expressed in millimeters.

The coordinates obtained at this point will be relative to the receiver set as the origin. The next step is to create an operational zone that is unobstructed by the receivers. The operational zone is an imaginary box where the user will be able to interact with the computer. Unfortunately, the area directly between all of the receivers cannot be used, even though it is the optimal placement for TDOA calculations. This is because the ultrasonic transducers have a specific transmission and receiving directional range of about $45°$ from their center axis. This limits the possible locations for this device, so the operational zone must be offset away from the receivers. The operational zone is shown in Figure 7. This shows the offsets added to the random data points to simulate the location of the operational area.

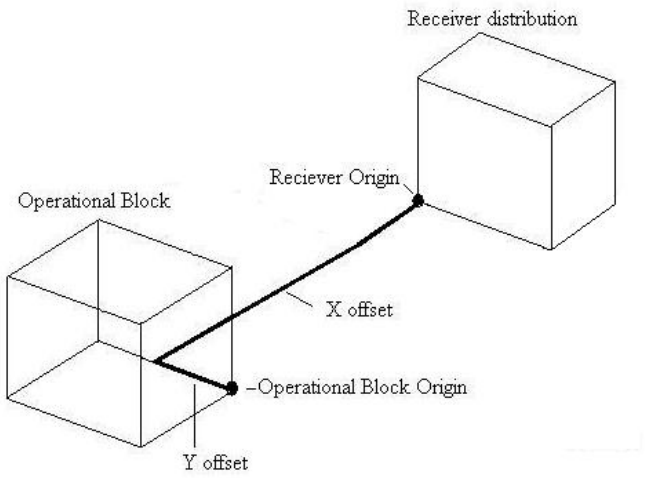


Figure 7: Operational Block

3.6 PC Interfacing

The requirements for the connection between the microcontroller and the PC dictate the interface. The available options for PC communication are the PS/2 port, USB ports, or the Serial COM port. The original mouse was developed for serial communications over the COM port which communicated with the computer using the RS-232 protocol. Data is transmitted bi-directionally through 9 pins on the port, including control signals. The availability of power over the serial line is limited to about 10 mA at 12 V, which is not sufficient for powering the microcontroller and sensors. The largest problem however with using a serial mouse is that it is an old standard, and supports a limited two buttons and two axes. Although the interface and the transmitted packet could be defined, this method would require the development of a specialized driver for the device thereby eliminating any entirely portable applications. The portability of this device is one of the driving requirements for the design, so the PS/2 interface would also

not be the best choice for the interface. PS/2 connections are not hot-pluggable and therefore cannot be connected to the computer while it is on. This leaves as the only logical solution for implementation the USB interface.

The universal serial bus (USB) is a versatile standard which devices of almost any kind can operate on. This has become the standard for peripheral communication on modern computers. Input devices have migrated from older PS/2 and serial standards to USB, of which one of the most notable advantages is hot-plug capability. The USB interface supports four major modes of data transfer: Control, Bulk, Interrupt, and Isochronous. Without going into extensive detail, the implementation of these transfer procedures exemplifies the 'Universal' nature of the interface. Luckily, USB devices are also classified based on their use; Human Interface Devices, or HIDs are a class of USB devices which categorizes interaction between the computer and the user. Such devices include the mouse, keyboard, joysticks, remote controls, bar-code readers, etc. HIDs running at full speed can transfer 64,000 bytes per second, whereas a low speed device can operate at 800 bytes per second. Because of the nature of this project, the HID classification of devices will be the most logical path if USB is to be utilized as an interface. The primary reason to implement using USB-HID is reducing the need to develop specialized drivers for the device. The HID standard is open enough to support input devices with many axes of freedom, and as many logical buttons as one pleases and will be the method of connectivity for this device [15].

The USB port will typically supply 100 mA at 5 V to the attached device and will supply a maximum of 500 mA. Devices such as external hard drives which consume more than 500mA of power must be self-powered while connected to the USB hub. The wireless mouse will not exceed the 500 mA supplied by the USB hub and therefore will operate sufficiently with hub power. The USB connector takes four lines; the mouse will use the V_{cc} and GND ports for power and communicate with the PC using the data lines.

USB can run at two speeds: low-speed (USB 1.1) and full-speed (USB 2.0). The speed of the device is interpreted by the PC using one of two pull-up resistor configurations. Figure 8 shows the configurations for each device speed. Once the speed of the device has been decided, the

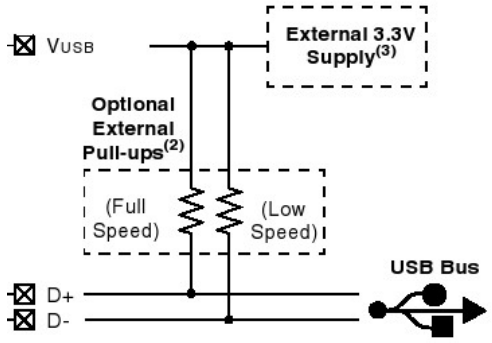


Figure 8: USB Device Speed Configurations[7]

device is then responsible for transmitting data on the data lines at the appropriate frequency. These two requirements are the sole differences between implementing the devices as high-speed versus low-speed.

3.6.1 Hardware Requirements

The major requirement for programming a USB capable device is a microprocessor which supports the USB protocol. This narrows the choice of microprocessors substantially, but if requirements between blocks conflict, a separate USB chip could be implemented such as the Maxim MAX3420E, a USB peripheral controller. In order for a microcontroller to support the USB interface, it must contain the function calls and data structures that allow it to handshake with the computer and then transfer data.

The major disadvantage to a USB implementation is the complication of sending information according to the USB protocol [15]. Implementation using C compiler packages makes developing an interface as simple as several function calls. When the device connects to the system, the system checks the impedance of the input rails to determine the communication speed of the peripheral. Once the speed is determined, all forward communications are at 48MHz or 11MHz for USB 2.0 or USB 1.1, respectively. The host then asks the device for a blueprint of information that it plans to send with each packet called a descriptor. When the host receives the descriptor from the client it then assigns it an address and loads the required driver on its system to communicate with the device. The driver sends the last known or default configuration to the

device, and the connection is then established.

Any communication between the device and the PC is done through descriptors. Upon each transaction, a structure of data is transmitted over the bus which contains any data such as positional information, button status, or LCD display information. In the case of the computer mouse, the movement coordinates and button status are sent to the host with each descriptor.

4 Implementation

The previous chapter outlined the methods and requirements for each subsystem block. In this chapter we discuss the realization of these methods. The implementation of these blocks has been defined in a way that documents the functionality and expected results of each section. The following sections include a discussion of power modules on both the transmitter and receiver devices, the transmitter ring design, receiver signal conditioning, sampling methods, TDOA processing, and interfacing the design with the PC. Each section outlines the challenges associated with its implementation and as these designs are only part of a prototype, their shortcomings are also discussed. Finally, ideas are suggested on how to improve the design, with a working prototype as the desired final result.

4.1 Power Block Implementation

To begin realizing the power specifications, a visualization of the final product is needed. Prior to the initial research into the transmitter power system, it was assumed that the source would be a button cell, which would rest on top on the user's finger. With the discovery of lithium polymer batteries that can be designed to take any shape comes a radically changed design perception. Because of the flexible nature of lithium polymer cells, nearly any mold is possible for commercial development, and ideally the prototype should closely match the design for a completed product. The final visualization for the end product is a battery that doubles as the ring itself, with a gap that allows for size adjustment. The prototype realizes this concept almost exactly. Ultra thin cells from PowerStream electronics are 500 microns thick, and can be bent into a ring, as shown in figures 9(a) and 9(b). With a capacity of 45 mAh, they are perfect for the task of running a low power microprocessor and ultrasonic transducer. The PCB with the transmitting components will rest on the battery, on top of the user's finger.

(a) Top View

(b) Battery Shaped as a Ring

Figure 9: Lithium Polymer battery

The PCB includes charging circuitry to allow the battery to be charged right on the receiver. It uses a MAX1555 charging IC, which is designed to charge lithium polymer batteries. The charging circuitry is powered by the USB power supply provided by the board.

As for powering the receiver module, the USB port used for PC interfacing will source the entire board. Initial testing of the components set the current draw to be between 100 and 200 mA, depending on whether or not the battery is charging. A single USB port can source up to 500 mA, therefore the use of USB power for the receiver is acceptable.

4.2 Transmitter

The implementation of the transmitter involves selecting a microcontroller, programming it to produce proper output, and designing the supporting hardware to run the microcontroller and drive the transducer from the generated signal. This section details the transducer and microcontroller selection, signal design specifics, hardware design, and firmware implementation required to create a complete transmitter unit.

4.2.1 Transducer Selection and Testing

Based on the transducer parameters described in section 3.3.1 and research of numerous transducer options, we selected a pair of transducers produced by Kobitone with the transmitter and receiver part numbers 255-400ST12 and 255-400SR12, respectively. See Appendix A.1 for

the data sheet for these devices. These transducers are ideal for our purpose because of their directionality, size and cost. They are more omnidirectional than most other transducers, having minimal attenuation in the range of 0° to 45°. They also happen to be some of the smallest transducers available, at a size of 9.9×12.7 mm. As an added bonus, they are relatively inexpensive at only \$4.79 each for low quantities.

According to the data sheets for the transducers, their center frequency is 40.0 kHz, however it is still important to know the precise center frequency as well as the bandwidth and frequency response.

This specification was verified experimentally by performing a frequency sweep. This was done by setting the transmitter and receiver at a fixed distance apart facing each other and slowly sweeping the input frequency through a range of values. Measurements were taken every 100 Hz from 38 kHz to 42 kHz and additionally every 50 Hz between 39.50 kHz and 40.50 kHz. The output was monitored to see at which frequency ranges the highest response occurred. In these tests, the best response was at 40.3 kHz, with very minimal attenuation at 40.0 kHz.

Figure 10 shows the frequency response of the transducers in decibels. The peak is fairly flat from 40.00 to 40.45 kHz and drops off steeply on either side. The -6 dB points are at about 40.0 kHz and 40.5 kHz. The attenuation rate in the stop bands is approximately -6 dB per 100 Hz.

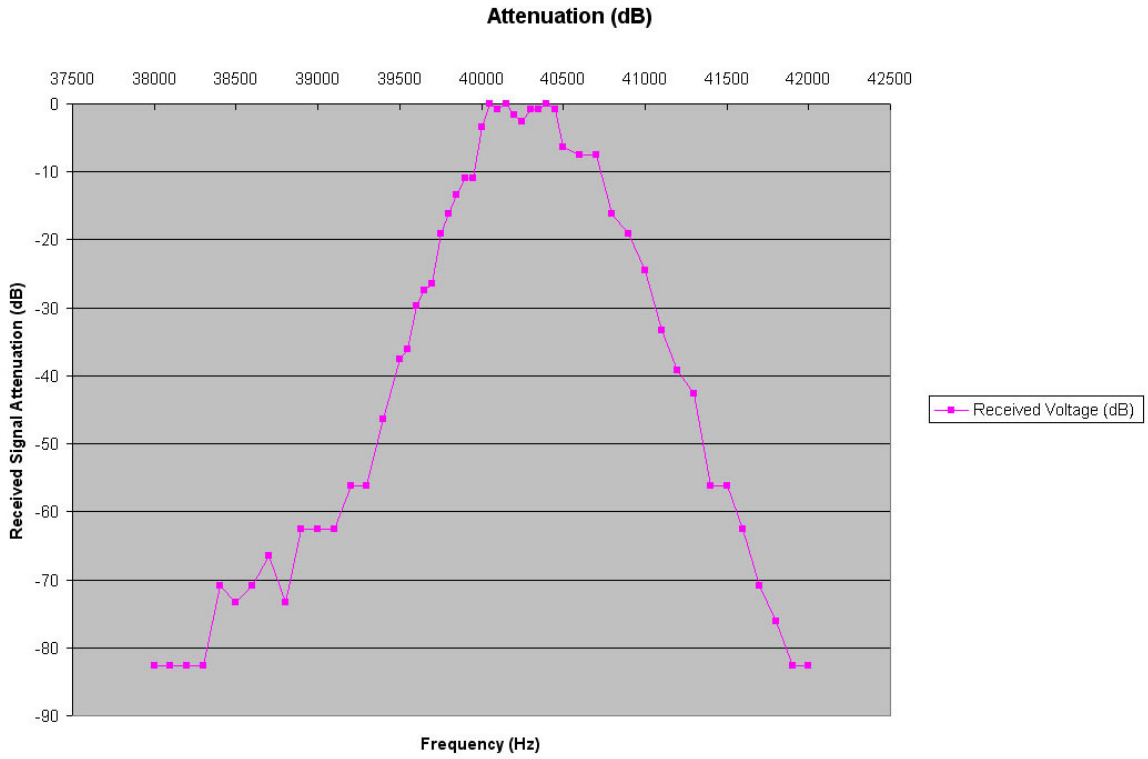


Figure 10: Transmitter Frequency Response (dB)

These results are perfect for this application because of the center frequency and frequency response. The transducers perform well at 40 kHz, which is the set frequency that will be provided to the transducers. The fact that the response drops off very quickly outside of the center frequency allows the transducers to operate as a tight bandpass filter, blocking any acoustic noise outside of the signal range. This eliminates some components from the analog signal processing section because an initial acoustic bandpass filter is not required.

4.2.2 Transmitter Microcontroller Selection

When selecting a microcontroller (MCU), there are several requirements to keep in mind while evaluating available options. The device must be low power and come in a small, low pin-count package. It should have an external crystal connection for a 40 kHz crystal and should have at least one available digital Input/Output line. Additionally, it should be able to be programmed using tools and software that are inexpensive or already available to us.

A variety of different chips from several manufacturers such as Microchip, TI, Maxim and EM Microelectronic were explored. Common problems with many chips were the lack of an external crystal connection and unavailability of programming tools. EM Microelectronic makes a microcontroller that comes in an 8 pin package and draws a mere 2 μA of current, however programming tools for it are prohibitively expensive, which ruled it out as an option.

After evaluating several different options, it was determined that the best choice is TI's MSP430 series of microcontrollers, and more specifically, the MSP430F2xxx series. The MSP430 is a ultra-low power mixed signal microcontroller that offers a large product line with varied features and multiple low power modes for power savings. The F2xxx series is one of TI's newer lines of flash-based chips and includes most of the lowest pin count MSP430 chips. For the purposes of this project, all of the MSP430F20xx chips would perform exactly the same, so the F2013 was selected because it is the standard chip offered with TI's USB programmer. This chip runs on 220 μA in active mode and less than 1 μA in some of its low power modes. It is available in 14-pin packages in both PDIP and TSSOP, which makes it easy to develop with, yet small enough for final designs. It also has the ability to accept an external crystal. MSP430 chips can be programmed either using a USB key Spy Bi-Wire interface or a parallel port JTAG programmer. JTAG cables are available in WPI's ECE labs and TI offers a development kit with a USB programmer for only $20, which was purchased for testing.

An additional benefit of using an MSP430 chip is that TI provides numerous application notes, including one describing the implementation of an ultrasonic range-finder using an MSP430 [10]. Although this document is written in reference to a higher pin count MSP430F413, it is still entirely applicable to the smaller versions such as the MSP430F2013. For the transmit side of TI's described design, the only necessary pins are the Xin and Xout external crystal connections and the auxiliary clock (ACLK) port, which are included on all MSP430 chips. The only major differences in implementation will be some slight changes to the code and hardware to account for different pin assignments and chip features.

Transmit Signal Design

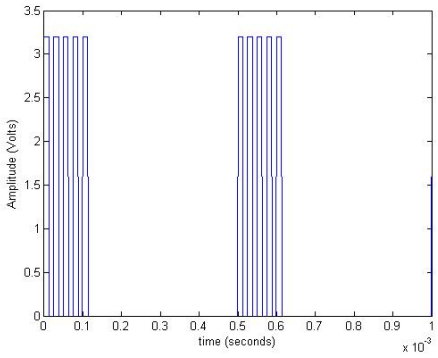
In choosing a signal to send to the receiver, a balance must be struck between accuracy and transmission rate. Because transmissions are being sent as square pulses, the duration of the signal determines the bandwidth and how accurately it will be received. On the other hand, a longer signal means a slower transmission rate and could cause the device to perform slowly.

Simulink simulations were run to determine that a pulse duration of 20 cycles of the 40 kHz wave would be adequate to produce a usable signal. These results are supported by the TI application note, which suggests a pulse duration of only 12 cycles [10]. Twelve cycles could be used if necessary, however 20 cycles provides a larger bandwidth without having an unreasonably long period, lending to higher accuracy.

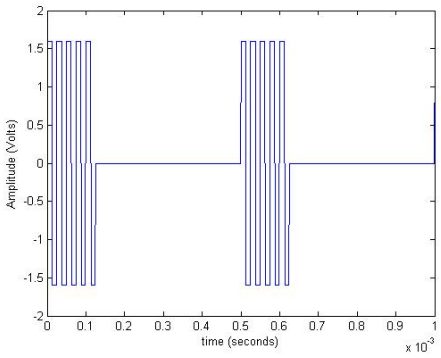
It was determined that a pulse rate of 100 Hz would be adequate, since it is faster than the refresh rate of nearly all computer monitors. This will provide a maximum amount of processing time, while still allowing the device to perform smoothly. This transmit rate translates to a pause of 380 cycles between pulses.

After the pulse is produced by the microcontroller, it is beneficial to amplify the signal to a higher voltage for increased output power. In most applications, this could be done using an amplifier or by switching a higher supply voltage. However, these options require a higher voltage power supply and this device is limited to the voltage provided by the battery. A more creative approach is introduced in TI's application note [10]. This method splits the signal into two paths, one of which is given a 180° phase shift using a logic inverter. When these two parts are added back together, they add to a voltage of twice the original signal. This higher voltage increases the output power of the transmitter, increasing the signal strength and range.

Figures 11(a) and 11(b) show simulations of the ideal signal output from the MCU and to the transducer.



(a) Ideal MCU Signal Output



(b) Ideal Output to Transducer

Figure 11: Transmitter Output

The schematic in Appendix A.3 shows all of the components used in the transmitter system design. The output of the MCU at pin 2 drives a transistor to provide adequate current for the transducer. The set of inverters serves to double the signal to a 6 V signal.

4.2.3 Firmware Implementation

To produce the proper signal from the microcontroller, firmware must be written to control it. The purpose of the firmware is to configure the microcontroller, set up its clocks and timing, and tell it to output the appropriate signals.

The flowchart in figure 12 shows the overall design of the firmware with minimal details. The program essentially loops through code that sends pulses continuously.

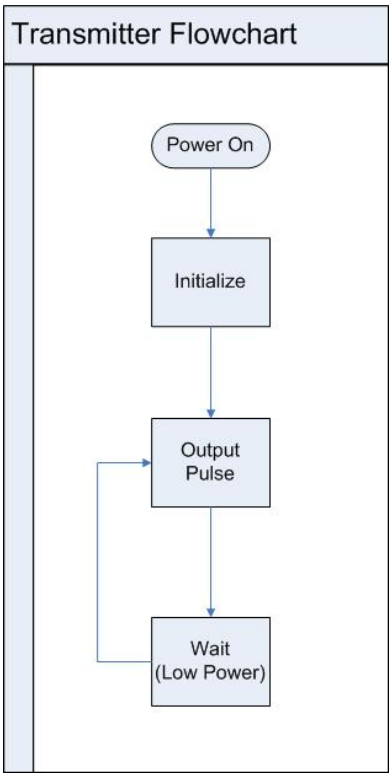


Figure 12: Transmitter Firmware Flowchart

The firmware is written using assembly language because of the control and efficiency gained by coding in assembly. The device's auxiliary clock (ACLK) is set to run off of the 40 kHz crystal that is attached to the MPU. The Capture/Compare Register (CCR) is used to count up to a specified value at a rate of one count per ACLK cycle and then trigger an interrupt.

The firmware sends a signal by following these steps:

1. Set Port 1.0 to output ACLK
2. Enter low power mode (LPM3)
3. Set CCR to count 20 ACLK cycles then interrupt to come out of low power mode
4. Turn off ACLK output on Port 1.0 then enter low power mode
5. Set CCR to count 380 ACLK cycles then interrupt
6. Repeat

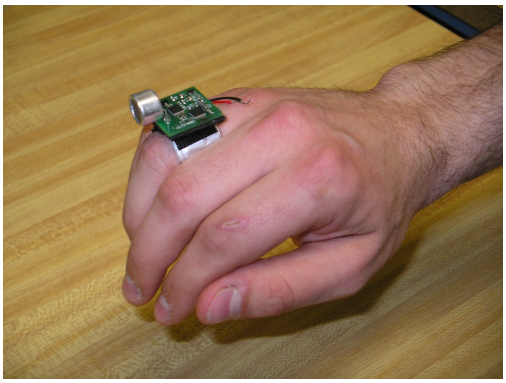
The complete firmware code can be seen in Appendix A.5

4.2.4 Board Layout and Physical Design

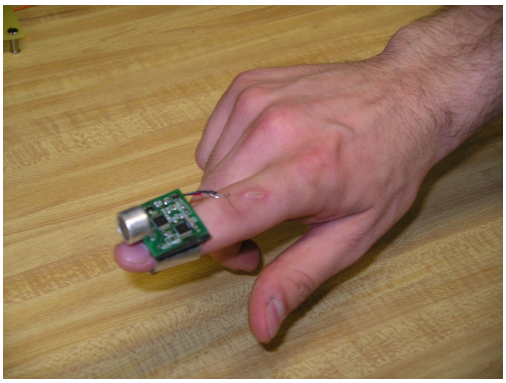
Appendix A.4 shows the PCB layout for the transmitter. The board size is 0.9 $\times$ 0.9 in, making it small enough that it fits into the design of the transmitter ring and is unobtrusive.

There are a few physical considerations that were taken into account for the design of the transmitter. How the ring will be worn, placement of the transducer and how the board will interface with the battery are issues that must be addressed.

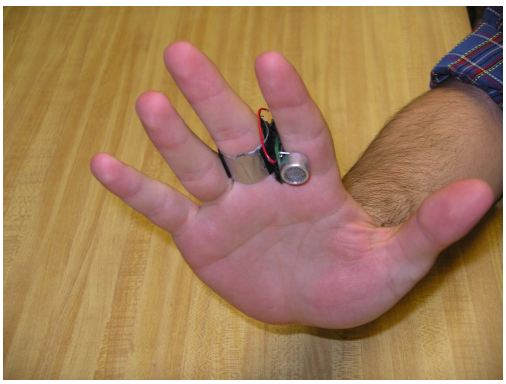
There are three different ways that the ring could potentially be used. It could be worn as a regular ring, mounted on the fingertip, or used on the palm side of the hand. All three of these positions Figures 13(a) to 13(c) show the different ways in which the ring can be worn.



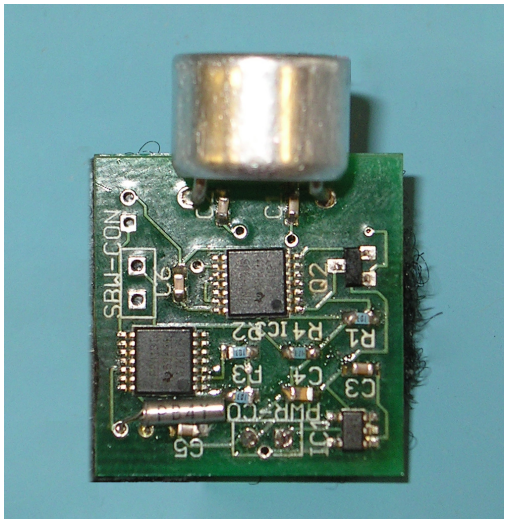
(a) Ring Worn on Finger



(b) Ring Worn on Fingertip



(c) Ring Worn on Palm



(d) Transmitter PCB

Figure 13: Transmitter Ring

The transducer should be placed on the circuit board in a manner such that the board will not be within the 45° transmit cone. The transducer's leads are bent 90° to allow it to be mounted facing forward and it is be mounted on the front edge of the board, hanging over the front edge by several millimeters.

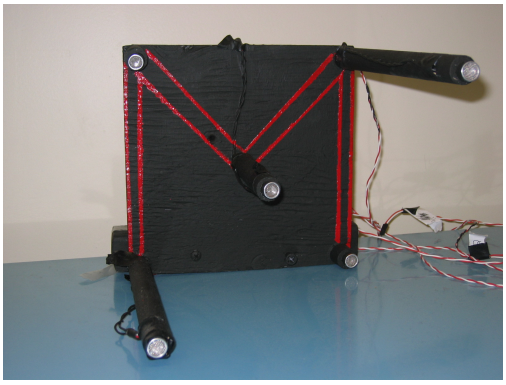
Figure 13(d) shows the populated transmitter PCB. Because the leads on the battery are on its front edge and spaced too far apart to be close to each other when the battery is wrapped around the user's finger, it is be necessary to extend the leads. Small wire leads are run from the battery leads to the PCB along the bottom of the board

4.3 Receiver Signal Conditioning

This section covers the initial stages of receiver, including the receiver array and the circuit implementation of the receiver analog signal processing. It has been decided that a mixed signal envelope detection scheme would be most applicable for this design due to its low cost. The number of receivers may be scaled with little consequence to the project budget. The functional range of the device is limited only by the AC Amplifier gain.

4.3.1 Receiver Array

The receiver array was first implemented with plywood and dowels to serve initial testing needs while the design for a more sturdy and precise array could be made. The first array is shown in 14(a). Using T-Frame extrusions from 80/20 Inc., and aluminum tubes cut with a CNC machine, an array was constructed that is structurally sound while allowing flexibility in the array design. This is because the transducer extrusions can be adjusted in increments of less than 1 mm, and the lengths can be interchanged for testing and optimization. A picture of this array can be seen in figure 14(b).



(a) Wooden Array (rev1)

(b) Metal Array (rev2)

Figure 14: Receiver Arrays

The functional circuit block diagram for an example receiver is shown in Figure 15. The signal enters the transducer, which acts like a bandpass filter centered around 40kHz. The signal is passed through an AC amplifier, which blocks any DC component and amplifies the signal

around 40 kHz. From this stage, the Precision Rectifier rectifies the signal. The signal then passes through a low pass filter, which transforms the rectified 40 kHz wave into a single sine wave shaped pulse. The peak of this signal corresponds to the signal that was on the receiver transducer, relative to other receiver channels. The signals are then digitized by the analog multiplexer and ADC.

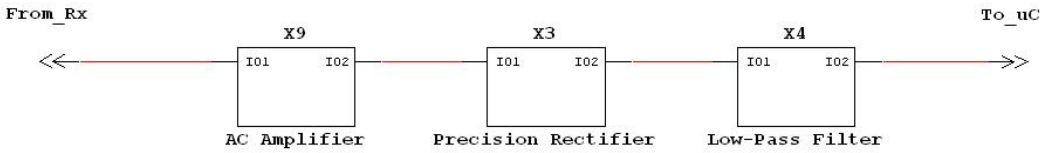


Figure 15: Receiver Signal Processing Schematic

4.3.2 Analog Components

The AC Amplifier section consists of a DC blocking high-pass filter, and a frequency dependent non-inverting gain amplifier. Experimentally, a minute DC offset voltage was observed on the signal from the transducer, and when gained by the ordinary non-inverting gain amplifier, the op amp saturates. The AC amplifier effectively blocks this DC offset voltage before it reaches the positive voltage input of the op amp, and ensures that the 40 kHz component of the signal receives the full gain of this stage. There is also a DC offset voltage caused by the non-ideality of the op amp. In order to prevent the DC offset voltage from becoming gained, a capacitor is placed in the negative feedback loop. This causes the DC gain of the op amp to be 1, and is illustrated in Equation 29.

The AC amplifier design is shown in Figure 16.

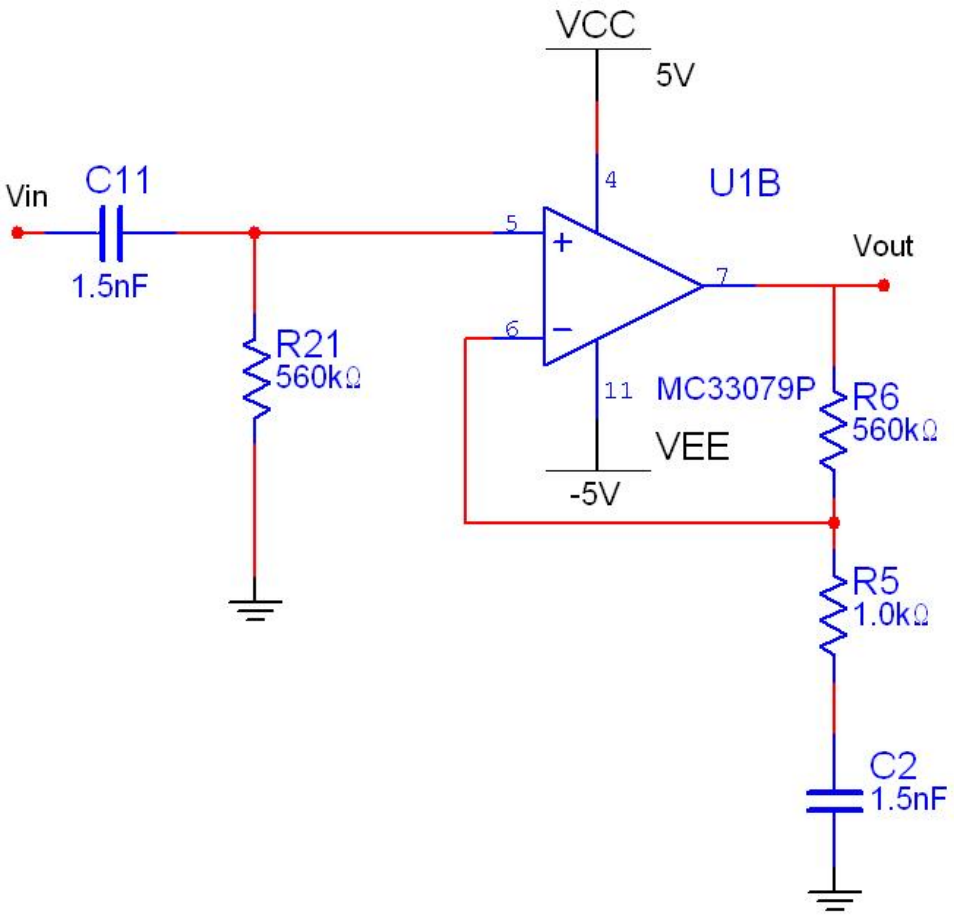


Figure 16: AC Amplifier

The governing equation of the high pass filter section is:

$$f_o = \frac{1}{2\pi R_{21} C_{11}} \text{Hz} \tag{25}$$

The R value was chosen to be 1 kΩ, and the cutoff frequency was 40 kHz. The capacitance was computed from the above equation to be 3.9 nF.

The governing equation of the non-inverting gain amplifier is first determined by:

$$f_o = \frac{1}{2\pi R_5 C_2} \text{Hz} \tag{26}$$

These two values set the frequency at which the gain is at its highest. Substituting in

impedance values for C_2 into the standard non-inverting gain equation leaves:

$$A = \frac{R_6}{R_5 + \frac{1}{\omega C_2}} + 1 \tag{27}$$

The overall transfer function of the AC amplifier is:

$$H(\omega) = \frac{R_{21}C_{11}\omega(R_5C_2\omega + R_6C_2\omega + 1)}{(1 + R_{21}C_{11}\omega)(R_6C_2\omega + 1)} \tag{28}$$

R_5 was selected to be 1 kΩ and C_2 was calculated to be 15 nF. The 3 dB bandwidth of H(ω) is 14 kHz. From a DC standpoint, the capacitor in the feedback loop appears to be an open circuit. This simplifies the gain equation in this case to:

$$A = \frac{R_6}{R_5 + \infty} + 1 \tag{29}$$

$$A = 1 \tag{30}$$

With properly chosen resistor and capacitor values, the AC amplifier stage amplifies the 40 kHz signal the most and blocks DC entirely. The transfer function of the AC amplifier is shown in Figure 17.

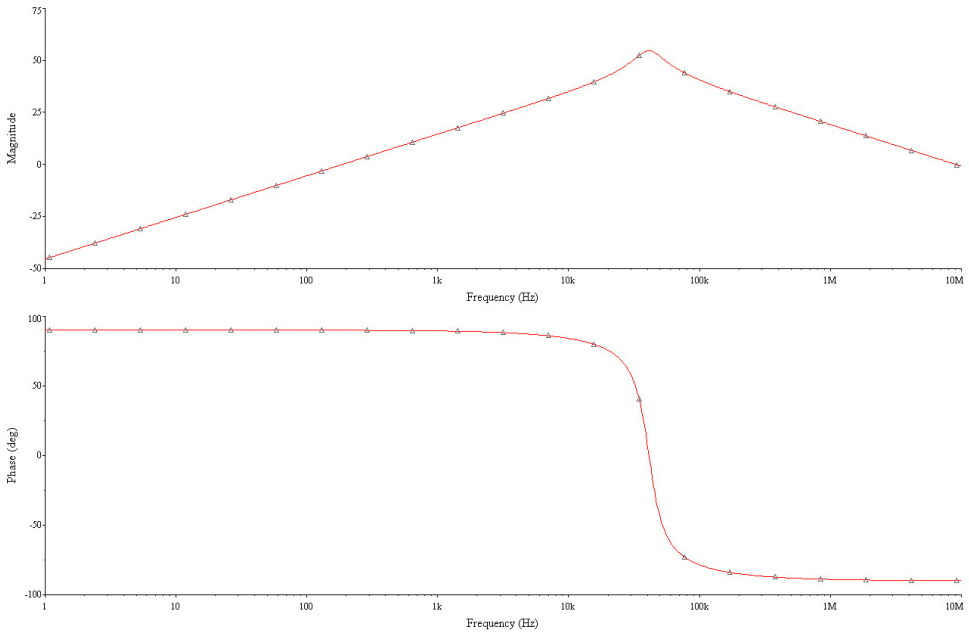


Figure 17: AC Amplifier Transfer Function

The R_6 value was determined experimentally from range tests conducted on the receiver prototype to be 560 kΩ. A gain of 500 at 40 kHz allows a received signal to be resolved by the dsPIC sampling algorithm for a receiver distance up to 1.5 meters.

Differential Amplifier

A second configuration that was considered was a differential amplifier. The purpose of using a differential amplifier was noted when the first revision of the PCB was constructed and tested. During the testing of this PCB, there was a level of unacceptable noise found on the ground lines of the AC Amplifier stage. The differential amplifier design uses a differential input of the transducer, instead of referencing the signal gain to the circuit ground. The main advantage of using a Differential Amplifier is that the common-mode noise can be substantially rejected. The diagram of this design may be found in Figure 18.

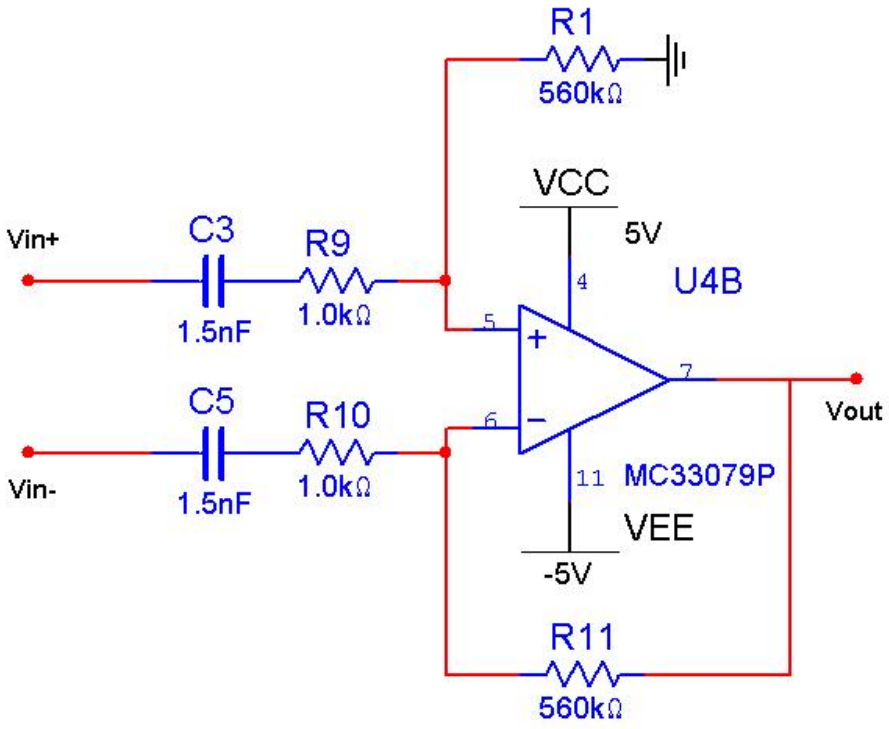


Figure 18: Differential Amplifier

The magnitude of the transfer functions for the AC amplifier and Differential Amplifier are the same, given the resistor and capacitor values in Figure 18. The second revision of the PCB has the potential to be switched from an AC amplifier to Differential Amplifier configuration.

Precision Rectifier

The Precision Rectifier stage consists of two diodes, two resistors, and an op-amp. This section is the first part of the envelope detection circuit. The diodes leave only the positive part of the sinusoid, and the op-amp is included to reduce the forward voltage drop to almost zero. This greatly improves the range of the transmitter and alleviates inclusion of a gain stage after rectification. The schematic for the precision rectifier is shown in Figure 19.

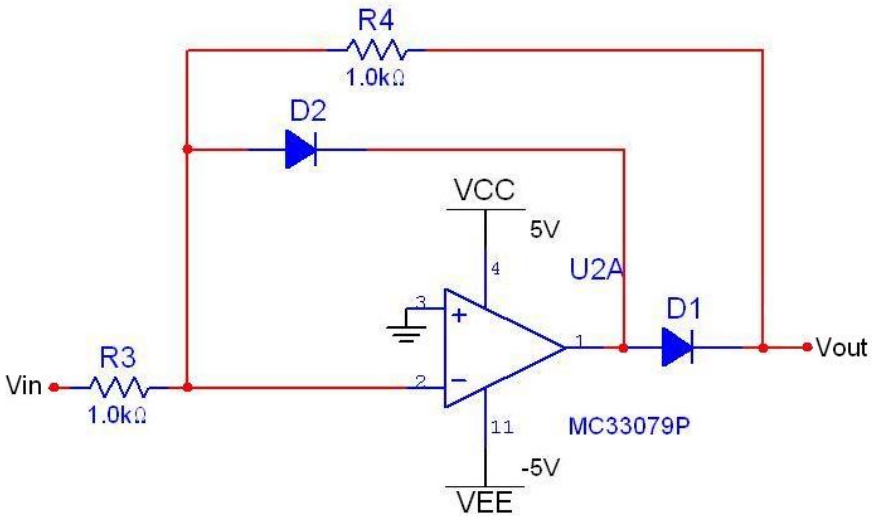


Figure 19: Precision Rectifier

The ideal gain equation for this circuit is:

$$G = \frac{R_4}{R_3} \tag{31}$$

The gain of this stage was selected to be unity, since the preceding AC Amplifier stage had sufficiently amplified the signal.

Low Pass Filter

The final design of the low pass filter block is a Chebyshev low pass filter design. The Chebyshev low pass filter has the benefits of a sharp roll-off after the cutoff frequency. The transfer function is shown in Figure 20.

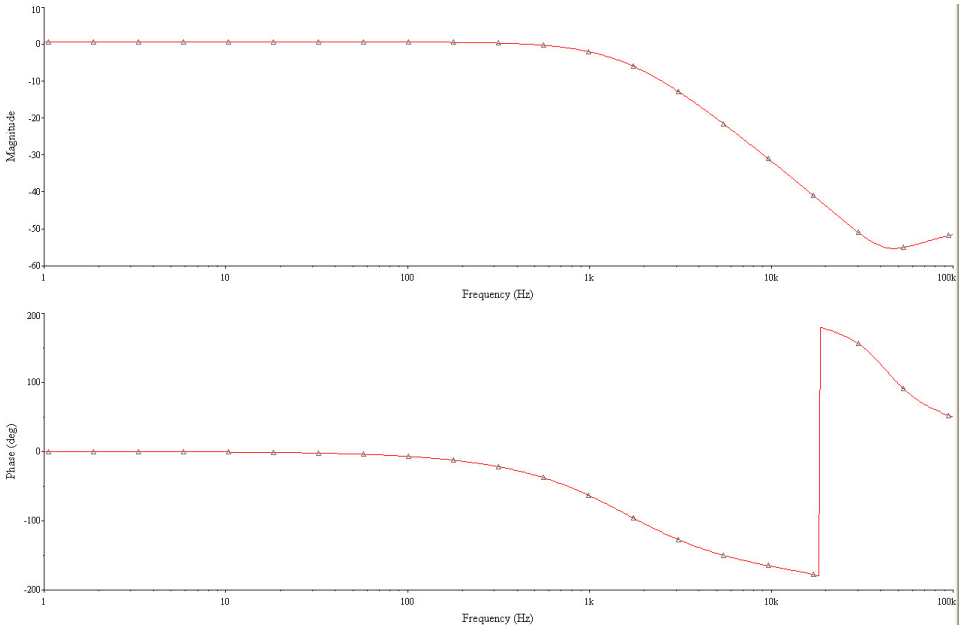


Figure 20: Chebyshev LPF Transfer Function

The design maximizes the attenuation of the signal after the cutoff frequency, f_c. The Chebyshev Filter was chosen for this characteristic in order to attenuate the 40 kHz carrier signal. The linearity of the phase of the signal is of little importance because the signal is band limited to ± 1 kHz. The diagram of the Chebyshev Low Pass Filter topology is shown in Figure 21.

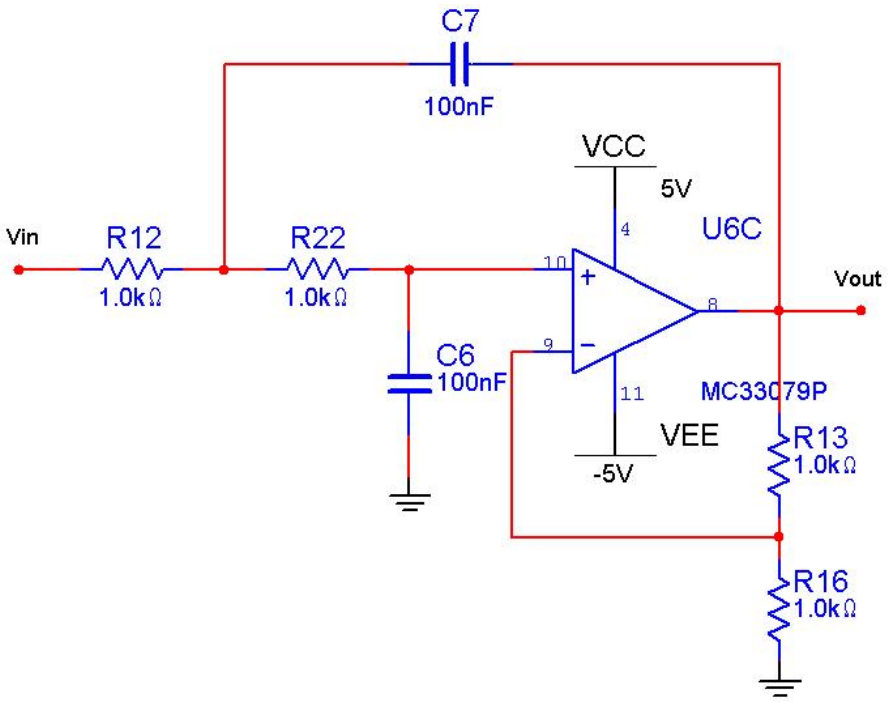


Figure 21: Chebyshev Low Pass Filter

The design equation for this filter is:

$$RC = \frac{1}{2\pi f_c f_n} \tag{32}$$

where f_c = 1700 Hz and $f_n = 0.9707$. In Figure 21, R = R12 = R22 = R16 and C = C6 = C7. The value of 0.1 μF was chosen for C. The cutoff frequency of 1700 Hz was determined experimentally to provide the most narrow peak pulse shape. The closest practical R value for this design is 1 kΩ. With these component values, the actual cutoff frequency of the filter is 1180 Hz. R13 in the negative feedback loop of the op-amp determine the gain of the filter.

Low Pass Filter-Integrator

The LPF-Integrator stage consists of four resistors, a capacitor, and an op-amp. The Integrator's function is to create a signal with a sharp peak at the end of the received signal pulse. The

pulse varies in amplitude as the transmitter-receiver distance changes. This poses a problem on edge triggered methods of signal detection without an automatic gain circuit. Since the transmitter is sending a pulse of energy, this design attempts to generate the highest signal voltage at the final edge of the pulse. The voltage of the signal at 40 kHz is integrated over time. This results in an increase in output voltage while the input voltage is non-zero. The integrator schematic may be found in Figure 22.

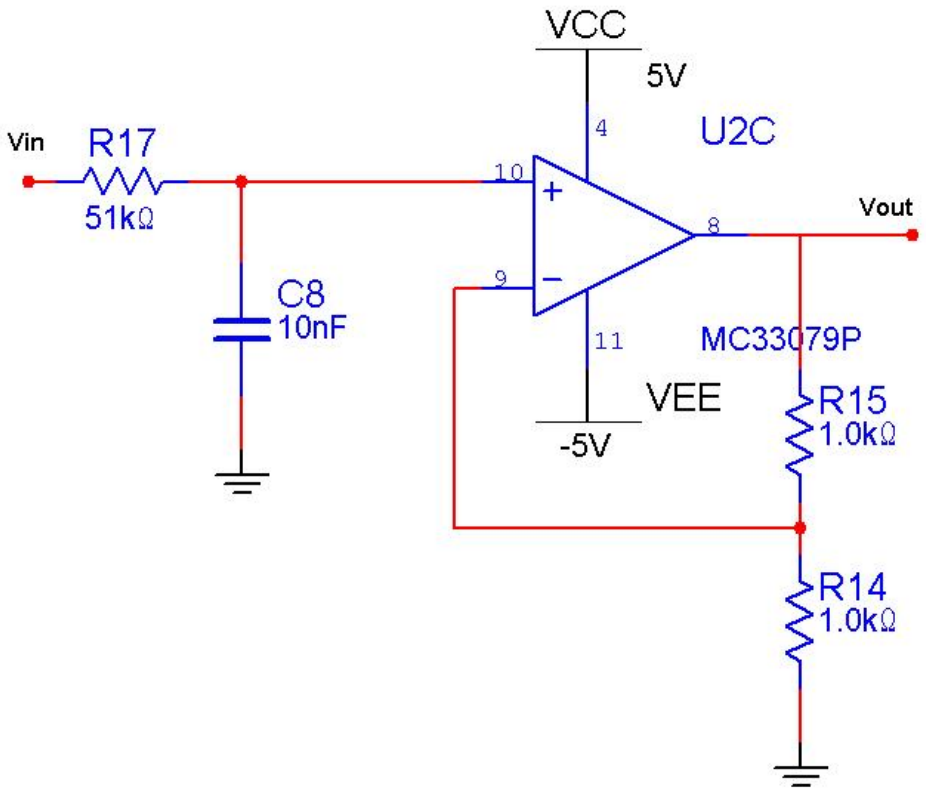


Figure 22: LPF/Integrator

It was found experimentally on the circuit in the lab that, although the design shown in Figure 22 behaves like an integrator in the 40 kHz range, it retains some 40 kHz ripple on the output. The performance is not ideal enough for the sampling block to resolve an accurate peak. The pulse shape is also not ideally square, and ideal integration results in a rounded peak shape. A requirement of the pulse peak shape is that it needs to be as sharp and consistent as possible,

so the LPF-Integrator design was revised into the Chebyshev Low-Pass Filter.

4.3.3 PCB Layout

The analog signal processing section of the receiver has completed its final stages of design and its PCB layout is shown in Appendix C.5. The layout utilized entirely surface mount components. The resistors, capacitors, and diodes each had a standard footprint of 0805. The four Op Amp ICs have SOIC-14 footprints.

Each of the op amp IC power rails are bypassed using 0.1 μF surface mount capacitors to minimize power supply noise. The green footprints to the left of Figure 61 are external connectors that will be wired to the receiver transducers on the receiver array. The green footprints at the bottom right in the figure are jumpers on the signal traces to the next system block. This is the sampling block, and includes the analog multiplexer and ADC.

4.4 Signal Processing

The purpose of the processing subsystem is to digitize the data coming from the five receivers, perform peak detection, determine time difference between the different receivers and calculate the location of the transmitter. These operations are performed on the dsPIC chip that is being used for the processing, along with some supporting hardware.

4.4.1 Hardware Selection

The hardware required to perform the processing operations is comprised of three major blocks: multiplexing, sampling and processing. Each of these hardware blocks corresponds to a specific chip. Hardware for the processing block was selected to fit the needs for processing capabilities and accuracy.

The first decision that had to be made was the processor necessary to perform the TDOA calculations, as this determines what additional hardware is required. The options available for processing include regular microcontrollers, microcontrollers with DSP instruction sets and DSP chips. Based on the amount of DSP operations used in the TDOA calculations, we determined

that we would be able to take advantage of a DSP instruction set, which ruled out regular microcontrollers as an option.

The selection of a DSP versus a microcontroller with DSP instructions is a trade-off between processing power and ease of implementation. DSP chips are capable of high speeds and efficient processing, but typically require significant supporting hardware and are generally not available in easy to use packages. DSP Microcontrollers such as Microchip's dsPIC series are designed to provide additional DSP capabilities while still maintaining the ease of use associated with microcontrollers. A DSP microcontroller is clearly easier to implement and is a good choice if it can fulfill the necessary processing requirements. To determine what type of processor we needed, we calculated the number of instructions that would be required to implement the TDOA solver. Based on this number of instructions, we were able to determine the time required for these calculations on a 30 MIPS dsPIC, which was significantly shorter than the available processing time. These calculations assured us that a dsPIC would provide enough processing power for this application.

Selection of the specific dsPIC model was based primarily on our requirements for I/O pins and packaging. The dsPIC33F family of chips were eliminated as options because the extra processing power is not necessary and they are not available in through-hole packages, which would make initial prototyping and testing difficult. This left the dsPIC30F family, which includes a range of pin counts and features. We required a minimum of 12 I/O pins for ADC input, 3 pins to set the multiplexer select lines and 1 to provide a debugging LED, for a total of 16 required I/O pins. This ruled out any 14-pin packages and made a 28-pin package the clear choice, as they provide enough I/O pins without being excessively large. The dsPIC30F3013 was chosen because it provides the most RAM and ROM of the dsPIC chips and is available in 28-pin DIP and SOIC packages.

To achieve a maximum speed of 30 MIPS for the dsPIC, it must have an internal clock of 120 MHz. The dsPIC clock is provided using a 7.37 MHz oscillator with a 16x PLL multiplier which steps up the clock speed to 120 MHz.

The ADC and multiplexer were selected based on their accuracy and speed. The most effective way to run the ADC is using the same clock that is provided to the microcontroller. This method does not require an additional clock source and ensures that the ADC will be synchronized with the dsPIC instructions. Since the dsPIC clock runs at 7.37 MHz, the ADC must be capable of running at least this fast. Based on the speed requirement and the desire for 12 bits of accuracy, the Analog Devices AD9220 was selected. This chip is capable of providing 12-bit analog-to-digital conversion at a rate of up to 10 MHz.

By running the ADC at the processor clock speed of 7.3728 MHz and multiplexing the input to it with five different channels, each receiver will be sampled at about 1.5 MSPS. The output of this converter is a set of 12 parallel lines representing the 12-bit digitized signal.

The multiplexing of the five channels is performed by an Analog Devices ADG608 analog multiplexer. This device is capable of switching fast enough to change channels once per ADC cycle. The select lines for the multiplexer are controlled by the dsPIC, permitting precise control of the switching.

4.4.2 Sampling Coding

The software end of sampling consists of four blocks: multiplexer control, reading and storing samples from the ADC, peak detection, and determining delays for output to the processing block. These operations are all performed in the dsPIC chip and are implemented as functions within the sampling and processing code.

Controlling the multiplexer involves controlling both the multiplexer chip and the separation of receiver data within the processor. To change between the five channels, the multiplexer needs five different 3-bit values placed on the select lines.

The first scheme for creating the select lines was to use a counter. This method was chosen for two reasons. The first reason for using the counter was because it did not require the code on the processor be synchronized to the clock. This is because the ADC and the counter will be running synchronously because they are drawing from the same clock. Since the processor

is running at four times the speed of the clock the read instruction for sampling the ADC can happen at any time during the clock cycle. This does not make a difference because the ADC is valid during the entire clock cycle. The second reason for using the external counter was for sampling speed. If the processor is not taking extra cycles to set the select lines, the sampling will be faster.

In testing we found that all the high frequency lines were radiating noise into other parts of the system and the counter was especially problematic. The design could be built without the counter with slightly more processor involvement. The problems cause by the counter far outweighed the benefits. The multiplexer was reconfigured to manually pass selected values through to the input pins of the dsPIC. Through careful timing of instructions it is ensured that the receiver being read is always known. It is important to ensure that information about which receiver is currently being processed is appropriately passed to all of the software blocks within the sampling and processing system. With the counter removed, the processor must be synchronized with the clock because the processor will need to issue instructions based on the clock edge. The goal is to output the address to the multiplexer three instructions before the rising edge of the clock. This ensures that the multiplexer is at its most stable level when the ADC reads the voltage triggered by the clock edge.

Reading samples from the ADC involves a function to read the data off of the 12 digital input lines. This data needs to be stored in memory for the peak detection algorithm. To increase sampling speed, only the most significant 10 bits of the ADC are in use, because the dsPIC can only read in 10 bits on a single port. Reading all 12 bits would require reading two different ports, then combining the bits into a single value and would significantly increase the time required to read a sample. The reasons the 12-bit ADC is still being used are to provide the option for higher accuracy and to avoid the risk of changing to a new chip with limited testing time.

Determining time delays involves starting a counter when the first receiver passes a set threshold. The peak detection block looks at the currently received sample and compares it to the previous maximum sample for that channel. If this is a new maximum, then it is stored in mem-

ory, along with the counter value for the time at which this value was received. On the rising edge, a new maximum is reached on most samples, however once the peak is reached, the highest value has been obtained and no new maximums will be reached. After 2 ms from when the first channel's threshold has been exceeded, all peaks will have been received and the time differences can then be calculated.

The method to sample the signal needed to be changed multiple times for the system to become functional. The first path taken involved the counter, so after initializations and once the sampling function was called, the processor would reset the counter and wait until channel 1 was selected. At which point sampling would begin synchronously. This was changed due to noise generated by the counter.

The next version involves manually controlling the select lines of the multiplexer with the microcontroller. The multiplexer value is changed once every four instruction cycles to provide the ADC with a new value every clock cycle. To account for the ADC pipeline delay of 3 ADC cycles, the first multiplexer changes are issued a set number of cycles before the port is read and changed every 4 cycles after the first control instruction. Once it has waited, the processor then reads and stores port B every four instructions for all five select lines. After all five channels have been stored the processor starts issuing multiplexer control commands while it runs the peak detection algorithm.

The peak detection algorithm also went through several revisions. The first revision valued simplicity and time efficiency rather than eloquence and error detection. This version was implemented when the counter was still in the circuit, and was short enough to run in-line with the sampling code while the microcontroller was waiting for the counter. Every pass through the sampling algorithm the amplified and integrated voltage of each receiver was compared to the previous sample, and if the former was greater than the latter, that value and time would be stored in the dsPIC's memory. While this method would work for a clean signal, the amount of noise on the PCB caused many false peaks, and thus the method was deemed inaccurate.

4.4.3 Final implementation

To replace this scheme, a curve fitting algorithm is used. To use a curve fitting algorithm many samples of the incoming signal needed to be sampled and stored. This is required because the sampled signal needs to be compared to the stored signal in this method. The first revision in this new approach was the sampling. The previous sampling revisions were based on the simple sample and compare method with no storage of the sampled value. The samples were not stored due to limited memory space to work with. This forced us to move from a low intelligence sampling method to one with the ability to acquire only the values on the rising edge of the curve. The current flow chart of the code can be seen below in Figure 23.

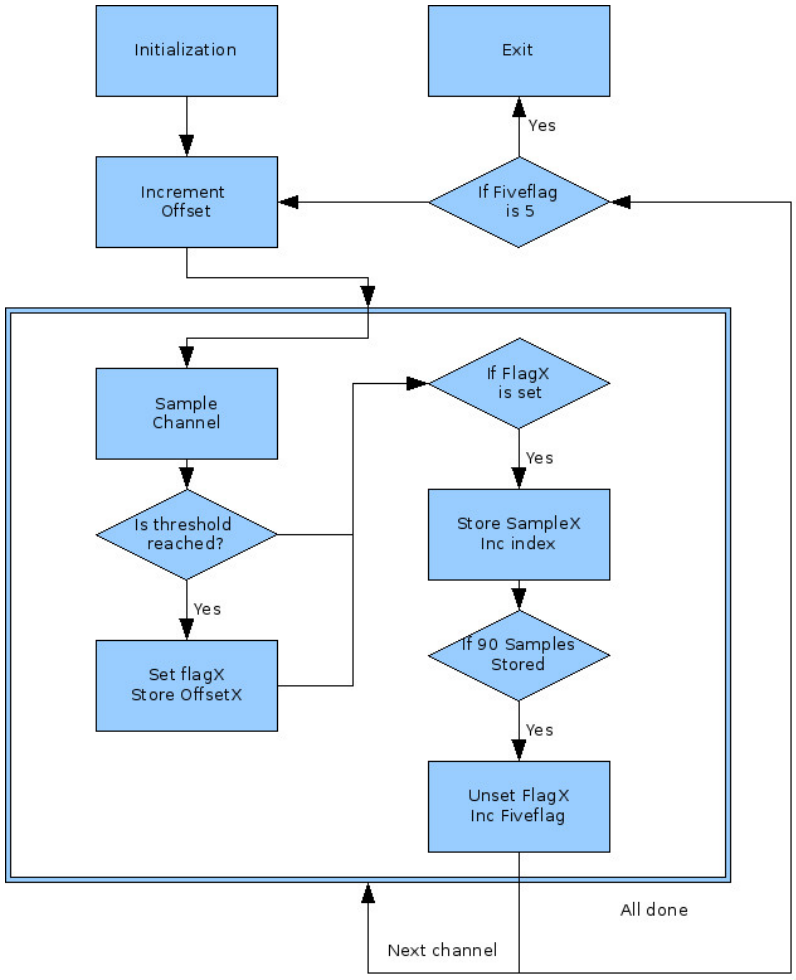


Figure 23: Sampling flow

The flow chart shows the code starting by incrementing a variable called offset. The offset variable will act as the counter in this revision. Every time each of the channels is sampled it will increment this value which will happen every 6.1035 μs. This code runs the same 3 routines on all five receivers. The first part of the routine looks for a threshold which when exceeded will set the flag for the channel and store the offset value. This flag controls whether the channel will be storing the sampled values or not. The stored offset for that channel will be the sampling initial time. The routine then checks if the flag is on and will store the value it sampled and increment its index so the next sample will go to the next array position. If the channel has sampled 90 times it will increment another flag signaling that it has finished its sampling. Once all five channels have gone through this routine it checks to see if all the channels have received all 90 values. If so it will exit the sampling function. The result of this code is 90 samples each 6.1035 μs apart from each channel starting at the threshold. Figure 24 shows an example of a sampled curve.

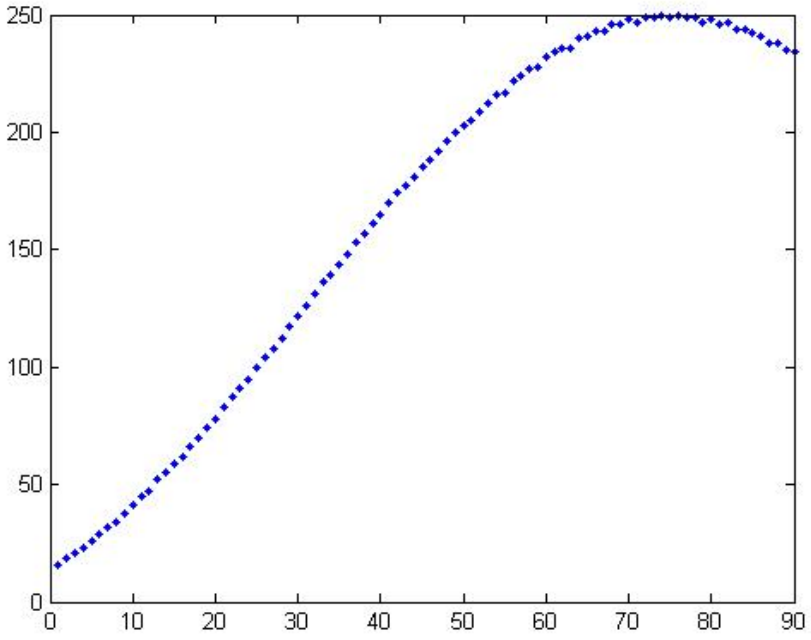


Figure 24: Sampling output

Using our sampling dsPIC and our sampling scheme we outputted with a serial port over a hundred pulse shapes for each channel. Using MATLAB to average the stored pulses a characteristic version of our pulse shape was created to be used for. The premise behind curve fitting is that if difference difference between stored and sampled was summed at a certain shift and graphed versus the time shift there would be a minimum value defining were the two curves match up. So one approach to determining the time at which the curves fit would involve shifting and comparing the errors until a minimum is reached. This is ineffective for our scenario because each error calculation takes 3 ms in the current configuration. So a different approach is required to minimize error.

The idea of our algorithm was to intelligently shift the signal until it reached low error instead of shifting by a known amount until we reached the lowest error. To do this we have the error itself determine the shift so the higher the error the more it move to correct itself. However to do this the error must approach zero which is not possible when the amplitude of the signal is different. So the first step in this implementation is to determine the max value in the sampled curve and gain the sampled signal based on the difference between the stored and sampled max value. With the gain signal the closer the signal get to fitting the smaller the error which means smaller shift steps until the error reaches zero or crosses to the other side and becomes negative. Fortunately the shift is not going to vary by much from pulse to pulse so we can store the last pulse for the next shift calculation.

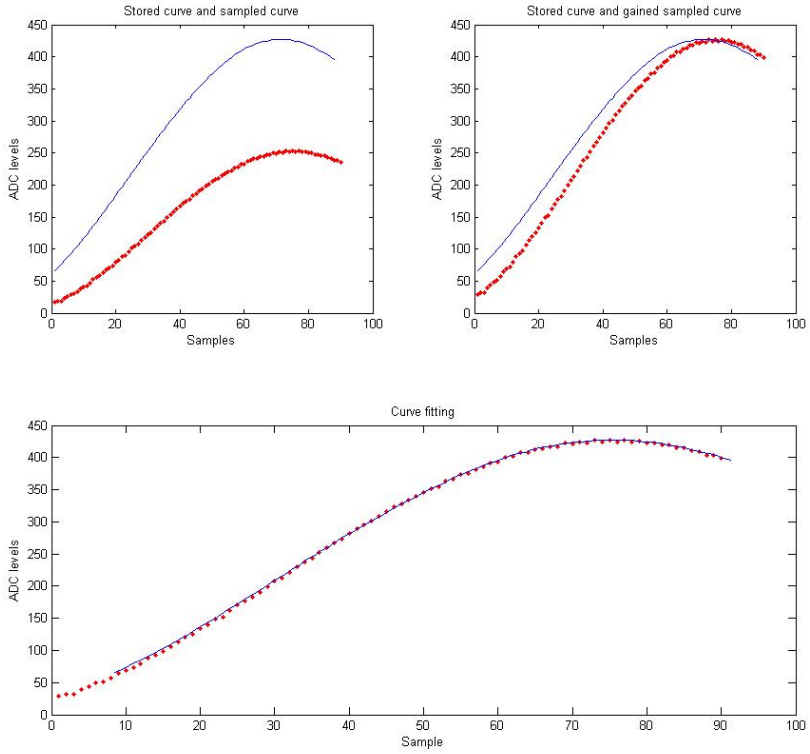


Figure 25: Curve fitting gain and shift

At this point we have a time value for the start of sample and the difference from that time to the peak. To get the absolute peak time we subtract the shift from the offset value. This give the time difference from when the sampling started to when the peak is reached. A single receiver is considered the center point and the peak time for this receiver is subtracted from all of the other values. This array of time differences will be passed to the processing block, which performs time difference of arrival calculations on the data to determine 3D position.

4.4.4 Simulation

The location of the receivers and the rate of sampling will significantly affect the accuracy of the calculations. Because these factors influence hardware design, it would be difficult to determine the true effect with physical testing. Because of this, a MATLAB-based simulator was developed to determine the requirements for sampling rate and receiver location.

The full simulation code is found in Appendix B.8. The simulator uses code to simulate the other blocks in our system which allows testing of the TDOA calculation code under non-ideal conditions. The MATLAB function is given a set of receiver coordinates, the number of times to repeat the procedure, and the sampling rate. It then outputs a matrix of all the errors and the locations corresponding to the error.

```
function[matrix_array,cord_set] = proccessingblock_rev4(rec_dist,Res,rate)
```

The first section of the simulator describes the behavior of the transmitter. It creates a random location within our desired operating block which signifies the location of the transmitter. An arbitrary location is created so that the same starting points are not used every simulation and thus avoiding missing values between our stepping amount. This also allows a small sample to achieve a full representation of the system.

```
Rx = (rand - 0.5) * 0.2;
Ry = (rand - 0.5) * 0.2 + 0.5;
Rz = (rand - 0.5) * 0.2 + 0.1;
R = [Rx Ry Rz];
```

Next is the transmission simulation. To simulate the transmission of our signal, the distance between the created coordinate and every receiver is calculated. Each distance is then divided by the speed of sound which produces the transmission time to each of the receivers.

```
P = repmat(R,5,1);
d = sqrt(sum((P - rec_dist).^2,2));
T = d / 340;
```

The next section simulates the sampling process. First, each of the delays is calculated in relation to the first receiver. This section also simulates the sampling rate by truncating the delays at the next multiple of the input rate. This effectively simulates the sampling because after the signal is received and the value goes high, the processor will not read it until the next time it samples.

```
delta_T(1) = T(1) - T(1);
delta_T(2) = T(1) - T(2);
delta_T(3) = T(1) - T(3);
```

```
delta_T(4) = T(1) - T(4);
delta_T(5) = T(1) - T(5);
T_col = ceil(delta_T / rate);
T_col = T_col * rate;
```

This value is then fed into the `bard_solver` to calculate the location of the transmitter. Once this process has finished, the error in the calculated location is found by using the difference between the generated location and the calculated location. This provides for graphing the error and seeing the accuracy trend.

```
[est,est2] = Bard_solver(rec_dist_OOPS,T_col,340);
error = sqrt(sum((R-est2').^2));
error2 = sqrt(sum((R-est').^2));
x(n) = min(error,error2);
```

Returning to the source of the `bard_solver` code it was found that the solver outputs two solutions. Running MATLAB in debugging mode revealed that one solution is correct while the other coordinates are based on another possible solution outside the operational area. To correct this in the simulation, the value with the minimum error is taken as the correct solution.

The first source of variation introduced in simulations is the input sampling rate. To test this, a static distribution of receivers is set in the simulation, which is then run using variations of the sampling rate. The following set of coordinates is used as the distribution of receivers in the prototype receiver array.

Position	X (m)	Y (m)	Z(m)
A	0	0	0
B	0	0.2	-0.2
C	0.1	0.1	-0.1
D	0.2	0	-0.2
E	0.2	0.2	0

Table 3: Distribution of Receivers

Using this distribution, the sampling rate is varied, starting at the desired 62 μs and decrementing until the desired error of one millimeter is calculated, as shown in Figure 26.

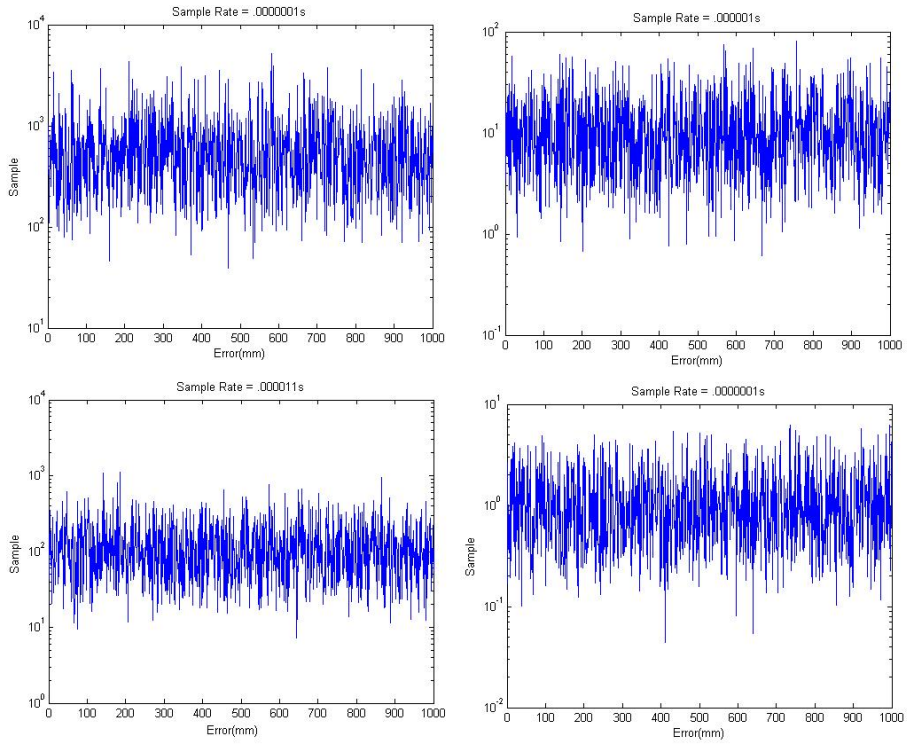


Figure 26: Sampling Rate Test

A sampling rate of 1 μs is needed to reach the desired accuracy, but assuming optimal distribution, the desired accuracy could have been obtained from 11 μs of error. This means that there are better distributions of receivers possible. To determine the optimal receiver distribution, the least accurate axis needs to be calculated. For this reason, the error calculation was modified to look at each axis individually. The distributions in this section were determined by the design goals. The receivers will be on their own fixture so that standard distances will be known, and this simulation confines them to a ten centimeter cube. The most representative of our results is shown by figure 27. The top left graph shows a single plane distribution. This distribution has an average error greater than 10 cm and for the axis perpendicular to the plane the error is over a meter. The best distribution for this system is to have a square of receivers and a receiver in the middle in terms of x and z axes, then distribute the receivers on the y-axis so they are on three levels.

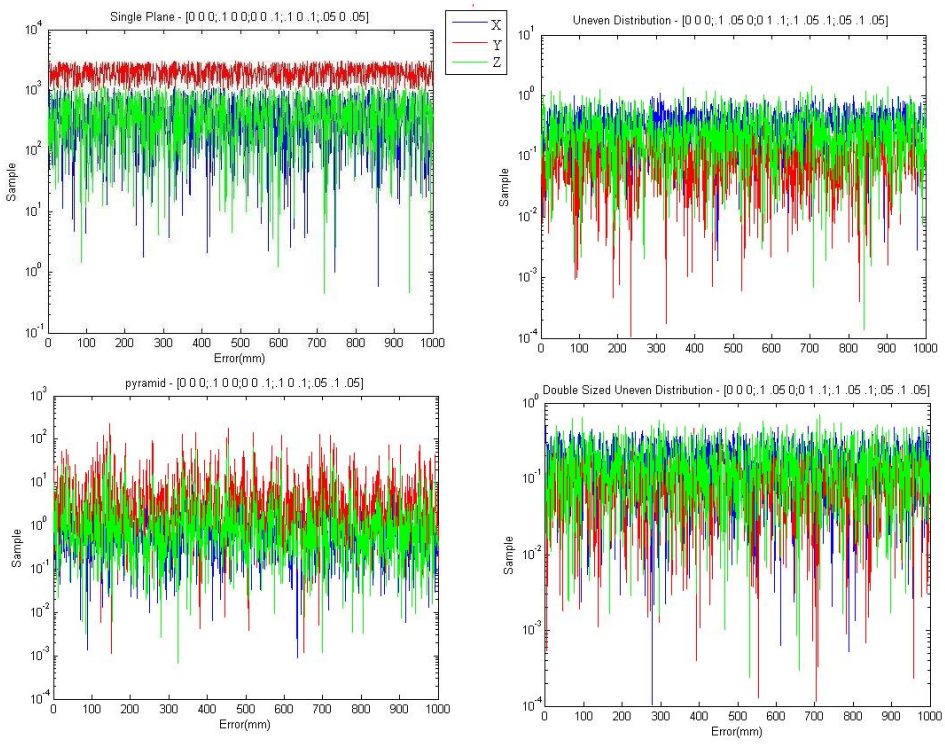


Figure 27: Distribution Test

Another non-trivial source of error comes from uncertainty in the location of the receiver. When measuring the distance between the receivers it is expected to have some error. Even if care is taken in measuring the distances, an error of half a millimeter could still be possible. Knowing the large amount of error that will be caused by a variation in the location of the receiver and the value used for calculations, a calibration method must be used to rectify this.

4.4.5 TDOA Calculation Coding

The MATLAB version of the `bard_solver` can not be run directly on the dsPIC. The microcontroller can only run code written in assembly or C. Therefore, the MATLAB code needed to be converted to one of these forms. The use of C allows for multidimensional arrays which helps recreate the matrices that MATLAB uses, making it the obvious choice for coding. The C code version of the `bard_solver` is in Appendix B.6.

Variables `pinvA` and `phi` are calculated each time the MATLAB version of the `bard_solver` runs, however the two variables are independent of the time delays and are only based on the locations of the receivers. This allows us to calculate the values once in MATLAB for our configuration and enter the variables as constants. The calculation of these values is one of the largest portions of the MATLAB version due to the calculation of the pseudo-inverse of the receiver matrix. The removal of this code saves considerable processing time.

The rest of the `bard_solver` is simple matrix math. However, when running in C the matrix math must be done manually, meaning each element in the array needs to be calculated separately. This is a risky way to code because there are many possible sources for error which greatly increases time required to debug the code. However this approach gives more control in what is done with the processor allowing for decreased looping and redundancy in calculations. This code needs to take as little time to run as possible so the sampling is ready for the next pulse so ensuring that this runs efficiently is important.

Once the C version of the code worked on a PC it was not difficult to implement onto the dsPIC. Porting the code to the dsPIC was relatively easy because it mimics already functional code. When an error occurs, each stage of the code can be compared to the equivalent code for MATLAB response. When programming on the dsPIC the only barrier was an issue with the addressing, where certain variables were getting unexpectedly cleared. Better declarations were used to solve this problem. To test the functionality of this block, delays with known solutions were manually entered. Once the output of the MATLAB version matched the code on the processor, the code was considered functional.

4.5 Hardware Interface

When determining the system design characteristics, the largest factor in choosing a microcontroller is the availability of USB support. The microcontroller must be able to communicate easily through USB to the computer and to the processing microcontroller through UART. This processor's sole purpose is to communicate data to the PC while the dsPIC is processing new

data.

Most sources on the Internet that communicate with the PC over USB utilize the PIC18Fxx5x series devices. This family of devices was developed with a USB2.0 interface to replace the PIC16C745 microcontroller which supported a USB1.1 interface. Due to the many designs and user support of this family of devices, it will be used to develop the prototype. The four PIC18Fxx5x devices and their characteristics are shown in Table 4. The PIC18F2455 was chosen for the design due to its smaller form factor and lack of the need for more I/O lines.

Device	Flash	Instructions	I/O	A/D
PIC18F2455	24K	12288	24	10
PIC18F2550	32K	16384	24	10
PIC18F4455	24K	12288	35	13
PIC18F4550	32K	16384	35	13

Table 4: PIC18Fx5xx family differences [7]

The USB specification document is several hundred pages long and goes into extreme detail on USB communication between peripherals and the PC. Implementing a USB stack for this device is a chore that goes beyond the scope of this project. There exist two C compilers that are available for the PIC microcontrollers which allow a much simpler implementation of USB connection. Both compilers supply example code which implements a HID USB mouse that rotates in a circle on the PC. Although elementary, this code will serve as testing software for the microcontroller circuit which will confirm that a USB device can properly attach and communicate with the PC. Once a working example is obtained, the code will then be used as a building block to design the customized code for the project.

The Microchip C18 Compiler specifically supports the PIC18Fxx5x series of chips and the USB extensions. The compiler is offered on Microchip's website [6] as a student demonstration version. In accordance with the C18 suite, Microchip has developed a USB Bootloader for the PIC18Fxx5x series of microcontrollers. The software itself specifically says it supports the PIC18F4550 chip, but with some modification of the source code, it can be compiled to support the rest of the family. The USB Bootloader takes advantage of the USB capability of the device to allow serial

programming of the FLASH memory over USB. The USB bootloader source is downloadable from Microchip's website but must be compiled using the C18 compiler. Once compiled, this software is loaded using the PIC burner as with any other PIC device. The device will enter programming mode if the PRG button is active while the device is rebooting. Once in programming mode, the device communicates with the PC using a Microchip supplied driver. The device can then be programmed using the Microchip USB Programming application. Utilizing this application will allow for rapid prototyping of code at most locations as opposed to needing to unmount the chip and use the PICSTART programmer. The implementation circuit must then be developed around the hardware. This includes the button configuration, clock frequency, and bypass capacitors to filter the input voltages.

The PIC18F2455 can support a variety of clock speeds and input formats. The most efficient clock implementation for the PIC is a crystal resonator which is excited by the device's clock pins. The clock is also controlled using two decoupling capacitors for which Microchip has suggested different values according to the desired frequency. This input is then interpreted through an internal Schmitt trigger which quantizes the oscillation. The simplest of clock formats is the use of an external clock. An oscillator IC was used to generate a square wave which is interpreted by the microcontroller as an external clock. The PIC then makes no attempt to condition or excite the clock pins, but uses the given input as the system clock. According to various resources on the Internet, the most widely used input frequency is 20MHz. All examples provided by PIC C compilers incorporate a 20MHz clock. The system clock goes through three stages in which the frequency is manipulated. Depending on the USB speed, the clock frequency is set appropriately. The diagram in figure 4.5 below shows the clock divisions.

After meticulously deciding upon a clock configuration to use, the remainder of the circuit configuration must be decided upon. The test circuit design shown in Appendix C.4 was developed from the PIC18Fxx5x datasheet [7], the Microchip C18 compiler example code, and the schematics and suggestions of several hobbyists on the Internet. This circuit was constructed on a standard breadboard using a DIP (Dual In-line Package) PIC18F2455. A butchered USB cable

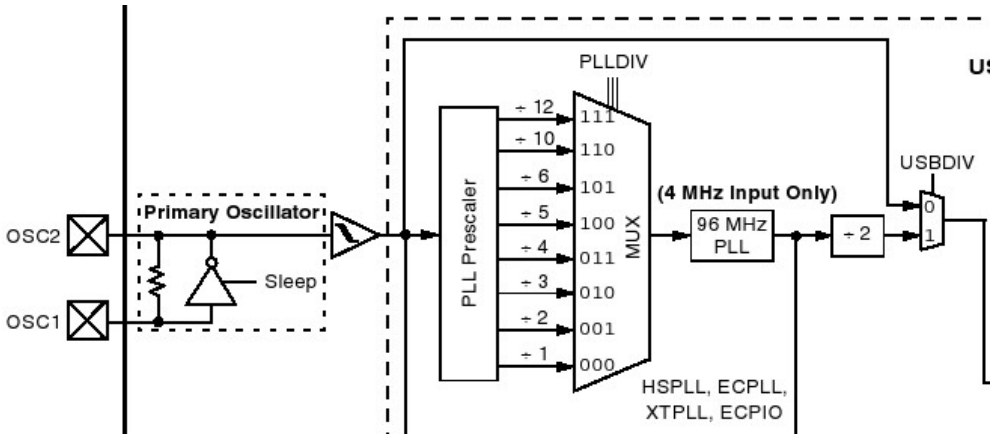


Figure 28: Clock Division for PIC18F2550 [7]

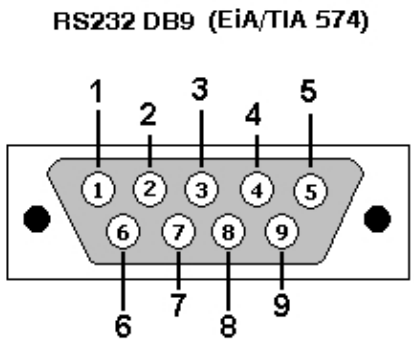
was also tied to the appropriate lines on the circuit. The thought arose that perhaps problems with the enumeration of the USB device could be due to the inefficiencies that lie in the breadboarded assembly and the data lines of the USB cable which run at 48Mhz. Fortunately there exists a similar circuit to the breadboarded device called the USB Bit Whacker [13]. This device is manufactured by SparkFun.com in a PCB layout with a PIC18F2455 chip and the supporting hardware for the Microchip USB Bootloader code. Using this device in a small package would provide portability to the device as well as enable programmability via USB. The slight disadvantage of using these boards is the lack of access to the PIC chip itself. If the USB bootloader somehow gets overwritten with a segment of code, the bootloader would have to be replaced using in-circuit serial programming (ICSP). These devices were purchased and were used as the primary development boards during the early stages of design for the PC interface side of the project.

4.5.1 Hardware Debugging

The PIC microcontroller used in our design is equipped with a USART port which allows for serial communication on Tx/Rx lines located on pins 17 and 18 respectively. These lines can attach to a PC's serial port and aid in debugging the device at various lines in the code. The Microchip C18 compiler supplies the option to establish RS-232 communication over the USART. The baud rate for the connection can be specified through preprocessor directives. Once the RS-

Pin No.	Name
1	Data Carrier Detect
2	Receive Data
3	Transmit Data
4	Data Terminal Ready
5	Signal Ground
6	Data Set ready
7	Request to Send
8	Clear to Send
9	Ring Indicator

(a) RS-232 Pinout



(b) DB-9 connector [2]

Figure 29: RS-232 Connection

232 connection to the PC is defined, output can be written using `fprintf()`. The physical connection between the PC and the microcontroller can be established by incorporating a logic inverter to the Tx pin of the device; the output of which can then be sent to the serial port of the PC. The only pins necessary for PC communication are the Transmit Data, Receive Data, and Signal Ground lines. Figure 4.5.1 shows the pinout for RS-232 communication with the PC's serial port. The other pins for the connector are only used for fast and reliable communications with flow control. The communication used for debugging will be one-way data sent to the PC from the PIC.

Once the device's firmware attempts to initialize USB communication, debugging can take place between the computer and the device. Once attached to the PC and powered up, the USB hub will sense the impedance of the PIC denoting the operation speed of the USB. If debugging messages are enabled in the kernel's USB modules, the system logs will report all USB activity. Using this feature, higher level communication between the device and the PC can be monitored to ensure communication is working properly.

4.5.2 Software Design

Although the PIC18FXX5X datasheet thoroughly describes the availability of configuration registers and implementation of processor instructions in assembly, it does not delve into writing and compiling C code for the device. The C compiler manuals also are not as in-depth as desired, however the respective companies provide example code which is generically written to perform simple tasks. Microchip C18 compiler provides a simple USB Virtual COM port, a simple Mass Storage Device, as well as a HID mouse which rotates in a circle. These examples are also used for hobbyist projects found on the Internet. It seems as though the best method of implementing the device is to branch from the provided example.

All USB examples contain the same wrapper code. The Device powers on and goes through initialization for USB communication with the host. The device is considered connected by the PC and the PIC begins in a small loop. The first instruction `USBTasks()` services the hardware by polling the host computer for data. If data is found, it diverges into the driver servicing subroutine. Following this procedure, the next called is `ProcessIO()`. This subroutine performs whatever function the device is configured to execute. This subroutine is located in `user/user_mouse.c`. For the mouse example, the code enters a counting loop that cycles the x and y coordinates so the mouse moves in a circle. This was a good start for the device implementation. The major problem implementing this example code was configuring the PIC circuit as well as ensuring the example code, which came significantly undocumented, matched the circuit. Some modifications needed to be made to the code in order for it to be detected and operate by the PC. The major modification which required hours of debugging was commenting out the `USE_USB_BUS_SENSE_IO` directive. If the device is using bus sense, additional circuitry must be wired onto the I/O pins. If the chip is told to use Bus Sense and that circuitry is not connected, the device will enumerate and immediately shutdown and disconnect.

It is a requirement that the `USBTasks()` subroutine be executed routinely in order to send the the USB page-data to the PC. If the code manages to hang at a certain point and the subroutine is not executed, the device will be rejected by the PC and fail. Therefore it is important that

when the USB PIC is listening on the USART lines from the dsPIC, it does not block unless data is on the bus. A quick check to see if the USART line is a logical 0 (idle state for USART is logical 1), is done prior to reading data from the port. The dsPIC will serially transmit three single precision floating point numbers (`float`) representing the x, y, and z coordinates of the mouse. This value will be immediately processed.

The HID specification requires positional data for the mouse type to be relative. Considering the dsPIC is quite busy with sampling and calculations, the calculation of relative position as well as bounding box calculations and cursor sensitivity will take place on the USB PIC. The input coordinates into the USB PIC will first be checked against the bounding box of operation for the user; This will expel erroneous data. If the data is considered invalid, the pointer will not move. When the coordinates are considered valid, the last known coordinates are subtracted from the recent coordinates and a multiplier is applied.

At this point, mouse gestures could be easily added to the code. For a proof of concept, we have decided to include a clicking gesture. If the code detects a high acceleration in the depth direction, it will emulate a mouse click. This will allow the user to "poke" toward the screen, causing the hardware to register a mouse click. Ideally, the Z-axis would be mapped to the scroll wheel which in most 3D applications operates the depth dimension.

When the data has been tuned and is ready to be sent to the PC, the `HIDTxReport()` is called which takes in the data buffer as an argument. This sends the data payload to the internal USB handling functions predefined by Microchip, Inc. The buffer which is sent to the PC is shown below in Table 5.

Data	Range	Type
X-Axis	-127 to 128	Signed Byte
Y-Axis	-127 to 128	Signed Byte
Z-Axis	-127 to 128	Signed Byte
Mouse BTN 1	0,1	Bit

Table 5: Data payload to `USBTasks()`

5 Testing and Results

The purpose of this section is to analyze our system, quantify our results and evaluate the operation of the device. Testing results for each block of the system are described and compared to original goals. The entire system is then examined in terms of its performance and compliance to desired specifications.

5.1 Testing Plan

The following is the detailed testing plan used to verify the functionality and performance of the system. It steps through the various components of the system testing them one at a time and then testing for system integration each time a new functionality benchmark is reached.

A. Transmitter

i. Connect power to transmitter.
ii. Use an oscilloscope to probe the leads of the transmit transducer on the transmitter to view the waveform being provided to the transducer.
    1. Pass: Square pulses of a 40 kHz wave can be seen on the oscilloscope.
    2. Fail: Else. Check that the MSP430 is being powered and is properly programmed.
iii. Connect a receive transducer to an oscilloscope and place the transducer within 40 cm of the transmitter and less than 45° from the center axis of the transmitter.
    1. Pass: Consistent curved pulses of a 40 kHz wave can be seen on the oscilloscope.
    2. Fail: Else.

B. AC Amplifier Test

i. Connect power to the receiver circuit board.
ii. Use a function generator to provide a test input of: $10^{-2} * \sin(2\pi 40000t)$. Apply this signal to the AC amplifier input of each receiver channel.
    1. Pass: Signal is amplified by the expected gain factor of 300 $\pm$10%. The expected signal is either a clipped or unattenuated sinusoid with amplitude proportional to the test input when measured with an oscilloscope.
    2. Fail: Else. Check the power connections to the receiver circuit board, and ensure that the components are the correct values.
iii. Use a bench power supply to provide a test input of 1 V DC. Apply this signal to the AC amplifier input of each receiver channel.
    1. Pass: 1 V DC is observed on the output of each channel with an oscilloscope.
    2. Fail: Else. Check the power connections to the receiver board, and ensure that the components are the correct values.

C. Precision Rectifier

i. Connect power to the receiver circuit board. Ground the input to the AC amplifier block for all channels.

ii. Use a function generator to provide a test input of: $1 * \sin(40000 * 2\pi * t)$. Apply this signal to the precision rectifier block input of each receiver channel.
    1. Pass: Signal is a rectified version of the test input when observed with an oscilloscope. Some negative component may be observed, but this should not exceed -0.5 V.
    2. Fail: Else. Check that all diodes are oriented correctly, and that the receiver board is powered.

iii. Ground the precision rectifier inputs for all channels.
    1. Pass: The measured output magnitude is less than 50 mV when observed with an oscilloscope.
    2. Fail: Else. Check the power connections to the receiver board, and ensure that all diodes are oriented correctly.

D. Chebyshev Low Pass Filter

i. Connect power to the receiver circuit board.

ii. Ground the Chebyshev Low Pass Filter inputs for all channels.
    1. Pass: Output voltage is less than 350mv when observed on an oscilloscope.
    2. Fail: Else. Check the power connections to the receiver board, and ensure that components are the correct values.

iii. Connect power to the receiver circuit board. Ground the input to the AC amplifier block for all channels.

iv. Use a function generator to provide the test input: $\sin(40000 \times 2\pi \times t) + 1$
    1. Pass: Output voltage of each channel is greater than 1 V and contains less than 10 mV of 40 kHz ripple when measured with an oscilloscope. Each channel filter also passes part A.
    2. Fail: Else. Check the power connections to the receiver board, and ensure that components are the correct values.

v. Determine the 3 dB cutoff for the filter by sweeping the input frequency.
    1. Pass: Filter cutoff is within 10% of the desired cutoff frequency.
    2. Fail: Else. Check the power connections to the receiver board, and ensure that components are the correct values.

E. Transmitter to Receiver Integration Testing Integration Testing

i. Connect a single transducer to all AC amplifier inputs of the Analog Front End Block.

ii. Activate the transmitter to generate a transmit pulse.

iii. Place transmitter 20 cm away from the receiver transducer.

iv. Connect power to the receiver circuit board.

v. Test each block for the expected signal.
    1. AC Amplifier Expected Output: Pulsed sinusoidal input amplified by gain when measured with an oscilloscope.
    2. Precision Rectifier Output: Appears to be a rectified version of pulsed sinusoid when measured with an oscilloscope.
    3. Low Pass Filter Output: Slightly distorted pulse shape, less than 5 mV of 40 kHz ripple when measured with an oscilloscope.

vi. Measure delays of each channel with an oscilloscope. Trigger the oscilloscope on the pulse generated on the transmit transducer. Use the other channel of the scope to measure the output peak of the Chebyshev Low Pass Filter block. Use this data for calibration.

F. dsPIC Basic Operation

i. Probe the dsPIC oscillator's clock output on an oscilloscope. Use the shortest ground leads possible to avoid introducing noise to the measurements.
   1. Pass: The clock has no more than 12% of the peak voltage in switching noise and ripple noise during the high or low state is not larger than 5% of the peak voltage.
   2. Fail: Else. Check the oscillator's solder joints.

ii. Program the dsPIC with code that begins by initializing PORTD bit 8 as an output and then turns it on.

iii. Reset the dsPIC using switch S2.
   1. Pass: Almost immediately after the switch is released, the LED should light indicating that the dsPIC is powered, receiving a clock signal and able to run.
   2. Fail: Else.

G. Multiplexer

i. Remove jumpers JP1-JP5 to disconnect the sampling block from the analog front end.

ii. Set a variable power supply to 1 V and measure the AC noise on the power supply.

iii. Connect the variable power supply to the digital side of JP1 and connect the power supply ground to the circuit ground.

iv. Use the dsPIC code to set the multiplexer to pass channel 0 by setting PORTF=0. Read the DC output voltage of the multiplexer (pin 8) using an oscilloscope for each input.
   1. Pass: The output of the multiplexer has the same DC voltage as the input to the channel and changes when the channel input voltage is varied.
   2. Fail: Else.

v. Measure the AC noise on the output of the multiplexer and compare to the noise on the power supply being used to apply the input. Record the frequency of any noticeable periodic noise.
   1. Pass: The output of the multiplexer has no more than 10 mV of additional noise beyond that on the power supply.
   2. Fail: Else.

vi. Repeat for the remaining 4 channels, setting PORTF = 16, 48, 112 and 96 for channels 1-4, respectively and connecting the input to the appropriate jumper.

H. Analog-to-Digital Converter

i. Remove jumpers JP1-JP5 to disconnect the sampling block from the analog front end.

ii. Program the dsPIC to select receiver line 0 of the multiplexer

iii. Apply an adjustable input voltage to the sampling side of JP1 and set the voltage to 1 V. Make sure this power supply ground is connected to the circuit ground.

iv. Set a breakpoint in the dsPIC code and open up the watch window to view PORTB. Run 25 times, watching PORTB after each execution and keeping track of the minimum and maximum values of PORTB. Record the range of values seen.
    1. Pass: The average PORTB value displayed is 205 $\pm$ 20 and the value does not vary more than $\pm$ 3.
    2. Fail: Else.

v. Vary input voltage from 0 to 5 V, continuing to run after each small change.
    1. Pass: The value of PORTB should follow the changing input voltage.
    2. Fail: Else.

vi. Repeat ADC testing procedure for the remaining 4 channels, appropriately changing the input voltage connection and dsPIC multiplexer select line.

I. Sampling

i. Connect jumper JP1-JP5 from the sampling block to the analog front end.

ii. Power on the transmitter and firmly affix it in a position so that it is within the 40 cm of the receiver array and so its center axis is at an angle of less than 45 degrees from all receive transducers.

iii. Set up the dsPIC to continuously sample.
    1. Pass: Time between the rising edge of receiver one and the next rising edge is consistently 6.1 μ
    2. Fail: Else.

iv. Set up the dsPIC continuously sample and serial output all of the full curves for MATLAB
    1. Pass: The sampled curve for each channel looks like the scoped curve
    2. Fail: Else.

J. Bard Solver

i. Using a set of reasonable time delays, run the MATLAB Bard solver to obtain coordinate calculations.

ii. Disable the dsPIC sampling code and provide the same time delays as input to the dsPIC Bard solver.
    1. Pass: Output of the dsPIC Bard solver should be identical to the MATLAB version to within 0.5 mm in each coordinate axis.
    2. Fail: Else.

i. Power on the transmitter and firmly affix it in a position so that it is within the appropriate range and angle of the receiver array.

ii. Set up dsPIC code to send the Bard solver coordinate output through debugger mode of MPLAB.
    1. Pass: The location specified by the Bard solver does not vary more than ± 2.5 mm in any direction and the average location does not drift over time.
    2. Fail: Else. Note that the device can still be functional to a lesser degree of accuracy without meeting the precision and accuracy specifications.

K. PIC Communication

i. Configure dsPIC to output the values found in table 11 in floating point to UART.

ii. Configure USB PIC to receive through UART.

iii. Connect USB PIC to a computer through USB and ensure that it is detected properly.

1. Pass: The cursor on the screen moves in an "M" shape according to the provided coordinates.
2. Fail: Else.

x[ ]	=	[0.10,0.10,0.11,0.12,0.13,0.14,0.15,0.16,0.17,0.18,0.19,0.20]
y[ ]	=	[0.12,0.12,0.14,0.16,0.18,0.20,0.18,0.16,0.14,0.16,0.18,0.20]
z[ ]	=	[0.10,0.10,0.11,0.12,0.13,0.14,0.15,0.16,0.17,0.18,0.19,0.20]

Table 6: Testing vectors for dsPIC communication

L. Total System Integration

i. Power on the transmitter and firmly affix it onto the translation stage in a position so that it is within the appropriate range and angle of the receiver array.

ii. Ensure that receiver board is correctly connected to the computer through USB and detected.

iii. Adjust the translation stage by increments of 5 mm in each axis.

1. Pass: A movement of 5mm causes a noticeable movement on the computer screen in all axes and there is minimal jitter of the cursor.
2. Fail: Else.

iv. Remove transmitter from translation stage, attach to battery and affix to user's finger.

1. Pass: Movement of the user's hand causes appropriate movement on the screen.
2. Fail: Else.

M. Battery Charger

i. Measure the voltage of a depleted battery. The voltage should be less than 3.0 V

ii. Connect the battery to the battery charging connectors.

iii. Allow battery to charge for 1 hour and then remove from charger.

1. Pass: Battery voltage is greater than 3.7 V.
2. Fail: Else. Note that the battery charger is a secondary function and the system can still be effective without proper charging functionality.

5.2 Power Results

The power for the transmitter functions as expected, with the battery lasting for over 40 hours of continuous use between charges. Future improvements could include an ON/OFF switch, or a one-axis accelerometer controlled by the MSP430 to save battery power when the device is not in use. In a mass produced design, the battery shape could be customized to make it sturdier and further improve battery life.

The power for the receiver circuitry has one major flaw that has not yet been remedied. The -5 V rail has a large inductive kick at the switching frequency of the MAX764 voltage inverter. This spike is partially due to the PCB layout, but is worsened by an inadequate design in terms of voltage smoothing. On the PCB, the inductor is located next to the 5 V power line, and this allows the spike to easily travel through this line. Moving the inductor closer to the -5 V IC, and hence nearer to the conditioning capacitors, would diminish this leakage effect. Future additions to the circuit that would help alleviate this voltage spike are a charge pump or a voltage regulator connected to the -5 V output, and also the implementation of an inductor that has better shielding than the one currently being used.

Aside from this issue, the power for the receiver performs as expected. The receiver circuit draws less than 100 mA when running without the transmitter battery connected to the charging circuit, and with the MAX1555 regulating the current draw from the battery to 100 mA, there is an overhead of at least 300 mA beyond the USB port's maximum current rating of 500 mA. Future additions to this circuit could be include switch so that when the battery is charging, the rest of the circuit turns off to save power and limit the maximum current draw.

5.3 Transmitter Results

Testing of the transmitter involves making sure that it can produce an accurate and consistent signal. The transmitter's output at the microcontroller is viewed by probing the ACLK output pin on an oscilloscope. The final transmitter signal is viewed by connecting an oscilloscope probe to a receive transducer and viewing the received signal that has propagated through the air.

Figure 30 shows the actual signal output from the MCU to the transmit transducer, which looks very close to the ideal signal.

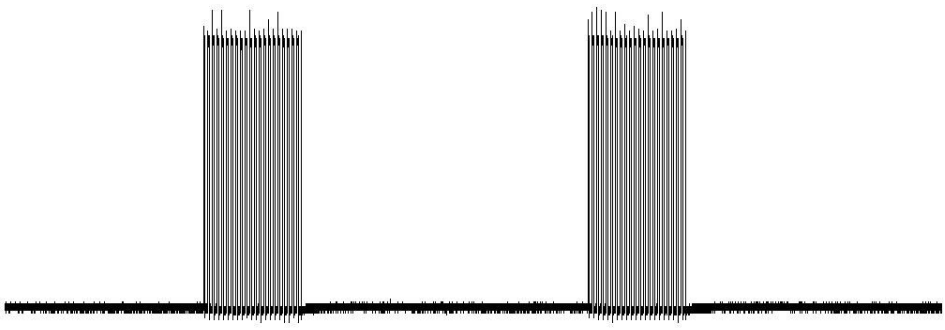


Figure 30: Transmitter MCU Output

Figure 31 shows the actual signal received at the array. The signal appears to be clean enough to detect an edge for processing. The additional waveforms that appear within the envelope of the signal are due to aliasing caused by the oscilloscope.

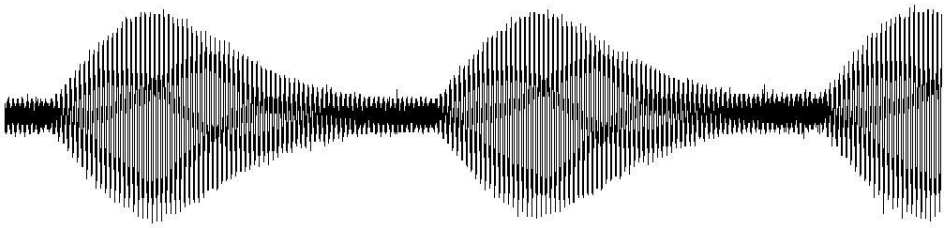


Figure 31: Received Signal Waveform

The transmitter is fully functional and meets its specifications, as shown by figures 30 and 31. It has been assembled onto a final printed circuit board and attached to the battery. Figure 32 shows the final assembled transmitter ring with battery.

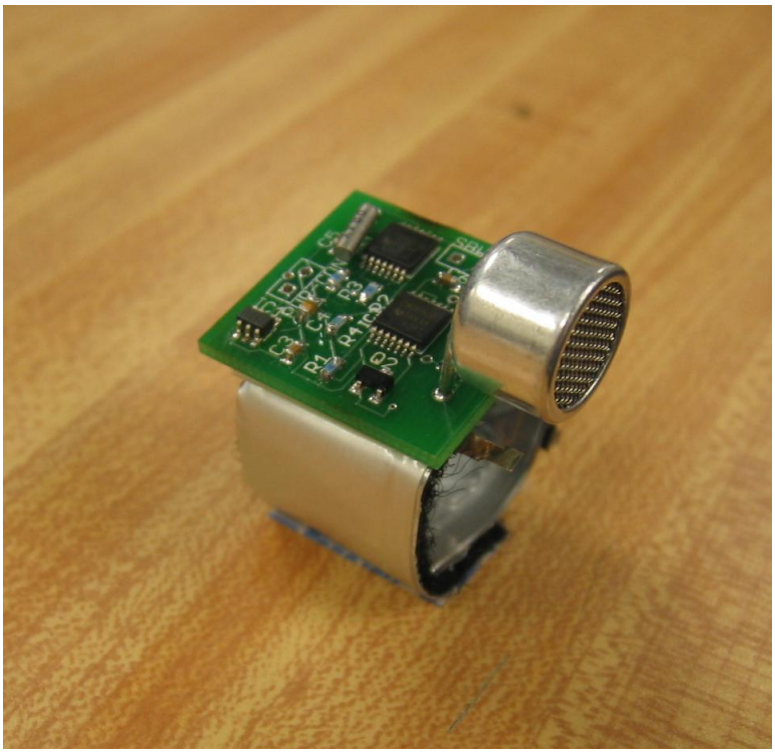


Figure 32: Transmitter Photo

5.4 Signal Conditioning Results

The receiver signal conditioning testing ensures that the analog signal conditioning components of the receiver circuit board are operating according to the requirements specified in the Methodology and Implementation Chapters of the report. The testing plan outlines the requirements of the AC Amplifier, Precision Rectifier, and Chebyshev Low Pass Filter blocks for each channel individually, and this section shows the results of these functional tests. The blocks are then tested together, with a sample signal pulse to show that they are able to function from end to end.

Before beginning the testing of the signal conditioning block, the power supply must be

checked. The MC33079 op amp is a dual supply op amp, and requires +5 V and -5 V power rails. A noiseless and stable power supply is crucial for analog electronics to operate at full functionality. If there is noise on the power rails supplying the signal conditioning block, this noise will propagate through the signal lines on each channel and these undesired effects will be present on the output. Power supply testing conducted prior to the testing of this section ensures that there is a nominal amount of noise on the signal. For sake of isolation from the power functional block, a lab bench power supply was used. Despite being a commercial product, a small amount of noise was detected when the rail was measured with an oscilloscope. The ripple may be seen in Figure 33.

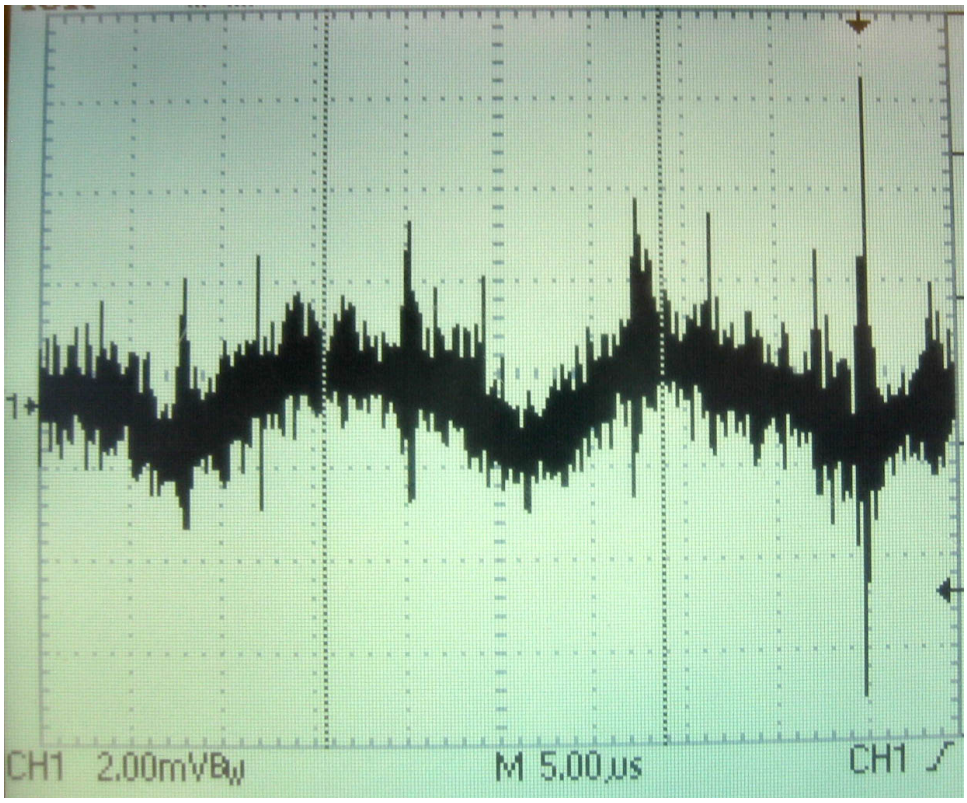


Figure 33: Power Supply Ripple

The power supply ripple frequency is approximately 54 kHz. The magnitude of this ripple was measured to be approximately 2 mV, with non-periodic voltage spikes of up to 14 mV. Since the ripple voltage of the power supply is close to the average magnitude of the received signal,

this will have effects on the signal voltages if proper precautions are not taken. For this reason, 0.1 μF bypass capacitors between the +5 V rail and ground and the -5 V rail and ground are included in the circuit design. Despite these precautions, the power rails on the receiver PCB were measured with an oscilloscope to have about 2 mV of remaining 54 kHz noise.

5.4.1 AC Amplifier Operation

The AC Amplifier is tested to ensure that it blocks DC voltages on its input, and only applies unity gain to the DC offset voltage of the op amp. A trim resistor could have been included in this stage of the design to reduce this DC offset, but board space was a premium, and the DC offset voltages of the MC33079 are a maximum of 15 mV. This was deemed acceptable, as it does not affect the overall pulse shape of the received signal.

A test input of a 20 mV peak-to-peak 40 kHz sine wave was applied to the input of each AC Amplifier channel. The outputs were measured with an oscilloscope, and a sample output is shown in Figure 34.

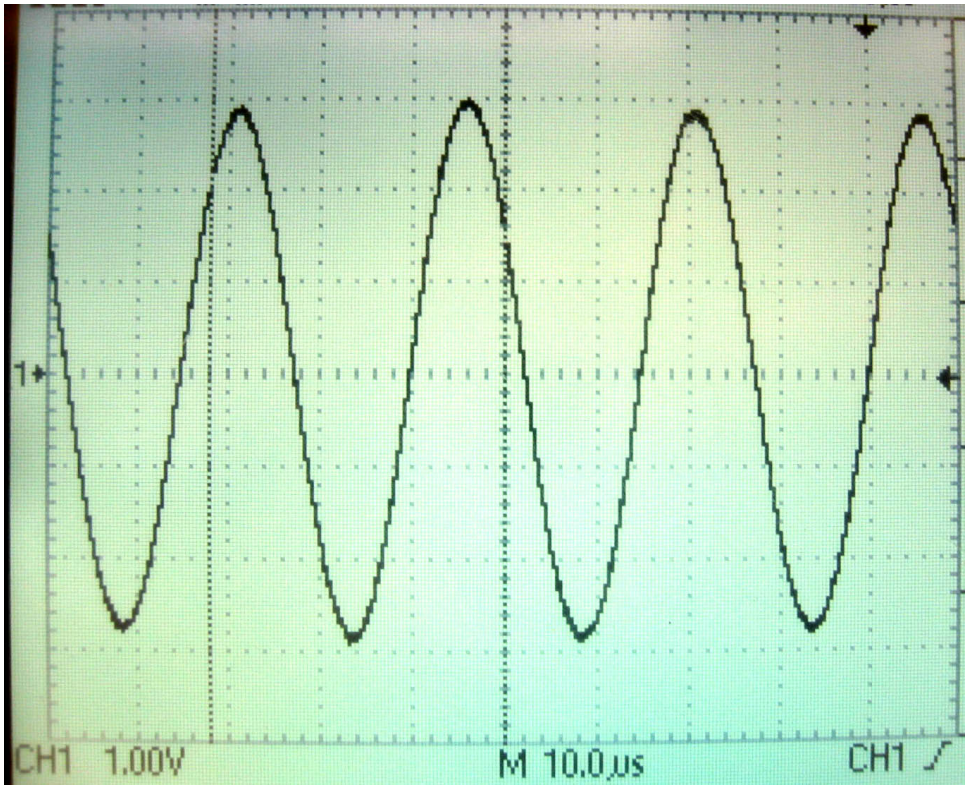


Figure 34: AC Amplifier Sample Output (Test A)

Table 7 shows the results of this testing.

Channel	Unattenuated 40 kHz Sinusoid
0	Pass
1	Pass
2	Pass
3	Pass
4	Pass

Table 7: AC Amplifier Testing Results (Test A)

The gain of each channel was measured to be roughly 49.5 dB.

The second test applied to the AC Amplifier block is a DC voltage blocking test. A DC voltage of 1 V was applied to the signal inputs of each AC amplifier channel. A sample output voltage from this test is shown in Figure 35.

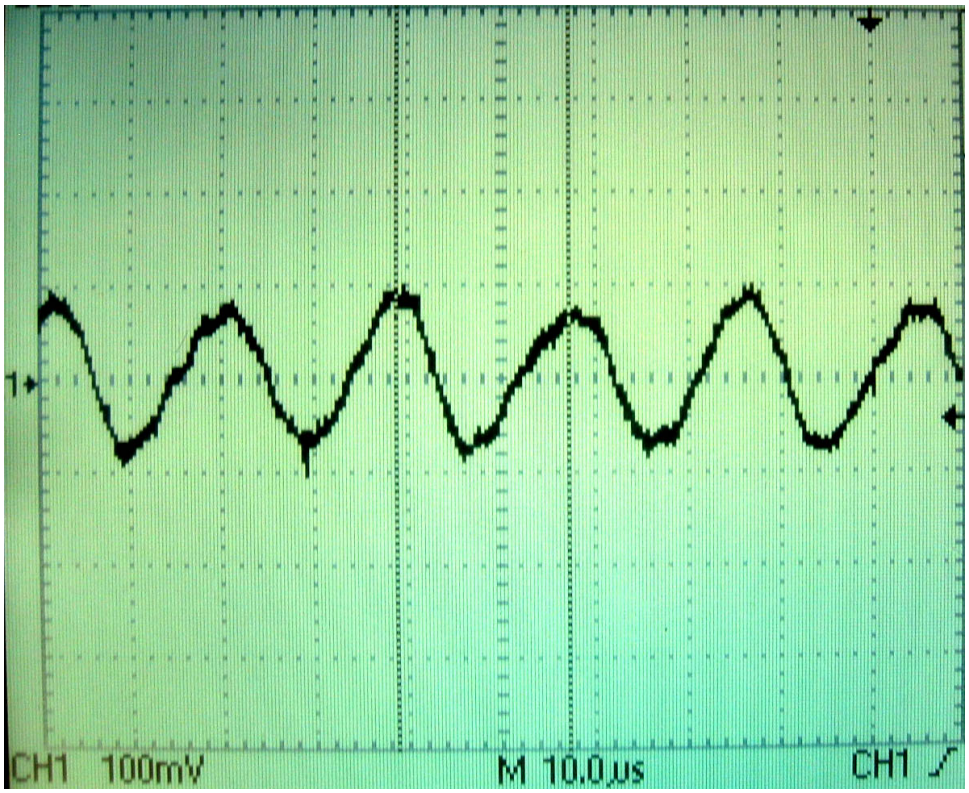


Figure 35: AC Amplifier Sample Output (Test B)

The expected output of this section is a constant DC voltage between ground and 15 mV. The 54 kHz oscillation that is present on the bench power supply voltage is measured on the output of this signal with the oscilloscope, however. After a gain of 48 dB, any small 54 kHz fluctuation on the DC power supply input is expected to have a peak-to-peak voltage of approximately 200 mV. This test determines the DC blocking ability of the AC Amplifier. For this criteria, the AC Amplifier passes on all channels. The results of the testing are tabulated in Table 8.

Channel	Attenuates DC
0	Pass
1	Pass
2	Pass
3	Pass
4	Pass

Table 8: AC Amplifier Testing Results (Test B)

5.4.2 Precision Rectifier Operation

The Precision Rectifier block is tested for each channel of the receiver to ensure that any negative component of the amplified signal is removed. This is required to detect the signal envelope and provide the correct voltage levels for the input to the ADC. The first test outlined in the testing plan is the application of a 2 V peak-to-peak 40 kHz sine wave to the input of the precision rectifier stages for each channel. For this test, the inputs to the AC Amplifier blocks were grounded to prevent interference with the input voltage. Since the op amp of the AC Amplifier block is attempting to push its v+ and v- inputs into equivalence, the input voltage applied to the Precision Rectifier stage is distorted. Figure 36 shows the distortion when a pure 40 kHz sine wave is applied to the input of the Precision Rectifier when measured with an oscilloscope.

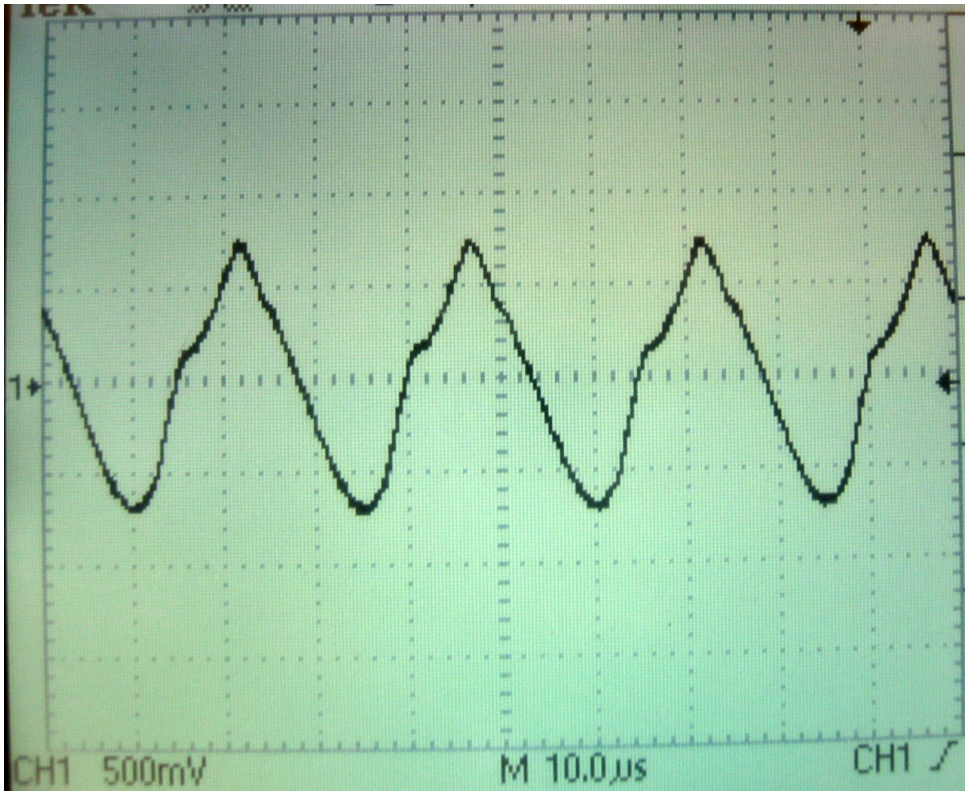


Figure 36: Precision Rectifier Sample Input (Test A)

A sample output is shown in Figure 37 when measured with an oscilloscope.

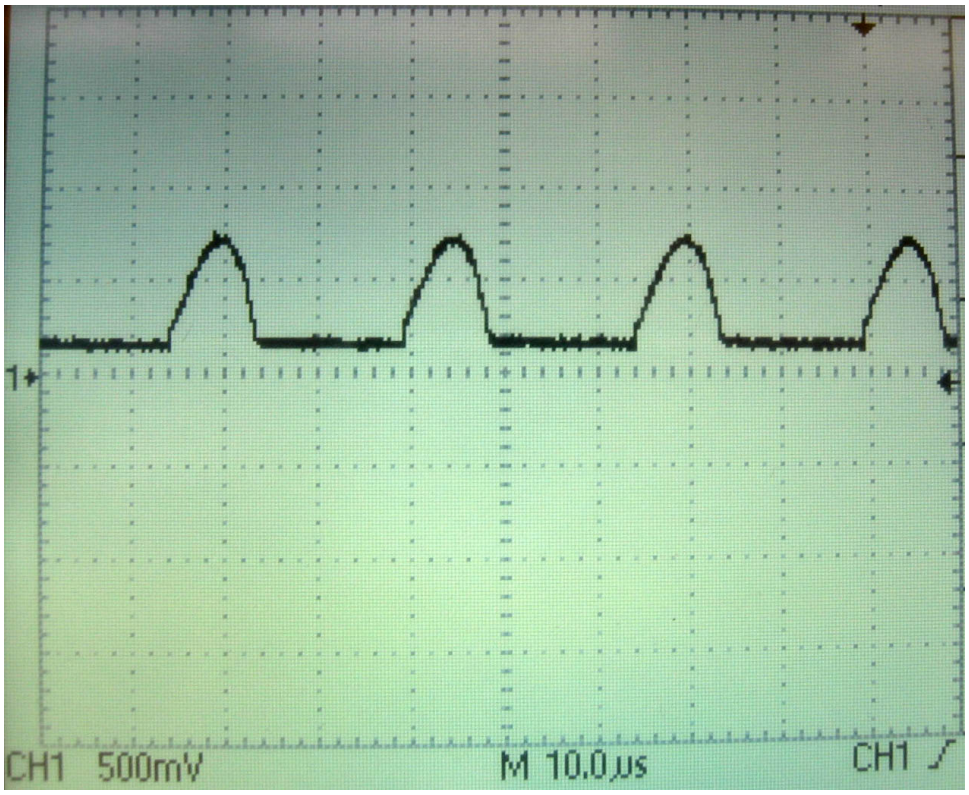


Figure 37: Precision Rectifier Sample Output (Test A)

The input signal was applied to all channels in a similar manner, and they showed the same result. The results of the tests are shown in Table 9

Channel	Rectifies Signal
0	Pass
1	Pass
2	Pass
3	Pass
4	Pass

Table 9: Precision Rectifier Testing Results (Test A)

The next test of the Precision Rectifier block shows the behavior of the signal when the inputs are grounded. The sample output was measured with an oscilloscope and is shown in Figure 38.

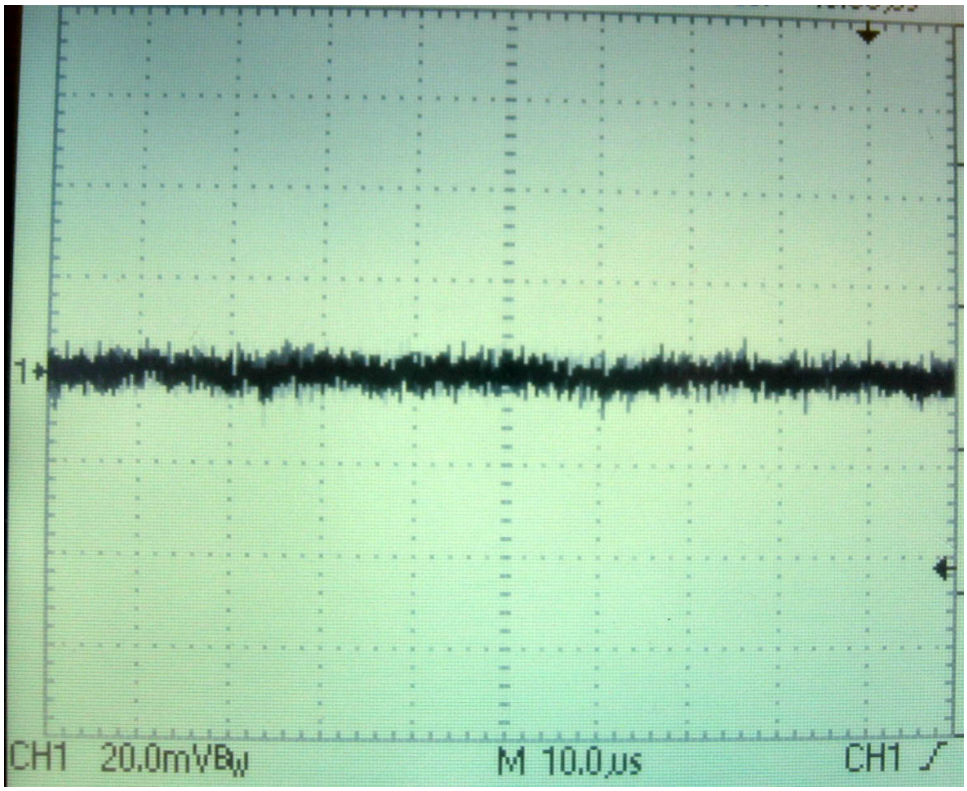


Figure 38: Precision Rectifier Sample Output (Test B)

There is no difference between the signal shown in Figure 38 and when the oscilloscope probe is grounded. The results of the tests are shown in Table 10.

Channel	No DC Distortion
0	Pass
1	Pass
2	Pass
3	Pass
4	Pass

Table 10: Precision Rectifier Testing Results (Test B)

5.4.3 Chebyshev LPF Operation

The Chebyshev Low Pass Filter is tested for each channel of the receiver in order to ensure that the filter is not oscillating when the input is grounded and that the 40 kHz component of the signal is filtered out. For the first test, the input to each Chebyshev filter is grounded, and

the output is measured using an oscilloscope. The measured waveform is shown in Figure 39.

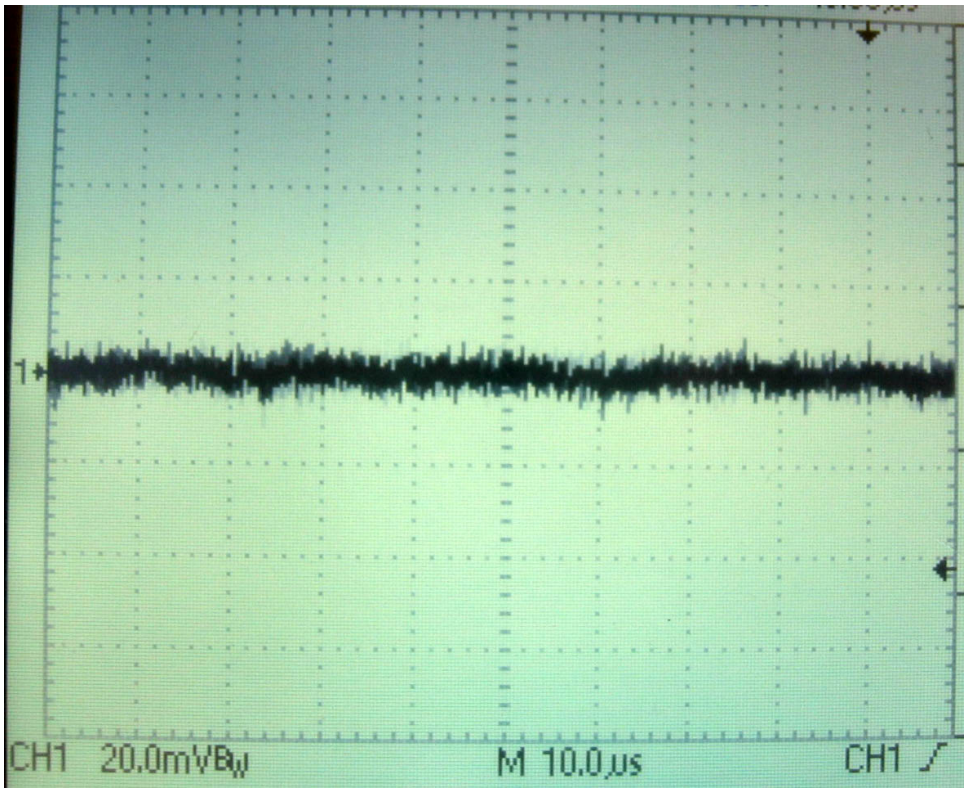


Figure 39: Chebyshev Sample Output (Test A)

As a low pass filter, the measured output behaves as expected for this test. The tabulated form of the results are shown in Table 11.

Channel	Grounded Output
0	Pass
1	Pass
2	Pass
3	Pass
4	Pass

Table 11: Cheybshev Low Pass Filter Testing Results (Test A)

The second test that was conducted on the Chebyshev low pass filter utilized a 2 V peak-to-peak 40 kHz sinusoid with a 1 V DC offset generated by a function generator as input. The sample waveform is shown in Figure 40.

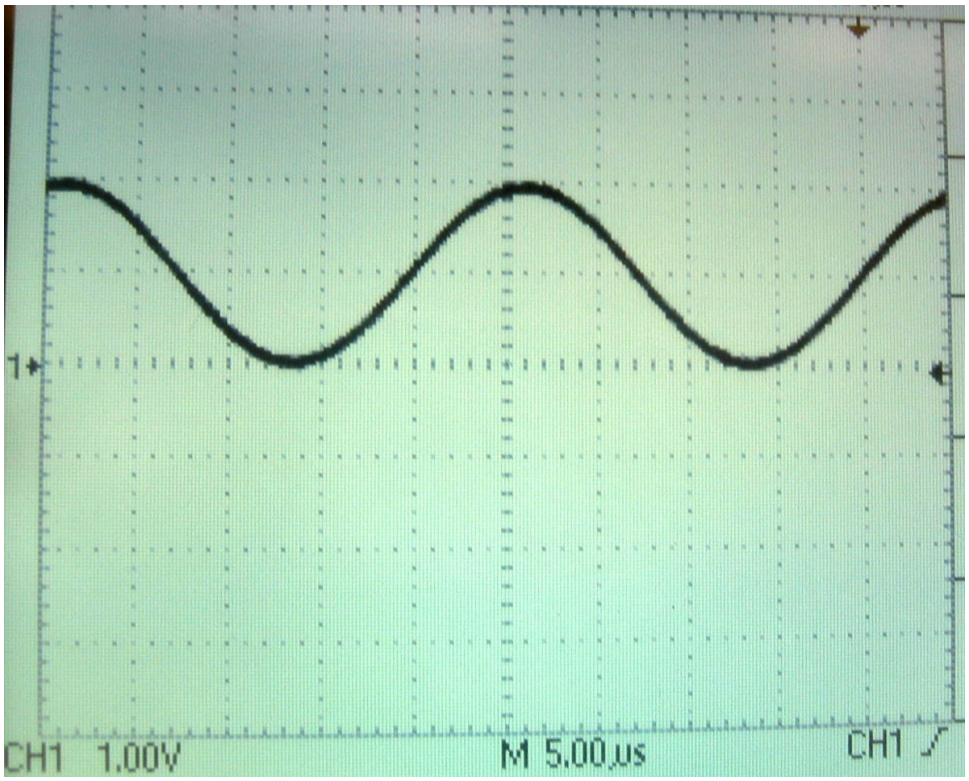


Figure 40: Chebyshev Sample Input (Test B)

The signal was applied to each channel's Chebyshev filter and the output was measured using an oscilloscope. A sample waveform is shown in Figure 41.

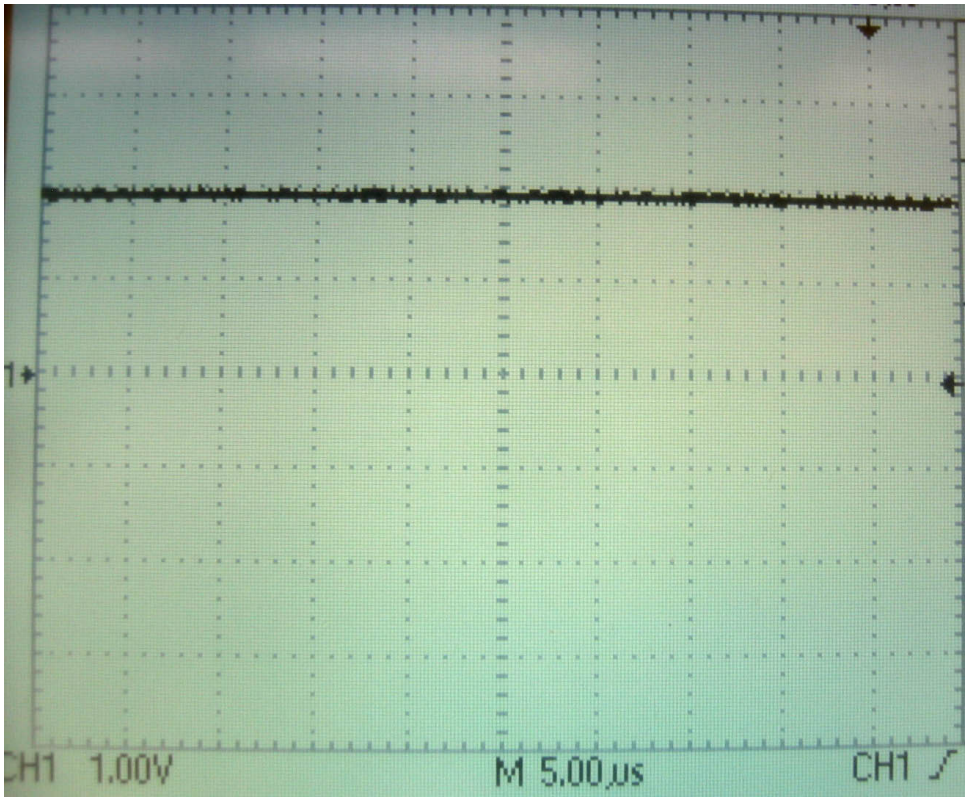


Figure 41: Chebyshev Sample Output (Test B)

The result is a DC voltage of 2 V. This is because the filter is designed with a pass-band gain of 2. The 40 kHz ripple is attenuated from the signal and matches the theoretical predictions for the output of the filter with this particular signal input. The test input was applied to each channel, and the tabular form of the results is shown in Table 12

Channel	2 V DC
0	Pass
1	Pass
2	Pass
3	Pass
4	Pass

Table 12: Cheybshev Low Pass Filter Testing Results (Test B)

5.4.4 Transmitter-Receiver Integration Testing

The transmitter and receiver system are tested as one functional block to ensure that the expected pulse shape arrives at the output of the Chebyshev low pass filter. The transmitter was tested prior to the analog signal processing to ensure that it was functioning properly. The transmitter was powered and placed 20 cm from a receiver transducer, in order to simulate normal operational ranges. The positive terminal of the same receiver transducer was connected to the signal input of each AC Amplifier stage. This setup helps to determine the latencies associated with each channel in a following system test. The output of each AC amplifier block was measured with an oscilloscope and a sample waveform is shown in Figure 42.

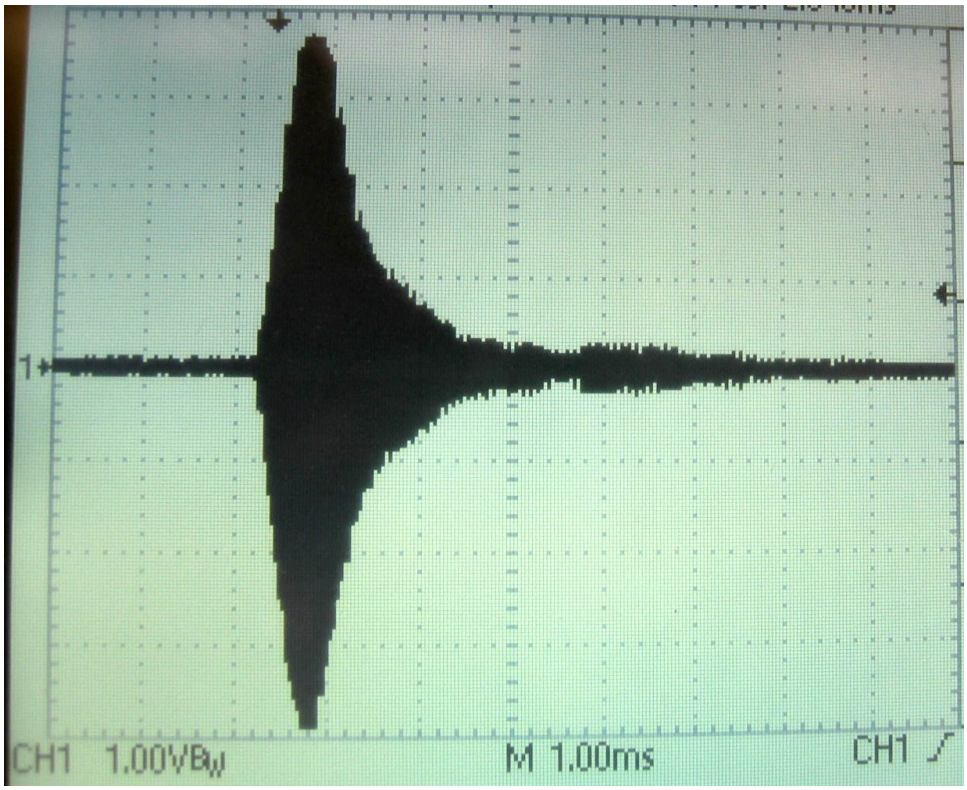


Figure 42: AC Amplifier Pulse Sample

The sample waveform is a gained version of the signal that is received from the transmitter on the receiver transducer. An image of the received signal is shown in Figure 31. The output of each Precision Rectifier block was measured with an oscilloscope and a sample waveform is

shown in Figure 43.

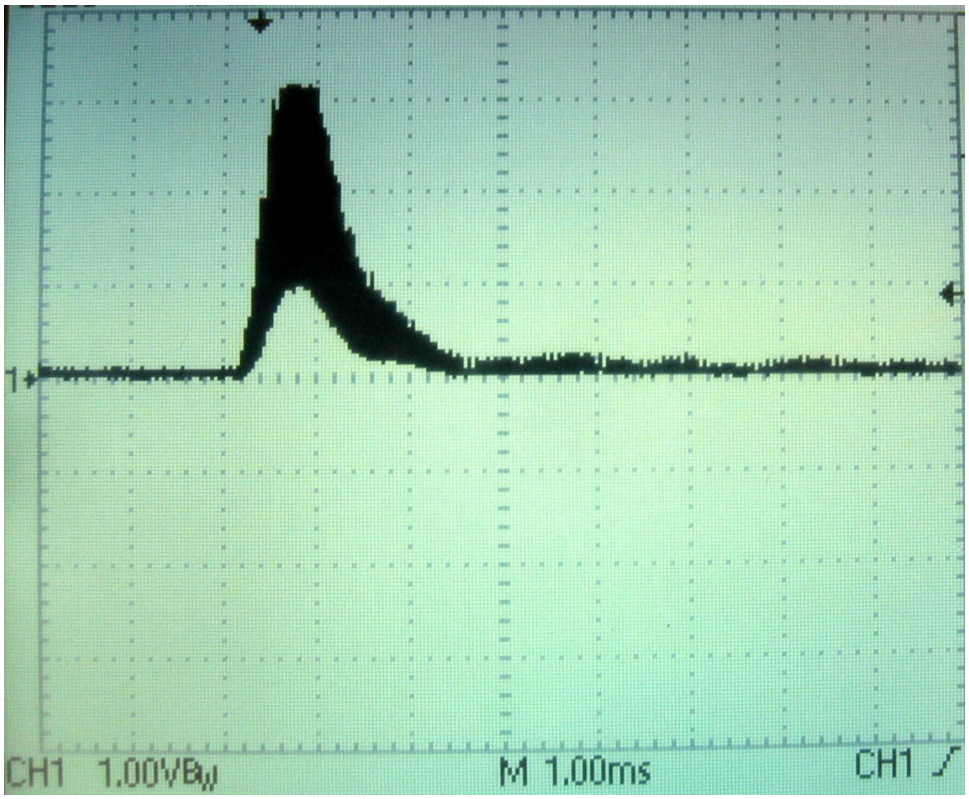


Figure 43: Precision Rectifier Pulse Sample

This signal is a rectified version of the signal shown in Figure 42, and was observed to be similar on all channels. The output of each Chebyshev Low Pass Filter block was measured with an oscilloscope and a sample waveform is shown in Figure 44.

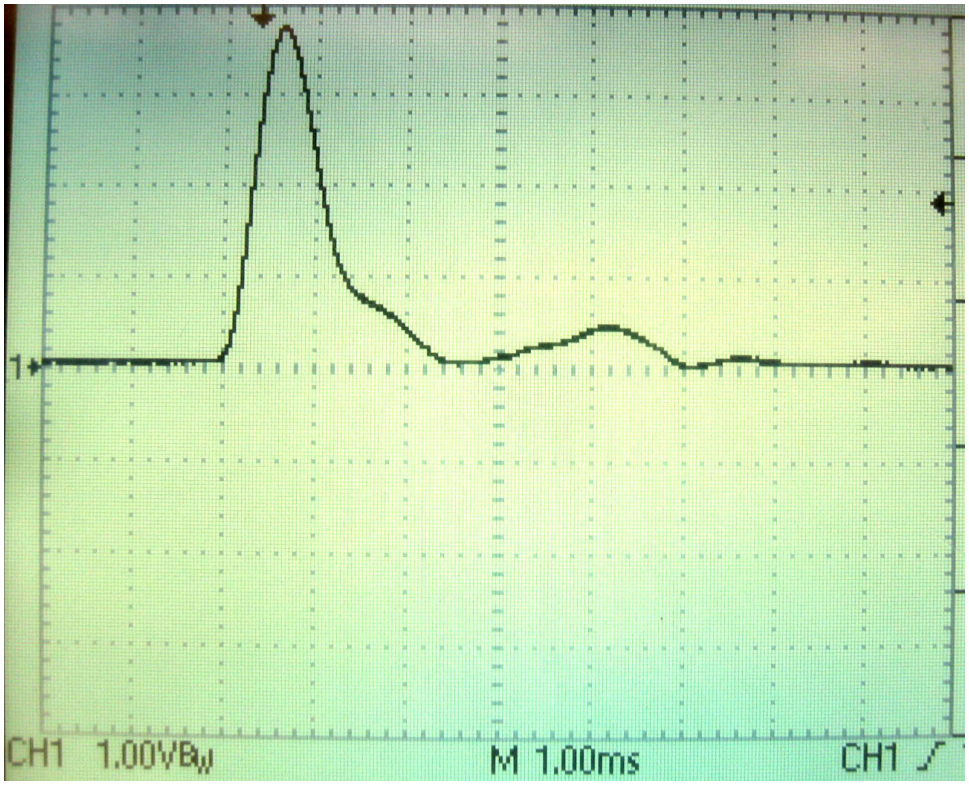


Figure 44: Chebyshev Pulse Sample

The signal is the expected waveform. The critical part of the pulse is located around the peak. The lobe effects to the right of the peak in Figure 44 are caused by multipath effects, and have not been observed to reach pulse peak height, which is important for accurate peak detection. The ripple voltage of the 40 kHz signal at the peak of the pulse was measured with an oscilloscope, and was shown to be approximately 5 mV peak-to-peak. The ADC in its current configuration can resolve down to a minimum of 5 mV, and the ripple is within this margin. A sample of the measured pulse ripple is shown in Figure 45 as measured with an oscilloscope. The ripple magnitude was measured to be similar on all 5 channels.

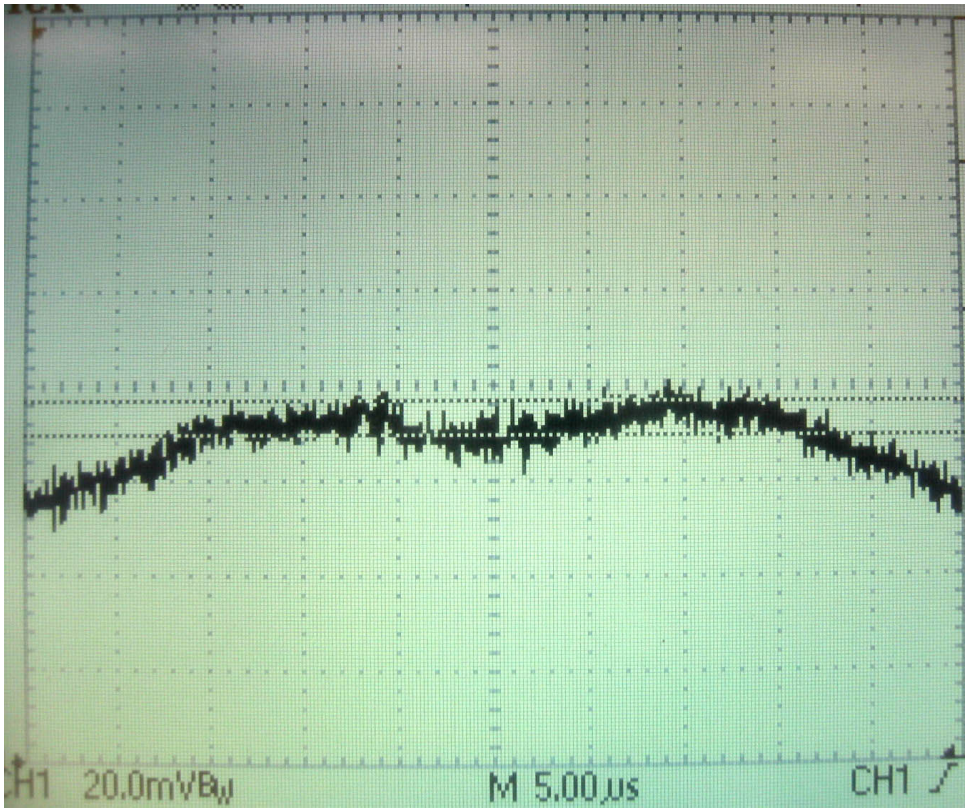


Figure 45: Chebyshev Pulse Ripple

A tabular form of the results of the preceding tests is shown in Table 13.

Channel	AC Amplifier	Precision Rectifier	Chebyshev LPF	Ripple Voltage
0	Pass	Pass	Pass	Pass
1	Pass	Pass	Pass	Pass
2	Pass	Pass	Pass	Pass
3	Pass	Pass	Pass	Pass
4	Pass	Pass	Pass	Pass

Table 13: Transmitter-Receiver Integration Results

Latency Measurement

With the above configuration it is easy to measure the circuit latency of each channel from the time the pulse is transmitted to the time the signal peak is detected on the output of the Chebyshev low pass filter. The latency was measured with an oscilloscope using the signal on the transmit transducer as a trigger point. The Chebyshev filter peaks were measured on each channel

to be within 5 μs of each other. Using this technique and the sensitivity of the equipment in the lab, the difference in delays could not be resolved further than this. There was some fluctuation in these measurements due to multipath issues and the orientation of the oscilloscope with respect to the Transmitter-Receiver channel.

5.5 Processing Results

Testing the processing block involves checking that the hardware for the multiplexer, Analog to Digital Converter (ADC) and dsPIC all work independently and then combining them to test the entire block. After hardware is determined to be functional, it is necessary to test the software. This section discusses the testing and results for the processing block from the output of the analog front end to the input of the USB interface.

5.5.1 dsPIC Basic Operation

Before any testing of the processing block can proceed, it is necessary to verify that the dsPIC microcontroller is functional at a basic level. The clock output of the crystal oscillator is first measured to ensure that the clock signal is to specification. This test lights an LED to verify that the dsPIC is receiving power and an adequate clock signal and that it can run properly and output to an I/O port. The ability to turn on an I/O port is one of the most basic functions of a microcontroller and is a good test of basic functionality. As none of the advanced I/O features of the dsPIC are being used, this test can qualify that the processor is functional.

The maximum switching noise observed on the clock was 480 mV, which corresponds to less than 10% of the 5V peak voltage, which is better than the allowable 12%. The maximum ripple noise is 240 mV, corresponding to just under 5% of the peak and therefore passes the 5% requirement. The clock successfully passed both tests and can therefore be considered to perform adequately.

The dsPIC was programmed using a copy of the processing code with all of the functions commented out except for the initialization. When the dsPIC is programmed and reset, it effectively turns on the LED and can therefore be considered functional.

5.5.2 Multiplexer

The multiplexer was tested to determine that it can correctly pass the selected signal without introducing significant noise. As described in the testing plan, for multiplexer performance to be considered acceptable, it must be able to pass the signal to output and the noise introduced must be minimal at the point of sampling.

Table 14 shows the results of this testing.

Channel	Passes Input to Output	Noise	Noise Acceptable
0	Pass	10 mV	Pass
1	Pass	10 mV	Pass
2	Pass	10 mV	Pass
3	Pass	10 mV	Pass
4	Pass	10 mV	Pass

Table 14: Multiplexer Testing Results

5.5.3 Analog-to-Digital Converter

The Analog-to-Digital Converter (ADC) is tested to ensure that it can provide a proper digital representation of an analog voltage to the dsPIC and that its output is stable. It must also be able to correctly reflect changes to the input voltage at its output. Because functionality of the dsPIC has already been tested, the values can be read into the dsPIC for simplicity of viewing.

For each multiplexer channel, the ADC was read 25 times to determine the average output and the output noise. Table 15 shows the range of values obtained for each channel.

Channel	Minimum	Maximum	Pass / Fail
0	191	196	Pass
1	189	194	Pass
2	192	199	Pass
3	190	196	Pass
4	187	192	Pass

Table 15: ADC Output Results

The next test was a qualitative test of the ADC output's ability to follow input changes. The output correctly tracked the input for all channels across a range of 0 to 5 V, showing outputs

of 0 to 1023. Based on these tests, the ADC can be considered fully functional.

5.5.4 Sampling

The sampling was tested to determine that it is sampling at a consistent rate of 6.10 μs between samples for each channel and that it starts sampling at the intended threshold with a proper shape. The sampling rate is checked using the output of the multiplexer. Each channel is meant to output for 1.22 μs before switching to the next channel, for a total of 6.10 μs between samples on a given channel. The sampling rate is verified experimentally by measuring the time between when a channel is selected and when it is selected again after all other channels are selected. Figure 46 shows an oscilloscope capture of the multiplexer output. It goes through all five channels in a period of 6.10 μs, which proves it is sampling at the correct rate.

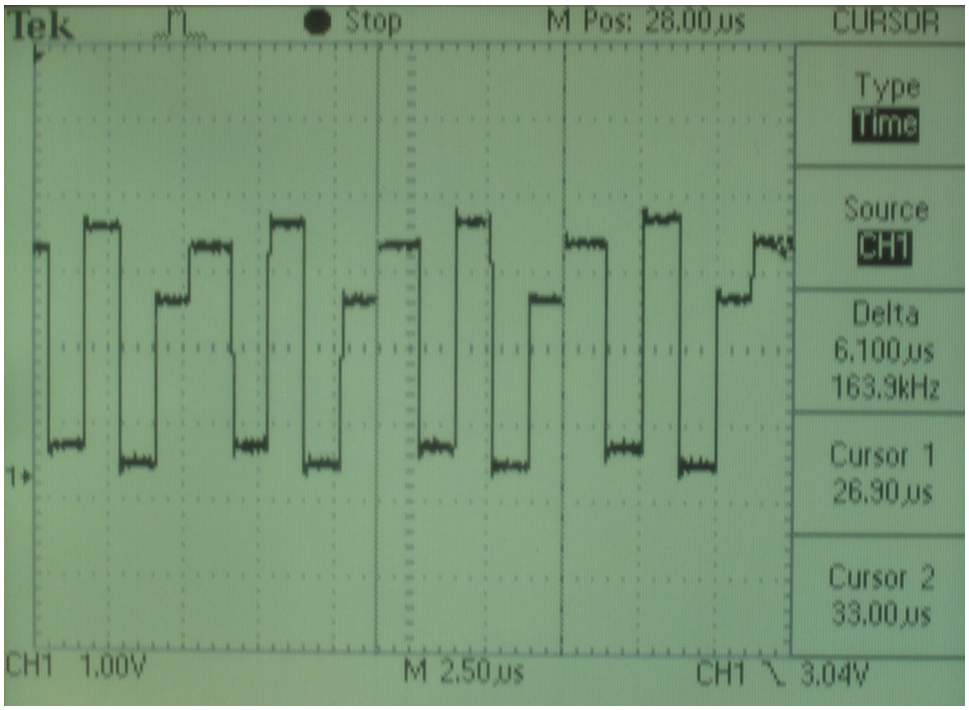


Figure 46: Sampling Time

Figure 47 shows the shape of the sampled curves. This curve is the basis for the curve fitting algorithm and must be the correct shape. The sampled curve shape is similar to the shape of the scoped channel, so the sampling code passes.

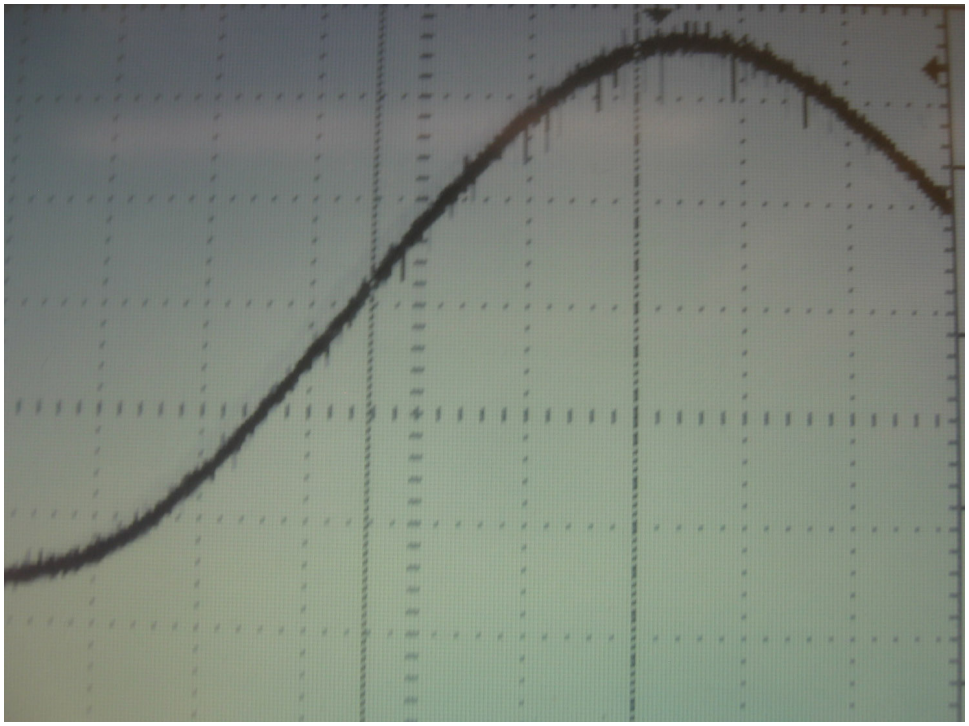


Figure 47: Curve Shape

5.5.5 Bard Solver

To prove the Bard solver is working properly, the output of the dsPIC should be the same as the proven MATLAB version. This can be verified by entering the same time delays into MATLAB and the dsPIC and comparing the coordinate outputs. Figure 48 shows a comparison between the results of the MATLAB and dsPIC versions of the Bard solver. The outputs match, showing that the dsPIC version of the Bard solver functions properly.

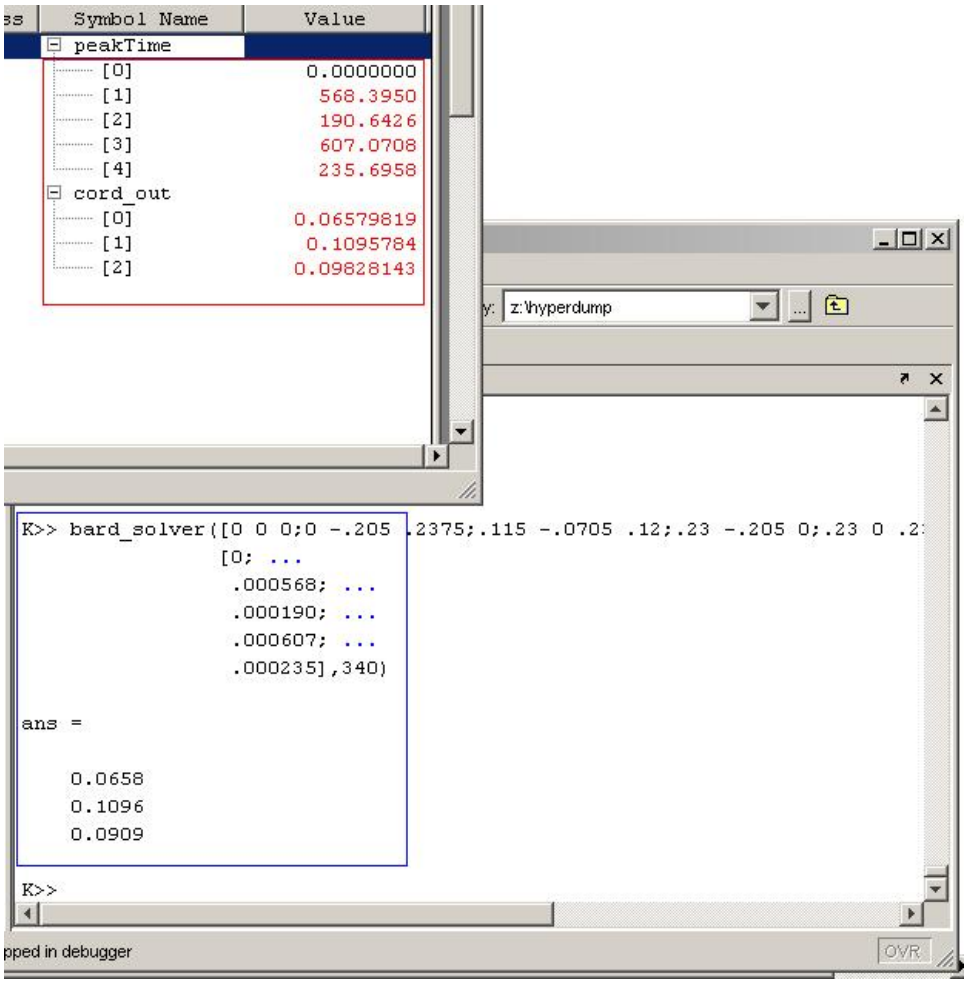


Figure 48: Bard Output

5.6 Processor Communication Testing

This section will evaluate the performance and functionality of the interface between the device and the PC. The functionality of the device depends on the ability for the device to establish and maintain communication with the PC as well as regularly receive ring coordinates from the dsPIC30F3013 (dsPIC). Performance of this subsystem shall not hinder the system in any way. Coordinates sent to the PIC18F2455 (usbPIC) should be used in trivial calculations and immediately sent to the PC before the next coordinates are transmitted by the dsPIC. The system's overall performance cannot be lagged by the usbPIC and therefore operations must be as fast as possible.

5.6.1 USB connectivity

One of the goals for the device is ease of portability. Specifically, the adoption of the USB/HID protocol which is understood by all modern computers. Other protocols such as PS/2 are not plug and play, and a non-standard USB device requires special drivers to be installed. USB communication with the PC was trivial due to the example code provided by Microchip. These examples contained the structure for code which would maintain the USB connection and was responsible for transmitting the data payload to the PC. The code which was written by the team must assure that the `USBTasks()` function be called regularly. Therefore code must be carefully written to not poll for I/O or become stuck in conditional loops.

5.6.2 System integration

The PIC18F2455 chip used for PC-Connectivity was originally planned to be programmed on the fly over USB. The ease of the bootloader could potentially allow the end-user to customize the device and the firmware. However, for the proof of concept device, it was logical to leave out the bootloader and program the firmware directly using the Microchip ICD2 debugger.

The output of the `bard_solver()` is three floating point values representing the X, Y, and Z absolute positions of the transmitter. Floating point types are 4 bytes each. The total size of the data transmitted over the USART to the usbPIC will be 14 bytes as shown in Figure 5.6.2. The last two bytes are used as an `ACK` for the usbPIC to notify the dsPIC that its data was received.

x-data (4 bytes)	y-data (4 bytes)	z-data (4 bytes)	ACK "ok" (2 bytes)

Table 16: USART dsPIC to usbPIC data packet

Initially the team had planned to read data over the USART by polling the receive pin between USB transfers. This proved a solid idea for the test bit rate of 9600 BAUD. Equation 33 below shows when the `bard_solver()` has output its data every 10 ms, a baud rate of 19,200 is required. With such a high baud rate, polling the USART lines for incoming data would not be feasible. The alternative is to introduce an interrupt which will trigger and divert the program

counter when data is available on the USART lines. The accompanying code can be seen in Appendix B.7.

$$\frac{14 \text{ bytes}}{10 \text{ ms}} \times 8 \text{ bits} = 11,200 \text{ baud} \rightarrow 19,200 \text{ baud} \tag{33}$$

The coordinate input was first tested by applying the coordinates which were shown in Table11 in the testing plan. At first these coordinates would move the mouse appropriately in the shape of an 'M'. Occasionally the device would become out of sync between coordinate values and the mouse would jerk across the screen and disconnect. This was temporarily solved by adding the `ACK` packet to the transmission.

When the test coordinates were replaced with the generated coordinates from the `bard_solver()`, the received data would quickly fall out of sync. Several tests would show cursor movement which reflected the motion of the mouse, but this would only last several seconds. It was clear that the device was repeatedly dropping packets of data and losing synchronization. Due to time constraints, this issue was not resolved. The issue here is the complexity of the dsPIC calculations as well as the constant need for USB transmission from the usbPIC to the PC. During this time interrupts cannot be active and therefore any incoming data to the device will be waiting on the buffer, and not handled correctly. A possible approach to this problem is using a more advanced method of handshaking which would allow the two devices to synchronize and transmit data reliably. In the near future, this issue will be investigated and solved.

5.7 System Integration Results

This section analyzes the results of system integration and discusses how the device performs as a complete system. Although the connection between the dsPIC and USB interface could not be completed, the coordinate output of the dsPIC can be passed to a computer through an RS-232 serial interface and analyzed in MATLAB.

These results were determined with the system only calibrated using estimated measurements of the receiver locations. Imprecise knowledge of the receiver locations can introduce large

amounts of error, so it can be expected that a properly calibrated system will perform far better. Even in these non-ideal circumstances, the precision and motion tracking is reasonable. These results show that the device performs effectively and has potential to be quite precise.

5.7.1 Precision

Precision refers to the device's ability to report the same output given a constant input. This is measured by reading the output of the Bard solver in MATLAB with the transmitter fixed in place.

Figure 49 shows the radial error for the integrated system. This is a measure of the radius of error around a fixed point due to inaccuracies in the system. The average error is approximately 2.5 mm and the standard deviation is 3.54 mm.

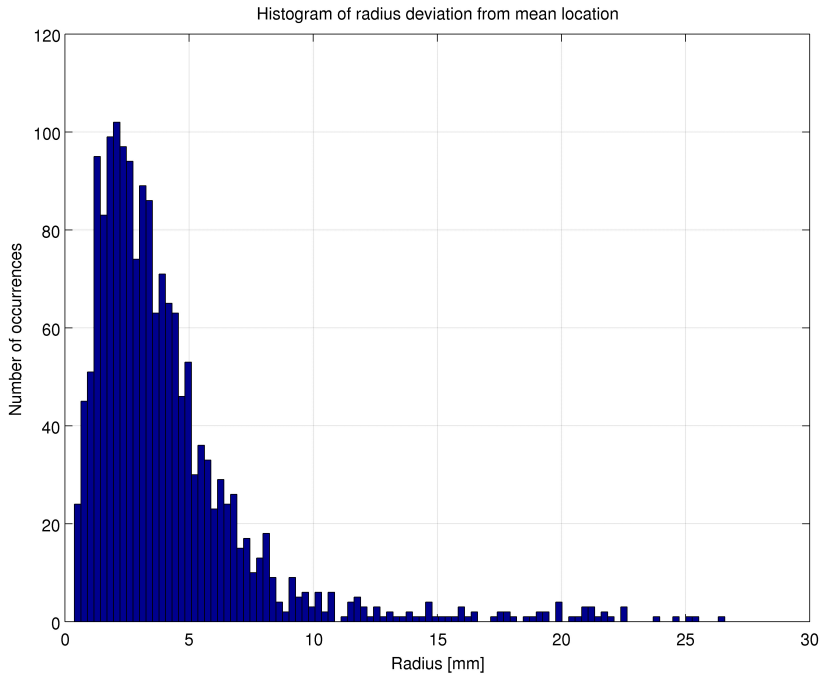
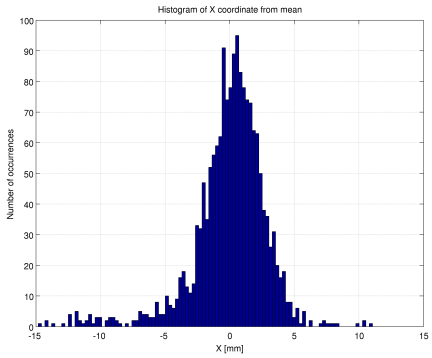


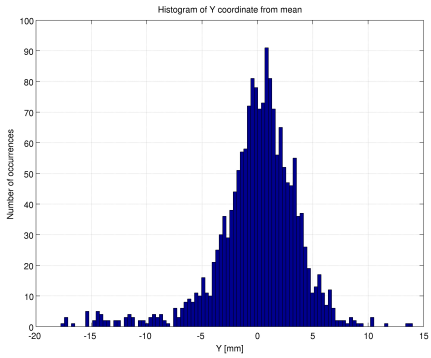
Figure 49: Bard Solver Radial Error Histogram

Figure 50 shows the single-axis error for the X, Y and Z axes, respectively. The standard deviations for these error plots are 2.88 mm, 3.75 mm and 2.76 mm, from left to right. In the current minimally calibrated state, the Y-axis has significantly more error than the other two

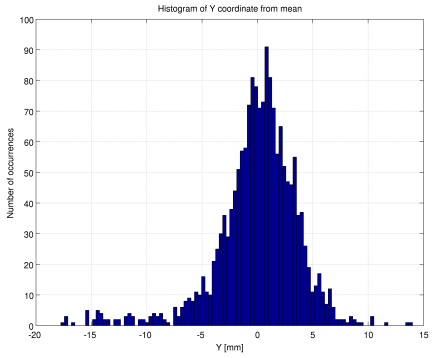
axes. With improved calibration, the error could be reduced in all axes, which would significantly improve the radial error.



(a) Bard Solver X-Axis Error Histogram



(b) Bard Solver Y-Axis Error Histogram



(c) Bard Solver Z-Axis Error Histogram

Figure 50: Bard Solver Single-Axis Error Histograms

5.7.2 Accuracy

The system's accuracy is defined by its ability to accurately report the location of the transmitter. Accuracy has not yet been tested because it is very difficult to determine the precise location of the transmitter. Although a knowledge of accuracy is relevant, it is not critical, because the main purpose of the system is to track relative movements of the transmitter.

5.7.3 Motion Tracking

Motion tracking is a determination of how well the system can track the movement of the transmitter.

Figure 51 shows a plot of motion tracking. Data is collected with the transmitter in one position (red X's), then it is moved by 8mm in the -X direction and additional data is collected (green O's). The two sets of data points are centered about different points approximately 8mm apart. This shows that the device is able to properly detect a change in transmitter location and display it at the output.

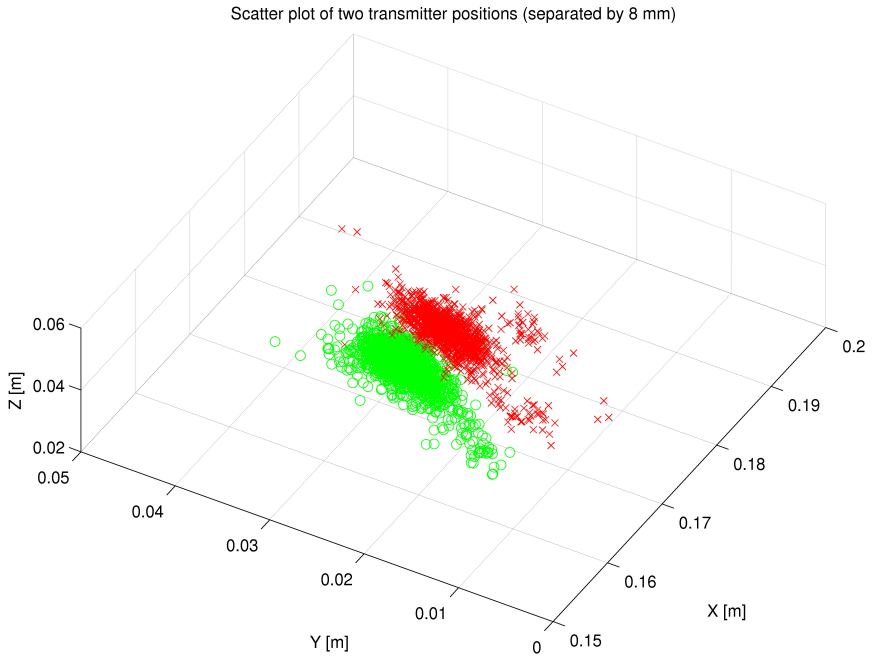


Figure 51: Motion Tracking Plot

Due to the lack of calibration, the data points in figure 51 are not circular because there is more error in the Y-axis than the other two. This agrees with figure 50, which shows the increased error for the Y-axis.

Although the data is not especially precise at this point, the system is capable of showing a response on the computer when the mouse is moved. Further calibration will yield improved results and will increase the system's accuracy, precision and ability to track motion.

6 Conclusions

This section discusses the overall design, specifications, future improvements and potential applications. Based on the device performance and potential improvements, conclusions are drawn regarding the effectiveness of the device and marketability.

6.1 The Final Design

The final design consists of a small, discrete, low-power transmitter, that transmits a 40 kHz ultrasonic pulse, and a receiver system that converts these pulses into 3D positional coordinates. The receiver array is composed of a specially designed aluminum frame that houses five receiver transducers. The signals received by this array are shaped, sampled, and their peaks are determined. This process involves curve fitting regression that compares each signal's received pulse shape to that of a standard stored pulse shape. This allows increased noise resistance over direct sampling methods by averaging the error of the signal and provides high time resolution with a low number of sampled points. The peak times are accurately converted into positional coordinates using the Bard solver algorithm. These positional coordinates are transmitted through serial communication to the PC, and their coordinates are displayed in MATLAB. The UART communication between dsPIC and usbPIC was not fully implemented, so full USB functionality was not achieved. Regardless, the simulation in MATLAB is able to demonstrate movement of the ring in three-dimensional space in real-time and shows reasonable precision, despite the fact that it is not fully calibrated.

Appendix D lists the parts used in the prototype design for the complete system. At under $150 for a single unit and less than $85 in larger quantities, the cost of this device is quite reasonable for its innovative functionality. Compared with the Logitech 3D mouse, which sells for $1500 or more, it could be offered at a much more affordable price. It may not be inexpensive enough to completely replace the standard mouse, but could certainly become popular for both business and personal use.

6.2 Performance Specifications

Table 17 shows a summary of the system specifications. Overall the system has met the project goals, while remaining at a reasonable price level.

Parameter	Specification	Units
Precision (3D Radial)	3.54	mm
Cost	$148	per unit, single quantity (Appendix D)
	$83	per unit, quantity of 1000+ (Appendix D)
Transmitter Weight	10	grams
Transmitter Battery Life	40	Hours (Approximate)
Operational Box	20 x 20 x 20	cm
Minimum Distance From Receiver	20	cm

Table 17: System Specifications

6.3 Unimplemented Design Features

Two critical design features of this device were not implemented. The attempted implementation of the -5 V rail introduced an unacceptable amount of noise into the signal conditioning and pulse shaping areas of the printed circuit board. The UART communication between dsPIC and usbPIC was also not fully implemented. This cripples the functionality of the mouse to rely on the PC to read transmitted 3D coordinates.

Future revisions of the project could include a power system fix that could be easily implemented. The source of noise generated by the current -5 V rail design is the unshielded inductor of the buck converter. This inductor propagates the switching signal into free space, and into the +5 V power line that run close to the component traces to the chip. This issue may be resolved by using an inductor-less design. Charge pumps for this purpose exist, and many configurations require less components than the current design. A pi filter on the output of the -5 V power block may also decrease the effects of any switching noise associated with power block operation. This fix has been implemented in the lab with success in suppressing noise on the -5 V power line. The +5 V and ground lines were also affected by the -5 V power block, so an entire redesign of the -5 V power system is recommended.

The UART Communication will also be fixed in a future revision of the project. The problems associated with UART Communication have been identified as synchronization errors between the dsPIC and usbPIC. The PC first must recognize the usbPIC as an HID mouse, before data that is transmitted by the dsPIC can be accepted into the usbPIC. The dsPIC currently sends coordinate data to the usbPIC as soon as it is available. The usbPIC's purpose is to scale the data and transmit it to the PC as HID mouse coordinates. The synchronization software that controls the flow of communication between dsPIC and usbPIC must be resolved. Without this functionality, serial communication may be read from the dsPIC UART communication transmit line. Although the device does not meet its final goals as an end-to-end computer mouse we believe the challenge of the project, ultrasonic tracking, was fully demonstrated in MATLAB. It is no significant challenge to implement the mouse itself, it would simply require more time. More time in this project was spent perfecting the detection algorithm for the incoming signal.

6.4 Future Device Upgrades

Future upgrades to the design include integration of the receiver array into the monitor, increased transmission ranges, mouse gesture recognition, and a relative position mode. While the design demonstrates accurate 3D position functionality, and 2D functionality comparable to a standard table-top mouse, implementing these upgrades will allow the average user to seamlessly interact with cutting-edge virtual environments. This device may also revolutionize the way that computer input is defined.

Integrating the receiver array into the corners of the monitor would be a crucial feature in this device's acceptance into the consumer or professional environments. The current receiver configuration is designed to be a comfortable fit between functional range and positional error minimization. The larger the array, the further away the user needs to be to ensure that each receiver receives the transmitted pulse. This also requires the functional range to be extended to compensate for user distance. A larger configuration minimizes the amount of positional error associated with small discrepancies in receiver transducer location. A smaller configuration, while

more portable, increases this error. Incorporating the receiver transducers into the corners of the monitor would seamlessly integrate the array into a bulky piece of equipment that is standard for most current PCs. This solution places the receiver transducers in a non-optimal configuration for error within the functional box. In a precisely machined setup, this source of error would be minimized, and could allow for a smaller array. With improvements in calibration algorithms and positional accuracy in the processing aspect of the design, other receiver configurations such as the one suggested may be possible. The number of receivers may be decreased to four transducers, since this is the minimum required for the Bard algorithm. This introduces additional sources of error, but may be calibrated so that they are minimized. If calibration is thorough enough, the mouse may be marketed as a stand-alone device with an array that can be adhered to the monitor and calibrated by the user.

The transmission range of the device may be increased through the use of a higher voltage power source. The current Lithium Polymer battery that allows the transmitter to have its characteristic ring shape is limited to $+3.7$ V. This limits the transmitted signal amplitude to about 7 V peak-to-peak. A second battery in series with the original battery will allow the transmitter to send a pulse at twice its current amplitude. This would increase the operational range of the device and allow the receiver array to be larger. Implementing this upgrade would allow the user to incorporate this device into presentations, and increase the overall spatial resolution. The transmitter power modes could also be toggled to provide higher device lifetimes for situations when the high power mode is not necessary.

The transmitter could be outfitted to implement a double or triple pulse scheme for digital modulation, allowing transmission of button click signals. The modulation scheme was discussed previously in section 3.4.3. This would allow the transmitter to send click or mode data using a momentary push button connected to an available I/O port on the MSP430 transmitter microcontroller. This method would require additional processing of pulse succession on the dsPIC after it has determined a pulse peak. An example of the push button functionality is a refined-scale mode. In this mode, the on-screen cursor could enter into a state where large changes

in transmitter position result in smaller on-screen cursor changes. This would allow the user to refine the accuracy of the cursor when precision motions are required.

A versatile method to increase the functionality of the device is the implementation of a hardware-based gesture command system. This system may be implemented on the usbPIC microcontroller. The microcontroller could have a set of stored positional coordinates that represent different mouse commands. For example, a mouse click could be implemented by having the usbPIC recognize rapid forward and backward motion in the Z-axis position of the transmitter. A highlight operation could be implemented by having the usbPIC recognize rapid initial forward Z-axis motion of the transmitter, indicating a click and hold. The hold state of the cursor would be active until the transmitter is moved to its previous location on the Z-axis. This would indicate a release of the click state. With the HID standard, a maximum of 7 mouse states may be programmed and triggered on the usbPIC with different transmitter gestures.

Another way to improve the marketability of the device is to streamline its ergonomic and aesthetic design. Different colors, sizes, and shapes of the transmitter could be created to appeal to both commercial and home users. A housing could be created to cover the PCB and transducer to make the device look more streamlined and protect the circuitry. Custom batteries could be created so that they are thicker and not as wide. For durability, additional padding and an outer coating could be applied to the battery ring.

Many improvements could be made upon this mouse to increase its functionality if more time and energy could be spent on developing this project. Many of these upgrades require little time, such as the gesture-based mouse commands. Other upgrades, such as transmitted pulse digital modulation, and advanced calibration techniques may require some hardware system redesign. While this project serves only as a proof of concept for potential future endeavors in this field, these improvements will only further enhance the user's control of the human-computer interface.

6.5 Applications and The Future

The applications for this device are limitless and range from an improved interface for standard 2D programs to advanced applications specific to this device. This device could prove more ergonomic for everyday computing tasks and would certainly be valuable for applications like Google Earth that use the scroll wheel for a Z-axis. The 3D interface would be especially helpful in applications capable of using a true 3D interface, such as CAD drawing packages. As the Nintendo Wii has shown, a 3D input interface is very effective for video games and introduces a level of realism and interaction not possible with a standard 2D mouse.

It is also possible to envision some applications created specifically for this interface, because of the ability to track motion in three dimensions. One such application could be a program marketed to students studying music education. The computer could track the motion of the conductor's hand to analyze and evaluate their conducting motion. This could be used to compare their motion to desirable conducting strokes or to analyze their control over beat and dynamics. By integrating a stored database of patterns created by professional conductors, the program could even teach students how to conduct in the style of their favorite professional.

An application that could expand beyond using the three dimensions graphically is a digital musical instrument that changes based on movement of the mouse. For example, the axes of the mouse could be set to control pitch, volume and timbre, allowing the user to play the instrument with hand motions. This functionality could be implemented as a standalone application or could be designed in an existing environment, such as Max/MSP by Cycling '74, which is a graphical music programming language that is already capable of responding to mouse input.

With a new interface like this in use, third party developers could certainly design new applications that would take advantage of the intuitive connection between user input and motion on the screen. This device could pave the way for a new revolution in computer input technology.

References

[1] J.D. Bard, F.M. Ham, and W.L. Jones. An algebraic solution to the time difference of arrival equations. *Southeastcon '96. 'Bringing Together Education, Science and Technology'., Proceedings of the IEEE*, pages 313–319, 1996.

[2] ZyTrax Inc. RS-232 pinout on DB9 connector(EIA/TIA 574). Retrieved 12 January, 2007 from `http://www.zytrax.com/tech/layer_1/cables/tech_rs232.htm`, 2007.

[3] Aseem Kohli. Accelerometer mouse. Retrieved October 11, 2006 from `http://instruct1.cit.cornell.edu/courses/ee476/FinalProjects/s2005/mousewebpageKM249_AK288/index.htm`, 2005.

[4] Shirley Li, Joseph Cheng, and Matt Tanwentang. 3D wireless mouse. Retrieved October 11, 2006 from `http://web.mit.edu/6.111/www/s2005/PROJECT/Groups/2/main.html`, 2005.

[5] Logitech Inc. 3D mouse and head tracker techical reference manual. Retrieved 18 December, 2006 from `http://www.vrdepot.com/manual-tracker.pdf`, 1992.

[6] Microchip Technology Inc. MPLAB C18 compiler. Retrieved 18 December, 2006 from `http://www.microchip.com/C18/`, 2006.

[7] Microchip Technology Inc. PIC18F2455/2550/4455/4550 data sheet. Retrieved 7 December, 2006 from `http://ww1.microchip.com/downloads/en/DeviceDoc/39632c.pdf`, 2006.

[8] John Kangchun Perng, Brian Fisher, Seth Hollar, and Kristofer S. J. Pister. Acceleration sensing glove (ASG). Retrieved October 11, 2006 from `http://web.mit.edu/6.111/www/s2005/PROJECT/Groups/2/main.html`, 2005.

[9] Daniel V. Rabinkin, Richard J. Renomeron, Arthur Dahl, Joseph C. French, James L. Flanagan, and Michael H. Bianchi. A dsp implementation of source location using microphone

arrays. *The Journal of the Acoustical Society of America, Volume 99, Issue 4.*, pages 2503–2529, 1996.

[10] Murugavu Raju. Ultrasonic distance measurement with the MSP430. Retrieved December 17, 2006 from `http://www.ti.com/litv/pdf/slaa136a`, 2001.

[11] R. Raskar, P. Beardsley, J. van Baar, Y. Wang, P. Dietz, J. Lee, D. Leigh, and T. Willwacher. High precision RFID location sensing. Retrieved October 10, 2006 from `http://www.merl.com/people/raskar/Sig04/`, 2004.

[12] Craig Ross and Ricardo Goto. Proximity security system. Retrieved October 10, 2006 from `http://instruct1.cit.cornell.edu/courses/ee476/FinalProjects/s2006/cjr37/Website/index.htm`, 2006.

[13] Brian Schmalz. UBW (USB Bit Whacker) project. Retrieved 18 December, 2006 from `http://greta.dhs.org/UBW/index.html`, 2006.

[14] Thompson Inc. Gyration optical air mouse. Retrieved October 10, 2006 from `http://www.gyration.com`, 2006.

[15] USB Implementers Forum. Universal serial bus specification. Retrieved 7 December, 2006 from `http://www.usb.org/developers/docs/`, 2006.

[16] Wikipedia. Mouse (computing). Retreived 6 February, 2007 from `http://en.wikipedia.org/wiki/Computer_mouse`, 2007.

[17] Maria Wikström, Ulrika Ahnström, Johan Falk, and Peter Händel. Implementation of an acoustic location-finding system for TDOA measurements. In *Nordic Matlab Conference 2003*, October 2003.

A Transmitter Appendices

A.1 Transducer Data Sheet

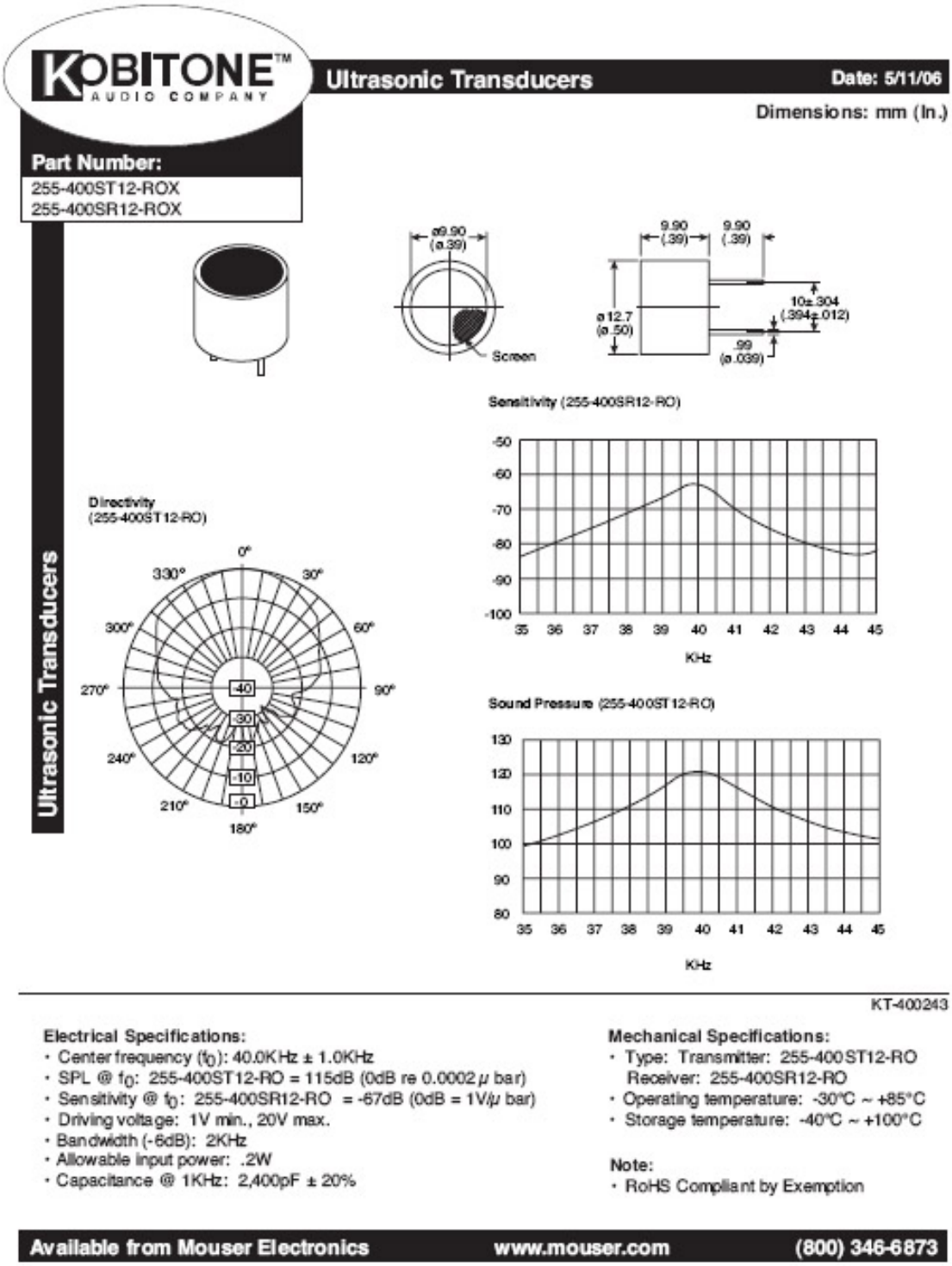


Figure 52: Transducer Data Sheet

A.2 Transducer Frequency Response

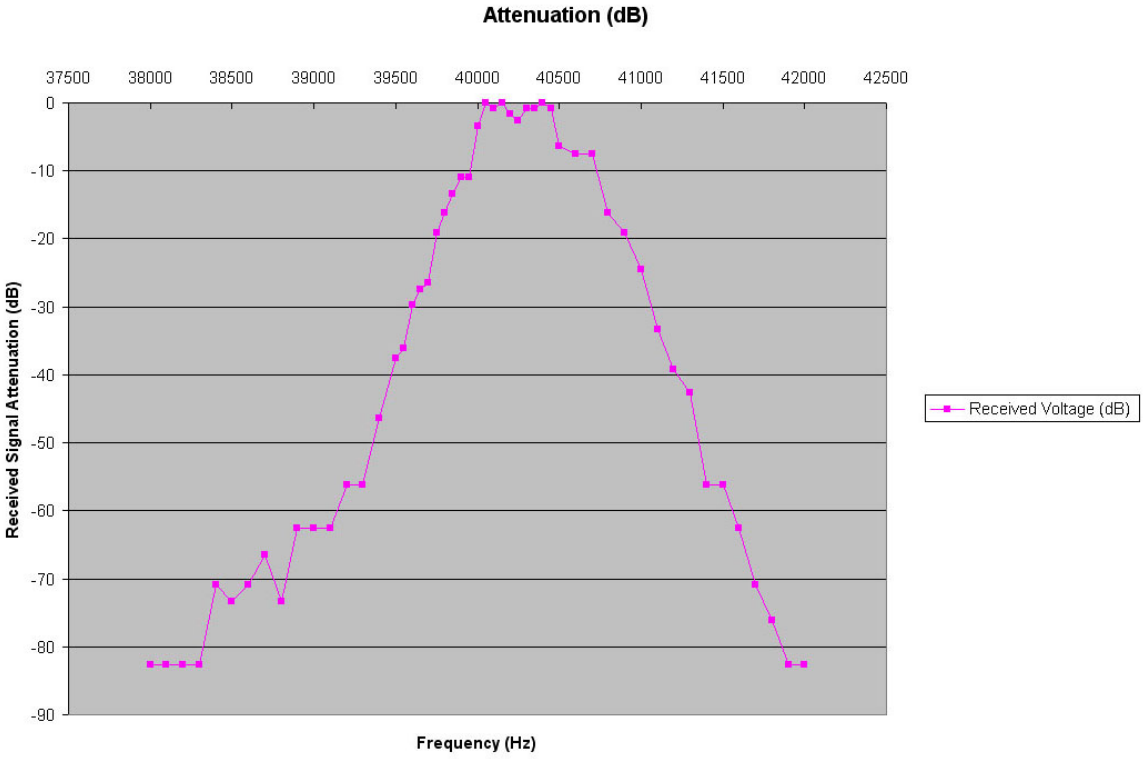


Figure 53: Transmitter Frequency Response (dB)

A.3 Schematics

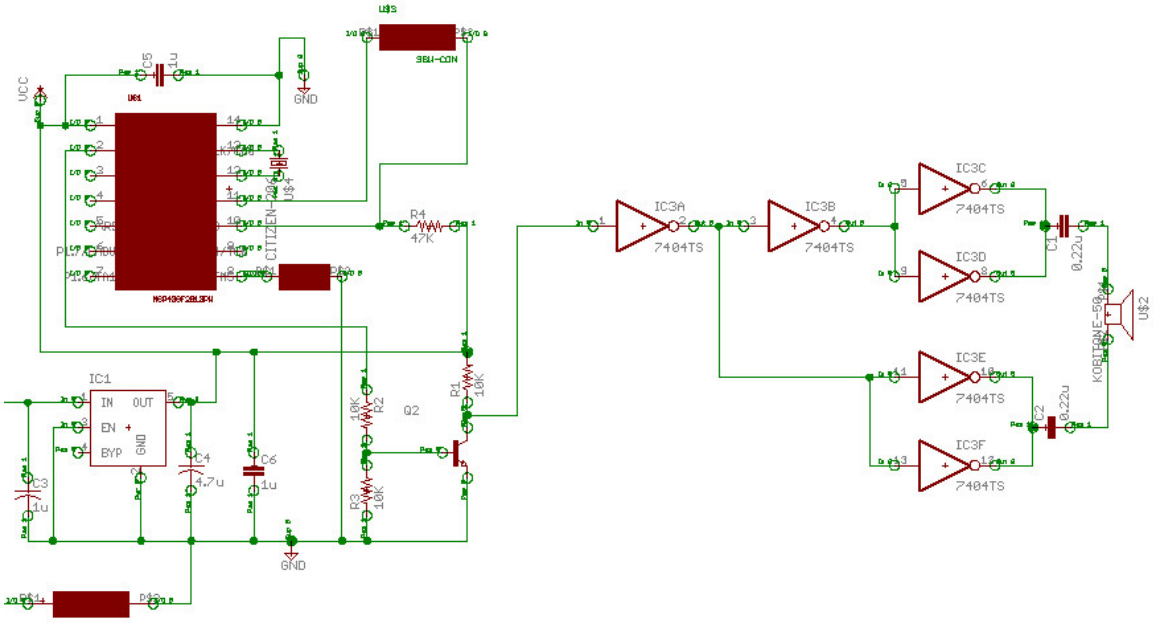


Figure 54: Transmitter Schematic

A.4 PCB Layout

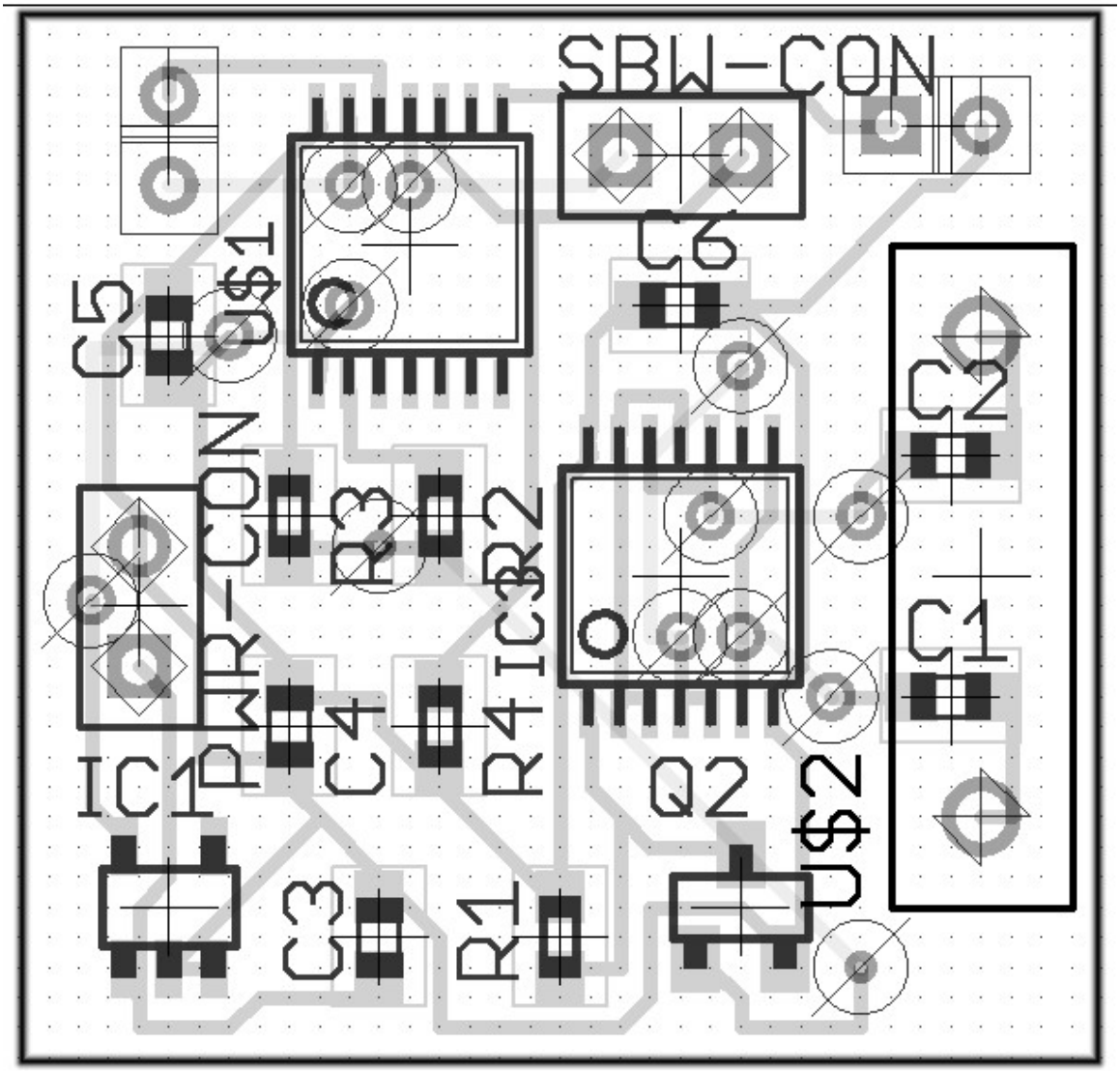


Figure 55: Transmitter PCB Layout

A.5 Firmware Code

```
;*******************************************************************************
;   Ultrasonic Transmitter
;
;   Description: This program operates MSP430 normally in LPM3, pulsing 40 kHz
;   ACLK on P1.0 for 20 cycles at 80 cycle intervals.
;   All I/O configured as low outputs to eliminate floating inputs.
;   Current consumption does increase when transmitter is powered on P1.0.
;   ACLK = LFXT1 = 40000, MCLK = SMCLK = default DCO
;   //* External 40 kHz watch crystal installed on XIN XOUT is required for ACLK *//
;
;               MSP430F20xx
;            -----------------
;        /|\|              XIN|-
;         | |                 | 40kHz
;         --|RST          XOUT|-
;           |                 |
;           |             P1.0|--> oscillator control transistor
;
;   Christian Banker
;   December 2006
;   Built with IAR Embedded Workbench Version: 3.41A
;*******************************************************************************
#include  "msp430x20x3.h"
;-------------------------------------------------------------------------------
            ORG     0F800h                  ; Program Reset
;-------------------------------------------------------------------------------
RESET       mov.w   #0280h,SP               ; Initialize stackpointer
            call    #Init_Device            ; Device initialization
                                            ;
Mainloop    bis.b   #BIT0,&P1SEL            ; Set P1.0 to transmit ACLK
            mov.w   #20,&CCR0               ; 20 cycle 40 kHz burst
            bis.w   #LPM3+GIE,SR            ; Wait for CCR0 interrupt
            bic.b   #BIT0,&P1SEL            ; Reset P1.0
            mov.w   #60,&CCR0               ; 60 cycle 40 kHz burst
            bis.w   #LPM3+GIE,SR            ; Wait for CCR0 interrupt
            jmp     Mainloop                ; Repeat
                                            ;
;-------------------------------------------------------------------------------
Init_Device;       Device Initialization
;-------------------------------------------------------------------------------
StopWDT     mov.w   #WDTPW+WDTHOLD,&WDTCTL  ; Stop WDT
SetupTA     mov.w   #TASSEL_1+ID_0+MC_1,&TACTL   ; ACLK, upmode
SetupC0     mov.w   #CCIE,&CCTL0            ; CCR0 interrupt enabled
SetupX      bis.b   #XCAP0,&BCSCTL3         ; Turn on internal load capacitors
            bis.b   #XCAP1,&BCSCTL3         ; for the XTAL to start oscillation
            call    #ClkDelay               ; Delay for oscillator to stabilize
SetupP1     mov.b   #0FFh,&P1DIR            ; All P1.x outputs
            clr.b   &P1OUT                  ; All P1.x reset

SetupP2     mov.b   #0FFh,&P2DIR            ; All P2.x outputs
            clr.b   &P2OUT                  ; All P2.x reset
;-------------------------------------------------------------------------------
ClkDelay;       Software delay
;-------------------------------------------------------------------------------
             push    #0FFFFh                ; Delay to TOS
DL1          dec.w   0(SP)                  ; Decrement TOS
             jnz     DL1                    ; Delay over?
             incd    SP                     ; Clean TOS
             ret                            ; Return from subroutine
;-------------------------------------------------------------------------------
TA0_ISR;     Common ISR for CCR1-4 and overflow
;-------------------------------------------------------------------------------
             add.w   &TAIV,PC               ; Add TA interrupt offset to PC
             jmp     CCR0_ISR               ; CCR0
             reti                           ; CCR1 - no source
             reti                           ; CCR2
```

```
CCR0_ISR     bic.w   #CCIFG,&CCTL0
             bic.w   #LPM0,0(SP)             ; Exit LPM0 on reti
             reti                            ;

;-------------------------------------------------------------------------------
WDT_ISR  ;  Exit LPM3 on reti
;-------------------------------------------------------------------------------
            bic.w   #LPM3,0(SP)             ; Clear LPM3 from TOS
            reti                            ;
                                            ;
;-------------------------------------------------------------------------------
;           Interrupt Vectors
;-------------------------------------------------------------------------------
            ORG     0FFFEh                  ; MSP430 RESET Vector
            DW      RESET                   ;
            ORG     0FFF4h                 ; WDT Vector
            DW      WDT_ISR                 ;
            ORG     0FFF2h                  ;Timer_A0 vector
            DW      TA0_ISR

            END
```

B Processing Appendices

B.1 Main

```
#include <p30f3013.h>
#include <uart.h>
#include <stdlib.h>
#include <stdio.h>
#include <math.h>
#include <string.h>
#include "h/curvefitting.h"

/* $Id: main.c 64 2007-04-20 04:49:33Z bk $ */

//Macros for Configuration Fuse Registers:
//Invoke macros to set up  device configuration fuse registers.
//The fuses will select the oscillator source, power-up timers, watch-dog
//timers, BOR characteristics etc. The macros are defined within the device
//header files. The configuration fuse registers reside in Flash memory.
_FOSC(CSW_FSCM_OFF & ECIO_PLL16 );  //Run this project using an external crystal
_FWDT(WDT_OFF);                 //Turn off the Watch-Dog Timer.
_FBORPOR(MCLR_EN & PWRT_OFF);   //Enable MCLR reset pin and turn off the
                                 //power-up timers.
_FGS(CODE_PROT_OFF);             //Disable Code Protection

/** V A R I A B L E S
******************************************************/
#define led_off() PORTDbits.RD8 = 0;
#define led_on() PORTDbits.RD8 = 1;
unsigned int tuxisgay[34];
static void initPIC(void);
extern void sample();
extern void sync();
void transmit_coords();
void init_uart();
//static void InitializeUSART(void);

float x[]={0.1,0.1,0.11,0.12,0.13,0.14,0.15,0.16,0.17,0.18,0.19,0.1},
y[]={0.1,0.1,0.1,0.1,0.1,.1,0.1,0.1,0.1,0.1,0.1,.1},
z[]={0.1,0.1,0.1,0.1,0.1,0.1,0.1,0.1,0.1,0.1,0.1,0.1};

int main(void)
{
unsigned int portb_buf;
int thisTime[5], lastTime[5];
unsigned int k;

unsigned int led_tmr = 0;
initPIC();
init_uart();
led_on();

dhat[0] = 4;
dhat[1] = 2;
dhat[2] = 2;
dhat[3] = 7;
dhat[4] = 4;

off[0]=0;
off[1]=0;
off[2]=0;
off[3]=0;
off[4]=0;

while(1)
{
```

```
off[0]=0;
off[1]=0;
off[2]=0;
off[3]=0;
off[4]=0;

sample();
if((off[0] != 1) && (off[1] != 1) && (off[2] != 1) && (off[3] != 1) && (off[4] != 1))
{
curvefit();
bard_solver();
}

transmit_coords();

}
return 0;
}

void initPIC(void)
{
TRISB = 0x03FF; //Port B - all inputs 0-9
TRISC = 0x6000; //Port C - pins 13,14 inputs, 15 output
TRISF = 0x0004; //Port F  -  4,5,6 are Mux Switches.
TRISD = 0x0200; //Port D all outputs except for RD9 = Clock
ADPCFG = 0x03FF; //set ADPCFG to 1 for TRISB input.
return;
}

void init_uart(void)
{
//provided by Cody Brenneman
U1MODEbits.USIDL = 0; //Continue during idle
U1MODEbits.ALTIO = 0; //Use U2TX and U2RX pins, not U2ATX and U2ARX
U1MODEbits.WAKE = 0; //wake-up disabled
U1MODEbits.LPBACK = 0; //Loopback mode is disabled
U1MODEbits.ABAUD = 1; //Not going to use autobaud, but it's now on the U2RX pin
U1MODEbits.PDSEL = 0; //8-bit data, no parity
U1MODEbits.STSEL = 0; //Use one stop bit

U1STAbits.UTXISEL = 1; //Interrupt when transmit buffer becomes empty
U1STAbits.UTXBRK = 0; //no break? what's a break?
U1STAbits.URXISEL = 0; //interrupt on every character recievied
U1STAbits.ADDEN = 0;  //Address detect mode disabled

// U1BRG = 97;                  //19200 for 30MIPS
U1BRG = 32;                  //57600 for 30MIPS
// U1BRG = 194; //9600 for 30MIPS

U1MODEbits.UARTEN = 1; //UART is now enabled

U1STAbits.UTXEN = 1; //Enable transmitting

return;
}


void transmit_coords() {
char x_buf[5] = {0,0,0,0,0} ; //clear memory for the UART buffer.
char y_buf[5] = {0,0,0,0,0};
char z_buf[5] = {0,0,0,0,0};
char ack_packet[2] = {0,0};
int i;
while (1) {
for (i = 0; i < 12; i++){
strncpy(x_buf,(char*) &(x[i]),4); //cast x as a char[]
strncpy(y_buf,(char*) &(y[i]),4); //cast y as a char[]
strncpy(z_buf,(char*) &(z[i]),4); //cast z as a char[]
```

```
// Calculate checksum from received values
//checksum[0] =  (x_buf[0] ^ x_buf[1] ^ x_buf[1] ^ x_buf[3]) ^
// (y_buf[0] ^ y_buf[1] ^ y_buf[2] ^ y_buf[3] ) ^
// (z_buf[0] ^ z_buf[1] ^ z_buf[2] ^ z_buf[3]) ;

putsUART1(x_buf);
putsUART1(y_buf);
putsUART1(z_buf);
led_off();
getsUART1(2,ack_packet,65534); //get the ack packet, timeout of 3ms.
led_on();
}
}
}
```

B.2 Sampling

```
#include <p30f3013.h>
#include <stdio.h>
#include <math.h>
#include "h/curvefitting.h"

//#define thresh    115
#define thresh 150

#define PORTF_MUXMASK 0x8F
#define setMUX(bits) (PORTF=(PORTF & PORTF_MUXMASK) | (bits))

void sample()
{
int sampData0 = 0;
int sampData1 = 0;
int sampData2 = 0;
int sampData3 = 0;
int sampData4 = 0;
int j = 0;
int k = 0;
int l = 0;
int m = 0;
int i = 0;
int fiveflag;
int offset = 0;
int flag0;
int flag1;
int flag2;
int flag3;
int flag4;
int pre;
int po;
fiveflag = 0;
j = 0;
k = 0;
l = 0;
m = 0;
n = 0;
i = 0;
flag0 = 0;
flag1 = 0;
flag2 = 0;
flag3 = 0;
flag4 = 0;
offset = 0;
//fortesting
setMUX(0x70);
asm("nop");
```

```
asm("nop");
asm("nop");
asm("nop");
asm("nop");
asm("nop");
asm("nop");
asm("nop");
asm("nop");
asm("nop");
asm("nop");
asm("nop");
asm("nop");
asm("nop");
asm("nop");
asm("nop");
asm("nop");
asm("nop");
asm("nop");
asm("nop");
asm("nop");
asm("nop");
asm("nop");
asm("nop");
while(PORTB > thresh/5)
{
}
setMUX(0x00);
asm("nop");
asm("nop");
asm("nop");
asm("nop");
asm("nop");
asm("nop");
asm("nop");
asm("nop");
asm("nop");
asm("nop");
asm("nop");
asm("nop");
asm("nop");
asm("nop");
asm("nop");
asm("nop");
asm("nop");
asm("nop");
asm("nop");
asm("nop");
asm("nop");
asm("nop");
asm("nop");
asm("nop");
asm("nop");
while(PORTB > thresh/5)
{
}
//end
sync();
asm("nop");
asm("nop");
asm("nop");
asm("nop");
while(fiveflag != 5)
{
offset++;

/*r0*/
sampData0 = PORTB;
setMUX(0x10);


if((sampData0 >thresh) && (j == 0))
```

```
{
flag0 = 1; //if threshold turn sampling sampling for receiver
off[0] = offset;
asm("nop");
}
else
{
if(sampData0 > thresh)
{
}
else
{
asm("nop");
asm("nop");
asm("nop");
}
asm("nop");
}

if(flag0 == 1)
{
curveData0[j] = sampData0;
j++;
}
else
{
asm("nop");
asm("nop");
asm("nop");
asm("nop");
asm("nop");
asm("nop");
}

if((j == K_DATA) && (flag0 == 1)) //Check for terminal count on receiver0
{
flag0 = 0; //if terminal turn off sampling for receiver0
fiveflag++;
}
else
{
if(j == K_DATA)
{
}
else
{
asm("nop");
asm("nop");
asm("nop");
asm("nop");
}
asm("nop");
}
//end

asm("nop");

/*r1*/
sampData1 = PORTB;
setMUX(0x30);

if((sampData1 >thresh) && (k == 0))
{
flag1 = 1; //if threshold turn sampling sampling for receiver
off[1] = offset;
asm("nop");
}
else
```

```
{
if(sampData1 > thresh)
{
}
else
{
asm("nop");
asm("nop");
asm("nop");
}
asm("nop");
}

if(flag1 == 1)
{
curveData1[k] = sampData1;
k++;
}
else
{
asm("nop");
asm("nop");
asm("nop");
asm("nop");
asm("nop");
asm("nop");
}

if((k == K_DATA) && (flag1 == 1)) //Check for terminal count on receiver0
{
flag1 = 0; //if terminal turn off sampling for receiver0
fiveflag++;
}
else
{
if(k == K_DATA)
{
}
else
{
asm("nop");
asm("nop");
asm("nop");
asm("nop");
}
asm("nop");
}
//end

asm("nop");

/*r2*/
sampData2 = PORTB;
setMUX(0x70);

if((sampData2 > thresh) && (l == 0))
{
flag2 = 1; //if threshold turn sampling sampling for receiver
off[2] = offset;
asm("nop");
}
else
{
if(sampData2 > thresh)
{
}
else
{
```

```
asm("nop");
asm("nop");
asm("nop");
}
asm("nop");
}

if(flag2 == 1)
{
curveData2[l] = sampData2;
l++;
}
else
{
asm("nop");
asm("nop");
asm("nop");
asm("nop");
asm("nop");
asm("nop");
}

if((l == K_DATA) && (flag2 == 1)) //Check for terminal count on receiver0
{
flag2 = 0; //if terminal turn off sampling for receiver0
fiveflag++;
}
else
{
if(l == K_DATA)
{
}
else
{
asm("nop");
asm("nop");
asm("nop");
asm("nop");
}
asm("nop");
}
//end

asm("nop");
asm("nop");


/*r3*/
sampData3 = PORTB;
setMUX(0x60);

if((sampData3 > thresh) && (m == 0))
{
flag3 = 1; //if threshold turn sampling sampling for receiver
off[3] = offset;
asm("nop");
}
else
{
if(sampData3 > thresh)
{
}
else
{
asm("nop");
asm("nop");
asm("nop");
}
```

```
asm("nop");
}

if(flag3 == 1)
{
curveData3[m] = sampData3;
m++;
}
else
{
asm("nop");
asm("nop");
asm("nop");
asm("nop");
asm("nop");
asm("nop");
}

if((m == K_DATA) && (flag3 == 1)) //Check for terminal count on receiver0
{
flag3 = 0; //if terminal turn off sampling for receiver0
fiveflag++;
}
else
{
if(m == K_DATA)
{
}
else
{
asm("nop");
asm("nop");
asm("nop");
asm("nop");
}
asm("nop");
}
//end

asm("nop");
asm("nop");
asm("nop");



/*r4*/

setMUX(0x00);
sampData4 = PORTB;

if((sampData4 > thresh) && (n == 0))
{
flag4 = 1; //if threshold turn sampling sampling for receiver
off[4] = offset;
asm("nop");
}
else
{
if(sampData4 > thresh)
{
}
else
{
asm("nop");
asm("nop");
asm("nop");
}
asm("nop");
```

```
}

if(flag4 == 1)
{
curveData4[n] = sampData4;
n++;
}
else
{
asm("nop");
asm("nop");
asm("nop");
asm("nop");
asm("nop");
asm("nop");
}

if((n == K_DATA) && (flag4 == 1)) //Check for terminal count on receiver0
{
flag4 = 0; //if terminal turn off sampling for receiver0
fiveflag++;
}
else
{
if(n == K_DATA)
{
}
else
{
asm("nop");
asm("nop");
asm("nop");
asm("nop");
}
asm("nop");
}
//end
}
asm("nop");
asm("nop");
asm("nop");
asm("nop");
return;
}
```

B.3 Syncing

```
.global _sync

_sync:

; clock is always at phase D for this next instruction. Rising edge will occur after this next instruction.
sync_loop:
BTSS PORTD,#9
BRA sync_loop  ; loops until we find a '1'
nop
sync_loop2:
BTSC PORTD,#9
BRA sync_loop2 ; loops until we find a '0' this is 3 instructions after the last read (hence phase is moved 1 to the left)
nop

return
```

B.4 Curve Fitting

```
#include <stdio.h>
#include <math.h>
#include "h/curvefitting.h"

volatile float        _YBSS(4) peakTime[5] = {0,0,0,0,0};
volatile int          _YBSS(2) off[5] = {0,0,0,0,0};
volatile unsigned int _YBSS(2) n = 0;
volatile float        _YBSS(4) dhat[5];
volatile int          _YBSS(2) flag = 0;

unsigned  int  _YBSS(2) curveData1[K_DATA];
unsigned  int  _YBSS(2) curveData0[K_DATA];
unsigned  int  _YBSS(2) curveData2[K_DATA];
unsigned  int  _YBSS(2) curveData3[K_DATA];
unsigned  int  _YBSS(2) curveData4[K_DATA];

float _PBSS(4) stoData[N_STORED] = {118.72,128.18,136.91,144.97,152.46,160.68,
170.25,179.11,187.32,196.13,206.46,216.27,
225.16,234.71,245.79,256.07,265.6,275.35,
286.65,297.52,307.54,317.5,329.1,340.33,
350.43,360.53,372.39,383.71,394.07,404.18,
415.7,427.18,437.48,447.24,458.57,469.82,
479.82,489.32,500.13,510.98,520.57,529.4,
539.49,549.81,558.92,567.14,576.39,585.93,
594.25,601.57,609.9,618.69,626.28,632.68,
640.08,647.76,654.34,659.83,665.93,672.65,
678.15,682.44,687.29,692.75,697.13,700.27,
703.77,708.12,711.35,713.27,715.45,718.13,
720.23,720.73,720.56,722.74,724.53,724.03,
722.88,723.59,723.88,722.98,721.31,720.42,719.49};

extern float offdum;

void curvefit(void)
{
float gain;
float c1;
float c2;
int left;
float delta;
float offdum;
float slope;
float peak[5];
float xhat[5];
int p;
int k;

//R0
p = 0;
n=0;
while(p<1)
{
if(peak[0]<curveData0[n])
{
peak[0]=curveData0[n]; //find max value
}
else
{
p++; //find location fo max value
}
n++;
}
gain = stopeak/peak[0]; //calculate gain

k = n - 1;
p = 0;
```

```
while(p < 1)
{
delta = 0;
left=floor(dhat[0]);
c1=dhat[0]-left;
c2=1-c1;
for(n=0;n<k;n++)
{
delta = delta + (gain*curveData0[n]) - ((c1*stoData[left+n+1])+(c2*stoData[left+n]));  //find error between signals
}
p++;
dhat[0] = (delta/1000) + dhat[0]; //add scaled error creating new close shift
}

//end


//R1
p = 0;
n=0;
while(p<1)
{
if(peak[1]<curveData1[n])
{
peak[1]=curveData1[n];
}
else
{
p++;
}
n++;
}
gain = stopeak/peak[1];


p = 0;
k = n - 1;
while(p < 1)
{
delta = 0;
left=floor(dhat[1]);
c1=dhat[1]-left;
c2=1-c1;
for(n=0;n<k;n++)
{
delta = delta + (gain*curveData1[n]) - ((c1*stoData[left+n+1])+(c2*stoData[left+n]));
}
p++;
dhat[1] = (delta/1000) + dhat[1];
}
//end


//R2
p = 0;
n=0;
while(p<1)
{
if(peak[2]<curveData2[n])
{
peak[2]=curveData2[n];
}
else
{
p++;
}
n++;
}
```

```
gain = stopeak/peak[2];

k = n - 1;
p = 0;
while(p < 1)
{
gain = stopeak/curveData2[79];
delta = 0;
left=floor(dhat[2]);
c1=dhat[2]-left;
c2=1-c1;
for(n=0;n<(stotime-left);n++)
{
delta = delta + (gain*curveData2[n]) - ((c1*stoData[left+n+1])+(c2*stoData[left+n]));
}
p++;
dhat[2] = (delta/1000) + dhat[2];
}
//end



//R3
p = 0;
n=0;
while(p<1)
{
if(peak[3]<curveData3[n])
{
peak[3]=curveData3[n];
}
else
{
p++;
}
n++;
}
gain = stopeak/peak[3];
k = n - 1;
p = 0;
while(p < 1)
{
gain = stopeak/curveData3[79];
delta = 0;
left=floor(dhat[3]);
c1=dhat[3]-left;
c2=1-c1;
for(n=0;n<(stotime-left);n++)
{
delta = delta + (gain*curveData3[n]) - ((c1*stoData[left+n+1])+(c2*stoData[left+n]));
}
p++;
dhat[3] = (delta/1000) + dhat[3];
}
//end



//R4
p = 0;
n=0;
while(p<1)
{
if(peak[4]<curveData4[n])
{
peak[4]=curveData4[n];
}
```

```
else
{
p++;
}
n++;
}
gain = stopeak/peak[4];
delta = 101;
k = n - 1;
while(p < 1)
{
delta = 21;
gain = stopeak/curveData4[79];
delta = 0;
left=floor(dhat[4]);
c1=dhat[4]-left;
c2=1-c1;
for(n=0;n<(stotime-left);n++)
{
delta = delta + (gain*curveData4[n]) - ((c1*stoData[left+n+1])+(c2*stoData[left+n]));
}
p++;
dhat[4] = (delta/1000) + dhat[4];
}
//end


offdum = (float)off[0];
peakTime[0] = (offdum * dt);

offdum = (float)off[1];
peakTime[1] = (offdum * dt) + 0.6782;

offdum = (float)off[2];
peakTime[2] = (offdum * dt) + 1.3563;

offdum = (float)off[3];
peakTime[3] = (offdum * dt) + 2.7127;

offdum = (float)off[4];
peakTime[4] = (offdum * dt) + 5.4253;

peakTime[0] = peakTime[0] - (dhat[0]*dt);
peakTime[1] = peakTime[1] - (dhat[1]*dt);
peakTime[2] = peakTime[2] - (dhat[2]*dt);
peakTime[3] = peakTime[3] - (dhat[3]*dt);
peakTime[4] = peakTime[4] - (dhat[4]*dt);

peakTime[1] = peakTime[1] - peakTime[0];
peakTime[2] = peakTime[2] - peakTime[0];
peakTime[3] = peakTime[3] - peakTime[0];
peakTime[4] = peakTime[4] - peakTime[0];

off[1] = off[1] - off[0];
off[2] = off[2] - off[0];
off[3] = off[3] - off[0];
off[4] = off[4] - off[0];

return;
}
```

B.5 Bard Solver

```
function [x_hat, x_hat2] = bard_solver(A,delta_t,c)
% bard_solver Estimates event location given sensor geometry and TDOA info.
%
% Bard Positioning Solution
%  Algorithm based on mathematics in the paper, "An Algebraic Solution to
%  the Time Difference of Arrival Equations" by John D. Bard, Fredric M.
%  Ham, and W. Linwood Jones.
%   Over Determined Case: >=N+2 Receivers for N Dimensions.
%
% Inputs:
%   A: rows of sensor/receiver [x,y,{z}] coordinates
%   delta_t: column vector of TDOAs, delta_t(1) = t1 - t1
%   c: speed of light (units: m/s), scalar.
% Outputs:
%   x_hat: row vector of [x,y,z] estimated transmitter position. (m)
%   x_hat2: extraneous solution in over-determined case of >=N+2 sensors,
%       possibly correct solution in under or minimally determined case
%       of < N+2 sensors. Units: Meters.
%
% Author: Benjamin Woodacre 7/14/2003

% If more than one TDOA set is passed, call vectorized solver.
if size(delta_t,2) > 1
    [x_hat, x_hat2] = vector_bard_solver(A,delta_t,c);
    return;
end

c_bar = c.*delta_t;
m = 0.5*(diag(A(:,:,1)*A(:,:,1).') - (c.^2)*diag(delta_t*delta_t.'));
if size(A,3) >= 2
    pinvA = A(:,:,2).';
    % A(:,:,2) = [];
else
    pinvA = pinv(A); % Calculate once. Use four times.
end
phi = pinvA.'*pinvA;

%Quadratic solution
aa = (c_bar.'*phi*c_bar - 1);
bb = -(2*m.'*phi*c_bar);
cc = (m.'*phi*m);
root = sqrt(bb^2-4*aa*cc);
sL = ((-bb + [root -root])/(2*aa));
sL = real(sL);
m=m(:,ones(1,size(sL,2)));
xL  = pinvA*(m-c_bar*sL);
if sL(1)<0 && sL(2)>0
    sL(1) = sL(2);
elseif sL(1)>0 && sL(2)<0
    sL(2) = sL(1);
end

sumr = sum(abs(A(:,:,1)*xL - m+c_bar*sL));
[temp,minL] = min(sumr);

if minL == 1
    x_hat = xL(:,1);
    x_hat2 = xL(:,2);
else
    x_hat = xL(:,2);
    x_hat2 = xL(:,1);
end
% disp('Bard Solver done!');
```

B.6 dsPIC's TDOA Calculations

```
#include <math.h>
#include <stdio.h>

extern unsigned int peakTime[5];
extern float cord_out[3];

void bard_solver()
{
float a_prime[3][5];
float pinv[3][5];
float phi[5][5];
float s1[5][2];
float s2[5][2];
float xl[3][2];
float c = 340;
float delta_t[5];
float delta_c[5];
float A[5][3];
float m[5][2];
float A_bar[5];
int n,l,p;
float aa_sum[5];
float bb_sum[5];
float cc_sum[5];
float root;
float sL[2];
float c_bar[5];
float c_bar2[5];
float Delta_c[5];
float A_diag[5];
float aa = 0;
float bb = 0;
float cc = 0;

delta_t[0] = (float)peakTime[0]/30;
delta_t[1] = (float)peakTime[1]/30;
delta_t[2] = (float)peakTime[2]/30;
delta_t[3] = (float)peakTime[3]/30;
delta_t[4] = (float)peakTime[4]/30;

delta_t[1] = delta_t[1] - delta_t[0];
delta_t[2] = delta_t[2] - delta_t[0];
delta_t[3] = delta_t[3] - delta_t[0];
delta_t[4] = delta_t[4] - delta_t[0];
delta_t[0] = 0;

printf("%04d,%04d,%04d,%04d,%04d\r\n",delta_t[0],delta_t[1],delta_t[2],delta_t[3],delta_t[4]);


c_bar[0] = c * delta_t[0] / 1000000;
c_bar[1] = c * delta_t[1] / 1000000;
c_bar[2] = c * delta_t[2] / 1000000;
c_bar[3] = c * delta_t[3] / 1000000;
c_bar[4] = c * delta_t[4] / 1000000;

delta_c[0] = c_bar[0] * c_bar[0];
delta_c[1] = c_bar[1] * c_bar[1];
delta_c[2] = c_bar[2] * c_bar[2];
delta_c[3] = c_bar[3] * c_bar[3];
delta_c[4] = c_bar[4] * c_bar[4];

A_diag[0] = 0;
A_diag[1] = .0636;
A_diag[2] = .0227;
A_diag[3] = .0639;
A_diag[4] = .0626;
```

```
A[0][0] = 0;
A[0][1] = 0;
A[0][2] = 0;

A[1][0] = 0;
A[1][1] = -.1763;
A[1][2] = .1802;

A[2][0] = .0891;
A[2][1] = -.0958;
A[2][2] = .0744;

A[3][0] = .1811;
A[3][1] = -.1763;
A[3][2] = 0;

A[4][0] = .183;
A[4][1] = 0;
A[4][2] = .1706;

pinv[0][0]=0;
pinv[0][1]=-2.8115;
pinv[0][2]=.5921;
pinv[0][3]=2.4897;
pinv[0][4]=2.7113;
pinv[1][0]=0;
pinv[1][1]=-2.5397;
pinv[1][2]=-.8551;
pinv[1][3]=-2.6667;
pinv[1][4]=3.0551;
pinv[2][0]=0;
pinv[2][1]=2.7993;
pinv[2][2]=.2787;
pinv[2][3]=-2.9508;
pinv[2][4]=2.7842;

phi[0][0]=0;
phi[0][1]=0;
phi[0][2]=0;
phi[0][3]=0;
phi[0][4]=0;
phi[1][0]=0;
phi[1][1]=22.1905;
phi[1][2]=1.12871;
phi[1][3]=-8.4873;
phi[1][4]=-7.5881;
phi[2][0]=0;
phi[2][1]=1.2871;
phi[2][2]=1.1593;
phi[2][3]=2.9319;
phi[2][4]=-.2311;
phi[3][0]=0;
phi[3][1]=-8.4873;
phi[3][2]=2.9319;
phi[3][3]=22.0169;
phi[3][4]=-9.6121;
phi[4][0]=0;
phi[4][1]=-7.5881;
phi[4][2]=-.2311;
phi[4][3]=-9.6121;
phi[4][4]=24.4366;

m[0][0] = .5 * (A_diag[0] - delta_c[0]);
m[1][0] = .5 * (A_diag[1] - delta_c[1]);
m[2][0] = .5 * (A_diag[2] - delta_c[2]);
m[3][0] = .5 * (A_diag[3] - delta_c[3]);
m[4][0] = .5 * (A_diag[4] - delta_c[4]);
```

```
aa_sum[0] = 0;
aa_sum[1] = c_bar[1] * phi[1][1] + c_bar[2] * phi[2][1] + c_bar[3] * phi[3][1] + c_bar[4] * phi[4][1];
aa_sum[2] = c_bar[1] * phi[1][2] + c_bar[2] * phi[2][2] + c_bar[3] * phi[3][2] + c_bar[4] * phi[4][2];
aa_sum[3] = c_bar[1] * phi[1][3] + c_bar[2] * phi[2][3] + c_bar[3] * phi[3][3] + c_bar[4] * phi[4][3];
aa_sum[4] = c_bar[1] * phi[1][4] + c_bar[2] * phi[2][4] + c_bar[3] * phi[3][4] + c_bar[4] * phi[4][4];

aa = aa_sum[0] * c_bar[0] + aa_sum[1] * c_bar[1] + aa_sum[2] * c_bar[2] + aa_sum[3] * c_bar[3] + aa_sum[4] * c_bar[4] - 1;

bb_sum[0] = 0;
bb_sum[1] = m[1][0] * phi[1][1] + m[2][0] * phi[2][1] + m[3][0] * phi[3][1] + m[4][0] * phi[4][1];
bb_sum[2] = m[1][0] * phi[1][2] + m[2][0] * phi[2][2] + m[3][0] * phi[3][2] + m[4][0] * phi[4][2];
bb_sum[3] = m[1][0] * phi[1][3] + m[2][0] * phi[2][3] + m[3][0] * phi[3][3] + m[4][0] * phi[4][3];
bb_sum[4] = m[1][0] * phi[1][4] + m[2][0] * phi[2][4] + m[3][0] * phi[3][4] + m[4][0] * phi[4][4];

bb = bb_sum[0]*c_bar[0] + bb_sum[1]*c_bar[1] + bb_sum[2]*c_bar[2] + bb_sum[3]*c_bar[3] + bb_sum[4]*c_bar[4];

bb = bb * -2;

cc = bb_sum[0]*m[0][0] + bb_sum[1]*m[1][0] + bb_sum[2]*m[2][0] + bb_sum[3]*m[3][0] + bb_sum[4]*m[4][0];


root = sqrtf(bb * bb - 4 * aa * cc);

if(root > 0)
{
root = root;
}
else
{
root = 0;
}

sL[0] = ((-bb + root)/(2 * aa));
sL[1] = ((-bb + -root)/(2 * aa));

m[0][1] = m[0][0];
m[1][1] = m[1][0];
m[2][1] = m[2][0];
m[3][1] = m[3][0];
m[4][1] = m[4][0];

s1[0][0] = sL[0] * c_bar[0];
s1[1][0] = sL[0] * c_bar[1];
s1[2][0] = sL[0] * c_bar[2];
s1[3][0] = sL[0] * c_bar[3];
s1[4][0] = sL[0] * c_bar[4];

s1[0][1] = sL[1] * c_bar[0];
s1[1][1] = sL[1] * c_bar[1];
s1[2][1] = sL[1] * c_bar[2];
s1[3][1] = sL[1] * c_bar[3];
s1[4][1] = sL[1] * c_bar[4];

s2[0][0] = m[0][0] - s1[0][0];
s2[1][0] = m[1][0] - s1[1][0];
s2[2][0] = m[2][0] - s1[2][0];
s2[3][0] = m[3][0] - s1[3][0];
s2[4][0] = m[4][0] - s1[4][0];

s2[0][1] = m[0][1] - s1[0][1];
s2[1][1] = m[1][1] - s1[1][1];
s2[2][1] = m[2][1] - s1[2][1];
s2[3][1] = m[3][1] - s1[3][1];
s2[4][1] = m[4][1] - s1[4][1];

xl[0][0] = pinv[0][0]*s2[0][0] + pinv[0][1]*s2[1][0] + pinv[0][2]*s2[2][0] + pinv[0][3]*s2[3][0] + pinv[0][4]*s2[4][0];
xl[1][0] = pinv[1][0]*s2[0][0] + pinv[1][1]*s2[1][0] + pinv[1][2]*s2[2][0] + pinv[1][3]*s2[3][0] + pinv[1][4]*s2[4][0];
```

```
xl[2][0] = pinv[2][0]*s2[0][0] + pinv[2][1]*s2[1][0] + pinv[2][2]*s2[2][0] + pinv[2][3]*s2[3][0] + pinv[2][4]*s2[4][0];

xl[0][1] = pinv[0][0]*s2[0][1] + pinv[0][1]*s2[1][1] + pinv[0][2]*s2[2][1] + pinv[0][3]*s2[3][1] + pinv[0][4]*s2[4][1];
xl[1][1] = pinv[1][0]*s2[0][1] + pinv[1][1]*s2[1][1] + pinv[1][2]*s2[2][1] + pinv[1][3]*s2[3][1] + pinv[1][4]*s2[4][1];
xl[2][1] = pinv[2][0]*s2[0][1] + pinv[2][1]*s2[2][1] + pinv[2][2]*s2[2][1] + pinv[2][3]*s2[3][1] + pinv[2][4]*s2[4][1];

cord_out[0] = xl[0][0];
cord_out[1] = xl[1][0];
cord_out[2] = xl[2][0];

if(xl[1][0] < 0)
{
cord_out[0] = xl[0][1];
cord_out[1] = xl[1][1];
cord_out[2] = xl[2][1];
}

}
\end
```

B.7 usbPIC Code

```
static void InitializeSystem(void);
static void InitializeUSART(void);
void rx_handler(void);
void USBTasks(void);
signed char x_delta = 0, y_delta = 0; //Two byte relative movement vectors
float x_prev = 0,y_prev = 0,z_prev = 0;
/** V E C T O R  R E M A P P I N G *****************************************/
/*
extern void _startup (void);        // See c018i.c in your C18 compiler dir
#pragma code _RESET_INTERRUPT_VECTOR = 0x000800
void _reset (void)
{
    _asm goto _startup _endasm
}
#pragma code

*/

#pragma code rx_interrupt = 0x8
void rx_int (void)
{
if (PIR1bits.RCIF){  //was it the receive interrupt?
_asm goto rx_handler _endasm
}
}

#pragma interrupt rx_handler
void rx_handler (void)
{
#define CORD_MUL  640 //Sensitivity of the mouse (256 levels/20cm)
#define CLK_THRSHLD 0.1 //Movement threshold for the click gesture
char input_buf[15]; //input buffer from the USART
char str_idx = 0; //will index the strings (1-30)
char x_buf[5] = {0,0,0,0,0},y_buf[5]= {0,0,0,0,0},z_buf[5]= {0,0,0,0,0}; //coordinates are floats
char checksum = 0, lcl_checksum = 0; //local and found checksums
float x = 0,y = 0,z = 0;

char trash; //offset data input by one.
led1_off();

//cast x,y,z as character buffers and read 4 bytes (float length).

// getsUSART(trash,1);
getsUSART(x_buf,4);
getsUSART(y_buf,4);
getsUSART(z_buf,4);
putrsUSART("ok");

//calculate local checksum
// lcl_checksum =  (x_buf[0] ^ x_buf[1] ^ x_buf[1] ^ x_buf[3]) ^
// (y_buf[0] ^ y_buf[1] ^ y_buf[2] ^ y_buf[3] ) ^
// (z_buf[0] ^ z_buf[1] ^ z_buf[2] ^ z_buf[3]) ;
  //compare.  if the checksums dont match, throw this shit out and wait again
//if (lcl_checksum != checksum) return;

x = *((float*) x_buf);
y = *((float*) y_buf);
z = *((float*) z_buf);


if (x == 0.1 && y == 0.12 && z == 0.1)  led0_off();

if (x_prev == 0 &&  y_prev == 0 && z_prev == 0)
{x_prev = x; y_prev = y; z_prev = z;};

// find difference in position and magnify it.
```

```
x_delta = (char) (CORD_MUL * ( x - x_prev ));
y_delta = (char) (CORD_MUL * ( y - y_prev ));

//set previous values to current values.
x_prev = x;
y_prev = y;
//led1_on();
/* Clear the interrupt flag */
PIR1bits.RCIF = 0;
INTCONbits.GIEH = 1;  //enable interrupts again.
}
#pragma code
/******************************************************************************
 * Function:        void main(void)
 *****************************************************************************/
void main(void)
{
InitializeSystem();
InitializeUSART();
led1_on();
led0_on();
// putrsUSART("USB Magicmouse WPI Major Qualifying Project\r\n");
// putrsUSART("Mike Cretella 2007\r\n");
// putrsUSART("Diagnostics Active...\r\n\r\n");
    while(1)
    {
INTCONbits.GIEH = 0;
       USBTasks();         // USB Tasks
INTCONbits.GIEH = 1;
ProcessIO();        // See user\user.c & .h
    }//end while
}//end main

/******************************************************************************
 * Function:        static void InitializeSystem(void)
 *****************************************************************************/
static void InitializeSystem(void)
{

    TRISB = 0x00; //set portb to all output.
    TRISC = 0x80; //set portc to outputs, Rx input
    ADCON1 |= 0x0F;                 // Default all pins to digital

    #if defined(USE_USB_BUS_SENSE_IO)
    tris_usb_bus_sense = INPUT_PIN; // See io_cfg.h
    #endif

    #if defined(USE_SELF_POWER_SENSE_IO)
    tris_self_power = INPUT_PIN;
    #endif

    mInitializeUSBDriver();         // See usbdrv.h

    UserInit();                     // See user.c & .h

}//end InitializeSystem


static void InitializeUSART(void)
{
OpenUSART( USART_TX_INT_OFF &
USART_RX_INT_ON &
USART_ASYNCH_MODE &
USART_EIGHT_BIT &
USART_CONT_RX &
USART_BRGH_HIGH,
624 );
baudUSART(BAUD_16_BIT_RATE & BAUD_WAKEUP_OFF & BAUD_AUTO_OFF);
```

```
/* Enable interrupt priority */
RCONbits.IPEN = 1;
/* Make receive interrupt high priority */
IPR1bits.RCIP = 1;
/* Enable all high priority interrupts */
INTCONbits.GIEH = 1;
/* Enable receive as interrupt */
PIE1bits.RCIE = 1;
}

/********************************************************************************
 * Function:        void USBTasks(void)
 ********************************************************************************/
void USBTasks(void)
{
    /*
     * Servicing Hardware
     */
    USBCheckBusStatus();                    // Must use polling method
    if(UCFGbits.UTEYE!=1)
        USBDriverService();                 // Interrupt or polling method

}// end USBTasks
```

B.8 Simulators

B.8.1 Single error Ouput

```
function[matrix_array,cord_set] = proccessingblock_rev4(rec_dist,Res,rate)

x = randint(1,Res,[0,0]);

rec_dist_OOPS = rec_dist;
%rec_dist_OOPS(2,3) = rec_dist_OOPS(2,3) + 500e-6;
%rec_dist_OOPS(3,3) = rec_dist_OOPS(3,3) + 500e-6;
%rec_dist_OOPS(4,2) = rec_dist_OOPS(4,2) + 500e-6;
%rec$_dist_OOPS(5,1) = rec_dist_OOPS(5,1) + 500e-6;

for n = 1:Res
    Rx = (rand - 0.5) * 0.2;
    Ry = (rand - 0.5) * 0.2 + 0.2;
    Rz = (rand - 0.5) * 0.2 + 0.1;
    R = [Rx Ry Rz];
%   R = randint(1,3,[.19e6,.38e6])/1e6;

    P = repmat(R,5,1);
    d = sqrt(sum((P - rec_dist).^2,2));

    delta_d(1) = d(1) - d(1);
    delta_d(2) = d(1) - d(2);
    delta_d(3) = d(1) - d(3);
    delta_d(4) = d(1) - d(4);
    delta_d(5) = d(1) - d(5);

    T = delta_d / 340;

    T_col = ceil(T / rate)';
    T_col = T_col * rate;

%    [est,est2] = Bard_solver(rec_dist(1:4,:),T_col(1:4),340);
    [est,est2] = Bard_solver(rec_dist_OOPS,T_col,340);

    %if (est(1) < -.1 || est(1) > .1) && (est(2) < .2 || est(2)> .4) && (est(3) < 0 || est(3) > .2)
    %    error = sqrt(sum((R-est2').^2));
    %else
    %    error = sqrt(sum((R-est').^2));
    %end

    error = sqrt(sum((R-est2').^2));
    error2 = sqrt(sum((R-est').^2));

    cord_set(1,n) = est(1);
    cord_set(2,n) = est(2);
    cord_set(3,n) = est(3);

%x(n) = error
    x(n) = min(error,error2)*10e3;
end

matrix_array = x;
```

B.8.2 Axial error Ouput

```
function[matrix_array1,matrix_array2,matrix_array3,cord_set] = proccessingblock_rev4_XYZ(rec_dist,Res,rate)

x = randint(1,Res,[0,0]);

rec_dist_OOPS = rec_dist;
%rec_dist_OOPS(2,3) = rec_dist_OOPS(2,3) + 500e-6;
%rec_dist_OOPS(3,3) = rec_dist_OOPS(3,3) + 500e-6;
%rec_dist_OOPS(4,2) = rec_dist_OOPS(4,2) + 500e-6;
%rec$_dist_OOPS(5,1) = rec_dist_OOPS(5,1) + 500e-6;

for n = 1:Res
    Rx = (rand - 0.5) * 0.2;
    Ry = (rand - 0.5) * 0.2 + 0.2;
    Rz = (rand - 0.5) * 0.2 + 0.1;
    R = [Rx Ry Rz];
%   R = randint(1,3,[.19e6,.38e6])/1e6;

    P = repmat(R,5,1);
    d = sqrt(sum((P - rec_dist).^2,2));


    delta_d(1) = d(1) - d(1);
    delta_d(2) = d(1) - d(2);
    delta_d(3) = d(1) - d(3);
    delta_d(4) = d(1) - d(4);
    delta_d(5) = d(1) - d(5);
%   delta_d(6) = d(1) - d(6);
%   delta_d(7) = d(1) - d(7);
%   delta_d(8) = d(1) - d(8);

    T = delta_d / 340;

    T_col = ceil(T / rate)';
    T_col = T_col * rate;


%   [est,est2] = Bard_solver(rec_dist(1:4,:),T_col(1:4),340);
    [est,est2] = Bard_solver(rec_dist_OOPS,T_col,340);

    %if (est(1) < -.1 || est(1) > .1) && (est(2) < .2 || est(2)> .4) && (est(3) < 0 || est(3) > .2)
    %    error = sqrt(sum((R-est2').^2));
    %else
    %    error = sqrt(sum((R-est').^2));
    %end

    error11 = sqrt(sum((R(1)-est2(1)').^2));
    error12 = sqrt(sum((R(1)-est(1)').^2));
    error21 = sqrt(sum((R(2)-est2(2)').^2));
    error22 = sqrt(sum((R(2)-est(2)').^2));
    error31 = sqrt(sum((R(3)-est2(3)').^2));
    error32 = sqrt(sum((R(3)-est(3)').^2));



    cord_set(1,n) = est(1);
    cord_set(2,n) = est(2);
    cord_set(3,n) = est(3);


%x(n) = error
    x1(n) = min(error11,error12)*10e3;
    x2(n) = min(error21,error22)*10e3;
    x3(n) = min(error31,error32)*10e3;
end

matrix_array1 = x1;
```

```
matrix_array2 = x2;
matrix_array3 = x3;
```

B.9 RS232 MATLAB Calculations

```
var1 = 0;
B=[0,0,0];

%stay in infinite loop
while (var1 = 0){

% Opens a serial communication object
m1=serial('COM4','BaudRate', 9600,'Parity', 'none','DataBits',8,'StopBits', 1);
set(m1,'InputBufferSize',1024); % ini=512
fopen(m1)
m1.ReadAsyncMode = 'continuous';
readasync(m1);

% reads an value from microcontroller
v=fread(m1,[1,5],'uchar');

fclose(m1);

%current measured receiver locations
A = [0,0,0;.1375,0,0;.1375,.15,0;0,.15,0;.073,.073,.005];

bard_solver(A,v,340);

%concatenate coordinate onto C, extract X, Y, and Z values, plot
C = cat(1,B,x_hat);
X = C( : , 1);
Y = C( : , 2);
Z = C( : , 3);
plot3(X,Y,Z)

end
```

C Receiver Construction

C.1 Power Charger Schematic

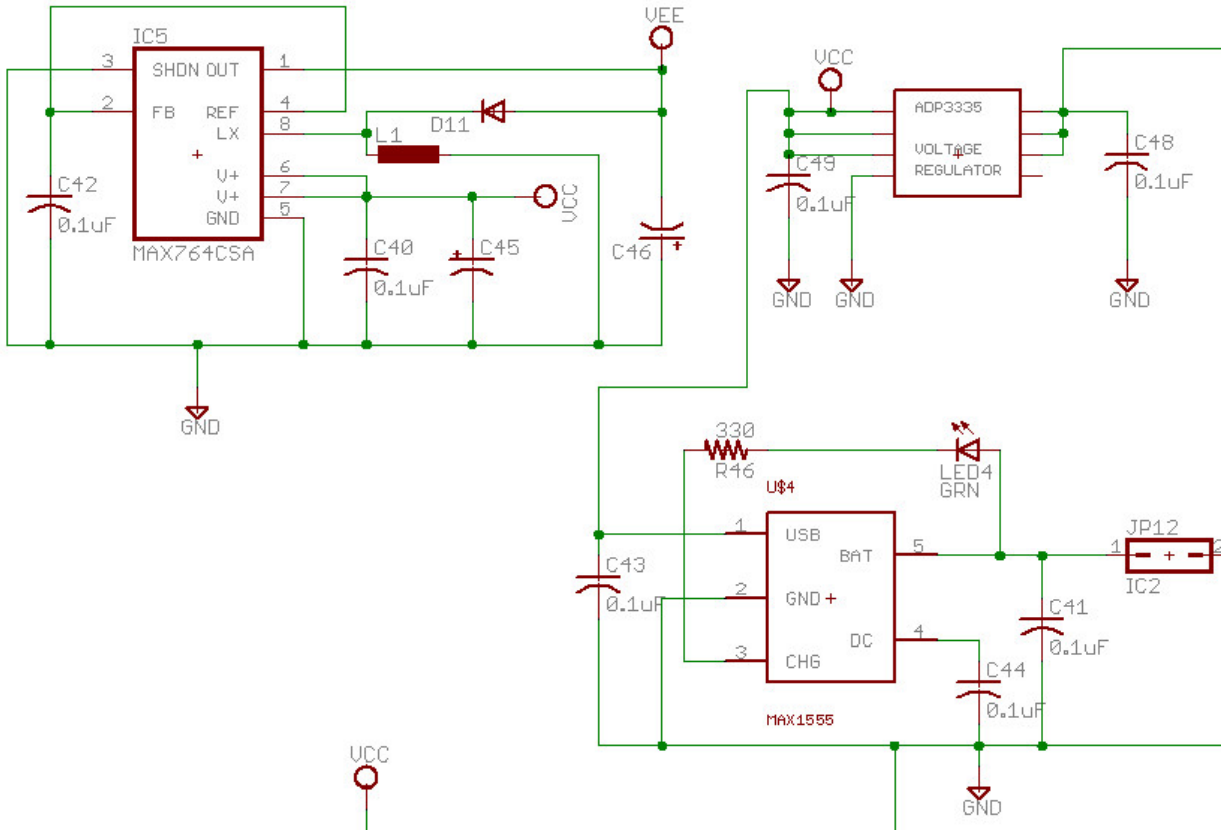


Figure 56: Power Connection and Charging Unit

C.2 Analog Schematic

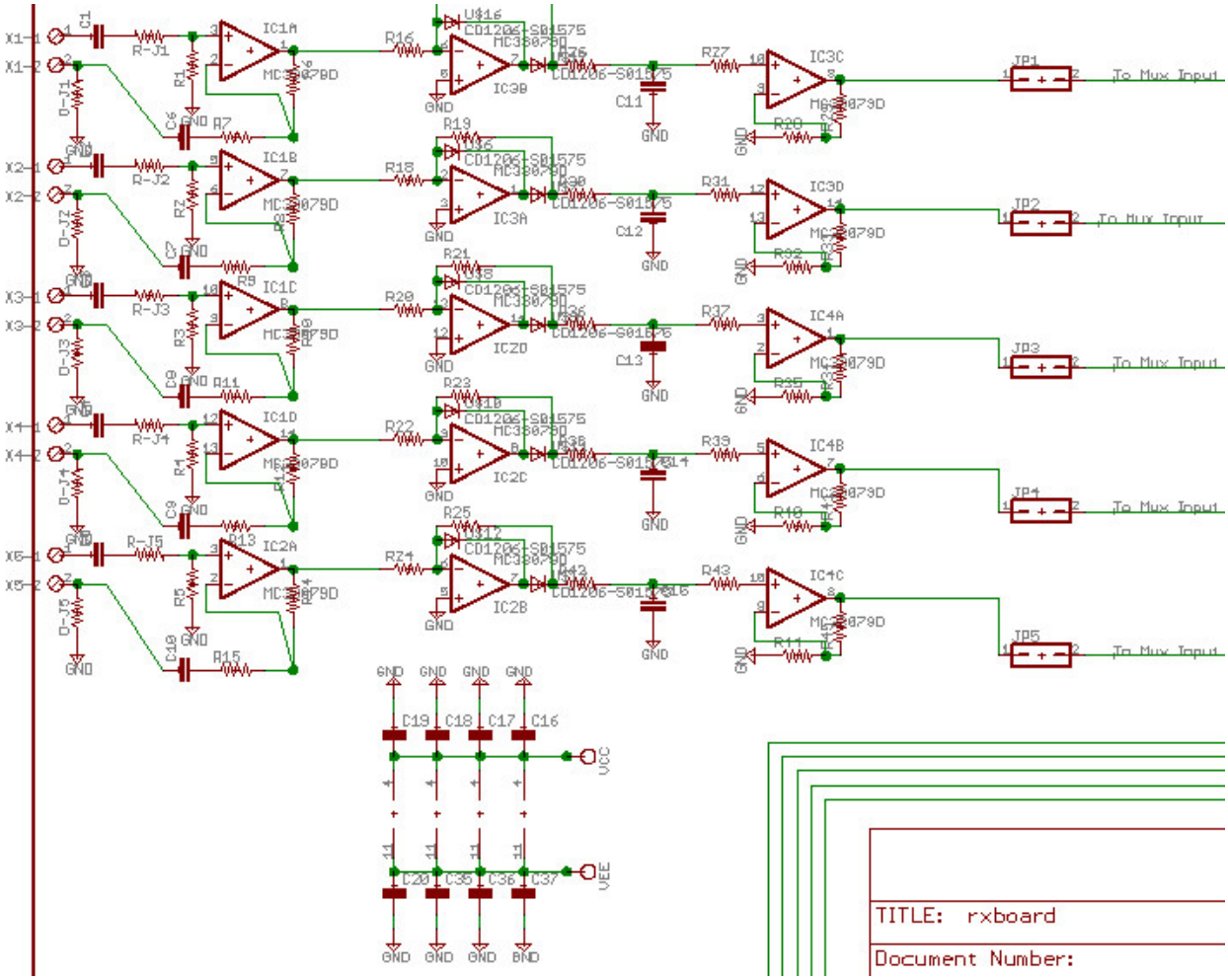


Figure 57: Analog Filtering

C.3 dsPIC Schematic

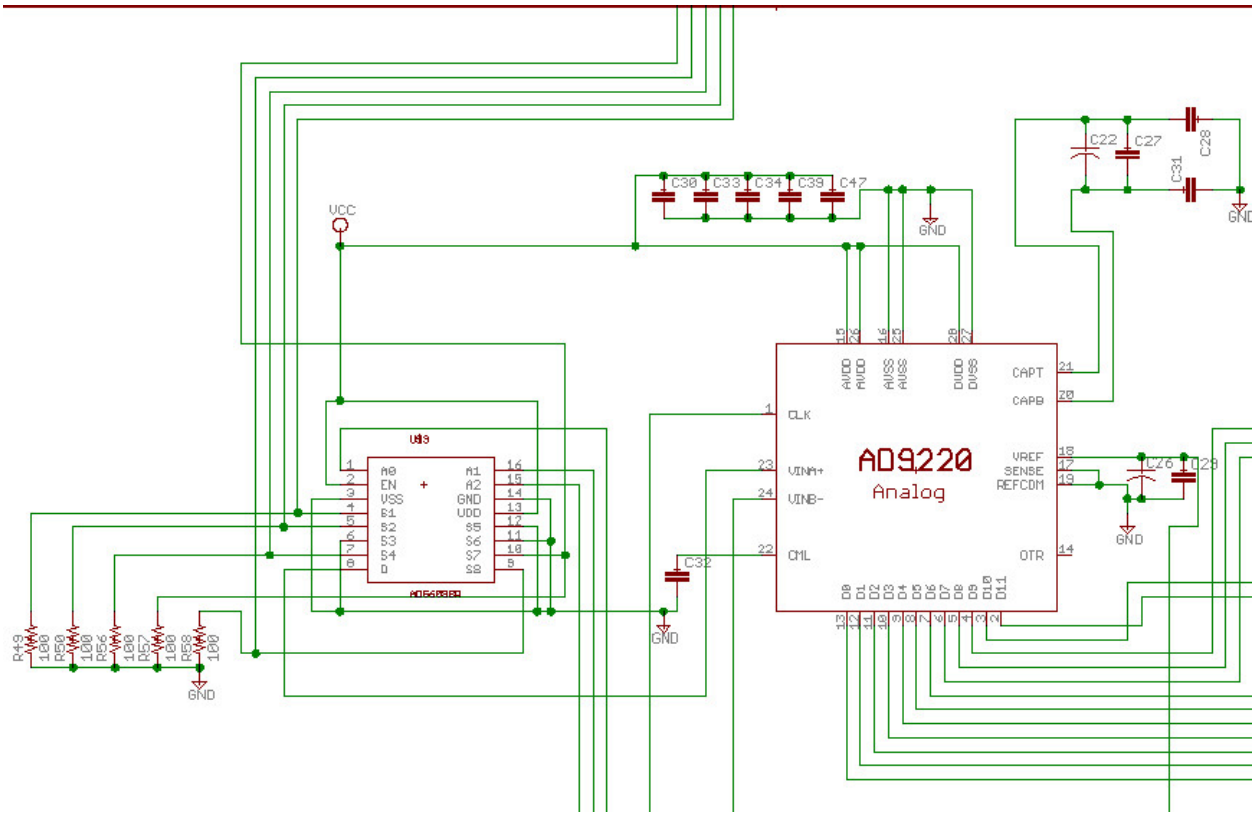


Figure 58: ADC Schematic

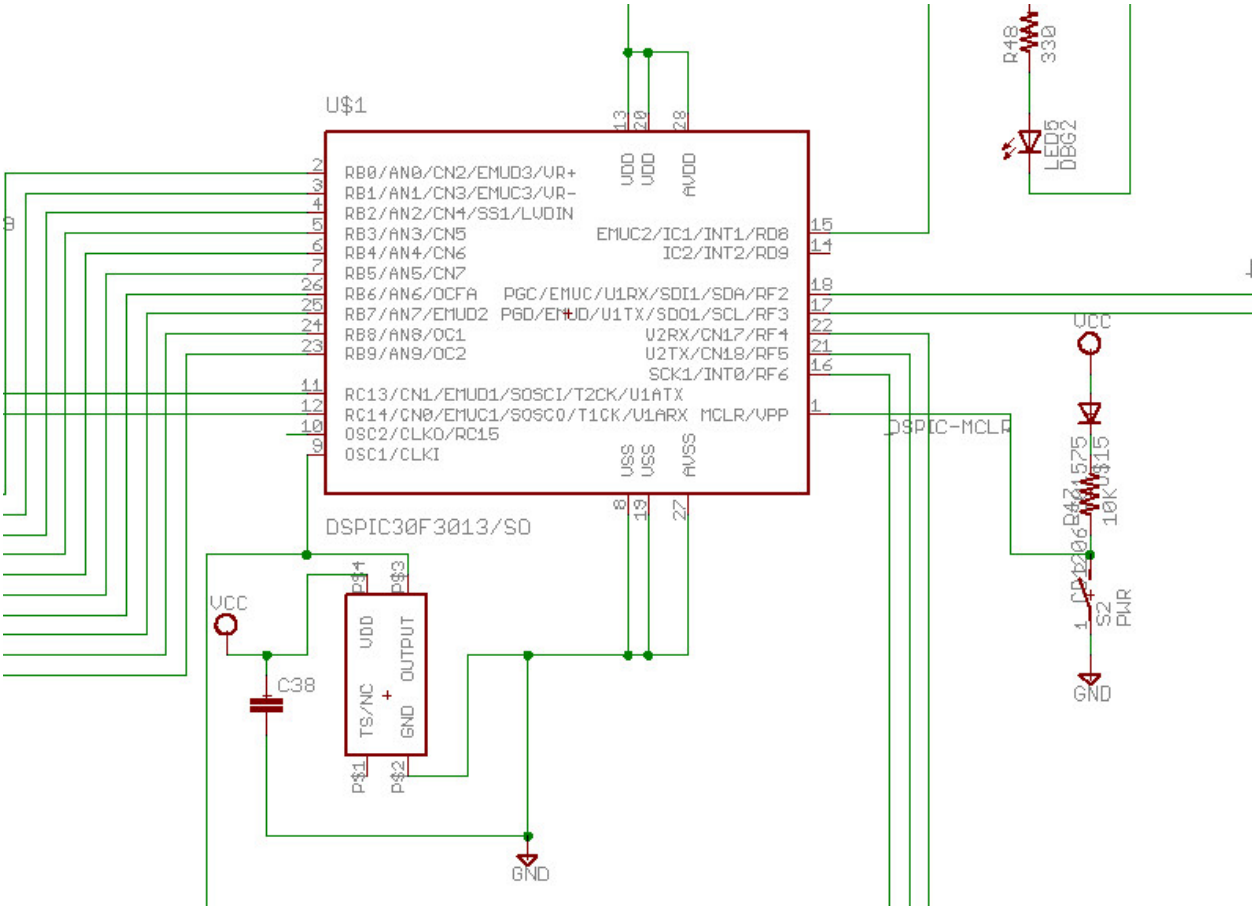


Figure 59: dsPIC configuration

C.4 USB-PIC Schematic

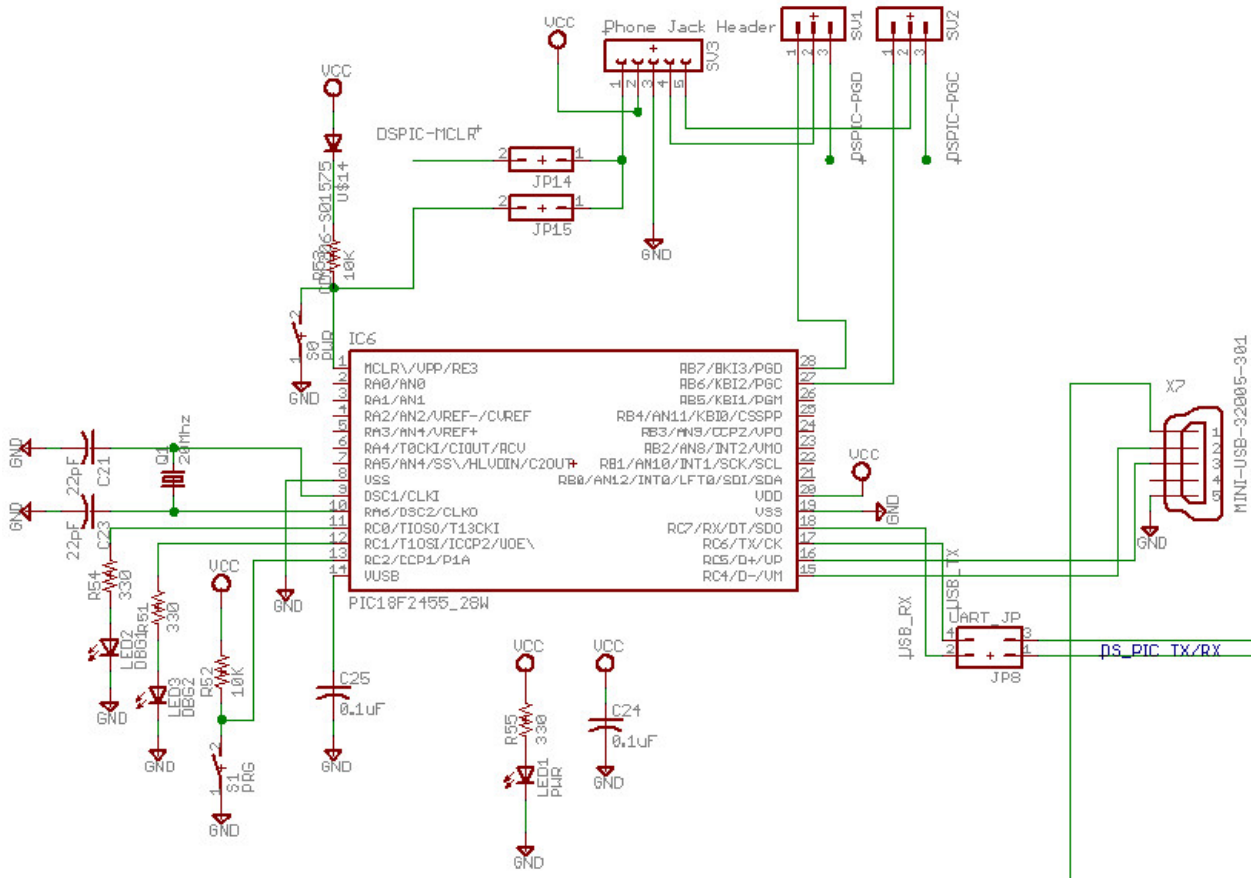


Figure 60: USB-PIC, PC programming, and USB connection

C.5 Receiver PCB

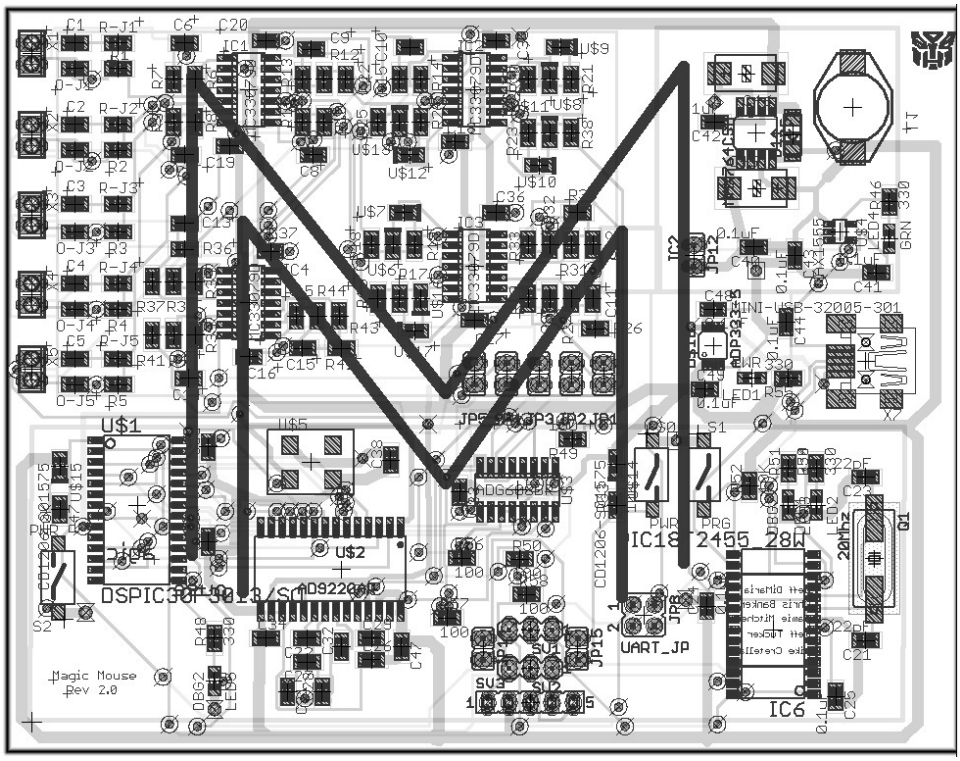


Figure 61: Analog Signal Processing PCB Layout

D Parts List

Part	Supplier	Part No.	Qty.	1	1000	Total (Unit)	Total (1000)
Analog Front End Hardware							
Diodes	Mouser	CD1206-S01575	10	0.06	0.03	0.6	0.3
Op Amps	Mouser	MC33079DG	4	1.03	0.535	4.12	2.14
Resistor (1k Ohm)	Mouser	CRCW08051K00JNEA	35	0.02	0.016	0.7	0.56
Resistor (560k Ohm)	Mouser	260-560K-RC	10	0.04	0.016	0.4	0.16
Capacitor 1.5 nF)	Mouser	GRM216R71H152KA01D	10	0.1	0.014	1	0.14
Capacitor(0.1 uF)	Mouser	08055C104KAT2A	20	0.17	0.1	3.4	2
Receiver Transucer	Mouser	255-400SR12-ROX	5	5.24	3.53	26.2	17.65
Power Systems							
Charger IC	Maxim IC	MAX1555	1	1.34	0.85	1.34	0.85
Battery	PowerStream Tech.	Li-Poly	1	10	5	10	5
Transmitter Voltage Regulator	Texas Instruments	TPS77030DBVR	1	0.93	0.34	0.93	0.34
Power (-5 v) Inverter IC	Maxim	MAX764	1	2.49	2.38	2.49	2.38
Inductor (47 uH)	Coilcraft	D03316P-473MLB	1	1.07	0.58	1.07	0.58
Capacitor (100 uF)	Mouser	TPSY107M010R0100	1	3.69	2.46	3.69	2.46
Capacitor (68 uF)	Mouser	TPSE686M020R0150	1	1.5	0.85	1.5	0.85
Diode	Mouser	821-ES1B	1	0.11	0.08	0.11	0.08
Red LED	Mouser	597-5411-407F	1	0.5	0.219	0.5	0.219
Capacitor (0.1 uF)	Mouser	GRM21BF51C105ZA01L	3	0.11	0.04	0.33	0.12
usbPIC-PC Interface							
Crystal Osc (20 MHz)	Mouser	ABLS-20.000MHZ-B2-T	1	0.45	0.23	0.45	0.23
Resistor (1k Ohm)	Mouser	CRCW08051K00JNEA	2	0.02	0.016	0.04	0.032
Capacitor(0.1 uF)	Mouser	08055C104KAT2A	2	0.17	0.1	0.34	0.2
Capacitor (0.18 uF)	Mouser	C0805C184K4RACTU	2	0.63	0.156	1.26	0.312
Diodes	Mouser	CD1206-S01575	1	0.06	0.03	0.06	0.03
USB Connector	Mouser	56579-0576	1	1.9	0.92	1.9	0.92
Blue LED	Mouser	LTST-C170TBKT	2	0.36	0.178	0.72	0.356
dsPIC							
dsPIC	Digikey	dsPIC30F3013	1	9.15	5.48	9.15	5.48
ADC	Digikey	AD9220	1	9.26	6.86	9.26	6.86
Analog Mux	Digikey	ADG608BR	1	4.46	2.52	4.46	2.52
Oscillator	Mouser	XO57CTECNA7M3728	1	4.56	3.65	4.56	3.65
Diode	Mouser	CD1206-S01575	1	0.06	0.03	0.06	0.03
Resistor (330 Ohm)	Mouser	RK73H2ATTD3300F	4	0.08	0.016	0.32	0.064
Resistor (1k Ohm)	Mouser	CRCW08051K00JNEA	1	0.02	0.016	0.02	0.016
Capacitor(0.1 uF)	Mouser	08055C104KAT2A	7	0.17	0.1	1.19	0.7
Capacitor (10 uF)	Mouser	80-T491A104K035	10	0.18	0.08	1.8	0.8
Red LED	Mouser	597-5411-407F	1	0.5	0.219	0.5	0.219
Resistor (510 Ohm)	ECE Shop	510 Ω 1/4W Resistor	5	0.2	0.02	1	0.1
Transmitter							
Tx Transducer	Mouser	255-400ST12-ROX	1	5.24	3.53	5.24	3.53
Hex Inverter	Mouser	CD4069UBPW	1	0.26	0.144	0.26	0.144
MSP430	Texas Instruments	MSP430F2013I-PWR	1	3.71	2.1	3.71	2.1
Crystal - 40 kHz	Mouser	CFV20640.000KAZFB	1	1	0.52	1	0.52
NPN Transistor	Mouser	MMBT-3904	1	0.06	0.022	0.06	0.022
Res 47k 0603	Mouser	RK73H1JTTD4702F	1	0.08	0.014	0.08	0.014
res 10k 0603	Mouser	RK73H1JTTD1002F	3	0.08	0.014	0.24	0.042
cap .22uf 0603	Mouser	C0603C224Z4VACTU	2	0.1	0.024	0.2	0.048
Misc Hardware							
Pin Headers	ECE Shop	Conn-Hdr-M & SR	41	0.08	0.04	3.28	1.64
Shunts	Mouser	15-38-1026	20	0.2	0.16	4	3.2
Switches	Digikey	EVQ-PPFA25	3	0.9	0.449	2.7	1.347
Jumpers (0 Ohm)	Mouser	263-0-RC	10	0.05	0.014	0.5	0.14
Receiver Array	80/20		1	30	10	30	10
PCB	4pcb.com		1	66	2	66	2
Twisted-Pair Wire (20 gauge)	ECE Shop	Feet:	10	0.1	0.01	1	

Total Unit Cost (with PCB prototype) 213.74 83.095 (per 1000) Total Unit Cost (before PCB prototype) 147.74